TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Numerische Mechanik

Flexible Aggregation-based Algebraic Multigrid Methods for Contact and Flow Problems

Tobias Wiesner

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 1. September 2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 4. Februar 2015 angenommen.

Abstract

It is beyond controversy that computer simulations of physical processes get more and more important in many fields of industrial applications covering a broad range from classical engineering disciplines (e.g., automotive, aerospace) to bio-medical applications. Specifically, computer simulations can provide new insights and a better understanding of processes that one cannot obtain from classical experiments due to technical limitations or economic aspects. There is no doubt that the complexity of the mathematical models grows with the increasing capacity and availability of computer resources coming along with new challenges for the numerical solution techniques. Within contemporary numerical solution methods, internally large linear systems have to be solved using iterative techniques which heavily rely on efficient internal preconditioning methods. The principal focus of this thesis are so-called aggregation-based algebraic multigrid preconditioners which are well-established for certain classes of linear problems due to optimal performance and scaling properties. However, multigrid methods in general are expert systems such that they cannot be applied for other problem classes as a black box. In order to develop "Flexible Aggregation-based Algebraic Multigrid methods for Contact and Flow problems", a flexible multigrid framework is required that allows for problem-specific enhancements. Once the basic ideas of multigrid methods have systematically been introduced, all core components of an aggregation-based algebraic multigrid method are explained as they serve as building blocks in the flexible software framework.

Different problem classes, such as contact and flow problems, come along with very specific demanding matrix properties in the associated linear systems. The intention of this work is to illustrate ways to enable multigrid methods for new classes of problems with minimal adaptions in the algorithms. In the context of convection-dominated flow problems, this thesis explores general concepts for handling non-symmetric problems with aggregation-based algebraic multigrid methods. An important contribution of this work is a novel transfer operator smoothing strategy for non-symmetric problems based on advanced concepts such as a flexible pattern strategy combined with explicit mode constraints. The algorithm is perfectly integrated in the multigrid framework and can be used together with a Petrov–Galerkin strategy for building valid restriction operators for non-symmetric problems.

Another class of challenging problems results from contact mechanics using mortar finite elements. Both, a condensed and a saddle point formulation are considered in detail for developing adapted aggregation-based multigrid preconditioners. When using a condensed contact formulation one can avoid the natural saddle point structure of a contact problem (with contact constraints) and has to solve a structurally non-symmetric problem with a non-diagonally dominant system matrix. Whereas standard aggregation-based algebraic multigrid methods are failing, a cheap column permutation method combined with smart strategies to improve the robustness, such as an observer mechanism to keep track of the diagonal-dominance of the linear operators, lead to efficient multigrid preconditioners.

The saddle point formulation for the contact problems requires a different approach. Here, the focus of this work is on multigrid methods for saddle point problems, including a full multigrid approach where indefinite block smoothers perform the coupling of the primary and secondary variables of the saddle point problem on each multigrid level. Several well-known block smoothers from the literature are discussed and explored in the context of saddle point problems arising from contact problems. Additionally, a new contact-specific aggregation strategy for Lagrange multipliers is developed which seems more intuitive than existing alternatives. All the proposed methods and enhancements are seamlessly embedded in the surrounding flexible algebraic multigrid framework. The proposed enhancements both for the condensed and the saddle point formulation lead to robust and efficient algebraic multigrid preconditioners as is shown with several large numerical examples.

The intention of this thesis is to demonstrate the usage of a general flexible algebraic multigrid framework to develop problem-specific enhancements for new demanding problem classes using concrete examples arising from problems in computational contact mechanics and computational fluid mechanics, stating that all presented concepts and methods in this thesis are more general and allow for further extensions to more complex multiphysics applications.

Zusammenfassung

Es ist unbestritten, dass Computersimulationen von physikalischen Prozessen in vielen Bereichen industrieller Anwendungen, begonnen bei klassischen Ingenieursdisziplinen (Automobilindustrie, Luft- und Raumfahrtindustrie) bis hin zu biomedizinischen Anwendungen, an Bedeutung gewinnen. Computersimulationen können insbesonders neue Einsichten und ein tieferes Verständnis gewähren, die sich aus klassischen Experimenten wegen technischer Einschränkungen oder aus wirtschaftlichen Gesichtspunkten nicht gewinnen lassen. Es besteht kein Zweifel, dass die Komplexität der mathematischen Modelle mit der wachsenden Verfügbarkeit von Rechenkapazität zunimmt, welche mit neuartigen Herausforderungen für die numerischen Lösungsverfahren einhergeht. In zeitgemäßen numerischen Methoden sind intern sehr große lineare Gleichungssysteme mittels iterativer Verfahren zu lösen, welche stark auf effiziente, interne Vorkonditionierungsverfahren angewiesen sind.

Das Hauptaugenmerk dieser Dissertation liegt auf sogenannten aggregationsbasierten algebraischen Mehrgitterverfahren, die aufgrund von optimalen Skalierungseigenschaften für bestimmte Klassen von linearen Problemen als Vorkonditionierer etabliert sind. Mehrgitterverfahren im allgemeinen sind jedoch Expertensysteme, so dass sie nicht als „black box“ auf andere Problemklassen angewandt werden können. Für die Entwicklung „Flexibler aggregationsbasierter algebraischer Mehrgitterverfahren für Kontakt- und Strömungsprobleme“ benötigt man ein flexibles Framework das problemspezifische Erweiterungen erlaubt. Nachdem die grundlegenden Ideen von Mehrgitterverfahren methodisch eingeführt worden sind, werden alle Grundkomponenten für aggregationsbasierte algebraische Mehrgitterverfahren erklärt, da sie die Bausteine in einem flexiblen Software Framework darstellen.

Unterschiedliche Problemklassen, wie Kontakt- und Strömungsprobleme, bringen besondere und durchaus anspruchsvolle Matrixeigenschaften in den zugehörigen linearen Gleichungssystem mit sich. Das Ziel dieser Arbeit ist es Wege aufzuzeigen mit minimalen Anpassungen Mehrgitterverfahren für neue Problemklassen zu öffnen. Im Bereich konvektionsdominierter Strömungen untersucht diese Arbeit allgemeine Konzepte für die Handhabung unsymmetrischer Probleme mit aggregationsbasierten algebraischen Mehrgittermethoden. Ein wichtiger Beitrag dieser Arbeit ist mit einer neuartigen Glättungsstrategie für Transferoperatoren im Falle von unsymmetrischen Problemen gegeben, welche auf fortschrittlichen Konzepten, wie einer flexiblen „Pattern“-Strategie (für die Belegungsstruktur der Matrizen) kombiniert mit zusätzlichen Nebenbedingungen (für Fehelermoden), basiert. Der Algorithmus ist perfekt in das Mehrgitterframework integriert und kann mit einer Petrov–Galerkin Methode für zulässige Restriktionsoperatoren für nicht-symmetrische Probleme verwendet werden.

Eine andere Klasse herausfordernder Probleme ergibt sich aus dem Bereich der Kontaktmechanik mit sogenannten Mortar-basierten Finite-Element-Methoden. Sowohl eine kondensierte Formulierung als auch eine Formulierung als Sattelpunktproblem werden für die Entwicklung angepasster aggregationsbasierter Mehrgittervorkoniditionierer im Detail betrachtet. Mit der kondensierten Formulierung lässt sich die natürliche Sattelpunktstruktur eines Kontaktproblems (mit Kontaktnebenbedingungen) vermeiden und man hat ein strukturell unsymmetrisches Problem mit einer nicht diagonaldominanten Systemmatrix zu lösen. Während Standardverfahren im Bereich aggregationsbasierter Mehrgittermethoden scheitern, führt eine günstige Spaltenvertauschung kombiniert mit intelligenten Strategien zur Verbesserung der Robustheit, wie einem Kontrollmechanismus zur Überwachung der Diagonaldominanz der linearen Operatoren, zu effizienten Mehrgittervorkonditionierern.

Für die Kontaktprobleme in Sattelpunktformulierung ist ein anderer Ansatz erforderlich. In diesem Zusammenhang konzentriert sich dieser Teil der Arbeit auf Mehrgitterverfahren für Sattelpunktprobleme. Dies schließt einen Mehrgitteransatz ein, in welchem Blockverfahren zur Glättung verwendet werden, welche die Kopplung von Primär- und Sekundärvariablen des Sattelpunktproblems auf allen Gitterebenen übernehmen. Verschiedene wohlbekannte Blockmethoden zur Glättung aus der Literatur werden diskutiert und im Zusammenhang von Sattelpunktproblemen für Probleme aus der Kontaktmechanik untersucht. Zusätzlich wird eine neue kontaktspezifische Aggregationsstrategie für Lagrange Multiplikatoren entwickelt welche eingängiger erscheint als existierende Alternativen. Alle vorgestellten Verfahren und Erweiterungen sind nahtlos in das umgebende Framework für Algebraische Mehrgitterverfahren eingebettet. Wie mit mehreren großen numerischen Beispielen gezeigt wird, liefern die vorgeschlagenen Anpassungen sowohl für den kondensierten Fall wie auch für die Sattelpunktformulierung jeweils robuste und effiziente Algebraische Mehrgitterverfahren zur Vorkonditionierung.

Ziel dieser Arbeit ist es aufzuzeigen wie ein allgemeines und flexibles Framework für Algebraische Mehrgitterverfahren verwendet werden kann, um problemspezifische Anpassungen für neue anspruchsvolle Problemklassen zu entwickeln. Hierfür werden konkrete numerische Beispiele aus dem Bereich der rechnergestützten Kontaktmechanik und Fluidmechanik verwendet, wobei angemerkt werden soll, dass alle in dieser Arbeit vorgestellten Konzepte und Methoden allgemein gehalten sind und so eine Erweiterung zu noch komplexeren physikalischen Anwendungen erlauben.

Contents

Nomenclature

Superscripts and Subscripts

$(\cdot)^{(\mathcal{N}_i)}, (\cdot)_{\mathcal{N}_i}$	Subdomain
$(\cdot)^{(\mathcal{M})}$	Master
$(\cdot)^{(\mathcal{S})}$	Slave
$(\cdot)_{\mathcal{A}}$	Active slave node part
$(\cdot)_{\mathcal{I}}$	Inactive slave node part
$(\cdot)_{\mathsf{c}}$	coarse level part
$(\cdot)_{\mathsf{f}}$	fine level part
$(\cdot)_{\mathsf{D}}$	Indices associated with Dirichlet boundary conditions
$(\cdot)_{\mathsf{c}}$	Contact
$(\cdot)^{\mathrm{h}}$	Discrete form with meshsize h
$(\cdot)^{k}$	k-th iterate
$(\cdot)_{\ell}$	Level ℓ
A^{T}	Transpose of matrix A
$(\cdot)^{\boldsymbol{u}}$	Variable associated with displacement degrees of freedom $\boldsymbol{u}$
$(\cdot)^{\boldsymbol{\lambda}}$	Variable associated with Lagrange multipliers $\boldsymbol{\lambda}$
$(\cdot,\cdot)_0$	Standard scalar product in L^2
$(\cdot,\cdot)_e$	Euclidean scalar product in $\mathbb{R}^d$
$(\cdot,\cdot)_{\mathrm{A}}$	Energy scalar product with $(u,v)_{\mathrm{A}} := (\mathrm{A}u,v)_e$
$(\cdot,\cdot)_D$	Diagonally scaled inner product with $(u,v)_D := (Du,v)_e$
$\|\cdot\|_0$	L^2 norm associated with standard scalar product in L^2
$\|\cdot\|_{1,\Omega}$	H^1 norm on domain Ω
$\|\cdot\|_e$	Euclidean norm in $\mathbb{R}^d$
$\|\cdot\|_{\mathrm{A}}$	Energy norm with $\|u\|_A := \sqrt{(u,u)_{\mathrm{A}}}$
$\|\cdot\|_D$	Diagonally scaled norm with $\|u\|_D := \sqrt{(u,u)_D}$
$\|\mathrm{A}\|_2$	Spectral norm of matrix A defined by $\|A\|_2 := \max\left\{\frac{(Ax,y)_e}{\|x\|_e\|y\|_e}, 0 \neq x,y \in \mathbb{R}^d\right\}$

Linear system

A	Linear operator
b	Right hand side
n	Number of degrees of freedom

r	Residual or defect
e	Error of exact solution and approximate solution
x	Solution vector
$\widehat{x}$	Exact solution of fine level problem
$\rho(\mathrm{A})$	Spectral radius of matrix A
D	Diagonal part of A
E	Strictly lower triangle part of A
F	Strictly upper triangle part of A
W	Preconditioning matrix $W \approx \mathrm{A}$

Multigrid specific operators

γ	Multigrid cycle parameter ($\gamma = 1$: V-cycle, $\gamma = 2$: W-cycle)
I	Identity matrix
ℓ_{max}	Maximum level index denoting maximum number of multigrid levels
M	Iteration Matrix
M_C	Coarse grid correction matrix
M_{MG}	Iteration matrix of multi level method
$M_{\mathscr{S}}$	Iteration matrix of level smoothing method $\mathscr{S}$
M_T	Iteration matrix of two level method
ν	Number of level smoothing sweeps
ν_1	Number of pre-smoothing sweeps
ν_2	Number of post-smoothing sweeps
ω	Damping parameter in relaxation-based smoothers
P	Prolongation operator
R	Restriction operator
Q	Approximate A within relaxation-based methods
Φ^{CLC}	Coarse level correction method
Φ^{TLM}	Two-level method
Φ^{MLM}	Multi-level method
$\mathscr{S}$	Level smoother

Aggregation-based Multigrid

$\mathscr{A}_\ell$	Aggregates on multigrid level ℓ
$B^{(\mathrm{r})}$	Left (near) null space vectors
$B^{(\mathrm{p})}$	Right (near) null space vectors
C	Set of aggregated nodes in aggregation algorithm
$m_{\mathscr{A}_\ell}$	Number of aggregates $\#(\mathscr{A}_\ell)$ on level ℓ
m_ℓ	Number of nodes on level ℓ
n_B	Number of near null space vectors
$G(\mathrm{A})$	Graph of matrix A
N	Pattern matrix with allowed transfer operator patterns
$\mathrm{Nb}_\ell(i)$	Set of neighboring nodes around node i on level ℓ (see Section 3.3.1)
n	Mapping of degrees of freedom to corresponding nodes (see Definition 3.3.1)
$\widehat{P}$	Tentative (non-smoothed) prolongation operator
$\widehat{R}$	Tentative restriction operator ($\widehat{R} = \widehat{P}^{\mathsf{T}}$)

R	Set of non-aggregated nodes in aggregation algorithms
S	Schur complement operator

Domains and boundaries

d	Dimension of problem domain ($d \in \{2, 3\}$)
$\partial\Omega_0$	Boundary of Ω_0 in reference configuration
$\partial\Omega_t$	Boundary of Ω_t in current configuration
γ_c	Potential contact part of domain boundary $\partial\Omega_t$ in current configuration
γ_D	Dirichlet part of domain boundary $\partial\Omega_t$ in current configuration
γ_N	Neumann part of domain boundary $\partial\Omega_t$ in current configuration
Γ_c	Potential contact part of domain boundary $\partial\Omega_0$ in reference configuration
Γ_D	Dirichlet part of domain boundary $\partial\Omega_0$ in reference configuration
Γ_N	Neumann part of domain boundary $\partial\Omega_0$ in reference configuration
Ω_0	Domain in reference (material) configuration
Ω_t	Domain in current (spatial) configuration

Governing equations of solid mechanics

$\boldsymbol{f}_0$	Body force in reference configuration
$\boldsymbol{f}$	Body force in current configuration
$\boldsymbol{N}, \boldsymbol{n}$	Unit normal vector in reference and current configuration
Φ_t	Mapping between reference and current configuration (see Section 5.1)
ρ_0, ρ	Reference and current mass density
$\mathbf{r}^{\boldsymbol{u}}$	Residual vector associated with displacement degrees of freedom $\boldsymbol{u}$
$\mathbf{r}^{\boldsymbol{\lambda}}$	Residual vector associated with Lagrange multipliers $\boldsymbol{\lambda}$
t	Time
T	Total simulation time
$\widehat{\boldsymbol{t}}_0$	Prescribed pseudo-traction in reference configuration
$\widehat{\boldsymbol{t}}$	Prescribed traction in current configuration
$\boldsymbol{u}$	Displacement
$\dot{\boldsymbol{u}}$	Velocity
$\ddot{\boldsymbol{u}}$	Acceleration
$\widehat{\boldsymbol{u}}_0$	Initial displacement at time $t = 0$
$\widehat{\dot{\boldsymbol{u}}}_0$	Initial velocity at time $t = 0$
$\boldsymbol{x}$	Position in current configuration
$\boldsymbol{X}$	Position in reference configuration

Contact mechanics

$\mathcal{A}$	Node set for active slave contact nodes
$\mathfrak{F}$	Coefficient of friction
g_n	Gap function
$\mathbf{g}$	Vector with discrete weighted gap information at the contact interface
$\mathcal{I}$	Node set for inactive slave contact nodes
$\boldsymbol{\lambda}$	Lagrange multiplier vector
$\mathcal{M}$	Node set for master contact nodes
p_n	Normal contact traction (pressure) in current configuration

$\mathcal{S}$	Node set for slave contact nodes
$\boldsymbol{v}_{\tau,\mathsf{rel}}$	Relative tangential velocity vector

Stresses and constitutive laws

E	Young's modulus
$\boldsymbol{P}$	First Piola–Kirchhoff stress tensor

Function spaces & Variational formulation

$m(\cdot,\cdot), a(\cdot,\cdot)$	(Bilinear) forms for variational formulation (see (5.22) and (5.23))
$f(\cdot)$	Linear form in variational formulation (see (5.24))
$\boldsymbol{\mathcal{N}}$	Trace space of $\boldsymbol{\mathcal{V}}^{(\mathcal{N}_2)}$
$\boldsymbol{\mathcal{M}}$	Dual space of $\boldsymbol{\mathcal{N}}$ defining the function space for the Lagrange multipliers
$\boldsymbol{\mathcal{U}}$	Solution space
$\boldsymbol{\mathcal{V}}$	Space for weighting functions

Discrete formulation

h	Mesh size parameter
$\boldsymbol{\mathcal{M}}^{\mathrm{h}}$	Discrete space for the Lagrange multipliers
ϕ_p	The p-th basis function for Lagrange multipliers $\boldsymbol{\lambda}$ associated with node p
ψ_p	Standard nodal basis for displacements $\boldsymbol{u}$ associated with node p
$\mathcal{S}_1$	Finite element space (linear elements) associated with the mesh $\mathcal{T}_{\mathrm{h},\Omega}$
$\mathcal{T}_{\mathrm{h},\Omega}$	Triangulation on domain Ω with maximum meshsize h
$\boldsymbol{\mathcal{U}}^{\mathrm{h}}$	Discrete solution space
$\boldsymbol{\mathcal{V}}^{\mathrm{h}}$	Discrete space for weighting functions
W	Algebraic representation of projection operator (see (6.6))

Abbreviations

AMG	Algebraic multigrid
AMLI	Algebraic multilevel iteration
CG	Conjugate gradient
DBC	Dirichlet boundary condition
DOF	Degree of freedom
FEM	Finite element method
GMRES	Generalized minimal residual
GS	Gauss-Seidel method (see page 10)
ILU	Incomplete lower upper triangular matrix
KKT	Karush–Kuhn–Tucker
LBB	Ladyzhenskaya–Babuška–Brezzi
NCP	Nonlinear complementarity (see Section 5.5.1)
NTS	Node-to-segment contact approach
PDE	Partial differential equation
SGS	Symmetric Gauss-Seidel method (see page 11)
SIMPLE	Semi-implicit method for pressure-linked equations
SOR	Successive overrelaxation method (see page 11)
STS	Segment-to-segment contact approach

CHAPTER 1

Introduction

I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail

("Law of the hammer", Abraham H. Maslow [128])

This quotation, known as Maslow's "Law of the hammer", describes the problem of the so-called method-centered approach where the problems are adapted in such a way that existing tools can be applied. In Maslow [128] the author explains with simple examples how the science in general may suffer from this way of thinking.

In this thesis a problem-centered approach is pursued instead of a method-centered approach. In fact, often there already exist efficient tools that are designed for slightly different problems and it is the most natural thing trying to adapt them to make them work for the new classes of problems. In other words: in a problem-centered approach one tries to adapt existing tools to solve the given problems instead of changing or simplifying the problems to meet the prerequisites of the tools (or solvers). This way the problem-centered approach may even stimulate new innovative concepts for challenging problems.

Typical problems arising from computational fluid mechanics and computational contact mechanics are known to be particularly challenging for iterative (linear) solvers for different reasons. With both contact and flow problems in mind, this thesis presents several ways to adapt the so-called *aggregation-based algebraic multigrid methods* (AMG) as tools for the efficient solution of the corresponding demanding linear systems.

1.1. Motivation

The principal focus of this thesis is on aggregation-based AMG preconditioners, that play an important role for efficiently solving large linear systems. For the motivation one has to answer two questions: why are efficient multigrid preconditioners for iterative solvers of general interest? Why could this thesis be interesting if one can buy ready-to-use simulation software with different linear solvers and multigrid preconditioners included?

To answer the first question one should briefly look at the typical steps required to perform numerical simulations:

Modeling phase: In the modeling phase the physical problem is described in a mathematical language usually resulting in a set of partial differential equations (PDEs).

Discretization phase: Then, after a proper reformulation of the problem, the continuous mathematical model has to be discretized leading to a finite-dimensional problem that can be solved on a computer with limited resources. In this thesis the *finite element method (FEM)* is used for the spatial discretization of the continuous problem. For non-steady or transient problems with evolving time, one needs a time integration or time discretization scheme with discrete time steps.

Solution phase: Depending on the discrete models, one usually ends up solving a general nonlinear discrete problem in each time step. There are many different solution strategies for certain classes of nonlinear problems known with different prerequisites and properties. Just to mention a few: there are classical Newton methods and variants such as inexact Newton methods, Gauss–Newton methods, and Levenberg–Marquardt methods (cf. Marquardt [127], Moré [134]) which interpolate between Gauss–Newton methods and gradient descent methods. A general overview on nonlinear solution strategies is provided, e.g., by Fletcher [67]. In this thesis Newton-like methods, which internally rely on the solution of large linear systems, are used for solving the nonlinear problems. The challenging part here is to obtain a sufficiently good approximate solution within a reasonable time.

Illustrating the typical algorithmic layout of the solution phase for a non-steady problem, Figure 1.1 reveals that the preconditioning methods are the core component in the overall solution process. Therefore, it is clear that the design of efficient iterative solution techniques for large linear systems is of high interest. It is beyond controversy that computer simulations of physical processes get more and more important in many fields of industrial applications. Undoubtedly, today's computer models grow with the increasing capacity and availability of computer resources. Internally, large linear systems occur which can only be solved using iterative techniques, such that the performance of iterative solvers heavily relies on efficient preconditioning methods for large linear systems. Algebraic multigrid preconditioners are among the most versatile and efficient state-of-the-art preconditioning methods that are currently known for gaining optimal performance and scaling properties for certain problem classes (e.g. Trottenberg et al. [186]). To answer the second question, why this thesis could be interesting if ready-to-use solvers and preconditioners already exist, one has to understand that iterative solvers and multigrid methods are expert systems, which develop their full potential only if they are correctly used. The patient reader may find a more detailed answer in the next sections where a rough overview of state-of-the-art iterative solving and preconditioning ideas is given before the exact research objectives for this thesis are specified.

1.2. Solution strategies for linear systems

The problems considered in this thesis can be written in their most general form as

$$\mathrm{A}x = b \tag{1.1}$$

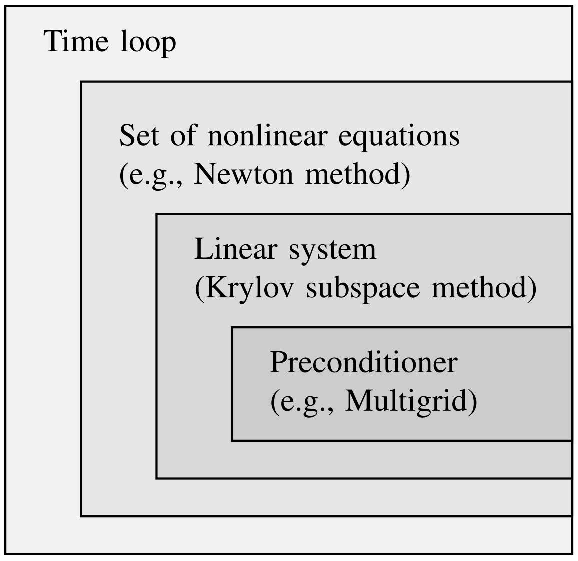


Figure 1.1.: Nested algorithmic layout of implicit time integration scheme.

with $\mathrm{A} \in \mathbb{R}^{n \times n}$ a sparse matrix and $x, b \in \mathbb{R}^n$. Keeping in mind that the linear systems in (1.1) usually arise from linearizations of a nonlinear system within a (multiphysics) simulation of different physical processes, it is clear that one has to deal with different classes of linear systems that are governed by the mathematical properties of the corresponding linear operator A (such as symmetry or the eigenvalue spectrum).

1.2.1. Direct solvers

Direct solving strategies for solving sparse linear systems have the advantage that they often can be used as a black-box method. Internally, they are based on some kind of decomposition, which can be, e.g., a QR-decomposition or a LU-decomposition. The interested reader is referred to Duff et al. [56] for a general overview of the underlying techniques.

In contrast to direct solvers, iterative techniques cannot solve arbitrary linear systems of regular equations, such that – to this day – there are still applications which are dominated by direct sparse solvers. But contemporary iterative solvers are an interesting alternative to direct solvers for linear problems resulting from finite element formulations in computational fluid mechanics or computational contact mechanics. Especially due to the extreme memory requirements of direct sparse solvers with scaling properties often being far away from the perfect scaling $\mathcal{O}(n)$ for an increasing problem size n, direct solvers are not an option for large scale linear problems from the applications considered in this thesis. It shall be noted that direct sparse solvers are furthermore more complex to implement in parallel than iterative techniques. So, with the recent developments in parallelization, iterative solution techniques are much more favorable.

1.2.2. Iterative solution techniques for linear systems

This section gives a brief introduction of iterative linear solvers and the idea of so-called preconditioning methods needed to solve linear systems as given in (1.1). Iterative solution techniques try to solve a linear system approximately by iteratively improving an initial guess x^0 until the solution vector x^k is sufficiently close to the exact (but unknown) solution x of (1.1). In Saad and van der Vorst [171] an overview of contemporary iterative solution processes is given with a short review of the history of iterative solvers and their development in the 20-th century.

In contrast to direct solvers (cf. Section 1.2.1) which can be understood as perfect black-box methods, iterative solution methods are expert systems which require a sufficient in-depth understanding of the underlying principles before they can be successfully applied to concrete problems. This makes them hard to use for general industrial applications, but the correct usage of these iterative solving strategies allows for solving large problems which are unattainable for direct methods in a reasonably low time. In this thesis, well-established iterative linear solving strategies are used such as Krylov-subspace methods represented by the "conjugate gradient" (CG) method (cf. Hestenes and Stiefel [92]) for symmetric problems, or the "generalized minimal residual" (GMRES) method (cf. Saad and Schultz [168]) as well as the "biconjugate gradient stabilized" (BiCGstab) method (cf. van der Vorst [188]) for non-symmetric problems. For an overview on Krylov-subspace methods, the interested reader may refer to, e.g., Liesen and Strakos [119]. In particular, the iterative Krylov subspace method used in this thesis is the preconditioned GMRES method which is known to be a robust iterative method that also can solve non-symmetric linear systems. In Appendix A the interested reader can find a brief discussion of the underlying principles and the algorithm of the preconditioned GMRES method.

Independent of the used iterative linear solver, one can only obtain optimal performance when combining the iterative solver with a suitable preconditioning method.

1.2.3. The Idea of preconditioning

The speed of convergence of iterative solvers mainly depends on the spectral properties of the given linear operator A (cf. Hackbusch [86], Saad [170]). In practice, the usage of additional preconditioning methods is recommended to improve the performance of the iterative solving process.

The main idea of preconditioning is to construct a cheap and easy-to-invert approximation W of A, such that all eigenvalues of the preconditioned operator $\mathrm{A}W^{-1}$ are close to one. Then, instead of the original linear system (1.1) one solves the preconditioned linear system that is given by

$$\mathrm{A}W^{-1}\widetilde{x} = b \quad \text{with} \quad Wx = \widetilde{x}. \tag{1.2}$$

If W is a sufficient approximation of A, iterative methods should be able to solve the preconditioned linear system (1.2) within a significantly smaller number of linear iterations than the original system (1.1).

Remark 1.2.1 (Left- versus right-preconditioning). The preconditioning strategy given in (1.2) is known as right-preconditioning in the literature, since the preconditioning matrix W is applied from the right. As alternative one can also use left-preconditioning, i.e., $W^{-1}\mathrm{A}x = W^{-1}b$, or combine left- and right-preconditioning to a two-sided preconditioning strategy resulting in $W_1^{-1}\mathrm{A}W_2^{-1}\widetilde{x} = W_1^{-1}b$ with $W_2x = \widetilde{x}$.

In this thesis only right-preconditioning is used. Right-preconditioning preserves the original right-hand side vector b. Consequently, the residual norm is relative to the initial system, i.e., $b - \mathrm{A}x$, since the algorithm obtains the residual $b - \mathrm{A}x = b - \mathrm{A}W^{-1}\widetilde{x}$, implicitly. This allows the formulation of stopping criteria for the linear solver based on the original non-preconditioned residual.

There are quite a few different preconditioning techniques for iterative solvers which are based on different principles. The reader is referred to Saad and van der Vorst [171] or Benzi [21] for a general overview of the most important classes of preconditioners with plenty of comments on history and literature for further reading. Just to give some examples: the *incomplete factorization* methods exist in many variants and are all based on the idea to control the fill-in in a LU-decomposition. Another class of preconditioners is given by the *sparse approximate inverse* methods which try to approximate the inverse of A iteratively. Many other iterative processes, such as relaxation-based methods, can also be used as preconditioners. Relaxation-based methods also play an important role in context of the so-called *multigrid* methods which certainly represent one of the most important classes of preconditioning methods. In this thesis, the focus is on the design of special variants of multigrid preconditioners for different applications in computational fluid mechanics and computational contact mechanics.

1.3. Basic idea of multigrid methods

Multigrid methods are comprehensively discussed in the next chapters of this thesis. In this section only a short explanation of the fundamental idea of multigrid methods is given.

Multigrid methods are based on the idea of reconstructing the fine level solution x in (1.1) using information from some coarse representations of the fine level problem. One makes heavily use of the fact that well-known and cheap iterative smoothing processes are more effective in reducing high-oscillatory error components, whereas they are not able to tackle low-frequency errors. By using different resolutions of the fine level problem one can effectively reduce the relative high-oscillating error components on the corresponding multigrid levels with cheap error smoothing methods. The iterative solving process for the fine level solution x has significant benefit from the corresponding corrections of the coarser levels. Therefore, multigrid can be understood as applying cheap smoothing methods for reducing different error components on different multigrid levels. This effect is known as “smoothing” effect (see Section 2.2).

Even though multigrid methods can be used as standalone solvers, here they are used as preconditioners within the GMRES method (cf. Appendix A) to increase the robustness of the overall solution process. So, as an essential part of the iterative linear solving strategy (see Section 1.2.2), multigrid preconditioners can be considered to be a tool for efficiently solving linear systems.

As already mentioned in Section 1.1, multigrid methods are expert systems which consist of different components that have to properly work together. Concrete parameter choices for the multigrid preconditioner can make the difference between optimal convergence or divergence of the iterative solver, such that there is no “general tool” in the form of a preconditioner for all classes of linear problems. In particular, there is no general efficient iterative solver with appropriate (multigrid) preconditioners that one can buy and expect to work properly for challenging systems just by “pressing a button”. Hence, when developing “Flexible Aggregation-based Algebraic Multigrid methods for Contact and Flow problems”, it is important to accept that it is not possible to provide one multigrid method for all problem types in a black-box sense. Instead, one needs a flexible multigrid framework with different algorithmic building blocks that allows to construct appropriate multigrid methods for many classes of problems (e.g., resulting from computational fluid mechanics or computational contact mechanics). This complies with our general way of thinking of a problem-centered approach in the sense of adapting our tools and methods for our problems.

1.4. Research objective

Aggregation-based AMG methods are known for their useful properties such as optimal convergence for certain classes of problems or the fact that they can be applied to problems with unstructured meshes without requiring the user to provide coarse meshes. However, aggregation-based AMG methods (as well as multigrid methods in general) are expert systems that often have to be adapted to work for problem classes other than they have originally been developed for. What is missing is a flexible AMG framework which allows for quick extensions for specific problems that can also be used by non-expert users.

1.4.1. Specification of requirements

The central requirements for a multigrid framework are

Flexibility: Flexibility is important for several reasons: First of all, one needs a *flexible framework* that allows building multigrid hierarchies using different algorithmic building blocks that are designed for the problem-specific needs. Especially for multiphysics simulations with different physical and mathematical fields, it is necessary to design problem-specific block preconditioners which may use multigrid ideas in different ways (cf. Keyes et al. [103]). The idea is to reuse common building blocks and only replace parts of the overall algorithm in a flexible way where it is necessary. So, modularity is required in a flexible framework allowing to reuse and recombine different building blocks which makes it an ideal tool for doing research on multigrid concepts.

Flexibility is not only important for the overall framework, but also plays an important role for the different building blocks of the framework itself. In a flexible framework one needs *flexible algorithms*, which allow for problem-specific adaptions, since they are essential as building blocks in the AMG framework. In that sense, flexibility is by far more than just a sufficient number of user parameters that allow the user to control the major algorithmic behavior.

Flexibility is also a desired property in context of the software design. Today, the algorithms have to run on a multitude of different hardware platforms and software environments. So, a *flexible software design* is necessary which allows the multigrid methods to be used on high-performance clusters as well as on usual desktop workstations. Finally, a *flexible software interface* has to make sure that the AMG framework can easily be used with other software packages.

Usability: Many people are dreaming of black-box solvers and preconditioners. But as a matter of fact, one can always find a problem where such black-box methods will fail. Since one can hardly avoid problem-specific solver parameters at all, the challenge is to keep the number of essential parameters small and easy to understand such that also a user with only average knowledge about iterative linear solvers can find a proper set of parameters to use the multigrid methods. Of course, one can try to hide some of the complexity from the average user. However, as already mentioned in Section 1.1, one has to accept that multigrid methods in general and aggregation-based AMG methods in particular are expert systems with a multitude of parameter choices. Therefore, *robust methods* which can forgive bad parameter choices up to a certain extent are of high interest. Nevertheless, it is important to have a sufficient understanding of the used methods, since even with robust methods one can never fully compensate for the potential misuse of the tools caused by ignorance of the user.

1.4.2. Proposal for a flexible aggregation-based AMG framework

This work utilizes a new flexible AMG framework, called MUELU (see Prokopenko et al. [161]), which has been developed in context of this work in a close collaboration with the TRILINOS project (cf. Heroux and Willenbring [89], Heroux et al. [90]) conducted by Sandia National Laboratories. MUELU provides a highly flexible modern software framework addressing all requirements of a modern multigrid framework with respect to extensibility, usability and flexibility, such that it is prepared for the next generation of applications. To the author's knowledge it is the first modular and object-oriented aggregation-based AMG code fully written in C++ which has support for a wide variety of state-of-the-art transfer operators for both symmetric and non-symmetric problems.

The scientific contributions of this work are:

- the development of novel smoothed aggregation transfer operators for non-symmetric problems resulting from examples with convective character (cf. Chapter 4).
- the design of flexible AMG preconditioners for contact problems in condensed formulation (cf. Chapter 6).
- the implementation of a robust interface aggregation routine for Lagrange multipliers arising from contact problems in saddle point formulation (cf. Chapter 7).
- successful application of the above AMG methods to large structural contact examples in condensed and saddle point contact formulation with more than one million degrees of freedom.

In summary, the extensions and adaptions of existing multigrid methods proposed in this thesis represent an important step forward towards more robust large-scale simulations for problems in computational contact mechanics and computational fluid mechanics.

All models and problems described in this work (e.g., the structural contact problems based on mortar finite element methods as described in Chapter 5) as well as the application-specific non-standard enhancements of the multigrid methods are implemented in the in-house finite element software package BACI (cf. Wall and Gee [208]), developed at the Institute for Computational Mechanics and the Mechanics & High Performance Computing Group at Technische Universität München. This multi-purpose parallel research code is written in C++ and builds on top of the TRILINOS libraries. The author of this thesis is an active member of the core developer team of the new software package MUELU (cf. Gaidamour et al. [69]) which is publicly available as part of the TRILINOS project (cf. Heroux and Willenbring [89], Heroux et al. [90]). Thus, the multigrid-specific methods have been implemented directly in MUELU.

1.5. Outline

The remainder of this thesis is organized as follows:

Chapter 2 is devoted to a brief introduction to the general ideas of multigrid methods. Both the concepts of level smoothing and coarse-level correction are reviewed. Before introducing the multigrid algorithm some special focus is put on the prerequisites of the level smoothers for the multigrid method. Finally, some basic results from the multigrid convergence theory are presented. In this thesis multigrid methods are used as preconditioners within a preconditioned GMRES method. The reader may refer to **Appendix A** for a description of the GMRES method.

Chapter 3 comprises all details on the setup algorithm of an aggregation-based AMG method. The different phases of the aggregation methods are described in detail as well as the steps to generate non-smoothed and smoothed aggregation transfer operators. A special emphasis is placed on the algorithmic design of the different building blocks within the setup routine which define the highly flexible multigrid framework.

Chapter 4 extends the aforementioned methods to non-symmetric problems resulting from examples with convection. In order to apply aggregation-based AMG methods to non-symmetric problems one has to replace the standard Galerkin approach introduced in Chapter 2 by a Petrov–Galerkin approach that is combined with an appropriate non-symmetric transfer operator smoothing strategy. A novel flexible transfer operator smoothing strategy is presented that is motivated by ideas from ideal Schur complement based transfer operators. **Appendix B** is closely connected to this part of the thesis, where the effect of transfer operator smoothing is studied for simple 1D problems in order to gain a better understanding of the underlying smoothing principles. **Appendix C** presents some conceptual ideas of the handling of Dirichlet boundary conditions within multigrid methods which often seems to be an under-estimated topic.

In **Chapter 5**, the relevant governing equations of nonlinear solid mechanics and contact mechanics are outlined. In addition, the basic concepts of weak formulations, finite element discretization and nonlinear solution techniques are briefly reviewed. Mortar finite element methods are explained in more detail, since they play the decisive role for the formulation of our contact problems.

In **Chapter 6** a new aggregation-based AMG method is developed for contact problems in the condensed formulation. Our intention is to demonstrate how the knowledge about the methods (tools) and the problem allows the enabling of standard aggregation-based multigrid techniques with only minor modifications for a new class of problems (such as structural contact problems in condensed formulation). With the proposed flexible multigrid method one can iteratively solve large contact problems with more than one million degrees of freedoms, which is a major step towards robust contact simulations for industrial applications.

In **Chapter 7** saddle point AMG preconditioners designed for contact problems in saddle point formulation are developed. A new interface aggregation strategy for the Lagrange multipliers, that are used to couple the structural equations and contact constraints, is introduced. With numerical examples one can see the effect of different block smoothers that are aware of the saddle point structure and demonstrate the robustness of the proposed method in this thesis.

In **Appendix D**, some comments are given on the parallelization of aggregation-based multigrid methods including some basic scaling studies of the proposed methods both for flow and contact problems.

Finally, the conclusion and outlook given in **Chapter 8** summarize the most important results and accomplishments and give some comments on possible improvements.

CHAPTER 2

Basics of Multigrid methods

This chapter gives a very general introduction to the ideas of multigrid methods and therefore builds the methodical fundament of the work. It is not meant to replace a good textbook on multigrid methods, but to explain the general concepts of multigrid in an intuitive way.

First, a brief review of typical relaxation-based iterative methods is given that are often used as *smoothing methods* within multigrid. Here, our special emphasis is put on the role of diagonal dominant input matrices for this class of smoothing methods, as diagonal dominance gets important in Chapter 6. Then, the effect of the smoothing property is demonstrated using some small examples that are supposed to provide some conceptual understanding of the underlying mathematical principles.

Next, the multigrid algorithm is introduced beginning with the concept of coarse level correction and a two-level algorithm, which is then extended to a multigrid method. Here, the difference of so-called *Geometric Multigrid (GMG)* methods and *Algebraic Multigrid (AMG)* methods is explained. Since only algebraic multigrid methods are used in this thesis, special attention is put on the definition of algebraically smooth errors. The underlying ideas are motivated and illustrated by small examples.

The rest of the chapter is devoted to the basic multigrid convergence theory. Our intention is to give a rough overview on the existing convergence theory and the most common concepts that are used within the convergence proofs in the literature with a special focus on algebraic multigrid methods.

In summary, background knowledge about the different components of multigrid methods is provided, including *level smoothing* and *coarse-level correction* and their interplay. For developing application-specific advanced multigrid methods, it is essential to understand that a multigrid method can basically be understood as a smartly ordered sequence of cheap smoothing methods. Therefore, one should carefully check whether the input matrix also fulfills the minimum requirements for these level smoothing methods.

2.1. Relaxation-based iterative methods

Relaxation methods belong to the most elementary class of iterative methods for linear systems, which is well-studied in abundant publications and books (cf. Axelsson [6], Hackbusch [86], Varga [200]). Even though they could be used to solve linear systems iteratively, they are mainly used as preconditioner within an outer iterative solver due to their slow convergence. Relaxation-based methods are known to be cheap and easy to implement and – what is most interesting from the multigrid perspective – they have a so-called smoothing property (cf. Section 2.2).

2.1.1. Definition of relaxation-based methods

Since relaxation methods play an important role as smoothers within multigrid methods, it is worth to give a brief review of relaxation-based smoothing methods.

Definition 2.1.1. Assume $Q \in \mathbb{R}^{n\times n}$ to be an invertible matrix. The prototype for common relaxation-based methods to solve the linear system (1.1) is defined by the iteration

$$x^{k+1} := x^k - Q^{-1}\big(\mathrm{A}x^k - b\big) = \big(I - Q^{-1}\mathrm{A}\big)x^k + Q^{-1}b \tag{2.1}$$

starting with some initial guess $x^0 \in \mathbb{R}^n$.

Here, $M := \big(I - Q^{-1}\mathrm{A}\big)$ is the iteration matrix of the relaxation-based method and $k \in \mathbb{N}_0$ denotes the iteration index.

The concrete choice of Q in (2.1) defines different typical iterative methods that are found in the literature. Common choices are

Richardson iteration: For the *Richardson* iteration one has to use $Q := \theta(k)I$ with $\theta \in \mathbb{R}$. One can distinguish the *stationary Richardson* iteration with fixed θ and the *instationary Richardson* iteration, where θ is allowed to vary with the iteration index k. For the case that A has only positive eigenvalues, the Richardson iteration converges, if and only if it is $0 < \theta < \frac{2}{\lambda_{\max}(\mathrm{A})}$, where $\lambda_{\max}(\mathrm{A})$ denotes the maximum eigenvalue of A (cf. Hackbusch [86, Theorem 4.4.2]).

Jacobi iteration: With the uniquely defined splitting

$$\mathrm{A} = D - E - F, \quad \text{where} \tag{2.2a}$$
$$D \ \text{ is a diagonal matrix describing the diagonal part of A,} \tag{2.2b}$$
$$E \ \text{ is a strictly lower triangular matrix of A and} \tag{2.2c}$$
$$F \ \text{ is a strictly upper triangular matrix of A,} \tag{2.2d}$$

one obtains the *Jacobi* iteration by setting $Q := D$. Often an additional damping parameter $\omega > 0$ is introduced, i.e., Q is chosen as $Q := \frac{1}{\omega}D$ for the *damped Jacobi* iteration. Thus, the damped Jacobi method has the form

$$x^{k+1} := x^k - \omega D^{-1}\big(\mathrm{A}x^k - b\big) = \big(I - \omega D^{-1}\mathrm{A}\big)x^k + \omega D^{-1}b. \tag{2.3}$$

Gauss–Seidel iteration: Based on the splitting in (2.2) the *(forward) Gauss–Seidel* iteration is defined by $Q := D - E$. Note that in contrary to the Jacobi (and the Richardson) iteration, the Gauss–Seidel iteration depends on the ordering of the indices. The Gauss–Seidel method is known to converge for positive definite matrices monotonically with respect to the energy norm (cf. Hackbusch [86, Theorem 4.4.18]). There are further variants of the Gauss–Seidel iteration: using $Q := D - F$ defines the *backward Gauss–Seidel* method. Combining the backward and forward Gauss–Seidel method gives the *symmetric Gauss–Seidel* iteration with $Q := (D - E)D^{-1}(D - F)$.

Successive overrelaxation methods (SOR): Introducing a relaxation factor ω, the SOR method is defined by $Q := \omega(D - \omega E)^{-1}$ with the splitting as defined in (2.2). Obviously, the SOR method coincides with the Gauss–Seidel method if $\omega = 1$ is chosen. For $0 < \omega < 1$, the SOR method often is referred to as *underrelaxation method*, whereas for $\omega > 1$ it is also called *overrelaxation method*. One can show that the SOR method is convergent for symmetric positive definite matrices A if $0 < \omega < 2$ holds (cf. Hackbusch [86, Theorem 4.4.21]).

For a more general overview of relaxation-based iterative methods including the exact prerequisites of A for convergence, the reader is referred to standard literature (e.g., Axelsson [6], Hackbusch [86] and others). A discussion of the exact prerequisites on the choice of damping parameters ω for the different relaxation-based methods can be found in textbooks (e.g., James [100], Varga [200]).

2.1.2. Diagonal dominance and convergence theory for relaxation-based methods

In a linear system the matrix properties dominate the convergence of iterative solution methods. In this section the role of the *diagonal dominance* of matrix A for relaxation-based methods is discussed.

Definition 2.1.2 (Diagonal dominance)**.** A matrix $\mathrm{A} = \big(a_{ij}\big) \in \mathbb{R}^{n\times n}$ is called *irreducible (weakly) diagonally dominant*, if

$$\big|a_{ii}\big| \geq \sum_{\substack{j=1\\ j\neq i}}^{n} \big|a_{ij}\big| \tag{2.4}$$

for all $1 \leq i \leq n$, with strict inequality for at least one i. A $n \times n$ matrix is *strictly diagonally dominant* if strict inequality holds in (2.4) for all i.

Irreducible weak diagonal dominance of A is an important property for sufficient convergence criteria of relaxation-based methods. Here, a completely intuitive approach is used to understand the importance of A being diagonal dominant in context of relaxation-based smoothing methods.

Example 2.1.3 (Relaxation-based smoothers and diagonal dominance)**.** The following minimal example helps to understand the role of diagonal dominance for relaxation-based smoothers. Consider the linear system that is given by

$$\begin{pmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \text{with the exact solution } \widehat{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{2.5}$$

Let ε be a parameter with $|\varepsilon| \neq 1$. Using $x^0 = \big(1, 1\big)^{\mathsf{T}}$ as initial guess for the iterative process, the k-th iterate with a Jacobi iteration (see (2.3) with $\omega = 1$) is found to be $x^k = \big(\varepsilon^k, \varepsilon^k\big)^{\mathsf{T}}$.

For $\varepsilon < 1$ the system matrix in (2.5) is (strictly) diagonal dominant and x^k converges against the exact solution $\widehat{x}$ for $k \to \infty$. However, for $\varepsilon > 1$ the system matrix is non-diagonally dominant and the Jacobi iteration diverges.

For problems with only slightly non-diagonally dominant rows a reasonable choice of the damping parameter ω may lead to a convergent iterative method. But, depending on the eigenvalue spectrum, one may find relaxation-based iterative methods to fail for matrices with some non-diagonally dominant rows.

Remark 2.1.4 (Diagonal dominance for FEM matrices). Matrices arising from finite element discretizations for many applications turn out to be weakly diagonal dominant. In practice, Dirichlet boundary conditions are incorporated in the system matrices by zeroing out the corresponding lines in the matrix and putting entries of value one on the diagonal. Following Definition 2.1.2, the Dirichlet rows automatically transform a weakly diagonally dominant matrix to an irreducible (weakly) diagonally dominant matrix.

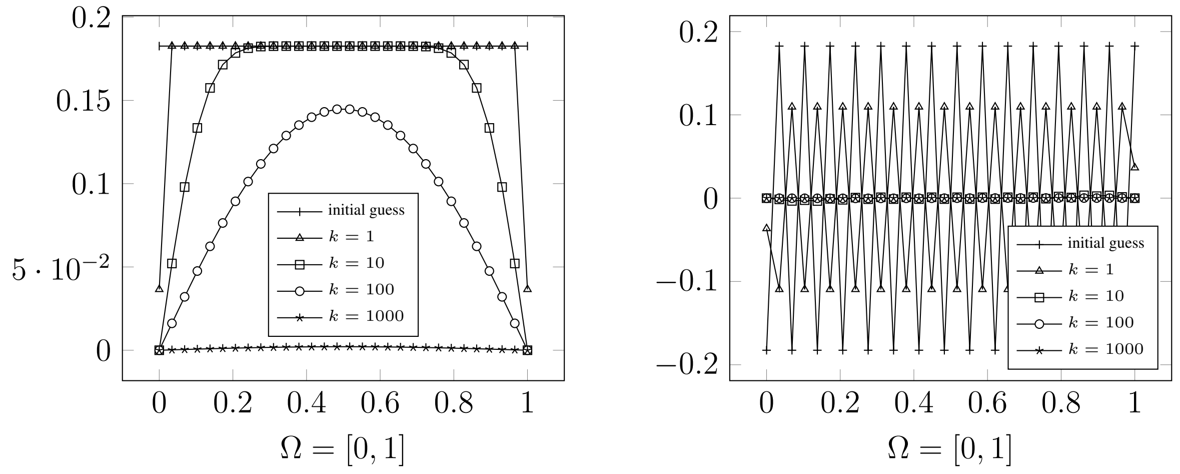
It follows from Gershgorin's theorem that a strictly diagonally dominant matrix A is nonsingular and therefore has a uniquely determined inverse A^{-1}. This statement can be weakened and proven to be true for irreducible weakly diagonally dominant matrices. For the proof the reader is referred to Taussky [184], Varga [200] and references therein. Diagonal dominance of a matrix turns out to be an easy-to-check criterion for convergence in context of relaxation-based methods. For multigrid methods especially the smoothing property is primarily essential.

2.2. Smoothing effect

The idea behind multigrid methods in general is based on the observation that many cheap iterative methods for solving linear systems, such as relaxation-based methods (see Section 2.1.1), behave differently for certain (low-frequency) error modes than for high-frequency error modes. The following example demonstrates the error smoothing effect and shows how the character of the initial guess vector x^0 influences the behavior of the solution process.

Example 2.2.1 (Smoothing effect of Jacobi method). To demonstrate the smoothing effect of relaxation-based methods a very simple linear system is used resulting from the discretization of a 1D finite difference stencil of the Laplace equation $-\Delta u = 0$ on the domain $\Omega = [0, 1]$ with homogeneous Dirichlet boundaries. Using a finite difference discretization with $n = 30$ equidistant nodes, the resulting linear system has the form $\mathrm{A}\widehat{x} = 0$ with $\mathrm{A} := \langle -1, 2, -1 \rangle \in \mathbb{R}^{n\times n}$. Here, the $\langle -1, 2, -1 \rangle$ is a short notation for tridiagonal operators with entries of value 2 on the main diagonal and -1 on the off diagonals. Homogeneous Dirichlet boundaries have been incorporated in A by removing the non-zeros in the first and last row and putting an entry of value one on the diagonal. The $\widehat{x}$ denotes the exact solution of the linear system, which is known to be $\widehat{x} = 0$. To solve the linear system we apply some sweeps with the damped Jacobi method starting with different initial guesses $x_a^0 = \frac{1}{\sqrt{n}}\big(1, 1, 1, \dots, 1\big)^{\mathsf{T}} \in \mathbb{R}^n$ and $x_b^0 = \frac{1}{\sqrt{n}}\big(-1, 1, -1, \dots, (-1)^i, \dots\big)^{\mathsf{T}}_{i=1\dots n}$.

With the known exact solution $\widehat{x} = 0$, it follows $x^k = e^k$ for the approximate solution vector x^k. Thus, Figure (2.1) shows the error e^k when applying $k \in \{1, 10, 100, 1000\}$ sweeps with a damped Jacobi iteration ($\omega = 0.8$) using the initial guess x_a^0 and x_b^0 respectively. By comparing the Figures 2.1a and 2.1b, one finds a different behavior for the constant vector x_a^0 as initial guess compared to the highly oscillatory initial guess in x_b^0. The high oscillatory error components obviously are damped out efficiently after a quite small number of iterations, whereas the



(a) Error e^k using the constant vector x_a^0 as initial guess.

(b) Error e^k using the highly oscillatory vector x_b^0 as initial guess.

Figure 2.1.: Demonstration of smoothing effect for the damped Jacobi iteration ($\omega = 0.8$) applied to a 1D diffusion equation on $\Omega = [0, 1]$. The number of Jacobi sweeps to approximate the exact solution $x = 0$ is visually compared when starting with a constant initial guess in Figure 2.1a or with a highly oscillating initial guess in Figure 2.1b.

low-frequency error components, which are represented by the constant vector x_a^0 for the initial guess, are slow in convergence.

This is a general observation: Relaxation-based methods tend to converge slowly for low-frequency error components in e, whereas high-frequency components are damped very effectively. This motivates the term *smoothing method* for relaxation-based iterations as high-frequency parts are damped out quickly such that only the “smooth” low-frequency errors are remaining. More detailed studies of this effect can be found in the textbooks Hackbusch [86, 87], Trottenberg et al. [186] or Wesseling [212] based on local Fourier analysis.

Motivated by this observation, the multigrid principle now attempts to rescue relaxation-based methods by projecting out the error components which converge slowly.

2.3. Coarse level correction – Motivation for multigrid

Relaxation methods alone suffer from slow-converging error components such that the convergence speed is not satisfactory (see Example 2.2.1 from Section 2.2). To overcome this issue one introduces coarser representations of the fine level problem and apply smoothing methods on different levels to optimally reduce error components with a relatively high frequency. Then, the coarse level correction is supposed to complement the smoothing process and handle the low-frequency error modes.

2.3.1. Geometric multigrid versus algebraic multigrid

In order to find coarse representations of the fine level problem, there exist different techniques. In general, one has to distinguish geometric and algebraic multigrid methods (cf. Haase and Langer [84]).

When thinking in geometric terms, a classification of error components into low- and high-frequency components is induced by the underlying mesh, i.e., a low frequency error component on a fine mesh can be interpreted as a high-frequency error component on a coarser mesh. This gives rise to the development of *Geometric Multigrid (GMG)* methods, where low-frequency components of the error are transferred to a coarser mesh, such that they can effectively be handled by relaxation-based smoothing methods (cf. Hackbusch [85, 87]). Geometric multigrid methods use coarse meshes that are explicitly generated or given by the user. For each mesh the smoothing process is to be selected, such that it can effectively reduce the corresponding high-frequency error components relative to the current mesh. In geometric multigrid methods low-frequency error components are smooth in a geometric sense (cf. Figure 2.1a versus Figure 2.1b).

In contrary to geometric multigrid methods, the *Algebraic Multigrid (AMG)* approach is not based on meshes, but uses purely algebraically defined coarse representations of the fine level problem, which makes them particularly interesting for problems on unstructured meshes. Instead of low-frequency error components relative to an underlying mesh, the term *slow-converging* error components is used. A precise definition of *slow-converging* (or *algebraically smooth*) error components can be found in Section 2.5.2 in this thesis. For algebraic multigrid methods the smoothing process is fixed on each multigrid level and suitable transfer operators have to be constructed by exploiting the information from (algebraically) smooth slow-converging error components. Technically one can distinguish classical (or Ruge Stüben) AMG methods (cf. Alber [5], Brandt [38], Ruge and Stüben [167], Stüben [181], Stüben and Trottenberg [182], Stüben [183]) and (smoothed) aggregation-based AMG methods (cf. Brezina et al. [43], Gee et al. [71], Olson and Schroder [150], Vaněk et al. [192, 199]). For the first class, one selects a set of coarse level nodes from the set of fine level nodes using different coarse level selection strategies (cf. Adams [2], Alber and Olson [4]). The fine level node information is then interpolated using the information from the selected coarse level nodes.

In this thesis, our focus is on aggregation-based AMG methods. Instead of selecting a set of coarse level nodes from the fine level nodes, subsets of fine level nodes are agglomerated into so-called aggregates, which represent a "super node" on the next coarser multigrid level. Concrete details on how to generate multigrid transfer operators following the concept of aggregation-based AMG methods can be found in Chapter 3. In the next subsection, the general concept of the coarse level correction is briefly introduced, which is, besides the smoothing, one of the key ingredients of all multigrid methods.

2.3.2. Coarse level correction iteration matrix

In this section the focus is on the general abstract concept of coarse level correction, which is shared by all kinds of multigrid methods.

Assume $P_{\ell+1} : \mathbb{R}^{n_{\ell+1}} \to \mathbb{R}^{n_\ell}$ to be a linear and injective mapping from a coarse level $\ell + 1$ to a fine level ℓ, which is referred to as prolongation operator. In a similar way the restriction R is introduced as a linear and surjective mapping $R_{\ell+1} : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_{\ell+1}}$, which maps fine-level information to coarse-level information. One can think of the prolongation and restriction operators as rectangular matrices $P_{\ell+1} \in \mathbb{R}^{n_\ell \times n_{\ell+1}}$ and $R_{\ell+1} \in \mathbb{R}^{n_{\ell+1} \times n_\ell}$ of rank $n_{\ell+1}$ for all levels $\ell \geq 0$ with $\ell = 0$ denoting the finest level. A common choice for the restriction operator is to use $R = P^{\mathsf{T}}$. Then, the coarse level matrix $\mathrm{A}_{\ell+1}$ is built using the *Galerkin product* given by

$$\mathrm{A}_{\ell+1} = R_{\ell+1}\mathrm{A}_\ell P_{\ell+1} \text{ for } \ell \geq 0. \tag{2.6}$$

Remark 2.3.1 (Non-singular coarse level matrices). For a non-singular fine level matrix A_ℓ transfer operators $P_{\ell+1}$ and $R_{\ell+1}$ with full rank are needed to obtain non-singular coarse level matrices $\mathrm{A}_{\ell+1}$ from (2.6). In particular, the prolongation operator $P_{\ell+1}$ (and the restriction operator $R_{\ell+1}$) must not contain zero columns (and zero rows, respectively), which would result in a singular matrix $\mathrm{A}_{\ell+1}$ containing zero rows.

Let $\widehat{x} \in \mathbb{R}^{n_\ell}$ denote the exact solution of the problem $\mathrm{A}_\ell \widehat{x} = b_\ell$ on level ℓ and x_ℓ be an approximation of the level solution $\widehat{x}$ obtained from some steps using a smoothing iteration from Section 2.1.1. Then,

$$e_\ell := x_\ell - \widehat{x} \tag{2.7}$$

represents the corresponding error on level ℓ between the exact solution and the approximation after some smoothing iterations. Vice versa: the exact solution can be calculated from (2.7) using

$$\widehat{x} = x_\ell - e_\ell. \tag{2.8}$$

Obviously, the error e_ℓ on level ℓ satisfies the equation

$$\mathrm{A}_\ell e_\ell = r_\ell, \tag{2.9}$$

where $r_\ell := \mathrm{A}_\ell x_\ell - b_\ell$ denotes the residual or defect on level ℓ.

Similar to (2.9), one can declare a coarse error $e_{\ell+1}$ by the coarse-level equation

$$\mathrm{A}_{\ell+1} e_{\ell+1} = r_{\ell+1}, \tag{2.10}$$

with the coarse residual defined by $r_{\ell+1} := R_{\ell+1} r_\ell$. Supposing the fine level error e_ℓ to be smooth, one can express it by means of the coarse error $e_{\ell+1}$ with $e_\ell \approx P_{\ell+1} e_{\ell+1}$. Assuming that the coarse level equation (2.10) can be solved exactly, it follows from (2.8), (2.9) and (2.10) that

$$\begin{aligned} \widehat{x} = x_\ell - e_\ell \approx x_\ell - P_{\ell+1} e_{\ell+1} = \\ x_\ell - P_{\ell+1}\mathrm{A}_{\ell+1}^{-1} r_{\ell+1} = x_\ell - P_{\ell+1}\mathrm{A}_{\ell+1}^{-1} R_{\ell+1}\big(\mathrm{A}_\ell x_\ell - b_\ell\big). \end{aligned} \tag{2.11}$$

Using (2.11) one can define an iterative method by $x_\ell^k \mapsto x_\ell^{k+1} := \Phi^{CLC}(x_\ell^k, b_\ell)$ for the coarse level correction with

$$\Phi^{CLC}(x_\ell, b_\ell) := x_\ell - P_{\ell+1}\mathrm{A}_{\ell+1}^{-1} R_{\ell+1}\big(\mathrm{A}_\ell x_\ell - b_\ell\big). \tag{2.12}$$

From (2.12) one finds the iteration matrix of the coarse level correction as

$$M_{C_\ell} = I - P_{\ell+1}\mathrm{A}_{\ell+1}^{-1} R_{\ell+1}\mathrm{A}_\ell. \tag{2.13}$$

Theorem 2.3.2. *Assume* A *to be symmetric. Let* $P_{\ell+1}$ *and* $R_{\ell+1}$ *be chosen according to Remark 2.3.1 to guarantee a non-singular coarse level matrix* $\mathrm{A}_{\ell+1}$*. Then, under the assumption* $R_{\ell+1} = P_{\ell+1}^{\mathsf{T}}$ *and the Galerkin product* (2.6) *the coarse level correction operator* M_C *is an orthogonal projector with respect to the energy inner product* $(\cdot,\cdot)_{\mathrm{A}}$*, with* $Rg\big(M_{C_\ell}\big)$ *being orthogonal to* $Rg\big(P_{\ell+1}\big)$.

Proof. A simple calculation gives

$$(P_{\ell+1}, M_{C_\ell})_{\mathrm{A}} = P_{\ell+1}^{\mathsf{T}}\mathrm{A}_\ell\big(I - P_{\ell+1}\mathrm{A}_{\ell+1}^{-1}R_{\ell+1}\mathrm{A}_\ell\big) = \\ P_{\ell+1}^{\mathsf{T}}\mathrm{A}_\ell - P_{\ell+1}^{\mathsf{T}}\mathrm{A}_\ell P_{\ell+1}\mathrm{A}_{\ell+1}^{-1}R_{\ell+1}\mathrm{A}_\ell = P_{\ell+1}^{\mathsf{T}}\mathrm{A}_\ell - P_{\ell+1}^{\mathsf{T}}\mathrm{A}_\ell = 0.$$

□

The orthogonality of the prolongation operator $P_{\ell+1}$ and the coarse level correction shows mathematically the complementary meaning of the coarse level correction and the fine level information, which is separated by the transfer operators.

Remark 2.3.3. In case of a non-symmetric matrix A, the error propagation operator M_C of the Galerkin two-grid correction using $R = P^{\mathsf{T}}$ is an oblique projection in the sense that the spaces involved are not orthogonal with respect to any known, practical inner product (cf. Brezina et al. [44]). For this reason it is recommended to use a Petrov–Galerkin coarsening for the non-symmetric case, dropping the constraint $R = P^{\mathsf{T}}$ and building the restriction operator separately (see also Section 4.2.2).

2.4. Multigrid algorithms

Putting together the complementary concepts of level smoothing (see Section 2.2) and coarse level correction (see Section 2.3) a two-level method can be introduced and recursively extended to multigrid methods.

2.4.1. Two-level algorithm

Let $M_{\mathscr{S}}^{\nu_1}$ and $M_{\mathscr{S}}^{\nu_2}$ denote the iteration matrix of a smoothing method $\mathscr{S}_\ell$ (e.g., a relaxation-based smoother from Section 2.1.1) with ν_1 pre- and ν_2 post-smoothing sweeps on the fine level. Then, with (2.12), the two-level method is defined by $\Phi^{TLM} := \mathscr{S}^{\nu_2} \circ \Phi^{CLC} \circ \mathscr{S}^{\nu_1}$ with the corresponding iteration matrix

$$M_T = M_{\mathscr{S}}^{\nu_2}\, M_C\, M_{\mathscr{S}}^{\nu_1}. \tag{2.14}$$

The two-level method from (2.14) can be algorithmically formulated as shown in Algorithm 1. First, ν_1 pre-smoothing sweeps are applied to the solution vector x_ℓ on the fine level. Then, the residual vector r_ℓ is calculated using the pre-smoothed solution vector and restricted to the coarse level. Next, the coarse level correction $e_{\ell+1}$ is calculated using a direct solver. The fine level solution vector x_ℓ is then updated with the prolongated coarse level correction. Finally, the current solution vector is smoothed by applying ν_2 iterations with the post-smoothing method.

2.4.2. Multi-grid algorithm

The extension of the two-level method from Algorithm 1 to a multigrid method is straightforward by recursively applying the two-level method for the coarser levels. Similarly to (2.14) the

Algorithm 1: Two-level multigrid algorithm

```
*Two-level multigrid method*
**Procedure** $\Phi^{TLM}$ ($x_\ell$,$b_\ell$)
    *Apply $\nu_1$ pre-smoothing sweeps using $\mathscr{S}$ as smoother*
    **for** $i \leftarrow 1$ **to** $\nu_1$ **do**
        $x_\ell \leftarrow \mathscr{S}_\ell(x_\ell, b_\ell)$
    **end**
    *Calculate fine-level residual/defect*
    $r_\ell \leftarrow \mathrm{A}_\ell x_\ell - b_\ell$
    *Restrict fine level residual $r_\ell$ to coarse level*
    $r_{\ell+1} \leftarrow R_{\ell+1} r_\ell$
    *Solve for the coarse error correction $e_{\ell+1}$ on the coarse level $\ell + 1$*
    $e_{\ell+1} \leftarrow \mathrm{A}_{\ell+1}^{-1} r_{\ell+1}$
    *Prolongate coarse level error correction $e_{\ell+1}$ to fine level*
    $e_\ell \leftarrow P_{\ell+1} e_{\ell+1}$
    *Correct fine level solution $x_\ell$ using the fine level error correction $e_\ell$*
    $x_\ell \leftarrow x_\ell - e_\ell$
    *Apply $\nu_2$ post-smoothing sweeps using $\mathscr{S}$ as smoother*
    **for** $i \leftarrow 1$ **to** $\nu_2$ **do**
        $x_\ell \leftarrow \mathscr{S}_\ell(x_\ell, b_\ell)$
    **end**
    **return** $x_\ell$
```

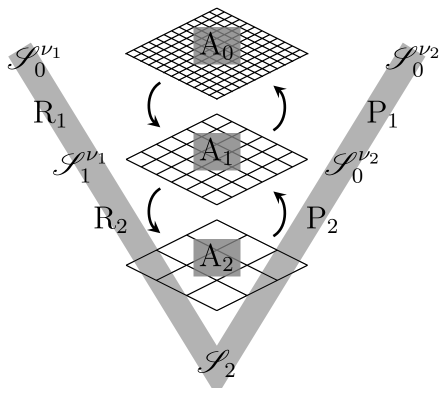


Figure 2.2.: Visualization of a V-cycle multigrid algorithm with 3 multigrid levels. For visualization purposes coarse meshes are used to represent the coarse levels. Note that in an algebraic multigrid context the coarse levels are built from purely algebraic information (with no coarse meshes involved).

iteration matrix M_{MG} for the multigrid case is recursively defined by

$$\begin{aligned} M_{MG}^{(\ell_{\max})} &= 0 && \text{(coarsest level)} \\ M_{MG}^{(\ell_{\max}-1)} &= M_T && \text{(two-level correction)} \\ M_{MG}^{(\ell)} &= M_{\mathscr{S}}^{\nu_2}\left(I - P_\ell\Big(I - \big(M_{MG}^{(\ell+1)}\big)^\gamma\Big)\mathrm{A}_{\ell+1}^{-1}R_\ell\mathrm{A}_\ell\right)M_{\mathscr{S}}^{\nu_1} && \text{(multigrid correction)} \end{aligned} \tag{2.15}$$

for $\ell = 0, \ldots, \ell_{\max} - 2$ and $\gamma \in \{1, 2\}$. Algorithm 2 describes algorithmically the recursively defined multi-level method from (2.15). For $\ell_{\max} = 0$ it corresponds to a direct solve on a single level. With $\ell_{\max} = 1$, the method defined in (2.15) coincides with the two-level algorithm from (2.14).

Note that in contrast to the two-level method, where just a direct solver was used for calculating the coarse level error $e_{\ell+1}$, the multi-level method in Algorithm 2 allows to call the coarse correction step more than once. If $\gamma = 1$ in Algorithm 2, one obtains the so-called V-cycle as shown in Figure 2.2 for a 3 level multigrid method. Choosing $\gamma = 2$ results in the so-called W-cycle (cf. Hackbusch [87, Section 2.5]). With $\nu_1 = 0$ and $\nu_2 \neq 0$ one obtains a so-called “saw-tooth” cycle (cf. Wesseling [211]) and in the literature one can find more special multi-grid cycles.

Remark 2.4.1 (Multigrid as preconditioner). For the examples in this thesis, the multigrid method is used as preconditioner within a GMRES solver (cf. Section 1.2.2 and Appendix A). That is, applying the preconditioner W to a vector v via $y = W^{-1}v$ (see Remark A.2.2) corresponds to performing some multigrid cycles applied to v. In this thesis, one V-cycle sweep is applied for preconditioning.

2.5. Convergence theory

In the previous section the multigrid method has been defined as a recursion of cheap smoothing methods and a complementary coarse level correction. Before discussing aggregation-based

Algorithm 2: Multigrid algorithm

```
Multi-level multigrid method
Procedure $\Phi^{MLM}$ $(x_\ell, b_\ell, \ell, \ell_{\max})$
    Check $\ell$ for coarsest level
    if $\ell == \ell_{\max}$ then
        Solve for the coarse level problem
        $x_\ell \leftarrow \mathrm{A}_\ell^{-1} b_\ell$
        return $x_\ell$
    end
    Apply multigrid method recursively
    else
        Apply $\nu_1$ pre-smoothing sweeps using $\mathscr{S}$ as smoother
        for $i \leftarrow 1$ to $\nu_1$ do
            $x_\ell \leftarrow \mathscr{S}_\ell(x_\ell, b_\ell)$
        end
        Calculate coarse level residual
        $r_{\ell+1} \leftarrow R_{\ell+1}\big(\mathrm{A}_\ell x_\ell - b_\ell\big)$
        Initialize coarse level error $e_{\ell+1}$
        $e_{\ell+1} \leftarrow 0$
        Recursively call $\Phi^{MLM}$ for the next coarser level
        for $i \leftarrow 1$ to $\gamma$ do
            $e_{\ell+1} \leftarrow \Phi^{MLM}$ $(e_{\ell+1}, r_{\ell+1}, \ell + 1, \ell_{\max})$
        end
        Correct fine level solution $x_\ell$
        $x_\ell \leftarrow x_\ell - P_{\ell+1} e_{\ell+1}$
        Apply $\nu_2$ post-smoothing sweeps using $\mathscr{S}$ as smoother
        for $i \leftarrow 1$ to $\nu_2$ do
            $x_\ell \leftarrow \mathscr{S}_\ell(x_\ell, b_\ell)$
        end
        return $x_\ell$
    end
```

AMG methods in Chapter 3 with specific details about the construction of appropriate transfer operators, it is our intention to provide some insight into the basics of multigrid convergence theory with a particular focus on algebraic multigrid methods.

First, the concept of algebraically smooth errors is motivated, which builds the counterpart of geometric smoothness that geometric multigrid methods rely on. Then, the so-called smoothing and approximation properties are introduced, which are widely used in context of multigrid convergence theory. For reasons of simplicity, A is assumed to be symmetric positive definite. The section is closed with a brief literature overview of different convergence theorems, including comments on non-symmetric problems and specific multigrid theory for aggregation-based multigrid methods that are in detail introduced in Chapter 3 and later on used throughout all examples in this thesis.

2.5.1. Interplay of smoothing and coarse level correction

To gain a better understanding of the interplay of the smoothing and the coarse correction step in a multigrid method and its meaning for the convergence it is worth to have a closer look to the following convergence theorem for the V-cycle given in Ruge and Stüben [167]:

Theorem 2.5.1. *Let* A *be symmetric positive definite. Assume that the transfer operators* P_ℓ *and* $R_\ell := P_\ell^{\mathsf{T}}$, $\ell = 1, \ldots, \ell_{\max}$ *have full rank and that* $\mathrm{A}_{\ell+1} := R_{\ell+1}\mathrm{A}_\ell P_{\ell+1}$. *Furthermore, assume that for all error vectors* e_ℓ *either*

$$\left\|M_{\mathscr{S}_\ell}^{\nu_2} e_\ell\right\|_{\mathrm{A}}^2 \leq \|e_\ell\|_{\mathrm{A}}^2 - \delta_2 \|M_{C_\ell} e_\ell\|_{\mathrm{A}}^2 \tag{2.16a}$$

$$\text{or } \left\|M_{\mathscr{S}_\ell}^{\nu_1} e_\ell\right\|_{\mathrm{A}}^2 \leq \|e_\ell\|_{\mathrm{A}}^2 - \delta_1 \left\|M_{C_\ell} M_{\mathscr{S}_\ell}^{\nu_1} e_\ell\right\|_{\mathrm{A}}^2 \tag{2.16b}$$

or both conditions hold with some $\delta_1 > 0$ *and* $\delta_2 > 0$ *independently of* e_ℓ *and* ℓ.

i) In case (2.16a) *holds, the V-cycle has a convergence factor with respect to the energy norm* $\|\cdot\|_{\mathrm{A}}$ *bounded above by* $\sqrt{1-\delta_2}$, *provided that at least* ν_2 *post-smoothing steps are performed after each coarse level correction step.*

ii) In case (2.16b) *holds, the V-cycle convergence factor is bounded above by* $\frac{1}{\sqrt{1+\delta_1}}$, *provided that at least* ν_1 *pre-smoothing steps are performed before each coarse level correction step.*

iii) If both (2.16b) *and* (2.16a) *hold the V-cycle convergence factor is bounded above by* $\frac{\sqrt{1-\delta_2}}{\sqrt{1+\delta_1}}$, *provided that* ν_1 *and* ν_2 *pre- and post-smoothing sweeps are performed.*

Proof. For the proof the reader is referred to Ruge and Stüben [167, Section 4.3.1]. □

The conditions in (2.16) reflect the interplay between the smoothing and the coarse level correction. Let's have a closer look to (2.16a) for the interpretation: error components $\hat{e_\ell}$ in e_ℓ, which cannot be efficiently be reduced by the coarse level operator M_C (i.e., $\|M_{C_\ell}\hat{e}_\ell\|_{\mathrm{A}} \approx \|\hat{e}_\ell\|_{\mathrm{A}}$), have to be reducible by the smoothing operation with the iteration matrix $M_{\mathscr{S}_\ell}^{\nu_2}$. Vice versa: for error components $\hat{e_\ell}$ in e_ℓ, which are effectively reduced by the coarse level correction M_C (i.e., $\|M_{C_\ell}\hat{e}_\ell\|_{\mathrm{A}} \ll \|\hat{e}_\ell\|_{\mathrm{A}}$), the smoothing operator $\mathscr{S}_\ell$ with the iteration matrix $M_{\mathscr{S}_\ell}^{\nu_2}$ is allowed to be ineffective, since these error components are approximately in the range of the prolongation operator P. The interpretation of (2.16b) is very similar, if one considers error components of $e_\ell^{\nu_1} := M_{\mathscr{S}_\ell}^{\nu_1} e_\ell$ after applying ν_1 pre-smoothing sweeps using $\mathscr{S}_\ell^{\nu_1}$.

Remark 2.5.2. Similar bounds can also be found in McCormick [131, Lemma 2.3, Theorem 3.4 and Section 5] or in Mandel et al. [123]. The proofs usually have a recursive character and are based on the so-called smoothing and approximation properties in a two-level setting. Smoothing and approximation properties are introduced in Section 2.5.3. For a very recent overview of theoretical bounds for the V-cycle the reader also may refer to MacLachlan and Olson [122].

2.5.2. Algebraically smooth error

As known from Section 2.2, cheap iterative methods show slow convergence for certain error modes. Therefore, it is an important prerequisite for overcoming the slow convergence to characterize and detect such slowly converging error components.

In the context of multigrid methods, an error is regarded as smooth, if it can be approximated properly on some coarse level. However, there are some differences in the concept of smooth errors when looking at geometric and algebraic multigrid methods as already mentioned in Section 2.3.1. In geometric multigrid methods the term "smooth" is used in a more "natural" geometric sense, i.e., the smoothness of an error e_ℓ is always relative to a given grid. For algebraic multigrid methods without underlying coarse meshes, it is not possible to use the term "smooth" in a classical geometric way. For the construction of algebraic multigrid methods a purely algebraic definition of the term "smooth" is essential to be able to distinguish slowly converging (algebraically smooth) error components from error components, which can be reduced effectively by the smoothing method on the current level.

Definition 2.5.3 (Smooth error)**.** An error e_ℓ is defined to be (algebraically) smooth, if it is slowly converging with respect to the smoother $\mathscr{S}_\ell$, which is equivalent to $\|M_{\mathscr{S}_\ell} e_\ell\|_{\mathrm{A}} \approx \|e_\ell\|_{\mathrm{A}}$.

So, an error e_ℓ is called smooth if the smoothing process $\mathscr{S}_\ell$ cannot efficiently further reduce the error, such that the coarse level correction has to take care of it. Therefore, one main objective of algebraic multigrid methods is to construct prolongation operators for which smooth errors e_ℓ are in the range $Rg\big(P\big)$ of the prolongation operators P.

Remark 2.5.4 (Smooth error in context of relaxation-based methods). For common relaxation-based smoothers (see Section 2.1.1) the common iteration matrix has the form $M_{\mathscr{S}_\ell} := I - Q^{-1}\mathrm{A}$. For characterizing the smoothness of an error e_ℓ using $\|M_{\mathscr{S}_\ell} e_\ell\|_{\mathrm{A}} \approx \|e_\ell\|_{\mathrm{A}}$, one finds the relation

$$e_\ell^{k+1} = \big(I - Q^{-1}\mathrm{A}\big)e_\ell^k = e_\ell^k - Q^{-1}\mathrm{A}e_\ell^k \;\approx\; e_\ell^k \quad \Leftrightarrow \quad Q^{-1}\mathrm{A}e_\ell^k = Q^{-1}r_\ell^k \approx 0. \tag{2.17}$$

That is, for common relaxation schemes an algebraically smooth error e_ℓ is characterized by the residual $r_\ell = \mathrm{A}_\ell e_\ell$ being small in some norm relative to the current level ℓ compared to the error e_ℓ itself. The following example helps to gain a better understanding of algebraically smooth errors.

Example 2.5.5 (Characterization of algebraic smoothness of errors.)**.** To demonstrate the meaning of the term algebraically smooth errors, the Example 2.2.1 presented in Section 2.2 is reviewed. In Figure 2.3 the residual r and the error e is plotted in some norm over the number of Jacobi sweeps k. First, one can see again how the error reacts differently depending on the initial solution. When using the high-oscillatory initial solution x_b^0, as defined in Example 2.2.1, the convergence and error reduction is quite fast compared with the slow-convergent behavior for the low-frequency initial solution vector x_a^0. One can also see that the residual norm is notably

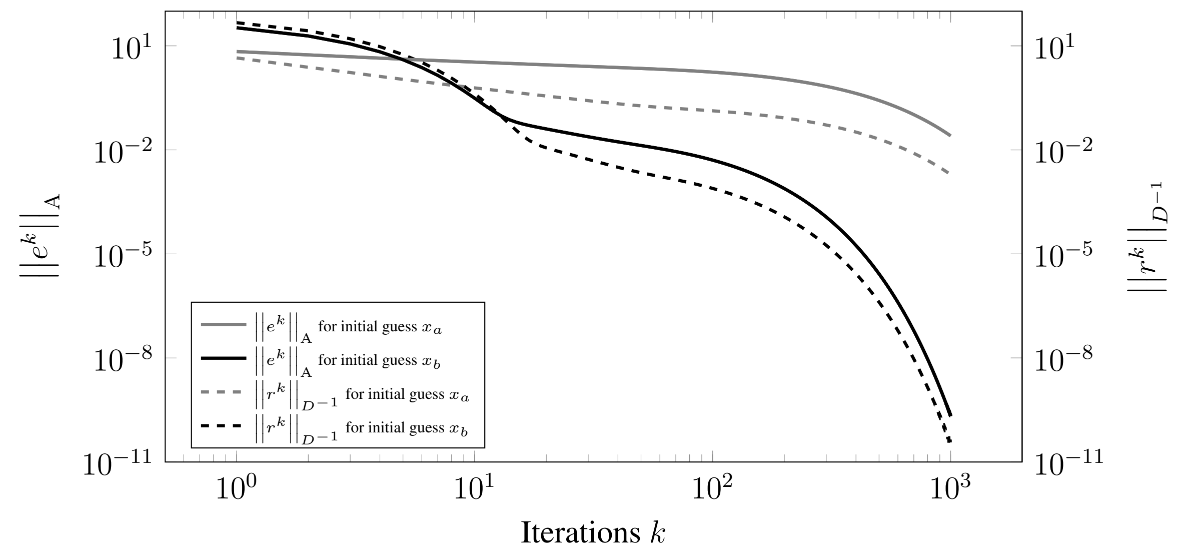


Figure 2.3.: Plot of error and residual over number of Jacobi sweeps k for the 1D diffusion example from Example 2.2.1 using different initial solution vectors. The error norm and residual norm is motivated in Section 2.5.3.1. The low-frequency starting solution represented by a normalized constant vector x_a^0 is compared with a highly oscillatory initial guess x_b^0 as defined in Example 2.2.1.

below the error norm at some point, especially if the error reduction is ineffective. Thus, the residual norm can be used as indicator for algebraic smoothness. The specific choice of the error and residual norm is motivated and further discussed in Section 2.5.3.1.

Algebraic multigrid methods are based on the Definition 2.5.3 of algebraically smooth errors. These include not only classical algebraic multigrid methods (e.g., Mandel et al. [123], Ruge and Stüben [167]), but also more recent algebraic multigrid methods based on aggregation (e.g., Vaněk et al. [192, 199]). There are more advanced methods (cf. Brezina et al. [43]), which heavily rely on the definition of algebraically smooth errors to detect near-kernel modes of the level matrix A_ℓ for adaptive smoothed aggregation.

In the corresponding literature one often finds different variants of smoothing and approximation properties which are essential for proving convergence.

2.5.3. Smoothing and approximation properties

The intention of this section is to give a brief overview of different formulations of so-called smoothing and approximation properties that are in use throughout the literature on multigrid convergence theory. The idea is to have some more practically useful criteria than (2.16) for convergence, which reflect the complementary effect of the smoothing process and the coarse level correction step.

2.5.3.1. Smoothing and approximation property according to Stüben

Following Ruge and Stüben [167, Section 4.3.2], the assumptions in (2.16) can be split into two separate inequality assumptions for the smoothing and coarse level correction. With $\alpha_2 >$

0, $\beta_2 > 0$, such that it is $\delta_2 = \frac{\alpha_2}{\beta_2}$ in (2.16a), one can satisfy (2.16a) by requiring

$$\left\|M_{\mathscr{S}_\ell}^{\nu_2} e_\ell\right\|_{\mathrm{A}}^2 \leq \|e_\ell\|_{\mathrm{A}}^2 - \alpha_2 \|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}^2 , \tag{2.18a}$$

$$\|M_{C_\ell} e_\ell\|_{\mathrm{A}}^2 \leq \beta_2 \|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}^2 . \tag{2.18b}$$

Similarly, the two inequality conditions

$$\left\|M_{\mathscr{S}_\ell}^{\nu_1} e_\ell\right\|_{\mathrm{A}}^2 \leq \|e_\ell\|_{\mathrm{A}}^2 - \alpha_1 \left\|M_{\mathscr{S}_\ell}^{\nu_1} e_\ell\right\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}^2 , \tag{2.19a}$$

$$\|M_{C_\ell} e_\ell\|_{\mathrm{A}}^2 \leq \beta_1 \|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}^2 \tag{2.19b}$$

with $\alpha_1 > 0$, $\beta_1 > 0$ and $\delta_1 = \frac{\alpha_1}{\beta_1}$ are sufficient for (2.16b) to hold.

Smoothing property: The two conditions in (2.18a) and (2.19a) describe the so-called smoothing property in context of algebraic multigrid methods (cf. Stüben [181, A.3.2]).

Definition 2.5.6 (Smoothing property – Stüben)**.** A smoothing method $\mathscr{S}_\ell$ for $\ell \geq 0$ satisfies the *smoothing property* with respect to a symmetric positive definite matrix A, if

$$\|M_{\mathscr{S}_\ell} e_\ell\|_{\mathrm{A}}^2 \leq \|e_\ell\|_{\mathrm{A}}^2 - \alpha \|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}^2 \tag{2.20}$$

holds for $\alpha > 0$ independent of e_ℓ.

In the definition of the smoothing property (2.20) one has replaced $\|M_{C_\ell} e_\ell\|_{\mathrm{A}}$ from (2.16) by $\|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}$, which turns out to have similar characteristic properties, but does not depend on the coarse level correction operator M_{C_ℓ} from (2.13). The smoothing property (2.20) implies that the smoother $\mathscr{S}_\ell$ is efficient in reducing the error e_ℓ as long as $\alpha \|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}^2$ is rather large compared to $\|e_\ell\|_{\mathrm{A}}^2$. On the other hand, e_ℓ is algebraically smooth according to Definition 2.5.3, if $\|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}} \ll \|e_\ell\|_{\mathrm{A}}$. As one can easily verify, it is

$$\left(D^{-1} r_\ell, r_\ell\right)_e = \left(D^{-1}\mathrm{A}e_\ell, \mathrm{A}e_\ell\right)_e = \|e_\ell\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}^2 \ll \|e_\ell\|_{\mathrm{A}}^2 = (e_\ell, \mathrm{A}e_\ell)_e = (e_\ell, r_\ell)_e , \tag{2.21}$$

which indicates that the error e_ℓ is algebraically smooth, if the (scaled) residuals are much smaller than the error itself.

Note that the smoothing property holds for many typical iterative smoothing methods such as relaxation-based methods like Jacobi, Gauss–Seidel or symmetric Gauss–Seidel methods. The corresponding proofs can be found in the literature (cf. Stüben [181, Section A.3.2]).

Remark 2.5.7. The choice of the $\|\cdot\|_{\mathrm{A}^\intercal D^{-1}\mathrm{A}}$ norm for the error e_ℓ is not mandatory. In MacLachlan and Olson [122] a generalization is discussed, which might lead to sharper bounds using weak approximation assumptions for some specific problems.

Approximation property: The conditions (2.18b) and (2.19b) are referred to as *approximation condition* or *approximation property* (see, e.g., Ruge and Stüben [167, Section 4.3.2]).

Definition 2.5.8 (Approximation property – Stüben)**.** Suppose P and R to be transfer operators of full rank with $\mathrm{A}_{\ell+1} = R_{\ell+1}\mathrm{A}_\ell P_{\ell+1}$. Then the coarse level correction satisfies the approximation property, if

$$\|M_{C_\ell} e_\ell\|_{\mathrm{A}}^2 \leq \beta \, \|e_\ell\|_{\mathrm{A}^\mathsf{T} D^{-1}\mathrm{A}}^2 \tag{2.22}$$

holds for all $\beta > 0$.

The definitions of the smoothing and approximation properties in Definition 2.5.6 and 2.5.8 are widely used for algebraic multigrid methods. Even though the choice of the norms may sometimes vary in the details, all definitions in the literature are based on the same general principle of distinguishing algebraically smooth and algebraically non-smooth error components. In MacLachlan and Olson [122] there is a comprehensive overview with different extensions and generalizations of above smoothing and approximation properties giving some insight into the predictive capabilities of theoretical bounds of algebraic multigrid methods. However, the details are beyond the scope of this thesis.

2.5.3.2. Smoothing and approximation property according to Hackbusch

In context of geometric multigrid methods more "classical" smoothing and approximation properties are used in the literature (cf. Hackbusch [86, 87]). Following Hackbusch [86, Definition 10.6.3] one can define a slightly different smoothing property using

Definition 2.5.9 (Smoothing property – Hackbusch)**.** An iterative method $\mathscr{S}_\ell$ for $\ell \geq 0$ satisfies the *smoothing property*, if there are functions $\eta(\nu)$ and $\overline{\nu}(\mathrm{h})$ independent of ℓ, such that

$$\|\mathrm{A}_\ell \mathscr{S}_\ell^\nu\|_2 \leq \eta(\nu) \, \|\mathrm{A}_\ell\|_2 \qquad \text{for all } 0 \leq \nu < \overline{\nu}(\mathrm{h}_\ell), \ell \geq 0, \tag{2.23a}$$

$$\lim_{\nu\to\infty} \eta(\nu) = 0, \tag{2.23b}$$

$$\lim_{\mathrm{h}\to 0} \overline{\nu}(\mathrm{h}) = \infty \;\text{ or }\; \overline{\nu}(\mathrm{h}) = \infty. \tag{2.23c}$$

Remark 2.5.10. If (2.23a) and (2.23b) hold with $\overline{\nu}(\mathrm{h}) = \infty$, then the iteration $\mathscr{S}_\ell$ is convergent.

In (2.23a) the matrix norm $\|\cdot\|_2$ denotes the spectral norm. The main difference of (2.23a) compared to the smoothing property in (2.18a) or (2.19a) is that there is no separate handling of different error components with different norms. In fact, the smoothing property (2.23a) is very often used in context of geometric multigrid methods where one has not to distinguish between algebraically smooth and algebraically non-smooth error components.

Relaxation-based smoothers are known to satisfy the smoothing property from Definition 2.5.9 (cf. Hackbusch [87, Section 6.3]). The smoothing property for incomplete factorization methods is studied in Wittum [218, 219]. Non-symmetric problems are handled by perturbation arguments (cf. Hackbusch [87]), but only give reasonable results if the non-symmetry is not dominant. A very extensive discussion of the smoothing property with proofs for the different classes of smoothers can also be found in Wesseling [212].

The approximation property (according to Hackbusch [86, Section 10.6.3]) is supposed to quantify the requirement $P_{\ell+1}\mathrm{A}_{\ell+1}^{-1}R_{\ell+1}r_\ell \approx \mathrm{A}_\ell^{-1}r_\ell$ by

$$\left\|P_{\ell+1}\mathrm{A}_{\ell+1}^{-1}R_{\ell+1}r_\ell - \mathrm{A}_\ell^{-1}r_\ell\right\|_e \leq C_{\mathrm{A}} \frac{\|r_\ell\|_e}{\|\mathrm{A}_\ell\|_2} \qquad \text{for all } \ell \geq 1. \tag{2.24}$$

Therein $C_{\mathrm{A}} > 0$ is a constant independent of ℓ. The expression in (2.24) can be rewritten by means of the spectral norm as given in the following definition:

Definition 2.5.11 (Approximation property – Hackbusch)**.** Suppose P and R to be transfer operators of full rank with $\mathrm{A}_{\ell+1} = R_{\ell+1}\mathrm{A}_\ell P_{\ell+1}$. Then the coarse level correction satisfies the approximation property, if

$$\left\|\mathrm{A}_\ell^{-1} - P_{\ell+1}\mathrm{A}_{\ell+1}^{-1}R_{\ell+1}\right\|_2 \leq \frac{C_{\mathrm{A}}}{\|\mathrm{A}_\ell\|_2} \tag{2.25}$$

holds for all levels $\ell \geq 1$.

One important difference of (2.25) compared to the approximation property formulated in (2.22) is the different choice of the norm. However, it is easy to see that the approximation properties in (2.24) and (2.22) are equivalent when using $\mathrm{A}_\ell e_\ell = r_\ell$.

2.5.4. Full multigrid convergence

Standard techniques of multigrid convergence theory use the smoothing and approximation property from Section 2.5.3 to prove two-level convergence and extend it to the multilevel case (see, e.g., Hackbusch [87, Theorem 6.1.7, Theorem 7.1.2]). However, most of the early proofs suffer from some additional technicalities. For example, they may be valid only for the W-cycle with a sufficiently large number of smoothing steps (cf. Hackbusch [85, Theorem 4.4]). In practice, however, only one or just a few smoothing sweeps are used in a multigrid method.

Braess and Hackbusch [28], Braess [31], McCormick [131] and Verfürth [204] overcome this issue by analyzing the interplay between smoothing and coarse level correction directly, which allows for sharp estimates. For symmetric and positive definite matrices the classical proof for the V-cycle convergence is presented in Hackbusch [87, Theorem 7.2.2]. The connection between two-grid convergence bounds and multigrid convergence bounds is also studied in Napov and Notay [139]. The authors in Mandel et al. [123] prove convergence for symmetric positive definite matrices for any positive number of pre-and post-smoothing sweeps both for the V- and W-cycle. For linear systems resulting from the discretization of an elliptic partial differential equation, the V-cycle multigrid is known to have optimal convergence properties in the sense that convergence is independent of the number of levels and the mesh discretization parameter h.

Here, a V-cycle convergence theorem is given which is designed for purely algebraic multigrid methods. It can be found similarly in Mandel et al. [123], McCormick [131], Ruge and Stüben [167] and in a slightly extended variant in Napov [137].

Theorem 2.5.12 (Convergence of V-cycle according to McCormick [131])**.** *Assume* A_0 *to be symmetric positive definite. Let* $M_{MG}^{(\ell_{\max})}$ *define the multigrid method with prolongation operators* P_ℓ *and restriction operators* $R_\ell = P_\ell^{\mathsf{T}}$ *with* $\ell = 1, \dots, \ell_{\max}$. *The coarse level matrices are generated by the Galerkin product* $\mathrm{A}_{\ell+1} = R_{\ell+1}^{\mathsf{T}}\mathrm{A}_\ell P_{\ell+1}$ *for* $\ell = 0, \dots, \ell_{\max} - 1$. *Let* $M_{\mathscr{S}_\ell}^{\nu} := \left(I - Q^{-1}\mathrm{A}_\ell\right)^\nu$ *with* $\rho\left(M_{\mathscr{S}_\ell}\right) < 1$ *define the smoothing process with a symmetric* $Q = Q^{\mathsf{T}}$ *and assume the number of pre- and post-smoothing sweeps to be* $\nu_1 = \nu_2 =: \nu$. *Then one obtains uniform convergence for the V-cycle (*$\gamma = 1$*), i.e.,*

$$\rho\left(M_{MG}^{(\ell_{\max})}\right) \leq 1 - \delta, \tag{2.26}$$

where

$$\delta := \min_{\ell} \min_{\boldsymbol{v}_\ell \in \mathbb{R}^{n_\ell}} \frac{\|\boldsymbol{v}_\ell\|_{\mathrm{A}_\ell}^2 - \left\|M_{\mathscr{S}_\ell}^{\nu} \boldsymbol{v}_\ell\right\|_{\mathrm{A}_\ell}^2}{\left\|M_{C_\ell} \boldsymbol{v}_\ell\right\|_{\mathrm{A}_\ell}^2}. \tag{2.27}$$

Proof. For the proof refer to McCormick [131, Theorem 3.4 and Section 5] or Napov [137, Theorem 3.1]. □

In the corresponding proof of Theorem 2.5.12 one finds δ defined as the minimum over all levels of an expression involving two consecutive levels. A closer look reveals the role of the approximation property from Section 2.5.3.1 in the definition of δ in (2.27). In fact, δ gets maximal, if both $\left\|M_{\mathscr{S}_\ell}^{\nu} \boldsymbol{v}_\ell\right\|_{\mathrm{A}_\ell}^2$ and $\left\|M_{C_\ell} \boldsymbol{v}_\ell\right\|_{\mathrm{A}_\ell}^2$ get small (cf. smoothing and approximation property in Section 2.5.3.1).

In Napov [137] as well as in Napov and Notay [138] the reader can find a quite extensive comparison of the above convergence bound and the classical V-cycle convergence bound given in Hackbusch [87, Theorem 7.2.2]. For a very recent overview of convergence bounds the reader also might refer to MacLachlan and Olson [122].

Remark 2.5.13 (Multigrid convergence for non-symmetric problems). Throughout the whole Section 2.5 A has been assumed to be symmetric positive definite. However, in practice one often has to deal with non-symmetric problems for certain applications. There is quite an abundant literature on geometric multigrid methods for non-symmetric systems specifically targeting fluid dynamics (cf. Brandt and Livne [35], Brandt and Yavneh [36, 40], Thomas et al. [185]). The books by Trottenberg et al. [186] and Wesseling [213] as well as the article Wesseling and Oosterlee [214] contain further references. Away from the more application-specific literature, in Notay [148] the sharp two-grid estimates from Falgout et al. [66] for the symmetric case are generalized for the non-symmetric case. In contrary to geometric multigrid methods, it turns out that the theory of algebraic multigrid methods is much less developed for non-symmetric problems, even though they are often applied to non-symmetric problems in practice. A few notable algebraic multigrid contributions which specifically target non-symmetric systems include Bank et al. [13], Dutto et al. [57], Gravemeier et al. [79], Lallemand et al. [114], Lonsdale [121], Mavriplis and Venkatakfrishnan [129], Mavriplis [130] and Gee et al. [72].

Remark 2.5.14 (Multigrid convergence for aggregation-based multigrid). In this thesis, only aggregation-based AMG methods are used, which are introduced in the next chapter. They can be understood as algebraic multigrid methods with a special technique to build the transfer operators, which is based on so-called aggregates. These specialized techniques often make it difficult to apply standard algebraic multigrid convergence theory, such that the theoretical concepts have to be adapted. In Muresan and Notay [135] an algebraic analysis of aggregation-based multigrid methods is presented for symmetric positive definite systems using (non-smoothed) transfer operators and some special assumptions on the aggregation. A novel aggregation strategy is developed in Napov and Notay [140] with a guaranteed convergence rate for sparse symmetric positive definite linear systems. The analysis of aggregation-based algebraic multigrid methods for non-symmetric convection-diffusion equations is considered in Notay [144] at least for a simplified two-level case. The development of convergence theory for aggregation-based methods and its specializations is still a field of active research. Anyway, it shows good performance in practice and therefore is the method of choice for our problems.

CHAPTER 3

Aggregation-based AMG methods

In Chapter 2 merely the basic idea of multigrid methods has been explained without giving details on how to build the multigrid hierarchies. In the following this gap is closed by discussing the details of the multigrid setup process for the class of *aggregation-based Algebraic Multigrid* methods. Algebraic multigrid methods have the advantage that they do not rely on an underlying hierarchy of (user-generated) meshes and therefore can easily be applied for problems on unstructured meshes. However, for applying multigrid methods in general (and aggregation-based AMG methods in particular), one needs a minimum of background knowledge of the underlying principles, since multigrid methods in general cannot be expected to show satisfactory performance when used in a black-box manner. This chapter can be understood as the basis for the routines which are later extended and adapted for application-specific requirements.

After a brief discussion of differences between classical and aggregation-based AMG methods, a systematic introduction to the overall layout of modern aggregation-based AMG setup routines is presented. To be prepared for the next generation of applications a flexible software framework is absolutely necessary (see also Keyes et al. [103] and Olson et al. [151]). Thus, one main contribution of this work is the design of our new flexible multigrid framework MUELU which is now part of the Trilinos libraries (cf. Heroux et al. [90]). It turns out to be a well-suited tool not only for doing research on new multigrid concepts (e.g., Chapter 4), but also for designing new multigrid preconditioners to tackle today's challenges from modern applications (such as multiphysics problems).

In this thesis the focus is on smoothed aggregation multigrid methods. Therefore, the aggregation procedure is described in detail in Section 3.3, since it has significant influence on the convergence behavior of an aggregation-based multigrid method. Making use of the knowledge about the aggregation one can easily develop application-specific extensions to certain problem classes resulting from computational contact mechanics (see Chapters 6 and 7). The next topic is the construction of appropriate transfer operators P and R. Besides technical details on the construction of the so-called tentative transfer operators, it is shown how to extend them using state-of-the-art smoothing strategies for the transfer operators.

(a) Typical matrix example for 2D diffusion equation using Finite Elements or Finite Differences on a regular mesh with canonical row- or column-wise node ordering.

$$A = \begin{pmatrix} A_{FD} & -I & & & \\ -I & A_{FD} & -I & & \\ & -I & A_{FD} & -I & \\ & & -I & A_{FD} & -I \\ & & & -I & A_{FD} \end{pmatrix}$$

$$\text{with } A_{FD} := \begin{pmatrix} 4 & -1 & & & 0 \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ 0 & & & -1 & 4 \end{pmatrix}$$

(b) Corresponding mesh and matrix stencil. The nodes are represented by the diagonal values of A in Figure 3.1a. The off-diagonal entries define the node-connectivity.

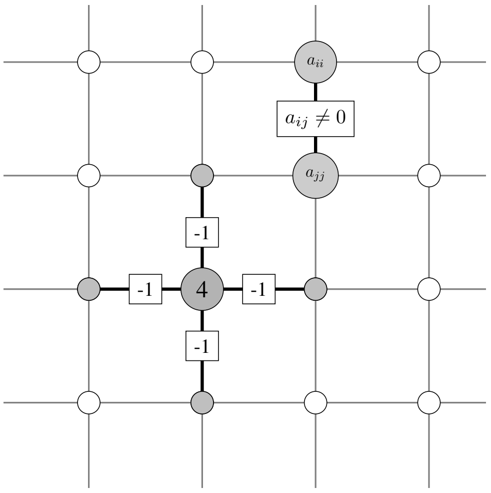


Figure 3.1.: Relation between matrix A and mesh connectivity demonstrated for a typical linear operator resulting from a 2D Finite difference example using a canonical row- or column-wise node ordering of the mesh nodes.

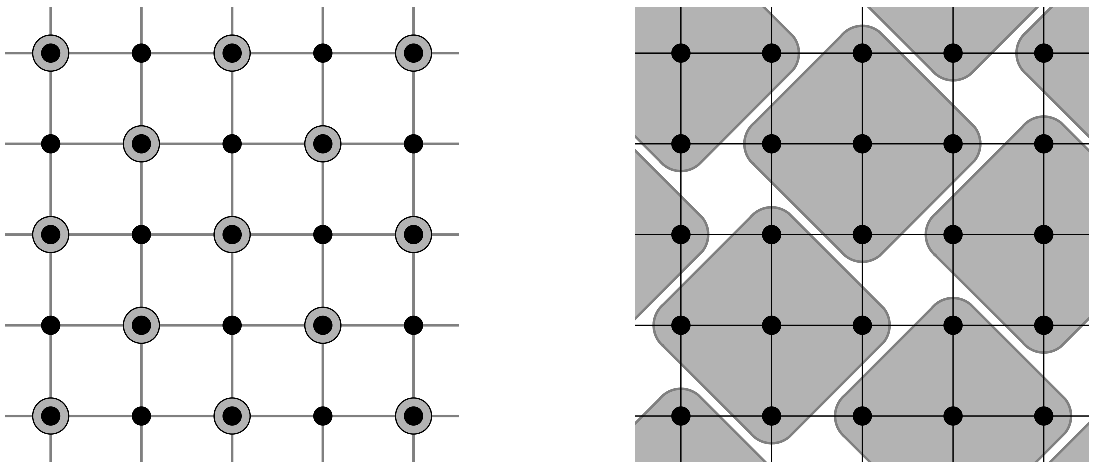
3.1. Algebraic multigrid methods

3.1.1. Motivation for algebraic multigrid methods

The weak point of geometric multigrid methods is that they are based on a user-provided hierarchy of coarse meshes (cf. Section 2.3.1). Especially if complex geometries and unstructured meshes are involved, this turns out to be a problem, since for many engineering applications it is very hard to generate coarse meshes for the given fine level problem. Instead of geometric multigrid principles one can use algebraic multigrid (AMG) methods, which do not rely on a set of (nested) user-provided meshes, but make use of the fine level operator A to coarsen the problem. In fact, the mesh connectivity can be extracted from the graph of the linear operator A by looking at the non-zero off-diagonal entries of the matrix. For demonstration purposes, Figure 3.1 shows the relation of the fine level matrix A (or the graph $G(A)$ of A) and the corresponding mesh connectivity for a 2D diffusion operator resulting form a finite difference discretization on a structured regular mesh with canonical row- or column-wise node ordering.

Algebraic multigrid methods can interpret the information of the fine level operator A to internally reconstruct the corresponding mesh connectivity that is used to build coarse level problems. This also works in case of highly an-isotropic meshes, varying coefficients in the underlying equations, and problems that are dominated by convective phenomena.

For a general introduction to algebraic multigrid methods the reader is referred to Ruge and Stüben [167] and Trottenberg et al. [186].


(a) Classical coarsening in Ruge Stüben algebraic multigrid methods. The small black nodes denote nodes on the fine level. The large gray nodes declare nodes which have been selected by the coarsening routine for the coarse level.

(b) Coarsening using an aggregation-based algebraic multigrid method. The black nodes represent fine level nodes. The gray color defines the aggregates. Each aggregate corresponds to one coarse level "node" on the coarse level.

Figure 3.2.: Conceptual difference of classical Ruge Stüben type algebraic multigrid methods and aggregation-based algebraic multigrid methods shown for a typical linear operator resulting from a 2D finite difference example using a canonical row- or columnwise node ordering of the mesh nodes.

3.1.2. Classification of algebraic multigrid methods

One has to distinguish standard AMG methods that are also known as classical Ruge–Stüben AMG methods (cf. Ruge and Stüben [167]), and so-called aggregation-based AMG methods (cf. Vaněk et al. [192]).

Classical Ruge Stüben AMG methods are based on a coarsening process which first selects a subset of coarse level nodes from all fine level nodes. The algebraic coarsening routines are mainly based on the properties of M-matrices to construct transfer operators P and R. These traditional approaches are known to be effective for a wide range of problems (cf. Brandt [38], Trottenberg et al. [186]). Common coarse variable selection routines are based on minimal independent sets (cf. Cleary et al. [49], Stüben [183]) using algebraic information from the underlying problem only. Once a set of coarse level nodes is found, the method defines interpolation operators for the transfers between the fine and coarse levels. However, considering application-specific information in the design of interpolation operators, such as rigid body modes for elasticity problems, comes with some extra complexities (cf. Baker et al. [11]).

More general approaches for selecting the set of coarse variables use the concept of compatible relaxation (an idea first introduced by Brandt [39]) to gauge the quality of the coarse variable set. An overview on how to generalize the AMG framework is given in Falgout and Vassilevski [64], including general smoothing and coarsening processes and several compatible relaxation methods. In Brannick and Zikatanov [41] the authors present an adaptive AMG setup algorithm that uses compatible relaxation to optimize the set of coarse variables. The nonzero supports for the coarse space basis are determined by approximation of the so-called two-level "ideal" interpolation operator. In this thesis similar ideas are used to improve multigrid transfer operators in a different context (see Section 4.4). The coarsening algorithm used in Brannick and Zikatanov [41] to construct the coarse variable set is described in Brannick and Falgout [42] based on the compatible relaxation algorithm in Livne [120].

In Figure 3.2a the general concept of classical coarsening is visualized. The small nodes in black color denote fine level nodes, whereas the large gray nodes have been selected to be coarse level nodes. The Figure 3.2a just shows exemplarily how the coarse variables could be chosen from the fine level variables for obtaining a coarsening rate 2. The transfer operators are built using some interpolation techniques, such that the fine level nodes are interpolated by information from the neighboring coarse level variables.

Remark 3.1.1 (Implementations & Software). Current implementations of Ruge Stüben AMG methods are available in PyAMG (cf. Bell et al. [20]) and HYPRE (cf. Falgout and Yang [65]).

In contrary to classical Ruge Stüben AMG methods, the class of aggregation-based AMG methods (cf. Vaněk et al. [192]) replaces the concept of fine and coarse level nodes by so-called aggregates. That is, aggregation-based AMG methods just use a different technical concept to define the coarse level problems and the transfer operators. Aggregates are defined by agglomerating fine level nodes using algebraic information about the node connectivity only, which is represented by the graph of the matrix A. Transfer operators are built directly using the set of aggregates. Near null space modes (such as rigid body modes in elasticity, cf. Section 3.4) are automatically considered during construction of the transfer operators per design. In order to obtain optimal convergence properties, the transfer operators can be smoothed in a separate smoothing step (see Smoothed Aggregation methods in Section 3.5).

Figure 3.2b shows how the black fine level nodes are aggregated following the mesh connectivity of the Finite Difference Laplace example visualized by the solid black lines. All aggregates in gray shape represent one coarse level variable on the next coarser level. That is, the 5-point aggregates shown in Figure 3.2b would lead to a coarsening rate of 5.

Remark 3.1.2 (Implementations & Software). Aggregation-based AMG methods are available in PyAMG (cf. Bell et al. [20]) as well as the software packages ML (cf. Gee et al. [73]) and MueLu in the Trilinos project (cf. Heroux et al. [90]). A commercial implementation of an aggregation-based AMG method is available with AGMG (cf. Notay [147]).

3.2. General objectives and design of aggregation-based AMG

In this section the algorithmic design as well as more general objectives for an aggregation-based AMG method are discussed. Whereas many aspects (such as aggregation algorithms, prolongator smoothing etc.) can be found in the according literature, the interplay of the different algorithmic parts is not or only very briefly discussed in the papers (see, e.g., Blaheta [25]). However, when applying or developing new multigrid techniques, it is highly important to have a sufficient understanding of the underlying processes. Therefore, the design of an aggregation-based AMG method is explained from top to bottom starting with the overall overview of the different algorithmic ingredients before proceeding with concrete details in the next sections.

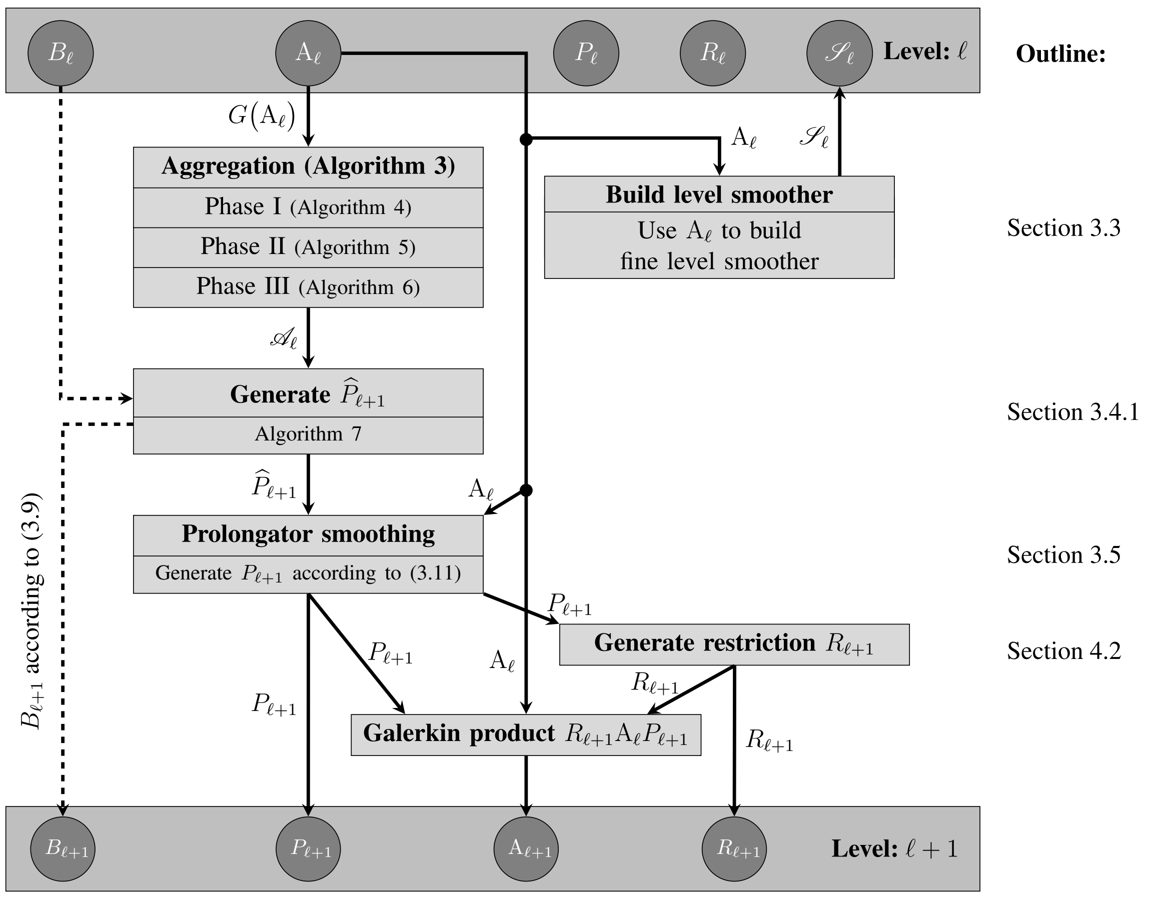


Figure 3.3.: Setup phase for a Smoothed Aggregation AMG multigrid method.

3.2.1. Algorithmic design of aggregation-based AMG

An algebraic multigrid method usually consists of a *setup phase*, where the coarsening algorithms build the coarse level problems with the corresponding transfer operators, and a *solution* or *iteration phase*, where the usual multigrid cycles, as described in Section 2.4.2, are performed using the multigrid hierarchy built during the setup phase.

3.2.1.1. Setup phase

Figure 3.3 is meant to guide through the setup process for multigrid transfer operators in a smoothed aggregation multigrid method providing a global overview of the different steps. Furthermore, it also serves as outline for the next sections in this chapter, which explain the different phases during the multigrid setup. Note that the steps in Figure 3.3 are performed iteratively starting with $\ell = 0$ on the finest level until the maximum allowed number of multigrid levels $\ell_{\max}$ is reached or the coarse level size is below a certain user-prescribed bound.

Before discussing the details in the next sections, a brief review of the overall outline of the setup phase is given in Figure 3.3. On the fine level one needs the fine level matrix A_ℓ as input together with an approximation of the fine level (near) null space $B_\ell \in \mathbb{R}^{n_\ell \times n_B}$, which satisfies $A_\ell B_\ell \approx 0$ at least away from the Dirichlet boundaries (cf. Section 3.4.1). Then, the transfer operators $P_{\ell+1}$ and $R_{\ell+1}$ can be built together with the coarse level near null space $B_{\ell+1}$ and the coarse level operator $A_{\ell+1}$. The graph $G(A_\ell)$ of the fine level matrix A_ℓ is used as input for the aggregation algorithm, which provides a set of aggregates $\mathscr{A}_\ell$ (cf. Section 3.3). These can be used to define a tentative prolongation operator $\widehat{P}_{\ell+1}$ with non-smoothed piece-wise constant coarse level basis functions (cf. Section 3.4.1). The tentative prolongation operator $\widehat{P}_{\ell+1}$ also indirectly defines the coarse level near null space vectors $B_{\ell+1}$ (see equation (3.9) in Section 3.4.1). In order to improve the transfer operators, one can optionally apply a smoothing sweep for the non-smoothed transfer operator basis functions in the so-called prolongation smoothing step (cf. Section 3.5). The new prolongation operator $P_{\ell+1}$ is used as input for generating a corresponding restriction operator $R_{\ell+1}$. The most simple choice to define a restriction operator is just to use $R_{\ell+1} = P_{\ell+1}^{\mathsf{T}}$. More advanced methods that are also appropriate for non-symmetric linear systems are discussed in Section 4.2.2. Once the transfer operators $P_{\ell+1}$ and $R_{\ell+1}$ are built, the Galerkin product (2.6) can be calculated to obtain the coarse level matrix $\mathrm{A}_{\ell+1}$. Note that the transfer operators P_ℓ and R_ℓ on the fine level ℓ are not only needed to generate the coarse level problem, but also to be used in the solution phase (see Section 3.2.1.2). On each multigrid level the corresponding set of level smoothers for pre- and post-smoothing during the iteration phase is created using A_ℓ as input.

3.2.1.2. Solution phase

In the solution phase the multigrid hierarchy built during the setup phase is just used within the multigrid algorithm as described by Algorithm 2 in Section 2.4.2 for iteratively reducing the different error components on the corresponding multigrid levels. One can use the multigrid method as solver and perform several multigrid sweeps with the V-cycle to solve the linear problem. Alternatively, you can use the multigrid method as preconditioner within an outer Krylov subspace solver such as a CG method or GMRES and perform one multigrid sweep for preconditioning. If not otherwise stated, always 1 sweep through the V-cycle from Algorithm 2 with $\gamma = 1$ is used as preconditioner for a GMRES solver (cf. Appendix A).

3.2.2. Objectives and requirements for the prolongation operators

The selection of the prolongation operators for the multigrid method is dictated by the desire to achieve good convergence behavior with reasonable computational complexity of the algorithms. Here, the desired properties of our prolongation operator are specified in terms of the support and shape of the coarse basis functions (cf. Mandel et al. [124], Vaněk et al. [192, 197]).

(O1) *Strong couplings:* The support of the coarse basis functions should follow strong couplings in the underlying matrix graph. Two degrees of freedom i and j in a matrix $\mathrm{A}_\ell = \left(a_{ij}\right)_{i,j=1}^{n_\ell}$ on level ℓ are said to be strongly coupled if $|a_{ij}|$ is relatively large compared with $\sqrt{|a_{ii}a_{jj}|}$. Note that the underlying physics is reflected in the coefficients of the matrix A_ℓ, e.g., in the sense that neighboring nodes are strongly coupled in the direction of anisotropy.

(O2) *Limited overlap:* It is assumed that a constant K exists, such that for any coarse basis function its support intersects at most K supports of other coarse basis functions. The constant K is supposed to be small to guarantee sparsity of the resulting coarse level matrices $\mathrm{A}_{\ell+1}$. This is important as a prolongation smoothing method (see Section 3.5) comes along with the costs of increasing the supports of the basis functions (cf. Gee et al. [71], Mandel et al. [124]).

A common measure for the sparsity (and the memory footprint) of a multigrid method is given by the multigrid operator complexity (see, e.g., Gee et al. [71]).

Definition 3.2.1 (Operator complexity)**.** The multigrid *operator complexity* is defined as

$$OC = \frac{\sum_{\ell=0}^{\ell_{\max}-1} \#\mathrm{nnz}(\mathrm{A}_\ell)}{\#\mathrm{nnz}(\mathrm{A}_0)} \tag{3.1}$$

and describes the ratio of non-zero entries of all multigrid level matrices A_ℓ with $\ell = 0, \dots, \ell_{max} - 1$ and the number of non-zeros on the finest level.

In practice, it is important to keep track of the operator complexity to make sure that the memory consumption does not exceed hard memory limits.

(O3) *Preservation of (near) null space modes:* The span of the basis functions on the coarse level should contain (near) null space modes, at least away from Dirichlet boundaries (cf. Mandel et al. [124], Mandel [125], Vaněk et al. [192, 197]). Near null space modes correlate with (algebraically) smooth error components which are meant to be treated on the coarser levels where relaxation-based methods can reduce the smooth error modes more effectively (cf. Section 2.5.2).

(O4) *Minimization of energy:* The basis functions on the coarse levels should have as small energy as possible (at least in cases where a notion of energy is defined). This objective is motivated by the observation that minimizing the energy of the basis functions contracts the condition number of the coarse level matrices. For more details the reader may refer to Mandel et al. [124], Olson et al. [153] and the comments in Section 3.5.2.

3.3. Aggregation algorithm

The aggregation is one of the central key components within an aggregation-based AMG method. For our examples in this thesis a general graph-based aggregation method is used that is described in detail in the following sections. It can be understood as an extended variant of the basic aggregation algorithm described in Vaněk et al. [197, Algorithm 2] or Vaněk et al. [192].

It shall be mentioned that for special applications some basic knowledge about the underlying discretization of the (partial) differential equations may be very helpful to define an appropriate aggregation strategy. One can think of many options for highly problem-specific aggregation strategies designed for different applications and problems. In the work by Olson and Schroder [150] a conforming aggregation for high-order discontinuous finite elements with distance-based aggregation is introduced. Furthermore, Napov and Notay [140] describe a novel aggregation strategy with a guaranteed convergence rate for symmetric positive definite systems, and in Olson et al. [152] the authors review different so-called strength-of-connection concepts for classical and smoothed aggregation AMG methods.

3.3.1. Notation

Primarily interested in non-scalar problems, such as 2D or 3D elasticity, one has more than 1 degree of freedom for each node on the underlying fine-level mesh. It is assumed that all degrees of freedom are associated with the nodes of the mesh and that the number of degrees of freedom per node is constant. These assumptions allow a very efficient implementation, but one can state that they could also be extended to the more general case of a variable number of degrees of freedom per node or mixed element formulations. Neither of this is necessary for the applications in this thesis. The aggregation algorithm is based on the node graph of the matrix representing the mesh on the finest level. Let m_0 denote the number of nodes on the finest level ($\ell = 0$) and n_0 the corresponding number of degrees of freedoms. Then, one can define a surjective mapping between the degrees of freedom and the corresponding nodes:

Definition 3.3.1 (Mapping between degrees of freedom and corresponding nodes)**.** With each node and degree of freedom numbered consecutively by $1, \dots, m_0$ and $1, \dots, n_0$ one can define $\mathrm{n} : \{1, \dots, n_\ell\} \to \{1, \dots, m_\ell\}$ as the mapping of the degrees of freedom to the associated node id's.

Here, the graph of the matrix A_ℓ is denoted as $G(\mathrm{A}_\ell) := (g_{ij})_\ell$, where the entries g_{ij} are given by

$$(g_{ij})_\ell = \begin{cases} 1 & \text{if } \exists\, k, l \in \{1, \dots, n_\ell\} : a_{kl} \neq 0 \text{ with } \mathrm{n}(k) = i \text{ and } \mathrm{n}(l) = j, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

with $i, j = 1, \dots, m_\ell$. For any node $i \in \{1, \dots, m_\ell\}$ the neighborhood of nodes Nb_ℓ is defined as the set

$$\mathrm{Nb}_\ell(i) = \Big\{ j \in \{1, \dots, m_\ell\} : (g_{ij})_\ell = 1 \Big\}. \tag{3.3}$$

Remark 3.3.2 (Node). The term *node* is geometrically clearly defined in context of a mesh on the finest level. However, there is no mesh or geometry associated with the coarse levels in the AMG setting. In this thesis, the term *node* is used to describe a strictly algebraic entity consisting of a list of degrees of freedom which allows us to use the term *node* also for coarser levels. Each aggregate which agglomerates some nodes on the fine level ℓ defines one node on the coarse level $\ell + 1$, and each degree of freedom associated with that coarse level node is a coefficient of a particular basis function in the coarse-level basis expansion defined by the transfer operators (cf. Section 2.3.2).

3.3.2. Matrix filtering

Especially for problems with some inherent anisotropy one often uses the graph of a filtered fine level operator $\mathrm{A}_\ell^{\mathrm{fl}}$ instead of applying the aggregation algorithm to the graph of the full matrix A_ℓ (cf. Gee et al. [71], Vaněk et al. [192]). The filtered fine level operator $\mathrm{A}_\ell^{\mathrm{fl}} = (a_{ij}^{\mathrm{fl}})$ is given by

$$(a_{ij}^{\mathrm{fl}})_\ell = \begin{cases} a_{ij} & \text{if } |a_{ij}| \geq \varepsilon \text{ for } i \neq j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad (a_{ii}^{\mathrm{fl}})_\ell = a_{ii} + \sum_{\substack{j=1 \\ i \neq j}}^{n_\ell} (a_{ij} - a_{ij}^{\mathrm{fl}}) \tag{3.4}$$

for some $\varepsilon > 0$. Dropping small entries highlights the strong couplings within the matrix, which is important in order to meet the objective **(O1)** for the prolongation operator (cf. Section 3.2.2). The special treatment of the diagonal entries in (3.4) makes sure that the sum of entries in a row of the filtered matrix $\mathrm{A}_\ell^{\mathrm{fl}}$ is zero whenever the sum of the entries in a row of A_ℓ is zero. When applying a common dropping strategy, the resulting graph of the filtered matrix $\mathrm{A}_\ell^{\mathrm{fl}}$ can be expressed by $G(\mathrm{A}_\ell^{\text{fl}}) := G^{\text{fl}}(\mathrm{A}_\ell) = (g_{ij}^{\text{fl}})_\ell$ with

$$(g_{ij}^{\text{fl}})_\ell = \begin{cases} 1 & \text{if } \exists\, k,l \in \{1,\dots,n_\ell\} : |a_{kl}| > \varepsilon \text{ with } \mathrm{n}(k) = i \text{ and } \mathrm{n}(l) = j \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

with $i,j = 1,\dots,m_\ell$ and a properly chosen dropping parameter $\varepsilon > 0$. The aggregation algorithm then can directly operate on the filtered graph $G^{\text{fl}}(\mathrm{A}_\ell)$.

In the following, the superscript for the filtered graph is dropped. Consequently, the filtered and the non-filtered graph is denoted by $G(\mathrm{A}_\ell)$. If not stated otherwise, it is $\varepsilon = 0$, i.e., the non-filtered graph is used.

Remark 3.3.3 (Dirichlet boundary conditions). Since condensing out Dirichlet boundary conditions from the linear systems would be too expensive, one often keeps them in the linear system. Of course, the matrix rows describing the Dirichlet boundary conditions affect the aggregation process. Most often, these Dirichlet conditions are not a major issue for the multigrid method, especially if multigrid is used as preconditioner within an iterative solver. However, in some special cases Dirichlet information is not correctly processed, when using a multigrid algorithm. Then it is necessary to put some more attention in the handling of boundary conditions. In Appendix C the reader finds a detailed discussion on the topic of Dirichlet boundary conditions in the context of AMG methods.

3.3.3. Aggregation algorithm

The aggregation algorithm is used to build a set of node aggregates $\{\mathscr{A}_\ell^{(i)}\}_{i=1}^{m_{\mathscr{A}_\ell}}$ forming a disjoint covering of the index set of all nodes on the current level ℓ, such that

$$\bigcup_{i=1}^{m_{\mathscr{A}_\ell}} \mathscr{A}_\ell^{(i)} = \{1,\dots,m_\ell\}, \tag{3.6}$$

where m_ℓ denotes the number of nodes and $m_{\mathscr{A}_\ell}$ stands for the number of aggregates on level ℓ. Note that the number of aggregates $m_{\mathscr{A}_\ell}$ on level ℓ naturally defines the number of nodes $m_{\ell+1}$ on the next coarser level, i.e., $m_{\ell+1} = \operatorname{card}\Big(\{\mathscr{A}_\ell^{(i)}\}_{i=1}^{m_{\mathscr{A}_\ell}}\Big) = m_{\mathscr{A}_\ell}$ (cf. Remark 3.3.2).

Algorithm 3 gives the outline of the aggregation process used in this thesis, which is based on the aggregation algorithm described in Vaněk et al. [197, Algorithm 2].

The user can specify the minimum and maximum size of the aggregates by the user parameters m_{min} and m_{max}, respectively. The parameter m_{maxngh} is meant for advanced users and basically denotes how many neighbor nodes at maximum are allowed to be already aggregated in different neighboring aggregates. Its effect will be discussed in detail in Section 3.3.4. In the following, algorithmic details are given which are very helpful to understand the behavior of the aggregation routines. In Remark 3.3.7 one can find some further general comments on proper choices for the user parameters which may be helpful to find a working and efficient set of parameters for the multigrid aggregation routines.

As one can see from Algorithm 3, first the set of all non-aggregated nodes R is initialized. The empty sets C and $\mathscr{A}_\ell$ are declared which are needed for storing the aggregated node ids and the aggregates built during the aggregation process. The aggregation itself is performed in three consecutive phases which are explained in more detail in the following.

Algorithm 3: Aggregation algorithm.

```
Procedure Aggregate($m_{\text{min}}$, $m_{\text{max}}$, $m_{\text{maxngh}}$, $G(\mathrm{A}_\ell)$)
    Initialization
    $\mathrm{R}_\ell \leftarrow \{1, \ldots, m_\ell\}$    // Set of remaining non-aggregated nodes
    $\mathrm{C}_\ell \leftarrow \emptyset$              // Set of aggregated nodes
    $\mathscr{A}_\ell \leftarrow \emptyset$              // Set of aggregates

    Phase I: Tentative aggregation
    $(\mathscr{A}_\ell, \mathrm{C}_\ell, \mathrm{R}_\ell) \leftarrow$ PhaseI($\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$, $G(\mathrm{A}_\ell)$, $m_{\text{min}}$, $m_{\text{max}}$, $m_{\text{maxngh}}$)
    Phase II: Enlarging the tentative aggregates
    $(\mathscr{A}_\ell, \mathrm{C}_\ell, \mathrm{R}_\ell) \leftarrow$ PhaseII($\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$, $G(\mathrm{A}_\ell)$, $m_{\text{max}}$)
    Phase III: Aggregation of left-over nodes
    $(\mathscr{A}_\ell, \mathrm{C}_\ell, \mathrm{R}_\ell) \leftarrow$ PhaseIII($\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$, $G(\mathrm{A}_\ell)$)
    Return aggregates
    return $\mathscr{A}_\ell$
```

3.3.4. Phase I: Tentative aggregation

In the first aggregation phase one tries to select disjoint strongly coupled neighborhoods to build a set of tentative aggregates as an initial tentative covering of all fine level nodes m_0. Algorithm 4 basically loops through all non-aggregated nodes from $\widehat{\mathrm{R}}_\ell$ and tries to build new aggregates. The ordering of the non-aggregated nodes defines the ordering of how the Phase I aggregates are built, i.e., it considerably affects the size, the shape and placement of the Phase I aggregates. One can either use a random ordering, the natural ordering (defined by the node ordering of the underlying mesh) or a graph ordering which is following the node connectivity in the graph.

Besides the ordering of the non-aggregated nodes, the user has some more control over the Phase I aggregation process through the user parameters. Algorithm 4 is a slightly extended version of the first step in Vanek's aggregation algorithm (cf. Vaněk et al. [197, Algorithm 2]), where only aggregates are accepted which contain at least m_{min} nodes. This way one can declare a lower bound for the coarsening rate of the multigrid algorithm. The upper limit m_{max} of the aggregate size gives the user more control of the coarsening rate and helps to create aggregates of similar size throughout the whole domain. With m_{maxngh} the user can choose an upper bound for the number of neighbor nodes which are allowed to be already aggregated in a different existing aggregate. This parameter is meant for advanced users only and provides some control over the density of the tentative aggregates. The recommended choice $m_{\text{maxngh}} = 0$, for example, guarantees all tentative aggregates from Phase I to have a minimum graph distance of 2. Choosing $m_{\text{maxngh}} > 0$ may lead to tentative aggregates which are more closely together. However, this is a simplified and rough explanation of the effect of this parameter. In practice, the connections and the ordering of the nodes in the graph $G(\mathrm{A}_\ell)$ dominate the placement and shape of the tentative aggregation phase.

Example 3.3.4. Let's have a look at a simplified example to get a better understanding of the Phase I aggregation process from Algorithm 4. A regular 8×8 mesh is considered with strong connections only in horizontal and vertical, but not in diagonal directions (e.g., resulting from

Algorithm 4: Aggregation algorithm: Phase I

```
Procedure PhaseI ($\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$, $G(\mathrm{A}_\ell)$, $m_{\text{min}}$, $m_{\text{max}}$, $m_{\text{maxngh}}$)
    Create local copy of remaining nodes in $\mathrm{R}$
    Set $\widehat{\mathrm{R}}_\ell \leftarrow \mathrm{R}_\ell$

    Number of aggregates built in Phase I
    $k \leftarrow 0$

    Check all remaining nodes whether they can be aggregated
    while $\widehat{\mathrm{R}}_\ell \neq \emptyset$ do
        Select a new id from the non-aggregated remaining nodes
        (Use natural node ordering, graph based ordering or random ordering).
        Select node $i \in \widehat{\mathrm{R}}$

        Define empty set of node id's for a new tentative aggregate
        $\mathscr{A}_\ell^{(\text{tent})} \leftarrow \emptyset$

        Loop over all neighboring nodes $j$ of the selected node $i$
        for $j \in \mathrm{Nb}_\ell(i)$ do
            If node $j$ is not aggregated and size of tentative aggregate is not too big...
            if $(j \notin \mathrm{C}_\ell)$ & $(\#\mathscr{A}_\ell^{(\text{tent})} < m_{\text{max}})$ then
                ...add node $j$ to tentative aggregate
                $\mathscr{A}_\ell^{(\text{tent})} \leftarrow \mathscr{A}_\ell^{(\text{tent})} \cup \{j\}$
            end
        end

        Check if more than $m_{\text{min}}$ nodes are contained in tentative aggregate
        if $\left(\#\mathscr{A}_\ell^{(\text{tent})} > m_{\text{min}}\right)$ & $\left(\#(\mathrm{C}_\ell \cap \mathrm{Nb}_\ell(i)) \leq m_{\text{maxngh}}\right)$ then
            Accept tentative aggregate
            $k \leftarrow k + 1$
            $\mathscr{A}_\ell^{(k)} \leftarrow \mathscr{A}_\ell^{(\text{tent})}$

            Update node sets
            $\mathrm{C}_\ell \leftarrow \mathrm{C}_\ell \cup \mathscr{A}_\ell^{(\text{tent})}$
            $\widehat{\mathrm{R}}_\ell \leftarrow \widehat{\mathrm{R}}_\ell \setminus \mathscr{A}_\ell^{(\text{tent})}$
        else
            Discard tentative aggregate. Remove node $i$ from local list of remaining nodes
            $\widehat{\mathrm{R}}_\ell \leftarrow \widehat{\mathrm{R}}_\ell \setminus \{i\}$
        end
    end

    Update list of remaining nodes that are not in an aggregate yet
    $\mathrm{R}_\ell \leftarrow \mathrm{R}_\ell \setminus \mathrm{C}_\ell$

    Return aggregates and node sets of aggregated and non-aggregated node id's
    return $\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$
```

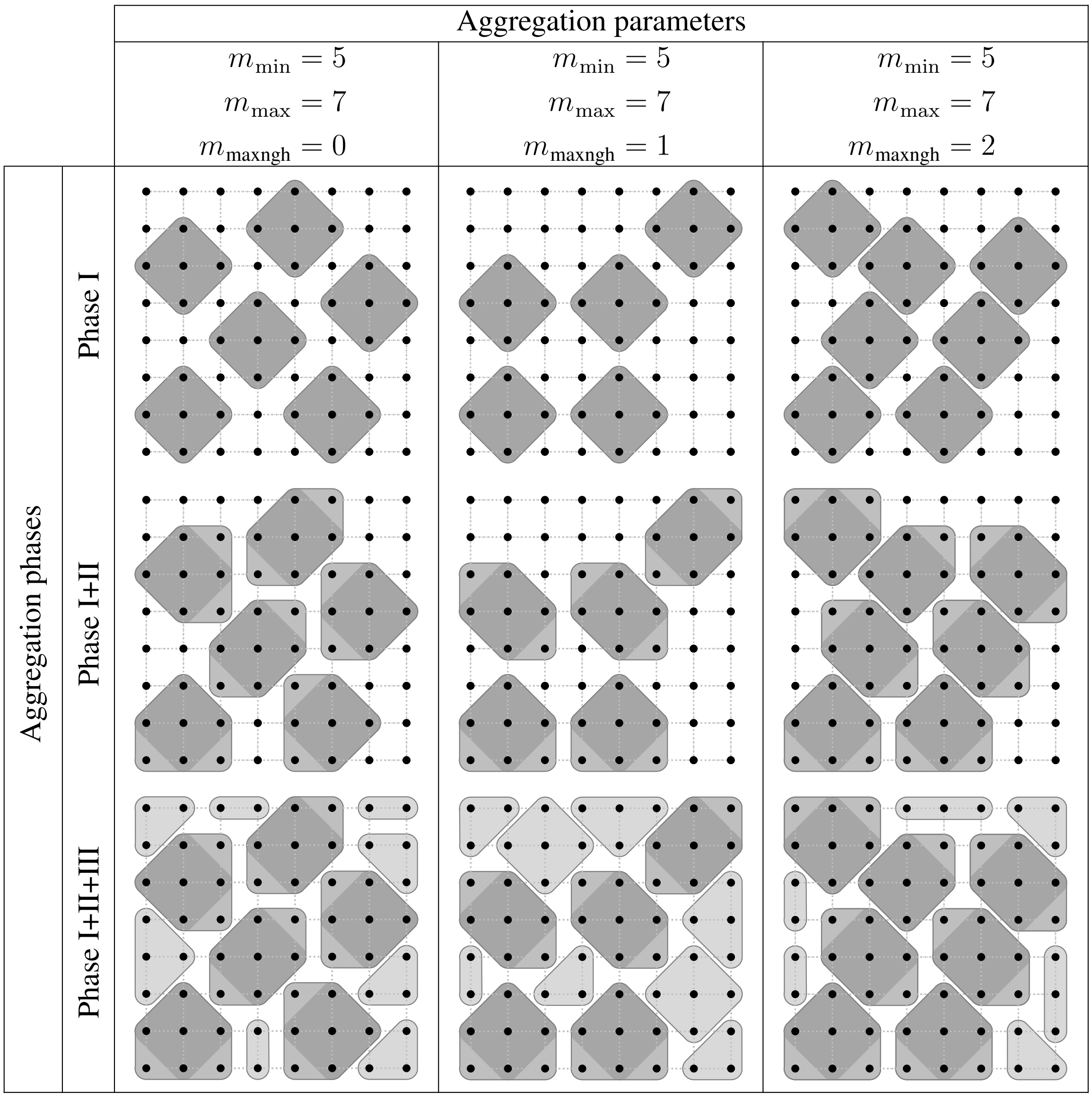

Aggregation parameters
$m_{\text{min}} = 5$ $m_{\text{max}} = 7$ $m_{\text{maxngh}} = 0$
$m_{\text{min}} = 5$ $m_{\text{max}} = 7$ $m_{\text{maxngh}} = 1$
$m_{\text{min}} = 5$ $m_{\text{max}} = 7$ $m_{\text{maxngh}} = 2$
Aggregation phases
Phase I
Phase I+II
Phase I+II+III

Figure 3.4.: Effect of aggregation algorithm parameters. The tentative aggregates chosen in Phase I (cf. Algorithm 4) are colored in dark gray, the extended aggregates from Phase II (cf. Algorithm 5) are colored in intermediate gray color and the aggregates in light gray have been added by Phase III (cf. Algorithm 6). The non-aggregated nodes in $\widehat{\mathrm{R}}_0$ from Algorithm 4 are ordered randomly.

a 2D finite difference discretization). This assumption is just meant to demonstrate the effect of different aggregation parameters.

In the first row of Figure 3.4 the effect of different aggregation parameters is shown for the Phase I aggregation process. The parameter $m_{\min}$ is chosen as $m_{\min} = 5$ which is the maximal reasonable value, since each node has (at maximum) 4 possible neighbor nodes as shown by the dotted lines. Choosing $m_{\min} > 5$ would make the aggregation Phase I algorithm to finish without generating tentative aggregates. The parameter $m_{\max}$ allows to control the maximum size of the aggregates. In the tentative aggregation phase it is usually less important and only affects the Phase I aggregation if the number of actual neighbor nodes exceeds the user-given limit in $m_{\max}$. In contrary to $m_{\min}$, the $m_{\max}$ parameter is a sharp bound, i.e., even in the later aggregation phases the aggregation algorithm will never hurt the maximum allowed size of aggregates. Figure 3.4 also demonstrates the effect of m_{maxngh} limiting the number of nodes in the direct neighborhood that are aggregated in different aggregates. However, due to the somewhat limited mesh connectivity, a higher value for this parameter does not automatically result in more tentative aggregates. In fact, the actual placement is highly dependent on the ordering of the remaining non-aggregated nodes $\widehat{\mathrm{R}}_\ell$ in Algorithm 4 and the connectivity of the graph.

3.3.5. Phase II: Enlarging the tentative aggregates

The Phase II of the aggregation process tries to add the left-over nodes from Phase I to existing neighboring aggregates. In Algorithm 5, all aggregates in the neighborhood for each left-over node i are determined. For each neighboring aggregate, one counts the internal nodes which have a strong connection to the left-over node i. Then the aggregate with the highest count is selected, which does not exceed the maximum allowed size of $m_{\max}$ nodes for an aggregate after the left-over node i has been added. It is important to understand that the Phase II algorithm can only add left-over nodes with a maximum graph distance of 1 to existing aggregates. That is, there might be some left-overs after Phase II which have a graph distance > 1 to the next aggregate. This usually happens close to the boundaries where no Phase I aggregates have been placed (cf. second row in Figure 3.4 for the Example 3.3.4). However, this is not a real problem, as with each node the aggregate is growing and allows for more neighbor nodes to be added. Therefore, similar to Phase I, the ordering of the remaining nodes $\widehat{\mathrm{R}}_\ell$ has significant influence on which nodes are chosen to be added to aggregates. The only limiting factor is the parameter $m_{\max}$ for the maximum size of the aggregate. Needless to say that the Phase II algorithm can be skipped if $m_{\min} = m_{\max}$.

Remark 3.3.5 (Alternative selection criteria). The maximum link criterion as described by Algorithm 5 is only one possible method to decide which aggregate the left-over node i should be added to. Another reasonable criterion would be to choose the neighboring aggregate which has minimum size. This leads to more balanced aggregate sizes. A similar effect could also be achieved by a reasonable choice of the minimum and maximum size parameters for the aggregates.

3.3.6. Phase III: Aggregation of left-over nodes

The primary goal of the aggregation is to make sure that all m_ℓ fine level nodes are aggregated after Phase III. Therefore, a new aggregate is built for each remaining left-over node i which could not be aggregated during Phase I or Phase II (cf. Figure 3.4 for the Example 3.3.4). All

Algorithm 5: Aggregation algorithm: Phase II.

```
Procedure PhaseII ($\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$, $G(\mathrm{A}_\ell)$, $m_{\max}$)
    Create local copy of remaining nodes in $\mathrm{R}$
    Set $\widehat{\mathrm{R}}_\ell \leftarrow \mathrm{R}_\ell$

    Check all remaining nodes whether they can be aggregated
    while $\widehat{\mathrm{R}}_\ell \neq \emptyset$ do
        Select a new id from the non-aggregated remaining nodes
        Select node $i \in \widehat{\mathrm{R}}$

        Create empty list of aggregate id's which are in the neighborhood of node $i$
        ${\mathscr{A}_\ell}^{(\mathrm{ngh})} \leftarrow \emptyset$

        Loop over all neighboring nodes $j$ of node $i$
        for $j \in \mathrm{Nb}_\ell(i)$ do
            Collect aggregate id's of neighboring nodes $j$ around node $i$
            for $k \in \mathscr{A}_\ell$ do
                if $j \in {\mathscr{A}_\ell}^{(k)}$ then
                    ${\mathscr{A}_\ell}^{(\mathrm{ngh})} \leftarrow {\mathscr{A}_\ell}^{(\mathrm{ngh})} \cup \{k\}$
                end
            end
        end

        Check whether node $i$ can be added to a neighboring aggregate
        while ${\mathscr{A}_\ell}^{(\mathrm{ngh})} \neq \emptyset$ do
            Find the aggregate id with most connections to node $i$
            $k_{\max} \leftarrow \underset{k \in \mathscr{A}_\ell}{\arg\max}\Big(\#\big({\mathscr{A}_\ell}^{(k)} \cap {\mathscr{A}_\ell}^{(\mathrm{ngh})}\big)\Big)$

            if $\#{\mathscr{A}_\ell}^{(k_{\max})} < m_{\max}$ then
                Add node $i$ to aggregate $k_{\max}$
                ${\mathscr{A}_\ell}^{(k_{\max})} \leftarrow {\mathscr{A}_\ell}^{(k_{\max})} \cup \{i\}$
                ${\mathscr{A}_\ell}^{(\mathrm{ngh})} \leftarrow \emptyset$

                Update node sets
                $\mathrm{R}_\ell \leftarrow \mathrm{R}_\ell \setminus \{i\}$
                $\widehat{\mathrm{R}}_\ell \leftarrow \widehat{\mathrm{R}}_\ell \setminus \{i\}$
                $\mathrm{C}_\ell \leftarrow \mathrm{C}_\ell \cup \{i\}$
            else
                Try another aggregate
                ${\mathscr{A}_\ell}^{(\mathrm{ngh})} \leftarrow {\mathscr{A}_\ell}^{(\mathrm{ngh})} \setminus \{k_{\max}\}$
                $\widehat{\mathrm{R}}_\ell \leftarrow \widehat{\mathrm{R}}_\ell \setminus \{i\}$

            end
        end
    end

    Return aggregates and node sets of aggregated and non-aggregated node id's
    return $\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$
```

its non-aggregated neighbor nodes are added to the new aggregate ignoring the user parameters m_{min} and m_{maxngh}.

Algorithm 6: Aggregation algorithm: Phase III

```
Procedure PhaseIII($\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$, $G(\mathrm{A}_\ell)$)
    Number of aggregates
    $k \leftarrow \#(\mathscr{A}_\ell)$
    Check all remaining nodes whether they can be aggregated
    for $i \in \mathrm{R}_\ell$ do
        Create a new aggregate for the left over node $i$
        $k \leftarrow k + 1$
        $\mathscr{A}_\ell^{(k)} \leftarrow \{i\}$
        Loop over all nodes $j$ in the neighborhood of the selected node $i$
        for $j \in \mathrm{Nb}_\ell(i)$ do
            If node $j$ is not aggregated, add it to the new aggregate
            if $j \notin \mathrm{C}_\ell$ then
                $\mathscr{A}_\ell^{(k)} \leftarrow \mathscr{A}_\ell^{(k)} \cup \{j\}$
            end
        end
        Update node sets
        $\mathrm{R}_\ell \leftarrow \mathrm{R}_\ell \setminus \mathscr{A}_\ell^{(k)}$
        $\mathrm{C}_\ell \leftarrow \mathrm{C}_\ell \cup \mathscr{A}_\ell^{(k)}$
    end
    Return aggregates and node sets of aggregated and non-aggregated node id's
    return $\mathscr{A}_\ell$, $\mathrm{C}_\ell$, $\mathrm{R}_\ell$
```

Remark 3.3.6 (Motivation). When using a multigrid method as standalone solver, it is important that all nodes are aggregated. In the worst case, non-aggregated left-over nodes would not be considered in the multigrid solution process at all. This may lead to severe convergence problems. In case of the multigrid method being used as a preconditioner within an outer Krylov subspace solver, non-aggregated left-over nodes are not a major problem since they still would be handled by the Krylov solver.

Note that in contrary to usual inner nodes it makes sense for (homogeneous) Dirichlet nodes to be dropped in the aggregation. For more details the reader may refer to Remark 3.3.3 and Appendix C.

3.3.7. Coarse level aggregates

In Figure 3.4 the effect of different parameters on the aggregation of the fine level nodes is studied for the artificial Example 3.3.4. The black nodes on the 8×8 mesh are aggregated into the gray aggregates. Each of these gray aggregates on level 0 represents one node on level 1. Figure 3.5 shows exemplarily how the corresponding coarse level aggregates on level 1 may look

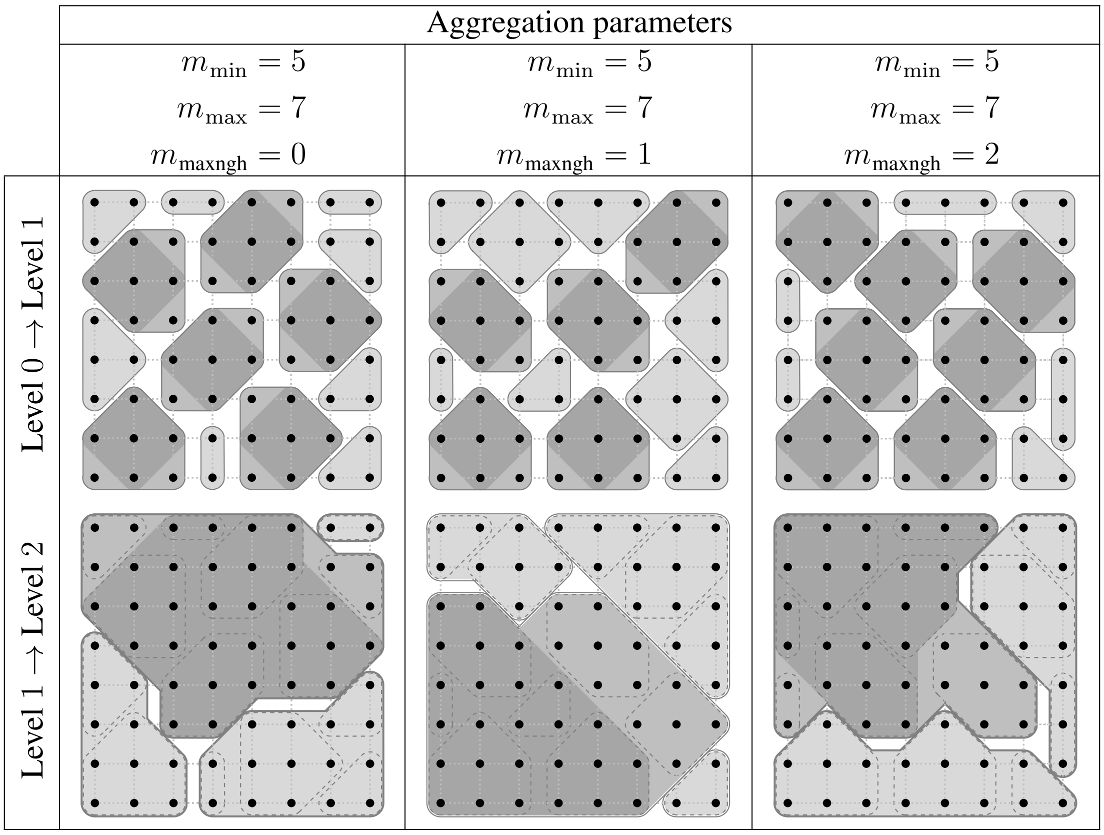


Figure 3.5.: Coarse level aggregates.

like when applying Algorithm 3 to the fine level aggregates on level 0. Note that the maximum number of neighboring nodes (or aggregates) might change as one can see from Figure 3.5. In this example, the maximum number of neighboring nodes on the finest level 0 is fixed to be 4 by assumption. On level 1 however, one finds up to 8 neighboring nodes, since the connections between nodes are not restricted to horizontal and vertical directions any more. With the increasing number of connections the influence of the parameter $m_{\max}$ is also changing: whereas it had no meaning for the Phase I algorithm on level 0, it actively restricts the size of aggregates on level 1 in the tentative aggregation Phase I algorithm.

Example 3.3.4 has been chosen with the purpose of demonstrating some extreme behavior of the aggregation algorithm to give the reader a better understanding of the meaning of the aggregation parameters. It shall be mentioned that for real world problems, one usually finds a more reasonable connectivity structure in A_ℓ, where different reasonable parameter choices have less drastic effects.

Remark 3.3.7 (Reasonable parameter choices). Even though the aggregation process is mainly dominated by the underlying mesh connectivity and the ordering of the non-aggregated nodes $\widehat{\mathrm{R}}_\ell$, the user is still responsible for reasonably choosing the user parameters. It is certainly worth to put together some general thoughts about the proper choice of the user parameters for the aggregation routines as described in the previous sections.

Minimum size of aggregates: A proper choice of $m_{\min}$ depends on the number of neighboring nodes of each node of the corresponding mesh or level. It definitely should be smaller than the average number of connections for the inner nodes away from the boundaries. Otherwise one can skip the Phase I phase completely and save computational costs. A reasonable choice of $m_{\min}$ allows to guarantee a minimum coarsening rate. In case of a regular 2D mesh on the finest level, one has to choose a value smaller than 9; in case of a regular 3D mesh, $m_{\min}$ should be smaller than 27. A good choice would be, e.g., 6 (in 2D) or 18 (in 3D), respectively.

Maximum size of aggregates: The $m_{\max} \geq m_{\min}$ parameter allows to define an upper bound for the coarsening process. This helps to make the coarsening rate uniform over all multigrid levels. Sometimes this is important, especially on coarser levels where the number of off-diagonal entries in the level matrices is increasing. Of course, the user should choose $m_{\max} > m_{\min}$ with $m_{\min}$ sufficiently small to give the Phase I algorithm enough freedom to choose appropriate tentative aggregates. Typical choices for regular fine level meshes are 27 (for 3D) and 9 (for 2D).

Maximum number of aggregated neighbors: For graphs arising from finite element meshes the choice $m_{\mathrm{maxngh}} = 0$ gives reasonable results, since it guarantees a minimal graph distance of 2 for two distinct tentative aggregates throughout the whole domain. Left-over nodes between two tentative aggregates are aggregated latest in Phase III.

Example 3.3.8 (3D aggregation)**.** Figure 3.6 shows a typical example for aggregates on an unstructured tetrahedral mesh. The aggregation parameters have been chosen exemplarily to produce rather small aggregates of a size between 12 and 20 nodes. The corresponding coarsening rate is between 19 and 20 and therefore limited by $m_{\max}$. The parameter $m_{\mathrm{maxngh}} = 16$ is rather big and causes the Phase II algorithm to fail in extending existing aggregates. One can also see how the number of nonzero entries per row in the level matrix A_ℓ is increasing with ℓ, which gives a rough idea how the number of node connections is changing on the coarser levels. Note that the number of non-zeros per row is also influenced by other factors such as the concrete choice of the transfer operators (see Sections 3.4.1 and 3.5).

3.4. Tentative prolongation operators

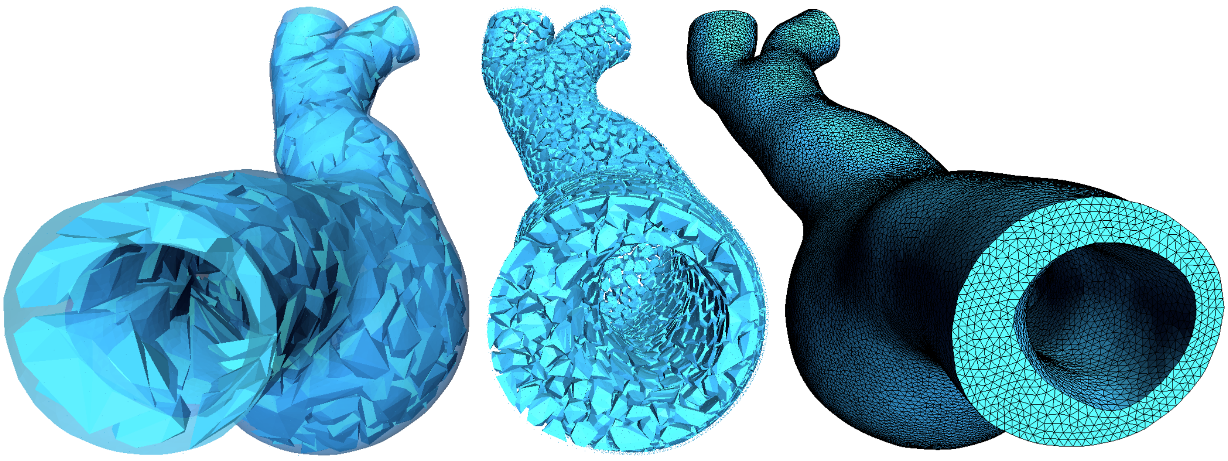
The aggregates from Section 3.3 are used to generate tentative prolongation operators $\widehat{P}_{\ell+1} \in \mathbb{R}^{n_\ell \times n_{\ell+1}}$ for $\ell = 0, \dots, \ell_{\max} - 1$ which define the transfer of coarse level information from level $\ell + 1$ to the fine level ℓ. These two-level prolongation operators can be used to define the *composite tentative prolongator* $\widehat{P}^1_\ell : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_0}$ by

$$\widehat{P}^1_\ell = \widehat{P}_1 \cdots \widehat{P}_\ell \quad \text{for } \ell > 0 \text{ and} \quad \widehat{P}^1_0 = I \text{ for } \ell = 0, \tag{3.7}$$

which represents the transfer of a coarse vector from level ℓ to the finest level 0 (cf. Vaněk et al. [198]).

3.4.1. Prerequisites for tentative prolongation operators

For the construction of $\widehat{P}_{\ell+1}$, one needs the set of aggregates $\left\{\mathscr{A}_\ell^{(i)}\right\}_{i=1}^{m_{\mathscr{A}_\ell}}$ of level ℓ together with the corresponding near null space vectors $B_\ell \in \mathbb{R}^{n_\ell \times n_B}$ which satisfy $\mathrm{A}_\ell B_\ell \approx 0$ at least away from the Dirichlet boundaries.



ℓ	m_ℓ	$m_{\mathscr{A}_\ell}$	Phase I / II / III	n_ℓ	nnz/row	# procs
0	966003	48905	48637 / 0 / 268	2898009	44.13	32
1	48905	2552	2089 / 0 / 463	293430	292.65	4
2	2552	–	–	15312	596.42	1

Figure 3.6.: Example for aggregation of an unstructured tetrahedral mesh. The aggregation parameters have exemplarily been chosen to be $m_{\min} = 12$, $m_{\max} = 20$ and $m_{\text{maxngh}} = 16$ with a natural ordering of the nodes defined by the meshing algorithm on the finest level. m_ℓ denotes the number of nodes and n_ℓ the number of degrees of freedom on level ℓ. $m_{\mathscr{A}_\ell}$ is the number of aggregates built on ℓ. The aggregation has been performed in parallel using 32 processors on the finest level.

Remark 3.4.1 (Near null space vectors). Following the considerations in Vaněk et al. [192], B_0 is typically chosen to be a generator of near null space modes for the matrix A_0 on the finest level. In the context of finite elements this means the kernel of the matrix A_0 obtained from the finite element model without essential boundary conditions (such as Dirichlet boundaries). Near null space modes, determined by the element definition and the geometry, are often well-known and can easily be calculated. In the scalar case using Lagrange elements, the near null space modes are given by non-trivial constant vectors. For non-scalar systems the same may apply component-wise, e.g., for Navier–Stokes problems one can choose component-wise constant vectors as good approximation of the zero energy modes. For linear elasticity the near null space modes are given by the rigid body modes and therefore the number of near null space vectors might differ from the number of degrees of freedom per node. For example, for 3D elasticity one has 3 degrees of freedom for the displacement variables in each direction but 6 rigid body modes (3 for the translation in each spatial direction and 3 rotatory modes) as zero energy modes defining the near null space vectors of the operator A_0. The near null space vectors for the coarse level matrices A_ℓ, $\ell > 0$ can be generated from the near null space B_0 during the setup of the tentative transfer operators as described in the following. Therefore, it is sufficient to provide only the near null space modes for the finest level.

The near null space vectors are used to extend the node-based information from the aggregates to define transfer operators declared for the corresponding degrees of freedom. Therefore, the meaning of the near null space vectors B_ℓ for the multigrid method is twofold:

(N1) *Coarse level modes:* First and most important for the multigrid principle: the set of near null space vectors selects the slowly converging modes which are supposed to be transferred to the next coarser level where they are tackled by the level smoother more efficiently. Usually, the near null space contains the constant vectors which represent the error modes that have to be eliminated by a direct solver (on the coarsest level). To obtain optimal results, one has to wisely choose a minimal set of linear independent near null space vectors that covers all important error modes. On the other hand one should avoid to choose too many near null space vectors as the number of near null space vectors n_B implicitly defines the number of degrees of freedom per node on the coarse level. A too high number of near null space vectors leads to a higher number of degrees of freedom per node on the coarser levels and may drastically increase the operator complexity (see Definition 3.2.1).

(N2) *Transfer operator pattern:* From the technical point of view, the set of near null space vectors also defines the local aggregate-wise nonzero pattern of the tentative transfer operator (see Section 3.4.2). Often, the set of near null space vectors is provided by the outer application, e.g., as the set of rigid body modes for structural problems, which again is based on some intrinsic assumptions regarding the ordering of degrees of freedom. Whereas in practice the multigrid will work even if one only has disturbed near null space vectors in **(N1)**, it is highly essential that the sparsity pattern of the level matrices A_ℓ fits to the near null space vectors provided by the application. If the local aggregate-wise near null space vectors contain (nearly) linearly dependent vectors, the coarse level problem $\mathrm{A}_{\ell+1}$ resulting from the Galerkin product (2.6) might tend to be nearly singular, even if the fine level operator A_ℓ is non-singular (cf. Remark 6.2.3). This can be avoided by an appropriately chosen pattern (see Section 4.4).

The next section gives some details on the construction of the tentative prolongation operator which may also provide some insight in the meaning of prerequisite **(N2)**.

3.4.2. Construction principle

In order to enforce objective **(O3)** from Section 3.2.2 one has to make sure that

$$Rg\big(B_0\big) \subset Rg\big(\widehat{P}_\ell^1\big) \quad \text{for all } \ell = 1, \ldots, \ell_{\max} \tag{3.8}$$

holds for the composite prolongation operator $\widehat{P}_\ell^1$ from (3.7). The idea is to preserve necessary properties of the fine level matrix on the coarser levels such as the slowly converging (algebraically) smooth error modes, which are represented by the near null space modes. The objective as formulated in (3.8) is met by the recursive definition of the coarse level null space vectors using the expression

$$\widehat{P}_{\ell+1} B_{\ell+1} = B_\ell \tag{3.9}$$

for $\ell = 0, \ldots, \ell_{\max} - 1$.

The equation (3.9) is ideal to simultaneously create the tentative prolongation operator $\widehat{P}_{\ell+1}$ and the set of n_B coarse near null space vectors $B_{\ell+1} \in \mathbb{R}^{n_\ell \times n_B}$ during the multigrid setup phase.

Algorithm 7: Construction algorithm for generating $\widehat{P}$.

```
Procedure $\widehat{P}_{\ell+1}$ ($\{\mathscr{A}_\ell^{(i)}\}_{i=1}^{m_{\mathscr{A}_\ell}}$,$B_\ell$)
    *Extract local null space blocks from corresponding aggregates*
    Partition $B_\ell$ into blocks $B_\ell^{(i)} \in \mathbb{R}^{n_i \times n_B}$, $i = 1, \dots, m_{\mathscr{A}}$
    *Perform local QR-decomposition*
    Decompose $B_\ell^{(i)} = Q_\ell^{(i)} R_\ell^{(i)}$ with $Q_\ell^{(i)} \in \mathbb{R}^{n_i \times n_B}$ an orthogonal matrix and
    $R_\ell^{(i)} \in \mathbb{R}^{n_B \times n_B}$ an upper triangular matrix
    *Define global tentative prolongation operator*
    Build tentative transfer operator $\widehat{P}_{\ell+1} := \operatorname{diag}\big(Q_\ell^{(1)}, \dots, Q_\ell^{(m_{\mathscr{A}_\ell})}\big)$
        $$\widehat{P}_{\ell+1} = \begin{pmatrix} Q_\ell^{(1)} & & & \\ & Q_\ell^{(2)} & & \\ & & \ddots & \\ & & & Q_\ell^{(m_{\mathscr{A}_\ell})} \end{pmatrix} \in \mathbb{R}^{n_\ell \times n_{\ell+1}}$$
    *Build global coarse null space*
    Set coarse level null space vector
        $$B_{\ell+1} = \begin{pmatrix} R_\ell^{(1)} \\ R_\ell^{(2)} \\ \vdots \\ R_\ell^{(m_{\mathscr{A}_\ell})} \end{pmatrix} \in \mathbb{R}^{n_{\ell+1} \times n_B}$$
```

As one can see from Algorithm 7, a local QR-decomposition of the building blocks $B_\ell^{(i)} \in \mathbb{R}^{n_i \times n_B}, i = 1, \dots, m_{\mathscr{A}}$ of the fine level null space B_ℓ is used to locally satisfy (3.9). Here, n_i denotes the number of degrees of freedom of aggregate $\mathscr{A}_\ell^{(i)} \in \{\mathscr{A}_\ell^{(i)}\}_{i=1}^{m_{\mathscr{A}_\ell}}$. For each building block $B_\ell^{(i)}$ one extracts the rows corresponding to the non-trivial support of the local basis functions of aggregate i. That is, the property (3.9) is enforced aggregate by aggregate when applying the QR-decomposition as visualized in Figure 3.7. This means that the columns of the tentative prolongator $\widehat{P}_{\ell+1}$ are formed by orthonormalized restrictions of the columns of B_ℓ onto the aggregate $\mathscr{A}_\ell^{(i)}$. For each aggregate, this construction gives rise to n_B degrees of freedom on the coarse level, represented by the corresponding n_B columns in the prolongation operator $\widehat{P}_{\ell+1}$.

Note that the local pattern of the building blocks $B_\ell^{(i)} \in \mathbb{R}^{n_i \times n_B}$ implicitly defines the sparsity pattern of the tentative prolongator $\widehat{P}_{\ell+1}$ (see **(N2)** in Section 3.4.1). Due to the block diagonal structure of $\widehat{P}_{\ell+1}$ the coarse level basis functions in the columns of $\widehat{P}_{\ell+1}$ by construction do

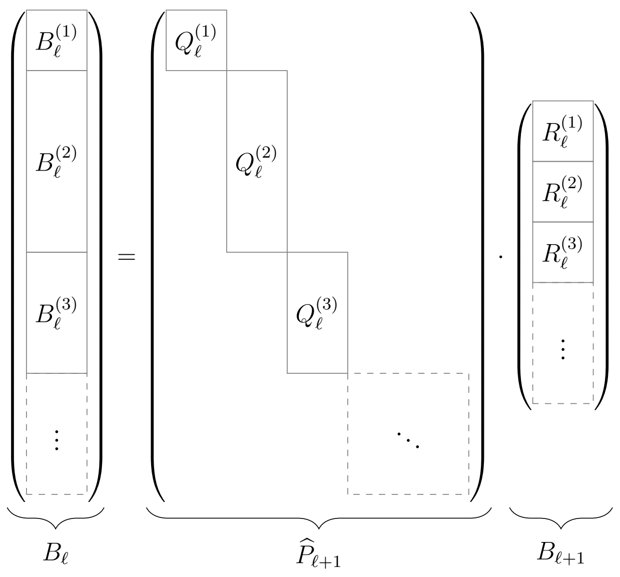


Figure 3.7.: Schematic view of local QR-decomposition to generate tentative prolongation operator $\widehat{P}_{\ell+1}$ and coarse null space vectors $B_{\ell+1}$ from the fine level null space vectors B_ℓ.

not overlap and therefore satisfy the objective **(O2)** from Section 3.2.2. Furthermore, due to the orthonormality of the $Q_\ell^{(i)}$ blocks in the local QR-decomposition it follows that $\left(\widehat{P}_\ell^1\right)^\mathsf{T} \widehat{P}_\ell^1 = I$ for every ℓ which is necessary to bound the convergence rate of the multigrid method (cf. Vaněk et al. [198]). The construction principle of the coarse level null space $B_{\ell+1}$ in (3.9) describes the recursive formulation of the null space preservation property **(O3)**, since for $\mathrm{A}_\ell B_\ell \approx 0$ it holds

$$\mathrm{A}_{\ell+1} B_{\ell+1} = \widehat{R}_{\ell+1} \mathrm{A}_\ell \widehat{P}_{\ell+1} B_{\ell+1} = \widehat{R}_{\ell+1} \mathrm{A}_\ell B_\ell \approx 0, \tag{3.10}$$

i.e., $B_{\ell+1}$ represents the near null space for the coarse level operator $\mathrm{A}_{\ell+1}$.

Remark 3.4.2 (Prerequisites for QR decomposition). Be aware that the local QR-decomposition is only defined for $n_i \geq n_B$. However, this condition is not met, e.g., for elasticity problems in 3D where one may have $n_i = 3$ on the finest level $\ell = 0$ but $n_B = 6$. This happens if you have too small aggregates with only one node (single node aggregate). So, for elasticity problems single node aggregates need special treatment (see Remark 6.3.2 for some more details).

3.5. Prolongation smoothing methods

The idea of smoothed aggregation AMG methods is to apply a simple smoothing procedure to improve the convergence properties of the multigrid method by reducing the energy of the coarse level basis functions. It has been introduced in Vaněk [193, 194] for symmetric positive definite problems and then further developed in Vaněk et al. [192, 197]. Even though the non-smoothed tentative prolongation operators recently have found some more attention (see Emans [61], Kim et al. [106], Napov and Notay [139]), smoothing the coarse level basis functions in the transfer operators often leads to a notable improvement of the multigrid convergence.

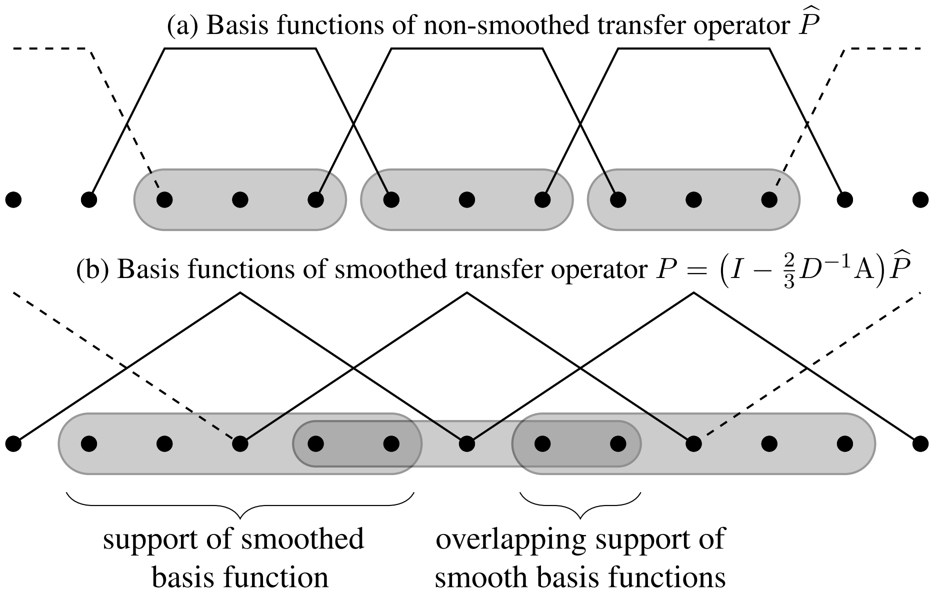

(a) Basis functions of non-smoothed transfer operator $\widehat{P}$
(b) Basis functions of smoothed transfer operator $P = \big(I - \frac{2}{3}D^{-1}\mathrm{A}\big)\widehat{P}$
support of smoothed basis function
overlapping support of smooth basis functions

Figure 3.8.: Shape and support of transfer operator basis functions for 1D example.

3.5.1. Classical smoothed aggregation methods

To reduce the energy of the coarse level basis functions (see objective **(O4)** in Section 3.2.2), one sweep with a Jacobi iteration is applied to the basis functions of the non-smoothed tentative prolongation operator $\widehat{P}_{\ell+1}$ such that one obtains the smoothed aggregation prolongation operator given by

$$P_{\ell+1} := M_J\widehat{P}_{\ell+1} = \big(I - \omega D^{-1}\mathrm{A}_\ell\big)\widehat{P}_{\ell+1} \tag{3.11}$$

with M_J denoting the iteration matrix of the Jacobi iteration (see Section 2.1.1) and $\omega \geq 0$ a damping factor.

Example 3.5.1 (Shape and support of smoothed coarse level basis functions)**.** To study the effect of transfer operator smoothing a simple tridiagonal matrix is used with the matrix stencil $\mathrm{A}_0 = \langle -1, 2, -1\rangle$ resulting from a 1D discretization of a purely diffusive problem. For the aggregates ideal 3 node aggregates are built. In Figure 3.8 the shape and the support of the non-smoothed prolongation basis functions is given and compared against the shape and the support of prolongation basis functions after applying one smoothing sweep using (3.11). For the damping parameter $\omega = \frac{2}{3}$ one obtains piece-wise linear hat functions for that idealized example. The smoothing step not only affects the shape of the basis functions but also widens the support of the basis functions in all directions of (strong) connections of the matrix A_0 (see, e.g., Gee et al. [71]), such that the support of neighboring smoothed basis functions is overlapping. A more detailed study on the effect of transfer operator smoothing (including non-symmetric problems) can be found in Appendix B.

The overlapping basis functions lead to more non-zero entries in the resulting coarse level matrices and therefore may significantly increase the memory consumption and the operator complexity as introduced in Definition 3.2.1. When applying a dropping strategy to highlight the strong connections in A_ℓ, it might also be advantageous to use the resulting filtered matrix $\mathrm{A}_\ell^{\mathrm{fl}}$ in (3.11) instead of A_ℓ to gain some control over the operator complexity especially in case of highly anisotropic problems. This way, one can avoid a higher operator complexity caused by transfer operator smoothing for comparably weak and unimportant connections. For further discussion on special strategies of smoothed aggregation multigrid methods for anisotropic problems the reader is referred to Gee et al. [71].

Remark 3.5.2 (Smoothed aggregation for symmetric positive definite problems). In fact, for symmetric positive definite problems one uses the following slightly modified variant

$$P_{\ell+1} := \left(I - \frac{\omega_{sym}}{\widetilde{\lambda}} D^{-1}\mathrm{A}_\ell\right)\widehat{P}_{\ell+1} \tag{3.12}$$

with an approximation $\widetilde{\lambda} \geq \rho\big(D^{-1}\mathrm{A}_\ell\big)$ instead of (3.11). Our implementation applies a small number of iterations with the Power method which is – at least in the symmetric case – appropriate to calculate an approximation for $\widetilde{\lambda}$ with reasonable computational effort (cf. Golub and van der Vorst [77]). In Vaněk et al. [199] one can find the proof that for symmetric positive definite problems with a coarsening rate of 3 in all spatial directions the choice $\omega_{sym} = \frac{4}{3}$ in (3.12) is optimal in the sense of minimizing the spectral radius of the coarse level matrices. For the Example 3.5.1, where one has a symmetric tridiagonal matrix modeling a 1D problem with perfect 3 node aggregates, (3.12) with $\omega_{sym} = \frac{4}{3}$ recovers the smoothed basis functions from Figure 3.8, since the eigenvalue approximation gives $\rho\big(D^{-1}\mathrm{A}_0\big) \lessapprox 2$.

The current development of smoothed aggregation algebraic methods goes into different directions: In Brezina et al. [43, 44] the authors describe an adaptive method for finding near null space vectors for the underlying problem utilizing the concept of algebraic smoothness. Very recent work has been done in context of aggressive coarsening for smoothed aggregation AMG methods and massive smoothing of the transfer operators (cf. Vaněk [195], Vaněk and Brezina [196]). Instead of relaxation-based smoothing methods for the transfer operators as used in the classical smoothed aggregation approach and extensions (Sala and Tuminaro [173], Wiesner et al. [217]) one uses a transformed Chebyshev polynomial in A to improve convergence for rapid coarsening which cannot be compensated by a higher number of level smoothing sweeps (cf. Vaněk [195]). Another extension of the smoothed aggregation method is given with the energy minimization approaches that are described in more detail in the next section.

3.5.2. Energy minimization approaches

In between there are different more elaborate transfer operator strategies based on the concept of energy minimization of the transfer operator basis functions (cf. **(O4)** in Section 3.2.2). Among others, the interested reader might find a good starting point in Brandt [39], Brannick and Zikatanov [41], Kolev and Vassilevski [107], Mandel et al. [124], Vassilevski [203] and Wan et al. [209] to dive into the topic of energy minimization transfer operators.

An interesting energy minimization approach is presented in the work by Sala and Tuminaro [173]. Therein, the authors introduce a flexible energy minimization strategy based on a set of local damping parameters associated with the coarse level basis functions instead of a single global damping parameter ω as used for the classical smoothed aggregation. That is, (3.11) is replaced by the smoothing process

$$P_{\ell+1} := M_J \widehat{P}_{\ell+1} = \widehat{P}_{\ell+1} - D^{-1}\mathrm{A}_\ell \widehat{P}_{\ell+1}\bar{\Omega}, \tag{3.13}$$

where $\bar{\Omega} = \operatorname{diag}\big(\bar{\omega}_j\big)_{j=1}^{n_{\ell+1}} \in \mathbb{R}^{n_{\ell+1}\times n_{\ell+1}}$ is a diagonal matrix with the diagonal elements defined by

$$\bar{\omega}_j = \frac{\left(\widehat{P}^{(j)}_{\ell+1}, D^{-1}\mathrm{A}_\ell \widehat{P}^{(j)}_{\ell+1}\right)_{\mathrm{A}_\ell^\mathsf{T}\mathrm{A}_\ell}}{\left\|D^{-1}\mathrm{A}_\ell \widehat{P}^{(j)}_{\ell+1}\right\|^2_{\mathrm{A}_\ell^\mathsf{T}\mathrm{A}_\ell}}. \tag{3.14}$$

The $\widehat{P}^{(j)}_{\ell+1}$ denotes the j-th column or basis function of the tentative transfer operator $\widehat{P}_{\ell+1}$. The expression (3.14) can be found as the minimum of

$$\min_{\bar{\omega}_j} \|\mathrm{A}_\ell P_{\ell+1}\|_F^2 = \sum_{j=1}^{n_{\ell+1}} \min_{\bar{\omega}_j} \left\|\big(I - \bar{\omega}_j D^{-1}\mathrm{A}_\ell\big)\widehat{P}^{(j)}_{\ell+1}\right\|^2_{\mathrm{A}_\ell^\mathsf{T}\mathrm{A}_\ell}. \tag{3.15}$$

The choice of the minimization problem (3.15) is motivated by considerations based on ideal transfer operators given in Brannick and Zikatanov [41], Sala and Tuminaro [173] and Trottenberg et al. [186, Section A.2.3]. The Frobenius norm is adopted primarily for convenience as it leads to easily computable quantities. The diagonal matrix $\bar{\Omega}$ with varying coefficients in (3.13) breaks the null space preservation property **(O2)**. To overcome this issue the diagonal matrix $\bar{\Omega}$ is shifted twice: one can easily find a $n_\ell \times n_\ell$ diagonal matrix $\hat{\Omega}$ using the relationship $\hat{\Omega}\widehat{P}_{\ell+1} = \widehat{P}_{\ell+1}\bar{\Omega}$. Then one can choose the local damping factors $\tilde{\omega}_i = \max\Big\{0, \min_{j, a_{ij}\neq 0} \hat{\omega}_j\Big\}$ and define the according $n_\ell \times n_\ell$ diagonal matrix $\tilde{\Omega}$ with the row-based local damping factors $\tilde{\omega}_i$. The corresponding transfer operator smoothing process

$$P_{\ell+1} := M_J \widehat{P}_{\ell+1} = \widehat{P}_{\ell+1} - D^{-1}\tilde{\Omega}\mathrm{A}_\ell \widehat{P}_{\ell+1} \tag{3.16}$$

satisfies the null space preservation property by definition.

An alternative energy minimization approach for building coarse basis functions is presented in Mandel et al. [124]. Instead of only one Jacobi sweep a fast projected gradient descent algorithm is used for the solution of a constraint minimization problem which preserves the near null space modes with additional constraints on the supports of the prolongator basis functions. Wan et al. [209] give an alternative formulation of the constrained minimization problem using a saddle point system, where the conjugate gradient (CG) method is applied to solve the Schur complement system resulting from the saddle point problem. A general interpolation strategy using energy minimization for AMG methods is proposed in Olson et al. [153]. Again, each column of the prolongation operator P is minimized in an energy-based norm while enforcing the preservation of given near-null space modes on a prescribed fixed sparsity pattern to keep the operator complexity low. More elaborate solution strategies such as CG-based and GMRES-based methods are used for the minimization problem. The next chapter proposes a flexible smoothed aggregation transfer operator strategy for non-symmetric problems arising from linear systems with convective character. It is based on similar concepts of enforcing near-null space modes and sparsity patterns as the methods in Olson et al. [153].

CHAPTER 4

A flexible smoothed aggregation approach for convection-dominated problems

This chapter contains more advanced topics regarding multigrid transfer operators for non-symmetric linear systems arising from convection-dominated problems, which still can be considered as challenging for multigrid methods. For example, non-symmetric linear systems typically arise from problems with some convective character, such as flow problems modeled by the (incompressible) Navier–Stokes equations.

To extend state-of-the-art smoothing techniques for the transfer operator basis functions to the non-symmetric case, one basically needs two main ingredients: First of all, the standard Galerkin approach has to be replaced by a Petrov–Galerkin approach which allows for a different smoothing of the restriction operator basis functions. Additionally, the Petrov–Galerkin approach has to be combined with an appropriate smoothing method for the non-symmetric transfer operators. It is important that the non-symmetry of the linear operator A_ℓ is taken into account when smoothing the transfer operator basis functions. Both the Petrov–Galerkin approach and the non-symmetric transfer operator smoothing process are necessary for building robust and efficient prolongation and restriction operators for the non-symmetric case.

In this chapter, a novel smoothed aggregation method is developed for non-symmetric problems. To the author's knowledge this is the first attempt to combine mode preservation and sparsity pattern constraints which have been introduced for symmetric problems in context of energy minimization methods (e.g., Mandel et al. [124], Olson et al. [153]) with a Petrov–Galerkin approach for non-symmetric problems (e.g., Sala and Tuminaro [173]), to obtain optimized smooth transfer operators for non-symmetric systems. Specifically, a contribution of this work is that these level transfers can be viewed as projections of idealized operators into a space which restricts sparsity patterns while guaranteeing that the action of the resulting coarse discretization applied to certain modes accurately reflects the action of the Schur complement. The proposed procedure is simpler to implement, computationally less expensive than the one proposed in Olson et al. [153], and in our opinion the present Galerkin framework is clearer and more relevant for understanding non-symmetric systems. Additionally, it is shown how the generation of sparsity patterns for grid transfers can be integrated with an algorithm for generating grid transfer coefficients. This way, it allows for a maximum of flexibility regarding application-specific enhancements. Finally, it should be noted that our grid transfer algorithm is quite similar to the steepest descent approach in Mandel et al. [124] for symmetric systems. However, steepest descent for "energy minimization" no longer makes sense in the non-symmetric case and so the interpretation pursued here is that of a Richardson iteration.

In the beginning a short overview of existing multigrid transfer operator strategies for the non-symmetric case is given and compared with our new approach. Then, the Petrov–Galerkin approach for non-symmetric systems (cf. Section 4.2) and our novel transfer operator smoothing strategy are introduced which incorporates mode preservation techniques and sparsity pattern constraints (cf. Section 4.4) motivated by ideas of the ideal transfer operators (cf. Section 4.3). Section 4.5 gives numerical justification for the new level transfers using convection-diffusion and flow problems.

4.1. Multigrid for non-symmetric problems

In the symmetric case, the AMG theory is often based on bounding an energy norm of grid transfer basis functions which follows naturally when one has a *symmetric positive definite* (SPD) operator to define an energy norm. While there exist a multitude of AMG algorithms, variants of classical algebraic multigrid (cf. Brandt et al. [37], Ruge and Stüben [166]) and the smoothed aggregation multigrid method (cf. Vaněk et al. [192]), are the most heavily used. Even though such AMG schemes have been applied to non-symmetric systems, they are much less developed for this case than for SPD matrices. A few notable algebraic multigrid contributions which specifically target non-symmetric systems include Dutto et al. [57], Gravemeier et al. [79], Lallemand et al. [114], Lonsdale [121], Mavriplis and Venkatakfrishnan [129], Mavriplis [130] and Bank et al. [13]. The majority of these follow agglomeration or non-smoothed aggregation ideas where the emphasis is on producing a coarse discretization matrix that maintains desirable properties of the fine level discretization operator (e.g., conservation or stability properties) as opposed to attempting to find *optimal* grid transfers with some approximation property in mind.

Some papers that look more closely at approximation properties include Brezina et al. [44], Giddings and Fish [74], Guillard and Vaněk [81], Mense and Nabben [132], Notay [148], Olson et al. [153], Sala and Tuminaro [173] and Wagner [207]. In Brezina et al. [44], an approximation property is presented for non-Hermitian systems which guarantees two-level mesh independent convergence, though level transfers are based on a relatively expensive process. A Fourier analysis of smoothing for a two-level multigrid method that is applied to a one-dimensional non-symmetric advection-diffusion problem can be found in Giddings and Fish [74]. Guillard and Vaněk [81] gives a theoretical analysis corresponding to a smoothed aggregation-based multigrid algorithm for non-symmetric systems. In Mense and Nabben [132], a convergence analysis is performed that is suitable for non-symmetric systems corresponding to multilevel methods based on an approximate factorization. An algebraic analysis is also presented in Notay [148] for non-smoothed aggregation. In Sala and Tuminaro [173] a generalization of the smoothed aggregation idea for non-symmetric systems is proposed while in Olson et al. [153] an iterative framework is given for improving an initial prolongator or restrictor based on energy minimization ideas where energy is defined using the matrix $A^T A$ with A as the discretization matrix.

In Wagner [207], criteria are formulated for accurately approximating *smooth* error using a Euclidean norm.

Based on a Petrov–Galerkin approach (see Section 4.2), our novel smoothed aggregation approach for non-symmetric problems builds on the connections between multigrid and approximate factorization techniques for 2×2 block systems. These approaches have a long history going back to the 1980's and are based on the idea of eliminating the $(1, 1)$ block to produce a reduced or Schur complement system. This Schur complement can be viewed as a coarse discretization matrix within a two-level multigrid method where *idealized Schur complement grid transfers* are effectively defined by the algebraic elimination process. A true multigrid process is obtained by approximating these idealized transfers and by recursively applying the technique to the coarse discretization matrix. Theoretically, the key idea is to show that the preconditioning matrix is spectrally equivalent to the linear system that one wishes to solve. A full description of approximate block factorization preconditioners along with a comprehensive analysis can be found in Vassilevski [201], which builds on two primary articles (cf. Axelsson and Vassilevski [7, 8]). In Axelsson and Vassilevski [8], an algebraic multilevel iteration or AMLI iteration (cf. Kraus and Margenov [108]) is proposed where the inverse of the Schur complements are approximated by matrix polynomials based on the coarse discretization matrix. A key feature of AMLI methods is that it is possible to show that the preconditioning technique is optimal without making regularity assumptions. The theoretical analysis is further extended and unified in Notay [142] and some analogies between multigrid and AMLI are also made. There has been much recent theoretical and practical interest in AMLI extensions especially in the context of aggregation-based methods which includes approaches relevant for non-symmetric systems (cf. Notay [144]). A different perspective is now given which uniquely combines ideas associated with approximative Schur complements (or idealized grid transfer operators), energy minimization, and non-symmetric systems. Before discussing the transfer operator smoothing some details are given on the Petrov–Galerkin framework for building restriction operators.

4.2. Restriction operator strategies

The Petrov–Galerkin approach explained in this section is the first key ingredient in a multigrid method for non-symmetric problems. Note that the Petrov–Galerkin method has only an effect in combination with an appropriate smoothing strategy for the transfer operator basis functions (see also Section 4.4).

4.2.1. Restriction operators for symmetric problems

Before focusing on non-symmetric problems, the standard Galerkin approach is briefly reviewed which is widely used for symmetric problems. In the standard Galerkin approach the restriction operator is just built from the transpose of the prolongation operator P, i.e., $R = P^{\mathsf{T}}$. As a consequence, for symmetric $\mathrm{A}_\ell = \mathrm{A}_\ell^{\mathsf{T}}$ the coarse level remains symmetric as one can see from

$$\mathrm{A}_{\ell+1} = P_{\ell+1}^{\mathsf{T}} \mathrm{A}_\ell P_{\ell+1} = \left(P_{\ell+1}^{\mathsf{T}} \mathrm{A}_\ell^{\mathsf{T}} P_{\ell+1}\right)^{\mathsf{T}} = \mathrm{A}_{\ell+1}^{\mathsf{T}}.$$

Using smoothed transfer operators, as introduced in Section 3.5.1, the smoothed restrictor has the form

$$R_{\ell+1} := P_{\ell+1}^{\mathsf{T}} = \Big(\big(I - \omega D^{-1} \mathrm{A}_\ell\big) \widehat{P}_{\ell+1}\Big)^{\mathsf{T}} = \widehat{R}_{\ell+1}\big(I - \omega \mathrm{A}_\ell^{\mathsf{T}} D^{-1}\big), \tag{4.1}$$

where $\widehat{R} := \widehat{P}^{\mathsf{T}}$ denotes the non-smoothed tentative restriction operator. For symmetric problems with $\mathrm{A} = \mathrm{A}^{\mathsf{T}}$, the transposed of the (smoothed) prolongation operator is a good choice for the restriction with comparably low numerical costs, since the smoothing or energy minimization is performed only once for the prolongation operator. With an efficient implementation one can even avoid explicitly building the transpose of P and resemble the effect of the restriction operator implicitly within the Galerkin product (2.6). This is also true for the prolongation smoothing strategies based on more elaborate energy minimization principles from Section 3.5.2.

4.2.2. Petrov–Galerkin approach for non-symmetric problems

A simple but valid choice for transfer operators in the non-symmetric case are the non-smoothed tentative prolongation and restriction operators $\widehat{P}$ and $\widehat{R} = \widehat{P}^{\mathsf{T}}$, since assuming a symmetric graph of A_ℓ the corresponding non-smoothed basis functions are independent of the symmetry of the fine level operator A_ℓ. But the tentative transfer operators cannot guarantee optimal multigrid convergence, leading to higher iteration numbers compared to smoothed transfer operators.

There are different ideas to improve the transfer operators in the non-symmetric case, such as applying smoothing to the symmetric part of the operator only (cf. Dendy [51]). In Vaněk et al. [198] a Petrov–Galerkin approach for calculating the coarse level problem with $R \neq P^{\mathsf{T}}$ is recommended, since Galerkin coarsening with $R = P^{\mathsf{T}}$ does not guarantee convergence rather stability for non-symmetric problems (cf. Brezina et al. [44]). In case of smoothed aggregation prolongation operators,

$$R_{\ell+1} := \widehat{R}_{\ell+1}\big(I - \omega \mathrm{A}_\ell D^{-1}\big) \tag{4.2}$$

is used as smoothing strategy for the restrictor. That is, in contrast to (4.1), one uses A_ℓ instead of $\mathrm{A}_\ell^{\mathsf{T}}$. In (4.2), $\widehat{R} = \widehat{P}^{\mathsf{T}}$ guarantees the prolongation and restriction operator to be compatible in the sense that they definitely share the minimum support of the non-smoothed transfer operator basis functions, since both are based on the same aggregates. The expression in (4.2) can be understood as smoothing the restriction basis functions with the heuristic goal of reducing their energy in an $\mathrm{A}_\ell \mathrm{A}_\ell^{\mathsf{T}}$-norm instead of an $\mathrm{A}_\ell^{\mathsf{T}} \mathrm{A}_\ell$ norm (cf. Section 3.5.2). The damping parameter ω in (4.2) is usually chosen to be the same as for the prolongator in (4.1). Note that for the non-symmetric case one cannot use the formulation as described in Remark 3.5.2 which uses a cheap iterative Power algorithm for approximating the maximum eigenvalue of $D^{-1}\mathrm{A}_\ell$ that may fail for non-symmetric matrices A_ℓ. Instead, the user is supposed to provide the damping parameter ω.

Remark 4.2.1 (Comparison of standard Galerkin and Petrov–Galerkin). Obviously, (4.2) coincides with (4.1) for $\mathrm{A}_\ell = \mathrm{A}_\ell^{\mathsf{T}}$ and therefore it is a consistent extension of the standard Galerkin process to the non-symmetric case. Therefore, the Petrov–Galerkin method can always be used for symmetric problems, too. Appendix B demonstrates the effect of the Petrov–Galerkin approach in comparison with the standard Galerkin method for smoothed transfer operators in a simplified two-level method applied to a 1D example.

The principle of the Petrov–Galerkin approach can be generalized and used in combination with any other prolongation smoothing (or energy minimization) method. As an example, the work by Sala and Tuminaro [173] introduces the Petrov–Galerkin approach together with local damping factors for prolongation smoothing as described in (3.16). Having local damping factors for each prolongation or restriction basis function is very helpful in the non-symmetric case to overcome the issue of properly choosing appropriate damping factors ω which could be based more on the diffusive or the convective part of the non-symmetric operator A.

Note that even if it is technically possible to combine the Petrov–Galerkin method with any other prolongation smoothing method, this does not automatically mean that the resulting method is stable for non-symmetric problems. It is important that the transfer operator smoothing strategy also considers the non-symmetry of A_ℓ correctly. In the following a generalized non-symmetric smoothed aggregation approach for non-symmetric problems is proposed based on the Petrov–Galerkin principle.

4.3. Ideal transfer operators

It is assumed that a suitable procedure has been applied to coarsen the fine level matrix graph which effectively partitions the original degrees of freedom into two sets corresponding to a subset of fine-level variables which are transferred to the next coarser level as coarse-level variables (c-points), and the remaining "fine-level only" variables (f-points). So, the c-points are strictly defined as a subset of the original fine-level degrees of freedom. This induces a block partitioning of the system (1.1) given by

$$\begin{pmatrix} \mathrm{A}_{\mathrm{ff}} & \mathrm{A}_{\mathrm{fc}} \\ \mathrm{A}_{\mathrm{cf}} & \mathrm{A}_{\mathrm{cc}} \end{pmatrix} \begin{pmatrix} x_{\mathrm{f}} \\ x_{\mathrm{c}} \end{pmatrix} = \begin{pmatrix} b_{\mathrm{f}} \\ b_{\mathrm{c}} \end{pmatrix}. \tag{4.3}$$

To simplify the presentation the subscripts denoting the level ℓ are skipped.

Remark 4.3.1 (f/c splitting.). One finds such a f/c splitting also in a different context for defining, e.g., special f/c level smoothers (cf. Baker et al. [12]). In this section it is used for motivating the concept of ideal transfer operators (cf. Wiesner et al. [217]).

Assuming A_{ff} to be invertible, A has the corresponding LDU decomposition

$$\begin{pmatrix} \mathrm{A}_{\mathrm{ff}} & \mathrm{A}_{\mathrm{fc}} \\ \mathrm{A}_{\mathrm{cf}} & \mathrm{A}_{\mathrm{cc}} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \mathrm{A}_{\mathrm{cf}}\mathrm{A}_{\mathrm{ff}}^{-1} & I \end{pmatrix} \begin{pmatrix} \mathrm{A}_{\mathrm{ff}} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & \mathrm{A}_{\mathrm{ff}}^{-1}\mathrm{A}_{\mathrm{fc}} \\ 0 & I \end{pmatrix}, \tag{4.4}$$

where $S = \mathrm{A}_{\mathrm{cc}} - \mathrm{A}_{\mathrm{cf}}\mathrm{A}_{\mathrm{ff}}^{-1}\mathrm{A}_{\mathrm{fc}}$ is referred to as the Schur complement.

The above decomposition reveals an ideal two-level additive multigrid scheme where additive indicates that fine level relaxation and coarse level correction can occur in parallel (as opposed to the sequential two-level algorithm in Section 2.4.1). In particular, define

$$R^{\mathrm{opt}} = \begin{pmatrix} -\mathrm{A}_{\mathrm{cf}}\mathrm{A}_{\mathrm{ff}}^{-1} & I \end{pmatrix},\ P^{\mathrm{opt}} = \begin{pmatrix} -\mathrm{A}_{\mathrm{ff}}^{-1}\mathrm{A}_{\mathrm{fc}} \\ I \end{pmatrix} \quad \text{and} \quad \hat{I} = \begin{pmatrix} I \\ 0 \end{pmatrix}. \tag{4.5}$$

One can easily verify that $S = R^{\mathrm{opt}}\mathrm{A}P^{\mathrm{opt}}$,

$$\begin{pmatrix} I & 0 \\ \mathrm{A}_{\mathrm{cf}}\mathrm{A}_{\mathrm{ff}}^{-1} & I \end{pmatrix}^{-1} = \begin{pmatrix} \hat{I}^{\mathsf{T}} \\ R^{\mathrm{opt}} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} I & \mathrm{A}_{\mathrm{ff}}^{-1}\mathrm{A}_{\mathrm{fc}} \\ 0 & I \end{pmatrix}^{-1} = \begin{pmatrix} \hat{I} & P^{\mathrm{opt}} \end{pmatrix}. \tag{4.6}$$

Application of the inverses of the three operators in (4.4) is equivalent to restriction at the c-points (left expression in (4.6)) followed by solution of two systems: A_{ff} which can be interpreted as relaxation and $R^{\mathrm{opt}}\mathrm{A}P^{\mathrm{opt}}$, which is the coarse correction. Finally, the coarse correction is interpolated and added to the relaxation solution (rightmost expression in (4.6)). As this procedure is exact, it converges in one iteration. In general, inverting A_{ff} is not practical. However, this inverse can be approximated with a few relaxation sweeps on A_{ff} as A_{ff} is typically well-conditioned when the c-points are properly chosen.

Remark 4.3.2 (Conditioning of A_{ff}). Classical coarsening schemes with standard strength-of-connection measures give well-conditioned A_{ff} when applied to weakly diagonally dominant matrices. Recent approaches such as compatible relaxation aim to produce well-conditioned A_{ff} in more general settings (cf. Brannick and Falgout [42]).

Remark 4.3.3 (Root-node smoothed aggregation). Root-node smoothed aggregation refers to viewing each standard aggregate produced by the aggregation algorithm (cf. Section 3.3) as being centered around a root node. A natural choice for the root node associated with a new aggregate is the non-aggregated node with id i in the outer loop of Algorithm 4.

The ideal two-level method given by P^{opt} and R^{opt} from (4.5) is our starting point for non-symmetric multigrid transfers. However, these transfers are computationally expensive to compute, expensive to apply, and give rise to dense coarse level discretization matrices. Therefore, it is necessary to instead compute an approximation which is now considered in the following Lemma and subsequent Theorem.

Lemma 4.3.4. *Assume* A *to be a non-symmetric square matrix with a* f/c *splitting as given in* (4.3). *Define*

$$R = \begin{pmatrix} R_{\mathrm{f}} & I \end{pmatrix} \text{ and } P = \begin{pmatrix} P_{\mathrm{f}} \\ I \end{pmatrix}. \tag{4.7}$$

Then, the following matrix transformation holds

$$\begin{pmatrix} \mathrm{A}_{\mathrm{ff}} & \mathrm{A}_{\mathrm{fc}} \\ \mathrm{A}_{\mathrm{cf}} & \mathrm{A}_{\mathrm{cc}} \end{pmatrix} = \begin{pmatrix} I & 0 \\ -R_{\mathrm{f}} & I \end{pmatrix} \begin{pmatrix} \mathrm{A}_{\mathrm{ff}} & E_{\mathrm{fc}} \\ E_{\mathrm{cf}} & \mathrm{A}_{\ell+1} \end{pmatrix} \begin{pmatrix} I & -P_{\mathrm{f}} \\ 0 & I \end{pmatrix} \tag{4.8}$$

with $\mathrm{A}_{\ell+1} = R\mathrm{A}P,\ E_{\mathrm{fc}} = \begin{pmatrix} \mathrm{A}_{\mathrm{ff}}\ \mathrm{A}_{\mathrm{fc}} \end{pmatrix} P$ *and* $E_{\mathrm{cf}} = R \begin{pmatrix} \mathrm{A}_{\mathrm{ff}} \\ \mathrm{A}_{\mathrm{cf}} \end{pmatrix}$.

Proof. Verified by algebraic substitution of $\mathrm{A}_{\ell+1}, E_{\mathrm{fc}},$ and E_{cf} into (4.8). □

Theorem 4.3.5 (cf. Wiesner et al. [217])**.** *Assume a* f/c *splitting such that* A_{ff} *is invertible and* P_{f} *and* R_{f} *in* (4.7) *defined such that* $\mathrm{A}_{\ell+1}$ *is also invertible. Let* M *define a two-level additive multigrid preconditioner obtained by ignoring* E_{fc} *and* E_{cf} *in* (4.8) *given by*

$$M = \begin{pmatrix} I & 0 \\ -R_{\mathrm{f}} & I \end{pmatrix} \begin{pmatrix} \mathrm{A}_{\mathrm{ff}} & 0 \\ 0 & \mathrm{A}_{\ell+1} \end{pmatrix} \begin{pmatrix} I & -P_{\mathrm{f}} \\ 0 & I \end{pmatrix}. \tag{4.9}$$

Then,

$$i) \qquad \sigma\big(M^{-1}\mathrm{A}\big) = \big\{1\big\} \cup \Big\{1 \pm \sqrt{\sigma_j}, \sigma_j \in \sigma\big(\mathrm{A}_{\ell+1}^{-1} E_{\mathrm{cf}} \mathrm{A}_{\mathrm{ff}}^{-1} E_{\mathrm{fc}}\big)\Big\} \tag{4.10}$$

and

$$ii) \qquad \sigma(M^{-1}\mathrm{A}) = \big\{1\big\} \cup \big\{1 \pm \sqrt{1-\sigma_j}, \sigma_j \in \sigma(\mathrm{A}_{\ell+1}^{-1} S)\big\}. \tag{4.11}$$

$\sigma(K)$ is the set of eigenvalues of matrix K and σ_j denotes a single eigenvalue in this set.

Proof. Using (4.8) and (4.9) one finds

$$\delta \mathrm{A} := \begin{pmatrix} I & 0 \\ -R_\mathrm{f} & I \end{pmatrix} \begin{pmatrix} 0 & E_\mathrm{fc} \\ E_\mathrm{cf} & 0 \end{pmatrix} \begin{pmatrix} I & -P_\mathrm{f} \\ 0 & I \end{pmatrix}. \tag{4.12}$$

With the definition (4.12) it is

$$\begin{aligned} M^{-1}\mathrm{A} &= M^{-1}\big(M + \delta\mathrm{A}\big) = I + M^{-1}\delta\mathrm{A} \\ &= I + \begin{pmatrix} I & -P_\mathrm{f} \\ & I \end{pmatrix}^{-1} \begin{pmatrix} \mathrm{A}_\mathrm{ff}^{-1} & 0 \\ 0 & \mathrm{A}_{\ell+1}^{-1} \end{pmatrix} \begin{pmatrix} 0 & E_\mathrm{fc} \\ E_\mathrm{cf} & 0 \end{pmatrix} \begin{pmatrix} I & -P_\mathrm{f} \\ & I \end{pmatrix} \\ &= I + \begin{pmatrix} I & -P_\mathrm{f} \\ 0 & I \end{pmatrix}^{-1} \begin{pmatrix} 0 & \mathrm{A}_\mathrm{ff}^{-1} E_{fc} \\ \mathrm{A}_{\ell+1}^{-1} E_\mathrm{cf} & 0 \end{pmatrix} \begin{pmatrix} I & -P_\mathrm{f} \\ 0 & I \end{pmatrix}. \end{aligned} \tag{4.13}$$

One can then investigate the eigenvalues of

$$\begin{pmatrix} I & \mathrm{A}_\mathrm{ff}^{-1} E_\mathrm{fc} \\ \mathrm{A}_{\ell+1}^{-1} E_\mathrm{cf} & I \end{pmatrix}, \tag{4.14}$$

which is obtained from (4.13) by a similarity transformation. It is well known that matrices of the form

$$\begin{pmatrix} 0 & Y \\ X & 0 \end{pmatrix} \tag{4.15}$$

have eigenvalues given by $\{0\} \cup \{\pm\sqrt{\sigma_j}, \sigma_j \in \sigma(XY)\}$. Note that this relationship holds even when $\sigma(XY)$ includes negative or complex eigenvalues. The number of zero eigenvalues is the difference between the column and row dimensions of X when X has more columns than rows and when X has full row rank and Y has full column rank. By shifting (4.14), one obtains a system of the form given by (4.15) and $i)$ follows in a straight-forward fashion.

To finish the proof, the definitions of E_fc and E_cf are used to recognize that

$$\begin{aligned} E_\mathrm{cf}\mathrm{A}_\mathrm{ff}^{-1} E_\mathrm{fc} &= R\left[\mathrm{A} - \begin{pmatrix} 0 & 0 \\ 0 & S \end{pmatrix}\right] P \\ &= \mathrm{A}_{\ell+1} - S. \end{aligned}$$

Combining this with (4.10) completes the proof of $ii)$. $\square$

When $R_\mathrm{f} = P_\mathrm{f}^\mathsf{T}$, the transformation associated with (4.8) has been studied in a number of works and is often referred to as a hierarchical basis transformation (cf. Axelsson [6, Chapter 9] as well as Vassilevski [201, Section 3.4] along with associated references). This transformation has been used to obtain condition number bounds for $M^{-1}A$, which have a similar spirit to (4.11). As noted above, the most relevant previous results in our context typically assume that $R_\mathrm{f} = P_\mathrm{f}^\mathsf{T}$ and in addition require A to be symmetric positive definite, neither of which is required for the Theorem 4.3.5.

The simple eigenvalue expression given in (4.11) reveals that the two-level additive multigrid iteration is entirely governed by how well the coarse discretization approximates the Schur complement. Specifically, the eigenvalues of $\mathrm{A}_{\ell+1}^{-1}S$ must be near one. If, for example, the eigenvalues of $\mathrm{A}_{\ell+1}^{-1}S$ lie within a circle of radius r centered at $1 + 0i$ (independent of the mesh resolution), then all the eigenvalues of the preconditioned operator, $M^{-1}\mathrm{A}$, also lie within a circle of radius $\sqrt{r}$ centered at $1 + 0i$ (independent of the mesh resolution). This means that $\mathrm{A}_{\ell+1}$ must approximate the behavior associated with both small and large eigenvalues of S. As illustrated in Section 4.4.2, it is the small eigenvalue behavior of S which is hardest to capture in $\mathrm{A}_{\ell+1}$.

Before concluding this section, it is worth noting that the eigenvalues of the two-level iteration operator depend entirely on coarse level quantities, $\mathrm{A}_{\ell+1}$ and S. This greatly simplifies the application of Fourier analysis. In the classical Fourier-based smoothing analysis, an ideal behavior for the coarse level correction (that it annihilates all low frequency errors) is assumed, such that the approximate behavior of a two-level iteration is governed by the smoother's action on high frequency modes. By contrast, (4.9) assumes an ideal smoother. Fourier analysis for model problems can be applied to $\mathrm{A}_{\ell+1}^{-1}S$ and the Fourier transform yields a diagonal matrix. As $\mathrm{A}_{\ell+1}^{-1}S$ is a coarse level matrix, this analysis focuses on low frequency behavior of the coarse approximation. The simplicity of this analysis will be used later to demonstrate limitations of truncated Schur complements as well as indicate how these limitations can be remedied by forcing $\mathrm{A}_{\ell+1}$ to accurately reproduce S's behavior for eigenvectors associated with small eigenvalues.

4.4. A new smoothed aggregation approach

The Petrov–Galerkin approach from Section 4.2.2 alone does not guarantee a good method for non-symmetric problems. For example, a steepest descent method for improving the transfer operator basis functions (as proposed in Mandel et al. [124]) would lead to the same shape of transfer operator basis functions both for the prolongator and the restrictor due to a symmetric iterative minimization process. That is, the Petrov–Galerkin approach would have no effect on the transfer operator basis functions at all.

Motivated by the theoretical considerations in Section 4.3, our novel smoothed aggregation approach is now introduced which utilizes more than one Richardson-like iteration for an effective smoothing of the prolongation operator basis functions and therefore considers the non-symmetry of A_ℓ in the smoothing process. The new transfer operator smoothing method serves as the second key ingredient for flexible state-of-the-art transfer operators for non-symmtric problems on top of the Petrov–Galerkin framework from Section 4.2.2. Thus, the resulting transfers are designed for non-symmetric linear systems arising from convection-dominated flow problems.

Following the steps in Wiesner et al. [217], both the prolongation smoothing and restriction smoothing are explicitly discussed in the following. Similar to the classical smoothed aggregation method one uses a global damping strategy, but considers additional constraints such as a pattern constraint for the sparsity pattern of the transfer operators (cf. objective **(O2)** in Section 3.2.2) or the null space preservation constraint, to satisfy the objective **(O3)** from Section 3.2.2. The pattern constraint for the transfer operator sparsity pattern overcomes the issue of too dense matrix patterns leading to more expensive operator complexities. A user-prescribed sparsity pattern for the transfer operators allows for application-specific variants of transfer operators and makes it easy to avoid problems with (nearly) linear dependent transfer operator basis functions resulting from an energy minimization procedure.

In fact, the Galerkin framework presented here is a generalization of the smoothed aggregation approach which is ideal for non-symmetric systems allowing for application-specific sparsity patterns and providing more flexibility through additional mode constraints.

Remark 4.4.1 (Symmetric problems). Since symmetric problems can be understood as a special case of non-symmetric problems, the presented transfer operator smoothing strategy naturally works for symmetric problems as well.

4.4.1. Galerkin projections and grid transfers

Consider the construction of the fine-level part P_f of the prolongation operator and the fine-level part R_f of the restriction operator from (4.7), (4.8) as well as (4.9) from Section 4.3 via the solution of

$$\mathrm{A}_\mathrm{ff} P_\mathrm{f} = -\mathrm{A}_\mathrm{fc} \text{ and } R_\mathrm{f} \mathrm{A}_\mathrm{ff} = -\mathrm{A}_\mathrm{cf}. \tag{4.16}$$

Remark 4.4.2. Note that the rightmost expression is best thought of in terms of $\mathrm{A}_\mathrm{ff}^\mathsf{T} R_\mathrm{f}^\mathsf{T} = -\mathrm{A}_\mathrm{cf}^\mathsf{T}$.

Clearly, this recovers P^opt and R^opt from (4.5). However, for single-field problems the individual operators A_ff, A_fc and A_cf are often not available from an application and so it is more natural to consider the entire system matrix. First define the spaces

$$\mathcal{P}_0 := \left\{ P = \begin{pmatrix} P_s \\ I \end{pmatrix} : P_s \in \mathbb{R}^{n_\mathrm{f} \times n_\mathrm{c}} \right\} \text{ and } \mathcal{R}_0 := \left\{ R = \begin{pmatrix} R_s & I \end{pmatrix} : R_s \in \mathbb{R}^{n_\mathrm{c} \times n_\mathrm{f}} \right\}, \tag{4.17}$$

where n_f and n_c denote the number of rows in A_ff and A_cc, respectively. Then, consider the solution of

$$\mathrm{A}P = 0 \text{ with } P \in \mathcal{P}_0 \text{ and } R\mathrm{A} = 0 \text{ with } R \in \mathcal{R}_0 \text{ instead of (4.16).} \tag{4.18}$$

Without the constraints imposed by $\mathcal{P}_0$ and $\mathcal{R}_0$, the trivial transfers $P = 0$ and $R = 0$ would be the unique solutions to the systems in (4.18) if A is non-singular. As these trivial transfers do not reside in $\mathcal{P}_0$ and $\mathcal{R}_0$, it is not possible to satisfy (4.18). Instead, (4.18) must be weakened, such that it needs only be satisfied in a subspace. To do this, a Galerkin projection can be formulated with the help of the homogeneous spaces

$$\mathcal{P} := \left\{ P = \begin{pmatrix} P_s \\ 0 \end{pmatrix} : P_s \in \mathbb{R}^{n_\mathrm{f} \times n_\mathrm{c}} \right\} \text{ and } \mathcal{R} := \left\{ R = \begin{pmatrix} R_s & 0 \end{pmatrix} : R_s \in \mathbb{R}^{n_\mathrm{c} \times n_\mathrm{f}} \right\}. \tag{4.19}$$

Projection of (4.18) now yields

$$\mathrm{A}P \perp \mathcal{P} \text{ with } P \in \mathcal{P}_0 \text{ and } R\mathrm{A} \perp \mathcal{R} \text{ with } R \in \mathcal{R}_0, \tag{4.20}$$

where the notation $X \perp \mathcal{Y}$ indicates that for any matrix $Y \in \mathcal{Y}$ one has $Y^\mathsf{T} X = 0$. In this case, the $\perp$ conditions simply state that $\mathrm{A}P$ (and $R\mathrm{A}$) must only be zero at rows (columns) associated with f-points. In other words, $\mathrm{A}_\mathrm{ff} P_\mathrm{f} + \mathrm{A}_\mathrm{fc} = 0$ and $R_\mathrm{f} \mathrm{A}_\mathrm{ff} + \mathrm{A}_\mathrm{cf} = 0$ which corresponds to (4.16), i.e., the optimal Schur complement based grid transfers.

While obtaining Schur complement based transfers via a Galerkin projection may seem convoluted, it does provide an interesting mechanism for formulating other grid transfers. In particular, one can devise $\mathcal{Z}_0$ and $\mathcal{W}_0$ which are subspaces of $\mathcal{P}_0$ and $\mathcal{R}_0$ respectively (along with their corresponding homogeneous counter-parts) and instead consider

$$\mathrm{A}P \perp \mathcal{Z} \text{ with } P \in \mathcal{Z}_0 \quad \text{and} \quad R\mathrm{A} \perp \mathcal{W} \text{ with } R \in \mathcal{W}_0, \tag{4.21}$$

or equivalently

$$\mathrm{A}P \perp \mathcal{Z} \text{ with } P - P^{(0)} \in \mathcal{Z} \quad \text{and} \quad R\mathrm{A} \perp \mathcal{W} \text{ with } R - R^{(0)} \in \mathcal{W}, \tag{4.22}$$

where $P^{(0)} \in \mathcal{Z}_0$ and $R^{(0)} \in \mathcal{W}_0$ are initial guesses and $P - P^{(0)}$ and $R - R^{(0)}$ represent corrections.

As shown in the following, computationally attractive grid transfers can be obtained with a proper choice of $\mathcal{Z}_0$ and $\mathcal{W}_0$. The key is to restrict spaces in a way that reduces the computational costs of constructing and applying grid transfers without causing a significant degradation in associated convergence rates.

4.4.2. Constraining the solution space

Two types of constraints are now introduced. The first reduces the cost of computing, storing, and applying grid transfers by limiting the allowed sparsity patterns, essentially truncating a Schur complement computation. The second ensures that certain desirable properties are maintained by the resulting coarse level discretization matrix $\mathrm{A}_{\ell+1}$. Basically, these constraints are supposed to ensure the objectives **(O2)** and **(O3)** from Section 3.2.2 in the design of multigrid transfer operators.

(C1) *Sparsity pattern limit:* Assume for now that $N_\mathrm{f}^{(\mathrm{p})} \in \mathbb{R}^{n_\mathrm{f} \times n_\mathrm{c}}$ and $N_\mathrm{f}^{(\mathrm{r})} \in \mathbb{R}^{n_\mathrm{c} \times n_\mathrm{f}}$ are given binary matrices in

$$N^{(\mathrm{p})} := \begin{pmatrix} N_\mathrm{f}^{(\mathrm{p})} \\ I \end{pmatrix} \in \mathbb{R}^{n_\ell \times n_{\ell+1}} \quad \text{and} \quad N^{(\mathrm{r})} := \begin{pmatrix} N_\mathrm{f}^{(\mathrm{r})} & I \end{pmatrix} \in \mathbb{R}^{n_{\ell+1} \times n_\ell}. \tag{4.23}$$

Then, define

$$\mathcal{Z} := \Big\{ P \in \mathcal{P} \; : N_{ij}^{(\mathrm{p})} = 0 \;\; \Rightarrow \;\; P_{ij} = 0 \Big\}, \tag{4.24}$$

$$\mathcal{W} := \Big\{ R \in \mathcal{R} \; : N_{ij}^{(\mathrm{r})} = 0 \;\; \Rightarrow \;\; R_{ij} = 0 \Big\}, \tag{4.25}$$

$$\mathcal{Z}_0 := \Big\{ P \in \mathcal{P}_0 : N_{ij}^{(\mathrm{p})} = 0 \;\; \Rightarrow \;\; P_{ij} = 0 \Big\}, \tag{4.26}$$

and

$$\mathcal{W}_0 := \Big\{ R \in \mathcal{R}_0 : N_{ij}^{(\mathrm{r})} = 0 \;\; \Rightarrow \;\; R_{ij} = 0 \Big\}. \tag{4.27}$$

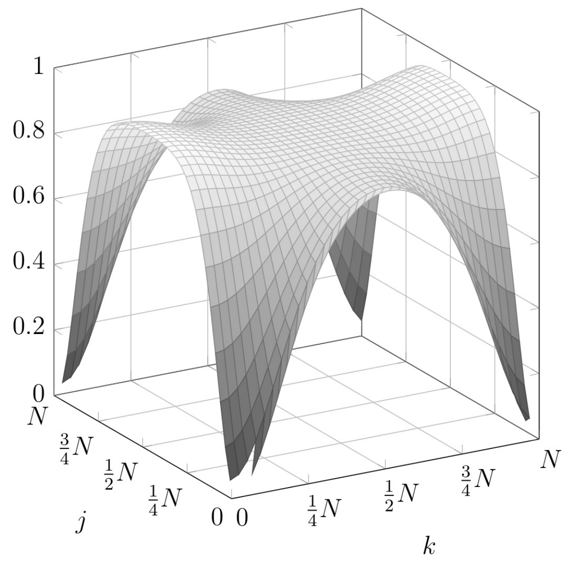


Figure 4.1.: Eigenvalues of $\mathrm{A}_{\ell+1}^{-1}S$ for (4.28) with $\gamma = 0.8$ shown in frequency space for truncated Schur complement based transfers with **(C1)** only.

Briefly, patterns will be chosen to capture large entries in the optimal grid transfers without causing too many non-zeros in the coarse level matrix $\mathrm{A}_{\ell+1}$ in order to consider objective **(O2)** from Section 3.2.2.

A small example can demonstrate the effect of constrained transfer operator patterns to the multigrid method.

Example 4.4.3 (Eigenvalue spectrum of $\mathrm{A}_{\ell+1}^{-1}S$ in frequency space)**.** From (4.11), it is known that the resulting grid transfers should give rise to coarse discretizations, where the eigenvalues of $\mathrm{A}_{\ell+1}$ somehow approximate those of S. Unfortunately, this is often not the case when (4.24)-(4.27) are used. To illustrate this, consider the following partial differential operator

$$u_{xx} + u_{yy} + \gamma u_y \tag{4.28}$$

defined on a unit square with $\gamma > 0$ and periodic boundary conditions. For the discretization a standard 5 point central difference stencil is used on a $3N \times 3N$ mesh. Let the c-points be chosen such that they define a regular $N \times N$ mesh with two f-points between each adjacent c-point in each coordinate direction. Let us further choose the sparsity pattern for the prolongator and restrictor such that each basis function (e.g, column for P and row for R) has 25 non-zeros values corresponding to a 5×5 region with a c-point at the center. Finally, define the grid transfers as the solution of (4.22) where (4.24)-(4.27) define the associated subspaces. Due to the periodic nature of the problem, S and the resulting $\mathrm{A}_{\ell+1}$ have a block circulant structure corresponding to a constant coefficient PDE stencil. This means that the operator $\mathrm{A}_{\ell+1}^{-1}S$ is diagonalized by the Fourier transform. Define

$$W_{kj} = \frac{e^{2\pi i j_1 k_1/N}\, e^{2\pi i j_2 k_2/N}}{\beta_j}$$

where

$$k_1 = 0, ..., N, \quad k_2 = 0, ..., N, \quad j_1 = 0, ..., N, \quad j_2 = 0, ..., N,$$

$$k = k_2 N + k_1 + 1, \;\; j = j_2 N + j_1 + 1,$$

and β_j is chosen such that $||W_{\cdot,j}||_2 = 1$. Here, $W_{\cdot,j}$ describes the j-th column of W. Then, $W^\mathsf{T} \mathrm{A}_{\ell+1}^{-1} S W = \Lambda$ with Λ a diagonal matrix of the eigenvalues of $\mathrm{A}_{\ell+1}^{-1} S$.

While this example is admittedly quite specialized, it allows us to graphically examine the frequency dependent aspects of different grid transfer choices. In particular, Figure 4.1 gives a surface plot of the eigenvalues in frequency space. The middle of the domain corresponds to high frequencies while the corners correspond to low frequency. Corner points are omitted due to the singularity of $\mathrm{A}_{\ell+1}$. The middle of the sides correspond to high frequency in one direction and low frequency in the other.

What the plot in Figure 4.1 reveals is typical for truncated Schur complements. While the high frequency approximation to the Schur complement is adequate, the eigenvalues of the preconditioned linear operator approach zero for low frequencies (as one approaches the corners). This implies that the low frequency behavior of $\mathrm{A}_{\ell+1}$ is not adequate. To prevent this, one must further restrict the Galerkin spaces. In our example, the corners of Figure 4.1 correspond to the constant mode and so it is clear that modes close to the constant in frequency space are problematic. With the knowledge from Section 2.5.2, it is clear that one has to preserve the algebraically smooth low frequency error which typically corresponds to the (near) null space modes of our problems (including the constants). Therefore, one has to further restrict the Galerkin spaces, such that application of $\mathrm{A}_{\ell+1}$ and S to a constant vector correspond exactly. The plotted function would be modified to enforce a value of one in the corners as opposed to approaching zero. Thus, the effect of the truncated Schur complement alone as shown in Figure 4.1 is sub-optimal from the multigrid perspective, which motivates the *mode preservation* or *null space preservation* constraint (cf. **(O3)** from Section 3.2.2).

(C2) *Mode preservation:* Suppose n_{B1} right vectors (i.e., $B_{\ell+1}^{(\mathrm{p})} \in \mathbb{R}^{n_{\ell+1} \times n_{B1}}$) and n_{B2} left vectors (i.e., $B_{\ell+1}^{(\mathrm{r})} \in \mathbb{R}^{n_{B2} \times n_{\ell+1}}$) are given for which one wants $\mathrm{A}_{\ell+1}$'s action to accurately approximate the Schur complement. That is,

$$\mathrm{A}_{\ell+1} B_{\ell+1}^{(\mathrm{p})} \approx S B_{\ell+1}^{(\mathrm{p})} \;\text{ and }\; B_{\ell+1}^{(\mathrm{r})} \mathrm{A}_{\ell+1} \approx B_{\ell+1}^{(\mathrm{r})} S. \tag{4.29}$$

In case of equality in (4.29), it follows that $B_{\ell+1}^{(\mathrm{r})}$ and $B_{\ell+1}^{(\mathrm{p})}$ are left and right eigenvectors of $S \mathrm{A}_{\ell+1}^{-1}$ and $\mathrm{A}_{\ell+1}^{-1} S$ respectively with corresponding eigenvalues of one. Thus, the multigrid preconditioner is ideal for preserving these modes. To enforce (4.29), one instead considers

$$P B_{\ell+1}^{(\mathrm{p})} = P^{\mathrm{opt}} B_{\ell+1}^{(\mathrm{p})} \quad \text{and} \quad B_{\ell+1}^{(\mathrm{r})} R = B_{\ell+1}^{(\mathrm{r})} R^{\mathrm{opt}}. \tag{4.30}$$

Pre-multiplication by $R\mathrm{A}$ on the left expression and post-multiplication by $\mathrm{A}P$ on the right expression yield

$$R\mathrm{A}P B_{\ell+1}^{(\mathrm{p})} = R\mathrm{A}P^{\mathrm{opt}} B_{\ell+1}^{(\mathrm{p})} \quad \text{and} \quad B_{\ell+1}^{(\mathrm{r})} R\mathrm{A}P = B_{\ell+1}^{(\mathrm{r})} R^{\mathrm{opt}} \mathrm{A}P. \tag{4.31}$$

This is equivalent to (4.29) with equality as $\mathrm{A}_{\ell+1} = R\mathrm{A}P$, $S = R^{\mathrm{opt}}\mathrm{A}P$, and $S = R\mathrm{A}P^{\mathrm{opt}}$. These Schur complement relations follow from the fact that $\mathrm{A}P^{\mathrm{opt}}$ (or $R^{\mathrm{opt}}\mathrm{A}$) is zero at f-point rows (or columns) and so this matrix triple product is independent of P_{f} (or R_{f}, respectively).

There are two problems with (4.30) from a practical perspective. The first is that null space vectors $B^{(\mathrm{r})}_{\ell+1}$ and $B^{(\mathrm{p})}_{\ell+1}$ are required on the coarse mesh. The second is that the action of P^{opt} and R^{opt} on the coarse null space must be approximated. It is assumed that the given fine level vectors are near null space vectors, such that

$$\mathrm{A}B^{(\mathrm{p})} \approx 0 \quad \text{and} \quad B^{(\mathrm{r})}\mathrm{A} \approx 0. \tag{4.32}$$

A definition of the near null space is given in Mandel et al. [124] and corresponds to the discrete representation of the zero energy modes of the principal part of the differential operator without any boundary conditions applied (cf. Section 3.4.1). In practice, matrices $B^{(\mathrm{p})}$ and $B^{(\mathrm{r})}$ are supplied by a user, such that their columns should span the left and right near null space. In fact, equality in (4.32) ensures that $B^{(\mathrm{p})} = P^{\mathrm{opt}}B^{(\mathrm{p})}_{\ell+1}$ and $B^{(\mathrm{r})} = B^{(\mathrm{r})}_{\ell+1}R^{\mathrm{opt}}$ with coarse vectors defined by injection. This follows from the fact that $\mathrm{A}P^{\mathrm{opt}}B^{(\mathrm{p})}_{\ell+1}$ and $B^{(\mathrm{r})}_{\ell+1}R^{\mathrm{opt}}\mathrm{A}$ are zero at the f-points. Thus, the final form of the constraints defining the *mode preservation* are

$$PB^{(\mathrm{p})}_{\ell+1} = B^{(\mathrm{p})} \quad \text{and} \quad B^{(\mathrm{r})}_{\ell+1}R = B^{(\mathrm{r})}. \tag{4.33}$$

Before continuing, it should be noted that the given vectors are not exactly null space vectors (especially near boundaries). This means that there is no equality in (4.32) and so (4.29) is only approximated. This is discussed further in Section 4.5.

Combining constraints associated with sparsity patterns and constraints to enforce near null space preservation gives rise to the final form of the Galerkin spaces

$$\mathcal{Z} := \left\{ P \in \mathcal{P} \,:\, N^{(\mathrm{p})}_{ij} = 0 \,\Rightarrow\, P_{ij} = 0,\ PB^{(\mathrm{p})}_{\ell+1} = 0 \right\}, \tag{4.34}$$

$$\mathcal{W} := \left\{ R \in \mathcal{R} \,:\, N^{(\mathrm{r})}_{ij} = 0 \,\Rightarrow\, R_{ij} = 0,\ B^{(\mathrm{r})}_{\ell+1}R = 0 \right\}, \tag{4.35}$$

$$\mathcal{Z}_0 := \left\{ P \in \mathcal{P}_0 : N^{(\mathrm{p})}_{ij} = 0 \,\Rightarrow\, P_{ij} = 0,\ PB^{(\mathrm{p})}_{\ell+1} = B^{(\mathrm{p})} \right\}, \tag{4.36}$$

and

$$\mathcal{W}_0 := \left\{ R \in \mathcal{R}_0 : N^{(\mathrm{r})}_{ij} = 0 \,\Rightarrow\, R_{ij} = 0,\ B^{(\mathrm{r})}_{\ell+1}R = B^{(\mathrm{r})} \right\}, \tag{4.37}$$

which require the sparsity patterns at f-points, $N^{(\mathrm{p})}_{\mathrm{f}}$ and $N^{(\mathrm{r})}_{\mathrm{f}}$, to be supplied along with the near null space vectors, $B^{(\mathrm{p})}$ and $B^{(\mathrm{r})}$. Assuming that there is a feasible solution (i.e., the spaces are not empty), grid transfers are guaranteed to exactly preserve the supplied modes and conform to the specified sparsity patterns. Figure 4.2 is the updated version of Figure 4.1, which includes the revised form of $\mathcal{Z}, \mathcal{Z}_0, \mathcal{W},$ and $\mathcal{W}_0$ where $B^{(\mathrm{r})}$ and $B^{(\mathrm{p})}$ correspond to the constant vector. It is clear that the mode preservation constraint gives rise to an $\mathrm{A}_{\ell+1}$, whose eigenvalues approximate those of S as illustrated in Figure 4.2, which is bounded above and bounded below (far from zero).

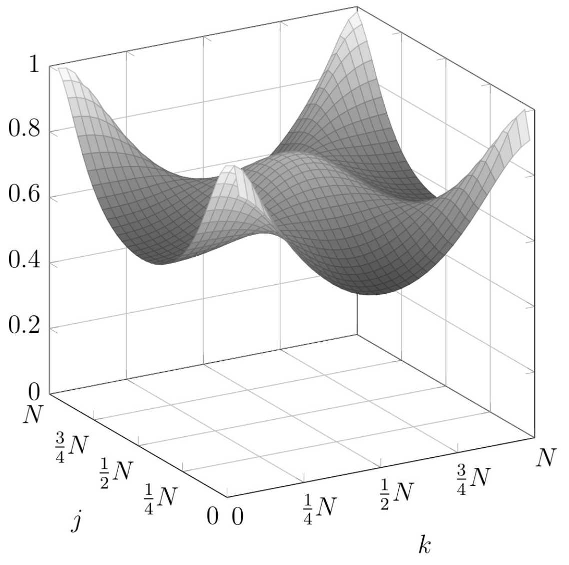


Figure 4.2.: Eigenvalues of $\mathrm{A}_{\ell+1}^{-1}S$ for (4.28) with $\gamma = 0.8$ shown in frequency space for restricted Schur complement based transfers (**(C1)** and **(C2)**).

4.4.3. Solution of the Galerkin system

Now, the solution of the Galerkin system given by (4.22) is considered. Since the ideas also apply for restriction in an obvious way, it is sufficient to describe only the prolongator (cf. Remark 4.4.2).

4.4.3.1. Notation

To do this, a notational change corresponding to a matrix-vector view as opposed to a matrix-matrix style is made. This notational change is only motivated by ease of notation and does not reflect the algorithmic implementation. In particular, an inverted hat accent is introduced to denote the conversion of a matrix to a vector, referred to as matrix squeezing. Specifically,

$$\check{P} = \begin{pmatrix} P_{.1} \\ \vdots \\ P_{.n} \end{pmatrix}, \tag{4.38}$$

where $P_{.j}$ describes the j-th column of P. If $Q = \mathrm{A}P$, it follows

$$\check{Q} = \big(\mathrm{I} \otimes \mathrm{A}\big)\check{P}, \text{ where } \mathrm{I} \otimes \mathrm{A} := \begin{pmatrix} \mathrm{A} & & \\ & \ddots & \\ & & \mathrm{A} \end{pmatrix}. \tag{4.39}$$

This notation allows the *squeezed* version of $\mathcal{Z}$ to be conceptually represented by a matrix, Z, whose columns correspond to an orthonormal set of k spanning vectors. In particular,

$$\delta P \in \mathcal{Z} \;\Rightarrow\; \exists u \in \mathbb{R}^k : \delta\check{P} = Zu. \tag{4.40}$$

If the matrix Z is available, it is possible to re-write

$$\mathrm{A}P \perp \mathcal{Z} \;\text{ with }\; P - P^{(0)} \in \mathcal{Z} \tag{4.41}$$

as a linear system involving the unknown vector u given by

$$Z^{\mathsf{T}}\big(\mathrm{I} \otimes \mathrm{A}\big)\big(Zu + \breve{P}^{(0)}\big) = 0, \tag{4.42}$$

which can be re-arranged to

$$Z^{\mathsf{T}}\big(\mathrm{I} \otimes \mathrm{A}\big)Zu = -Z^{\mathsf{T}}\big(\mathrm{I} \otimes \mathrm{A}\big)\breve{P}^{(0)}. \tag{4.43}$$

Once u is known the final prolongator then is $\breve{P} = Zu + \breve{P}^{(0)}$. Assuming that $\mathcal{Z}$ is non-empty and that A is non-singular, (4.43) has a unique solution and so P is unique. This unique solution is independent of the specific choice of $P^{(0)}$, which follows from the fact that the difference between two initial guesses $\breve{P}^{(0)} - \breve{P}'^{(0)}$ has the form Zv. Thus, the corresponding prolongators obtained via the Galerkin projection satisfy

$$\breve{P} - \breve{P}' = Z(u - u') + Zv. \tag{4.44}$$

However, non-singularity of $Z^{\mathsf{T}}\big(\mathrm{I} \otimes \mathrm{A}\big)Z$ as well as (4.43) imply that $u - u' = -v$ and so $\breve{P} - \breve{P}' = 0$.

4.4.3.2. A projected Richardson-like iteration

A Richardson-style iteration can be applied to (4.43) of the form

$$u^{(s)} = u^{(s-1)} + \omega^{(s-1)}\Big(-Z^{\mathsf{T}}\big(\mathrm{I} \otimes D^{-1}\mathrm{A}\big)\breve{P}^{(0)} - Z^{\mathsf{T}}\big(\mathrm{I} \otimes D^{-1}\mathrm{A}\big)Zu^{(s-1)}\Big) \tag{4.45}$$

with damping parameters $\omega^{(s)}$ and $u^{(0)} = 0$. The D^{-1} is the inverse of the diagonal of A and is used as a simple preconditioner within the Richardson iteration. Multiplying the above equation by Z and adding $\breve{P}^{(0)}$ yields

$$\breve{P}^{(s)} = \Big(I - \omega^{(s-1)}ZZ^{\mathsf{T}}\big(\mathrm{I} \otimes D^{-1}\mathrm{A}\big)\Big)\breve{P}^{(s-1)}, \tag{4.46}$$

where $\breve{P}^{(s-1)} = Zu^{(s-1)} + \breve{P}^{(0)}$. Finally, applying recursion gives

$$\breve{P}^{(s)} = \prod_{k=1}^{s}\Big(I - \omega^{(k-1)}ZZ^{\mathsf{T}}\big(\mathrm{I} \otimes D^{-1}\mathrm{A}\big)\Big)\breve{P}^{(0)}. \tag{4.47}$$

Formula (4.47) shows how $P^{(s)}$ is obtained after s Richardson steps applied to the initial prolongation operator $P^{(0)}$. Furthermore, (4.47) reveals that it is not necessary to explicitly form Z as long as one can apply the operator ZZ^{T} in a matrix-free way.

Before focusing on computational costs in more detail it is worth mentioning that only a rather small number of Richardson improvement iterations are used for practical computations. With the exception of a large aggregation experiment (cf. Section 4.5.2), no more than 3 Richardson iterations for smoothing the grid transfers turn out to be sufficient for most examples. It should also be kept in mind that the Richardson cost is only associated with AMG setup. Thus, it may not be necessary to always repeat this setup when addressing a sequence of related linear systems. In particular, it may be suitable to reuse grid transfers from a previous linear solve within the AMG hierarchy for the next linear system.

4.4.3.3. Practicalities

Now, more practical aspects associated with computing (4.47) shall be discussed. This includes the application of $\mathrm{I} \otimes D^{-1}\mathrm{A}$ to a prolongator, the determination of damping parameters $\omega^{(s)}$, the construction of $P^{(0)}$ as well as the application of ZZ^{T} to a prolongator. Recalling (4.39), it is clear that the first kernel applied to a prolongator P is the same as computing $D^{-1}\mathrm{A}P$ and then applying a matrix *squeeze* to the result. Thus, this is completely equivalent to a matrix-matrix multiply and so is easily provided. The second aspect is discussed further in Section 4.5. At this point, it shall be mentioned that different choices for $\omega^{(s)}$ correspond to different iterative methods. In some cases, they are defined by a user-provided constant (e.g., damped Jacobi). In other cases they may use spectral radius estimates of $D^{-1}\mathrm{A}$ (e.g., polynomial approximation methods) and in still other cases they may involve inner-products of squeezed matrices (e.g., Krylov methods) which is similar to Frobenius norm calculations. The third requirement is an initial feasible point. It is similar to a tentative prolongator in smoothed aggregation (cf. Vaněk et al. [192, 199]), and so, often those techniques can be applied (cf. Section 3.4). In smoothed aggregation, the basic idea is to form the non-zeros within each column of $P^{(0)}$ by injecting each near null space vector to each aggregate. Each aggregate corresponds to a local neighborhood around a c-point such that all f-points are assigned to one and only one aggregate. This procedure is computationally inexpensive and provides good initial guesses, though there are some issues in our context. This includes the fact that the number of near null space vectors must be equal to the number of degrees of freedom per node and that normalization must be applied to obtain the identity operator in (4.7). Both is no problem for typical applications with non-symmetric problems (e.g., convection-diffusion or Navier-Stokes examples). The fourth requirement is the main challenge and it boils down to a procedure for computing $ZZ^{\mathsf{T}}\check{V}$ given a matrix V. As Z is made up of orthonormal columns, it follows that ZZ^{T} is a projection (i.e., $ZZ^{\mathsf{T}} = (ZZ^{\mathsf{T}})^2$ and $ZZ^{\mathsf{T}} = (ZZ^{\mathsf{T}})^{\mathsf{T}}$). Furthermore, it can be decomposed into the *sparsity pattern limit* and *mode preservation*. Therefore, it is rewritten as

$$ZZ^{\mathsf{T}}\check{V} = \Pi_1\Pi_0\check{V}. \tag{4.48}$$

Π_0, the sparsity pattern projection **(C1)**, is equivalent to removing any non-zeros in V which lie outside of the user allowed sparsity pattern $N^{(\mathrm{p})}$. Π_1 represents the mode preservation constraint **(C2)**. Before outlining the most general case, a common special situation is described, when the near null space is only a single constant vector. Then, $\Pi_1\check{V}$ is equivalent to forcing all row sums in V to be zero via a projection (i.e., each entry in the i-th row is adjusted by subtracting the average of the non-zeros in V's i-th row). This can be easily generalized for PDE systems with k degrees of freedom per node and when the near null space is defined by k constant vectors (one associated with each degree of freedom). One essentially applies the same algorithm to the k sub-rows associated with each degree of freedom.

In the more general case, it holds

$$PB^{(\mathrm{p})}_{\ell+1} = B^{(\mathrm{p})} \;\Rightarrow\; (B^{(\mathrm{p})}_{\ell+1})^{\mathsf{T}} P^{\mathsf{T}} = (B^{(\mathrm{p})})^{\mathsf{T}}, \tag{4.49}$$

where the squeezed version is

$$C\check{G} = \check{F} \tag{4.50}$$

with $C = \mathrm{I} \otimes (B^{(\mathrm{p})}_{\ell+1})^{\mathsf{T}}$, $G = P^{\mathsf{T}}$, and $F = (B^{(\mathrm{p})})^{\mathsf{T}}$. If a dense sparsity pattern is permitted for P, $\Pi_1 \check{P}$ could be implemented via the projection $\left(I - C^{\mathsf{T}}\left(CC^{\mathsf{T}}\right)^{-1} C\right)\check{G}$. As CC^{T} is block diagonal, this is equivalent to

$$P^{\mathsf{T}}_{i\star} = P^{\mathsf{T}}_{i\star} - B^{(\mathrm{p})}_{\ell+1}\left[\left(B^{(\mathrm{p})}_{\ell+1}\right)^{\mathsf{T}} B^{(\mathrm{p})}_{\ell+1}\right]^{-1}\left(B^{(\mathrm{p})}_{\ell+1}\right)^{\mathsf{T}} P^{\mathsf{T}}_{i\star}, \tag{4.51}$$

where $P_{i\star}$ is the i-th row of P. This enforces *mode preservation* for a dense sparsity pattern as it updates all non-zeros within each row. It can, however, be easily modified to be appropriate for sparse prolongators. To do this, first construct H_i by taking a subset of rows from $B^{(\mathrm{p})}_{\ell+1}$, where the subset row indices are given by the nonzero columns in $N^{(\mathrm{p})}_{i\star}$. Then, replace (4.51) with

$$\tilde{P}^{\mathsf{T}}_{i\star} = \tilde{P}^{\mathsf{T}}_{i\star} - H_i[H_i^{\mathsf{T}} H_i]^{-1} H_i^{\mathsf{T}} \tilde{P}^{\mathsf{T}}_{i\star}, \tag{4.52}$$

where $\tilde{P}_{i\star}$ refers to the sub-vector of permitted non-zeros in the i-th row of P. While (4.52) might appear costly, it requires only inner products between sparse prolongator rows and null space vectors as well as inverting linear systems which are $n_B \times n_B$ where n_B is the number of null space vectors. There are a few technical issues in the general case which must be addressed with respect to invertability of $H_i^{\mathsf{T}} H_i$ and exactly satisfying (4.50). They are just mentioned here as they are discussed in detail in Olson et al. [153]. In particular, each prolongator row has n_B constraints. If these constraints are linearly dependent, then a pseudo-inverse must be used. Further, if the row rank associated with these constraints is s, then there must be s non-zeros in the associated prolongator row. Otherwise constraints are only satisfied in a least-squares sense.

Algorithm 8 summarizes the final algorithm. While matrix squeezing is retained for notational purposes, it is not actually needed within a code as Π_0 and Π_1 can be implemented to act directly on the unsqueezed matrix.

Before concluding, a discussion about expense is appropriate. Clearly, multigrid iteration costs (e.g., V-cycle times) should be comparable to other AMG approaches employing similar sparsity patterns. Thus, only setup costs need to be evaluated. In particular, if $s = 1$ in (4.47), $\omega^{(0)} = \frac{4}{3\rho(D^{-1}\mathrm{A})}$, and the $Z^T Z$ step is omitted, then the Galerkin procedure is identical to smoothed aggregation (see Section 3.5.1). The $Z^{\mathsf{T}} Z$ step is typically quite inexpensive (especially for PDE systems where the near null space corresponds to a representation of a set of constant functions with one for each physical variable) and so it is clear that the Galerkin projection construction is comparable to that of smoothed aggregation when $s = 1$. As one might expect (by comparison with smoothed aggregation), very few Richardson iterations are typically required. This is due to the local nature of the truncated Schur complement (or sparsity pattern constraint) and the fact that mode preservation further restricts the search space. Thus, the Galerkin framework provides generality over smoothed aggregation. This generality can be used to improve robustness and address application-specific enhancements.

Algorithm 8: Projected preconditioned Richardson iteration.

```
Procedure PRichardson($P^{(0)}$, $N_\ell^{(\mathrm{p})}$, $B_{\ell+1}^{(\mathrm{p})}$, $B^{(\mathrm{p})}$, A, $s$)
    Build constraint matrix (operating only on allowed non-zeros)
    Compute $C \leftarrow C\left(N^{(\mathrm{p})}, B^{(\mathrm{p})}\right)$
    Factor $CC^{\mathsf{T}}$
    Calculate $\Pi_1 \leftarrow \Pi_1\left(C, CC^{\mathsf{T}}\right)$
    Perform $s$ smoothing sweeps on $P^{(0)}$
    for $k = 1 \ldots s$ do
        Determine damping factor $\omega^{(k)}$
        Compute $\omega^{(k)} \leftarrow \omega\big(A, P^{(k)}\big)$
        Determine smoothing part without constraints
        $T_1 \leftarrow D^{-1}AP^{k-1}$
        Remove entries outside of $N_\ell^{(\mathrm{p})}$
        $\breve{T}_2 \leftarrow \Pi_0 \breve{T}_1$
        Incorporate near null space preservation constraint
        $\breve{T}_3 \leftarrow \Pi_1 \breve{T}_2$
        Perform prolongator smoothing
        $P^{(k)} \leftarrow P^{(k-1)} - \omega^{(k)} T_3$
    end
    return $P^{(s)}$
```

4.4.4. Schur complement based sparsity pattern strategy

One interesting aspect of the Galerkin grid transfer algorithm is that it can be employed with any sort of sparsity pattern. This makes our approach highly flexible in contrast to most AMG methods, where the sparsity pattern of the grid transfer is closely tied to the specific algorithm for generating grid transfer basis functions. This means that one is free to choose from a variety of possible sparsity pattern methods what makes the resulting transfers highly flexible for adaptions to problem-specific requirements. These can be developed independently of the grid transfer construction phase and they can directly target the grid transfer pattern as opposed to most traditional approaches, where sparsity patterns are indirectly determined by strong and weak connection decisions in the matrix A. In fact, the sparsity pattern allows for maximum control of the transfer operator basis functions on the level of degrees of freedom. The ability to define explicitly the sparsity pattern of the transfer operators is a property of the transfer operator strategy which finds more and more attention (cf. Falgout and Schroder [63], Schroder [174]).

Since the Galerkin projection method introduced in Section 4.4.1 approximates a limited Schur complement, a related approach is highlighted to obtain sparsity patterns. Our purpose here is not necessarily to champion a new sparsity pattern algorithm, but to instead demonstrate the flexibility of the Galerkin Schur complement approach. The main idea is to first generate a crude prolongator P^{crude} using Galerkin projection but with very few restrictions on spaces. Large entries in the crude approximation can then be used to determine a sparsity pattern.

Specifically, P^{crude} is computed by applying the Algorithm 8 but taking $\mathcal{Z}, \mathcal{W}, \mathcal{Z}_0,$ and $\mathcal{W}_0$ to be $\mathcal{P}, \mathcal{R}, \mathcal{P}_0,$ and $\mathcal{R}_0$ from (4.17) and (4.19). That is, sparsity pattern and mode preservation constraints are ignored. As discussed earlier, this would reproduce the ideal Schur complement transfers if the Richardson iteration is run to convergence. For P^{crude}, however, only a few sweeps are used. This limits cost and total memory consumption as the sparsity pattern of P^{crude} resembles $|\mathrm{A}|^s|P^{(0)}|$ when s Richardson sweeps are performed.

The sparsity pattern is then generated via

$$(N_{\text{f}}^{(\text{p})})_{ij} = \mathcal{F}(P_{\text{f}}^{\text{crude}}, i, j), \tag{4.53}$$

where $\mathcal{F}$ is a filter which could correspond to a threshold strategy, e.g.,

$$\mathcal{F}(P, i, j) = \begin{cases} 1 & \text{if } |P_{ij}| \geq \varepsilon \\ 0 & \text{else.} \end{cases}$$

Unfortunately, the threshold parameter ε is problem dependent and a priori not known. A flexible strategy is used for our tests, where non-zeros in P^{crude} are sorted based on absolute values. The sparsity pattern coincides with the $\mu n_\ell \ell^d$ largest entries, where $\mu > 0$ is a user-defined factor, ℓ is the multigrid level index, n_ℓ is the number of prolongator rows, and $d \in \{2, 3\}$ is the problem dimension.

The idea is to allow the level index to affect the filter while not fixing the number of non-zeros per row. A more restrictive filter may significantly reduce the operator complexity (see Definition 3.2.1). A weaker filter on coarse levels, however, can improve convergence without too much cost. In general the operator complexity for SchurComp transfer operators (see Table 4.1 and Section 4.4) with above pattern method ($\mu > 1$) is higher than for Emin transfer operators. The user, however, can control the resulting AMG operator complexity for SchurComp by a suitable choice of the parameter μ and can often significantly improve the AMG's convergence rate with only a slightly higher operator complexity (see Section 4.5).

Obviously, there are many other possible sparsity patterns choices. For example, experiments have been made with variants which just use μ non-zeros per row in the prolongation operator without any dependency of the multigrid level ℓ. A relatively straight-forward algorithm is chosen for simplicity as our intention is to highlight the flexibility associated with Galerkin projection prolongators as opposed to strongly advocating a specific sparsity pattern choice.

4.5. Numerical examples

Numerical results are reported to compare the Galerkin Schur complement algorithm from Section 4.4.3 with existing transfer operator strategies, the most simple being plain aggregation (PA-AMG), which uses aggregate-wise constant prolongation and restriction operator basis functions. Smoothed aggregation (SA-AMG) is more sophisticated as the basis functions of the PA-AMG prolongation operator are modified using a smoothing sweep. SA-AMG generally has better convergence properties for symmetric positive definite problems (cf. Vaněk et al. [192, 199]),

PA-AMG	Plain aggregation (aggregate-wise constant basis functions)
Emin	Transfer operators smoothing for non-symmetric problems as introduced in Sala and Tuminaro [173]
SchurComp(s, μ)	Galerkin Schur complement transfers with s transfer operator smoothing iterations in (4.47) and μ as a sparsity pattern threshold (see Section 4.4.3)

Table 4.1.: Overview of multigrid transfer operator strategies.

but it is not designed for non-symmetric problems as restriction is just defined as $R = P^{\mathsf{T}}$. The "Emin" transfer operators from Sala and Tuminaro [173] combine the Petrov–Galerkin approach from Section 4.2.2 with local transfer operators smoothing as illustrated in Section 3.5.2 and turn out to be very effective for many non-symmetric problems. Table 4.1 gives an overview of the transfer operator strategies that are used for the numerical experiments.

All tests use GMRES with one AMG V-cycle as a preconditioner. For all tested transfer operator strategies the same aggregation algorithm generates f/c splittings, where c points are aggregated root nodes. Within the Galerkin Schur complement (SchurComp) schemes the Richardson method uses a line search strategy to determine the $\omega^{(k-1)}$ parameters from (4.47). For each multigrid level, $\omega^{(k-1)}$ solves the quadratic minimization problem

$$\min_{\omega^{(k-1)}} \left\| \big(\mathrm{I} \otimes \mathrm{A}\big)\Big(I - \omega^{(k-1)}\Pi_1\Pi_0\big(\mathrm{I} \otimes D^{-1}\mathrm{A}\big)\Big)\breve{P}^{(k-1)} \right\|_2^2, \tag{4.54}$$

which is connected to GMRES as it minimizes a residual associated with (4.43).

Similar to Emin, restriction is separately computed using the same algorithms and the tentative prolongator is always used as an initial guess for the Galerkin Schur complement.

4.5.1. Double-glazing example

In Elman et al. [58] the double glazing problem is introduced as a model for the temperature distribution in a two-dimensional cavity with one external "hot" wall. A scalar convection-diffusion equation is considered as given by

$$-\kappa\Delta u(\boldsymbol{x}) + \boldsymbol{a}(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}) = b(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{4.55}$$

$$u(\boldsymbol{x}) = g(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Gamma_{\mathsf{D}}, \tag{4.56}$$

$$\frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}) = h(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Gamma_{\mathsf{N}}, \tag{4.57}$$

where $u(\boldsymbol{x})$ is the solution. The diffusion coefficient is $\kappa > 0$ and the convective velocity is $\boldsymbol{a}(\boldsymbol{x}) \in \mathbb{R}^d$ where d denotes the dimension of the problem (i.e., $d \in \{2, 3\}$). The functions $f, g,$ and h define forcing functions and boundary conditions and $\boldsymbol{n}$ denotes an outward-pointing normal vector on Γ.

For the double glazing problem the wind $\boldsymbol{a}(x, y) = \big(2y(1 - x^2), -2x(1 - y^2)\big)$ determines a recirculating flow with streamlines $\Big\{\boldsymbol{x} = \big(x, y\big) \;\Big|\; (1 - x^2)(1 - y^2) = \text{const}\Big\}$. The domain is

	$\kappa = \frac{1}{400}$		$\kappa = \frac{1}{1600}$		$\kappa = \frac{1}{6400}$	
transfer operators	64×64	128×128	64×64	128×128	64×64	128×128
PA-AMG	34 (1.12)	47 (1.12)	54 (1.12)	67 (1.12)	91 (1.12)	117 (1.12)
Emin	20 (1.13)	18 (1.13)	38 (1.13)	33 (1.13)	71 (1.13)	68 (1.13)
SchurComp (1, 1.0)	25 (1.12)	41 (1.12)	38 (1.13)	60 (1.13)	72 (1.14)	98 (1.14)
SchurComp (2, 2.0)	13 (1.19)	22 (1.17)	24 (1.21)	33 (1.17)	58 (1.22)	48 (1.19)
SchurComp (3, 3.0)	11 (1.26)	16 (1.23)	18 (1.30)	20 (1.24)	37 (1.30)	17 (1.26)
SchurComp (4, 3.0)	12 (1.26)	15 (1.23)	16 (1.30)	20 (1.24)	36 (1.30)	16 (1.26)

Table 4.2.: Double-glazing example – Number of GMRES iterations (and operator complexity) for different dual-biased meshes and varying viscosity κ. 1 damped Gauss-Seidel sweep ($\omega = 0.8$) is used for pre- and post-smoothing and a direct solver on coarsest mesh .

	$\kappa = \frac{1}{400}$		$\kappa = \frac{1}{1600}$		$\kappa = \frac{1}{6400}$	
transfer operators	64×64	128×128	64×64	128×128	64×64	128×128
PA-AMG	18 (1.12)	26 (1.12)	27 (1.12)	35 (1.12)	43 (1.12)	57 (1.12)
Emin	13 (1.13)	11 (1.13)	22 (1.13)	21 (1.13)	36 (1.13)	40 (1.13)
SchurComp (1, 1.0)	14 (1.12)	22 (1.12)	21 (1.13)	31 (1.13)	45 (1.14)	52 (1.14)
SchurComp (2, 2.0)	10 (1.19)	14 (1.17)	15 (1.21)	22 (1.17)	32 (1.22)	41 (1.19)
SchurComp (3, 3.0)	8 (1.26)	10 (1.23)	12 (1.30)	13 (1.24)	28 (1.30)	18 (1.26)
SchurComp (4, 3.0)	8 (1.26)	10 (1.23)	11 (1.30)	13 (1.24)	22 (1.30)	16 (1.26)

Table 4.3.: Double-glazing example – Number of GMRES iterations (and operator complexity) for different dual-biased meshes and varying viscosity κ. 3 damped Gauss-Seidel sweeps ($\omega = 0.8$) are used for pre- and post-smoothing and a direct solver on coarsest mesh .

	32×32	64×64	128×128	256×256
PA-AMG	30 (1.12)	43 (1.12)	57 (1.12)	67 (1.13)
Emin	26 (1.13)	36 (1.13)	40 (1.13)	45 (1.13)
SchurComp (1, 1.0)	25 (1.14)	45 (1.14)	52 (1.14)	67 (1.15)
SchurComp (2, 2.0)	23 (1.22)	32 (1.22)	41 (1.19)	62 (1.20)
SchurComp (3, 3.0)	24 (1.30)	28 (1.30)	18 (1.26)	32 (1.26)
SchurComp (4, 3.0)	23 (1.30)	22 (1.30)	16 (1.26)	24 (1.26)

Table 4.4.: Double-glazing example – Number of GMRES iterations (and operator complexity) with viscosity $\kappa = \frac{1}{6400}$ for different dual-biased meshes. An AMG preconditioner with 3 damped Gauss-Seidel sweeps ($\omega = 0.8$) and a direct solver on the coarsest mesh is used within GMRES.

the open set $\Omega = \left(-1,1\right)^2$. The source term b is zero, $\kappa \in \left\{\frac{1}{400}, \frac{1}{1600}, \frac{1}{6400}\right\}$, and

$$g(x,y) = \begin{cases} 1 \text{ for } y = 1 \\ 0 \text{ for } y = -1 \\ 0 \text{ for } x = \pm 1 \text{ and } y \neq 1, \end{cases} \tag{4.58}$$

describes the Dirichlet boundaries. These boundary conditions give rise to discontinuities at the hot wall corners. Bi-linear finite elements are utilized with SUPG stabilization.

If someone is interested in resolving the behavior at boundaries or corners without a too fine overall mesh, a regular quadrilateral mesh may not be optimal. For this reason a dual-biased mesh (bias factor 1.05) is used resulting in a subsequently refined mesh at the boundaries to properly resolve the solution at the boundaries and corners. The coarsest level matrix is fixed to be < 100 degrees of freedom which leads to 3 or 4 multigrid levels (depending on the mesh and the chosen transfer operator strategy). A damped symmetric Gauss–Seidel ($\omega = 0.8$) method is used as level smoother on the fine and intermediate levels and a direct solver is applied on the coarsest level. The fine level null space is approximated by a constant vector. For the results in Tables 4.2, 4.3 and 4.4, the user-given parameter μ defines the average number of non-zeros per row in the prolongator P and is constant over all multigrid levels. Convergence is declared when a relative residual reduction of 10^{-9} occurs. Comparing Tables 4.2, 4.3 and 4.4 one can see that the number of GMRES iterations is decreasing with better level smoothing, but the overall behavior of the different transfer operator strategies is not affected by the level smoother. Depending on the aggregate size a rather small number of transfer operator smoothing sweeps is sufficient for a significant reduction of GMRES iterations with only a moderate increase of the operator complexity. For example, SchurComp(4, 3.0) has an AMG operator complexity which is about twelve percent higher than that of Emin on a 256×256 mesh (see Table 4.4). Thus, one would expect SchurComp's cost per iteration to be about twelve percent higher than that of Emin. However, SchurComp requires about half as many iterations as Emin and so this higher cost per iteration would be easily offset.

Overall, the SchurComp(1, 1.0) method (with only one Richardson iteration and on average one nonzero per row) is generally inferior to Emin. This is partially due to the sparsity pattern of the prolongator which resembles that of PA-AMG. The strength of the SchurComp approach is that it can be used with different sparsity patterns and prolongator improvement iterations. The SchurComp method can be adjusted with only a very minor increase in AMG operator complexity such that it converges noticeably faster than Emin.

4.5.2. Aggressive coarsening

This example aims to study the effect of prolongator improvement (i.e., Richardson) iterations when using aggressive coarsening. A 2D convection diffusion problem is used, where $\boldsymbol{a}(x,y)$ is defined as

$$\boldsymbol{a}(x,y) = \begin{cases} 10 & \text{if } |y| < 0.1, \\ 0 & \text{else}, \end{cases} \tag{4.59}$$

for $(x,y) \in \Omega = [-1,1]^2$ with homogeneous Dirichlet boundaries. With $\kappa = 1.0$ one obtains a moderately non-symmetric linear system. In Figure 4.3 the solution using stabilized bi-linear finite elements is shown over a quadrilateral 72×72 mesh. In order to resolve the behavior within

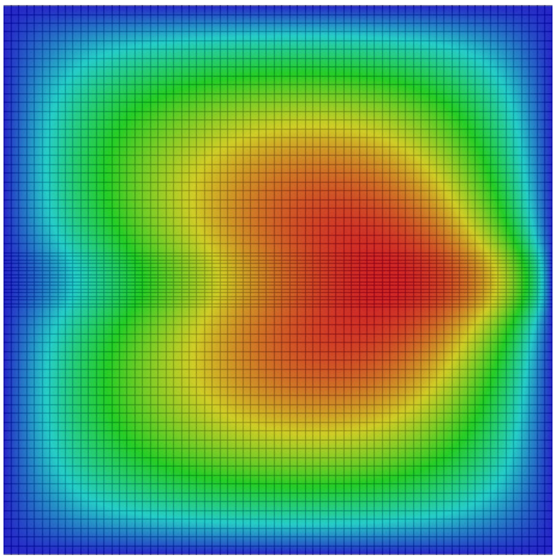
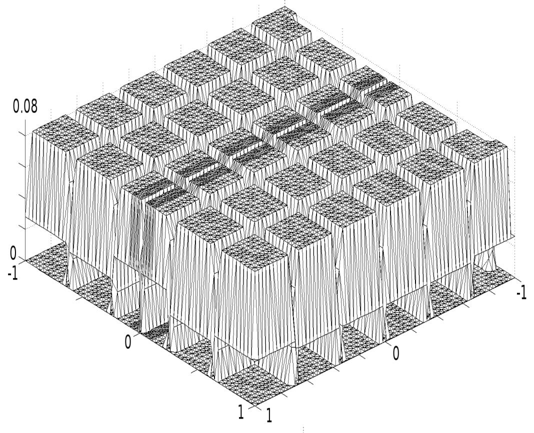


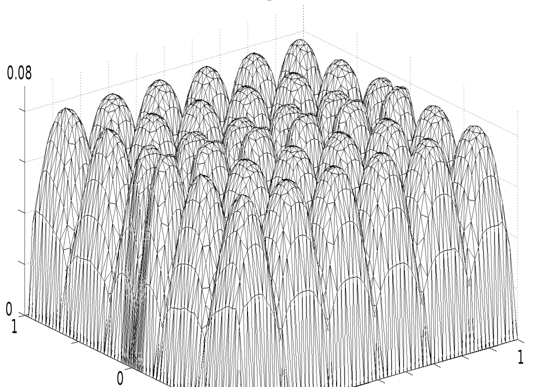
Figure 4.3.: Aggressive coarsening example – Solution of convection diffusion problem on 72×72 quadrilateral mesh with stabilized bi-linear finite elements.

the convective sub-domain the mesh is refined in y-direction for $-0.1 \leq y \leq 0.1$. Figure 4.4a visualizes the prolongator basis functions both for Emin and for SchurComp with 8 prolongator Richardson iterations, when aggregates of size 12×12 are constructed. Figure 4.4b plots a cut through a prolongator basis function generated by Emin and SchurComp along the x-axis for fixed $y = 0.05$ (convective region) and $y = 0.4$ (diffusive region). The figures illustrate the symmetric and smooth shape of the SchurComp basis functions for the purely diffusive region. They also show an asymmetric SchurComp basis function for the convective region which is slightly shifted in the direction of convection. This shifting of the basis functions in the convective direction is similar to that found within basis functions for some black-box matrix-dependent schemes, such as the ones given by Zeeuw [233]. It has been found that these convection-shifted basis functions tend to produce stable coarse grid discretization matrices in the context of black-box multigrid methods. One can find this as well in the context of the SchurComp method, where coarse grid stability was not an issue for the numerical experiments (see also Appendix B). Emin uses only one prolongator smoothing sweep (as the method is only defined for one prolongator smoothing sweep). Especially for big aggregates, the prolongator basis functions for Emin are very similar to the basis functions of the non-smoothed prolongation operator. In contrary to Emin, the SchurComp method allows for more smoothing iterations of the prolongator basis functions.

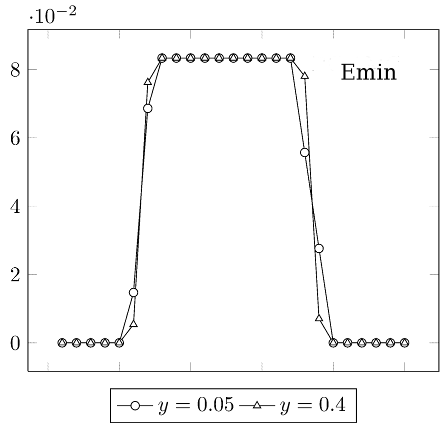
For the numerical test example bilinear finite elements with SUPG stabilization are used on a 648×648 mesh which corresponds to the mesh from Figure 4.3 being refined by a factor 9 in each coordinate direction. In Table 4.5 the number of GMRES iterations and the operator complexities for different transfer operator strategies and aggregation routines are given. GMRES is supposed to be converged, when the relative residual is reduced by a factor of 10^{-11}. The effect of regular ideal aggregates (aggregate size: 3×3), irregular aggregates using random root nodes and graph aggregation as well as regular aggregates of size 6×6 is compared. For the 3×3 and the irregular aggregates the multigrid setup routine builds 5 multigrid levels, whereas for the 6×6 aggregates

(a) Shape of Emin and SchurComp (8 Richardson iterations) prolongator basis functions.





(b) Cut through Emin and SchurComp (8 Richardson iterations) prolongator basis function at $y = 0.05$ (moderate convective) and $y = 0.4$ (diffusive only).



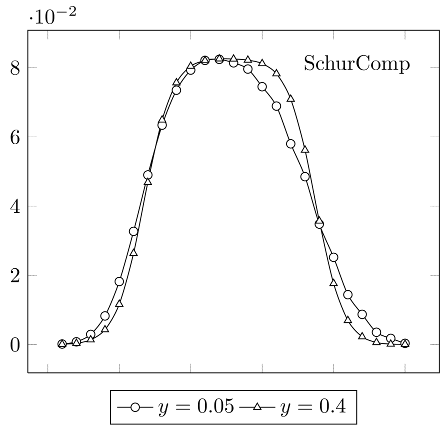


Figure 4.4.: Aggressive coarsening example – Moderate non-symmetric convection-diffusion problem with aggressive coarsening. Comparison of prolongator basis functions generated by Emin and SchurComp (aggregate size: 12×12 nodes).

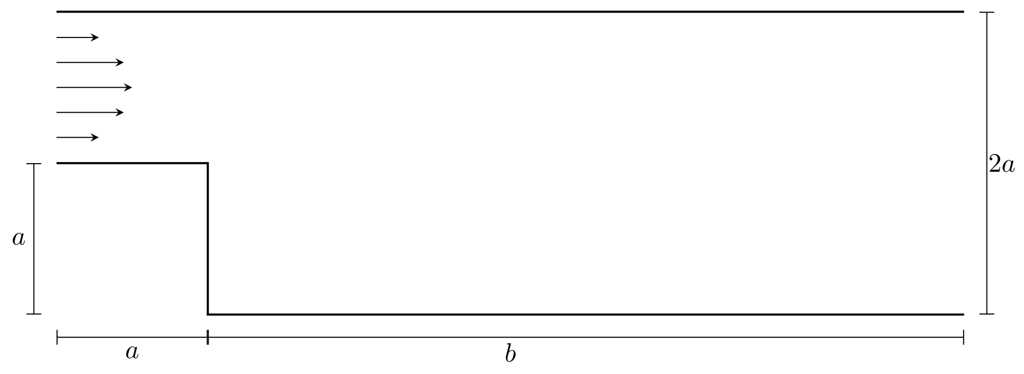
	648×648 mesh					
aggregate size	3×3		irregular		6×6	
transfer operator	GMRES iterations	operator complexity	GMRES iterations	operator complexity	GMRES iterations	operator complexity
PA-AMG	124	1.12530	194	1.06876	165	1.02857
Emin	34	1.12766	56	1.11653	73	1.02861
SchurComp (2, 2.0)	41	1.14536	42	1.08926	60	1.02996
SchurComp (3, 2.0)	41	1.14828	41	1.08991	56	1.03004
SchurComp (3, 3.0)	28	1.17900	28	1.13054	48	1.03330
SchurComp (4, 3.0)	28	1.18033	26	1.13118	45	1.03341
SchurComp (4, 4.0)	22	1.23926	27	1.17314	43	1.06598
SchurComp (5, 4.0)	22	1.24056	27	1.17377	40	1.06603
SchurComp (5, 5.0)	22	1.34273	27	1.21202	39	1.06718
SchurComp (6, 4.0)	22	1.34467	26	1.17404	38	1.06612
SchurComp (6, 5.0)	23	1.41455	27	1.21361	37	1.06733

Table 4.5.: Aggressive coarsening example – Comparison of GMRES iterations and operator complexity for regular ideal aggregates (aggregate size: 3×3 and 6×6) and irregular aggregates (using random root nodes and graph aggregation) on a 648×648 mesh.

we obtain 4 multigrid levels. As pre- and post-smoother 1 Gauss-Seidel sweep ($\omega = 0.8$) is applied on the finest and intermediate multigrid levels. A direct solver is used on the coarsest level. The user-given parameter μ for the pattern strategy within the SchurComp method defines the average number of non-zeros per row in the prolongator P and is constant over all multigrid levels.

A closer look at the results of Table 4.5 shows the difference between SchurComp and Emin. Especially for big aggregates (e.g. 6×6) SchurComp benefits from a higher number of prolongator improvement iterations to resemble the smooth solution of the problem. The prolongator sparsity pattern strategy gives the user full control over the operator complexity which is in general very low when using aggressive coarsening. With the irregular aggregates and randomly picked root nodes one can mimic the effect of less good aggregates as they often arise in parallel multigrid. When comparing the numbers of the ideal perfect 3×3 aggregates with the results for the irregular aggregates, the SchurComp methods are somewhat insensitive to the use of poor/irregular aggregates. Once again, this is due to the flexibility of SchurComp with respect to the number of Richardson iterations and the sparsity pattern which can be augmented to compensate for less than ideal aggregates. For small aggregates the choice of an appropriate pattern is in fact more important than a high number of prolongator improvement iterations.

Overall, SchurComp can converge in less than half as many iterations as that required for Emin in the case of irregular aggregates and 6×6 aggregates, but requires slightly higher complexities (generally under ten percent) which translates into a slightly higher cost per iteration.



(a) Geometry details.

mesh	a	b	#DOFs	description
coarse	1	15	5932	coarse mesh for tests with Reynolds numbers < 1500
medium	1	15	23062	medium mesh for tests with Reynolds numbers < 1500
fine	1	15	90922	fine mesh for tests with Reynolds numbers < 1500
medium2	1	30	45202	medium mesh with elongated geometry for tests with Reynolds number > 1500

(b) Mesh details and geometry parameters.

Figure 4.5.: Backward facing step example – Geometry and mesh details.

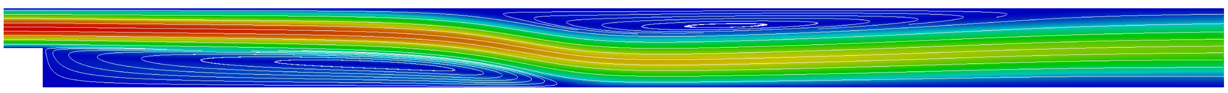


Figure 4.6.: Backward facing step example – Velocity streamlines for Reynolds number Re=1600.

4.5.3. Backward facing step

A linear problem is considered arising from discretization of the incompressible Navier-Stokes equations on a backward facing step domain. The details on modeling and discretization of flow problems is far beyond the scope of this thesis, but the interested reader may refer to the textbook by Donea and Huerta [52]. Figure 4.5 gives details about the geometry and mesh details used for our example whereas Figure 4.6 shows a velocity solution corresponding to $Re = 1600$ employing a equilateral discretization with stabilized finite elements.

Without giving further details the resulting linear equations have the block form

$$\begin{pmatrix} F & D^T \\ D & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{4.60}$$

where F, D^T and D are the discrete momentum, gradient and divergence entities, respectively. The $\mathbf{u}$ denotes the nodal vector containing the discrete velocity components in x- and y-direction. The p represents the associated pressure values in each node.

Note that our primary interest is the behavior of the smoothed transfer operators for typical linear systems arising in flow problems. Let $\mathbf{u}^i$ denote the nodal solution vector for the velocity

Reynolds number	800			1250			1600
mesh	coarse	medium	fine	coarse	medium	fine	medium2
PA-AMG	10 (1.11)	14 (1.12)	22 (1.12)	10 (1.11)	13 (1.12)	20 (1.12)	13 (1.12)
Emin	7 (1.14)	6 (1.14)	9 (1.13)	7 (1.14)	6 (1.14)	– (1.13)	6 (1.14)
SchurComp (1, 2.0)	8 (1.13)	8 (1.15)	12 (1.15)	7 (1.14)	8 (1.15)	11 (1.15)	8 (1.15)
SchurComp (2, 2.0)	8 (1.13)	8 (1.15)	11 (1.15)	7 (1.14)	8 (1.15)	10 (1.15)	7 (1.15)
SchurComp (3, 2.0)	8 (1.13)	8 (1.15)	11 (1.15)	7 (1.14)	8 (1.15)	10 (1.15)	7 (1.15)

Table 4.6.: Backward-facing step example: number of outer GMRES iterations (and operator complexity) for different Reynolds numbers and meshes. 1 sweep with damped Gauss-Seidel ($\omega = 1.0$) as pre- and post-smoother and a direct solver on the coarsest level. '–' denotes no convergence within 50 iterations.

degrees of freedom in the last Newton iteration before convergence. This way, one can avoid additional difficulties associated with the incompressibility constraint by considering only the solution of $F\big(\mathbf{u}^{(i-1)}\big)\mathbf{u}^{(i)} = f$ for the last nonlinear Newton iteration i before convergence of the nonlinear iteration.

A multigrid preconditioner is used where the coarsest level size is set to < 50 degrees of freedom. Convergence for GMRES is declared when the relative residual is reduced by 10^{-6}. On the finest and all intermediate multigrid levels, 1 sweep with damped Gauss-Seidel is employed for both pre-relaxation and post-relaxation. A direct solver is used for the coarsest level. For this PDE system the near null space corresponds to two constant vectors (one associated with each velocity degree-of-freedom). The pattern strategy from section 4.4.4 is used in conjunction with the SchurComp algorithm for generating grid transfers. Since it is a 2-dimensional example with two null space vectors the user-specified parameter μ for the transfer operator pattern is chosen to be two. That is, the average number of nonzeros per row in the prolongator P is two on the finest level and increasing on the intermediate and coarsest level depending on the multigrid level. This leads to slightly increased operator complexities for the SchurComp variants compared with Emin, as the average number of nonzeros per row in a Emin prolongator is between one and two over all multigrid levels.

Table 4.6 lists the GMRES iterations associated with the linear system for different Reynolds numbers. This example mimics an ideal situation for multigrid methods. With an appropriate level smoothing the numbers turn out to be independent of the Reynolds number and the underlying mesh. Using a rather low number of Richardson iterations for prolongator improvement together with a reasonable choice for the prolongator pattern one finds SchurComp to perform in a similar fashion to Emin in a realistic multigrid setting on an equilateral regular mesh. In particular, the operator complexities and total iterations required for convergence are nearly identical for the two methods.

4.6. Conclusion

In this chapter a new framework has been presented for generating transfer operators which are appropriate for non-symmetric problems arising from problems with convective character providing flexibility for further application-specific developments. It has been demonstrated that the

Reynolds number	800			1250			1600
mesh	coarse	medium	fine	coarse	medium	fine	medium2
PA-AMG	9 (1.11)	12 (1.12)	18 (1.12)	8 (1.11)	11 (1.12)	17 (1.12)	11 (1.12)
Emin	8 (1.14)	7 (1.14)	8 (1.13)	7 (1.14)	7 (1.14)	– (1.13)	7 (1.14)
SchurComp (1, 2.0)	7 (1.13)	8 (1.15)	10 (1.15)	7 (1.14)	7 (1.15)	9 (1.15)	7 (1.15)
SchurComp (2, 2.0)	7 (1.13)	7 (1.15)	9 (1.15)	6 (1.14)	7 (1.15)	9 (1.15)	7 (1.15)
SchurComp (3, 2.0)	7 (1.13)	7 (1.15)	9 (1.15)	6 (1.14)	7 (1.15)	9 (1.15)	7 (1.15)

Table 4.7.: Backward-facing step example: number of outer GMRES iterations (and operator complexity) for different Reynolds numbers and meshes. 1 sweep with damped Gauss-Seidel ($\omega = 1.4$) as pre- and post-smoother and a direct solver on the coarsest level. '–' denotes no convergence within 50 iterations.

proposed SchurComp transfer operators are competitive with state-of-the-art transfer strategies and show satisfactory performance in many situations. More examples can be found in Section D.4.2. The Galerkin perspective in Section 4.4 points toward an attractive software framework for continued development of robust and flexible AMG solvers.

However, the multigrid transfer operators are only one part within the whole multigrid algorithm. An advanced flexible transfer operator strategy alone does not make a flexible multigrid algorithm which allows to be adapted for application-specific needs. In fact, the term "flexibility" has an even more general meaning: Flexibility means that one can adapt each part of the multigrid method as "tool" to the given specific problem to obtain an optimal preconditioner. This is only possible if one has an in-depth understanding of both the multigrid algorithms and the problem. The details on the internal aggregation-based multigrid algorithms have already been discussed in Chapter 3 and 4. In the next chapters it is exemplarily shown how to develop efficient multigrid preconditioners for problems arising from computational contact mechanics making use of the flexibility of the AMG framework. In Chapter 5, a basic introduction to computational contact mechanics is given providing the basic background knowledge for deriving efficient multigrid preconditioners.

CHAPTER 5

Finite deformation contact mechanics

This chapter is supposed to give a brief introduction to the formulation of finite deformation contact problems. In the following chapters of this thesis specialized AMG strategies are developed for certain classes of demanding linear systems as they arise from contact problems in different formulations. Here, the necessary background knowledge about the underlying numerical models is provided.

Contact mechanics plays an important role in many industrial applications. Starting with first contributions to the numerical treatment of contact problems in the 1970s and 1980s (e.g., Francavilla and Zienkiewicz [68]), the so-called *node-to-segment* (NTS) approach is widely used to deal with finite deformation contact (cf. Hesch and Betsch [91], Laursen and Simo [117], Laursen [118], Simo and Hughes [176], Wriggers et al. [229]). While NTS methods are still very popular in engineering practice, in the meantime mortar-based contact formulations (which can be understood as successor of the *segment-to-segment* approach (STS), cf. Simo et al. [177], Zavarise and Wriggers [232]) find more and more attention as versatile modern methods for computational contact mechanics (cf. Popp [156]).

Our formulation of finite deformation contact problems is based on mortar finite element methods, which originally have been developed for domain decomposition (cf. Bernardi et al. [24], Belgacem and Maday [17], Belgacem [18], Krause and Wohlmuth [109]). But in the meantime mortar finite element methods have also found much attention in the context of contact and mesh tying problems (e.g., Belgacem et al. [19]). There are many contact specific topics that are far beyond the scope of this thesis and not discussed here. For example, only Lagrange multipliers are used to enforce the contact constraints. For a general overview and discussion of alternatives one might refer, e.g., to Alart and Curnier [3]. Other popular approaches, such as penalty methods, are not considered. Many more advanced and contact specific topics are covered in the textbooks by Laursen [116] and Wriggers [228].

The following sections describe all steps from Section 1.1 for contact problems, including the modeling phase, the discretization phase and finally the solution phase. Section 5.1 introduces the problem setup with the basic notation. Sections 5.2 and 5.3 complete the modeling stage

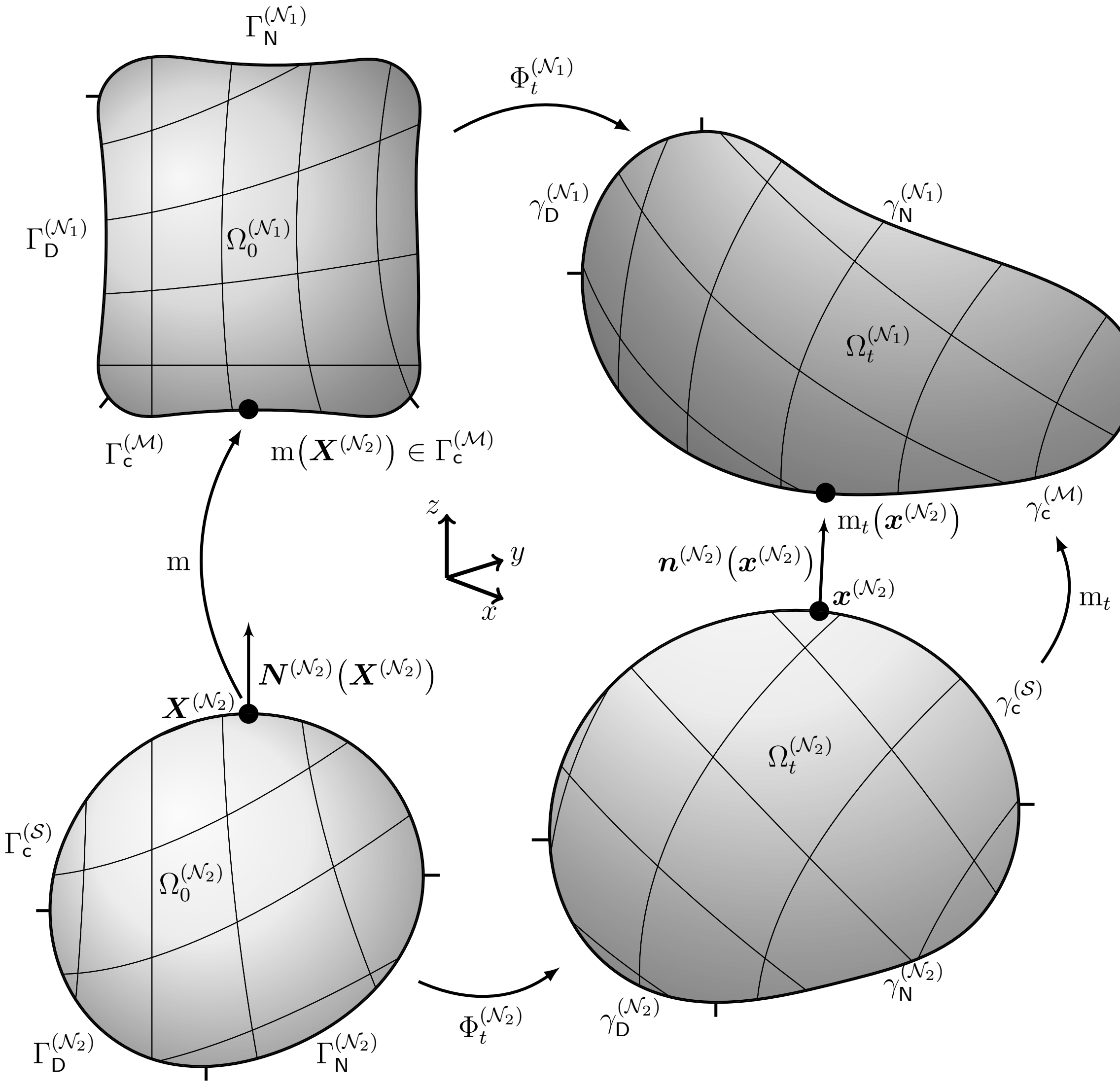

$\Gamma_N^{(\mathcal{N}_1)}$
$\Phi_t^{(\mathcal{N}_1)}$
$\Gamma_D^{(\mathcal{N}_1)}$
$\Omega_0^{(\mathcal{N}_1)}$
$\gamma_D^{(\mathcal{N}_1)}$
$\gamma_N^{(\mathcal{N}_1)}$
$\Omega_t^{(\mathcal{N}_1)}$
$\Gamma_c^{(\mathcal{M})}$
$\mathrm{m}\big(\boldsymbol{X}^{(\mathcal{N}_2)}\big) \in \Gamma_c^{(\mathcal{M})}$
z
y
x
m
$\mathrm{m}_t\big(\boldsymbol{x}^{(\mathcal{N}_2)}\big)$
$\gamma_c^{(\mathcal{M})}$
$\boldsymbol{n}^{(\mathcal{N}_2)}\big(\boldsymbol{x}^{(\mathcal{N}_2)}\big)$
$\boldsymbol{x}^{(\mathcal{N}_2)}$
m_t
$\boldsymbol{N}^{(\mathcal{N}_2)}\big(\boldsymbol{X}^{(\mathcal{N}_2)}\big)$
$\boldsymbol{X}^{(\mathcal{N}_2)}$
$\gamma_c^{(\mathcal{S})}$
$\Omega_t^{(\mathcal{N}_2)}$
$\Gamma_c^{(\mathcal{S})}$
$\Omega_0^{(\mathcal{N}_2)}$
$\gamma_N^{(\mathcal{N}_2)}$
$\Phi_t^{(\mathcal{N}_2)}$
$\gamma_D^{(\mathcal{N}_2)}$
$\Gamma_D^{(\mathcal{N}_2)}$
$\Gamma_N^{(\mathcal{N}_2)}$

Figure 5.1.: Problem configuration and basic notation.

by introducing the equations of nonlinear elastodynamics and extending the formulation to our prototype of a contact problem with two deformable solid bodies. In Section 5.4 the problem is discretized using the (mortar) finite element method. Therein, the focus is mainly on the different options for the choice of discrete Lagrange multipliers responsible for the coupling between the structural equations and the contact constraints. Finally, Section 5.5 gives a basic overview of the solution phase (cf. Figure 1.1). The concept of a semi-smooth Newton method used for solving the set of nonlinear equations resulting from the contact problem is discussed before closing the chapter with the final set of linearized equations, which are supposed to be solved iteratively using methods developed in the next chapters.

5.1. Problem setup

Without loss of generality, it is sufficient to discuss a contact problem with only two deformable bodies to derive all mathematical basics concerning contact kinematics and contact constraints. An extension to a multibody contact problem is straightforward. As a special case it also covers contact problems of Signorini type, i.e., the contact between a deformable body and a rigid obstacle.

Consider two bounded solid bodies, which are represented by $\Omega_0^{(\mathcal{N}_1)}, \Omega_0^{(\mathcal{N}_2)} \subset \mathbb{R}^d$ and $\Omega_t^{(\mathcal{N}_1)}, \Omega_t^{(\mathcal{N}_2)} \subset \mathbb{R}^d$ with $d \in \{2,3\}$ in the reference configuration and the current configuration as given in Figure 5.1. The domains are defined by $\Omega_0 = \Omega_0^{(\mathcal{N}_1)} \cup \Omega_0^{(\mathcal{N}_2)}$ and $\Omega_t = \Omega_t^{(\mathcal{N}_1)} \cup \Omega_t^{(\mathcal{N}_2)}$, respectively.

Each point $\boldsymbol{X}^{(\mathcal{N}_i)} \in \Omega_0^{(\mathcal{N}_i)}$, $i \in \{1,2\}$, in the reference configuration is mapped into the current configuration using the bijective nonlinear mapping $\Phi_t : \Omega_0 \times [0,T] \to \Omega_t$, $\big(\boldsymbol{X}, t\big) \mapsto \boldsymbol{x} = \Phi_t\big(\boldsymbol{X}, t\big)$, which is defined by

$$\Phi_t^{(\mathcal{N}_i)}\big(\boldsymbol{X}^{(\mathcal{N}_i)}, t\big) := \boldsymbol{X}^{(\mathcal{N}_i)} + \boldsymbol{u}^{(\mathcal{N}_i)}\big(\boldsymbol{X}^{(\mathcal{N}_i)}, t\big). \tag{5.1}$$

Given time t, the corresponding point $\boldsymbol{x}^{(\mathcal{N}_i)} \in \Omega_t^{(\mathcal{N}_i)}$ can be expressed as $\boldsymbol{x}\big(t\big) := \Phi_t\big(\boldsymbol{X}, t\big)$. In (5.1) the variable $\boldsymbol{u}\big(\boldsymbol{X}, t\big)$ denotes the displacement of $\boldsymbol{x}$ for a given time t relative to the reference coordinates $\boldsymbol{X}$ in the reference configuration.

The surfaces $\partial\Omega_0^{(\mathcal{N}_i)}$, $i \in \{1,2\}$ in the reference configuration are decomposed into three disjoint subsets $\Gamma_{\mathsf{D}}^{(\mathcal{N}_i)}$, $\Gamma_{\mathsf{N}}^{(\mathcal{N}_i)}$ and $\Gamma_{\mathsf{c}}^{(\mathcal{N}_i)}$, such that it is

$$\partial\Omega_0^{(\mathcal{N}_i)} := \Gamma_{\mathsf{D}}^{(\mathcal{N}_i)} \cup \Gamma_{\mathsf{N}}^{(\mathcal{N}_i)} \cup \Gamma_{\mathsf{c}}^{(\mathcal{N}_i)} \text{ with } \Gamma_{\mathsf{D}}^{(\mathcal{N}_i)} \cap \Gamma_{\mathsf{N}}^{(\mathcal{N}_i)} = \Gamma_{\mathsf{N}}^{(\mathcal{N}_i)} \cap \Gamma_{\mathsf{c}}^{(\mathcal{N}_i)} = \Gamma_{\mathsf{c}}^{(\mathcal{N}_i)} \cap \Gamma_{\mathsf{D}}^{(\mathcal{N}_i)} = \emptyset, \tag{5.2}$$

where $\Gamma_{\mathsf{D}}^{(\mathcal{N}_i)}$ and $\Gamma_{\mathsf{N}}^{(\mathcal{N}_i)}$ denote the Dirichlet and Neumann parts of the boundaries and $\Gamma_{\mathsf{c}}^{(\mathcal{N}_i)}$ the potential contact interface. Analogous to (5.2) the corresponding surface boundaries in the current configuration are defined by

$$\partial\Omega_t^{(\mathcal{N}_i)} := \gamma_{\mathsf{D}}^{(\mathcal{N}_i)} \cup \gamma_{\mathsf{N}}^{(\mathcal{N}_i)} \cup \gamma_{\mathsf{c}}^{(\mathcal{N}_i)} \text{ with } \gamma_{\mathsf{D}}^{(\mathcal{N}_i)} \cap \gamma_{\mathsf{N}}^{(\mathcal{N}_i)} = \gamma_{\mathsf{N}}^{(\mathcal{N}_i)} \cap \gamma_{\mathsf{c}}^{(\mathcal{N}_i)} = \gamma_{\mathsf{c}}^{(\mathcal{N}_i)} \cap \gamma_{\mathsf{D}}^{(\mathcal{N}_i)} = \emptyset. \tag{5.3}$$

5.2. Strong formulation

In this section, the governing equations for solid mechanics are formulated and later extended for contact mechanics to describe the behavior of the solid bodies. Since this thesis is not about material models, the discussion of constitutive relations between kinematic quantities (strains) and the material response (stresses) is completely skipped. The interested reader may refer to the literature (cf. Bonet and Wood [26], Holzapfel [95]). Throughout this thesis, only homogeneous bodies are considered undergoing purely elastic deformations with a hyper-elastic material behavior.

For each body the local balance of linear momentum reads

$$\rho^{(\mathcal{N}_i)}\ddot{\boldsymbol{u}}^{(\mathcal{N}_i)} - \operatorname{Div}\left(\boldsymbol{P}^{(\mathcal{N}_i)}\right) = \boldsymbol{f}^{(\mathcal{N}_i)} \qquad \text{in } \Omega_0^{(\mathcal{N}_i)} \times [0,T] \tag{5.4}$$

with the boundary conditions

$$\boldsymbol{u}^{(\mathcal{N}_i)} = \widehat{\boldsymbol{u}}^{(\mathcal{N}_i)} \qquad \text{on } \Gamma_{\mathsf{D}}^{(\mathcal{N}_i)} \times [0,T]\,, \tag{5.5a}$$

$$\boldsymbol{P}^{(\mathcal{N}_i)}\boldsymbol{N}^{(\mathcal{N}_i)} = \widehat{\boldsymbol{t}}_0^{(\mathcal{N}_i)} \qquad \text{on } \Gamma_{\mathsf{N}}^{(\mathcal{N}_i)} \times [0,T] \tag{5.5b}$$

for $i \in \{1,2\}$. Therein, $\boldsymbol{P}$ denotes the first Piola-Kirchhoff stress tensor. For a detailed definition the reader is referred to the literature for nonlinear continuum mechanics (e.g., Bonet and Wood [26]). Moreover, $\boldsymbol{N}^{(\mathcal{N}_i)}$ denotes the outward unit normal vector in the reference configuration on $\Gamma^{(\mathcal{N}_i)}$, and the $\widehat{\boldsymbol{u}}^{(\mathcal{N}_i)}$ and $\widehat{\boldsymbol{t}}_0^{(\mathcal{N}_i)}$ describe the prescribed displacements and surface tractions, respectively. To resolve the time dependency in (5.4) and to close the system, initial conditions have to be prescribed for the displacements and velocities, viz.

$$\boldsymbol{u}^{(\mathcal{N}_i)}\big(\boldsymbol{X},0\big) = \widehat{\boldsymbol{u}}_0^{(\mathcal{N}_i)} \qquad \text{in } \Omega_0^{(\mathcal{N}_i)}, \tag{5.6}$$

$$\dot{\boldsymbol{u}}^{(\mathcal{N}_i)}\big(\boldsymbol{X},0\big) = \widehat{\dot{\boldsymbol{u}}}_0^{(\mathcal{N}_i)} \qquad \text{in } \Omega_0^{(\mathcal{N}_i)}. \tag{5.7}$$

After the initial boundary value problem of nonlinear solid mechanics has been introduced with (5.4) to (5.7) one can specify the nonlinear contact conditions. From now on, $\Gamma_{\mathsf{c}}^{(\mathcal{N}_1)}$ is declared to be the so-called *master* (or *mortar*) side of the contact interface, denoted by $\Gamma_{\mathsf{c}}^{(\mathcal{M})}$. The contact boundary $\Gamma_{\mathsf{c}}^{(\mathcal{N}_2)}$ is referred to as the *slave* (or *non-mortar*) side of the contact interface. It is denoted as $\Gamma_{\mathsf{c}}^{(\mathcal{S})}$. The notation for $\gamma_{\mathsf{c}}^{(\mathcal{N}_i)}$, $i \in \{1,2\}$, is changed to $\gamma_{\mathsf{c}}^{(\mathcal{M})}$ and $\gamma_{\mathsf{c}}^{(\mathcal{S})}$, respectively. The superscript $\mathcal{M}$ stands for the *master* body and $\mathcal{S}$ for the *slave* body. The choice of the master and slave body is arbitrary and only important for defining potential contact surfaces between the two solid bodies.

For the formulation of the contact conditions one uses a predefined smooth interface mapping $\mathrm{m} : \Gamma_{\mathsf{c}}^{(\mathcal{S})} \to \Gamma_{\mathsf{c}}^{(\mathcal{M})}$ of a contact point $\boldsymbol{X}^{(\mathcal{N}_2)} \in \Gamma_{\mathsf{c}}^{(\mathcal{S})}$ on the slave contact interface to the master contact interface $\Gamma_{\mathsf{c}}^{(\mathcal{M})}$ (cf. Figure 5.1). Assuming that m is well defined, it is $\mathrm{m}_t\big(\gamma_{\mathsf{c}}^{(\mathcal{S})}\big) \subset \gamma_{\mathsf{c}}^{(\mathcal{M})}$ for all $t \in (0,T)$. Here, $\mathrm{m}_t : \gamma_{\mathsf{c}}^{(\mathcal{S})} \to \gamma_{\mathsf{c}}^{(\mathcal{M})}$ stands for the smooth interface mapping m in the current configuration for a specific time t.

Using the mapping Φ_t from (5.1) between the reference and current configuration one can define the normal gap at a point $\boldsymbol{X} \in \Gamma_{\mathsf{c}}^{(\mathcal{S})}$ in the reference configuration by

$$g_{\mathrm{n}}\big(\boldsymbol{X},t\big) = -\boldsymbol{n}^{(\mathcal{N}_2)}\Big(\Phi_t^{(\mathcal{S})}\big(\boldsymbol{X},t\big) - \Phi_t^{(\mathcal{M})}\big(\mathrm{m}(\boldsymbol{X}),t\big)\Big), \tag{5.8}$$

where $\boldsymbol{n}^{(\mathcal{N}_2)}$ denotes the outward unit normal vector on $\gamma_{\mathsf{c}}^{(\mathcal{S})}$ at $\boldsymbol{x} = \Phi_t^{(\mathcal{S})}\big(\boldsymbol{X},t\big)$.

With the splitting of the contact traction $\boldsymbol{t}_{\mathsf{c}}^{(\mathcal{S})}$ on the slave surface $\gamma_{\mathsf{c}}^{(\mathcal{S})}$ into normal and tangential components

$$\boldsymbol{t}_{\mathsf{c}}^{(\mathcal{S})} = p_{\mathrm{n}}\boldsymbol{n} + \boldsymbol{t}_{\tau} \tag{5.9}$$

the normal contact constraints are given by

$$g_\mathrm{n}\big(\boldsymbol{X},t\big) \geq 0 \qquad \text{on } \gamma_\mathsf{c}^{(\mathcal{S})} \times [0,T]\,, \tag{5.10a}$$
$$p_\mathrm{n}\big(\boldsymbol{X},t\big) \leq 0 \qquad \text{on } \gamma_\mathsf{c}^{(\mathcal{S})} \times [0,T]\,, \tag{5.10b}$$
$$p_\mathrm{n}\big(\boldsymbol{X},t\big)g_\mathrm{n}\big(\boldsymbol{X},t\big) = 0 \qquad \text{on } \gamma_\mathsf{c}^{(\mathcal{S})} \times [0,T]\,. \tag{5.10c}$$

The constraints (5.10) form a set of Karush-Kuhn-Tucker (KKT) conditions. The first KKT condition in (5.10a) describes the geometric constraint of non-penetration. Herein, p_n denotes the normal contact traction (pressure) in the current configuration. The second KKT condition in (5.10b) implies that no adhesive stresses are allowed in the contact zone. Finally, the complementary condition in (5.10c) makes sure that the gap is closed (i.e., $g_\mathrm{n} = 0$) when non-zero contact pressure occurs in the contact case or – vice versa – that the contact pressure is forced to be zero if the gap is open (i.e., $g_\mathrm{n} > 0$) in the non-contact case.

For frictionless sliding, the tangential part $\boldsymbol{t}_\tau$ of the slave side contact traction in (5.9) is supposed to vanish, i.e., $\boldsymbol{t}_\tau = \mathbf{0}$.

Remark 5.2.1 (Frictional contact). While a frictionless response (i.e., $\boldsymbol{t}_\tau = \mathbf{0}$) is a common modeling assumption, the real contact behavior for many industrial applications is governed by the frictional response to tangential loading. Details on frictional contact are far beyond the scope of this thesis. Here, only Coulomb's law is briefly mentioned, which is often used for dry friction. One possible notation of Coulomb friction is given by

$$\|\boldsymbol{t}_\tau\| - \mathfrak{F}|p_\mathrm{n}| \leq 0 \qquad\qquad \text{on } \gamma_\mathsf{c}^{(\mathcal{S})} \times [0,T]\,, \tag{5.11a}$$
$$\boldsymbol{v}_{\tau,\mathsf{rel}} + \beta\boldsymbol{t}_\tau = \mathbf{0} \quad \text{with } \beta \geq 0 \qquad \text{on } \gamma_\mathsf{c}^{(\mathcal{S})} \times [0,T]\,, \tag{5.11b}$$
$$\Big(\|\boldsymbol{t}_\tau\| - \mathfrak{F}|p_\mathrm{n}|\Big)\beta = 0 \qquad\qquad \text{on } \gamma_\mathsf{c}^{(\mathcal{S})} \times [0,T]\,. \tag{5.11c}$$

Herein, $\mathfrak{F} \geq 0$ is the friction coefficient, $\beta \geq 0$ is a scalar parameter and $\boldsymbol{v}_{\tau,\mathsf{rel}}$ denotes the relative tangential velocity as primary kinematic variable for frictional sliding. The first inequality (5.11a) implies that the magnitude of the tangential stress $\boldsymbol{t}_\tau$ does not exceed the threshold defined by the friction coefficient $\mathfrak{F}$ and the normal contact pressure p_n. The frictional response depends on the value of the scalar β. For $\beta = 0$ it follows from (5.11b) that $\boldsymbol{v}_{\tau,\mathsf{rel}} = \mathbf{0}$, i.e., the relative tangential movement in the contact zone is forced to be zero (stick state). In contrast, $\beta > 0$ describes the slip state which allows for some relative tangential movement in the contact zone. Equation (5.11c) can be interpreted as complementary condition for switching between the slip and stick situation. For further details on the modeling of frictional sliding, the reader is referred to, e.g., Kikuchi and Oden [104] or Wriggers [228].

For reasons of simplicity, only the frictionless case is discussed as additional friction terms have no significant effect on the final algebraic structure of the resulting discrete problems. This means that they are not essential for the development of linear solution strategies, such that all linear solution strategies should also work for the frictional case without any change.

Before proceeding with the weak variational formulation, let us collect all equations and constraints for the dynamical contact problem using the negative slave side contact traction $\boldsymbol{t}_\mathsf{c}^{(\mathcal{S})}$ as Lagrange multiplier, i.e., $\boldsymbol{\lambda} = -\boldsymbol{t}_\mathsf{c}^{(\mathcal{S})}$. In the following the normal part of the contact stress is denoted by $\lambda_\mathrm{n} := \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{n}$ and the tangential part by $\boldsymbol{\lambda}_\tau := \boldsymbol{\lambda} - \lambda_\mathrm{n}\boldsymbol{n}$.

The overall strong formulation for the dynamical contact problem reads as

$$\rho^{(\mathcal{N}_i)}\ddot{\boldsymbol{u}}^{(\mathcal{N}_i)} - \operatorname{Div}\left(\boldsymbol{P}^{(\mathcal{N}_i)}\right) = \boldsymbol{f}^{(\mathcal{N}_i)} \qquad \text{in } \Omega_0^{(\mathcal{N}_i)} \times [0,T]\,, \tag{5.12}$$

with the boundary conditions

$$\boldsymbol{u}^{(\mathcal{N}_i)} = \widehat{\boldsymbol{u}}^{(\mathcal{N}_i)} \qquad \text{on } \Gamma_{\mathsf{D}}^{(\mathcal{N}_i)} \times [0,T]\,, \tag{5.13a}$$

$$\boldsymbol{P}^{(\mathcal{N}_i)}\boldsymbol{N}^{(\mathcal{N}_i)} = \widehat{\boldsymbol{t}}_0^{(\mathcal{N}_i)} \qquad \text{on } \Gamma_{\mathsf{N}}^{(\mathcal{N}_i)} \times [0,T]\,, \tag{5.13b}$$

the initial conditions

$$\boldsymbol{u}^{(\mathcal{N}_i)}\big(\boldsymbol{X},0\big) = \widehat{\boldsymbol{u}}_0^{(\mathcal{N}_i)} \qquad \text{in } \Omega_0^{(\mathcal{N}_i)}, \tag{5.14a}$$

$$\dot{\boldsymbol{u}}^{(\mathcal{N}_i)}\big(\boldsymbol{X},0\big) = \widehat{\dot{\boldsymbol{u}}}_0^{(\mathcal{N}_i)} \qquad \text{in } \Omega_0^{(\mathcal{N}_i)}, \tag{5.14b}$$

the normal contact constraints

$$-g_{\mathrm{n}} \leq 0 \qquad \text{on } \gamma_{\mathsf{c}}^{(\mathcal{S})} \times [0,T]\,, \tag{5.15a}$$

$$\lambda_{\mathrm{n}} \geq 0 \qquad \text{on } \gamma_{\mathsf{c}}^{(\mathcal{S})} \times [0,T]\,, \tag{5.15b}$$

$$\lambda_{\mathrm{n}} g_{\mathrm{n}} = 0 \qquad \text{on } \gamma_{\mathsf{c}}^{(\mathcal{S})} \times [0,T]\,, \tag{5.15c}$$

and the frictionless sliding condition

$$\boldsymbol{t}_\tau = \boldsymbol{0} \qquad \text{on } \gamma_{\mathsf{c}}^{(\mathcal{S})} \times [0,T] \tag{5.16}$$

with $i \in \{1,2\}$. The set of equations (5.12) to (5.16) serves as starting point for the variational formulation.

5.3. Weak formulation

The solution spaces $\boldsymbol{\mathcal{U}}^{(\mathcal{N}_i)}$ are defined as

$$\boldsymbol{\mathcal{U}}^{(\mathcal{N}_i)} := \left\{\boldsymbol{u}^{(\mathcal{N}_i)} \in \left[H^1(\Omega_0^{(\mathcal{N}_i)})\right]^d \;:\; \boldsymbol{u}^{(\mathcal{N}_i)}|_{\Gamma_{\mathsf{D}}^{(\mathcal{N}_i)}} = \widehat{\boldsymbol{u}}_0^{(\mathcal{N}_i)}\right\}, \qquad i \in \{1,2\}\,, \tag{5.17}$$

with the corresponding product space given as

$$\boldsymbol{\mathcal{U}} := \prod_{i\in\{1,2\}} \boldsymbol{\mathcal{U}}^{(\mathcal{N}_i)} = \boldsymbol{\mathcal{U}}^{(\mathcal{N}_1)} \times \boldsymbol{\mathcal{U}}^{(\mathcal{N}_2)}. \tag{5.18}$$

Here, $d \in \{2,3\}$ denotes the dimension of the problem. Following the notation in Evans [62, chapter 5.9.2], the solution space for the dynamic problem can be written as $\boldsymbol{\mathcal{U}}_t := H^2\big([0,T]\,;\boldsymbol{\mathcal{U}}\big)$ which contains all functions $\boldsymbol{w} \in H^2([0,T])$ with $\boldsymbol{w} : [0,T] \to \boldsymbol{\mathcal{U}}$.

For the corresponding test or weighting space one has to use functions satisfying zero Dirichlet boundary conditions on $\Gamma_{\mathsf{D}}^{(\mathcal{N}_i)}$. Accordingly, the definition of the weighting spaces for the displacements $\boldsymbol{u}$ reads as

$$\boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_i)} := \Big\{ \boldsymbol{v}^{(\mathcal{N}_i)} \in \big[H^1(\Omega_0^{(\mathcal{N}_i)})\big]^d \ : \ \boldsymbol{v}^{(\mathcal{N}_i)}\big|_{\Gamma_{\mathsf{D}}^{(\mathcal{N}_i)}} = \boldsymbol{0} \Big\}, \qquad i \in \{1,2\}, \tag{5.19}$$

with the product space $\boldsymbol{\mathcal{V}}_{\boldsymbol{u}} := \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_1)} \times \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_2)}$. Both $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}_{\boldsymbol{u}}$ are equipped with the broken H^1-norm $\|\boldsymbol{v}\|_{1,\Omega_0}^2 := \sum_{i\in\{1,2\}} \|\boldsymbol{v}\|_{1,\Omega_0^{(\mathcal{N}_i)}}^2$.

The Lagrange multiplier space $\boldsymbol{\mathcal{M}}$ is defined to be the dual space of the trace space $\boldsymbol{\mathcal{N}}$ of $\boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_2)}$ restricted to $\gamma_{\mathsf{c}}^{(\mathcal{S})}$ in the current configuration. That is, one has $\boldsymbol{\mathcal{N}} := \big[H^{1/2}(\gamma_{\mathsf{c}}^{(\mathcal{S})})\big]^d$ and $\boldsymbol{\mathcal{M}} := \big[H^{-1/2}(\gamma_{\mathsf{c}}^{(\mathcal{S})})\big]^d$, respectively. For more details, the interested reader may refer to the literature, e.g., Hüeber [99, Section 2.1.2] or Popp [156, Section 4.1.2].

The Lagrange multipliers $\boldsymbol{\lambda}$ are chosen from a convex subspace of $\boldsymbol{\mathcal{M}}$ which is given by

$$\boldsymbol{\mathcal{M}}_+ := \Big\{ \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}} \ : \ \boldsymbol{\mu}_\tau = \boldsymbol{0}, \langle \mu_{\mathrm{n}}, \eta \rangle_{\gamma_{\mathsf{c}}^{(\mathcal{S})}} \geq 0, \ \eta \in \boldsymbol{\mathcal{N}}_+ \Big\}, \tag{5.20}$$

where $\boldsymbol{\mathcal{N}}_+ := \big\{ w \in H^{1/2}(\gamma_{\mathsf{c}}^{(\mathcal{S})}) \ : \ w \geq 0 \big\}$ describes a closed convex non-empty cone in $\boldsymbol{\mathcal{N}}$. The symbol $\langle \cdot, \cdot \rangle_{\gamma_{\mathsf{c}}^{(\mathcal{S})}}$ stands for the duality pairing between $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{N}}$ on the contact slave interface $\gamma_{\mathsf{c}}^{(\mathcal{S})}$ given by

$$\langle \boldsymbol{\lambda}, \boldsymbol{v} \rangle_{\gamma_{\mathsf{c}}^{(\mathcal{S})}} := \int_{\gamma_{\mathsf{c}}^{(\mathcal{S})}} \boldsymbol{\lambda} \boldsymbol{v} \mathsf{d}\boldsymbol{s}. \tag{5.21}$$

With the definition of the forms

$$m\big(\boldsymbol{u}, \boldsymbol{v}\big) := \sum_{i\in\{1,2\}} \int_{\Omega_0^{(\mathcal{N}_i)}} \rho^{(\mathcal{N}_i)} \boldsymbol{u}^{(\mathcal{N}_i)} \boldsymbol{v}^{(\mathcal{N}_i)} \mathsf{d}\boldsymbol{X}, \tag{5.22}$$

$$a\big(\boldsymbol{u}, \boldsymbol{v}\big) := \sum_{i\in\{1,2\}} \int_{\Omega_0^{(\mathcal{N}_i)}} \boldsymbol{P}^{(\mathcal{N}_i)} : \nabla \boldsymbol{v}^{(\mathcal{N}_i)} \mathsf{d}\boldsymbol{X} \tag{5.23}$$

as well as the linear form

$$f\big(\boldsymbol{v}\big) := \sum_{i\in\{1,2\}} \int_{\Omega_0^{(\mathcal{N}_i)}} \boldsymbol{f}^{(\mathcal{N}_i)} \boldsymbol{v}^{(\mathcal{N}_i)} \mathsf{d}\boldsymbol{X} + \sum_{i\in\{1,2\}} \int_{\Gamma_{\mathsf{N}}^{(\mathcal{N}_i)}} \widehat{\boldsymbol{t}}^{(\mathcal{N}_i)} \boldsymbol{v}^{(\mathcal{N}_i)} \mathsf{d}\boldsymbol{S} \tag{5.24}$$

and the definition of the jump

$$\big[\boldsymbol{u}(\boldsymbol{x}, t)\big] := \boldsymbol{u}^{(\mathcal{S})}(\boldsymbol{x}, t) - \boldsymbol{u}^{(\mathcal{M})}\big(\mathrm{m}_t(\boldsymbol{x}), t\big) \quad \text{for } (\boldsymbol{x}, t) \in \gamma_{\mathsf{c}}^{(\mathcal{S})} \times [0, T], \tag{5.25}$$

the variational formulation of the dynamical contact problem can be stated as:

Find $\big(\boldsymbol{u}, \boldsymbol{\lambda}\big) \in \boldsymbol{\mathcal{U}}_t \times \boldsymbol{\mathcal{M}}_+$ such that for all $t \in [0, T]$

$$m\big(\ddot{\boldsymbol{u}}, \boldsymbol{v}\big) + a\big(\boldsymbol{u}, \boldsymbol{v}\big) + \big\langle [v], \boldsymbol{\lambda} \big\rangle_{\gamma_{\mathsf{c}}^{(\mathcal{S})}} = f(\boldsymbol{v}), \qquad \boldsymbol{v} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}, \tag{5.26a}$$

$$\big\langle g_{\mathrm{n}}, \mu_{\mathrm{n}} - \lambda_{\mathrm{n}} \big\rangle_{\gamma_{\mathsf{c}}^{(\mathcal{S})}} \geq 0, \qquad \boldsymbol{\mu} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{\lambda}}, \tag{5.26b}$$

holds in combination with the weak form of the initial conditions (5.14a) and (5.14b)

$$\big(\boldsymbol{u}_0, \boldsymbol{v}\big)_0 = \big(\widehat{\boldsymbol{u}}_0, \boldsymbol{v}\big)_0, \quad \big(\dot{\boldsymbol{u}}_0, \boldsymbol{v}\big)_0 = \big(\widehat{\dot{\boldsymbol{u}}}_0, \boldsymbol{v}\big)_0, \quad \boldsymbol{v} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{u}} \tag{5.27}$$

with $\big(\cdot,\cdot\big)_0$ the L^2-scalar product on $\Omega_0 := \Omega_0^{(\mathcal{S})} \cup \Omega_0^{(\mathcal{M})}$. The $\boldsymbol{\mathcal{V}}_{\boldsymbol{\lambda}}$ denotes the space for the weighting functions for the Lagrange multipliers and can be considered to be identical to $\boldsymbol{\mathcal{M}}_+$ in the standard Bubnov–Galerkin case.

Note that in contrary to (5.22), which is a bilinear form, the form $a(\cdot,\cdot)$ in (5.23) is only linear in the second argument for general nonlinear material laws. Only in case of linearized elasticity the form $a(\cdot,\cdot)$ is bilinear.

5.4. Space discretization

5.4.1. Finite element spaces for structural degrees of freedom

To derive the discrete form of the hybrid variational formulation (5.26), standard conforming finite elements of lowest order are used. For the 2D case simplicial or quadrilateral meshes are built and for the 3D case tetrahedral or hexahedral meshes are used. Let $\mathcal{T}_{\mathrm{h},\Omega_0^{(\mathcal{N}_i)}}$ denote the triangulation of body $\Omega_0^{(\mathcal{N}_i)}$ with $i \in \{1,2\}$ and $\mathcal{T}_{\mathrm{h},\Omega_0} := \mathcal{T}_{\mathrm{h},\Omega_0^{(\mathcal{N}_1)}} \cup \mathcal{T}_{\mathrm{h},\Omega_0^{(\mathcal{N}_2)}}$ the triangulation of the full domain, where h describes the mesh size parameter as the maximum diameter over all elements using $\mathrm{h} := \max_{T\in\mathcal{T}_{\mathrm{h},\Omega_0}} \mathrm{h}_T$. The triangulation $\mathcal{T}_{\mathrm{h},\Omega_0}$ is assumed to be shape-regular and quasi-uniform.

Definition 5.4.1 (Shape-regular and quasi-uniform mesh)**.** A triangulation $\mathcal{T}_{\mathrm{h},\Omega_0}$ is called *shape-regular*, if and only if there is a constant $c > 0$, such that for all elements $T \in \mathcal{T}_{\mathrm{h},\Omega_0}$ it is $\rho_T \geq c\mathrm{h}_T$ with ρ_T denoting the diameter of a ball which is inscribed in element T. A triangulation is called *quasi-uniform*, if and only if there exists a constant $c > 0$, such that for all elements $T \in \mathcal{T}_{\mathrm{h},\Omega_0}$ it is $\mathrm{h}_T \geq c\mathrm{h}$.

Let $\mathcal{S}_1^{(\mathcal{N}_i)} := \mathcal{S}_1\big(\Omega_0^{(\mathcal{N}_i)}, \mathcal{T}_{\mathrm{h},\Omega_0^{(\mathcal{N}_i)}}\big)$ denote the finite element space for linear finite elements associated with the triangulation $\mathcal{T}_{\mathrm{h},\Omega_0^{(\mathcal{N}_i)}}$. Then, one can define the discrete function space

$$\boldsymbol{\mathcal{U}}^{(\mathcal{N}_i),\mathrm{h}} := \Big\{\boldsymbol{u}^{(\mathcal{N}_i)} \in \big[\mathcal{S}_1^{(\mathcal{N}_i)}\big]^d \;:\; \boldsymbol{u}^{(\mathcal{N}_i)}\big|_{\Gamma_\mathrm{D}^{(\mathcal{N}_i)}} = \widehat{\boldsymbol{u}}_0^{(\mathcal{N}_i)}\Big\} \subset \boldsymbol{\mathcal{U}}^{(\mathcal{N}_i)}, \qquad i \in \{1,2\}\,. \tag{5.28}$$

The product space is defined by $\boldsymbol{\mathcal{U}}^\mathrm{h} := \boldsymbol{\mathcal{U}}^{(\mathcal{N}_1),\mathrm{h}} \times \boldsymbol{\mathcal{U}}^{(\mathcal{N}_2),\mathrm{h}}$ and therefore $\boldsymbol{\mathcal{U}}_t^\mathrm{h} := H^2\big([0,T]\,;\boldsymbol{\mathcal{U}}^\mathrm{h}\big)$. It is assumed that the Dirichlet data $\widehat{\boldsymbol{u}}_0^{(\mathcal{N}_i)}$ is resolved by the discrete space $\boldsymbol{\mathcal{U}}^{(\mathcal{N}_i),\mathrm{h}}$ properly. The corresponding test space is given by $\boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^\mathrm{h} := \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_1),\mathrm{h}} \times \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_2),\mathrm{h}}$ with

$$\boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_i),\mathrm{h}} := \Big\{\boldsymbol{v}^{(\mathcal{N}_i)} \in \big[\mathcal{S}_1^{(\mathcal{N}_i)}\big]^d \;:\; \boldsymbol{v}^{(\mathcal{N}_i)}\big|_{\Gamma_\mathrm{D}^{(\mathcal{N}_i)}} = \boldsymbol{0}\Big\}, \qquad i \in \{1,2\}\,.$$

The spaces $\boldsymbol{\mathcal{U}}^{(\mathcal{N}_i),\mathrm{h}}$ as well as $\boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{(\mathcal{N}_i),\mathrm{h}}$ can be spanned by the standard nodal basis, that is defined by $\big\{\phi_p \boldsymbol{e}_k,\ \text{with}\ p = 1,\dots,m_0^{\boldsymbol{u}}\ \text{and}\ k = 1,\dots,d\big\}$. Therein, $\boldsymbol{e}_k$ denotes the k-th unit vector and ϕ_p the scalar standard nodal basis function associated with the node p. The number $m_0^{\boldsymbol{u}}$ represents the total number of the nodes in the finite element mesh $\mathcal{T}_{\mathrm{h},\Omega_0}$ for the displacement degrees of freedom $\boldsymbol{u}$. Exemplarily, Figure 5.2a shows the 1D standard basis functions ϕ_p.

5.4.2. Discrete Lagrange multiplier spaces

The Lagrange multiplier space inherits the $(d-1)$-dimensional mesh from the d-dimensional triangulation $\mathcal{T}_{\mathrm{h},\Omega_0}$ of Ω_0 at the slave side of the contact interface. The discrete Lagrange multiplier space is denoted corresponding to the Lagrange multiplier space $\boldsymbol{\mathcal{M}}$ as defined in Section 5.3 with $\boldsymbol{\mathcal{M}}^{\mathrm{h}}$. The discrete analogue $\boldsymbol{\mathcal{M}}^{\mathrm{h}}_{+}$ of the admissible Lagrange multiplier space $\boldsymbol{\mathcal{M}}_{+}$ from (5.20) is given by

$$\boldsymbol{\mathcal{M}}^{\mathrm{h}}_{+} := \left\{ \boldsymbol{\mu}^{\mathrm{h}} \in \boldsymbol{\mathcal{M}}^{\mathrm{h}} \ : \ \boldsymbol{\mu}^{\mathrm{h}}_{\tau} = \mathbf{0}, \langle \mu^{\mathrm{h}}, \eta^{\mathrm{h}} \rangle_{\gamma_{\mathrm{c}}^{(\mathcal{S})}} \geq 0, \ \eta^{\mathrm{h}} \in \mathcal{N}^{\mathrm{h}}_{+} \right\}, \tag{5.29}$$

with $\mathcal{N}^{\mathrm{h}}_{+}$ the discrete analogon to $\mathcal{N}_{+}$ based on the discrete trace space $\mathcal{N}^{\mathrm{h}}$ of $\boldsymbol{\mathcal{U}}^{(\mathcal{N}_2),\mathrm{h}}$ restricted to the contact slave interface $\gamma_{\mathrm{c}}^{(\mathcal{S})}$.

Let ψ_p denote the p-th basis function associated with node $p = 1, \dots, m_0^{\boldsymbol{\lambda}}$ where $m_0^{\boldsymbol{\lambda}}$ stands for the total number of nodes on the slave side of the contact interface $\Gamma_{\mathrm{c}}^{(\mathcal{S})}$. Then, the discrete multiplier space $\boldsymbol{\mathcal{M}}^{\mathrm{h}}$ is spanned by $\left\{ \psi_p \boldsymbol{e}_k \text{ with } p = 1, \dots, m_0^{\boldsymbol{\lambda}} \text{ and } k = 1, \dots, d \right\}$. In contrast to the basis functions for the discrete solution and test spaces $\boldsymbol{\mathcal{U}}^{\mathrm{h}}$ and $\boldsymbol{\mathcal{V}}^{\mathrm{h}}_{\boldsymbol{u}}$, where linear finite elements with the corresponding standard basis functions have been used, one can choose different basis functions for the discrete Lagrange multiplier spaces $\boldsymbol{\mathcal{M}}^{\mathrm{h}}$ and the corresponding test spaces $\boldsymbol{\mathcal{V}}^{\mathrm{h}}_{\boldsymbol{\lambda}}$.

Standard basis functions: First-order interpolation with standard basis functions for the Lagrange multipliers (i.e., $\psi_p = \phi_p$) represents the most commonly used discretization strategy in context of mortar methods, with applications ranging from classical domain decomposition for elasticity (cf. Puso [162]) to finite deformation contact (cf. Puso and Laursen [164]). First-order standard basis functions are known to be strictly positive, which turns out to be advantageous for the non-penetration condition in the formulation of the contact constraint by a positive gap function.

Dual basis functions: Motivated by the observation that the Lagrange multipliers represent tractions on the contact slave interface, Wohlmuth [225] proposes the use of so-called dual basis functions which are based on a so-called bi-orthogonality relation

$$\int_{\Gamma_{\mathrm{c}}^{(\mathcal{S})}} \phi_p \psi_q \mathrm{d}\boldsymbol{S} = \delta_{pq} \int_{\Gamma_{\mathrm{c}}^{(\mathcal{S})}} \phi_p \mathrm{d}\boldsymbol{S}, \tag{5.30}$$

where δ_{pq} denotes the Kronecker symbol, which is defined by

$$\delta_{ik} := \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{if } i \neq k. \end{cases} \tag{5.31}$$

The dual (or bi-orthogonal) basis functions have the same support as the corresponding standard basis functions, i.e., $\mathrm{supp}(\psi_p) = \mathrm{supp}(\phi_p)$. The bi-orthogonality condition in (5.30) as a construction principle for the dual Lagrange multiplier shape functions localizes the interface coupling conditions (cf. Wohlmuth [225]) and allows for an efficient condensation of the Lagrange multipliers from the global system of equations. The resulting linear systems are smaller in size and have no saddle point structure, which might be advantageous for the design of iterative solvers and preconditioners.

Figure 5.2 shows the standard and the dual basis functions for the case $d = 2$ in comparison. Obviously, the dual basis functions are not continuous and cannot be interpreted as a trace of conforming finite elements. However, the dual approach has the advantage that it heavily facilitates the treatment of typical mortar coupling conditions at the contact interface while preserving the mathematical optimality of the method (cf. Popp [156], Seshaiyer and Suri [175], Wohlmuth [225]).

Whereas originally introduced for mortar finite element methods in 2D, dual shape basis functions have been extended to 3D in Wohlmuth and Krause [227]. First-order interpolation using dual Lagrange multipliers is well analyzed in the general mortar setting (cf. Puso [162], Wohlmuth [223]) and for the (unilateral) contact (e.g., Hüeber and Wohlmuth [96], Popp et al. [157, 159]).

Remark 5.4.2 (Bubnov–Galerkin versus Petrov–Galerkin approach). In certain situations, the dual mortar method may lack robustness or even consistency. As one can see from Figure 5.2b, dual basis functions are not strictly positive, even for first-order interpolation. This may lead to severe problems arising from a nonphysical interpolation of the so-called weighted gap values which are the fundamental geometric measure for penetration in a mortar discretized setting. Further details can be found in Section 5.5 or in the works by Hüeber [99] and Popp [156].

In Popp et al. [160] the authors introduce several improvements for the dual mortar approach including a novel approach to unify the advantages of standard and dual mortar methods via a Petrov–Galerkin type of Lagrange multiplier interpolation leading to a more robust method. In contrary to the standard Bubnov–Galerkin approach (i.e., $\boldsymbol{\mathcal{V}}_{\boldsymbol{\lambda}}^{\mathrm{h}} = \boldsymbol{\mathcal{M}}_{+}^{\mathrm{h}}$), a Petrov–Galerkin approach is proposed for the Lagrange multipliers $\boldsymbol{\lambda}$. Dual basis functions are used for the Lagrange multiplier field $\boldsymbol{\lambda}$ in the solution space $\boldsymbol{\mathcal{M}}_{+}^{\mathrm{h}}$, which localizes the interface coupling conditions and eventually allows for an efficient condensation of the Lagrange multiplier degrees of freedom from the global linear system. On the other hand, standard shape functions are used for the test function space $\boldsymbol{\mathcal{V}}_{\boldsymbol{\lambda}}^{\mathrm{h}}$ of the Lagrange multiplier field. Standard shape functions have the advantage that they guarantee the gap function to be strictly positive, which may significantly improve the robustness of the resulting problem formulation. Therefore, the Petrov–Galerkin approach unifies the advantages of both the dual and the standard mortar method.

Now, the (semi-) discrete version of the hybrid variational formulation of (5.26) can be given: Find $\left(\boldsymbol{u}^{\mathrm{h}}, \boldsymbol{\lambda}^{\mathrm{h}}\right) \in \boldsymbol{\mathcal{U}}_t^{\mathrm{h}} \times \boldsymbol{\mathcal{M}}_{+}^{\mathrm{h}}$, such that for all $t \in [0, T]$

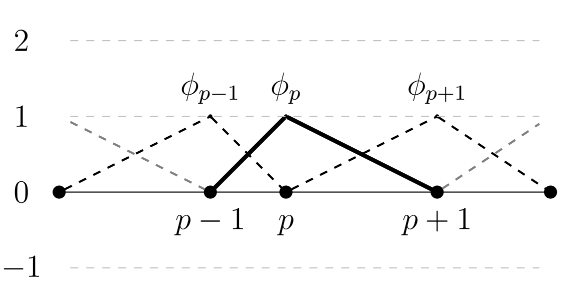
$$m\left(\ddot{\boldsymbol{u}}^{\mathrm{h}}, \boldsymbol{v}^{\mathrm{h}}\right) + a\left(\boldsymbol{u}^{\mathrm{h}}, \boldsymbol{v}^{\mathrm{h}}\right) + \left\langle \left[\boldsymbol{v}^{\mathrm{h}}\right], \boldsymbol{\lambda}^{\mathrm{h}} \right\rangle_{\gamma_{\mathrm{c}}^{(s)}} = f(\boldsymbol{v}^{\mathrm{h}}), \qquad \boldsymbol{v}^{\mathrm{h}} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{\mathrm{h}}, \tag{5.32a}$$

$$\left\langle g_{\mathrm{n}}, \mu_{\mathrm{n}} - \lambda_{\mathrm{n}} \right\rangle_{\gamma_{\mathrm{c}}^{(s)}} \geq 0, \qquad \boldsymbol{\mu} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{\lambda}}, \tag{5.32b}$$

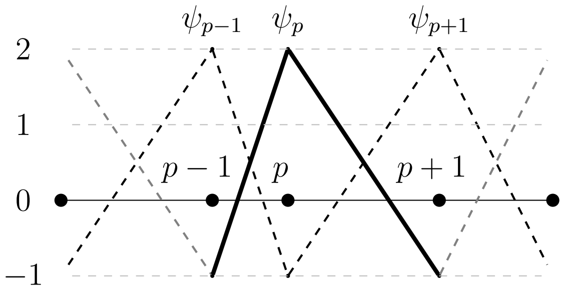
holds with the weak initial conditions

$$\left(\boldsymbol{u}_0^{\mathrm{h}}, \boldsymbol{v}^{\mathrm{h}}\right)_0 = \left(\widehat{\boldsymbol{u}}_0, \boldsymbol{v}^{\mathrm{h}}\right)_0, \quad \left(\dot{\boldsymbol{u}}_0^{\mathrm{h}}, \boldsymbol{v}^{\mathrm{h}}\right)_0 = \left(\widehat{\dot{\boldsymbol{u}}}_0, \boldsymbol{v}^{\mathrm{h}}\right)_0, \quad \boldsymbol{v}^{\mathrm{h}} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{\mathrm{h}}. \tag{5.33}$$

Remark 5.4.3 (Frictional case). Considering frictional contact in detail is far beyond the scope of this thesis as frictional effects are not essential for the methods developed in the next chapters.



(a) 1D standard basis functions.



(b) 1D dual basis functions.

Figure 5.2.: Comparison of standard and dual basis functions for $d = 2$.

The interested reader is referred to Gitterle [75], Gitterle et al. [76], Hüeber et al. [98], Puso and Laursen [163] and Wohlmuth [224] for more details on the mortar finite element discretization of frictional contact, stating that this list of literature is by far not complete.

5.5. Algebraic representation

Next, the algebraic formulation is derived in a matrix-vector notation based on nodal values. Since one is primarily interested in the algebraic structure of the resulting linear systems, it is sufficient to consider the frictionless quasi-static case. This way, additional algorithmic complexities arising from the time integration are circumvented, while preserving the algebraic structure of the final problems, which is what is essential for developing adapted multigrid methods. So, instead of (5.32), the following problem is considered:

Find $\left(\boldsymbol{u}^{\mathrm{h}}, \boldsymbol{\lambda}^{\mathrm{h}}\right) \in \boldsymbol{\mathcal{U}}^{\mathrm{h}} \times \boldsymbol{\mathcal{M}}_{+}^{\mathrm{h}}$, such that

$$a\left(\boldsymbol{u}^{\mathrm{h}}, \boldsymbol{v}^{\mathrm{h}}\right) + \left\langle \left[\boldsymbol{v}^{\mathrm{h}}\right], \boldsymbol{\lambda}^{\mathrm{h}} \right\rangle_{\gamma_{\mathrm{c}}^{(S)}} = f\left(\boldsymbol{v}^{\mathrm{h}}\right), \qquad \boldsymbol{v}^{\mathrm{h}} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{u}}^{\mathrm{h}}, \tag{5.34a}$$

$$\left\langle g_{\mathrm{n}}, \mu_{\mathrm{n}}^{\mathrm{h}} - \lambda_{\mathrm{n}}^{\mathrm{h}} \right\rangle_{\gamma_{\mathrm{c}}^{(S)}} \geq 0, \qquad \boldsymbol{\mu}^{\mathrm{h}} \in \boldsymbol{\mathcal{M}}_{+}^{\mathrm{h}}, \tag{5.34b}$$

with the initial conditions (5.33).

The nodes of the mesh $\mathcal{T}_{\mathrm{h},\Omega_0}$ are connected and form elements, which allow to formulate the approximate partitioning of the domain Ω_0 into $m_{\Omega_0}^{(\mathrm{e})}$ element subdomains by $\Omega_0 \approx \bigcup_{e=1}^{m_{\Omega_0}^{(\mathrm{e})}} \Omega_0^{(e)}$. For each element e, the discrete displacement solution $\boldsymbol{u}^{\mathrm{h},(\mathrm{e})}$ can be expressed element-wise as

$$\boldsymbol{u}^{\mathrm{h},(\mathrm{e})}(\boldsymbol{X}) = \sum_{p=1}^{m_0^{\boldsymbol{u},(\mathrm{e})}} \phi_p^{(\mathrm{e})}(\boldsymbol{X}) \boldsymbol{u}_p, \tag{5.35}$$

where $\boldsymbol{u}_p$ denotes the discrete nodal values of the displacement and $m_0^{\boldsymbol{u},(\mathrm{e})}$ describes the number of nodes associated with element e. The interpolation functions $\phi_p^{(\mathrm{e})}$ denote the (element) shape functions, that are involved in describing the element e.

The individual contributions of all elements e are sorted and assembled into a global nodal solution vector

$$\boldsymbol{u} = \left(\boldsymbol{u}_{\mathcal{N}_1}, \boldsymbol{u}_{\mathcal{M}}, \boldsymbol{u}_{\mathcal{S}}, \boldsymbol{u}_{\mathcal{N}_2}\right)^{\mathsf{T}}. \tag{5.36}$$

Therein, the $\boldsymbol{u}_{\mathcal{N}_i}$, $i \in \{1,2\}$, contains all degrees of freedom associated with the mesh nodes of the corresponding solid body without the nodes at the contact interface (master or slave). The degrees of freedom associated with the contact interface on the master and slave side are represented by $\boldsymbol{u}_{\mathcal{M}}$ and $\boldsymbol{u}_{\mathcal{S}}$, respectively. Be aware that the same notation $\boldsymbol{u}$ is used both for the (continuous) solution variable (cf. Section 5.2 and 5.3) and its (discrete) nodal vector representation. This simplifies the notation by avoiding an additional variable. The respective meaning of $\boldsymbol{u}$ is always obvious from the context. In the following, $\boldsymbol{u}$ always represents the global solution vector with the nodal displacements.

In a similar way one can describe the discrete Lagrange multipliers

$$\boldsymbol{\lambda}^{\mathrm{h}} = \sum_{p=1}^{m_0^{\boldsymbol{\lambda}}} \psi_p \boldsymbol{\lambda}_p, \tag{5.37}$$

where $m_0^{\boldsymbol{\lambda}}$ denotes the number of (slave) nodes carrying additional Lagrange multiplier degrees of freedom and $\boldsymbol{\lambda}_p$ describes the corresponding discrete nodal Lagrange multipliers. Again, all contributions $\boldsymbol{\lambda}_p$ can be assembled into a global nodal solution vector $\boldsymbol{\lambda}$ for the Lagrange multipliers. The meaning of $\boldsymbol{\lambda}$ again depends on the context, i.e., whether it describes the (continuous) Lagrange multipliers (cf. Section 5.2 and 5.3) or the global vector of nodal Lagrange multiplier degrees of freedom (Section 5.5 as well as Chapter 6 and Chapter 7).

The final spatially discretized formulation of the quasi-static frictionless problem (5.34) using the nodal vector representation emerges as

$$\mathsf{f}_{\text{int}}(\boldsymbol{u}) + \mathsf{f}_{\text{co}}(\boldsymbol{u}, \boldsymbol{\lambda}) = \mathsf{f}_{\text{ext}}, \tag{5.38}$$

$$-\left(\tilde{g}_{\mathrm{n,h}}\right)_j \leq 0, \quad \left(\lambda_{\mathrm{n}}\right)_j \geq 0, \quad \left(\tilde{g}_{\mathrm{n,h}}\right)_j \left(\lambda_{\mathrm{n}}\right)_j = 0, \quad j = 1, \ldots, m_0^{\boldsymbol{\lambda}}, \tag{5.39}$$

$$\left(\boldsymbol{\lambda}_\tau\right)_j = \mathbf{0}, \quad j = 1, \ldots, m_0^{\boldsymbol{\lambda}} \tag{5.40}$$

and is explained briefly in the following:

In analogy to the variational formulation (5.34a), the system (5.38) contains the global vector of nonlinear internal forces f_{int} and the global vector of external forces f_{ext}. The discrete vector of contact forces f_{co} is defined by $\mathsf{f}_{\text{co}}(\boldsymbol{u}, \boldsymbol{\lambda}) := \mathsf{C}(\boldsymbol{u})^{\mathsf{T}} \boldsymbol{\lambda}$ acting on slave and master sides of the contact interface and depends non-linearly on the current deformation $\boldsymbol{u}$. Therein, the discrete mortar contact operator has the form

$$\mathsf{C}(\boldsymbol{u}) := \begin{pmatrix} 0 \\ -\mathsf{M}^{\mathsf{T}} \\ \mathsf{D}^{\mathsf{T}} \\ 0 \end{pmatrix}. \tag{5.41}$$

Following the node ordering in (5.36), the zero blocks in (5.41) refer to lines associated with the inner nodes in the set $\mathcal{N}_i$, $i \in \{1,2\}$, away from the contact interface. The mortar coupling blocks M and D result from the discrete coupling conditions corresponding to the contact interface at the mortar (master) and non-mortar (slave) side.

The discretized version of the weak formulation in (5.34b) is equivalent to the set of point-wise normal contact constraints given by (5.39). Therein, the discrete weighted gap function $\left(\widetilde{g}_{\mathrm{n,h}}\right)_j$ at the slave node j is defined by $\left(\widetilde{g}_{\mathrm{n,h}}\right)_j = \int_{\gamma_{\mathrm{c}}^{(s)}} \psi_j g_{\mathrm{n,h}} \mathrm{d}s$ for $g_{\mathrm{n,h}}$ being a discrete version of the gap function g_{n} introduced in (5.8). A close look reveals that (5.39) basically represents a discrete version of the KKT conditions in (5.15a) to (5.15c) for normal contact with an additional weighting based on the Lagrange multiplier shape functions ψ_j.

A more detailed derivation of the discrete version of the contact constraints and the frictional terms is beyond the scope of the thesis. The reader may refer to Popp et al. [158] or Hüeber [99] if a more mathematical notation is preferred. Details especially on the formulation of the friction terms can also be found in Gitterle et al. [76], Hüeber et al. [98] and Gitterle [75].

For the following sections it is primarily important to understand that one has a discrete system of nonlinear equations with additional contact constraints which has to be solved numerically. Considering only the quasi-static case has the advantage that one does not have to deal with time integration. A full derivation of the algebraic formulation of contact problems including time integration and frictional terms is given, e.g., in Popp [156].

5.5.1. Semi-smooth Newton method

Here, the focus is on the nonlinear solution strategy for problem (5.38) with the constraints (5.39) and (5.40). The challenge is to incorporate the inequality constraints for the normal contact constraints (5.39) in the formulation of the nonlinear problem. A primal-dual active set strategy (or equivalently a semi-smooth Newton method, see Hintermüller et al. [93]) is used to resolve the contact non-linearity resulting from the inequality constraints as briefly outlined in Algorithm 9. Here, the intention is to give only a brief introduction to semi-smooth Newton methods. The basic idea of the primal-dual active set strategy is to rearrange the KKT conditions for the contact constraints, such that a Newton–Raphson like algorithm can be applied not only for the geometrical and material non-linearities, but also for the non-linearity resulting from the contact constraints. Let $\mathcal{S}$ denote the set of slave contact nodes. To reformulate the contact constraints (5.39) one has to introduce a semi-smooth nonlinear complementary (NCP) function C_j, which transforms the normal contact inequality constraints from (5.39) into an equality constraint equation, viz.

$$C_j(\boldsymbol{u},\boldsymbol{\lambda}) := \left(\lambda_{\mathrm{n}}\right)_j - \max\Big(0, \left(\lambda_{\mathrm{n}}\right)_j - c_{\mathrm{n}}\left(\widetilde{g}_{\mathrm{n,h}}\right)_j\Big) = 0, \quad c_{\mathrm{n}} > 0 \tag{5.42}$$

for each slave node $j \in \mathcal{S}$. One can easily verify that the equality constraint $C_j = 0$ is equivalent to the complete set of KKT conditions in (5.39) for arbitrary positive values of the complementarity parameter c_{n}. Here, the details are skipped and the reader is referred to Wohlmuth [224] and Popp [156] for a more elaborate discussion on the design of semi-smooth complementary functions.

Internally, one has to manage two disjoint subsets $\mathcal{A}$ and $\mathcal{I}$ of the slave nodes $\mathcal{S}$ with $\mathcal{A} \cup \mathcal{I} = \mathcal{S}$ and $\mathcal{A} \cap \mathcal{I} = \emptyset$. Here, $\mathcal{A}$ contains all *active* nodes which are currently in contact, and $\mathcal{I}$ describes the set of nodes which are currently not in contact. For numerically solving the constrained nonlinear problem (5.38) to (5.40) the nonlinear residual $\mathbf{r}(\boldsymbol{u},\boldsymbol{\lambda}) := \mathbf{f}_{\mathrm{int}}(\boldsymbol{u}) + \mathbf{f}_{\mathrm{co}}(\boldsymbol{u},\boldsymbol{\lambda}) - \mathbf{f}_{\mathrm{ext}}$ is defined, such that a Newton scheme can be applied to solve the nonlinear problem $\mathbf{r}(\boldsymbol{u},\boldsymbol{\lambda}) = \mathbf{0}$ subject to the constraints (5.39) and (5.40). In each iteration i of the resulting Newton loop, the linearized constrained system

$$\Big(\mathbf{K}_{\mathrm{T}}(\boldsymbol{u}^i) + \mathbf{K}_{\mathrm{co}}(\boldsymbol{u}^i,\boldsymbol{\lambda}^i)\Big)\Delta\boldsymbol{u}^{i+1} + \mathbf{C}(\boldsymbol{u}^i)\boldsymbol{\lambda}^{i+1} = -\mathbf{r}(\boldsymbol{u}^i,\boldsymbol{\lambda}^i), \tag{5.43a}$$

$$\left.\frac{\partial\big(\widetilde{g}_{\mathrm{n,h}}\big)_j}{\partial\boldsymbol{u}}\right|^i \Delta\boldsymbol{u}^{i+1} = -\big((\widetilde{g}_{\mathrm{n,h}})_j\big)^i \qquad \forall j\in\mathcal{A}^i, \tag{5.43b}$$

$$\big(\boldsymbol{\lambda}_{\mathcal{I}}\big)^{i+1} = \mathbf{0}, \tag{5.43c}$$

$$\left.\frac{\partial\big(\boldsymbol{\tau}_j^{\xi}\big)}{\partial\boldsymbol{u}}\right|^i \Delta\boldsymbol{u}^{i+1}\big(\boldsymbol{\lambda}_j\big)^i + \big(\boldsymbol{\tau}_j^{\xi}\big)^i\big(\boldsymbol{\lambda}_j\big)^{i+1} = 0 \qquad \forall j\in\mathcal{S}^i, \tag{5.43d}$$

$$\left.\frac{\partial\big(\boldsymbol{\tau}_j^{\eta}\big)}{\partial\boldsymbol{u}}\right|^i \Delta\boldsymbol{u}^{i+1}\big(\boldsymbol{\lambda}_j\big)^i + \big(\boldsymbol{\tau}_j^{\eta}\big)^i\big(\boldsymbol{\lambda}_j\big)^{i+1} = 0 \qquad \forall j\in\mathcal{S}^i \tag{5.43e}$$

is solved. Note that for presentation purposes, a semi-incremental formulation is chosen with displacement increments $\Delta\boldsymbol{u}^{i+1}$ and the Lagrange multipliers $\boldsymbol{\lambda}^{i+1}$ as solution variables in (5.43). The tangential stiffness matrix $\mathbf{K}_{\mathrm{T}}$ and the linearized contact forces are defined by

$$\mathbf{K}_{\mathrm{T}}(\boldsymbol{u}^i) := \left.\frac{\partial\big(\mathbf{f}_{\mathrm{int}}(\boldsymbol{u}) - \mathbf{f}_{\mathrm{ext}}\big)}{\partial\boldsymbol{u}}\right|^i \text{ and } \mathbf{K}_{\mathrm{co}}(\boldsymbol{u}^i,\boldsymbol{\lambda}^i) := \left.\frac{\partial\big(\mathbf{C}(\boldsymbol{u})\boldsymbol{\lambda}\big)}{\partial\boldsymbol{u}}\right|^i \tag{5.44}$$

with $\mathbf{C}(\boldsymbol{u})$ from (5.41). Equation (5.43b) describes the linearized normal contact constraint for the active nodes and (5.43c) defines the results for the Lagrange multipliers corresponding to inactive nodes. The constraints in (5.43d) and (5.43e) result from the consistent linearization of the frictionless tangential contact constraint (5.40). The two vectors $\boldsymbol{\tau}_j^{\xi}$ and $\boldsymbol{\tau}_j^{\eta}$ are defined as the tangent vectors building an orthonormal basis with the current node normal vector $\boldsymbol{n}_j$ at each slave point $j\in\mathcal{S}$. Therefore, it is $\boldsymbol{n}_j\cdot\boldsymbol{\tau}_j^{\xi} = 0$ and $\boldsymbol{\tau}_j^{\eta} = \boldsymbol{n}_j\times\boldsymbol{\tau}_j^{\xi}$.

The complementarity function (5.42) is also used to define the updated set of active and inactive nodes using

$$\begin{aligned}\mathcal{I}^i &:= \Big\{ j\in\mathcal{S} \;:\; \big((\lambda_{\mathrm{n}})_j\big)^i - c_{\mathrm{n}}\big((\widetilde{g}_{\mathrm{n,h}})_j\big)^i \le 0 \Big\} \text{ and}\\ \mathcal{A}^i &:= \Big\{ j\in\mathcal{S} \;:\; \big((\lambda_{\mathrm{n}})_j\big)^i - c_{\mathrm{n}}\big((\widetilde{g}_{\mathrm{n,h}})_j\big)^i > 0 \Big\}\end{aligned} \tag{5.45}$$

in each nonlinear iteration i of the semi-smooth Newton method.

Finally, it is checked whether the set of active nodes has converged and whether the current solution variables satisfy the convergence criteria $\|\mathscr{R}(\boldsymbol{u}^{i+1},\boldsymbol{\lambda}^{i+1})\| \le \varepsilon$ in some norm. Usually, the convergence criteria in $\mathscr{R}$ are a combination of different norms of scaled partial residual vectors of the different physical and mathematical fields. An appropriate choice for $\mathscr{R}$ is highly problem-specific.

As already mentioned before, many details of the semi-smooth Newton method are far beyond the scope of this thesis. A mathematical description of the semi-smooth Newton method can be found, e.g., in Christensen and Pang [48], Qi and Sun [165] or Christensen [47]. For the concept of primal-dual active set strategies in the framework of abstract variational inequalities one may refer to Hintermüller et al. [93] and Hintermüller et al. [94]. In context of contact problems

Algorithm 9: Semi-smooth Newton algorithm

```
*Initialize nonlinear solver*
$i \leftarrow 0$
Initialize $\mathcal{A}^0$ and $\mathcal{I}^0$, such that $\mathcal{A}^0 \cup \mathcal{I}^0 = \mathcal{S}$ and $\mathcal{A}^0 \cap \mathcal{I}^0 = \emptyset$
Initialize solution vector: $\big(\boldsymbol{u}^0, \boldsymbol{\lambda}^0\big)$
*Nonlinear Newton loop*
**for** $i \leftarrow 0$ **to** $i_{\max} - 1$ **do**
    *Calculate Newton update*
    Find the primal-dual pair $\big(\Delta\boldsymbol{u}^{i+1}, \boldsymbol{\lambda}^{i+1}\big)$ by solving
        $$\Big(\mathsf{K}_{\mathrm{T}}(\boldsymbol{u}^i) + \mathsf{K}_{\mathrm{co}}(\boldsymbol{u}^i, \boldsymbol{\lambda}^i)\Big)\Delta\boldsymbol{u}^{i+1} + \mathsf{C}(\boldsymbol{u}^i)\boldsymbol{\lambda}^{i+1} = -\mathbf{r}(\boldsymbol{u}^i, \boldsymbol{\lambda}^i)$$
    subject to the linearized constraints
        $$\left.\frac{\partial\big(\widetilde{g}_{\mathrm{n,h}}\big)_j}{\partial\boldsymbol{u}}\right|^i \Delta\boldsymbol{u}^{i+1} = -\big((\widetilde{g}_{\mathrm{n,h}})_j\big)^i \qquad \forall j \in \mathcal{A}^i,$$
        $$\big(\boldsymbol{\lambda}_j\big)^{i+1} = \mathbf{0} \qquad \forall j \in \mathcal{I}^i,$$
        $$\left.\frac{\partial\big(\boldsymbol{\tau}_j^{\xi}\big)}{\partial\boldsymbol{u}}\right|^i \Delta\boldsymbol{u}^{i+1}\big(\boldsymbol{\lambda}_j\big)^i + \big(\boldsymbol{\tau}_j^{\xi}\big)^i\big(\boldsymbol{\lambda}_j\big)^{i+1} = 0 \qquad \forall j \in \mathcal{S}^i,$$
        $$\left.\frac{\partial\big(\boldsymbol{\tau}_j^{\eta}\big)}{\partial\boldsymbol{u}}\right|^i \Delta\boldsymbol{u}^{i+1}\big(\boldsymbol{\lambda}_j\big)^i + \big(\boldsymbol{\tau}_j^{\eta}\big)^i\big(\boldsymbol{\lambda}_j\big)^{i+1} = 0 \qquad \forall j \in \mathcal{S}^i.$$
    *Update solution vector*
    Update $\boldsymbol{u}^{i+1} \leftarrow \boldsymbol{u}^i + \Delta\boldsymbol{u}^{i+1}$
    *Update inactive and active sets*
    Set $\mathcal{A}^{i+1}$ and $\mathcal{I}^{i+1}$ to
        $$\mathcal{I}^{i+1} := \Big\{j \in \mathcal{S} \ : \ \big((\lambda_{\mathrm{n}})_j\big)^{i+1} - c_{\mathrm{n}}\big((\widetilde{g}_{\mathrm{n,h}})_j\big)^{i+1} \le 0\Big\},$$
        $$\mathcal{A}^{i+1} := \Big\{j \in \mathcal{S} \ : \ \big((\lambda_{\mathrm{n}})_j\big)^{i+1} - c_{\mathrm{n}}\big((\widetilde{g}_{\mathrm{n,h}})_j\big)^{i+1} > 0\Big\}.$$
    *Check for convergence*
    **if** $\mathcal{A}^{i+1} = \mathcal{A}^i$ **and** $\|\mathscr{R}(\boldsymbol{u}^{i+1}, \boldsymbol{\lambda}^{i+1})\| \le \varepsilon$ **then**
        **return** solution vector $\big(\boldsymbol{u}^{i+1}, \boldsymbol{\lambda}^{i+1}\big)$
    **end**
**end**
```

the primal-dual active set strategy is used and described in Stadler [180], Kunisch and Stadler [111] as well as Hüeber et al. [98] with the references therein. Whereas the work of Hüeber and Wohlmuth [96] is, e.g., restricted to small deformation mortar contact problems, the work in Popp et al. [158, 159] presents a consistent extension to finite deformation mortar contact.

5.5.2. Algebraic notation of the linear system

For developing efficient iterative solution strategies based on multigrid methods, one is primarily interested in the structure of the linear systems in (5.43). A (two-solid body) contact problem can be written in matrix vector notation as

$$\left(\begin{array}{ccccc|cc}
\mathsf{K}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{M}} & 0 & 0 & 0 & 0 & 0 \\
\mathsf{K}_{\mathcal{M}\mathcal{N}_1} & \mathsf{K}_{\mathcal{M}\mathcal{M}} & \mathsf{K}_{\mathcal{M}\mathcal{I}} & \mathsf{K}_{\mathcal{M}\mathcal{A}} & 0 & -\mathsf{M}_{\mathcal{I}}^{\mathsf{T}} & -\mathsf{M}_{\mathcal{A}}^{\mathsf{T}} \\
0 & \mathsf{K}_{\mathcal{I}\mathcal{M}} & \mathsf{K}_{\mathcal{I}\mathcal{I}} & \mathsf{K}_{\mathcal{I}\mathcal{A}} & \mathsf{K}_{\mathcal{I}\mathcal{N}_2} & \mathsf{D}_{\mathcal{I}\mathcal{I}}^{\mathsf{T}} & \mathsf{D}_{\mathcal{I}\mathcal{A}}^{\mathsf{T}} \\
0 & \mathsf{K}_{\mathcal{A}\mathcal{M}} & \mathsf{K}_{\mathcal{A}\mathcal{I}} & \mathsf{K}_{\mathcal{A}\mathcal{A}} & \mathsf{K}_{\mathcal{A}\mathcal{N}_2} & \mathsf{D}_{\mathcal{A}\mathcal{I}}^{\mathsf{T}} & \mathsf{D}_{\mathcal{A}\mathcal{A}}^{\mathsf{T}} \\
0 & 0 & \mathsf{K}_{\mathcal{N}_2\mathcal{I}} & \mathsf{K}_{\mathcal{N}_2\mathcal{A}} & \mathsf{K}_{\mathcal{N}_2\mathcal{N}_2} & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & \mathsf{I} & 0 \\
0 & \mathsf{N}_{\mathcal{M}} & \mathsf{N}_{\mathcal{I}} & \mathsf{N}_{\mathcal{A}} & 0 & 0 & 0 \\
0 & 0 & \mathsf{F}_{\mathcal{I}} & \mathsf{F}_{\mathcal{A}} & 0 & 0 & \mathsf{T}_{\mathcal{A}}
\end{array}\right)
\left(\begin{array}{c}
\Delta\boldsymbol{u}_{\mathcal{N}_1} \\ \Delta\boldsymbol{u}_{\mathcal{M}} \\ \Delta\boldsymbol{u}_{\mathcal{I}} \\ \Delta\boldsymbol{u}_{\mathcal{A}} \\ \Delta\boldsymbol{u}_{\mathcal{N}_2} \\ \hline \boldsymbol{\lambda}_{\mathcal{I}} \\ \boldsymbol{\lambda}_{\mathcal{A}}
\end{array}\right)
= -
\left(\begin{array}{c}
\mathbf{r}^{\boldsymbol{u}}_{\mathcal{N}_1} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{M}} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{I}} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{A}} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{N}_2} \\ \hline \mathbf{0} \\ \mathbf{g}_{\mathcal{A}} \\ \mathbf{0}
\end{array}\right) . \qquad (5.46)$$

For ease of notation, the indices for the current Newton iteration i are dropped in (5.46). Obviously, the 2×2 block matrix in (5.46) describes a linear system with a generalized saddle point structure. The upper left block contains the entries of the tangential stiffness matrix and linearized contact forces from (5.43a). Even though formulated for two solid bodies, the generalization for n solid bodies is straightforward and only a matter of notation. The upper right block represents the coupling block $\mathbf{C}(\boldsymbol{u})$ from (5.41). The sixth row in (5.46) is equivalent to the constraint (5.43c). Note that in our notation the matrix has 8 block rows but only 7 block columns as the contact constraints for the active nodes have been split into the normal part for the contact constraint (5.43b) and the tangential part for the consistent linearizations of the frictionless contact condition in (5.43d) and (5.43e). The discrete vector $\mathbf{g}_{\mathcal{A}}$ contains all weighted gap values associated with the active nodes at the contact interface.

Be aware that the linear system is non-symmetric due to the use of vector-valued Lagrange multipliers. It is worth to mention that a scalar Lagrange multiplier would be sufficient for constraint enforcement in the frictionless case, which also would lead to a symmetric system. However, a more general formulation is preferred which can be extended by frictional constraints resulting in non-symmetric problems anyway.

Brunßen et al. [45] describe several general algorithmic solution concepts for structural contact problems (without friction) including an *Inexact Newton strategy*, where already Ruge–Stüben AMG methods are used. That is, a linear system close to (5.46) is solved iteratively using a preconditioned GMRES solver (cf. Appendix A) within the semi-smooth Newton method (cf. Algorithm 9). Instead of classical AMG methods, smoothed aggregation AMG preconditioners as introduced in Chapter 3 are explored for using with contact problems in this thesis. Hence, equation (5.46) serves as a starting point for the next chapters to develop efficient preconditioning techniques based on smoothed aggregation AMG methods.

CHAPTER 6

Algebraic multigrid for contact problems in condensed formulation

The saddle point structure of the linear systems in (5.46) poses a challenge for iterative solvers and preconditioners such that standard iterative preconditioning methods cannot be used. By using so-called dual basis functions (see Section 5.4.2) one can condense out the Lagrange multipliers and avoid the saddle point structure resulting in a condensed linear systems which should be appropriate for standard iterative solvers. However, severe convergence problems are observed when applying such iterative solvers, which make large scale contact simulations unfeasible, as direct parallel linear solvers are not a reasonable alternative any more (cf. Popp [156]).

With the background knowledge of the underlying problem from Section 5 and the in-depth understanding of the multigrid algorithms one can identify the problems for the iterative solvers induced by the contact formulation. The central contribution of this chapter is to show how it is possible to obtain robust multigrid methods with only minimal changes to the standard algorithms as presented in the Chapters 3 and 4 to enable aggregation-based AMG preconditioners for condensed contact problems. Again one has maximal benefit from the flexible design of the proposed multigrid framework (cf. Section 3.2.1) which can easily be adapted to the application-specific needs. A cheap column permutation strategy helps to improve the matrix properties to make them work with iterative multigrid methods. Moreover, a general observer mechanism keeps track of potentially problematic matrix entries which allows the algorithms to react accordingly. This leads to a substantial improvement of the robustness of the iterative linear solution methods which can be considered a major step forward towards industrial applications. Even though originally motivated by problems arising from contact mechanics in condensed formulation, all strategies in this chapter are more general and work for all kind of linear systems.

To the best of the author's knowledge this work is the first which successfully applies smoothed aggregation AMG preconditioners to problems arising from contact mechanics in a condensed formulation. This is demonstrated for large scale examples with more than one million degrees of freedom.

6.1. Structural contact problems in condensed formulation

6.1.1. Condensation

Dual basis functions as described in Section 5.4.2 can be exploited to simplify the linear system (5.46) from Section 5.5.2. Due to the bi-orthogonality condition (5.30) of the dual basis functions, the mortar matrices $\mathsf{D}_{\mathcal{AI}}$ as well as $\mathsf{D}_{\mathcal{IA}}$ in (5.46) vanish and the mortar matrices $\mathsf{D}_{\mathcal{AA}}$ as well as $\mathsf{D}_{\mathcal{II}}$ reduce to diagonal matrices.

Consequently, the saddle point system from Section 5.5.2 has the form

$$\left(\begin{array}{ccccc:cc} \mathsf{K}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{M}} & 0 & 0 & 0 & 0 & 0 \\ \mathsf{K}_{\mathcal{M}\mathcal{N}_1} & \mathsf{K}_{\mathcal{M}\mathcal{M}} & \mathsf{K}_{\mathcal{M}\mathcal{I}} & \mathsf{K}_{\mathcal{M}\mathcal{A}} & 0 & -\mathsf{M}_{\mathcal{I}}^{\mathsf{T}} & -\mathsf{M}_{\mathcal{A}}^{\mathsf{T}} \\ 0 & \mathsf{K}_{\mathcal{I}\mathcal{M}} & \mathsf{K}_{\mathcal{I}\mathcal{I}} & \mathsf{K}_{\mathcal{I}\mathcal{A}} & \mathsf{K}_{\mathcal{I}\mathcal{N}_2} & \mathsf{D}_{\mathcal{I}\mathcal{I}}^{\mathsf{T}} & 0 \\ 0 & \mathsf{K}_{\mathcal{A}\mathcal{M}} & \mathsf{K}_{\mathcal{A}\mathcal{I}} & \mathsf{K}_{\mathcal{A}\mathcal{A}} & \mathsf{K}_{\mathcal{A}\mathcal{N}_2} & 0 & \mathsf{D}_{\mathcal{A}\mathcal{A}}^{\mathsf{T}} \\ 0 & 0 & \mathsf{K}_{\mathcal{N}_2\mathcal{I}} & \mathsf{K}_{\mathcal{N}_2\mathcal{A}} & \mathsf{K}_{\mathcal{N}_2\mathcal{N}_2} & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & \mathsf{I} & 0 \\ 0 & \mathsf{N}_{\mathcal{M}} & \mathsf{N}_{\mathcal{I}} & \mathsf{N}_{\mathcal{A}} & 0 & 0 & 0 \\ 0 & 0 & \mathsf{F}_{\mathcal{I}} & \mathsf{F}_{\mathcal{A}} & 0 & 0 & \mathsf{T}_{\mathcal{A}} \end{array}\right) \left(\begin{array}{c} \Delta\boldsymbol{u}_{\mathcal{N}_1} \\ \Delta\boldsymbol{u}_{\mathcal{M}} \\ \Delta\boldsymbol{u}_{\mathcal{I}} \\ \Delta\boldsymbol{u}_{\mathcal{A}} \\ \Delta\boldsymbol{u}_{\mathcal{N}_2} \\ \hdashline \boldsymbol{\lambda}_{\mathcal{I}} \\ \boldsymbol{\lambda}_{\mathcal{A}} \end{array}\right) = - \left(\begin{array}{c} \mathbf{r}_{\mathcal{N}_1}^{\boldsymbol{u}} \\ \mathbf{r}_{\mathcal{M}}^{\boldsymbol{u}} \\ \mathbf{r}_{\mathcal{I}}^{\boldsymbol{u}} \\ \mathbf{r}_{\mathcal{A}}^{\boldsymbol{u}} \\ \mathbf{r}_{\mathcal{N}_2}^{\boldsymbol{u}} \\ \hdashline \mathbf{0} \\ \mathbf{g}_{\mathcal{A}} \\ \mathbf{0} \end{array}\right) \tag{6.1}$$

with the diagonal matrix blocks $\mathsf{D}_{\mathcal{AA}}$ and $\mathsf{D}_{\mathcal{II}}$. The diagonal form of the mortar block D gives rise to the condensation of the Lagrange multipliers $\boldsymbol{\lambda}$ from (6.1) allowing to avoid the saddle point structure. First, the sixth row and column in the system matrix (6.1) is removed making use of the fact that $\boldsymbol{\lambda}_{\mathcal{I}} = \mathbf{0}$. This way, one gets rid of the Lagrange multipliers $\boldsymbol{\lambda}_{\mathcal{I}}$ associated with the inactive nodes. Secondly, one can condense the Lagrange multipliers $\boldsymbol{\lambda}_{\mathcal{A}}$ associated with the active nodes using

$$\boldsymbol{\lambda}_{\mathcal{A}} := \mathsf{D}_{\mathcal{AA}}^{-1}\big(-\mathbf{r}_{\mathcal{A}}^{\boldsymbol{u}} - \mathsf{K}_{\mathcal{AM}}\Delta\boldsymbol{u}_{\mathcal{M}} - \mathsf{K}_{\mathcal{AI}}\Delta\boldsymbol{u}_{\mathcal{I}} - \mathsf{K}_{\mathcal{AA}}\Delta\boldsymbol{u}_{\mathcal{A}} - \mathsf{K}_{\mathcal{A}\mathcal{N}_2}\Delta\boldsymbol{u}_{\mathcal{N}_2}\big) \tag{6.2}$$

from the fourth row in (6.1). Introducing $\mathsf{P}_{\mathcal{A}} := \mathsf{D}_{\mathcal{AA}}^{-1}\mathsf{M}_{\mathcal{A}}$, the condensed system has the form

$$\begin{pmatrix} \mathsf{K}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{M}} & 0 & 0 & 0 \\ \mathsf{K}_{\mathcal{M}\mathcal{N}_1} & \mathsf{K}_{\mathcal{MM}} + \mathsf{P}_{\mathcal{A}}^{\mathsf{T}}\mathsf{K}_{\mathcal{AM}} & \mathsf{K}_{\mathcal{MI}} + \mathsf{P}_{\mathcal{A}}^{\mathsf{T}}\mathsf{K}_{\mathcal{AI}} & \mathsf{K}_{\mathcal{MA}} + \mathsf{P}_{\mathcal{A}}^{\mathsf{T}}\mathsf{K}_{\mathcal{AA}} & \mathsf{P}_{\mathcal{A}}^{\mathsf{T}}\mathsf{K}_{\mathcal{A}\mathcal{N}_2} \\ 0 & \mathsf{K}_{\mathcal{IM}} & \mathsf{K}_{\mathcal{II}} & \mathsf{K}_{\mathcal{IA}} & \mathsf{K}_{\mathcal{I}\mathcal{N}_2} \\ 0 & \mathsf{N}_{\mathcal{M}} & \mathsf{N}_{\mathcal{I}} & \mathsf{N}_{\mathcal{A}} & 0 \\ 0 & -\mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{AA}}^{-1}\mathsf{K}_{\mathcal{AM}} & \mathsf{F}_{\mathcal{I}} - \mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{AA}}^{-1}\mathsf{K}_{\mathcal{AI}} & \mathsf{F}_{\mathcal{A}} - \mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{AA}}^{-1}\mathsf{K}_{\mathcal{AA}} & -\mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{AA}}^{-1}\mathsf{K}_{\mathcal{A}\mathcal{N}_2} \\ 0 & 0 & \mathsf{K}_{\mathcal{N}_2\mathcal{I}} & \mathsf{K}_{\mathcal{N}_2\mathcal{A}} & \mathsf{K}_{\mathcal{N}_2\mathcal{N}_2} \end{pmatrix} \Delta\boldsymbol{u} = -\mathbf{r} \tag{6.3}$$

with

$$\Delta\boldsymbol{u} = \begin{pmatrix} \Delta\boldsymbol{u}_{\mathcal{N}_1} \\ \Delta\boldsymbol{u}_{\mathcal{M}} \\ \Delta\boldsymbol{u}_{\mathcal{I}} \\ \Delta\boldsymbol{u}_{\mathcal{A}} \\ \Delta\boldsymbol{u}_{\mathcal{N}_2} \end{pmatrix} \quad \text{and} \quad \mathbf{r} = \begin{pmatrix} \mathbf{r}_{\mathcal{N}_1}^{\boldsymbol{u}} \\ \mathbf{r}_{\mathcal{M}}^{\boldsymbol{u}} + \mathsf{P}_{\mathcal{A}}^{\mathsf{T}}\mathbf{r}_{\mathcal{A}}^{\boldsymbol{u}} \\ \mathbf{r}_{\mathcal{I}}^{\boldsymbol{u}} \\ \mathbf{g}_{\mathcal{A}} \\ -\mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{AA}}^{-1}\mathbf{r}_{\mathcal{A}}^{\boldsymbol{u}} \\ \mathbf{r}_{\mathcal{N}_2}^{\boldsymbol{u}} \end{pmatrix} \tag{6.4}$$

the solution increment and right-hand side vector. There are no Lagrange multipliers in the solution vector left, but the system still contains the normal contact constraints in the fourth row without any difference compared to (6.1). The same is true for the right-hand side vector $\mathbf{r}$ in

(6.4), which also contains the discrete gap information in the fourth row corresponding to the normal contact constraints. Therefore, one should pay attention when defining reasonable stopping criteria for the nonlinear solver or change to a fully incremental formulation (also for the contact constraints). As one can see from the second and fifth row in (6.3), the information from the active nodes at the slave side is projected onto the master nodes and incorporated in the friction constraints using the mortar blocks. From the AMG perspective, the linear operator in (6.3) is very challenging. The fine level matrix is structurally non-symmetric containing mixed equations with a differing mathematical and physical origin (structural equations and contact constraints). Therefore, the standard aggregation routine from Section 3.3 can neither separate the two solid bodies that are involved in the example, nor distinguish effectively the different meaning of equations. The level smoothers may work fine for degrees of freedoms associated with the inner nodes, but the degrees of freedom representing the contact interface may turn out to be problematic. Nevertheless, by avoiding the saddle point structure, dual Lagrange multipliers give rise to more efficient iterative solvers and sparse local stiffness matrices. Furthermore, dual Lagrange multipliers are known to provide the same accuracy as standard Lagrange multipliers (cf. Wohlmuth [226]). Before beginning with an in-depth analysis of the linear operator in (6.3), a short review is given on other existing multigrid approaches for mortar methods.

6.1.2. Multigrid methods for mortar finite elements

In between one can find some publications (e.g., Hüeber et al. [97, 98]), which apply mortar finite element methods with dual basis functions to contact problems. Some of them even mention the usage of multigrid methods as solution strategies but without giving concrete details.

In Krause and Wohlmuth [110] as well as Wohlmuth and Krause [227] the authors introduce a multigrid method based on the unconstrained product space for mortar finite element discretizations making use of dual basis functions for the Lagrange multipliers. Since our contact formulation is based on mortar finite element techniques, it may be worth to study this method to develop an aggregation-based AMG method for the condensed contact formulation as introduced in Section 6.1.1.

In the following the method proposed in Wohlmuth and Krause [227] is transformed to our notation. It is based on the unconstrained solution space $\boldsymbol{\mathcal{U}}$, where the mortar constraints are satisfied using an internal projection. Algebraically, the authors suggest the usage of a modified restriction operator

$$R^{\text{mod}}_{\ell+1} := \left(I - \left(\mathsf{C}(\boldsymbol{u})\right)_{\ell+1} \mathsf{W}^{\mathsf{T}}_{\ell+1}\right) \widehat{R}_{\ell+1} \tag{6.5}$$

with the coupling matrix blocks from (5.41) and the new operator

$$\mathsf{W}^{\mathsf{T}}_{\ell+1} := \begin{pmatrix} 0 & 0 & \mathsf{D}^{-1} & 0 \end{pmatrix}. \tag{6.6}$$

Using (5.41), a simple calculation shows that $R^{\text{mod}}_{\ell+1}$ is given by

$$R^{\text{mod}}_{\ell+1} = \begin{pmatrix} I & & & \\ & I & -\mathsf{M}^{\mathsf{T}}\mathsf{D}^{-1} & \\ & & 0 & \\ & & & I \end{pmatrix}_{\ell+1} \widehat{R}_{\ell+1}. \tag{6.7}$$

Note that $\mathsf{M}^\mathsf{T}\mathsf{D}^{-1}$ in (6.7) corresponds to the transpose of the mortar projection operator P introduced in context of (6.3). If $\mathbf{W}_\ell^\mathsf{T} r_\ell = 0$, i.e., the fine level residuum r_ℓ vanishes after $\nu_1 \geq 1$ smoothing steps for the non-mortar slave degrees of freedom, it immediately follows from (6.7) that

$$\mathbf{W}_{\ell+1}^\mathsf{T} R_{\ell+1}^{\text{mod}} r_\ell = \mathbf{W}_{\ell+1}^\mathsf{T} r_{\ell+1} = 0. \tag{6.8}$$

As a consequence, $r_{\ell+1}$ also vanishes for the non-mortar slave degrees of freedom. In order to satisfy

$$\mathbf{W}_\ell^\mathsf{T} r_\ell = 0 \tag{6.9}$$

exactly on the fine level ℓ, a special class of level smoothers is necessary. One can easily modify existing smoothers by adding one post-processing step that fixes the smoothed solution vector to comply with (6.9).

From (6.7) it is clear that the modified restriction operator (6.5) ignores the slave side degrees of freedom in the sense that they are added to the master degrees of freedom using the mortar projection operator P. Hence, they do not take part in the multigrid coarsening process itself. This works as special level smoothers are used to adapt the solution increment vectors accordingly. In fact, with the modified restriction operators based on the mortar projection operator P and the special level smoothers there are quite a lot of intrinsic adaptions in the multigrid method necessary where the multigrid method is very closely tied to the mortar application. Note that the naive usage of the modified restriction operator would also lead to singular coarse level matrices (when a Galerkin type coarsening is used) and therefore makes further reformulations of the problem necessary.

The method is admittedly designed and proven to work for mortar finite element problems producing nice results (cf. Krause and Wohlmuth [110], Wohlmuth [223], Wohlmuth and Krause [227]). However, structural contact problems bring some more complexities (such as frictional terms and effects from the (nonlinear) contact search). Nevertheless, the method can serve as inspiration for adapted multigrid methods for structural contact problems in a condensed formulation.

6.1.3. Concept and outline

In this chapter a new aggregation-based AMG method is developed for structural contact problems in condensed formulation as given in (6.3). The basic concept is to keep necessary special adaptions to standard multigrid algorithms as minimal as possible. That is, the contact multigrid algorithms should not rely on special modifications of the transfer operators as the approach presented in Wohlmuth and Krause [227].

In Section 6.2 the effect of standard plain aggregation AMG methods is studied when applied to condensed contact problems as given in (6.3). Then, one uses the findings from Section 6.2 to develop the multigrid methods presented in the Sections 6.3 and 6.4. Our first approach with the Hybrid PA-AMG method in Section 6.3 transfers the possibly problematic slave nodes one-to-one to the coarsest level. This way, the direct solver on the coarsest level can take care of the problematic part of the linear system. In some sense, this idea is similar to the multigrid method for mortar finite elements from Wohlmuth and Krause [227] as reviewed in Section 6.1.2, since the nodes at the slave contact interface are not involved in the multigrid coarsening. While they are completely ignored and projected onto the master side in the method from Wohlmuth and Krause [227], they are all kept without coarsening in the approach of this thesis. Hence, there is no need for a modified restriction operator and special level smoother classes in our approach. One just has to adapt the input matrices to generate single node aggregates for the slave nodes making sure that the solution increment is not disturbed in the level smoothers. This is a trivial adaption of the input for the aggregation method without touching the standard algorithms.

With the Contact PA-AMG method in Section 6.4, a full multigrid method is established for structural contact problems, which also incorporates the slave degrees of freedom in the coarsening process, whenever possible. One applies a special column permutation strategy to algebraically fix the linear systems arising from structural contact problems to work with the standard level smoothers. Again, one can follow the philosophy of keeping the changes of the multigrid method minimal. Note that the underlying physics of the problem is not changed but the multigrid algorithms are just supplied with appropriate information.

6.2. Naive approach

As a starting point, first the aggregation-based AMG method as introduced in Section 3 is naively applied to a simple two solid bodies test example in a condensed formulation resulting in linear systems of the form (6.3). This allows us to study the problems, which make the standard aggregation-based AMG methods fail.

6.2.1. Two solid bodies example

A simple 3D contact example is used as shown in Figure 6.1. Looking exemplarily at the case $\alpha_y = \alpha_z = 0$, one has two solid bodies with the same material parameters using a NeoHookean material ($\rho_0 = 0.1\frac{kg}{m^3}$, $E = 10$ GPa, $\nu = 0.3$). The initial gap between the two solid bodies is 0.02 units. The upper solid body (size: $0.8 \times 0.8 \times 0.5$ units) is moving down with constant velocity along the normal to the contact interface towards the lower fixed solid body (size: $1.0 \times 1.0 \times 1.0$ units). If not stated otherwise, a $10 \times 10 \times 10$ mesh is used for the discretization of each solid body. The simulation needs 50 time steps with a time step size of $0.01s$ on 4 processors. After 6 time steps ($t = 0.06s$) both bodies come into contact and are deformed. In this example frictionless contact is assumed. The idea is to reduce contact-specific effects (contact search for active set of nodes) to a minimum, such that one can focus on the linear solvers. That is, the contact zone is not changing once the two solid bodies are in contact.

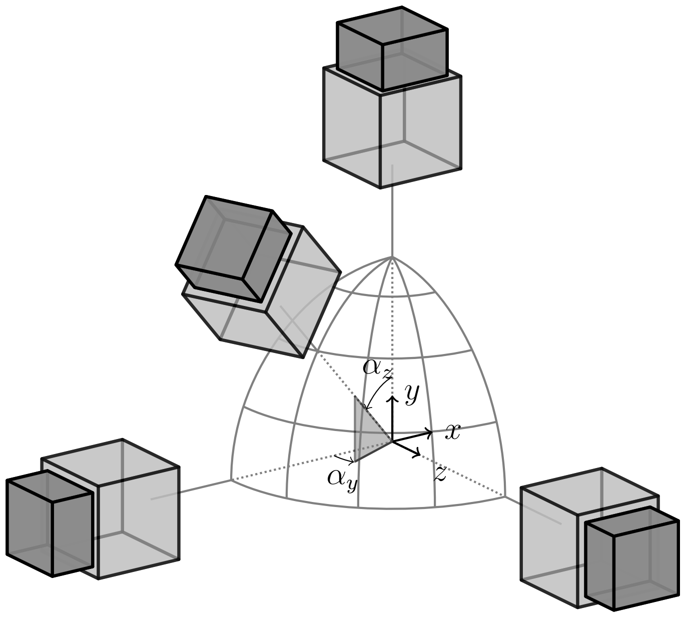
Of course, one expects the behavior of the linear solver to be independent of the exact geometric configuration. In particular, the number of linear iterations should be independent of the rotation angles α_y and α_z, since the underlying physics is not changing. For reasons of symmetry it is sufficient to vary α_y and α_z within $0 \leq \alpha_y, \alpha_z \leq \frac{\pi}{2}$.

6.2.2. Results for naive multigrid approach

The linear problem (6.3) has no saddle point structure any more due to the condensation of the Lagrange multipliers. Therefore, it is possible to apply some standard iterative methods to solve the linear system including aggregation-based AMG methods as described in Chapter 3. In particular, the choice of non-smoothed transfer operators (cf. Section 3.4) should result in a robust method that can deal with the topologically non-symmetric problem in (6.3).

In each time step, the nonlinear solver (see Section 5.5.1) is supposed to be converged if

$$\left\|\mathbf{r}^{\boldsymbol{u}}\right\|_e < 10^{-8} \ \wedge \ \left\|\Delta \boldsymbol{u}\right\|_e < 10^{-8} \tag{6.10}$$



Material:	NeoHooke
	$\rho_0 = 0.1 \frac{kg}{m^3}$
	$E = 10$ GPa
	$\nu = 0.3$
Timestep size:	$0.01s$
Time steps:	50
Mesh:	$10 \times 10 \times 10$

Figure 6.1.: Two solid bodies example – Geometric configuration and model parameters for two solid bodies example.

holds for the (nonlinear) residual $\mathbf{r}^{\boldsymbol{u}}$ and the displacement increment $\Delta \boldsymbol{u}$. The linear solver tolerance for the relative (linear) residual is set to

$$\left\| \frac{\mathbf{r}^k}{\mathbf{r}^0} \right\|_e < 10^{-8}, \tag{6.11}$$

where $\mathbf{r}^k$ denotes the (linear) residual in the iteration step $k = 0, \ldots, k_{\text{max}}$ of the linear GMRES solver.

Remark 6.2.1 (Stopping criteria). The combination of nonlinear and linear stopping criteria and solver tolerances in (6.10) and (6.11) are only exemplarily chosen and by no means meant to be optimal. The primary focus of this thesis is the behavior of the linear iterative solver. Therefore, a fixed tolerance is used for the linear system, which in practice may be too strong.

Table 6.1 gives the average number of linear GMRES iterations per nonlinear iteration over all 50 time steps for different rotation angles α_y and α_z. The number in brackets shows the maximum number of linear iterations needed in some time step during the simulation. The "–" means that the 50 time steps could not be finished successfully. Depending on the rotation angles α_y and α_z the simulation either fails to converge in some time step or even diverges immediately after first contact in the first linear system, when using a standard 3 level aggregation-based AMG preconditioner with non-smoothed transfer operators, 3 sweeps of damped symmetric Gauss–Seidel (0.7) as level smoothers and a direct solver on the coarsest level.

6.2.2.1. Non-diagonally dominant fine level matrix

A closer look at the specific problems reveals that, depending on the rotation angles α_y and α_z, the system matrix (6.3) becomes non-diagonally dominant. This is particularly the case if the normal-tangential coordinate system, that is used at the contact slave interface for the contact

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	–	–	–	–	–
	$\frac{1}{8}\pi$	–	–	–	–	–
	$\frac{1}{4}\pi$	–	57.1 (70)	–	–	–
	$\frac{3}{8}\pi$	25.8 (32)	–	–	–	–
	$\frac{1}{2}\pi$	23.9 (28)	–	–	–	–

Table 6.1.: Two solid bodies example – Average (maximum) number of linear GMRES iterations per nonlinear iteration (over all time steps) for different rotation angles α_y and α_z. A standard AMG (3 levels, 3 SGS (0.7) on all intermediate levels, direct solver on the coarsest level, minimum aggregate size: 9) is used as preconditioner.

(a) $\alpha_y = 0$ and $\alpha_z = 0$

$$\begin{pmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0.000 & \mathbf{1.000} & 0.000 & \cdots \\ \cdots & \mathbf{1.000} & 0.424 & -0.266 & \cdots \\ \cdots & -0.266 & -0.424 & \mathbf{1.000} & \cdots \\ & \vdots & \vdots & \vdots & \end{pmatrix}$$

(b) $\alpha_y = 0$ and $\alpha_z = \frac{1}{4}\pi$

$$\begin{pmatrix} & \vdots & \vdots & \vdots & \\ \cdots & \mathbf{1.000} & -1.000 & 0.000 & \cdots \\ \cdots & 0.404 & \mathbf{1.000} & -0.264 & \cdots \\ \cdots & 0.112 & -0.488 & \mathbf{1.000} & \cdots \\ & \vdots & \vdots & \vdots & \end{pmatrix}$$

(c) $\alpha_y = 0$ and $\alpha_z = \frac{1}{2}\pi$

$$\begin{pmatrix} & \vdots & \vdots & \vdots & \\ \cdots & \mathbf{1.000} & 0.000 & 0.000 & \cdots \\ \cdots & -0.424 & \mathbf{1.000} & -0.266 & \cdots \\ \cdots & 0.424 & -0.266 & \mathbf{1.000} & \cdots \\ & \vdots & \vdots & \vdots & \end{pmatrix}$$

Figure 6.2.: Two solid bodies example – 3×3 diagonal block from matrix A associated with slave node at contact interface resulting from two solid bodies example with a very coarse discretization ($2 \times 2 \times 2$ nodes per block) for given α_y and α_z. The matrix entries are equilibrated using the inverse of the maximum entry in each matrix row.

constraints, does not align with the cartesian coordinate system, that is used for the modeling of the structural equations. With the knowledge that the contact constraints are described in local coordinates based on the normal and tangential vectors at the contact interface, one finds that for $\alpha_y = \alpha_z = 0$ the normal vector at the contact interface is orthogonal to the cartesian x-axis (cf. Figure 6.1). Consequently, one obtains non-diagonally dominant rows for the corresponding nodes at the contact interface where the diagonal entries tend to zero. For the case $\alpha_y = 0$ and $\alpha_z = \frac{1}{2}\pi$, the normal-tangential coordinate system at the contact interface is aligned with the cartesian coordinate system. The resulting linear systems are diagonal dominant, such that the level smoother is effective (cf. Table 6.1). This can also be verified by the following simple example:

Example 6.2.2 (Diagonal dominance of rows corresponding to slave contact nodes)**.** The two solid bodies example from Section 6.2.1 is used with a very coarse discretization of $2 \times 2 \times 2$ nodes for each block. Then, the system matrix A in the first linear system in time step 6 after first contact occurs. One of the corner nodes at the contact interface is picked out for comparing the 3×3 block diagonal associated with that node in the system matrix A.

For this example, a node-wise ordering of the contact constraint equations is used with the normal contact constraint equation in the first row followed by two equations for the tangential constraints for each node at the contact interface. Figure 6.2 shows the 3×3 block diagonal of A associated with the chosen corner node for different combinations of α_y and α_z. The matrix entries have been equilibrated using the inverse of the maximum entry in each matrix row, such that the matrix is known to be diagonal dominant if the diagonal entries have all the absolute value of 1.0. As one can see from Figure 6.2a, the matrix rows associated with the slave nodes have a zero entry on the diagonal in the case $\alpha_y = \alpha_z = 0$. This example demonstrates that the matrix is not diagonal dominant if the normal vector at the contact interface is orthogonal to the cartesian x-axis. With the geometric configuration from Figure 6.1 it is obvious that for $\alpha_z = 0$ the normal vector at the contact interface is aligned with the cartesian y-axis. The consequence is that for such nodes the first row representing the contact normal constraint is related with the second column associated with the cartesian y-axis. Since the cartesian y-axis and the contact normal vector are perfectly aligned in case $\alpha_z = 0$, one obtains a zero entry on the diagonal in the first row. With an increasing angle α_z also the diagonal dominance of the 3×3 diagonal block is increasing, such that for $\alpha_z = \frac{1}{4}\pi$ one obtains a weakly diagonally dominant matrix block (see Figure 6.2b). For $\alpha_y = 0$ and $\alpha_z = \frac{1}{2}\pi$ the contact normal vector and the x-axis are aligned resulting in the diagonal dominant matrix given in Figure 6.2c.

So, one can see that the local ordering of the equations per node may cause extreme problems for the iterative smoothers within a multigrid method.

Remark 6.2.3 (Near null space). Beside the problems for the level smoother caused by the non-diagonally dominant matrix rows, there might also be a problem with the near null space vectors and the resulting pattern for the tentative transfer operators. For structural contact problems the rigid body modes provided by the finite element application are used as near null space vectors, which consist of the translatory modes and the rotatory modes in a fixed ordering based on the cartesian coordinates. If, e.g., the contact normal vector is orthogonal to the cartesian x-axis (e.g., for $\alpha_z = 0$ in Example 6.2.2), the nonzero pattern for the tentative prolongation operator is valid for a standard ordering in cartesian coordinates, but does not fit to the sparsity pattern of A with zeros on the diagonal. This hurts prerequisite **(N2)** in Section 3.4.1. In the worst case one obtains a singular coarse level matrix. For mortar contact formulations only linear momentum conservation is guaranteed (cf. Popp [156, Section 4.2.6]), which means that the constant translatory rigid body modes are a good approximation for the null space. However, angular momentum conservation is often not fulfilled exactly (at least without extra costs), such that the rotatory rigid body modes only provide a very rough approximation of the near null space. Nevertheless, in practice, one still can use the provided rigid body modes as approximation for the near null space as long as **(N2)** is not hurt.

6.2.2.2. Interface aggregates

With the system matrix given in (6.3) as input for the aggregation algorithm (see Section 3.3), the resulting aggregates may cross the contact interface. The aggregation algorithm cannot distinguish between the contact constraints and the structural equations which have a different character. Even though this is algebraically tolerable, it means from the physical point of view that the two solid bodies are melt together in the coarse representation (see also Figure 6.3a). From the mathematical point of view, both the information from the structural equations and the contact constraints are contained in the same aggregate. Consequently, on the coarse levels there is no clear distinction between structural equations and coarse contact constraints possible.

Therefore, aggregates that cross the contact interface may have some negative effect on the convergence not only when trying to reuse the full preconditioner (or only parts) for later calls to the linear solver. Mixing up contact constraints and structural equations in the same aggregate may degrade the overall behavior of the multigrid method, since the prolongation may not be able to recover the displacement variables appropriately by using mixed information from the coarse grid correction.

6.2.3. Multigrid approaches for non-diagonally dominant problems

Of course, one could apply node-based level smoothers, such as block Gauss–Seidel or block Jacobi methods, to overcome the issue of possibly non-diagonally dominant matrices. In contrary to DOF-based (or point-based) relaxation methods as introduced in Section 2.1.1, the block variants internally solve the local 3×3 (or 6×6) systems for each node using a direct solver, which again would internally use some pivoting strategy to improve the diagonal dominance and overcome issues with zeros on the diagonal. With the methods from Section 6.3 and 6.4, a completely different approach is followed: the input matrices are carefully adapted without changing the underlying problem to make it work for multigrid methods. So, instead of developing highly specialized multigrid strategies, no (or only minimal) adaptions are necessary in an existing multigrid code. With the Hybrid PA-AMG approach from Section 6.3 the coarsening at the contact interface is skipped and therefore the problems from the contact formulation during the multigrid setup phase are bypassed. In the Contact PA-AMG method from Section 6.4 one tries to improve the fine level input matrix A_0 by applying some permutation strategy together with some other enhancements to increase the overall robustness. The idea is somewhat similar to pivoting strategies known from direct solvers. But the global permutation strategy has the significant advantage that the permutation has to be done only once in the beginning for all possibly problematic matrix entries. Then, one can use the improved fine level matrix as input for standard multigrid methods even with DOF-based level smoothers. Once the input matrix is diagonally dominant, one can also apply standard transfer operator smoothing strategies without contact specific changes.

6.3. Hybrid direct-iterative AMG

As one can learn from the results in Section 6.2.2, the problematic rows in the system matrix (6.3) are associated with the nodes at the contact interface. In particular, the fourth and fifth block row in (6.3) and (6.4) turn out to be problematic with possibly non-diagonally dominant rows describing the contact (and frictional) constraints.

6.3.1. The Hybrid PA-AMG approach

In the Hybrid PA-AMG approach one tries to circumvent the problem of non-diagonally dominant rows which make standard relaxation-based level smoothers fail. It is based on the following two key concepts:

(H1) *Single node aggregates:* First, one puts all the slave nodes into single node aggregates to have a one-to-one representation of the slave contact degrees of freedom on all multigrid levels. Then, the direct coarse grid solver can take care of the problematic slave contact degrees of freedom. Figure 6.3b shows the aggregates used for the Hybrid PA-AMG method in comparison to standard aggregation without special handling of the contact slave nodes in Figure 6.3a for the two solid bodies example from Section 6.2.1. One can clearly see the layer of single node aggregates which separates the two solid bodies at the slave contact interface. Note that the aggregation parameters in Figure 6.3 have been adapted to produce similar coarsening rates for both cases. For producing single node aggregates only the input matrix for the aggregation routines is modified, such that no changes are necessary in the aggregation procedures.

(H2) *Level smoother input:* In addition to **(H1)**, the level smoothers must not disturb the fine level solution at the contact interface. In the Hybrid PA-AMG approach, one forces the level smoothers on the finest and intermediate levels to keep the entries in the solution vector corresponding to the slave node degrees of freedom to be fixed. This way one prevents the level smoothers from inciting error modes caused by the non-diagonally dominant rows. Technically, some artificial Dirichlet-like conditions are temporarily put on the slave contact interface rows of the input matrices for the level smoothers. So, no changes are necessary in the level smoother algorithms except a slight temporary modification of the input.

With **(H1)** and **(H2)** the contact constraints are transferred one-to-one to the coarsest level, where they are solved exactly by the direct coarse grid solver without disturbing the solution increment vector on the fine level for the slave contact degrees of freedom

Remark 6.3.1 (User provided information). The Hybrid PA-AMG algorithm needs the outer nonlinear semi-smooth Newton method to provide the information on the slave contact interface. As a consequence the Hybrid PA-AMG method is not a black-box method in the classical sense, since additional information has to be provided from outside.

Remark 6.3.2 (Implementation details). With the contact slave degrees of freedom provided by the outer semi-smooth Newton method, the extension of the aggregation method for single node aggregates is straightforward. The tricky part however comes with the local QR-decomposition for building the non-smoothed transfer operators $\widehat{P}$, since the QR-decomposition is not defined for problems with the number of degrees of freedom per node to be smaller than the number of null space vectors. In case of 3D elasticity one has 3 displacement degrees of freedom per node versus 6 near null space vectors described by the rigid body modes (cf. Remark 3.4.2). For single node aggregates it is necessary to transfer the corresponding fine level near null space information 1-to-1 to the coarse level appropriately. One can either bypass the local QR-decomposition or add support for a variable number of degrees of freedom per node. In our implementation the first option is used. Nevertheless, the necessary changes in the algorithms are minor compared to introducing special modified restriction operators and may help to increase the overall robustness of the multigrid algorithms.

6.3.2. Numerical results for test example

The Hybrid PA-AMG approach is tested using the same settings as in Section 6.2.2 with the same geometric setup and the solver tolerances as given in (6.10) and (6.11). Table 6.2 shows that the average number (maximum number) of linear iterations is constant for a wide range of

(a) Standard aggregation with aggregates crossing the contact interface

(b) Segregated aggregation with one point aggregates at the contact interface

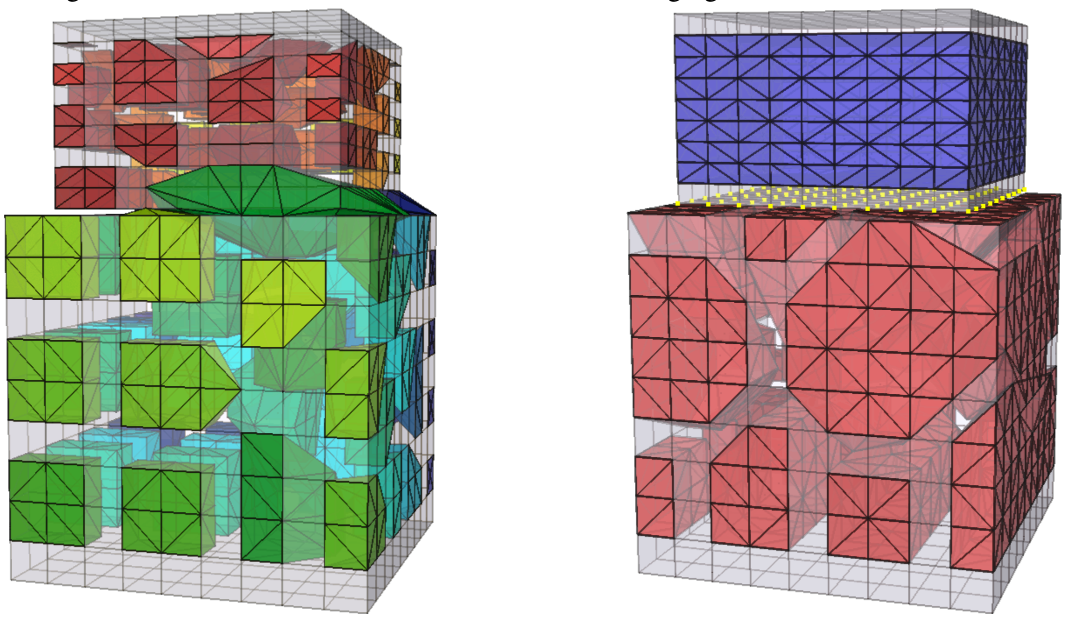


Figure 6.3.: Two solid bodies example – Comparison of standard aggregates and segregated aggregates of the Hybrid PA-AMG method in the reference coordinate system. In both cases the aggregation parameters have been chosen to produce a similar number of aggregates for a comparable coarsening rate.

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	32.1 (234)	31.7 (206)	46.7 (206)	31.6 (203)	32.0 (229)
	$\frac{1}{8}\pi$	29.1 (32)	29.1 (32)	29.1 (32)	29.1 (32)	32.0 (229)
	$\frac{1}{4}\pi$	29.1 (32)	29.1 (32)	29.1 (32)	29.1 (32)	32.0 (229)
	$\frac{3}{8}\pi$	29.1 (32)	29.1 (32)	29.1 (32)	29.0 (32)	32.0 (229)
	$\frac{1}{2}\pi$	29.1 (32)	29.1 (32)	29.1 (32)	29.1 (32)	36.1 (–)

Table 6.2.: Two solid bodies example – Average (maximum) number of linear GMRES iterations per nonlinear iteration (over all time steps) for different rotation angles α_y and α_z. Hybrid PA-AMG (3 levels, 3 SGS (0.7) on all intermediate levels, direct solver on the coarsest level, minimum aggregate size: 9) is used as preconditioner.

rotation angles α_y and α_z. Only for $\alpha_y = \frac{1}{2}\pi$ or $\alpha_z = 0$ the average number of iterations is disturbed by a significantly higher maximum number of iterations. A closer look at these problematic configurations reveals that only the first linear system within the nonlinear solver after first contact turns out to be hard to solve leading to the high iteration numbers in Table 6.2. This problem is related to temporary negative effects for the matrix conditioning resulting from the tangential coordinate vectors at the contact interface in relation to the global cartesian coordinate system, leading to nearly singular systems. However, it is known from our experiments that – once the contact is established – all other linear systems show exactly the same convergence behavior as for all other choices of α_y and α_z. One can overcome these type of issues by strategies introduced in Section 6.4.

Compared with the naive approach from Section 6.2, the Hybrid PA-AMG approach is a robust method that can deal with linear systems arising from the condensed contact formulation. Nevertheless, from the multigrid point of view the Hybrid PA-AMG method is not satisfactory, since the coarsening rate is suffering from the single node aggregates at the contact interface. For real world problems one will see a notable drop in the convergence rate, especially if the contact interface becomes dominant compared to the inner structural domain (see Section 6.5). In fact, the Hybrid PA-AMG method does not comply with the multigrid concept, since it is not using multigrid principles at the contact interface at all. This motivates the next section where a full multigrid method for structural contact problems is introduced.

6.4. Full AMG for structural contact problems

Whereas the Hybrid PA-AMG method treats all possibly problematic nodes in a very conservative way, one now tries to relax that behavior and use algebraic considerations to design a full multigrid method which also considers the nodes at the contact interface during the coarsening process.

6.4.1. Permutation strategy

For the experiments with the condensed contact formulations, one finds the system matrix A to be possibly non-diagonally dominant for the rows associated with the slave contact degrees of freedom. If the contact normal is orthogonal to the cartesian x-axis, one even can have zeros on the diagonal of A (see Example 6.2.2), which is disastrous for smoothing methods like Jacobi both for the level smoothing (cf. Section 2.1) and the optional transfer operator smoothing (cf. Section 3.5). It is a well known fact that relaxation-based methods (such as Jacobi or Gauss–Seidel) rely on the inverse of the diagonal of the system matrix A and therefore fail, if there are close-to-zero entries on the diagonal. Vice versa, it is evident that increasing the diagonal dominance of A is beneficial for convergence of relaxation-based methods. Therefore, row and column permutation operators Ψ_r and Ψ_c are introduced which are supposed to improve the diagonal dominance of the permuted linear system $\mathrm{A}^\Psi := \Psi_r \mathrm{A} \Psi_c$. From the solution x^Ψ of the permuted linear system $\mathrm{A}^\Psi x^\Psi = \Psi_r b$ one can easily retain the solution of the original non-permuted linear system by $x = \Psi_c x^\Psi$.

Remark 6.4.1 (Motivation). Note that the motivation behind the permutation is purely algebraic to improve the matrix properties for iterative solvers. By intention, the permutation operators are chosen to be $\Psi_r \neq \Psi_c$, such that A and A^Ψ are not unitarily equivalent. The idea is to improve the eigenvalue spectrum of the permuted operator A^Ψ compared to the original operator A.

Be aware that there are no assumptions made on the prerequisites of A. Therefore, one has no guarantee to find permutation operators Ψ_r and Ψ_c which improve the matrix properties with regard to the diagonal dominance of A.

6.4.1.1. Diagonal dominance in a multigrid context

As already stated in Chapter 2, a multigrid method can be understood as applying a sequence of cheap iterative smoothing methods on different levels, making use of the fact that high-oscillatory error components are damped out quickly on finer levels. Thus, it is clear that a multigrid method can only work, if the input matrix fulfills the minimum requirements for the level smoothers to damp out the error components effectively. For the common choice of relaxation-based smoothers this includes the diagonal dominance of the level matrix. To improve the diagonal dominance of the level matrices A_ℓ, it is worth to briefly discuss different ways to incorporate the permutation strategy in the multigrid context.

First of all, one could think about combining the permutation operators with the transfer operators to improve the diagonal dominance of the coarse level matrices, i.e.,

$$\mathrm{A}_1 = R_1^{\Psi_r} \mathrm{A}_0 P_1^{\Psi_c}, \tag{6.12}$$

where $R_1^{\Psi_r} := R_1 \Psi_r$ and $P_1^{\Psi_c} := \Psi_c P_1$ describe the permuted transfer operators. However, replacing the standard Galerkin product (2.6) with the non-permuted transfer operators P_1 and R_1 by the expression in (6.12) would not fix the convergence problem, since the level smoothers on the finest level $\ell = 0$ would still fail, when using the non-permuted non-diagonally dominant matrix A_0 as input. Instead of some special handling for the level smoothers on the finest level, it makes more sense to use the permuted matrix $\mathrm{A}_0^{\Psi} := \Psi_r \mathrm{A}_0 \Psi_c$ as input for the whole multigrid method both for the level smoothers on the finest level and the transfer operators. Then, one can also use the standard Galerkin product with the permuted level matrix, i.e.,

$$\mathrm{A}_1 = R_1 \big(\Psi_r \mathrm{A}_0 \Psi_c\big) P_1 = R_1 \mathrm{A}_0^{\Psi} P_1. \tag{6.13}$$

Note that 6.12 and (6.13) are equivalent.

Once the diagonal dominance of the input matrix has been restored by the permutation strategy, it is preserved on the coarser levels as long as reasonably smoothed transfer operators are used. Therefore, it is sufficient to fix the diagonal dominance of the input matrix A_0 on the finest level only. The reader is referred to Section 3.5 for more details on the principle of prolongation smoothing and to Appendix B for a small study on the influence of transfer operator smoothing to the diagonal dominance of the coarse level matrices.

6.4.1.2. Constrained permutation strategy

For reasons of algorithmic efficiency it is important to keep the computational costs for finding the permutation operators Ψ_r and Ψ_c low. Our permutation strategy is based on the following constraints:

(P1) *Column permutations only:* First of all, one chooses $\Psi_r = I$, i.e., one performs column permutations only. This way, one can half the computational costs per definition.

Remark 6.4.2 (Row permutations versus column permutations). Due to technical reasons in context of parallelization it is advantageous to perform column permutations only. Forcing the row permutation to be the identity operator inherently avoids communication of row information over processors, which would be necessary when permuting rows belonging to different processors. Column permutations do not affect the inter-processor communication pattern of the distributed level matrices A_ℓ.

(P2) *Slave degrees of freedom only:* It would be against intuition if additional knowledge about the problem and the linear system would not be helpful to increase efficiency and performance of the solution strategy. Assuming that the rows and columns associated with the slave degrees of freedom at the contact interface are known, one can further reduce the computational complexity: Instead of defining a global column permutation mapping $\psi : \{1, \ldots, n_0\} \to \{1, \ldots, n_0\}$, one restricts the set of candidates for column permutations to the slave degrees of freedom $\mathcal{D}_\mathcal{S}$ only. That is, one uses

$$\psi_\mathcal{S} : \mathcal{D}_\mathcal{S} \to \mathcal{D}_\mathcal{S}, i \mapsto \psi_\mathcal{S}(i). \tag{6.14}$$

Note that this simplification is optional, but can drastically reduce the computational costs depending on the number of slave degrees of freedom. One could even further reduce the number of candidates for column permutations by using the degrees of freedom corresponding to the active slave nodes $\mathcal{D}_\mathcal{A}$ instead of $\mathcal{D}_\mathcal{S}$.

(P3) *Node-internal column permutations only:* To improve the diagonal dominance of A, it is sufficient to allow only permutations of columns belonging to the same mesh node. This way, one can prevent an artificial mix-up of node information through global column permutations, but keep the underlying physical meaning of the degrees of freedom in each node. Another side-effect of this constraint is that the aggregation algorithm is not affected by such column permutations at all, since the node connectivity is not changed (cf. Section 3.3).

Considering above constraints the optimal column permutation strategy maximizes the diagonal dominance of the permuted matrix A_0^Ψ. For a mathematical description of the permutation operator Ψ_c the following optimization problem is introduced:

Definition 6.4.3 (Mathematical description of constrained permutation)**.** Let $\mathcal{D}_\mathcal{S}$ describe the $n_\mathcal{S}$ degrees of freedom associated with the $m_\mathcal{S}$ nodes $\mathcal{S}$ at the slave side of the contact interface. Furthermore, the mapping $\mathrm{n}|_{\mathcal{D}_\mathcal{S}}$ from Definition 3.3.1 restricted to the set of slave degrees of freedom $\mathcal{D}_\mathcal{S}$, defines the mapping

$$\mathrm{n}|_{\mathcal{D}_\mathcal{S}} : \mathcal{D}_\mathcal{S} \to \mathcal{S} \tag{6.15}$$

between the slave degrees of freedom and the corresponding slave nodes $\mathcal{S}$. Then, the bijective permutation mapping $\psi_\mathcal{S} : \mathcal{D}_\mathcal{S} \to \mathcal{D}_\mathcal{S}$, $i \mapsto \psi_\mathcal{S}(i)$ is mathematically defined by the solution of the constrained optimization problem

$$\begin{aligned} &\max_{\psi_\mathcal{S}} \prod_{i \in \mathcal{D}_\mathcal{S}} \left|\mathrm{A}_{i,\psi_\mathcal{S}(i)}\right| \\ \text{subject to} \quad &\mathrm{n}|_{\mathcal{D}_\mathcal{S}}\big(i\big) - \mathrm{n}|_{\mathcal{D}_\mathcal{S}}\big(\psi_\mathcal{S}(i)\big) = 0 \quad \forall i \in \mathcal{D}_\mathcal{S}. \end{aligned} \tag{6.16}$$

In (6.16) the $\mathrm{A}_{i,j}$ denotes the entry of the matrix A in the i-th row and j-th column. The column permutation operator follows directly from the permutation mapping $\psi_{\mathcal{S}}$ using

$$\Psi_c = \left(\Psi_c\right)_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ and } i \notin \mathcal{D}_{\mathcal{S}}, \\ 1 & \text{if } j = \psi_{\mathcal{S}}(i) \text{ and } i \in \mathcal{D}_{\mathcal{S}}, \\ 0 & \text{otherwise}, \end{cases} \tag{6.17}$$

for all $i, j = 1, \ldots, n_0$.

The maximum product transversal $\prod_{i \in \mathcal{D}_{\mathcal{S}}} \left|\mathrm{A}_{i,\psi(i)}\right|$ in (6.16) is a common way to measure the diagonal dominance of a matrix (cf. Duff and Koster [55], Olschowka and Neumaier [149]).

6.4.1.3. Computational complexity and further improvements

In general, it can be a quite complex task to find the optimal permutation maximizing a diagonal dominance criterion such as the maximum product transversal (cf. Duff and Koster [54, 55]). In our case, thanks to the constraint **(P3)**, the global optimum of the constrained optimization problem in (6.16) is easily found by solving a small optimization problem for each slave node. This way, one obtains the solution of the global constrained optimization problem as a combination of small optimization problems for each node. For a 2D problem one can either switch the two columns corresponding to the node or skip the column permutation for the node. In a 3D setting there are only six possible column permutations per node for the 3 corresponding matrix columns. Thus, the computational costs for finding the best permutation scale linearly with the number $m_{\mathcal{S}}$ of slave contact nodes $\mathcal{S}$. Usually, it holds $m_{\mathcal{S}} \ll m_0$, where m_0 denotes all nodes on the finest level, and therefore the computational costs can be considered to be reasonably small.

Remark 6.4.4 (Equilibration). Equilibration strategies are often used to improve the condition number of the matrices for better convergence of the iterative solution methods (see, e.g., Bauer [16], Sluis [179] and references therein). One can easily combine the column permutation with a simple row scaling strategy to further improve the condition number of the permuted matrix with nearly no additional extra costs, just by appropriately scaling the entries of Ψ_c. Of course, it is possible to use a more sophisticated equilibration strategy instead of a simple row scaling. An overview of different equilibration methods can be found, e.g., in Bradley [27].

6.4.1.4. Effect of permutation

To see the effect of the constrained column permutation strategy as described in Section 6.4.1.2, one can take a look at the two solid bodies test example from Section 6.2.1 again using a $10 \times 10 \times 10$ mesh for each of the two solid blocks. Thus, the contact interface at the slave side has also 10×10 nodes with altogether 300 degrees of freedom.

Figure 6.4 plots the maximum number of close-to-zero diagonal entries with absolute values smaller than 10^{-5} in the non-permuted equilibrated fine level matrix, which are known to be problematic for relaxation-based methods applied either for level smoothing or transfer operator smoothing. Comparing the data from Figure 6.4 with Figure 6.1, one finds that the column permutation can be skipped, if the angle between the contact normal and the cartesian x-axis is rather small. If the contact normal vector is orthogonal to the cartesian x-axis, one obtains 100 close-to-zero diagonal entries in time step 6 after first contact occurs. These entries belong

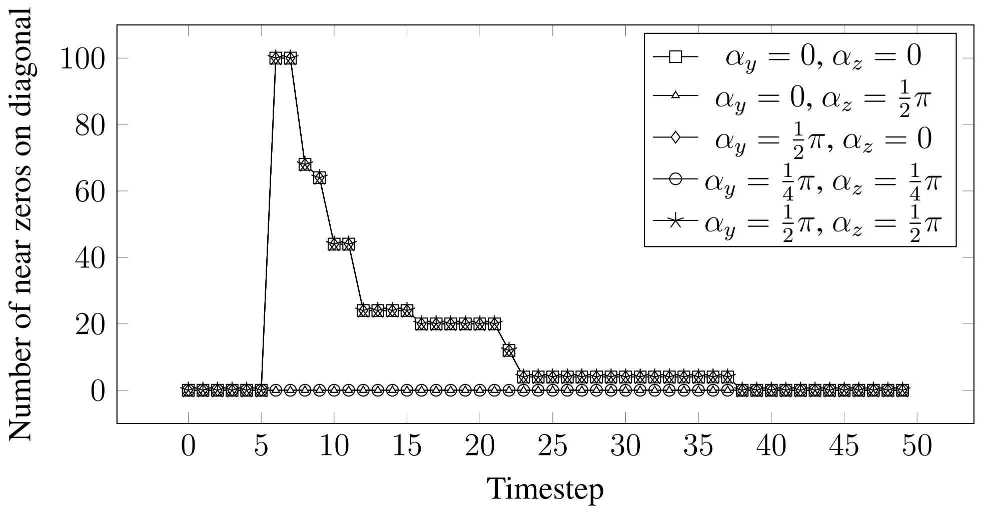


Figure 6.4.: Two solid bodies example – Maximum number of diagonal entries of the non-permuted system matrix with absolute values smaller than 10^{-5} per time step.

to the 100 degrees of freedom at the contact slave interface describing the scalar normal contact constraints. This corresponds to the effect observed in Example 6.2.2. Figure 6.4 also reveals, how the number of close-to-zero entries on the diagonal is reduced with evolving time. This may be the consequence of the deformation of the two solids at the contact interface resulting in an increased collinearity of the x-axis and the contact interface normal vector. Applying the column permutation enables the usage of iterative solvers for these linear systems by fixing problematic non-diagonally dominant matrix rows.

Table 6.3 gives the maximum number of columns permuted by the permutation strategy encoded in different colors in combination with the average number of non-diagonally dominant rows of the resulting permuted system matrix over the 50 time steps. Note that one can have 0, 2 or 3 permuted columns for each node. In our permutation algorithm the column permutation is performed if the permuted system has a better or equal maximum product transversal as defined in (6.16) than the non-permuted system. Obviously, one obtains diagonal dominant matrices, when the angle between the contact normal vector and the cartesian x-axis is smaller than 40.0 degrees. One can also see, how the maximum number of column permutations is increasing for greater angles α_y and α_z. The medium gray areas belong to the 200 permuted columns for the scalar normal contact constraints at the 100 slave nodes. The dark gray color stands for 300 permuted columns, that is, all columns corresponding to the slave interface degrees of freedom are permuted node-wise. The tick labels in the colorbar of Table 6.3 show the most frequent maximum numbers of permuted columns.

Once the columns are permuted, the resulting fine level matrix may still have some non-diagonally dominant entries. The permutation strategy is not perfect to fix the problem of locally rotated coordinate systems in all cases. This can be seen in particular for the combinations of α_y and α_z with a significant change of the number of permuted columns by the permutation strategy (denoted by the colors in Table 6.3). The number of possibly problematic rows and columns of the permuted linear systems is notably smaller than the expected maximum number

	α_y in [°]												
α_z in [°]	0.0	10.0	20.0	22.5	30.0	40.0	45.0	50.0	60.0	67.5	70.0	80.0	90.0
0.0	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1
10.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0
20.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0
22.5	14.2	13.8	13.8	13.8	13.8	13.8	13.8	13.8	13.8	13.8	13.8	13.8	14.2
30.0	17.7	17.1	15.4	14.7	12.2	10.6	10.5	10.6	10.8	13.3	14.1	16.9	17.7
40.0	17.9	17.8	17.6	17.7	19.3	16.3	13.6	11.4	17.5	20.9	21.0	18.5	17.8
45.0	21.2	27.5	26.9	26.6	25.2	23.8	20.7	16.3	24.4	28.2	29.7	34.8	23.4
50.0	17.9	18.1	25.1	26.4	29.3	27.6	27.5	31.6	77.1	79.0	73.5	23.1	17.9
60.0	17.7	17.7	17.8	17.7	17.9	24.3	66.2	96.6	92.5	26.4	25.2	18.5	17.6
67.5	14.2	13.4	11.4	11.5	13.9	21.3	57.0	82.6	47.6	40.8	32.0	16.0	14.2
70.0	14.0	14.0	11.5	10.8	14.1	19.8	54.1	79.0	29.3	42.3	40.3	15.2	14.0
80.0	14.0	14.0	13.9	12.9	16.7	18.2	50.2	22.2	18.6	16.6	15.9	25.5	14.0
90.0	15.1	14.0	14.0	14.2	17.7	17.8	23.4	17.9	17.6	14.2	14.0	14.0	15.2

Maximum number of column permutations

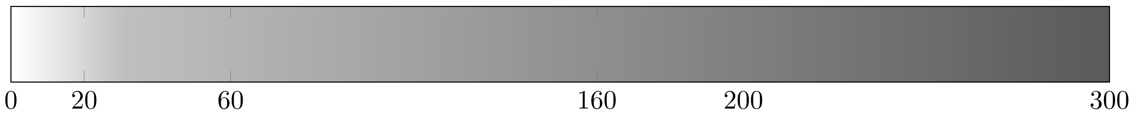


Table 6.3.: Two solid bodies example – Numbers: Average number of non-diagonally dominant rows of system matrix (after performing permutations) for different rotation angles α_y and α_z. Colors: Number of permuted columns to increase diagonal dominance of permuted matrix.

of 300 possibly non-diagonally dominant rows that would have been chosen by the Hybrid PA-AMG method in Section 6.3.1 for special handling with single node aggregates.

Remark 6.4.5 (Near null space and column permutation). Assuming that the column permutation is able to remove zeros from the diagonal of the fine level operator A, it fixes the problem of invalid sparsity patterns as described in Remark 6.2.3. However, the rigid body modes (especially the rotatory modes) still cannot be considered to be optimal in the sense of **(N1)** from Section 3.4.1. But our numerical examples in Section 6.5 show that the rigid body modes provide a sufficiently good approximation of the near null space modes, since the null space property is only locally disturbed at the contact interface (similar to Dirichlet boundaries).

6.4.2. Observer principle for aggregation strategy and level smoothers

With the column permutation one has a general way to improve and fix the matrix properties for the multigrid method. However, as one can already see from Table 6.3, the pure column permutation has also its limitations. Since the diagonal dominance of the matrix is of great importance for the multigrid preconditioner, an observer mechanism is introduced which keeps track of the local diagonal dominance of the rows of A. To keep the iterative methods working, the following techniques are used to handle the remaining possibly problematic non-diagonally dominant rows of A after the column permutation.

(S1) *Single node aggregates:* This is closely related to the Hybrid PA-AMG method described in Section 6.3.1. However, instead of putting all slave nodes $\mathcal{S}$ into single node aggregates, the information provided by the observer is used to only introduce single node aggregates for the nodes which are non-diagonally dominant after applying the column permutation. In comparison, the number of single node aggregates is significantly smaller than for the Hybrid PA-AMG method.

(S2) *Avoid diverging error modes in level smoothers:* The entries in the solution vector associated with non-diagonally dominant rows in the permuted matrix A_0^{Ψ} are preserved by introducing artificial Dirichlet-like conditions. This is important in combination with **(S1)** to prevent the level smoothers from inciting oscillatory error modes in the solution vector on the finest level.

(S3) *Avoid prolongation smoothing at contact interface:* To improve the convergence speed, one can apply transfer operator smoothing strategies as described in Section 3.5. However, since such transfer operator smoothing methods are based on Jacobi or Richardson-like methods which heavily rely on the diagonal dominance of the matrix, the transfer operator smoothing is skipped for the coarse basis functions at the contact interface. The matrix entries corresponding to the slave contact interface (see fourth row in matrix (6.3)) cannot be guaranteed to be appropriate for smoothing the transfer operator basis functions. Thus, the optimal performance of smoothed transfer operators away from the contact interface is combined with the robustness of non-smoothed transfer operators close to the slave contact interface. One could use the tangential stiffness matrix for a valid smoothing of the transfer operator basis functions. However, this information is usually not available for the solver and it would be too costly to keep the data (see also Section 6.4.3).

Remark 6.4.6 (Comparison of the Contact PA-AMG method with Hybrid PA-AMG method). The techniques **(S1)** together with **(S2)** correspond to **(H1)** and **(H2)** introduced in Section 6.3.1 for the Hybrid PA-AMG method. The main difference is that the Contact PA-AMG method uses the observer mechanism and restricts the effect of **(S1)** and **(S2)** only to the nodes which are detected to be problematic for the level smoothers and therefore must be treated carefully. The observer mechanism makes the Contact PA-AMG method more like a black-box strategy, since it automatically detects possibly problematic properties of the linear system for the inner iterative procedures. Information about the contact interface from outside is helpful but not essential.

6.4.3. Segregated aggregates

In the system matrix (6.3), the two solid bodies are algebraically melt together, such that the aggregation strategy from Section 3.3 may build aggregates that cross the contact interface (see Figure 6.3a). Note that the equations in (6.3) describe both the momentum equations and the contact constraints. While such crossing aggregates might be valid from the algebraical point of view, it is clear that the aggregation strategy does not take the physical meaning of the different equations into account. In order to represent the underlying physics, a clean distinction of the two solid bodies on all multigrid levels may be helpful for the overall solution process.

To avoid crossing aggregates, all coupling entries in A between the slave and master degrees of freedom are temporarily ignored in the input matrix for the aggregation strategy (cf. Section 3.3). That is, instead of (6.3) one uses

$$\begin{pmatrix} \mathsf{K}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{M}} & 0 & 0 & 0 \\ \mathsf{K}_{\mathcal{M}\mathcal{N}_1} & \mathsf{K}_{\mathcal{M}\mathcal{M}} + \mathsf{P}_{\mathcal{A}}^{\mathsf{T}}\mathsf{K}_{\mathcal{A}\mathcal{M}} & 0 & 0 & 0 \\ 0 & 0 & \mathsf{K}_{\mathcal{I}\mathcal{I}} & \mathsf{K}_{\mathcal{I}\mathcal{A}} & \mathsf{K}_{\mathcal{I}\mathcal{N}_2} \\ 0 & 0 & \mathsf{N}_{\mathcal{I}} & \mathsf{N}_{\mathcal{A}} & 0 \\ 0 & 0 & \mathsf{F}_{\mathcal{I}} - \mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{A}\mathcal{A}}^{-1}\mathsf{K}_{\mathcal{A}\mathcal{I}} & \mathsf{F}_{\mathcal{A}} - \mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{A}\mathcal{A}}^{-1}\mathsf{K}_{\mathcal{A}\mathcal{A}} & -\mathsf{T}_{\mathcal{A}}\mathsf{D}_{\mathcal{A}\mathcal{A}}^{-1}\mathsf{K}_{\mathcal{A}\mathcal{N}_2} \\ 0 & 0 & \mathsf{K}_{\mathcal{N}_2\mathcal{I}} & \mathsf{K}_{\mathcal{N}_2\mathcal{A}} & \mathsf{K}_{\mathcal{N}_2\mathcal{N}_2} \end{pmatrix} \tag{6.18}$$

as input for the aggregation algorithm (cf. Algorithm 3 in Section 3.3). Note that only the input for the aggregation is adapted. The level smoothers still use the original level matrices with only minor modifications resulting from the observer principle (see **(S2)** in Section 6.4.2).

Remark 6.4.7. In fact, for segregating the aggregates one should use the graph $G\big(\mathbf{K}_{\mathrm{T}}(\boldsymbol{u}^i)\big)$ from (5.44) without the linearized contact forces in $\mathbf{K}_{\mathrm{co}}(\boldsymbol{u}^i, \boldsymbol{\lambda}^i)$ instead of the graph of (6.18). However, the pure tangential stiffness matrix $\mathbf{K}_{\mathrm{T}}(\boldsymbol{u}^i)$ often is not available for the solver. Furthermore, it would probably be too expensive to keep the tangential stiffness matrix extra in memory for the aggregation.

Example 6.4.8 (Segregated aggregates)**.** Figure 6.5b shows the effect of using (6.18) as input for the aggregation strategy in contrast to the aggregation using the original full matrix (6.3). As expected, the aggregates cannot cross the contact interface. Note that in Figure 6.5b the aggregation parameters have been chosen to produce one big aggregate for the upper solid body on the slave side. Please also take notice of the four corner nodes at the slave contact interface, which obviously are not added to the big aggregate. The observer algorithm found the corner nodes to be problematic containing non-diagonally dominant rows. Therefore, they are aggregated into single-node aggregates, which are not visualized in Figure 6.5b (cf. Section 6.4.2).

When using smoothed transfer operators (cf. Section 3.5), segregating the aggregates is not sufficient to guarantee the coarse basis functions not to cross the contact interface. One also has to adapt the transfer operator smoothing process accordingly.

Remark 6.4.9 (Transfer operator smoothing). The modified matrix from (6.18) must also be used for prolongation smoothing to avoid artificial overlap of the transfer operator basis functions at the contact interface due to the transfer operator smoothing (cf. Gee et al. [71]). Note that for **(S3)** one should consider further modifications of (6.18) at the slave contact interface. Following (3.11), the smoothed prolongator is built by

$$P_{\ell+1} := \widehat{P}_{\ell+1} - \omega D^{-1}\mathrm{A}_{\ell}\widehat{P}_{\ell+1}. \tag{6.19}$$

For skipping the transfer operator smoothing at the slave contact interface, (6.19) is modified using

$$P_{\ell+1} := \widehat{P}_{\ell+1} - \omega D^{-1}\mathrm{A}_{\ell}^{\mathrm{mod}}\widehat{P}_{\ell+1}, \tag{6.20}$$

(a) Standard aggregation with crossing aggregates melting together the solid bodies on the coarse level.

(b) Segregated aggregation with single-node aggregates only for problematic nodes at the slave contact interface.

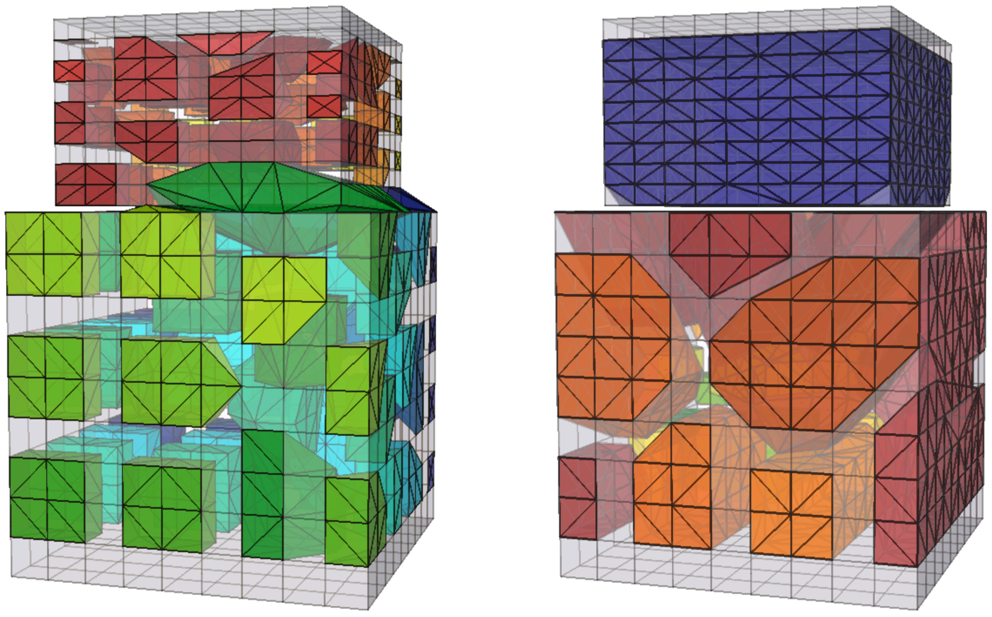


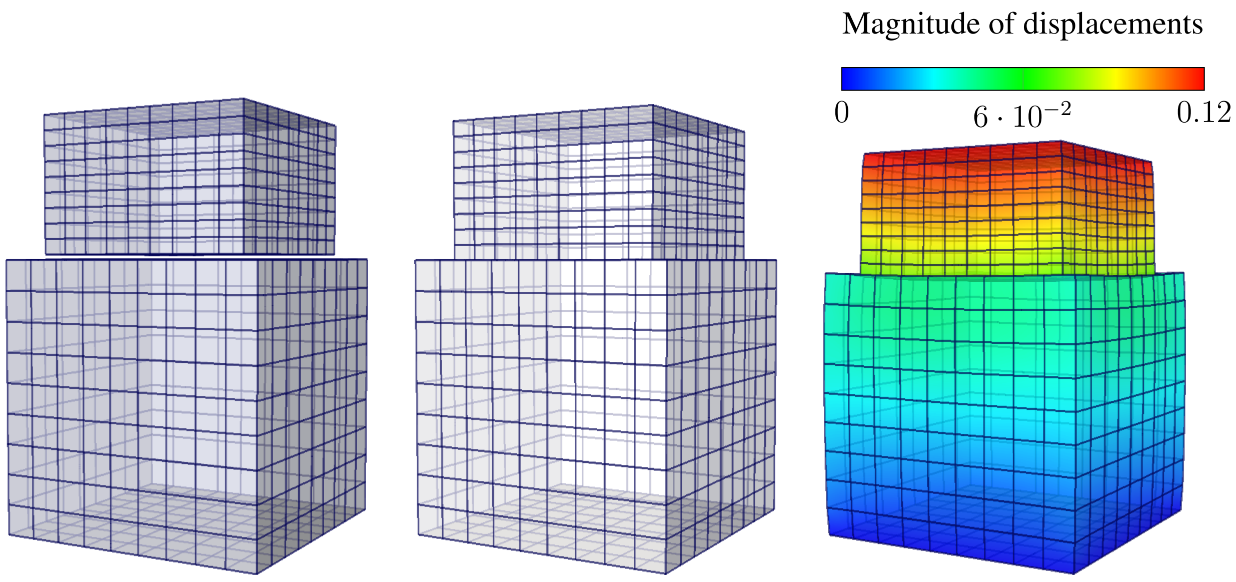
Figure 6.5.: Two solid bodies example – Comparison of standard aggregates and segregated aggregates of the Contact PA-AMG method in the reference coordinate system.

where D^{-1} is the inverse of the diagonal of the original (optionally permuted) matrix A_ℓ which is used as input for the multigrid method. One has to make sure that the input matrix A_ℓ is diagonally dominant in order to guarantee D^{-1} to be valid. The $\mathrm{A}_\ell^{\text{mod}}$ is defined as

$$\mathrm{A}_\ell^{\text{mod}} := \begin{pmatrix} \mathsf{K}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{M}} & 0 & 0 & 0 \\ \mathsf{K}_{\mathcal{M}\mathcal{N}_1} & \mathsf{K}_{\mathcal{M}\mathcal{M}} + \mathsf{P}_{\mathcal{A}}^{\mathsf{T}}\mathsf{K}_{\mathcal{A}\mathcal{M}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathsf{K}_{\mathcal{N}_2\mathcal{N}_2} \end{pmatrix}. \tag{6.21}$$

It is highly important to understand that the modified level matrices (6.18) and (6.21) are used only for aggregation and transfer operator smoothing. The original (permuted) level matrix A is still needed for building the level smoothers, such that one obtains a valid multigrid preconditioner for the original system. Also note that it would be too costly to build the modified matrices (6.18) and (6.21) explicitly and hold them in memory. Instead, one can easily incorporate the corresponding effects into the algorithms.

Remark 6.4.10 (Transfer operator sparsity pattern). Instead of adapting the input for the aggregation and prolongation smoothing using (6.18) and (6.21) one would obtain the same effect by prescribing an adapted sparsity pattern for the transfer operators (cf. Section 4.4.4). However, only very advanced transfer operator strategies allow for prescribing a transfer operator pattern (e.g., Schur complement based transfer operators as described in Section 4.4 or the methods



(a) Initial configuration $t = 0.0s$. (b) First contact $t = 0.06s$. (c) Deformed state at $t = 0.4s$.

Figure 6.6.: Two solid bodies example ($\alpha_y = \alpha_z = 0$) – Initial geometric configuration and characteristic deformation stages for first contact and after 40 time steps.

from Olson et al. [153]). For the applications in this chapter, it is easier to adapt the input rather than describe the sparsity patterns for the transfers.

6.5. Numerical examples

6.5.1. Two solid bodies example

With the two solid blocks example one can study the influence of the geometric configuration (with the relative alignment of the cartesian and the contact interface coordinate systems) to the behavior of the iterative linear solver. The geometric configuration is described in Section 6.2.1. A $10 \times 10 \times 10$ discretization is used for each solid body, such that the resulting linear system has 6000 degrees of freedom. The rotation angles α_y and α_z are varied in the range of 0 and $\frac{\pi}{2}$. Each simulation runs 50 time steps with a time step size of $0.01s$. Here, the condensed contact formulation based on dual shape functions is applied. The simulations are performed using 4 processors but no timings are given, as they cannot be reasonable for such small examples.

The nonlinear iteration stops if $\|\mathbf{r}_i^{\boldsymbol{u}}\|_e < 10^{-8}$ and $\|\Delta \boldsymbol{u}\|_e < 10^{-8}$ holds. The $\mathbf{r}_i^{\boldsymbol{u}}$ denotes the (nonlinear) residual after i Newton iterations and $\Delta \boldsymbol{u}$ denotes the displacement increment. The GMRES method is chosen as the linear solver with a 3 level Contact PA-AMG preconditioner (3 SGS (0.7) for the fine and intermediate level smoother and a direct solver on the coarsest level). For the linear solver the relative (linear) residual serves as stopping criterion which is given by

$$\left\| \frac{\mathbf{r}_i^k}{\mathbf{r}_i^0} \right\|_e < 10^{-8}, \tag{6.22}$$

where $\mathbf{r}_i^k$ denotes the (linear) residual in the iteration step $k = 0, \dots, k_{\max}$ within Newton step i. Primarily interested in the behavior of the linear solver, very strong solver tolerances are

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	21.2 (23)	21.2 (23)	21.2 (23)	21.7 (23)	21.7 (23)
	$\frac{1}{8}\pi$	20.5 (24)	20.9 (23)	21.0 (23)	21.2 (22)	21.3 (24)
	$\frac{1}{4}\pi$	21.2 (23)	21.6 (24)	21.5 (24)	22.2 (25)	23.9 (28)
	$\frac{3}{8}\pi$	18.6 (20)	19.4 (21)	25.8 (29)	35.5 (44)	23.9 (26)
	$\frac{1}{2}\pi$	18.6 (19)	18.6 (21)	27.2 (33)	21.3 (24)	23.1 (25)

Table 6.4.: Two solid bodies example – Average (maximum) number of linear GMRES iterations per nonlinear iteration (over all time steps) for different rotation angles α_y and α_z. Contact PA-AMG with 3 levels (3 SGS (0.7) on all intermediate levels, direct solver on the coarsest level, minimum aggregate size: 9) is used as preconditioner.

chosen to ensure that the results are comparable for the different multigrid methods introduced in Sections 6.2 and 6.3.

Table 6.4 gives the average number of GMRES iterations per nonlinear iteration over all 50 time steps. The number in brackets denotes the maximum number of linear iterations that occurred in the nonlinear solver during the simulation. Thanks to the column permutation (see Section 6.4.1) and the additional strategies to increase the robustness, such as the observer principle (see Section 6.4.2) and the segregated aggregates (see Section 6.4.3), the simulation succeeds for all rotation angles α_y and α_z. The number of linear iterations for the Contact PA-AMG method is also lower than for the Hybrid PA-AMG approach (cf. Table 6.2), but shows some higher variance. Nevertheless, compared to the results from Table 6.1, this is an enormous improvement with regard to the robustness of the multigrid method.

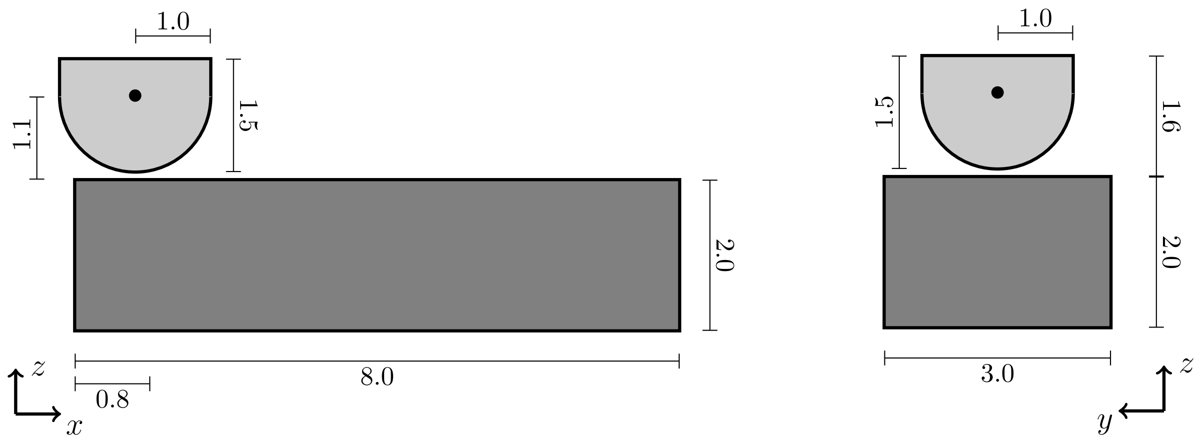
The next examples are supposed to demonstrate the robustness for some more realistic contact examples, where the contact interface is changing during the simulation. A comparison of the performance and computational costs of the Hybrid PA-AMG and Contact PA-AMG method for more realistic examples is also of interest.

6.5.2. 3D ironing example

With the 3D ironing example the behavior of the previously discussed multigrid methods from Sections 6.3 and 6.4 is further studied for structural contact problems in the condensed formulation. In this example, finite deformation contact of a half-spherical elastic die (Radius: 1 unit, Neo-Hookean model, $E_1 = 6896$, $\nu_1 = 0.32$, $\rho_1 = 1$) intruding into an elastic block (dimensions: $8 \times 2 \times 3$ units, Neo-Hookean model, $E_2 = 689.6$, $\nu_2 = 0.32$ and $\rho_2 = 1$) is analyzed (see Figure 6.7). First, the die is pressed into the block by prescribing a vertical displacement of 0.9 units in (negative) z-direction within $0.1s$ of simulation time (corresponding to 160 time steps with time step size $6.25 \cdot 10^{-4}s$). Then, it slides along the block in x-direction for further 240 time steps until a prescribed horizontal displacement of 1 unit is reached, just to demonstrate the robustness of the linear solver during the sliding phase. In this example, no friction is considered.

	PA-AMG	Hybrid PA-AMG (SGS)	Contact PA-AMG (SGS)
Transfer operator	PA-AMG	PA-AMG	PA-AMG
Column permutation	–	no	yes
Segregated aggregates	–	yes	yes
Contact interface	not used	1-to-1 (all levels)	aggregated
Level smoother	3 SGS (0.7)	3 SGS (0.7)	3 SGS (0.7)
Coarse solver	direct (KLU)	direct (KLU)	direct (KLU)

Table 6.5.: 3D ironing example – Preconditioner variants and parameters.



(a) Side view (b) Front view

Figure 6.7.: 3D ironing example – Problem setup.

Figure 6.8 shows some deformation states during the simulation and visualizes the mesh. For discretization, 468352 4-node tetrahedral elements are used yielding linear interpolation on the contact surfaces. The corresponding mesh has 84488 nodes, i.e., 253464 degrees of freedom.

In each time step, the Newton iteration is converged, if for the nonlinear residual vector $\mathbf{r}_i^{\boldsymbol{u}}$ and the displacement increment $\Delta\boldsymbol{u}$ holds $\|\mathbf{r}_i^{\boldsymbol{u}}\|_e < 10^{-6}$ and $\|\Delta\boldsymbol{u}\|_e < 10^{-6}$. For the linear solver the relative (linear) residual is checked for convergence using the criterion

$$\left\| \frac{\mathbf{r}_i^k}{\mathbf{r}_i^0} \right\|_e < 10^{-6}. \tag{6.23}$$

Again, $\mathbf{r}_i^k$ denotes the (linear) residual in the iteration step $k = 0, \dots, k_{\max}$ of the linear solver within a fixed Newton step i. The linear system is formulated using the condensed contact formulation with the Petrov–Galerkin approach for the basis functions (cf. Section 5.4.2).

Table 6.5 describes the different multigrid variants that are used as preconditioners within an outer GMRES solver. The naive multigrid approach with a standard aggregation-based AMG from Chapter 3 is compared with the special Hybrid PA-AMG method (see Section 6.3) and the more advanced Contact PA-AMG method from Section 6.4. The maximum number of multigrid

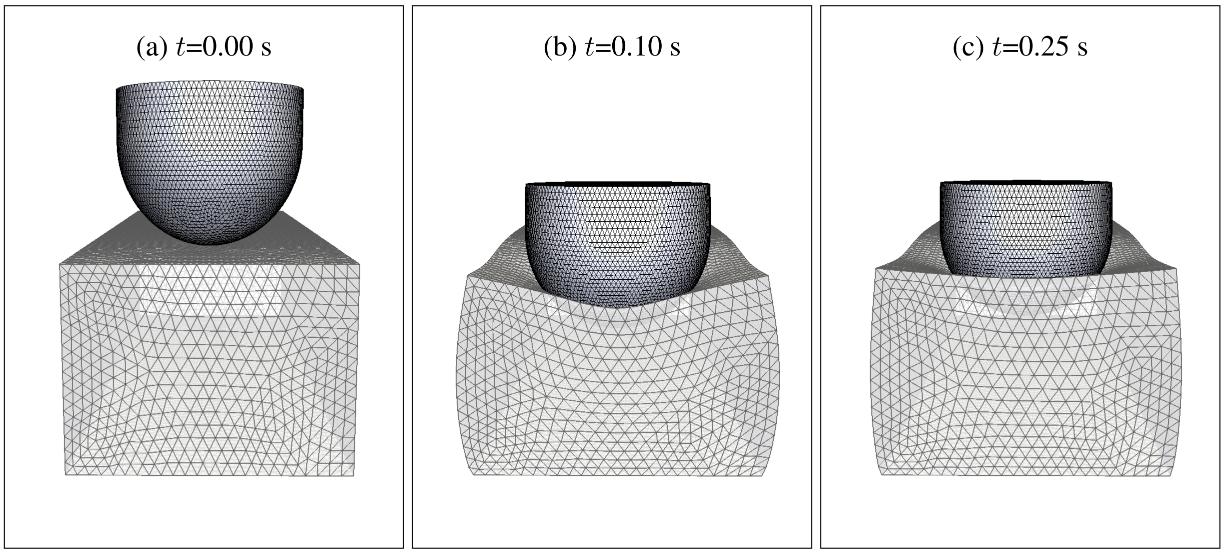


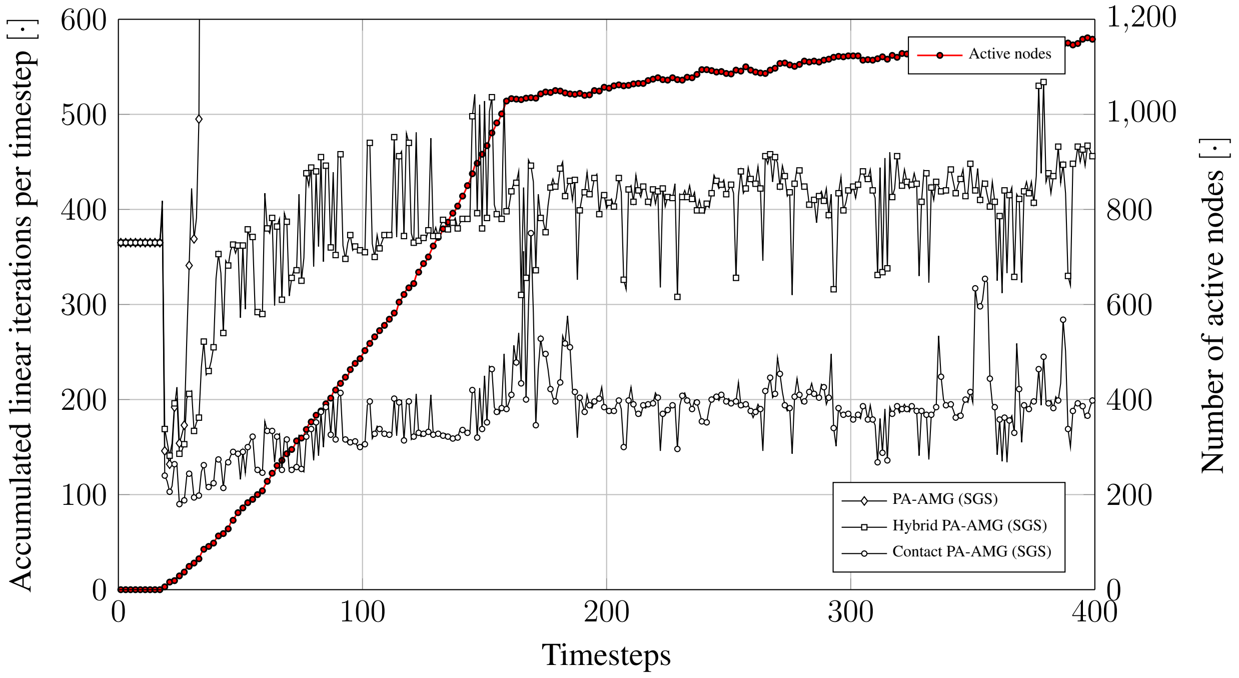
Figure 6.8.: 3D ironing example – Deformation at different times t

Method	Setup costs	Solver time	Overall solver time
PA-AMG (SGS)	–	–	–
Hybrid PA-AMG (SGS)	3242	10354	13596
Contact PA-AMG (SGS)	3581	2576	6157

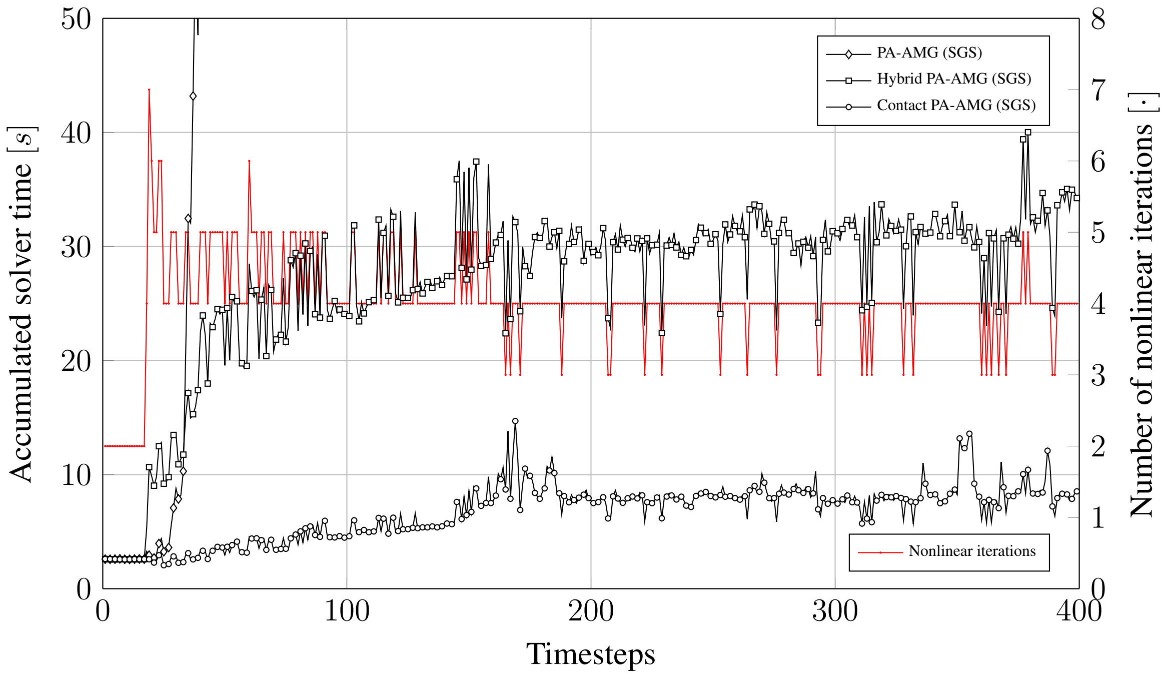
Table 6.6.: 3D ironing example – Solver timings in $[s]$.

levels is 5 and the maximum size of the problem on the coarsest level is set to 15000 degrees of freedom. Within the multigrid algorithm, an automatic rebalancing of the coarse level problem is used which reduces the number of processors involved in the coarse level in such a way that no processor can have less than 3000 degrees of freedom. The simulation runs on 32 processors on the finest level, distributed over 2 nodes with 2 Xeon (E5-2670, 2.6 GHz) Octocore CPUs each.

Figure 6.9a shows the accumulated number of linear iterations over the 400 time steps. That is, the number of linear iterations of all Newton steps within one time step are summed up. The red curve denotes the number of active nodes in each time step, after the active set has converged. One can easily see, how the number of active nodes is increasing over the first 160 time steps, when the die is intruding the block, whereas it remains rather constant during the sliding phase. As expected, the naive non-smoothed aggregation-based AMG method fails and the corresponding simulation stops after about 50 time steps, when the contact gets more dominant. The special contact multigrid methods Hybrid PA-AMG and Contact PA-AMG both are able to run the simulation for the full simulation time with 400 time steps. The Contact PA-AMG method clearly outplays the Hybrid PA-AMG approach in both terms of linear iterations and the solver time (see also Figure 6.9b). One can also find a clear correlation of the number of active nodes and the number of linear iterations (and the corresponding solver times). From Figure 6.9b one can learn that the big jumps in the number of linear iterations and the timings are related to a change in the number of Newton sweeps (cf. the red curve in Figure 6.9b).



(a) Accumulated number of linear GMRES iterations for all nonlinear iterations per timestep.



(b) Accumulated timings for the solution phase for all nonlinear iterations per timestep.

Figure 6.9.: 3D ironing example – Results for different AMG preconditioner variants.

		Transfer operators	
		Plain aggregation	Smoothed aggregation
Level smoothers	SGS	**PA-AMG (1 SGS 0.7)**	**SA-AMG (1 SGS 0.7)**
		Transfer operators: PA-AMG	Transfer operators: SA-AMG (0.4)
		Level smoother: 1 SGS (0.7)	Level smoother: 1 SGS (0.7)
	ILU	**PA-AMG (1 ILU(0)**	**SA-AMG (1 ILU(0))**
		Transfer operators: PA-AMG	Transfer operators: SA-AMG (0.4)
		Level smoother: 1 ILU(0)	Level smoother: 1 ILU(0)

Table 6.7.: Two tori impact example – Different AMG variants.

In Table 6.6 the solver timings are given for the different methods in comparison. The setup costs of the Hybrid PA-AMG and Contact PA-AMG methods are in the same range. The main difference is the number of linear iterations and the corresponding solver time. One can state that especially the Setup costs for the multigrid hierarchy are not optimal in sense of performance due to missing code optimizations.

6.5.3. Two tori impact example

Now, the different variants of the Contact PA-AMG method are applied to larger problems. Inspired by some similar analysis in Yang and Laursen [230] the problem setup of the two tori impact example with geometry and load conditions from Popp [156] is used. The initial setup of the two tori impact example is shown in Figure 6.10a. There are two thin-walled tori with a Neo-Hookean material model ($E = 2250$, $\nu = 0.3$, $\rho_0 = 0.1$) with a major and minor radius of 76 and 24 units and a wall thickness of 4.5 units. The right torus in Figure 6.10 lies in the xy-plane and is accelerated by a body force towards the left torus, which is rotated around the y-axis by 45 degrees. The simulation runs 200 time steps with a time step size of $0.05s$ using a generalized-α time integration scheme. Different characteristic stages of deformation for the two tori example are shown in Figure 6.10. In each time step, the nonlinear problem is solved using a semi-smooth Newton method. The convergence check for the Newton method is chosen to be

$$\left\| \frac{\mathbf{r}_i^{\boldsymbol{u}}}{\mathbf{r}_0^{\boldsymbol{u}}} \right\|_e < 10^{-6} \ \wedge \ \|\Delta \boldsymbol{u}\|_e < 10^{-7}, \tag{6.24}$$

where $\mathbf{r}_i^{\boldsymbol{u}}$ denotes the (nonlinear) residual after i nonlinear iterations. The $\Delta \boldsymbol{u}$ stands for the displacement increment.

The finite element mesh consists of 284672 first-order hexahedral elements with 350208 nodes. The corresponding fine level problem has 1050624 degrees of freedom. For solving the linear systems within the nonlinear Newton iteration a preconditioned GMRES method is applied. To check linear convergence the linear residual is tested for

$$\left\| \frac{\mathbf{r}_i^{k}}{\mathbf{r}_i^{0}} \right\|_e < 10^{-8}. \tag{6.25}$$

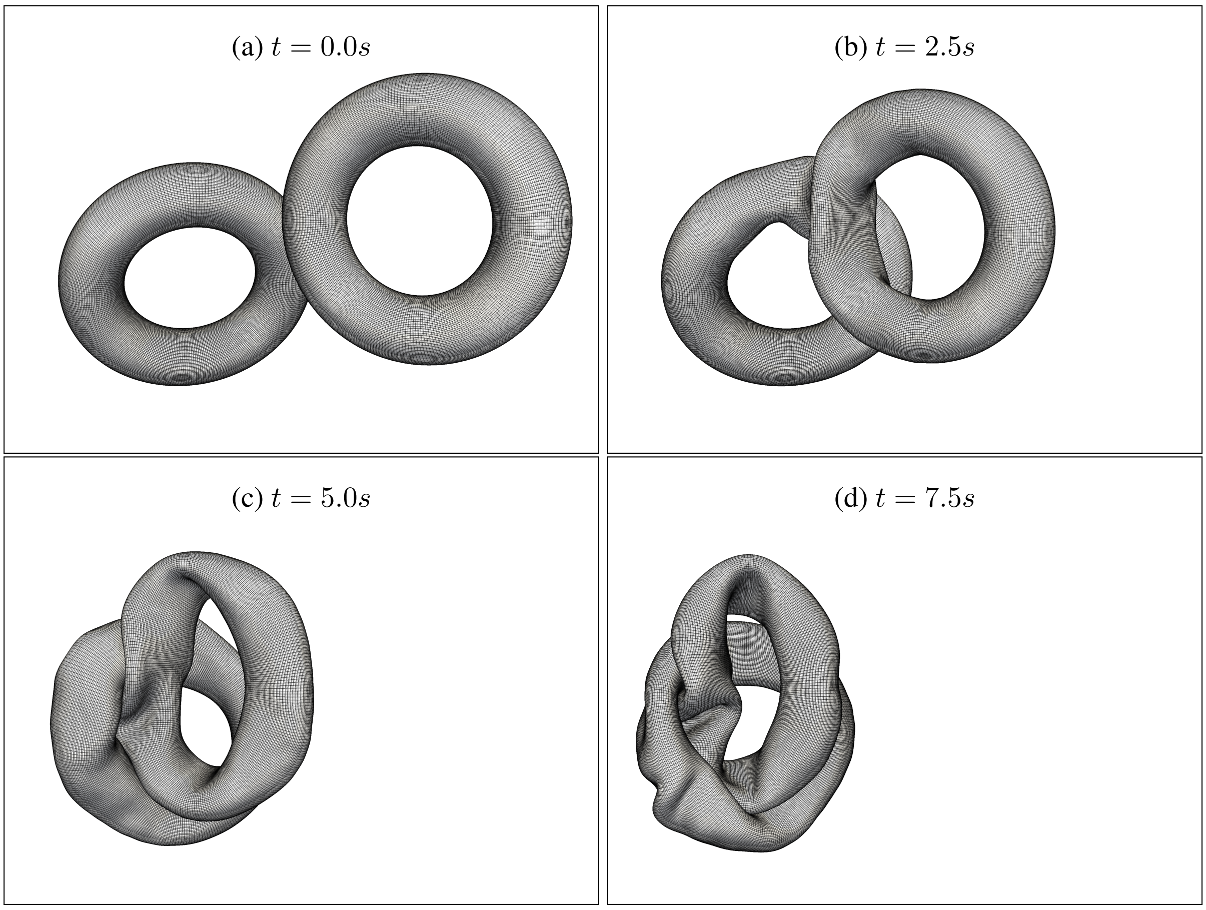


Figure 6.10.: Two tori impact example – Characteristic stages of deformation.

Here, the $\mathbf{r}_i^k$ denotes the linear residual in the k-th linear iteration in Newton step i. The linear system is formulated using the condensed contact formulation with the Petrov–Galerkin approach for the basis functions (cf. Section 5.4.2). With the knowledge from the previous numerical experiments, only different variants of the Contact PA-AMG multigrid preconditioner from Section 6.4 are compared. In our notation, PA-AMG and SA-AMG just denotes the corresponding transfer operator strategies within the Contact PA-AMG method. Different level smoothers and the effect of smooth transfer operators are subject of a more detailed study. In particular, ILU level smoothers, which are more expensive in the setup, are studied in comparison with symmetric Gauss–Seidel level smoothers with nearly no setup costs. Furthermore, the potential of reducing the linear iterations and the solver timings by using smoothed transfer operators is of high interest. Therefore, a (classical) smoothed aggregation transfer operator (cf. Section 3.5.1 and Remark 6.4.9) is compared against non-smoothed transfer operators (cf. Section 3.4.1). Note that it is very hard to find a proper transfer operator smoothing parameter, since the linear systems arising from structural contact problems in a condensed formulation are structurally non-symmetric (cf. Section 6.1.1). Therefore, a smoothed aggregation transfer operator is used with some careful damping parameter together with the Petrov–Galerkin approach for the restriction operator (cf. Section 4.2.2) to reflect the non-symmetry of the underlying operator in the setup phase of the multigrid method. Table 6.7 gives an overview of the variants of the tested preconditioning methods. The remaining multigrid parameters are fixed for all variants. The maximum coarse level size is chosen to be 5000 degrees of freedom. With the minimum size for an aggregate to be 18 nodes this corresponds to a 3 level multigrid method. The coarse solver is chosen to be a direct solver (KLU from the Amesos package in Trilinos, cf. Sala et al. [172]). The simulations are performed in parallel on 64 cores (spread over 8 computational nodes connected by an Infiniband network using 1 Intel Xeon (E5-2670, 2.6 GHz) Octocore CPU and up to 32 GB RAM on each node).

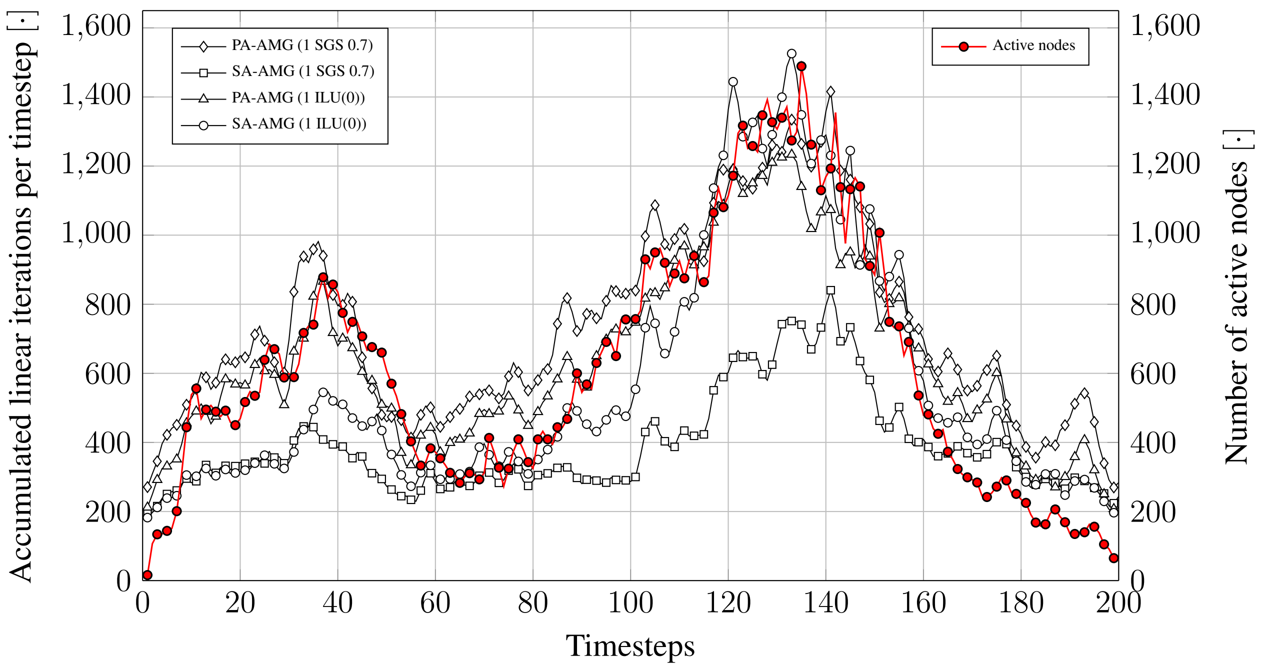
Figure 6.11a shows the accumulated number of linear GMRES iterations for all nonlinear Newton sweeps in each timestep. The red line in Figure 6.11a denotes the number of contact nodes in the corresponding time step, which have been found to be active by the active set strategy. Obviously, the number of linear iterations follows the number of active nodes in contact. Compared to the variants with non-smoothed transfer operators (PA-AMG), the usage of smoothed aggregation transfer operators (SA-AMG) with some appropriate smoothing parameters leads to a notable reduction of linear iterations. Assuming that the solution phase dominates the timings over the setup phase, a smaller number of linear iterations is highly desirable, since the timings for the solution phase are usually correlated with the number of linear iterations.

In Figure 6.11b the timings of the solution phase of all linear systems within one time step are summed up and plotted over the time steps. As expected, there is a clear correlation between the number of iterations and the timings. The red solid line in Figure 6.11b describes the number of nonlinear sweeps per time step for the PA-AMG (1 SGS 0.7) variant. It can be assumed to be representative for all other variants, too.

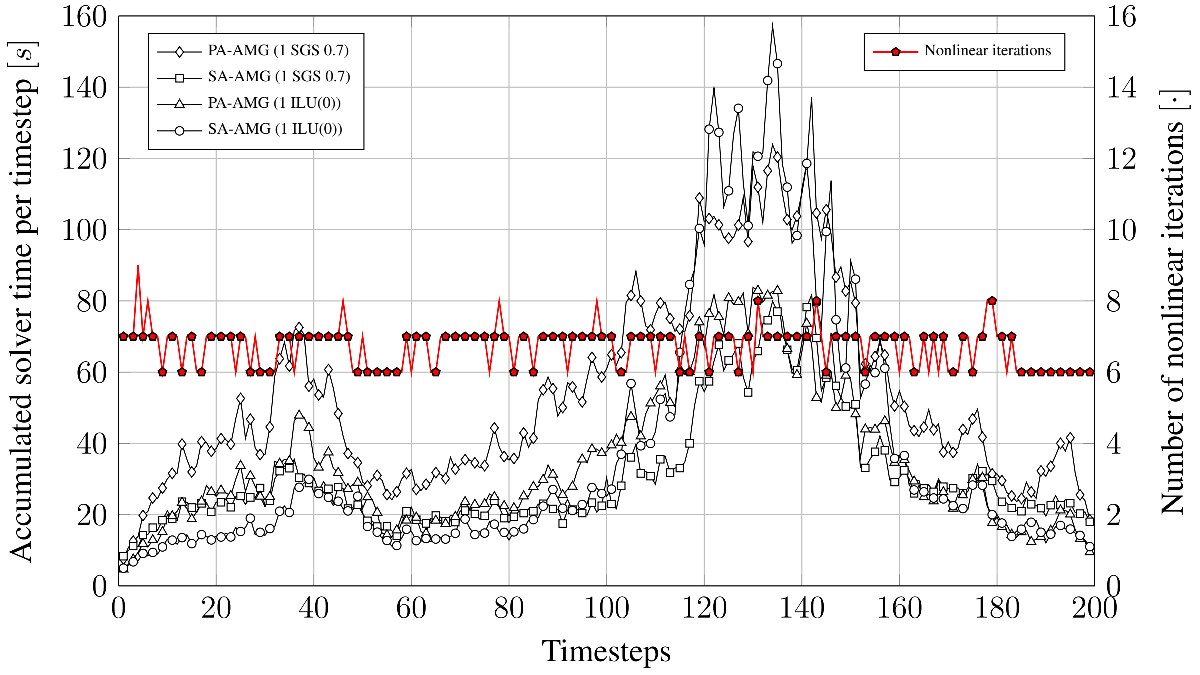
Figure 6.12 sums up the solver timings from Figure 6.11b over all time steps for the different methods and visualizes the time savings against the reference variant PA-AMG (1 SGS 0.7). For this example the SA-AMG (1 SGS 0.7) performs best for the full simulation and outplays the variants based on the ILU smoother. Interestingly, the performance of the SA-AMG (1 ILU(0)) variant drops for a high number of contact nodes (in time steps 120-140). Comparing the timings of the best method (SA-AMG (1 SGS 0.7)) and the reference method (PA-AMG (1 SGS 0.7)) one finds a time saving of approximately 80 minutes for the 200 time steps in the simulation.

However, to gain a full picture of the solver performance one should not neglect the setup costs, since in practice the only interesting number is the wall clock time both for setup and iteration. Table 6.8 lists the setup costs and the overall timings (setup and solution phase) for the different preconditioning variants. One can state that there is plenty space for code optimizations, especially for the setup costs. This means that the absolute timings are probably too high, but still can be used for direct comparisons. Nevertheless, these numbers reveal that – depending on the chosen level smoothers – the setup costs may play a significant role in the overall timings. Obviously, the setup for the ILU smoother nearly doubles the setup costs and therefore, the ILU variants are far away from being interesting for this example.

In this example the focus is on the behavior of the linear solver. Therefore, in (6.25) a fixed and quite strong tolerance is chosen for the linear solver. Beside of technical improvements in the implementation, there are many more ways to significantly reduce the computational costs related to the linear solver. First, one should try to reuse the multigrid preconditioner for more than one Newton sweep (within a time step) to save some setup costs. However, one has to be aware that this is only possible, if the active set has not changed between two Newton steps. Otherwise the methods described in Section 6.4.2 will not work properly. Additionally, in combination with the semi-smooth Newton method one could develop strategies to reasonably adapt



(a) Accumulated number of linear GMRES iterations for all nonlinear iterations per timestep.



(b) Accumulated timings for the solution phase for all nonlinear iterations per timestep.

Figure 6.11.: Two tori impact example – Results for different AMG preconditioner variants.

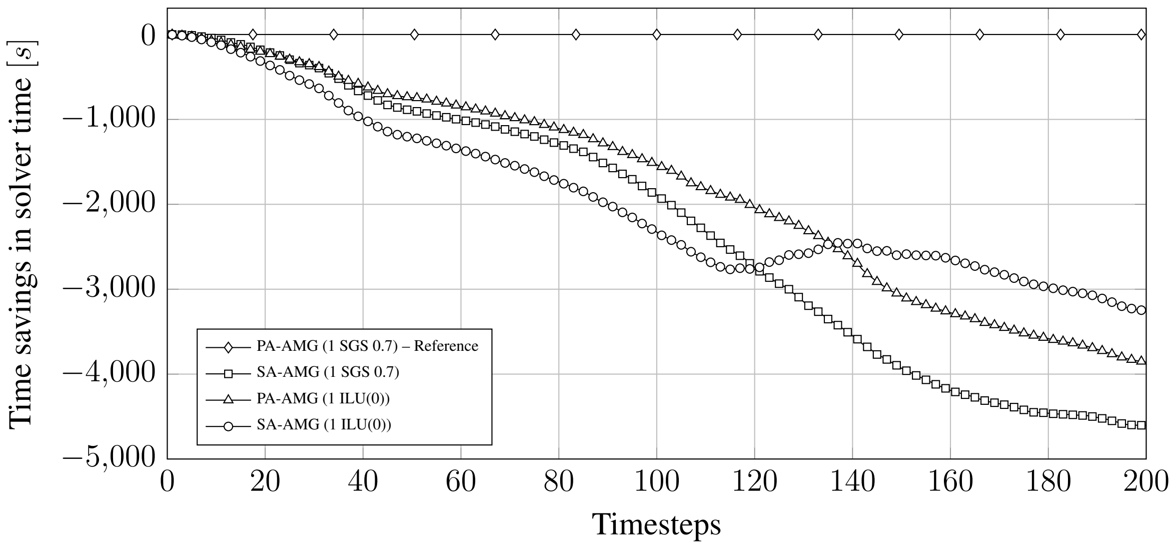


Figure 6.12.: Two tori impact example – Improvement of solver timings against reference method PA-AMG (1 SGS 0.7).

Method	Setup costs	Solver time	Overall solver time
PA-AMG (1 SGS 0.7)	9210	10809	20019
SA-AMG (1 SGS 0.7)	10260	6208	16468
PA-AMG (1 ILU(0))	19750	6943	26693
SA-AMG (1 ILU(0))	21820	7548	29368

Table 6.8.: Two tori impact example – Exemplary solver timings in $[s]$ of the different preconditioning variants from Table 6.7 for the full simulation (200 time steps).

the linear solver tolerance depending on the iterative solution of the outer nonlinear solver. This way, one can further reduce the number of linear iterations and the corresponding solver time.

6.5.4. Sliding example

This example demonstrates the effect of further common strategies to reduce the computational time for the linear solver. Inspired by an example from Hüeber [99], a frictionless sliding example is considered as introduced in Popp et al. [157]. Figure 6.13 describes the exact geometric configuration and shows the fine level mesh. The example consists of two half-cylindrical bodies. For both solids a Neo-Hookean material model is used with $E = 120,\ \nu = 0.3,\ \rho_0 = 0.3$ for the upper body and $E = 60,\ \nu = 0.25,\ \rho_0 = 0.5$ for the lower body.

The following Dirichlet conditions apply: The lower right surface (A) is completely fixed by homogeneous Dirichlet boundary conditions. The upper right surface (B) is fixed in the x- and y-directions. For the z-coordinate the movement by $-0.1\frac{t}{T}$ is prescribed with $T = 0.1s$ denoting the overall simulation time. The lower left surface (C) is fixed in the yz-plane. Finally, the upper left surface (D) is fixed in the y-direction with a prescribed displacement $-0.12\frac{t}{T}$ in z-direction. Both surfaces (C) and (D) can move freely in x-direction. The prescribed displacements are applied in 160 time steps of size $6.25 \cdot 10^{-4}s$. A discretization with 254016 8-node hexahedral elements is used, which corresponds to 275462 nodes and 826386 degrees of freedom. The con-

Multigrid settings		Level smoothers	
No. multigrid levels:	4	$\ell = 0:$	1 SGS (0.7)
Max. coarse level size:	5000	$\ell = 1:$	3 SGS (0.7)
Transfer operators:	PA-AMG	$\ell = 2:$	5 SGS (0.7)
Minimum aggregate size:	12	$\ell = 3:$	Direct solver

Table 6.9.: Sliding example – Multigrid parameters for the Contact PA-AMG method.

	Timestep 15			Timestep 80			Timestep 160		
ℓ	#act	#nodes	ratio	#act	#nodes	ratio	#act	#nodes	ratio
0	43	275462	0.02%	354	275462	0.13%	985	275462	0.36%
1	43	16432	0.26%	354	16295	2.17%	985	16064	6.13%
2	43	1144	3.76%	354	1406	25.18%	985	1913	51.49%
3	43	283	15.19%	354	575	61.57%	985	1200	82.08%

Table 6.10.: Sliding example – Exemplary multigrid hierarchies for different time steps. The '#nodes' column denotes the number of fine level nodes (or aggregates) for the different multigrid levels ℓ. '#act' is the number of (active) nodes, which is detected to be problematic and transferred to the coarsest level. 'ratio' denotes the fraction '#act' over '#nodes'.

tact problem is formulated in condensed notation based on the Petrov–Galerkin approach. The simulation runs on 6 Intel Xeon (E5-2670, 2.6 GHz) Octocore CPUs with altogether 48 cores on the finest level.

Solution strategy & Stopping criteria

In each time step a nonlinear problem is solved using a semi-smooth Newton method (cf. Section 5.5). The convergence check in the Newton method is given by

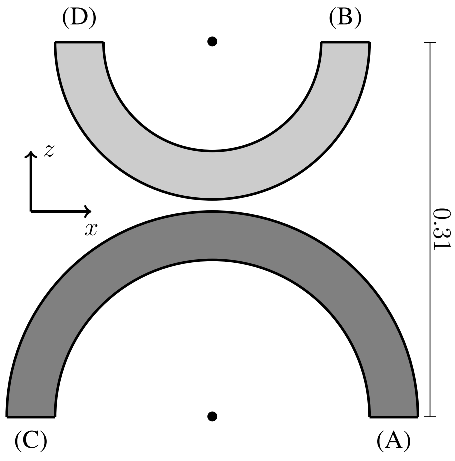
$$\|\mathbf{r}_i^{\boldsymbol{u}}\|_e < 10^{-8} \ \wedge \ \|\Delta \boldsymbol{u}\|_e < 10^{-8}, \tag{6.26}$$

where $\mathbf{r}_i^{\boldsymbol{u}}$ denotes the (nonlinear) residual after i nonlinear iterations. The $\Delta \boldsymbol{u}$ stands for the displacement increment.

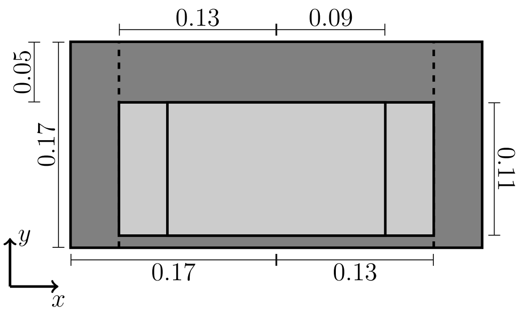
Again, one applies a preconditioned GMRES method for solving the linear systems within the nonlinear Newton iteration. As preconditioner, a Contact PA-AMG method with the parameters given in Table 6.9 is used. Note that the number of level smoother sweeps is increasing on the coarser levels. With the minimum size of 12 nodes for the aggregates one obtains an average coarsening rate larger than 12. Table 6.10 gives exemplary sizes for the multigrid hierarchies in different time steps. Obviously, the coarse level problems are quite small and applying more smoothing sweeps is very cheap on the coarser levels. The finer levels in the multigrid method may have some additional benefit from it.

Dirichlet BCs

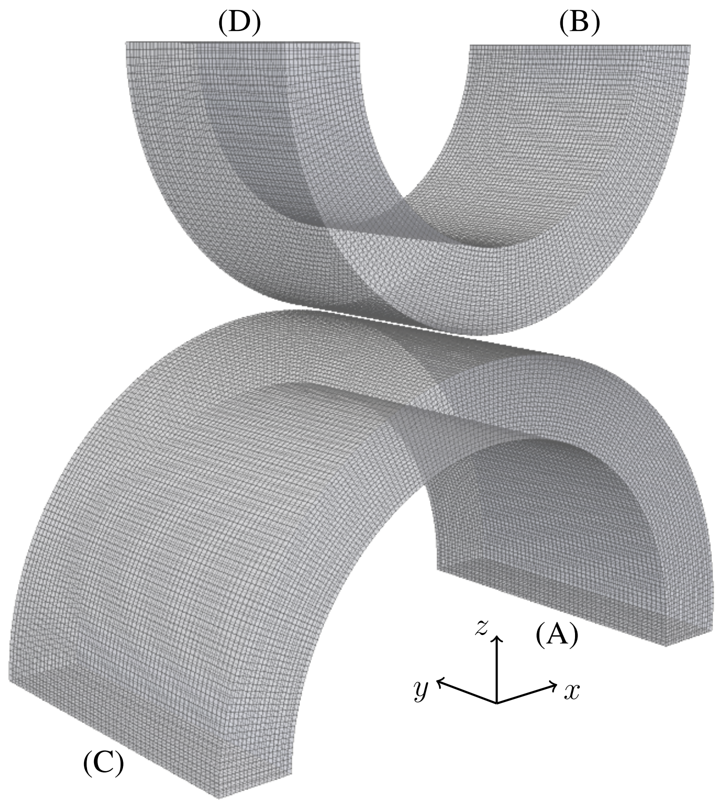
Surface	x	y	z
(A)	0.0	0.00	0.00
(B)	0.0	0.00	$-0.10\,\frac{t}{T}$
(C)	–	0.00	0.00
(D)	–	0.00	$-0.12\,\frac{t}{T}$



(a) Front view



(b) Top view



(c) Fine level mesh

Figure 6.13.: Sliding example – Problem setup and mesh

Method	# Setup calls	Setup time	Solver time	Overall time
PA-AMG (fixed tol., no reuse)	879	6712	18760	25472
PA-AMG (fixed tol., reuse)	520	4404	17290	21694
PA-AMG (adapted tol., no reuse)	891	6889	14940	21829
PA-AMG (adapted tol., reuse)	520	4444	14700	19144

Table 6.11.: Sliding example – Exemplary number of setup calls and timings in $[s]$ for the different preconditioning variants over all 160 time steps.

To improve the overall performance, our special focus is put on the comparison of different stopping criteria for the linear solver based on the relative residual.

Fixed relative tolerance: The convergence check is based on a fixed tolerance ε_{f} for the linear relative residual as given by

$$\left\|\frac{\mathbf{r}_i^k}{\mathbf{r}_i^0}\right\|_e < 10^{-6} =: \varepsilon_{\mathrm{f}}. \tag{6.27}$$

Again, the $\mathbf{r}_i^k$ denotes the linear residual after applying k iterations with the iterative linear solver in Newton step i. This is a very simple convergence check which has already been used in the previous examples, but completely ignores the progress of the outer Newton-like iteration. As a consequence one spends too much time in reducing the linear relative residual without having significant benefit from it in the nonlinear iteration.

Adapted relative tolerance: As an alternative one can use the following very simple strategy to adapt the linear tolerance ε_{a} for the linear solver. Instead of (6.27) one checks

$$\left\|\frac{\mathbf{r}_i^k}{\mathbf{r}_i^0}\right\|_e < \varepsilon_{\mathrm{a}}, \tag{6.28}$$

where ε_a is defined by

$$\varepsilon_a = \begin{cases} 10^{-6} & \text{if } i = 0, \\ 0.001 \cdot \frac{10^{-5}}{\left\|\mathbf{r}_i^{\boldsymbol{u}}\right\|_e} & \text{if } i > 0 \,\wedge\, \varepsilon_{\mathrm{a}} < \frac{10^{-5}}{\left\|\mathbf{r}_i^{\boldsymbol{u}}\right\|_e} \end{cases} \tag{6.29}$$

with $\mathbf{r}_i^{\boldsymbol{u}}$ denoting the nonlinear residual in Newton step i. The idea is that the linear tolerance ε_{a} is loosened, once the current nonlinear residual $\mathbf{r}_i^{\boldsymbol{u}}$ is small in some norm to save some linear iterations. The relative tolerance ε_{a} is updated to be at least 3 orders of magnitude lower than the current nonlinear residual. Note that this is not a true adaptivity in the field of inexact Newton methods, but just a simple heuristic rule to adapt the linear solver tolerances to save some computational time.

Multigrid hierarchy reuse strategy

To further reduce the computational costs, one can try to reuse the multigrid preconditioners as far as possible. That is, one builds the multigrid hierarchy only if the active set of contact nodes has changed compared to the previous solver call. This is necessary to update the set of possibly problematic nodes at the contact interface, which need some special treatment in the Contact PA-AMG method. Once the active set is converged, the existing multigrid preconditioner is reused for the remaining linear systems in the current time step.

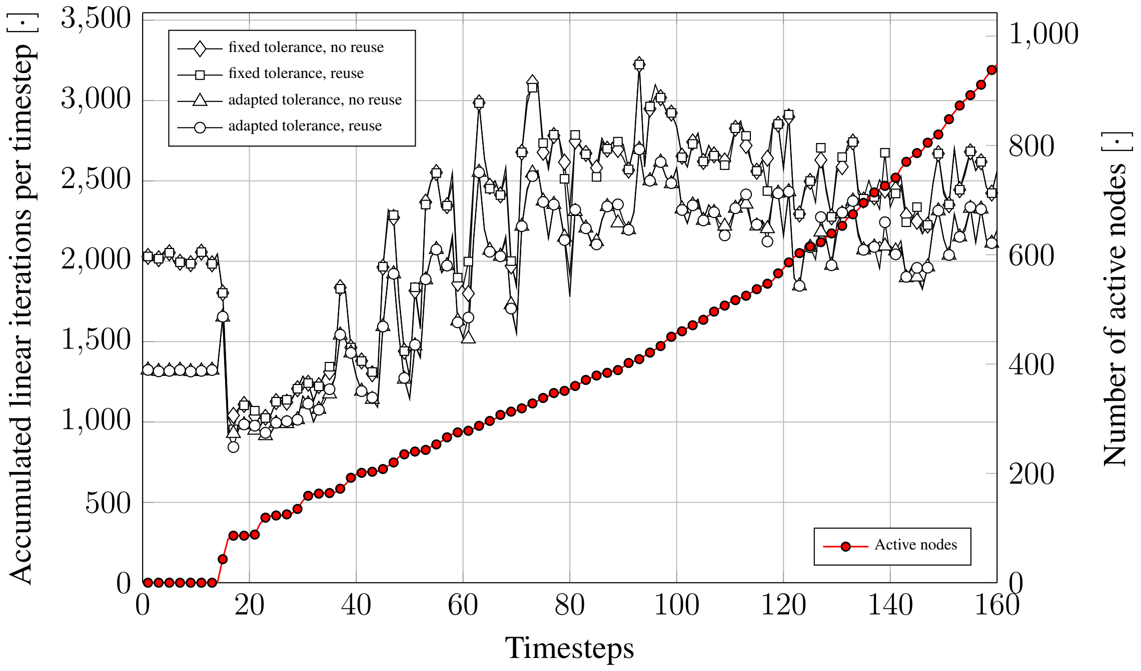
Results

Table 6.11 gives exemplary timings measured for the different preconditioning strategies using the previously described convergence criteria and reusing techniques. As one can clearly see, reusing the existing preconditioners can significantly reduce the number of multigrid setup calls and the corresponding setup costs. Adapting the tolerance for the linear solver depending on the nonlinear residual has some notable effect on the solution time, but may lead to a slightly higher number of necessary setup calls. With both reusing the multigrid hierarchy and wisely adapting the solver tolerances, one can save approximately 20% of the overall solver time in this example.

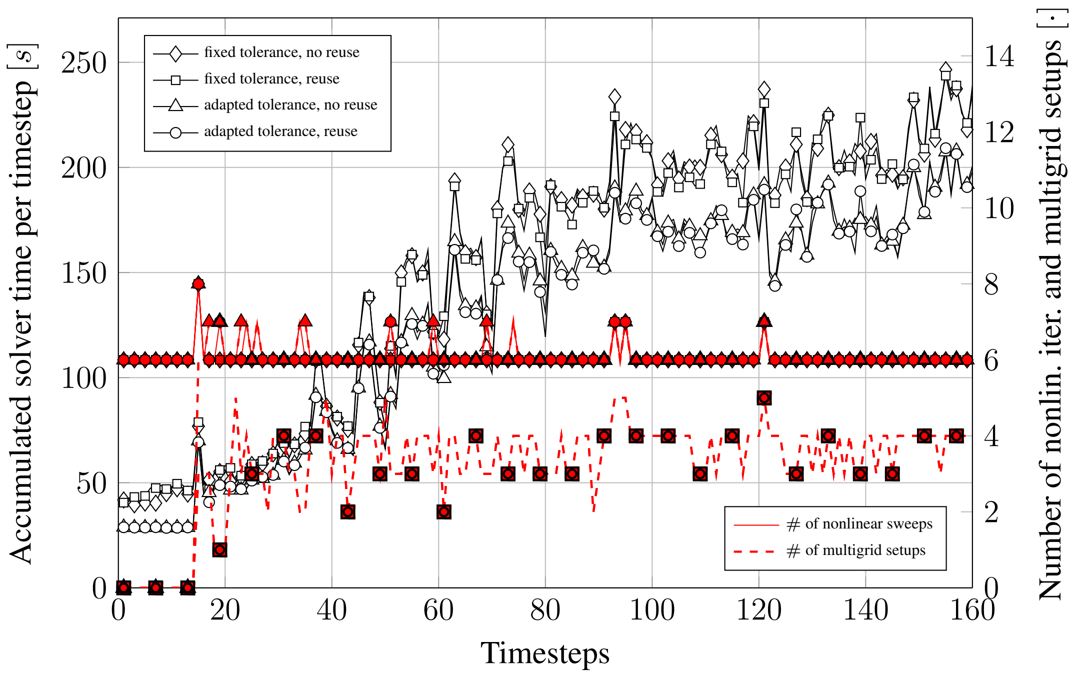
In Figure 6.14 the number of accumulated linear iterations and the solver timings is plotted over the time steps. Again one finds the number of linear iterations roughly following the number of active nodes at the contact interface, at least for the first 80 time steps before the number of linear iterations remains rather constant in this example.

With Figure 6.14a it becomes clear that reusing the preconditioner has no negative effect on the number of linear iterations, but allows to save a significant amount of time for the setup (cf. Table 6.11). Figure 6.14b shows the solver timings and the number of nonlinear sweeps per time step together with the number of multigrid setup calls. If the multigrid hierarchy is not reused, the number of setup calls for the multigrid preconditioner coincides with the number of nonlinear sweeps. Otherwise, the number of multigrid setup calls corresponds to the number of changes in the active set of contact nodes in the semi-smooth Newton method. In Figure 6.14a one finds significant local fluctuations in the number of linear iterations which seem to be related with the number of changes in the active set when comparing with the results from Figure 6.14b.

Figure 6.15 visualizes the aggregates on level $\ell = 2$ for different time steps with a close-up view of the corresponding contact zone. Each aggregate has a different color which represents the aggregate id. The small colored balls denote nodes on level $\ell = 2$ which are either part of the aggregate of the same color or represent single node aggregates in the contact zone. These correspond to the nodes which have been found to carry problematic information for the multigrid level smoothers and therefore are transferred to the coarsest level. It is quite obvious how the number of single node aggregates increases with the number of the active nodes at the contact interface. The single node aggregates away from the contact interface are resulting from the uncoupled aggregation algorithm in Section 3.3, when there are not enough nodes left on the current processor to build further aggregates.



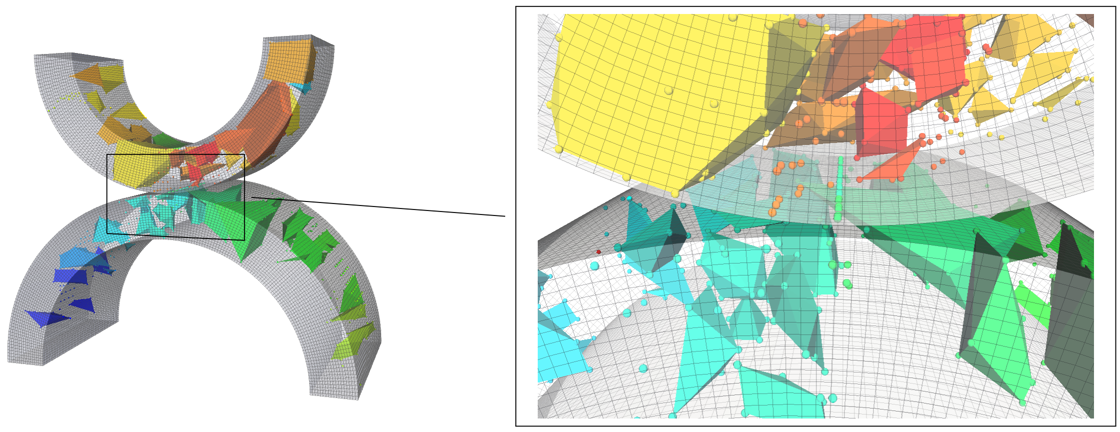
(a) Accumulated number of linear GMRES iterations for all nonlinear iterations per timestep.



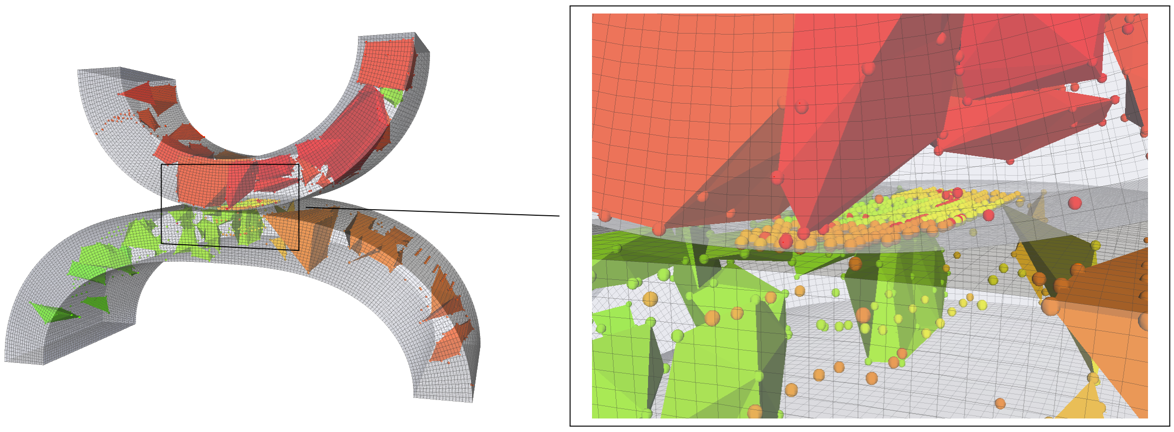
(b) Accumulated timings for the solution phase for all nonlinear iterations per timestep.

Figure 6.14.: Sliding example – Results for different linear stopping criteria and reuse strategies.

(a) Time step 15 (first contact)



(b) Time step 80



(c) Time step 160

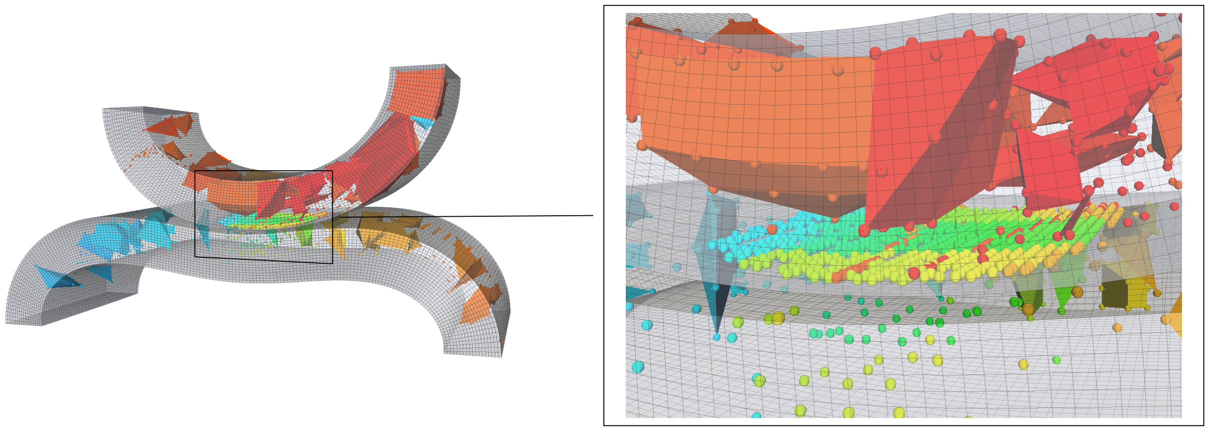


Figure 6.15.: Sliding example – Aggregates on multigrid level $\ell = 2$ representing the nodes on the coarsest level $\ell = 3$ for different time steps with a close-up view of the single node aggregates at the (active) contact interface for handling the possibly problematic matrix information for the level smoothers.

CHAPTER 7

Algebraic multigrid for contact problems in saddle point formulation

In this chapter, a full multigrid strategy is developed for structural contact problems in saddle point formulation. The idea is to deal with the original linear systems with saddle point structure as introduced in Section 5.5.2. This way, the computational time that is needed for condensing and transforming the linear systems to make them work with the multigrid methods from Chapter 6, can already be used for the solving process instead.

The idea of extending multigrid methods to saddle point systems is not new and can be found, e.g., in the context of Stokes and Oseen equations in literature (cf. Braess and Sarazin [29], Janka [101]). The main contribution of this work is the development of an interface aggregation strategy for generating Lagrange multiplier aggregates that are required for contact problems. The proposed method is simpler to implement, computationally less expensive than the ideas from Adams [1], and – in the author's opinion – the presented approach is more intuitive for coupling structural equations with contact constraints at a contact interface. Our interface aggregation strategy perfectly fits into our multigrid framework and can easily be combined with segregated transfer operators which allow to preserve the saddle point structure on the coarse levels. Additionally, different standard block smoothers for indefinite systems are introduced with some discussion on their properties for usage with contact problems.

In the numerical examples, the behavior of different multigrid block smoothers from literature is studied for contact problems in saddle point formulation. The results are compared with alternative SIMPLE based block preconditioners. Finally, the saddle point multigrid preconditioners are shown to solve large structural contact problems with more than one million degrees of freedom. The resulting multigrid preconditioners turn out to be very robust and often perform better than alternative block preconditioners.

7.1. Structural contact problems in saddle point formulation

7.1.1. Algebraic contact problem in saddle point formulation

The intention of this section is to develop a robust AMG methods for linear systems based on the saddle point formulation of structural contact problems. The idea is to define multigrid methods that make use of the saddle point block structure

$$\left(\begin{array}{ccccc|cc} \mathsf{K}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{M}} & 0 & 0 & 0 & 0 & 0 \\ \mathsf{K}_{\mathcal{M}\mathcal{N}_1} & \mathsf{K}_{\mathcal{M}\mathcal{M}} & \mathsf{K}_{\mathcal{M}\mathcal{I}} & \mathsf{K}_{\mathcal{M}\mathcal{A}} & 0 & -\mathsf{M}_{\mathcal{I}}^{\mathsf{T}} & -\mathsf{M}_{\mathcal{A}}^{\mathsf{T}} \\ 0 & \mathsf{K}_{\mathcal{I}\mathcal{M}} & \mathsf{K}_{\mathcal{I}\mathcal{I}} & \mathsf{K}_{\mathcal{I}\mathcal{A}} & \mathsf{K}_{\mathcal{I}\mathcal{N}_2} & \mathsf{D}_{\mathcal{I}\mathcal{I}}^{\mathsf{T}} & \mathsf{D}_{\mathcal{I}\mathcal{A}}^{\mathsf{T}} \\ 0 & \mathsf{K}_{\mathcal{A}\mathcal{M}} & \mathsf{K}_{\mathcal{A}\mathcal{I}} & \mathsf{K}_{\mathcal{A}\mathcal{A}} & \mathsf{K}_{\mathcal{A}\mathcal{N}_2} & \mathsf{D}_{\mathcal{A}\mathcal{I}}^{\mathsf{T}} & \mathsf{D}_{\mathcal{A}\mathcal{A}}^{\mathsf{T}} \\ 0 & 0 & \mathsf{K}_{\mathcal{N}_2\mathcal{I}} & \mathsf{K}_{\mathcal{N}_2\mathcal{A}} & \mathsf{K}_{\mathcal{N}_2\mathcal{N}_2} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & \mathsf{I} & 0 \\ 0 & \mathsf{N}_{\mathcal{M}} & \mathsf{N}_{\mathcal{I}} & \mathsf{N}_{\mathcal{A}} & 0 & 0 & 0 \\ 0 & 0 & \mathsf{F}_{\mathcal{I}} & \mathsf{F}_{\mathcal{A}} & 0 & 0 & \mathsf{T}_{\mathcal{A}} \end{array}\right) \left[\begin{array}{c} \Delta \boldsymbol{u}_{\mathcal{N}_1} \\ \Delta \boldsymbol{u}_{\mathcal{M}} \\ \Delta \boldsymbol{u}_{\mathcal{I}} \\ \Delta \boldsymbol{u}_{\mathcal{A}} \\ \Delta \boldsymbol{u}_{\mathcal{N}_2} \\ \hline \Delta \boldsymbol{\lambda}_{\mathcal{I}} \\ \Delta \boldsymbol{\lambda}_{\mathcal{A}} \end{array}\right] = - \left[\begin{array}{c} \mathbf{r}^{\boldsymbol{u}}_{\mathcal{N}_1} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{M}} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{I}} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{A}} \\ \mathbf{r}^{\boldsymbol{u}}_{\mathcal{N}_2} \\ \hline \mathbf{r}^{\boldsymbol{\lambda}}_{\mathcal{I}} \\ \mathbf{r}^{\boldsymbol{\lambda},\mathrm{n}}_{\mathcal{A}} \\ \mathbf{r}^{\boldsymbol{\lambda},\tau}_{\mathcal{A}} \end{array}\right], \tag{7.1}$$

as it has already been introduced in (5.46) in Chapter 5.5.2. Note that in contrast to (5.46) now a fully incremental formulation is used both for the displacement variables and the Lagrange multipliers. For reasons of simplicity we make use of the same notation as in Sections 5.5.2 and 6.1. Equation (7.1) perfectly fits into the definition of a generalized saddle point problem with the typical block structure

$$\begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix} \begin{bmatrix} \Delta \boldsymbol{u} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}^{\boldsymbol{u}} \\ \mathbf{r}^{\boldsymbol{\lambda}} \end{bmatrix}. \tag{7.2}$$

7.1.2. Solution strategies for saddle point problems

Since saddle point systems often arise in the modeling of mathematical and physical processes (i.e., incompressible Navier–Stokes equations, constrained optimization problems), there is a great interest in efficient solution methods for such systems with quite an abundant amount of literature on the theory of saddle point problems. The interested reader might refer to [23] as a starting point to dive into the theory of saddle point problems with an elaborate classification and a general overview of solving strategies for saddle point problems.

Linear systems with saddle point structure make special preconditioning and solving strategies necessary, which are aware of the special block structure of the underlying system matrix. Therefore, from the engineering point of view, saddle point problems often are considered to be complicated to solve. Aside from such “purely technical issues”, the saddle point formulation has the advantage that there is a clear distinction of the primary (i.e., displacement) variables and the (mathematical) Lagrange multipliers, such that the underlying physics is reflected in the block structure of the resulting system matrix (7.1). In general, an exact knowledge about the nature of the underlying equations is very important for the design of efficient solving strategies. Even if systems share the same type of block structure in their algebraic representation (such as the saddle point structure), the optimal choice of solving strategies still depends on the equations with the corresponding mathematical properties of the resulting linear systems. These are governed by the modeling and the discretization techniques. In context of saddle point problems the so-called *inf–sup condition* or *LBB condition*, named after Ladyzhenskaya, Babuška and Brezzi (e.g., Babuška [10], Ladyzhenskaya and Silverman [113]), plays an important role for the stability and convergence of the method. For a nice introduction to the role of the inf–sup condition in finite element methods the reader is referred to Bathe [15].

Many preconditioning methods that originally have been introduced for some concrete application (e.g., for the Stokes problem), belong to more general classes of block preconditioners. For a rough overview, one can mathematically distinguish (indefinite) block diagonal preconditioners (cf. Benzi and Simoncini [22], Benzi et al. [23], de Sturler and Liesen [50], Mandel [126]), block triangular preconditioners (cf. Axelsson and Neytcheva [9], Simoncini [178]), the class of (inexact) Uzawa preconditioners (cf. Bramble et al. [34], Elman and Golub [60], Zulehner [235]) and block approximate factorization preconditioners (cf. Elman et al. [59], Patankar and Spalding [154], Vuik et al. [205]). Some papers discuss specific design decisions for block preconditioners, e.g., “constraint preconditioning” in Keller et al. [102]. Others address the eigenvalue spectrum of the preconditioned matrices in general (cf. Murphy et al. [136], Notay [145], Zulehner [235]). It shall be mentioned that there exist other methods which in contrast to above block smoothers are based on some more local (cell-based or patch-based) information, such as Vanka-type smoothers (cf. Vanka [191], Wobker and Turek [220]). Of course, this list is far from being complete.

In this section, the focus is on multigrid methods for saddle point problems. Multigrid hierarchies may suffer from stability issues. Thus, the stability of the discretization scheme plays an important role for building coarse representations of the fine level problem in context of multigrid methods. In the work by Wabro [206] a coupled AMG method is developed and analyzed for a stabilized mixed finite element discretization of the Oseen equations, which utilizes special techniques to preserve stability on the coarser levels by scaling the standard Galerkin product (2.6). However, at least for the Stokes problem, extensive numerical studies in Janka [101] reveal that uniform inf–sup stability of coarse level operators is not necessary for obtaining a successful preconditioner. It is a common observation that the pure Galerkin product (2.6) generates efficient multigrid preconditioners (see, e.g., statements in Stüben [181]). Note that the aggregation-based transfer operators as introduced in Section 3.4 are inherently scaled by the aggregate size in the local QR-decomposition to compensate the coarsening effect on the numerical values within the transfer operators.

In the following, our special emphasis is placed on saddle point problems arising from structural contact problems. The theory for multigrid methods designed for this particular class of saddle point problems and the resulting linear systems has evolved starting from special multigrid methods for mortar finite element methods (e.g., Braess et al. [30], Gopalakrishnan and Pasciak [78]) to mortar finite element methods in saddle point formulation (e.g., Braess and Dahmen [32], Braess et al. [33]). Recent works on specific multigrid methods for structural contact problems in saddle point formulation are based on the ideas from the mortar finite element method and introduce a very flexible multigrid framework that can be applied to a wide variety of different mortar situations including structural contact problems (e.g., Wieners and Wohlmuth [215, 216]). Based on the saddle point formulation in Belgacem [18] for the mortar finite element method, the work by Wohlmuth [222] describes a multigrid method for the corresponding saddle point problems using (continuous) standard basis functions for the Lagrange multiplier space. This method is extended for dual basis functions for the non-nested Lagrange multiplier spaces in Wieners and Wohlmuth [215]. That is, the multigrid theory for mortar and contact problems covers standard Lagrange multiplier spaces as well as non-continuous dual Lagrange multiplier spaces (cf. Section 5.4.2). The mathematical proofs are originally designed for common mortar finite element problems. Nevertheless, the general ideas should also work for our class of problems arising from contact mechanics. Before introducing the design of full multigrid methods for linear systems as given in (7.1) a brief discussion ot the mathematical background is given in the following section.

7.1.3. Mathematical fundament and multigrid convergence theory

This section is meant to provide an overview of the mathematical basics of multigrid methods designed for mortar problems in saddle point formulation. Since the resulting linear systems are indefinite, it is clear that standard multigrid methods cannot be applied. Here, the approach presented in Wieners and Wohlmuth [215] is reviewed, as it describes the mathematical fundament for the methods that we want to use in the next sections.

Similarly to (5.34), a quasi-static mortar problem is considered, which is given in its discrete version by: Find $\big(\boldsymbol{u}^{\mathrm{h}}, \boldsymbol{\lambda}^{\mathrm{h}}\big) \in \boldsymbol{\mathcal{U}}^{\mathrm{h}} \times \boldsymbol{\mathcal{M}}^{\mathrm{h}}$, such that

$$\begin{array}{lclcll} a\big(\boldsymbol{u}^{\mathrm{h}}, \boldsymbol{v}^{\mathrm{h}}\big) & + & b\big(\boldsymbol{v}^{\mathrm{h}}, \boldsymbol{\lambda}^{\mathrm{h}}\big) & = & f(\boldsymbol{v}^{\mathrm{h}}), & \boldsymbol{v}^{\mathrm{h}} \in \boldsymbol{\mathcal{V}}^{\mathrm{h}}_{\boldsymbol{u}}, \\ b\big(\boldsymbol{u}^{\mathrm{h}}, \boldsymbol{\mu}^{\mathrm{h}}\big) & & & = & 0, & \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}^{\mathrm{h}}. \end{array} \tag{7.3}$$

Without giving details, $\boldsymbol{\mathcal{M}}^{\mathrm{h}}$ denotes the discrete Lagrange multiplier space. Following the conventions of the saddle point literature one defines $b\big(\boldsymbol{v}, \boldsymbol{\mu}\big) := \big\langle [\boldsymbol{v}] , \boldsymbol{\mu} \big\rangle_{\gamma_{\mathrm{c}}^{(s)}}$ using (5.21).

Remark 7.1.1. The system (7.3) describes a classical symmetric saddle point system resulting from a mortar problem. The main difference to the contact problem (5.34) is the type of constraints. In (7.3) one has an equality constraint, whereas for contact problems there is some additional complexity resulting from the inequality constraint in (5.34b).

In Wieners and Wohlmuth [215, Lemma 3.3] the authors prove the inf–sup condition

$$0 < C \leq \inf_{\boldsymbol{\mu}^{\mathrm{h}} \in \boldsymbol{\mathcal{M}}^{\mathrm{h}}} \sup_{\boldsymbol{v}^{\mathrm{h}} \in \boldsymbol{\mathcal{V}}^{\mathrm{h}}} \frac{b\big(\boldsymbol{v}^{\mathrm{h}}, \boldsymbol{\mu}^{\mathrm{h}}\big)}{\|\boldsymbol{v}^{\mathrm{h}}\|_1 \, \|\boldsymbol{\mu}^{\mathrm{h}}\|_M}$$

with a constant $C > 0$ independent of h. Therein, $\|\cdot\|_M$ denotes the canonical norm for the Lagrange multiplier space defined by

$$\|\boldsymbol{\mu}\|_M := \sup_{\boldsymbol{v} \in \boldsymbol{\mathcal{V}}} \frac{b\big(\boldsymbol{v}, \boldsymbol{\mu}\big)}{\|\boldsymbol{v}\|_1}, \quad \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}^{\mathrm{h}}.$$

Similar inf–sup conditions for slightly different norms can also be found in Wohlmuth [221] and Braess et al. [33]. The inf–sup stability is essential for the saddle point theory to prove stability of the discretization method (cf. Metsch [133]). In contrast to earlier works (e.g., Braess and Dahmen [32], Braess et al. [33], Wieners and Wohlmuth [216], Wohlmuth [222]) the proof does not require the Lagrange multiplier spaces to be nested (i.e., $\boldsymbol{\mathcal{M}}^{2\mathrm{h}} \not\subset \boldsymbol{\mathcal{M}}^{\mathrm{h}}$). That is, the proof is also valid for dual Lagrange multiplier spaces as introduced, e.g., in Kim et al. [105] and Wohlmuth [223, 225] which have locally supported basis functions and, in general, are non-nested.

For the convergence theory of multigrid methods the mesh-dependent norm

$$\|(\boldsymbol{v},\boldsymbol{\mu})\|_{\mathrm{h}}^2 = \|\boldsymbol{v}\|_{0,\Omega}^2 + \mathrm{h}^3 \|\boldsymbol{\mu}\|_{0,\Gamma_{\mathrm{c}}}^2 \tag{7.4}$$

is introduced in Wieners and Wohlmuth [215, Corollary 3.8] to prove the dual estimate

$$\left\|(\boldsymbol{u}^{\mathrm{h}} - \boldsymbol{u}^{2\mathrm{h}}, \boldsymbol{\lambda}^{\mathrm{h}} - \boldsymbol{\lambda}^{2\mathrm{h}})\right\|_{\mathrm{h}} \leq C\mathrm{h}^2 \|f\|_0\,, \quad C > 0, \tag{7.5}$$

assuming that $(\boldsymbol{u}^{2\mathrm{h}}, \boldsymbol{\lambda}^{2\mathrm{h}}) \in \boldsymbol{\mathcal{U}}^{2\mathrm{h}} \times \boldsymbol{\mathcal{M}}^{2\mathrm{h}}$ is the solution of a coarse representation of the contact problem. Then, the multigrid analysis is performed for the W-cycle setting using standard arguments from Hackbusch [86, 87] based on a smoothing and approximation property (cf. Section 2.5.3.2):

Approximation property: Let K_ℓ with

$$\mathsf{K}_\ell := \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_1 & -\mathsf{Z} \end{pmatrix}_\ell \tag{7.6}$$

denote the full symmetric saddle point matrix corresponding to (7.3) on multigrid level ℓ with P and R being appropriate multigrid transfer operators between the multigrid levels ℓ and $\ell+1$. Then, the result (7.5) can be used to prove the approximation property

$$\left\|\mathsf{K}_\ell^{-1} - P\mathsf{K}_{\ell+1}^{-1}R\right\| \leq C\mathrm{h}_\ell^2 \tag{7.7}$$

with a constant $C < \infty$ independent of the multigrid level ℓ. For the proof see Wieners and Wohlmuth [215, Lemma 4.3].

Smoothing property: In a saddle point multigrid method the level smoothers are responsible to consider the equality constraints in (7.3) in the solution process. The characteristic idea is to satisfy the equality constraints in each smoothing step of the multigrid level smoother. In Bank et al. [14] the authors introduce a broad class of saddle point preconditioners based on a block approximate factorization for block matrices as in (7.6), which rely on the solution of a modified Schur complement equation. The Braess–Sarazin smoother (cf. Braess and Sarazin [29]) can easily be identified as a special case in this class of Schur complement based saddle point smoothers. Furthermore, SIMPLE type smoothers and variants (cf. Elman et al. [59], Patankar and Spalding [154], Patankar [155]) are based on the same ideas of a block factorization and a modified Schur complement equation. The solution of the Schur complement system is often considered to be too expensive. However, in Zulehner [234] it is shown that the smoothing property is preserved even if the Schur complement block S is replaced by some approximation $\widetilde{S}$, i.e., an approximate inexact solution of the modified Schur complement equation is sufficient.

Let $\widetilde{\mathsf{K}}_\ell$ be defined as the approximate block system (7.6), where the block K is replaced by some positive definite approximation $\widetilde{\mathsf{K}}$ that is easy to invert. The $\widetilde{\mathsf{K}}$ is also used to build a good approximation of the Schur complement operator $\widetilde{S}_\ell := \mathsf{Z} + \mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}}$ within reasonable computational time. In practice, one can apply some iterative method for approximately solving the modified Schur complement system (e.g., some CG iterations with a Gauss–Seidel preconditioner as described in Wieners and Wohlmuth [215]). Then, Zulehner [234] proves that the smoothing property

$$\left\| \mathsf{K}_\ell \left(1 - \widetilde{\mathsf{K}}_\ell^{-1} \mathsf{K}_\ell\right)^\nu \right\|_e \leq \eta(\nu) \left\| \widetilde{\mathsf{K}}_\ell - \mathsf{K}_\ell \right\|_e \tag{7.8}$$

holds with $\eta(\nu) \to 0$ for $\nu \to \infty$ assuming that $0 \leq \mathsf{K}_\ell \leq \widetilde{\mathsf{K}}_\ell$ and $0 < \widetilde{S} \leq S \leq (1+\beta)\widetilde{S}$ for $\beta < \frac{1}{3}$. It shall be mentioned that there is a new analysis of block preconditioners for saddle point problems in Notay [145] based on eigenvalue estimates. Among others, it also includes the class of block approximate factorization preconditioners like Braess–Sarazin and provides some insight into the effect of different block preconditioners with internal approximations (of, e.g., the Schur complement S) to the eigenvalue spectrum of the preconditioned matrix.

Following the standard multigrid approach in Hackbusch [87], one obtains level independent convergence rates with the approximation property (7.7) and the smoothing property (7.8) at least for the two-level method, provided that the number of level smoothing sweeps is sufficiently large.

Independent from the theoretical proofs, the numerical results in the literature based on above mathematical fundament (cf. Wieners and Wohlmuth [215]) are quite promising both for standard as well as dual Lagrange multiplier spaces. It is a common experience that multigrid methods show good convergence behavior in practice even for problem classes where no full mathematical proof exists, yet (cf. Braess et al. [33], Wieners and Wohlmuth [215]). Therefore, it is interesting to study these methods also in an algebraic multigrid setting for the more complicated structural contact problems, even though the approximation and smoothing properties are formulated and proven for the classical multigrid convergence theory based on Hackbusch [87].

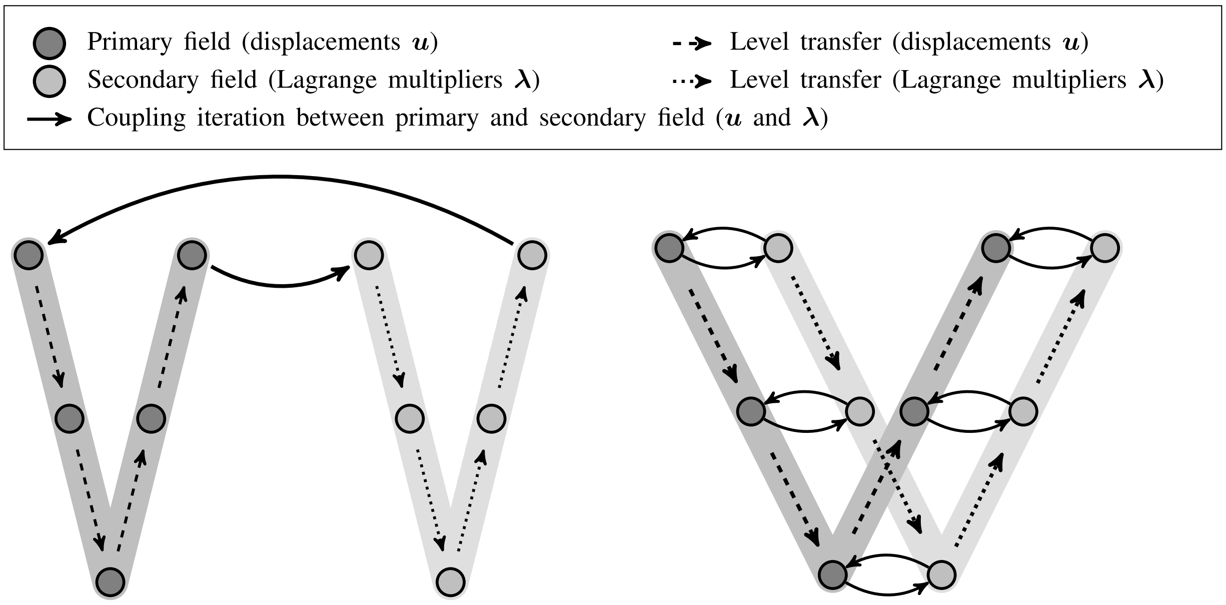
7.2. Full AMG design for coupled problems

7.2.1. Truly monolithic AMG for coupled problems

Solving a monolithic coupled block system as given in (7.1) requires a coupling algorithm for the different physical and mathematical fields which can deal with the saddle point structure. In general there are the following two different ways of how to use multigrid ideas together with widely-used Schur complement based methods for solving saddle point problems:

Nested multigrid approach: Well-known Schur complement based block preconditioners such as the SIMPLE method (cf. Patankar and Spalding [154]) and variants can be combined with multigrid methods in a straightforward manner as shown in Figure 7.1a, where multigrid methods serve as local single field smoother. The contact constraints can only be considered in the outer (SIMPLE) iteration. For the examples in this thesis, (cheap) variants of the SIMPLE algorithm as later described in Section 7.2.4.4 are used with some approximations that are introduced in Section 7.2.5.1. The general approach is already known in the literature and, e.g., described by Griebel et al. [80] and Stüben [181] for the Navier–Stokes equations.

Full multigrid approach: As an alternative to the nested approach, a truly monolithic algebraic multigrid method for saddle point problems is developed in this thesis, specifically for


Primary field (displacements $\boldsymbol{u}$)
Secondary field (Lagrange multipliers $\boldsymbol{\lambda}$)
Coupling iteration between primary and secondary field ($\boldsymbol{u}$ and $\boldsymbol{\lambda}$)
Level transfer (displacements $\boldsymbol{u}$)
Level transfer (Lagrange multipliers $\boldsymbol{\lambda}$)

(a) Nested Multigrid approach (b) Full Multigrid approach

Figure 7.1.: Algorithmic layout of a 3 level multigrid method for coupled problems (cf. Gee et al. [72], Küttler [112]).

problems arising from structural contact problems. The basic idea of the full multigrid approach is to keep the saddle point structure on the coarser levels and apply (indefinite) 2×2 block level smoothers, which can deal with the saddle point structure. As shown in Figure 7.1b, this approach basically switches the role of the multigrid cycle and the coupling iteration between the different fields using saddle point smoothers. This has the advantage that the contact constraints are considered on all multigrid levels. To keep the saddle point block structure with a clean distinction of the different underlying physical and mathematical fields on all multigrid levels, one needs special (segregated) transfer operators. In the next sections, all the necessary details for these transfer operators are discussed, including the aggregation strategy for the Lagrange multipliers and the level block smoothers that are used for the resulting saddle point problems on the different multigrid levels.

Remark 7.2.1 (Comparison to other methods). Similar techniques to the full multigrid approach are already known from the literature. They have been successfully applied to the Stokes equations (e.g., Braess and Sarazin [29], Janka [101]), the Oseen equations (e.g., Wabro [206]) and Navier–Stokes equations (e.g., Webster [210]). In contrary to these contributions, in our case one has to deal with an interface coupled problem, where some specialized techniques are required to define a proper coarsening process for the Lagrange multipliers. The full multigrid approach has also been applied to Fluid-Structure-Interaction (FSI) problems in Gee et al. [72]. Therein the authors propose the same algorithmic principle, but use a block Gauss–Seidel method as multigrid smoother to couple the different physical fields of the FSI problem. A similar approach based on classical AMG methods is also given in Yang and Zulehner [231] and Langer and Yang [115]. A FSI problem is a typical example for an interface coupled problem. However, it allows to coarsen the different physical fields independently from each other, which is not possible for structural contact problems with contact constraint equations. For contact problems the full multigrid approach is already pursued in Adams [1] in a slightly different variant as introduced later in this work.

7.2.2. Segregated transfer operators

To keep the characteristic saddle point block structure (7.2) on all multigrid levels, the common approach is to use *segregated* transfer operators

$$P_{\ell+1} = \begin{pmatrix} P^{\boldsymbol{u}} & 0 \\ 0 & \widehat{P}^{\boldsymbol{\lambda}} \end{pmatrix}_{\ell+1} \quad \text{and} \quad R_{\ell+1} = \begin{pmatrix} R^{\boldsymbol{u}} & 0 \\ 0 & \widehat{R}^{\boldsymbol{\lambda}} \end{pmatrix}_{\ell+1}, \tag{7.9}$$

as, e.g., introduced in Braess and Sarazin [29] or Adams [1]. The segregated block transfer operators (7.9) are put together from the transfer operator blocks for the different fields. Here, $P^{\boldsymbol{u}}$ and $R^{\boldsymbol{u}}$ describe the transfer operator blocks corresponding to the stiffness matrix block K in (7.2). The transfer operators $\widehat{P}^{\boldsymbol{\lambda}}$ and $\widehat{R}^{\boldsymbol{\lambda}}$ define the level transfer for the Lagrange multipliers.

Remark 7.2.2 (Number of multigrid levels). In the most general case, one could have a different number of multigrid levels for each physical or mathematical field. This is very likely to happen for interface-coupled problems (such as FSI problems) where the size of the domains of the different physical fields might vary significantly. To handle this, the most convenient way is to fill the "missing" transfer operator blocks with identity blocks for building the multigrid hierarchy. This way, the physical and mathematical fields with the smaller number of multigrid levels stop coarsening but are still involved in the coupling on the coarsest level (cf. Gee et al. [72]). However, in our case, one has no different (independent) fields, such that the structural equations and the corresponding Lagrange multipliers can be linked together on all multigrid levels by just using a smart aggregation technique (see Section 7.2.3 below).

The block diagonal structure in (7.9) guarantees that the primary displacement variables and the secondary Lagrange multipliers are not "mixed up" on the coarser levels. Using the standard Galerkin approach the coarse level system is given by

$$\begin{pmatrix} R^{\boldsymbol{u}} & 0 \\ 0 & \widehat{R}^{\boldsymbol{\lambda}} \end{pmatrix}_{\ell+1} \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix}_{\ell} \begin{pmatrix} P^{\boldsymbol{u}} & 0 \\ 0 & \widehat{P}^{\boldsymbol{\lambda}} \end{pmatrix}_{\ell+1} = \begin{pmatrix} R^{\boldsymbol{u}}\mathsf{K}P^{\boldsymbol{u}} & R^{\boldsymbol{u}}\mathsf{C}_1^{\mathsf{T}}\widehat{P}^{\boldsymbol{\lambda}} \\ \widehat{R}^{\boldsymbol{\lambda}}\mathsf{C}_2P^{\boldsymbol{u}} & -\widehat{R}^{\boldsymbol{\lambda}}\mathsf{Z}\widehat{P}^{\boldsymbol{\lambda}} \end{pmatrix}_{\ell+1}, \tag{7.10}$$

i.e., the coarse level matrix (7.10) still has the same block structure with a clear distinction of momentum and constraint equations as for the fine level problem.

Whereas for many coupled problems (e.g., Fluid-Structure-Interaction problems, Thermo-Structure-Interaction problems) it is straightforward to generate $\widehat{P}^{\boldsymbol{\lambda}}$ and $\widehat{R}^{\boldsymbol{\lambda}}$, in case of structural contact problems this is a non-trivial task, as one cannot use the Z block to generate valid aggregation information for the Lagrange multipliers due to an insufficient pattern of the Z block. Consequently, one needs a special routine for finding aggregates for the Lagrange multipliers $\boldsymbol{\lambda}$ to be able to build the (non-smoothed) transfer operators $\widehat{P}^{\boldsymbol{\lambda}}$ and $\widehat{R}^{\boldsymbol{\lambda}}$. For an interface coupled problem it seems natural to apply an interface aggregation method as introduced in Section 7.2.3 for coarsening the Lagrange multipliers.

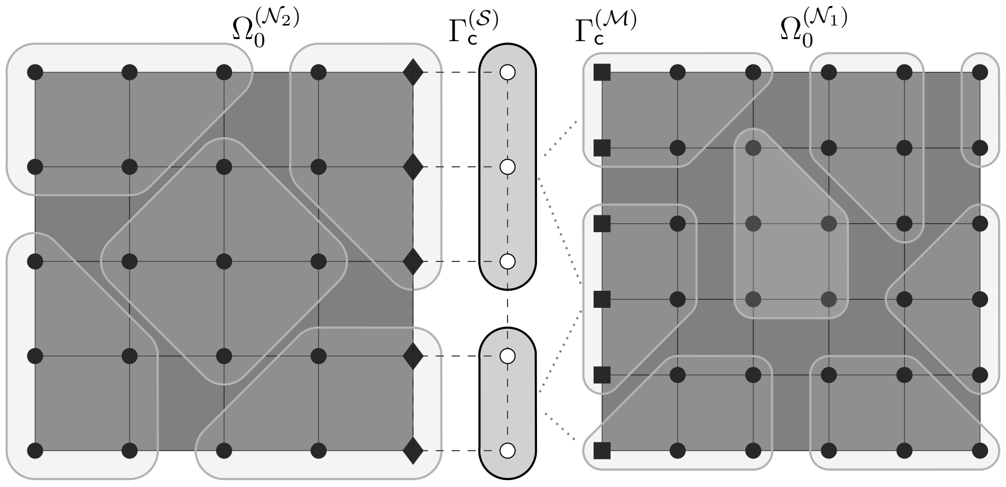
Remark 7.2.3 (Near null space vectors). With the knowledge from Section 3.4.1, a set of near null space vectors is necessary to generate the non-smoothed transfer operators $\widehat{P}^{\boldsymbol{u}}$ and $\widehat{P}^{\boldsymbol{\lambda}}$. The $\widehat{P}^{\boldsymbol{u}}$ is built directly from the symmetric block K in (7.2) representing the structural equations only. Therefore, the full rigid body modes with translatory and rotatory part are used as near null space vectors. However, for the Lagrange multiplier blocks $\widehat{P}^{\boldsymbol{\lambda}}$ one only uses component-wise constants as near null space vectors (for linear momentum conservation, cf. Popp [156, Section 4.2.6]). Be aware that this way one has 6 degrees of freedom per node on the coarse level for describing the displacement variables but only 3 degrees of freedom per node for the Lagrange multipliers in a 3D example. This also helps to keep the operator complexity small, since the saddle point approach in general suffers from a higher memory consumption due to keeping the Lagrange multipliers as extra variables. Of course, one can extend the set of near null space vectors if better near null space information is available.

Remark 7.2.4 (Transfer operator smoothing). As one might have already noticed, non-smoothed transfer operators $\widehat{P}^{\boldsymbol{\lambda}}$ and $\widehat{R}^{\boldsymbol{\lambda}}$ are used in (7.9). The reason is that in contrary to $P^{\boldsymbol{u}}$ and $R^{\boldsymbol{u}}$, where all available smoothing techniques for the transfer operators can be applied, there is no appropriate diagonal-dominant block Z for smoothing the transfer operator basis functions. Of course, one could generate a valid matrix for smoothing by hand (e.g., some kind of distance Laplacian), but this effort is suspected to only pay off if this information is also used somewhere else (see also the discussion in Section 7.2.3).

In the following, the focus is on the interface aggregation strategy that is used to link the number of multigrid levels for the Lagrange multipliers automatically with the number of multigrid levels for the displacement degrees of freedom.

7.2.3. Aggregation strategy for Lagrange multipliers

Most of the literature available on multigrid for contact problems is primarily on geometric multigrid methods with abundant work on saddle point smoothers (cf. Wieners and Wohlmuth [216], Wohlmuth [222], Zulehner [234]). In this thesis, the usage of a smoothed aggregation AMG method is proposed for the saddle point problems arising from structural contact problems. In contrast to geometric multigrid methods, there is not so much literature on aggregation-based AMG methods for contact problems in saddle point formulation. The only publication, the author is aware of covering all aspects of smoothed aggregation methods for structural contact problems in saddle point formulation, is Adams [1], which also discusses a special aggregation strategy for the Lagrange multipliers. To find aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ for the Lagrange multipliers, Adams [1] proposes to apply the standard aggregation algorithm (see Section 3.3) to the graph of a suitable matrix representing the Lagrange multipliers. A natural but inefficient choice for such a graph could be $G\big(\mathsf{C}_2\mathsf{C}_1^{\mathsf{T}}\big)$. However, according to Adams [1], this choice turns out to be not suitable for contact constraints because $\mathsf{C}_2\mathsf{C}_1^{\mathsf{T}}$ tends to be diagonal for aligned grids, which leads to slow coarsening. In Adams [1], the choice $G\big(\mathsf{C}_2 P^{\boldsymbol{u}}(P^{\boldsymbol{u}})^{\mathsf{T}}\mathsf{C}_1^{\mathsf{T}}\big)$ is found to be promising, where $P^{\boldsymbol{u}}$ denotes the prolongation operator associated with the displacement variables built from the aggregates for the K block in (7.6). Note that $P^{\boldsymbol{u}}(P^{\boldsymbol{u}})^{\mathsf{T}}$ can be understood as the symmetric matrix representing the extended connectivity defined by the coarse level basis functions. Therefore, the graph $G\big(\mathsf{C}_2 P^{\boldsymbol{u}}(P^{\boldsymbol{u}})^{\mathsf{T}}\mathsf{C}_1^{\mathsf{T}}\big)$ can be interpreted as the graph of a reasonable approximation of the Schur complement $S = \mathsf{Z} + \mathsf{C}_2\mathsf{K}^{-1}\mathsf{C}_1^{\mathsf{T}}$. Even though this approach is shown to work for some examples in Adams [1], it has several drawbacks. First, the graph used for the


$\Omega_0^{(\mathcal{N}_2)}$
$\Gamma_c^{(\mathcal{S})}$
$\Gamma_c^{(\mathcal{M})}$
$\Omega_0^{(\mathcal{N}_1)}$

●	Inner nodes with displacement degrees of freedom $\boldsymbol{u}_{\mathcal{N}_1}$ and $\boldsymbol{u}_{\mathcal{N}_2}$
◆	Nodes at slave contact interface with displacement degrees of freedom $\boldsymbol{u}_{\mathcal{A}}$ and $\boldsymbol{u}_{\mathcal{I}}$
○	Virtual "nodes" for Lagrange multiplier degrees of freedom $\boldsymbol{\lambda}_{\mathcal{A}}$ and $\boldsymbol{\lambda}_{\mathcal{I}}$
■	Nodes at master contact interface with displacement degrees of freedom $\boldsymbol{u}_{\mathcal{M}}$
⬭	Aggregate in $\mathscr{A}_\ell^{\boldsymbol{u}}$ for displacement degrees of freedom $\boldsymbol{u}_{\mathcal{N}_1}$ and $\boldsymbol{u}_{\mathcal{N}_2}$
⬬	Aggregate in $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ for Lagrange multipliers $\boldsymbol{\lambda}$

Figure 7.2.: Aggregation for contact example in saddle point formulation

aggregation of the Lagrange multipliers $\boldsymbol{\lambda}$ has to be built explicitly to serve as input for the standard aggregation algorithm. Secondly, one has to run the aggregation algorithm from Section 3.3 sequentially both for the displacement degrees of freedom and for the Lagrange multipliers. For the second run of the aggregation method one might have to use a different set of aggregation parameters to obtain optimal results, which further increases the complexity for the user. Even though indirectly based on the aggregation information from the displacement aggregates $\mathscr{A}_\ell^{\boldsymbol{u}}$ through $P^{\boldsymbol{u}}$, the resulting aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ for the Lagrange multipliers are built independently from the displacement aggregates $\mathscr{A}_\ell^{\boldsymbol{u}}$.

In this thesis, a different approach to build aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ for the Lagrange multipliers is proposed which does not suffer from above drawbacks. For finding appropriate Lagrange multiplier aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ no separate graph is built for the use in the aggregation strategy as proposed in Adams [1]. Instead, the interface information is algebraically reconstructed from the contact slave interface and interface aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ are built for the Lagrange multipliers using the aggregation information for the displacement degrees of freedom directly. This way, one can cheaply build interface aggregates for the Lagrange multipliers, which are by construction consistent with the corresponding displacement aggregates (see Figure 7.2).

Remark 7.2.5 (Segregated aggregates). With the knowledge of the master and slave degrees of freedom, the matrix entries between master and slave degrees of freedom in (7.2) are dropped "on the fly" when building the displacement aggregates $\mathscr{A}_\ell^{\boldsymbol{u}}$. This way one can guarantee the displacement aggregates $\mathscr{A}_\ell^{\boldsymbol{u}}$ not to cross the contact interface. Basically, this corresponds to using the modified matrix block

$$\begin{pmatrix} \mathsf{K}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{M}} & 0 & 0 & 0 \\ \mathsf{K}_{\mathcal{M}\mathcal{N}_1} & \mathsf{K}_{\mathcal{M}\mathcal{M}} & 0 & 0 & 0 \\ 0 & 0 & \mathsf{K}_{\mathcal{I}\mathcal{I}} & \mathsf{K}_{\mathcal{I}\mathcal{A}} & \mathsf{K}_{\mathcal{I}\mathcal{N}_2} \\ 0 & 0 & \mathsf{K}_{\mathcal{A}\mathcal{I}} & \mathsf{K}_{\mathcal{A}\mathcal{A}} & \mathsf{K}_{\mathcal{A}\mathcal{N}_2} \\ 0 & 0 & \mathsf{K}_{\mathcal{N}_2\mathcal{I}} & \mathsf{K}_{\mathcal{N}_2\mathcal{A}} & \mathsf{K}_{\mathcal{N}_2\mathcal{N}_2} \end{pmatrix} \tag{7.11}$$

as input for the standard aggregation routine. Instead of the graph of (7.11) one could directly use the graph $G\big(\mathbf{K}_{\mathrm{T}}(\boldsymbol{u}^i)\big)$ from (5.44) without the linearized contact forces in $\mathbf{K}_{\mathrm{co}}(\boldsymbol{u}^i, \boldsymbol{\lambda}^i)$ to obtain the same effect of segregating the aggregates. However, it is often too expensive to hold the tangential stiffness matrix extra in memory only for the aggregation.

The exact aggregation procedure is described in Algorithm 10. Assuming that the standard aggregates $\mathscr{A}_\ell^{\boldsymbol{u}}$ for the displacement degrees of freedom are available, one loops over the nodes at the slave contact interface and builds new aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ by collecting the corresponding Lagrange multiplier degrees of freedom. Beside the displacement aggregates $\mathscr{A}_\ell^{\boldsymbol{u}}$ one only needs the mortar matrix D as input for Algorithm 10 to algebraically reconstruct the contact interface. The new aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$ for the Lagrange multipliers are just the natural extension of the displacement aggregates $\mathscr{A}_\ell^{\boldsymbol{u}}$ at the interface. This way, one can keep the ratio of coarse level nodes at the slave contact interface and the coarse Lagrange multipliers constant, which also balances the ratio of contact constraints and inner structural displacement degrees of freedoms over all multigrid levels.

7.2.4. Block smoothers for saddle point problems

Classical relaxation-based methods as described in Section 2.1 can be extended to block matrices. However, in case of saddle point problems, special smoothing strategies are necessary to handle the saddle point block structure of the (indefinite) block matrix.

Typical saddle point smoothers are based on the same formulation as given in Definition 2.1.1, that is

$$\begin{bmatrix} \Delta\boldsymbol{u}^{k+1} \\ \Delta\boldsymbol{\lambda}^{k+1} \end{bmatrix} = \begin{bmatrix} \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} + Q^{-1}\left(\begin{bmatrix} \mathbf{r}_{\boldsymbol{u}}^k \\ \mathbf{r}_{\boldsymbol{\lambda}}^k \end{bmatrix} - \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} \right), \tag{7.12}$$

where Q describes the 2×2 block preconditioning matrix approximating the 2×2 block operator in (7.2). In the following a few of the classical block smoothers from the literature (e.g., Notay [145]) are introduced, stating that this list is by far not complete.

7.2.4.1. Indefinite block diagonal preconditioner

The block diagonal preconditioner for indefinite linear 2×2 block systems is based on the preconditioning matrix

$$Q := \begin{pmatrix} \frac{1}{\alpha}\mathsf{K} & 0 \\ 0 & -\frac{1}{\alpha}\widetilde{S} \end{pmatrix} \tag{7.13}$$

in (7.12). The $\alpha > 0$ denotes a relaxation or damping parameter and $\widetilde{S}$ is supposed to be an approximation of the Schur complement $S := \mathsf{Z} + \mathsf{C}_2\mathsf{K}^{-1}\mathsf{C}_1^{\mathsf{T}}$. Usually, one replaces the block K in the Schur complement operator S by a cheap and easy-to-invert approximation $\widetilde{\mathsf{K}}$. The coupling between the displacement degrees of freedom and the Lagrange multipliers in the off-

Algorithm 10: Aggregation algorithm for Lagrange multipliers.

```
Procedure LagMultAggregation($\mathscr{A}_\ell^{\boldsymbol{u}}$,D)
    Initialize empty set and counter for aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$
    $\mathscr{A}_\ell^{\boldsymbol{\lambda}} \leftarrow \emptyset,\ l \leftarrow 0$
    Initialize empty mapping of displacement aggregates to Lagrange multiplier
    aggregates
    $\mathsf{d}(k) \leftarrow \emptyset \quad \forall k = 1, \ldots, m_{\mathscr{A}_\ell^{\boldsymbol{u}}}$
    Loop over slave DOFs (rows of D)
    for $i \in \mathcal{D}_{\mathcal{S}}$ do
        Find displacement node $n^{\boldsymbol{u}}$ id corresponding to displacement DOF $i$
        $n^{\boldsymbol{u}} \leftarrow \mathrm{n}(i)$
        Find aggregate index $k$ that contains displacement node $n^{\boldsymbol{u}}$
        Find $k$ with $\mathscr{A}_\ell^{(k)} \in \mathscr{A}_\ell^{\boldsymbol{u}}$ where $n^{\boldsymbol{u}} \in \mathscr{A}_\ell^{(k)}$
        Loop over all Lagrange multipliers $j$
        for $j \in \mathcal{D}_{\boldsymbol{\lambda}}$ do
            Check whether Lagrange multiplier $j$ is coupled with row $i$
            if $\mathsf{D}_{i,j} \neq 0$ then
                Find pseudo node $n^{\lambda}$ for Lagrange multiplier $j$
                $n^{\lambda} \leftarrow \mathrm{n}(j)$
                Check whether to build a new Lagrange multiplier aggregate
                if $\mathsf{d}(k) = \emptyset$ then
                    Increment internal aggregation counter
                    $l \leftarrow l + 1$
                    Build a new aggregate and add Lagrange multiplier node $n^{\lambda}$
                    $\mathscr{A}_\ell^{(l)} \leftarrow \{n^{\lambda}\}$
                    Associate displacement aggregate $k$ with Lagrange multiplier
                    aggregate $l$
                    $\mathsf{d}(k) \leftarrow \{l\}$
                    Add new aggregate to set of Lagrange multiplier aggregates $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$
                    $\mathscr{A}_\ell^{\boldsymbol{\lambda}} \leftarrow \mathscr{A}_\ell^{\boldsymbol{\lambda}} \cup \mathscr{A}_\ell^{(l)}$
                else
                    Extend aggregate $0 \leq \mathsf{d}(k) \leq l$ with pseudo node
                    $\mathscr{A}_\ell^{(\mathsf{d}(k))} \leftarrow \mathscr{A}_\ell^{(\mathsf{d}(k))} \cup \{n^{\lambda}\}$
                end
            end
        end
    end
    Return aggregates for Lagrange multipliers
    return $\mathscr{A}_\ell^{\boldsymbol{\lambda}}$
```

diagonal blocks is completely dropped in the block preconditioning matrix Q from (7.13). Even though cheap, the error matrix of the (indefinite) block diagonal smoother, shows the limitations of the method for coupled problems.

Remark 7.2.6 (Error matrix). The error matrix for the indefinite block diagonal preconditioner can be calculated as

$$\mathsf{E}_{\text{IBD}} := A - Q = \begin{pmatrix} (1-\frac{1}{\alpha})\mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & -\mathsf{Z} + \frac{1}{\alpha}\widetilde{S} \end{pmatrix}. \tag{7.14}$$

Obviously, the contact constraints are not considered in the level smoothing process, which results in higher iteration numbers in the linear solver.

Algorithm 11 gives an algorithmic description of the iterative block smoothing process (7.12) with Q defined by (7.13). Internally, one needs the inverse of the block K and the approximate Schur complement operator $\widetilde{S}$ to solve for the displacement increments and the Lagrange multiplier increments. For an efficient implementation, one does not solve for the local block systems exactly. Instead, one applies a small number of smoothing sweeps with symmetric Gauss–Seidel or an ILU method (see also Section 7.2.5.1) to keep the computational costs low.

For some theoretical background about the mathematical properties of indefinite block diagonal preconditioners the reader is referred to, e.g., de Sturler and Liesen [50] or Benzi and Simoncini [22].

Algorithm 11: Indefinite block diagonal smoother.

```
Procedure IndefBlockDiagonal($\alpha$, $k_{\max}$)
    Apply $k_{\max}$ smoothing sweeps with the indefinite block diagonal algorithm
    for $k \leftarrow 0$ to $k_{\max} - 1$ do
        Determine prediction increments $\delta\boldsymbol{u}^{k+1}$ by solving approximately
        $\mathsf{K}\,\delta\boldsymbol{u}^{k+1} = \mathbf{r}_{\boldsymbol{u}}^k - \mathsf{K}\Delta\boldsymbol{u}^k - \mathsf{C}_1^{\mathsf{T}}\Delta\boldsymbol{\lambda}^k$
        Solve approximately for the Lagrange multiplier increment $\delta\boldsymbol{\lambda}^{k+1}$
        $-\widetilde{S}\,\delta\boldsymbol{\lambda}^{k+1} = \mathbf{r}_{\boldsymbol{\lambda}}^k - \mathsf{C}_2\Delta\boldsymbol{u}^{k+1} + \mathsf{Z}\Delta\boldsymbol{\lambda}^k$
        Update step: update solution variables
        $\Delta\boldsymbol{u}^{k+1} \leftarrow \Delta\boldsymbol{u}^k + \alpha\,\delta\boldsymbol{u}^{k+1}$
        $\Delta\boldsymbol{\lambda}^{k+1} \leftarrow \Delta\boldsymbol{\lambda}^k + \alpha\,\delta\boldsymbol{\lambda}^{k+1}$
    end
    Return smooth solution vector
    return $(\Delta\boldsymbol{u}^{k_{\max}}, \Delta\boldsymbol{\lambda}^{k_{\max}})$
```

7.2.4.2. Uzawa smoother

The (inexact) Uzawa smoothers can be understood as improvement of the indefinite block diagonal preconditioners from Section 7.2.4.1 by using

$$Q := \frac{1}{\alpha}\begin{pmatrix} \mathsf{K} & 0 \\ \mathsf{C}_2 & -\widetilde{S} \end{pmatrix} \tag{7.15}$$

instead of (7.13). Again, $\alpha > 0$ is a damping parameter and $\widetilde{S}$ describes a cheap approximation of the Schur complement S. The better Q approximates the block operator from (7.2), the lower the number of linear iterations will be when using the block smoother within a multigrid preconditioner. Therefore, adding the off-diagonal block C_2 to the block diagonal matrix Q in (7.13) may significantly reduce the number of linear iterations.

The Uzawa algorithm basically needs one more matrix-vector product by C_2 compared to the (indefinite) block diagonal preconditioner. These additional costs are negligible compared to the other operations such as finding (cheap) inverses of the diagonal blocks K and $\widetilde{S}$. With adding the off-diagonal coupling block, the smoother performs a one-way coupling in the sense that the Lagrange multiplier increments now depend on the current increment of the displacement degrees of freedom. This can significantly increase the quality of the block smoother. In each smoothing iteration one calculates a prediction for the displacement increments $\delta\boldsymbol{u}^{k+1}$, which are taken into account when solving for the corresponding Lagrange multiplier increments $\delta\boldsymbol{\lambda}^{k+1}$.

Algorithm 12: Uzawa smoother.

```
Procedure Uzawa($\alpha$, $k_{\max}$)
    Apply $k_{\max}$ smoothing sweeps with the Uzawa algorithm
    for $k \leftarrow 0$ to $k_{\max} - 1$ do
        Prediction step: determine prediction increments $\delta\boldsymbol{u}^{k+1}$ by solving approximately
        $\mathsf{K}\,\delta\boldsymbol{u}^{k+1} = \mathbf{r}_{\boldsymbol{u}}^{k} - \mathsf{K}\Delta\boldsymbol{u}^{k} - \mathsf{C}_1^{\mathsf{T}}\Delta\boldsymbol{\lambda}^{k}$
        Correction step: Solve approximately for $\delta\boldsymbol{\lambda}^{k+1}$
        $-\widetilde{S}\,\delta\boldsymbol{\lambda}^{k+1} = \mathbf{r}_{\boldsymbol{\lambda}}^{k} - \mathsf{C}_2\Delta\boldsymbol{u}^{k} + \mathsf{Z}\Delta\boldsymbol{\lambda}^{k} - \mathsf{C}_2\,\delta\boldsymbol{u}^{k+1}$
        Update step: update solution variables
        $\Delta\boldsymbol{u}^{k+1} \leftarrow \Delta\boldsymbol{u}^{k} + \alpha\,\delta\boldsymbol{u}^{k+1}$
        $\Delta\boldsymbol{\lambda}^{k+1} \leftarrow \Delta\boldsymbol{\lambda}^{k} + \alpha\,\delta\boldsymbol{\lambda}^{k+1}$
    end
    Return smooth solution vector
    return $\left(\Delta\boldsymbol{u}^{k_{\max}}, \Delta\boldsymbol{\lambda}^{k_{\max}}\right)$
```

Remark 7.2.7 (Error matrix). The error matrix for the Uzawa smoother is given by

$$\mathsf{E}_{\mathsf{UZ}} := A - Q = \begin{pmatrix} (1-\frac{1}{\alpha})\mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ (1-\frac{1}{\alpha})\mathsf{C}_2 & -\mathsf{Z}+\frac{1}{\alpha}\widetilde{S} \end{pmatrix}. \tag{7.16}$$

With $\alpha = 1$ and $\widetilde{S} = \mathsf{Z}+\mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}}$ it is easy to verify that the error matrix of the Uzawa smoother reduces to

$$\mathsf{E}_{\mathsf{UZ}} = \begin{pmatrix} 0 & \mathsf{C}_1^{\mathsf{T}} \\ 0 & \mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \end{pmatrix}. \tag{7.17}$$

Similar to the indefinite block smoother, it is not possible to replicate the contact constraints with the Uzawa block smoother, such that the second block row in the error matrix vanishes.

For a theoretical review of Uzawa like smoothers the reader is referred to Bramble et al. [34], Elman and Golub [60] or Zulehner [235]. In the following, two variants of block approximate smoothers are introduced which resemble the original block operator (7.2).

7.2.4.3. Braess–Sarazin smoother

Originally introduced for the Stokes problem by Braess and Sarazin [29], the Braess–Sarazin smoother belongs to the class of block approximate smoothers and is based on the approximation

$$Q := \begin{pmatrix} \alpha\widetilde{\mathsf{K}} & \mathsf{C}_1^\mathsf{T} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix} \approx \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^\mathsf{T} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix} \tag{7.18}$$

of the block preconditioning matrix Q in (7.12). Here, the parameter $\alpha > 0$ denotes a scaling parameter and $\widetilde{\mathsf{K}}$ describes an easy-to-invert approximation of K. In practice, one uses the diagonal of K as a cheap variant for the approximation $\widetilde{\mathsf{K}}$, i.e., $\widetilde{\mathsf{K}} = \mathrm{diag}(\mathsf{K})$.

Remark 7.2.8 (Error matrix). The error matrix for the Braess–Sarazin smoother is calculated by

$$\mathsf{E}_{\mathsf{BS}} := A - Q = \begin{pmatrix} \mathsf{K} - \alpha\widetilde{\mathsf{K}} & 0 \\ 0 & 0 \end{pmatrix}. \tag{7.19}$$

The error matrix in (7.19) reveals that the second block row in the blocked operator (7.2) is correctly retained in (7.18). This makes the Braess–Sarazin smoother a reasonable choice for constrained problems such as structural contact problems. By splitting (7.18) into

$$\begin{pmatrix} \alpha\widetilde{\mathsf{K}} & \mathsf{C}_1^\mathsf{T} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix} = \begin{pmatrix} \alpha\widetilde{\mathsf{K}} & 0 \\ \mathsf{C}_2 & -\mathsf{Z} - \frac{1}{\alpha}\mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^\mathsf{T} \end{pmatrix} \begin{pmatrix} I & \frac{1}{\alpha}\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^\mathsf{T} \\ 0 & I \end{pmatrix}, \tag{7.20}$$

the Braess–Sarazin algorithm can be defined on base of a prediction-correction scheme.

Algorithm 13: Braess–Sarazin smoother.

```
Procedure BraessSarazin($\alpha$, $k_{\max}$)
    Apply $k_{\max}$ smoothing sweeps with Braess–Sarazin algorithm
    for $k \leftarrow 0$ to $k_{\max} - 1$ do
        Prediction step: determine prediction $\Delta\boldsymbol{u}^{k+\frac{1}{2}}$ by calculating
        $\Delta\boldsymbol{u}^{k+\frac{1}{2}} = \Delta\boldsymbol{u}^k + \frac{1}{\alpha}\widetilde{\mathsf{K}}^{-1}\big(\mathbf{r}_{\boldsymbol{u}}^k - \mathsf{K}\Delta\boldsymbol{u}^k - \mathsf{C}_1^\mathsf{T}\Delta\boldsymbol{\lambda}^k\big)$
        Correction step: Solve approximately for $\delta\boldsymbol{\lambda}^{k+\frac{1}{2}}$
        $-\big(\mathsf{Z} + \frac{1}{\alpha}\mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^\mathsf{T}\big)\,\delta\boldsymbol{\lambda}^{k+\frac{1}{2}} = \mathbf{r}_{\boldsymbol{\lambda}}^k + \mathsf{Z}\Delta\boldsymbol{\lambda}^k - \mathsf{C}_2\Delta\boldsymbol{u}^{k+\frac{1}{2}}$
        Update solution variables
        $\Delta\boldsymbol{\lambda}^{k+1} \leftarrow \Delta\boldsymbol{\lambda}^k + \delta\boldsymbol{\lambda}^{k+\frac{1}{2}}$
        $\Delta\boldsymbol{u}^{k+1} \leftarrow \Delta\boldsymbol{u}^{k+\frac{1}{2}} - \frac{1}{\alpha}\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^\mathsf{T}\,\delta\boldsymbol{\lambda}^{k+\frac{1}{2}}$
    end
    Return smooth solution vector
    return $\big(\Delta\boldsymbol{u}^{k_{\max}}, \Delta\boldsymbol{\lambda}^{k_{\max}}\big)$
```

First, one solves for a prediction of the displacement variables $\Delta \boldsymbol{u}^{k+\frac{1}{2}}$ and a tentative increment for the Lagrange multipliers $\delta\boldsymbol{\lambda}^{k+\frac{1}{2}} := \Delta\boldsymbol{\lambda}^{k+\frac{1}{2}} - \Delta\boldsymbol{\lambda}^{k}$ using

$$\begin{pmatrix} \alpha\widetilde{\mathsf{K}} & 0 \\ \mathsf{C}_2 & -\mathsf{Z} - \frac{1}{\alpha}\mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k+\frac{1}{2}} - \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k+\frac{1}{2}} - \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{\boldsymbol{u}}^{k} \\ \mathbf{r}_{\boldsymbol{\lambda}}^{k} \end{bmatrix} - \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k} \end{bmatrix}. \tag{7.21}$$

Then, the corrected solution $\left(\Delta\boldsymbol{u}^{k+1}, \Delta\boldsymbol{\lambda}^{k+1}\right)$ is determined from

$$\begin{pmatrix} I & \frac{1}{\alpha}\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \\ 0 & I \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k+1} - \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k+1} - \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} = \begin{bmatrix} \Delta\boldsymbol{u}^{k+\frac{1}{2}} - \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k+\frac{1}{2}} - \Delta\boldsymbol{\lambda}^{k} \end{bmatrix}. \tag{7.22}$$

As one can easily see from the Braess–Sarazin algorithm given in Algorithm 13, the prediction step can be understood as one hard-coded sweep with a (damped) Jacobi iteration. In other words, the quality of the prediction for the displacement degrees of freedom is rather poor. Note that the scaling parameter α has a slightly different meaning than for other block smoothing methods. In the Braess–Sarazin method, it is used to weight the different summands in the approximate Schur complement $\mathsf{Z} + \frac{1}{\alpha}\mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}}$ relatively to each other.

7.2.4.4. SIMPLE variants

Originally introduced by Patankar and Spalding [154] the SIMPLE method is based on the approximate block factorization

$$Q := \begin{pmatrix} \mathsf{K} & 0 \\ \mathsf{C}_2 & -\widetilde{S} \end{pmatrix} \begin{pmatrix} I & \widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \\ 0 & \frac{1}{\alpha}I \end{pmatrix} = \begin{pmatrix} \mathsf{K} & \mathsf{K}\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & \left(1 - \frac{1}{\alpha}\right)\mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} - \frac{1}{\alpha}\mathsf{Z} \end{pmatrix} \approx \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & -\mathsf{Z} \end{pmatrix} \tag{7.23}$$

for (7.12). In (7.23) the $\widetilde{S}$ denotes an approximation of the Schur complement $S := \mathsf{Z} + \mathsf{C}_2\mathsf{K}^{-1}\mathsf{C}_1^{\mathsf{T}}$ with a cheap and easy-to-invert approximation $\widetilde{\mathsf{K}}$ of the block K. Equation (7.12) with Q from (7.23) can be reformulated as

$$\begin{pmatrix} \mathsf{K} & 0 \\ \mathsf{C}_2 & -\widetilde{S} \end{pmatrix} \begin{pmatrix} I & \widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \\ 0 & \frac{1}{\alpha}I \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k+1} - \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k+1} - \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{\boldsymbol{u}}^{k} \\ \mathbf{r}_{\boldsymbol{\lambda}}^{k} \end{bmatrix} - \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & \mathsf{Z} \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k} \end{bmatrix}, \tag{7.24}$$

which leads to a two step predictor-corrector scheme, i.e., solve first

$$\begin{pmatrix} \mathsf{K} & 0 \\ \mathsf{C}_2 & -\widetilde{S} \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k+\frac{1}{2}} - \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k+\frac{1}{2}} - \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{\boldsymbol{u}}^{k} \\ \mathbf{r}_{\boldsymbol{\lambda}}^{k} \end{bmatrix} - \begin{pmatrix} \mathsf{K} & \mathsf{C}_1^{\mathsf{T}} \\ \mathsf{C}_2 & \mathsf{Z} \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} \tag{7.25}$$

for an intermediate solution of the displacement variable $\Delta\boldsymbol{u}^{k+\frac{1}{2}}$ with a subsequent adaption of the Lagrange multipliers $\Delta\boldsymbol{\lambda}^{k+\frac{1}{2}}$ and then obtain the final solution by

$$\begin{pmatrix} I & \widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \\ 0 & \frac{1}{\alpha}I \end{pmatrix} \begin{bmatrix} \Delta\boldsymbol{u}^{k+1} - \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k+1} - \Delta\boldsymbol{\lambda}^{k} \end{bmatrix} = \begin{bmatrix} \Delta\boldsymbol{u}^{k+\frac{1}{2}} - \Delta\boldsymbol{u}^{k} \\ \Delta\boldsymbol{\lambda}^{k+\frac{1}{2}} - \Delta\boldsymbol{\lambda}^{k} \end{bmatrix}. \tag{7.26}$$

The full SIMPLE method is given in Algorithm 14.

Algorithm 14: SIMPLE smoother.

```
Procedure SIMPLE($\alpha$, $k_{\max}$)
    Apply $k_{\max}$ smoothing sweeps with SIMPLE algorithm
    for $k \leftarrow 0$ to $k_{\max} - 1$ do
        Prediction step: determine prediction $\Delta\boldsymbol{u}^{k+\frac{1}{2}}$ by solving approximately
        $\mathsf{K}\Delta\boldsymbol{u}^{k+\frac{1}{2}} = \mathbf{r}_{\boldsymbol{u}}^k - \mathsf{C}_1^{\mathsf{T}}\Delta\boldsymbol{\lambda}^k$
        Correction step: Solve approximately for $\delta\boldsymbol{\lambda}^{k+\frac{1}{2}}$
        $-\widetilde{S}\,\delta\boldsymbol{\lambda}^{k+\frac{1}{2}} = \mathbf{r}_{\boldsymbol{\lambda}}^k + \mathsf{Z}\Delta\boldsymbol{\lambda}^k - \mathsf{C}_2\Delta\boldsymbol{u}^{k+\frac{1}{2}}$
        Update step: update solution variables
        $\Delta\boldsymbol{\lambda}^{k+1} \leftarrow \Delta\boldsymbol{\lambda}^k + \alpha\,\delta\boldsymbol{\lambda}^{k+\frac{1}{2}}$
        $\Delta\boldsymbol{u}^{k+1} \leftarrow \Delta\boldsymbol{u}^{k+\frac{1}{2}} - \alpha\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}}\,\delta\boldsymbol{\lambda}^{k+\frac{1}{2}}$
    end
    Return smooth solution vector
    return $\left(\Delta\boldsymbol{u}^{k_{\max}}, \Delta\boldsymbol{\lambda}^{k_{\max}}\right)$
```

Remark 7.2.9 (Error matrix). The error for one sweep with the SIMPLE method is calculated by

$$\mathsf{E}_{\text{SIMPLE}} := A - Q = \begin{pmatrix} 0 & \mathsf{C}_1^{\mathsf{T}} - \mathsf{K}\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \\ 0 & -\mathsf{Z} - \mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} + \frac{1}{\alpha}\widetilde{S} \end{pmatrix}. \tag{7.27}$$

As one can see from (7.27), SIMPLE does not affect the terms that operate on the primary displacement variables, but it perturbs the Lagrange multipliers. With $\widetilde{S} = \alpha\mathsf{Z} + \alpha\mathsf{C}_2\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}}$ properly scaled, the error matrix reduces to

$$\mathsf{E}_{\text{SIMPLE}} = \begin{pmatrix} 0 & \mathsf{C}_1^{\mathsf{T}} - \mathsf{K}\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}} \\ 0 & 0 \end{pmatrix}. \tag{7.28}$$

That is, an appropriate approximation of the Schur complement S allows to exactly satisfy the contact constraints within one smoothing sweep.

The concrete choice for the approximation $\widetilde{\mathsf{K}}$ of the block K and the approximation $\widetilde{S}$ for the Schur complement operator gives rise to different variants of the SIMPLE method. Here, only variants are mentioned that are use later for the numerical examples.

SIMPLE: The classical SIMPLE method (cf. Patankar and Spalding [154], Patankar [155]) uses $\widetilde{\mathsf{K}} = \operatorname{diag}(\mathsf{K})$ as easy-to-invert approximation $\widetilde{\mathsf{K}}$ of K, together with $\widetilde{S} = \mathsf{Z} + \mathsf{C}_2\big(\operatorname{diag}(\mathsf{K})\big)^{-1}\mathsf{C}_1^{\mathsf{T}}$. The damping parameter α is chosen from the interval $(0, 1]$ and damps the update for the Lagrange multipliers (cf. Elman et al. [59]).

SIMPLEC: Variants of the SIMPLE method like SIMPLEC as introduced by Van Doormaal and Raithby [190] can be understood as an enhancement of the classical SIMPLE method. The general idea is to provide better approximations for the inverse of the block K. Instead of just using the diagonal of K for calculating the approximate inverse of K, the diagonal matrix containing the row sums of $\left|\mathsf{K}\right| = \left(|a_{ij}|\right)_{i,j=1,\dots,n_{\mathsf{K}}}$ is used. That is, $\widetilde{\mathsf{K}}$ is defined as

$$\widetilde{\mathsf{K}} = \operatorname{diag}\Big(\sum_{j=1}^{n_{\mathsf{K}}} |a_{ij}|\Big), \quad i = 1, \dots, n_{\mathsf{K}}. \tag{7.29}$$

The default choice for $\widetilde{S}$ is consequently $\widetilde{S} = \alpha \mathsf{Z} + \alpha \mathsf{C}_2 \widetilde{\mathsf{K}}^{-1} \mathsf{C}_1^{\mathsf{T}}$ with $\widetilde{\mathsf{K}}$ as defined in (7.29). So, the method is very similar to the classical SIMPLE method (cf. Elman et al. [59]).

7.2.5. Comparison of saddle point smoothing methods

7.2.5.1. Cheap variants of block smoothers

Based on the block splitting from (7.2) all block smoothing methods from Section 7.2.4 internally have to build the inverses of the matrix blocks on the diagonal. To keep the computational costs low, one does not solve for the block inverses exactly, but only apply a fixed number of smoothing sweeps with a relaxation-based method. The numerical examples in Section 7.3 show that such an approximation leads to efficient block smoothing methods. As a naming convention, the prefix "Cheap" is added to the name of the block smoothing method to indicate the usage of a cheap approximation for finding the inverse of the diagonal blocks. A more theoretical discussion on the mathematical consequences of approximations for the Schur complement S can be found in Zulehner [234].

7.2.5.2. Block smoothers for structural contact problems

In structural contact simulations the interesting but challenging part is the coupling of the different solid blocks at the contact interface. Mathematically, the contact problem is governed by the contact constraint equations. Since the coupling of the structural blocks takes place in the level smoother only, constraint smoothers (cf. Keller et al. [102]) are preferred, which put some special focus on the consideration of the constraint equations. In our case, the preferred choice are the block approximate smoothers such as the Braess–Sarazin and SIMPLE-based methods in Section 7.2.4, as they represent the contact constraints exactly.

In the Braess–Sarazin method, the approximation $\widetilde{\mathsf{K}} = \operatorname{diag}(\mathsf{K})$ is hard-coded with some scaling parameter $\alpha > 0$ and consistently used within the approximate Schur complement operator, which is defined by $\widetilde{S} = \mathsf{Z} + \frac{1}{\alpha} \mathsf{C}_2 \widetilde{\mathsf{K}}^{-1} \mathsf{C}_1^{\mathsf{T}}$. The scaling parameter α weights the different summands in the Schur complement operator $\widetilde{S}$, whereas in the SIMPLE algorithm the α can be understood as a pure damping parameter.

In contrary to the Braess–Sarazin method, the SIMPLE based methods keep the full K block whenever possible in the block factorization and use $\widetilde{\mathsf{K}}$ only where its inverse is required. Consequently, in our "cheap" variants of the SIMPLE method, more elaborate smoothing strategies can be used for the K block instead of a hard-coded Jacobi sweep. Therefore, one can think of the SIMPLE methods to allow for a more balanced quality of approximations for the displacement degrees of freedom and Lagrange multipliers for the contact constraints.

When using exact arithmetic, comparing the error matrices of the Braess–Sarazin smoother and the SIMPLE smoother shows that the Braess–Sarazin smoother preserves the contact constraints per default due to the fixed definition of the Schur complement operator. A similar behavior for the SIMPLE smoother can only be found when using a properly scaled approximation of the Schur complement (see (7.28) in Remark 7.2.9).

7.3. Numerical examples

7.3.1. Two solid bodies example

To study the effect of the different saddle point smoothers from Section 7.2.4 the geometric configuration from the two solid bodies example in Section 6.2.1 is used. The discretization is based on a $10 \times 10 \times 10$ mesh for each solid block with altogether 6000 displacement degrees of freedom and 300 Lagrange multipliers modeling the contact coupling constraints for the 10×10 slave nodes at the contact interface. The simulation runs 40 time steps with a time step size of $0.01s$. The nonlinear iteration inside each time step stops if either $\|\Delta \boldsymbol{u}\|_e < 10^{-8}$ holds for the Newton increment of the displacement degrees of freedom, or alternatively, if the conditions

$$\|\mathbf{r}_i^{\boldsymbol{u}}\|_e < 10^{-6} \wedge \|\mathbf{r}_i^{\boldsymbol{\lambda}}\|_e < 10^{-4} \tag{7.30}$$

hold for the nonlinear residuals $\mathbf{r}_i^{\boldsymbol{u}}$ and $\mathbf{r}_i^{\boldsymbol{\lambda}}$ in (7.1) after applying i Newton iterations. Within each Newton iteration the saddle point system (7.1) is solved iteratively using a preconditioned GMRES method with a 3 level AMG preconditioner as described in Section 7.2. Even though the problem size is rather small, 4 processors are used for all simulations to demonstrate that the algorithms also work in parallel. The iterative process for the linear system is considered to be converged, if it is

$$\left\|\frac{\mathbf{r}^k}{\mathbf{r}^0}\right\|_e < 10^{-8} \tag{7.31}$$

for the full residual vector $\mathbf{r}^k = \begin{bmatrix}\mathbf{r}^{\boldsymbol{u}} \\ \mathbf{r}^{\boldsymbol{\lambda}}\end{bmatrix}$ in the linear iteration step k. Here, the subscript i for the nonlinear Newton iteration is dropped.

Remark 7.3.1 (Stopping criteria). In this thesis the focus is on the behavior of the linear solver. Therefore a fixed stopping criterion is chosen in (7.31). This allows the comparison of different preconditioning techniques including their effect on the linear solution strategy. For real world problems, and especially for coupled multiphysics problems, the task of choosing appropriate stopping criteria for both the nonlinear and linear solver turns out to be quite challenging. Usually, one would choose a combination of different (length-scaled) norms for the partial vectors. In order to reduce the solver time in the inner linear solver, it is recommended to adapt the linear (relative) solver tolerance according to the residual norms of the outer nonlinear solver.

First, the effect of different saddle point smoothers on the number of linear iterations is explored. The results in Table 7.1 give the average number of linear iterations per time step for different combinations of the rotation angles α_y and α_z. The numbers in brackets denote the maximum number of linear iterations needed for solving one linear system during the full simulation. This way, one has a rough estimate of the variation of the number of linear iterations within the simulation. Table 7.1a shows the results for the indefinite block diagonal method as a multigrid level smoother (cf. Section 7.2.4.1). With a CheapUzawa block smoother, the number of iterations can be notably reduced compared to the indefinite block diagonal smoother as one can see from Table 7.1b. Furthermore, for the CheapUzawa smoother, the number of iterations does not show a dependence on the rotation angles α_y and α_z. Comparing the numbers from Table 7.1b with the results for the CheapBraessSarazin smoother in Table 7.1c, the CheapBraessSarazin smoother heavily suffers from the worse approximation of the displace-

(a) **Level smoother:** 3 Indef. BlockDiag. (0.7) with 1 SGS (0.7) + ILU(0)
Transfer operators: PA-AMG + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	80.8 (108)	80.0 (101)	78.5 (99)	82.1 (108)	83.4 (103)
	$\frac{1}{8}\pi$	79.6 (98)	79.7 (105)	79.9 (105)	80.2 (102)	82.0 (101)
	$\frac{1}{4}\pi$	71.4 (93)	70.5 (89)	67.5 (86)	72.8 (97)	68.9 (93)
	$\frac{3}{8}\pi$	80.0 (105)	74.9 (97)	67.4 (86)	70.8 (94)	81.5 (101)
	$\frac{1}{2}\pi$	83.2 (105)	78.5 (102)	68.3 (84)	77.8 (109)	81.4 (99)

(b) **Level smoother:** 3 CheapUzawa(0.7) with 1 SGS (0.7) + ILU(0)
Transfer operators: PA-AMG + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	26.2 (30)	26.0 (30)	24.6 (29)	24.6 (30)	26.0 (29)
	$\frac{1}{8}\pi$	26.8 (30)	25.8 (37)	25.3 (30)	24.7 (30)	27.3 (31)
	$\frac{1}{4}\pi$	25.9 (31)	25.1 (32)	25.3 (30)	28.7 (40)	26.0 (31)
	$\frac{3}{8}\pi$	25.9 (31)	25.0 (30)	24.9 (30)	25.5 (33)	25.3 (29)
	$\frac{1}{2}\pi$	26.1 (30)	25.4 (32)	25.5 (30)	26.8 (31)	26.0 (30)

(c) **Level smoother:** 3 CheapBraessSarazin(1.9) with ILU(0))
Transfer operators: PA-AMG + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	29.9 (37)	29.6 (36)	27.9 (33)	30.7 (38)	29.7 (37)
	$\frac{1}{8}\pi$	41.2 (59)	43.1 (68)	42.0 (58)	43.6 (63)	41.4 (59)
	$\frac{1}{4}\pi$	56.8 (82)	64.8 (86)	68.6 (95)	64.3 (86)	54.9 (75)
	$\frac{3}{8}\pi$	40.9 (60)	55.5 (74)	73.1 (102)	111.1 (141)	41.3 (61)
	$\frac{1}{2}\pi$	29.9 (37)	41.0 (55)	56.8 (79)	41.3 (61)	29.7 (36)

(d) **Level smoother:** 3 CheapSIMPLEC(0.5) with 1 SGS (0.7) + ILU(0)
Transfer operators: PA-AMG + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	20.5 (26)	19.3 (21)	18.8 (23)	19.3 (26)	19.3 (20)
	$\frac{1}{8}\pi$	20.1 (22)	19.7 (27)	20.0 (24)	19.8 (21)	21.2 (25)
	$\frac{1}{4}\pi$	20.1 (25)	19.9 (23)	20.1 (23)	22.0 (26)	20.8 (30)
	$\frac{3}{8}\pi$	19.7 (22)	19.8 (22)	19.8 (23)	20.0 (26)	19.7 (23)
	$\frac{1}{2}\pi$	19.3 (20)	19.6 (22)	20.4 (27)	20.7 (25)	20.5 (28)

Table 7.1.: Two solid bodies example – Average number of linear GMRES iterations per nonlinear iteration (over all 40 time steps) for different combinations of rotation angles α_y and α_z. As preconditioner a 3 level AMG method (PA-AMG + PA-AMG, minimum aggregate size: 6 nodes) is used with different level smoothers.

(a) **Level smoother:** 1 CheapSIMPLEC(0.7) with 1 SGS (0.7) + ILU(0)
Transfer operators: PA-AMG + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	37.5 (57)	35.6 (43)	35.2 (46)	36.1 (44)	35.7 (45)
	$\frac{1}{8}\pi$	36.4 (45)	37.6 (44)	38.5 (47)	36.3 (43)	38.1 (53)
	$\frac{1}{4}\pi$	39.1 (50)	37.8 (47)	37.0 (47)	39.2 (48)	37.0 (44)
	$\frac{3}{8}\pi$	36.3 (43)	37.0 (48)	37.8 (46)	37.2 (51)	36.4 (43)
	$\frac{1}{2}\pi$	36.2 (47)	38.0 (51)	39.8 (50)	36.9 (50)	36.6 (53)

(b) **Level smoother:** 1 CheapSIMPLEC(0.7) with 3 SGS (0.7) + ILU(0)
Transfer operators: PA-AMG + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	29.5 (41)	27.5 (34)	28.4 (36)	28.5 (37)	27.1 (36)
	$\frac{1}{8}\pi$	29.3 (37)	30.1 (40)	32.7 (40)	28.8 (35)	28.4 (38)
	$\frac{1}{4}\pi$	30.0 (35)	29.6 (37)	28.5 (37)	31.1 (38)	29.2 (36)
	$\frac{3}{8}\pi$	28.8 (36)	27.8 (34)	28.1 (34)	29.1 (39)	28.1 (34)
	$\frac{1}{2}\pi$	28.5 (38)	28.4 (34)	29.7 (40)	27.9 (35)	27.1 (35)

(c) **Level smoother:** 3 CheapSIMPLEC(0.7) with 3 SGS (0.7) + ILU(0)
Transfer operators: PA-AMG + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	15.5 (16)	17.0 (19)	16.9 (20)	15.3 (16)	15.2 (16)
	$\frac{1}{8}\pi$	16.2 (17)	16.1 (17)	16.3 (17)	15.9 (17)	16.1 (17)
	$\frac{1}{4}\pi$	16.0 (17)	16.0 (17)	16.0 (17)	15.6 (18)	15.6 (16)
	$\frac{3}{8}\pi$	15.7 (17)	15.9 (17)	15.9 (17)	15.5 (19)	15.6 (16)
	$\frac{1}{2}\pi$	15.6 (16)	15.8 (16)	15.7 (16)	15.4 (16)	15.1 (16)

(d) **Level smoother:** 3 CheapSIMPLEC(0.7) with 3 SGS (0.7) + ILU(0)
Transfer operators: SA-AMG(0.6) + PA-AMG

		α_y				
		0	$\frac{1}{8}\pi$	$\frac{1}{4}\pi$	$\frac{3}{8}\pi$	$\frac{1}{2}\pi$
α_z	0	11.6 (16)	11.5 (16)	11.5 (16)	11.6 (16)	11.6 (16)
	$\frac{1}{8}\pi$	12.5 (16)	12.3 (16)	12.5 (16)	12.9 (16)	12.7 (16)
	$\frac{1}{4}\pi$	14.0 (17)	13.8 (16)	14.0 (16)	14.3 (16)	13.6 (16)
	$\frac{3}{8}\pi$	12.4 (16)	13.6 (16)	14.1 (16)	13.4 (16)	12.3 (16)
	$\frac{1}{2}\pi$	11.6 (16)	12.7 (16)	13.8 (17)	12.4 (16)	11.6 (16)

Table 7.2.: Two solid bodies example – Average (maximum) number of linear GMRES iterations per nonlinear iteration (over all 40 time steps) for different rotation angles α_y and α_z. As preconditioner a 3 level AMG method (minimum aggregate size: 6 nodes) is used with different variants of CheapSIMPLEC.

ment degrees of freedom using one internal hard-coded Jacobi sweep (cf. Section 7.2.4.3). The resulting iteration numbers show an obvious dependency of the rotation angles. With a Cheap-SIMPLEC block smoother the number of iterations is lower than for the CheapUzawa smoother and independent from α_y and α_z when compared with the CheapBraessSarazin smoother (see Table 7.1d). So, the linear solver has some benefit from the two-way coupling of displacements and Lagrange multipliers within the AMG preconditioner. Compared to the Uzawa smoother, the additional computational costs are very low with only one additional matrix-vector product by $\widetilde{\mathsf{K}}^{-1}\mathsf{C}_1^{\mathsf{T}}$ per iteration. Therefore, CheapSIMPLEC is the preferred level smoother for our further experiments with some cheap approximations for the internal single fields using some sweeps with a (symmetric) Gauss–Seidel method or ILU.

Table 7.2 indicates how the number of CheapSIMPLEC coupling iterations and the quality of the single field smoothing methods within the CheapSIMPLEC smoother affect the number of linear iterations. Aside from the concrete parameter choices for the level smoother one can even further reduce the number of linear iterations with a reasonable transfer operator smoothing strategy (cf. Section 3.5) for the displacement block.

Remark 7.3.2 (Solver timings). By intention there are no solver timings given as the example is too small to perform reasonable measurements especially when using 4 processors for altogether only 6300 degrees of freedom.

The intention of this example is to compare typical saddle point smoothers within a full AMG preconditioner. One can observe the expected behavior that increasing the number of smoothing sweeps reduces the number of linear GMRES iterations. However, in practice, the variant with a smaller number of GMRES iterations may not always be the fastest method. This example shows that the proper choice of block level smoothing is essential for the overall performance of a saddle point multigrid method. The particular choice of the block smoothing method gives the user full control over the quality of the coupling with field-specific parameters and allows for fine-grained adaptions and problem-specific optimizations.

Remark 7.3.3 (Weak scaling). In this example a very simplified contact configuration has been chosen, where the active set of nodes in contact is not changing after first contact occurs. The intention was to study the effect of level smoothers to the results of the linear solver. From the mathematical point of view a weak scaling study would be interesting where the ratio of number of unknowns per processor is kept constant with an increasing problem size. However, it is hard to set up a reasonable example which allows drawing conclusions from, since the ratio of contact nodes and inner nodes is decreasing with uniform mesh refinement. With non-uniform mesh refinement one could keep the ratio of contact nodes and inner nodes constant, but would produce anisotropic meshes with all its consequences.

Anyway, with the experience from this example one can choose efficient level smoothers which provide results independent from the exact geometric configuration. In the next examples one can put some attention on effects for the linear solver caused by changes in the active set of contact nodes for larger problems.

7.3.2. 1000 rings example

The following example is meant to demonstrate the long term behavior of the linear solvers for a contact problem with multiple bodies. The considered setup consists of 1000 rings (Neo-Hookean material with $E = 210$, $\nu = 0.3$ and $\rho_0 = 7.83 \cdot 10^{-6}$). The rings initially are arranged in a rectangle as shown in Figure 7.3a. A gravitational force is inducing an acceleration in negative y-direction towards a rigid wall. The simulation runs for 4000 time steps with a time step size of $\Delta t = 0.0005s$. Figure 7.3 shows snapshots for different time steps. The discrete system has 110000 nodes for the rings, which correspond to 220000 degrees of freedom for the displacements.

In each time step, the nonlinear system is handled by a semi-smooth Newton method. As convergence criteria one chooses

$$\left\|\Delta \boldsymbol{u}\right\|_e < 10^{-8} \ \wedge \ \left(\left\|\mathbf{r}_i^{\boldsymbol{u}}\right\|_e < 10^{-8} \ \wedge \ \left\|\mathbf{r}_i^{\boldsymbol{\lambda}}\right\|_e < 10^{-6}\right). \tag{7.32}$$

Here, $\mathbf{r}_i^{\boldsymbol{u}}$ and $\mathbf{r}_i^{\boldsymbol{\lambda}}$ denotes the (nonlinear) residual for the displacement and Lagrange multiplier variables after i Newton iterations. Similarly, $\Delta \boldsymbol{u}$ denotes the solution increment for the displacement variables in the i-th Newton iteration.

The saddle point formulation is used for modeling the contact constraints. Therefore the effective size of the linear system is changing with the number of active nodes. Since the underlying contact configurations are changing drastically, it turns out to be a good example for testing the robustness of the preconditioners. A GMRES solver is applied for the linear systems with different variants of AMG preconditioners. The relative tolerance of convergence for the GMRES solver is set to

$$\left\|\frac{\mathbf{r}^k}{\mathbf{r}^0}\right\|_e < 10^{-8} \tag{7.33}$$

with $\mathbf{r}^k = \begin{bmatrix}\mathbf{r}^{\boldsymbol{u}} \\ \mathbf{r}^{\boldsymbol{\lambda}}\end{bmatrix}$ the full residual vector in the linear iteration step k. Again, the subscript i for the nonlinear Newton step is dropped.

The example is used to compare the results for the nested and the full multigrid approach for saddle point problems as introduced in Section 7.2.1. Table 7.3 gives an overview of the chosen preconditioner parameters for the level smoothers. For each class of multigrid preconditioners, only the variants are presented which give the best timings for our example and are able to accomplish the 4000 time steps of the full simulation.

The multigrid parameters are chosen to be the same for all preconditioner variants: the minimum size of the aggregates is set to 6 nodes for the 2D problem and the maximum coarse level size is set to 1000 degrees of freedom, which corresponds to a 3 level multigrid method. For the nested AMG approach with the SIMPLE based methods only an exact solve of the Schur complement equation using KLU leads to a robust method for the full 4000 time steps. For the full multigrid approach, 1 sweep with CheapSIMPLEC is used as level smoother, which internally applies 3 sweeps with symmetric Gauss–Seidel on the finest and inter-medium level and an ILU(0) on the coarsest level. The cheaper coarse level solver (ILU(0)) turns out to be sufficient with the full multigrid approach. So, compared to the nested approach, one can put more effort in a good prediction of the displacement variables on all multigrid levels.

For the full AMG variants different transfer operator strategies are compared, namely the non-smoothed (PA-AMG) transfer operators and the energy minimization approach with local damping parameters for transfer operator smoothing from Section 3.5.2, denoted by Emin. Local damping parameters allow for self-adapting optimal transfer operator smoothing which is probably more appropriate than one global damping parameter for such an example with drastic

Preconditioner type	
Full multigrid based methods	Nested multigrid based methods
PA-AMG (CheapSIMPLE) Transfer operators: PA-AMG Level smoother: 1 CheapSIMPLEC Level damping: 0.8 – Pred. smoother: 3 SGS (0.8) – Corr. smoother: ILU (0)	**CheapSIMPLE (PA-AMG)** Block prec.: 1 CheapSIMPLEC Block prec. damping: 0.8 – Pred. smoother: AMG – Transfer op.: PA-AMG – Level sm.: 1 SGS (0.8) – Corr. smoother: KLU
Emin (CheapSIMPLE) Transfer operators: Emin Level smoother: 1 CheapSIMPLEC Level damping: 0.8 – Pred. smoother: 3 SGS (0.8) – Corr. smoother: ILU (0)	**CheapSIMPLE (SA-AMG)** Block prec.: 1 CheapSIMPLEC Block prec. damping: 0.8 – Pred. smoother: AMG – Transfer op.: SA-AMG (0.8) – Level sm.: 1 SGS (0.8) – Corr. smoother: KLU

Table 7.3.: 1000 collapsing rings example – Different AMG variants.

changes in the contact interface. All the simulations have been run on 16 cores (spread over 2 Intel Xeon E5-2670 Octocore CPUs).

Figure 7.5a shows the accumulated number of linear iterations in each time step. One can see that the number of linear iterations is significantly lower for the full AMG variants than with the CheapSIMPLE based methods. This can be explained by the better approximation of the displacement degrees of freedom using 3 instead of 1 damped Gauss–Seidel sweeps. Looking at Figure 7.5a one finds the linear iterations for the CheapSIMPLE (PA-AMG) method to correlate well with the number of active nodes over the time steps, whereas the curves for the full saddle point AMG variants follow the long term behavior of the curve for the active nodes only. This might be a consequence of the fact that for the SIMPLE based method the contact constraints are only fulfilled on the finest level, whereas the full saddle point AMG methods try to satisfy the contact constraints on all multigrid levels by using SIMPLE based level smoothers. The number of nonlinear sweeps per time step is rather constant with 4 to 6 nonlinear iterations throughout the whole simulation. The accumulated timings for the solver time in Figure 7.5b are all relatively close. The data seems to be somewhat noisy due to the very high number of time steps. Furthermore, one should be aware of the fact that the linear system is rather small, such that the time measurements are not very accurate. Note that one has 4 to 6 nonlinear sweeps per time step, that is, one is solving 4 to 6 linear systems in approximately 3 seconds. So, the variability of the time measurements is quite large compared to the absolute solver timings, which are in the range of $0.5s$ per linear system. Anyway, one can clearly distinguish the timings

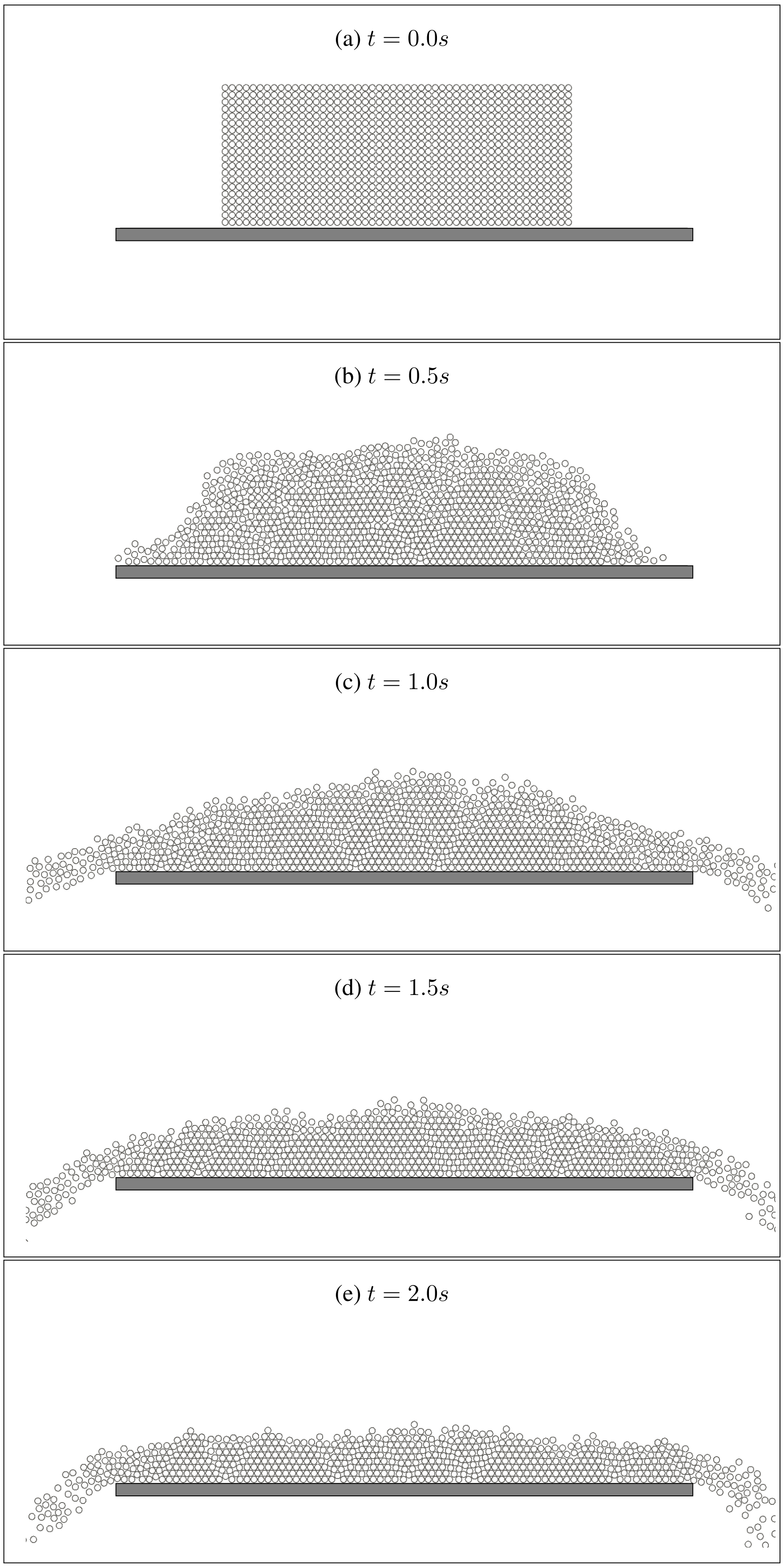


Figure 7.3.: 1000 collapsing rings example – Characteristic stages at different times.

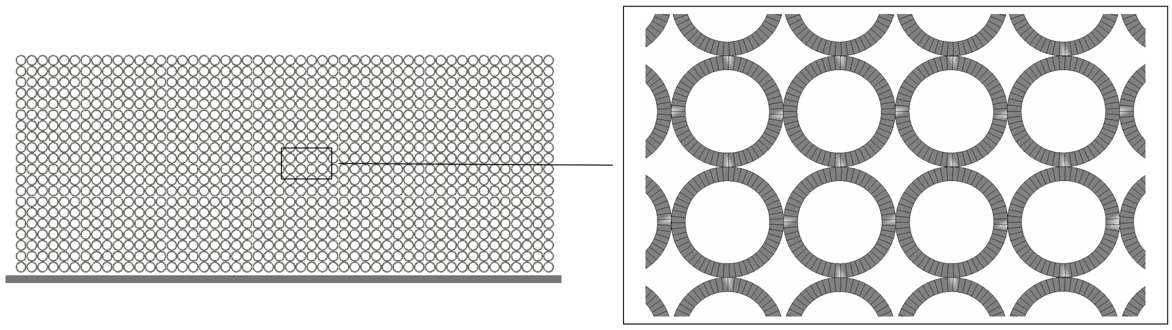


Figure 7.4.: 1000 collapsing rings example – Close-up view of initial stage.

for the CheapSIMPLE based variants in the light gray color and the AMG based variants denoted by the black color.

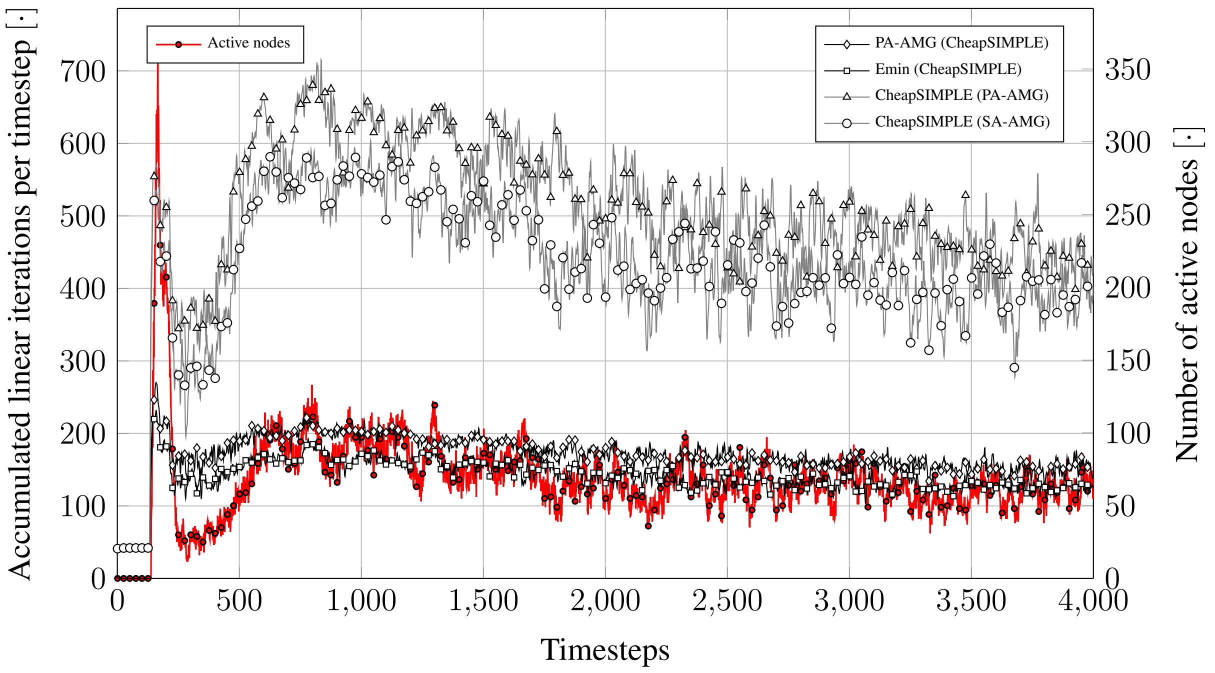
To get a better picture of the behavior in the solver timings, the accumulated savings in solver time are plotted over the full simulation in Figure 7.6. The PA-AMG (CheapSIMPLE) method serves as reference. One can see that the SIMPLE based methods need approximately 15 minutes more to solve all linear systems. Using smoothed transfer operators can save approximately 25 minutes of computational time compared to our reference method. However, all these different savings seem rather small compared to the overall solver time of approximately 6 hours.

Note that the solver time only includes the iteration phase of the GMRES method, but does not contain the setup phase of the preconditioner. The more interesting overall timings (setup phase and solving phase of the linear GMRES solver) can be found in Table 7.4. Please note that especially the timings for the setup phase are preliminary in the sense that the software has not been optimized for performance. The absolute timings can be considered to be somewhat smaller with optimized code variants. Nevertheless, the numbers confirm the findings from the Figures 7.6 and 7.5b, respectively. All methods are quite close with a slight advantage of the full AMG based methods.

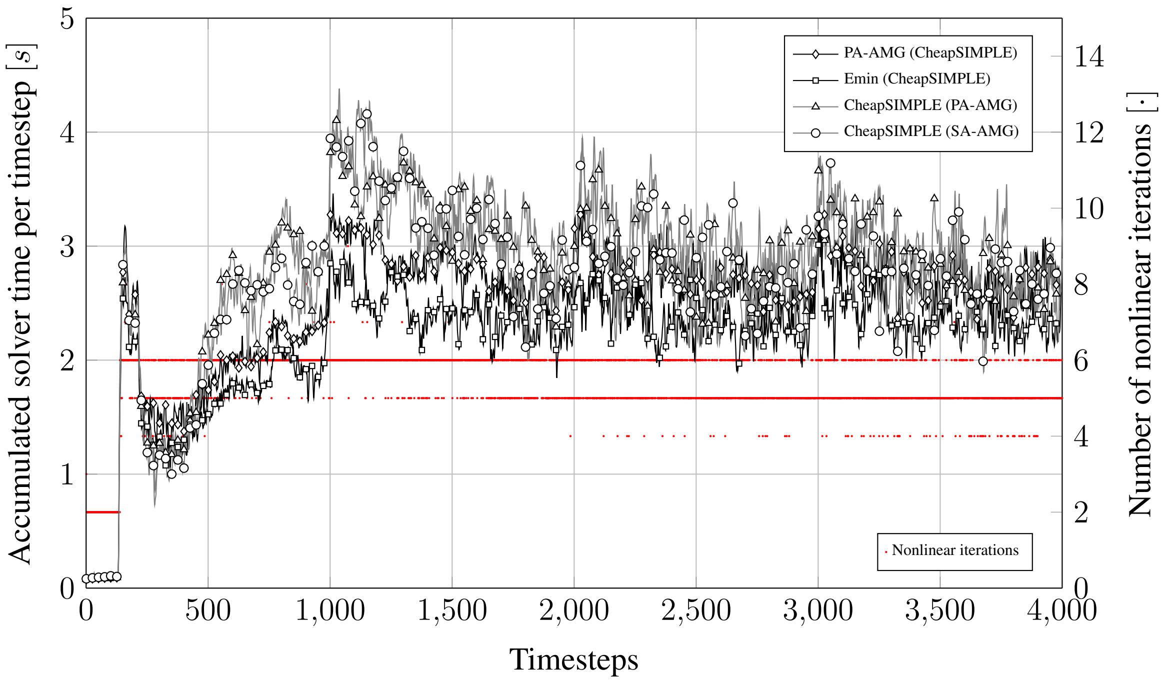
Method	Setup costs	Solver time	Overall solver time
PA-AMG (CheapSIMPLE)	11870	10013	21883
Emin (CheapSIMPLE)	12820	8679	21499
CheapSIMPLE (PA-AMG)	11730	11103	22833
CheapSIMPLE (SA-AMG)	12300	10763	23063

Table 7.4.: 1000 collapsing rings example – Exemplary timings in $[s]$ of the different preconditioning variants from Table 7.3 for the full simulation (4000 time steps).

Remark 7.3.4 (Further improvements). Similar to the strategies explained in Section 6.5.4 one can further reduce the timings. In particular, for reducing the setup costs one should reuse the full $P^{\boldsymbol{u}}$ and $R^{\boldsymbol{u}}$ blocks in (7.9), since they basically represent transfer operator blocks for pure structural problems. Then, one only has to rebuild the aggregates for the Lagrange multipliers to reflect changes in the contact set. Moreover, $\widehat{P}^{\boldsymbol{\lambda}}$ and $\widehat{R}^{\boldsymbol{\lambda}}$ could be reused as long as the contact set is not changing. For reducing the solver timings, one can think about adaptive stopping criteria.



(a) Accumulated number of linear GMRES iterations for all nonlinear iterations per timestep.



(b) Accumulated timings for the solution phase for all nonlinear iterations per timestep.

Figure 7.5.: 1000 collapsing rings example – Results for different AMG preconditioner variants.

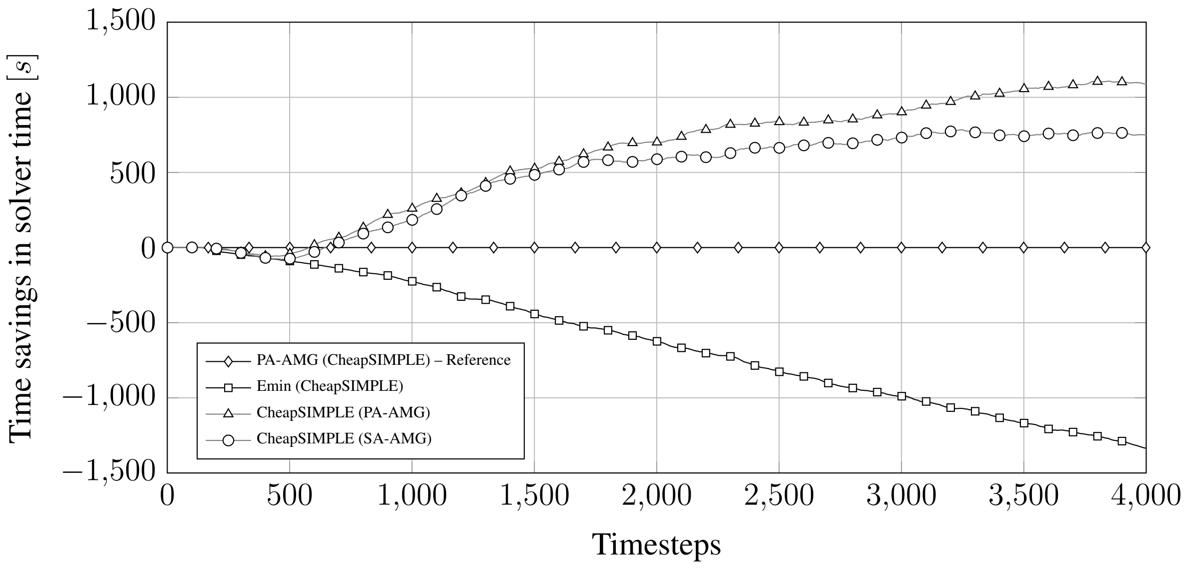


Figure 7.6.: 1000 collapsing rings example – Improvement of solver timings against reference method PA-AMG (CheapSIMPLE).

However, in the context of multiphysics problems, the adaptive design of stopping criteria is more complicated than for the single field case, as discussed above.

The idea of the next example is to compare the SIMPLE based and the full AMG based methods for significantly larger problems to get a better insight into the different behavior of preconditioner classes.

7.3.3. Two tori impact example

For studying the different AMG strategies on a larger example, the two tori impact example is reused with the exactly same problem configuration as in Section 6.5.3. The only difference is that now the original saddle point problems are solved instead of the condensed systems.

Consequently, the nonlinear stopping criteria are adapted to

$$\|\Delta \boldsymbol{u}\|_e < 10^{-7} \wedge \left(\left\| \frac{\mathbf{r}_i^{\boldsymbol{u}}}{\mathbf{r}_0^{\boldsymbol{u}}} \right\|_e < 10^{-8} \wedge \left\| \mathbf{r}_i^{\boldsymbol{\lambda}} \right\|_e < 10^{-4} \right). \tag{7.34}$$

Again, $\mathbf{r}_i^{\boldsymbol{u}}$ and $\mathbf{r}_i^{\boldsymbol{\lambda}}$ denotes the (nonlinear) residual for the displacement variables and Lagrange multipliers in the i-th Newton iteration. The quantity $\Delta \boldsymbol{u}$ describes the solution increment for the displacement variables only.

Remark 7.3.5 (Nonlinear convergence tolerances in comparison). In Section 6.5.3 the stopping criteria for the nonlinear solver have been chosen as

$$\|\Delta \boldsymbol{u}\|_e < 10^{-7} \wedge \left\| \frac{\mathbf{r}_i^{\boldsymbol{u}}}{\mathbf{r}_0^{\boldsymbol{u}}} \right\|_e < 10^{-6}, \tag{7.35}$$

which are based on similar ideas than the convergence criteria in (7.34). For the saddle point formulation a stronger tolerance is chosen for demonstration purposes. A relative residual is used of the displacement degrees of freedom combined with an absolute convergence criterion for the contact constraints. The stopping criterion for the solution increment $\Delta\boldsymbol{u}$ is identical. Of course, for the saddle point problem, one could add further additional convergence criteria (e.g., for the Lagrange multiplier increment $\Delta\boldsymbol{\lambda}$), but in our example this is not necessary. Here, one should point out that the nonlinear stopping criteria may have a different meaning depending on the (condensed versus saddle point) contact formulation. For example, for the choice of the solver tolerances one has to consider that there is usually no way to distinguish the structural equations and the contact constraints in the residual vector $\mathbf{r}_i^{\boldsymbol{u}}$ when using the condensed contact formulation with a standard implementation of a Newton algorithm.

For solving the linear Newton systems a preconditioned GMRES solver is applied. Primarily interested in the behavior of the linear solvers, one chooses a fixed tolerance for the linear solver, to be able to compare the different preconditioning variants without seeing effects of adaptive stopping criteria. The iterative process for the linear system is supposed to be converged if

$$\left\| \frac{\mathbf{r}^k}{\mathbf{r}^0} \right\|_e < 10^{-8} \tag{7.36}$$

holds for the full residual vector $\mathbf{r}^k = \begin{bmatrix} \mathbf{r}^{\boldsymbol{u}} \\ \mathbf{r}^{\boldsymbol{\lambda}} \end{bmatrix}$ in the linear iteration step k. The subscript i for the Newton iterations is dropped in the notation. This choice is identical to (6.25) from Section 6.5.3 for the same problem in condensed formulation. This allows for a (careful) comparison of the GMRES iterations in the condensed versus the saddle point case.

Table 7.5 gives an overview of the different tested preconditioner variants. It includes variants with the full multigrid approach, the nested multigrid approach and a SIMPLE based variant without multigrid at all. For the full AMG variants the transfer operators for the displacement blocks are varied. Particularly, non-smoothed transfer operators (PA-AMG) are compared with smoothed aggregation transfer operators (SA-AMG). In contrast to the previous example (see Section 7.3.2) where the transfer operators have been smoothed using Emin (cf. Section 3.5.2), for this example a standard SA-AMG approach seems to be sufficient, since the contact interface is not changing drastically. Local damping factors for transfer operator smoothing as used in Emin would come with additional computational costs without a significant effect on the linear iterations.

Figure 7.7a shows the accumulated number of iterations for solving all linear systems in one time step. Obviously, the SIMPLE based methods need more linear iterations than the AMG based methods. In this example there is nearly no difference between the non-smoothed transfer operator variant PA-AMG (CheapSIMPLE) and the smoothed transfer operator variant SA-AMG (CheapSIMPLE). Furthermore, there is no clear and obvious correlation between the number of linear iterations and the number of active nodes. Only for the SIMPLE based methods one can see a significant drop in the linear iterations for the last time steps in the simulation, which may correspond to the small number of nodes in contact.

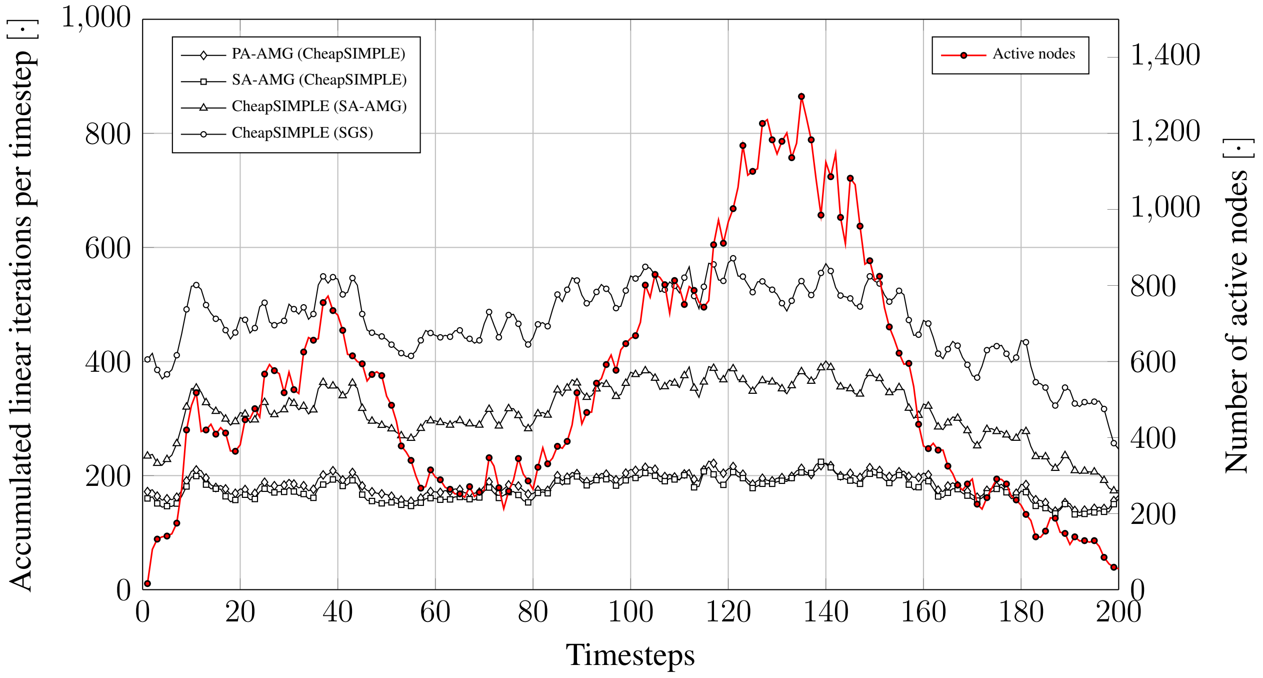
When looking at the corresponding solver timings over the time steps in Figure 7.7b, one finds the CheapSIMPLE (SA-AMG) method to be very close to the AMG based methods PA-AMG (CheapSIMPLE) and SA-AMG (CheapSIMPLE). For the AMG based methods one sweep with a CheapSIMPLEC method is applied on each level, which internally uses 1 sweep with a symmetric Gauss–Seidel iteration for the primary variable and 1 ILU sweep for the constraint

Preconditioner type			
Full multigrid based methods		SIMPLE based methods	
PA-AMG (CheapSIMPLE)		**CheapSIMPLE (SGS)**	
Transfer operators:	PA-AMG	Transfer operators:	–
Level smoother:	1 CheapSIMPLEC	Block prec.:	2 CheapSIMPLEC
Level damping:	0.8	Block prec. damping:	0.8
– Pred. smoother:	1 SGS (0.8)	– Pred. smoother:	3 SGS (0.8)
– Corr. smoother:	ILU (0)	– Corr. smoother:	ILU (0)
		CheapSIMPLE (SA-AMG)	
SA-AMG (CheapSIMPLE)		Block prec.:	2 CheapSIMPLEC
Transfer operators:	SA-AMG (0.4)	Block prec. damping:	0.8
Level smoother:	1 CheapSIMPLEC	– Pred. smoother:	AMG
Level damping:	0.8	– Transfer op.:	SA-AMG (0.4)
– Pred. smoother:	1 SGS (0.8)	– Level sm.:	2 SGS (0.8)
– Corr. smoother:	ILU (0)	– Corr. smoother:	ILU (0)

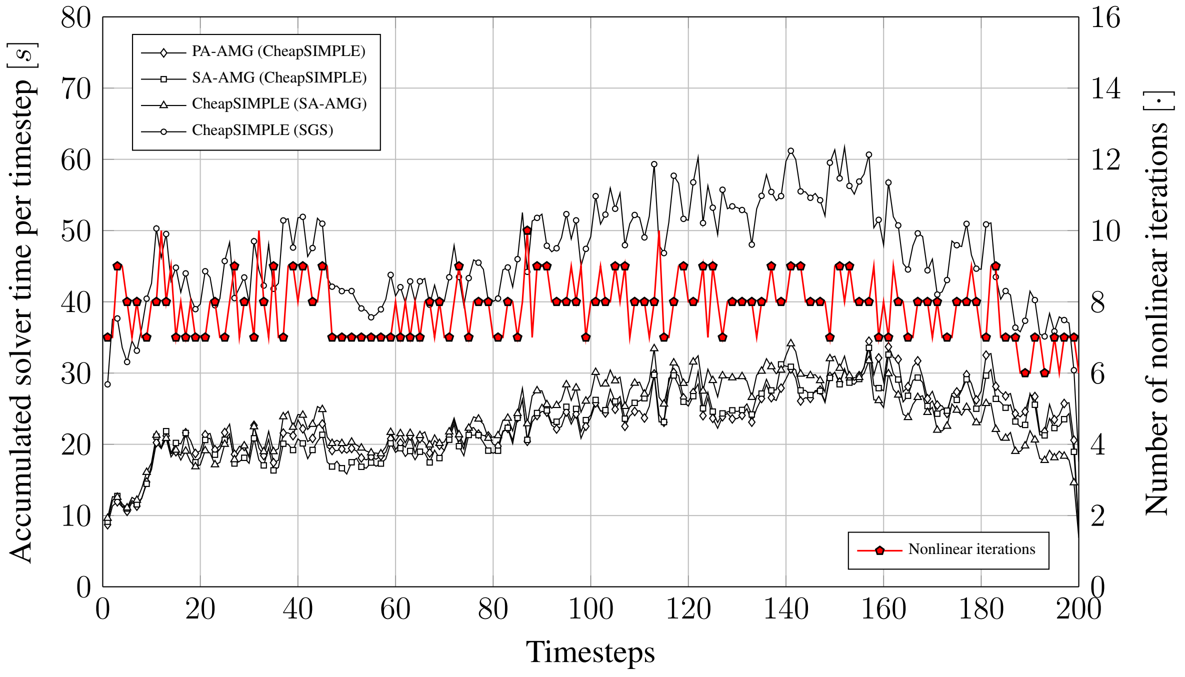
Table 7.5.: Two tori impact example – Different AMG variants.

equation. That is, quite a lot of time is invested in the coupling on all levels with the comparably expensive ILU method. In contrary to the AMG based method, the CheapSIMPLE (SA-AMG) method uses 2 sweeps with a CheapSIMPLE preconditioner for the coupling (on the finest level only). Internally, a 3 level AMG multigrid is used with 2 symmetric Gauss–Seidel sweeps for the level smoother and an ILU sweep for the constraint correction equation. These parameters have been found to result in a reasonably low number of linear iterations. For this example the experiment shows that the CheapSIMPLE (SA-AMG) method needs twice as many iterations as the SA-AMG (CheapSIMPLE) method, but the costs per iteration are only half of the costs of the SA-AMG (CheapSIMPLE). Nevertheless, the AMG based methods seem to have a small advantage, when the number of nodes in contact increases.

In Figure 7.7b one can also see an increase in the computational times of up to 50% over time despite the relatively constant number of linear iterations (cf. Figure 7.7a). Since all preconditioning variants show a similar behavior independent of the underlying methods, this is very likely to be an effect resulting from additional communication with the proceeding simulation time. Figure 7.8 shows the solver timings summed up over the full number of 200 time steps. Again, the PA-AMG (CheapSIMPLE) method serves as reference and is used to compare the other methods against it. First, the CheapSIMPLE (SGS) variant cannot compete with the other methods. All the AMG based methods and the SIMPLE method with an internal AMG method are rather close. Over the full simulation time, the difference of these methods is not more than 4 minutes in the solver time. Nevertheless, one finds the AMG based variants to perform better for the time steps with more nodes in contact.



(a) Accumulated number of linear GMRES iterations for all nonlinear iterations per timestep.



(b) Accumulated timings for the solution phase for all nonlinear iterations per timestep.

Figure 7.7.: Two tori impact example – Results for different saddle point preconditioner variants.

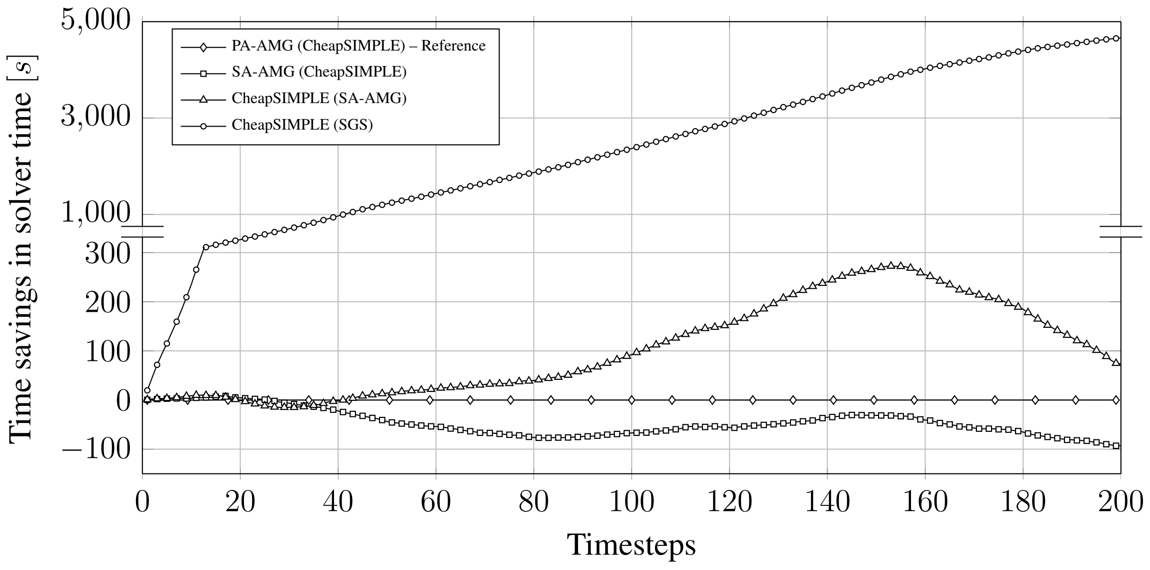


Figure 7.8.: Two tori impact example – Improvement of solver timings against reference method PA-AMG (CheapSIMPLE).

Last but not least, the overall timings for the linear solver in Table 7.6 are discussed. Except the CheapSIMPLE (SGS) variant, which is far away from the others, there is no clear winner. The setup costs are quite close, since for all methods the same transfer operators have to be built with only a small difference for smoothed versus non-smoothed transfer operators. Further code optimizations may allow to significantly reduce the setup costs. To reduce the solver timings, one can think about further strategies as already mentioned in Remark 7.3.4.

Method	Setup costs	Solver time	Overall solver time
PA-AMG (CheapSIMPLE)	11630	4658	16288
SA-AMG (CheapSIMPLE)	12250	4564	16814
CheapSIMPLE (SA-AMG)	12130	4731	16861
CheapSIMPLE (SGS)	10270	9320	19590

Table 7.6.: Two tori impact example – Exemplary timings in $[s]$ of the different preconditioning variants from Table 7.5 for the full simulation over 200 time steps.

Remark 7.3.6 (Comparison of condensed and saddle point formulation). In general one should be very careful with a direct comparison of the different solver strategies for the condensed formulation (cf. Chapter 6) and the saddle point formulation (cf. Chapter 7). But here, one can risk a short side glance to the timings from Table 6.8 in Section 6.5.3 as one solves the same example with (nearly) the same solver tolerances for the linear solvers, knowing that this is not sufficient for a full comparison as all timings are also influenced by other efffects (e.g., from the nonlinear solver, etc.). It is interesting to see, that in both cases the best variants are rather close. The saddle point AMG methods are a little bit more expensive in the setup costs, but give better results in the iteration phase.

CHAPTER 8

Summary and outlook

In summary, this thesis provides a piece of application-oriented fundamental research on novel multigrid concepts for problems typically arising in computational fluid mechanics and computational contact mechanics. In particular, a flexible AMG framework has been proposed which allows the design of problem-specific multigrid methods. To demonstrate both the use and the usage of the flexible multigrid framework, problem-specific enhancements for algebraic multigrid methods have been developed using concrete examples of challenging problem formulations of contact and flow problems.

After a brief review of the basic multigrid concepts in Chapter 2, the basic building blocks for an aggregation-based AMG algorithm are discussed in Chapter 3. Beside the methods and algorithms, the focus of this part of the thesis is on the framework for the setup of the multigrid hierarchies. In Chapter 4, a new contemporary transfer operator strategy serves as an example for a flexible algorithm which allows for problem-specific adaptions. Specifically, it is suitable for non-symmetric problems as they typically arise from problems with convection. The new scientific contribution concerning convection-dominated problems is the development of a novel transfer operator smoothing method which is based on advanced concepts such as a sparsity pattern strategy combined with mode preservation constraints and – in contrast to others – implies the non-symmetry of the linear operator in the smoothing process. It perfectly fits into the framework introduced in Chapter 3 and can be used with a Petrov–Galerkin approach building appropriately smoothed restriction operators for non-symmetric problems. In Appendix B the effect of non-symmetric transfer operator smoothing methods is exemplarily studied for a 1D convection-diffusion problem. The Petrov–Galerkin approach in combination with an appropriate transfer operator smoothing strategy, which considers the non-symmetry, has a significant effect on the character of the coarse level problems and may notably improve the convergence properties of the multigrid method in case of non-symmetric matrices resulting from convection-dominated problems.

Besides convection-dominated flow problems, contact mechanics using mortar finite elements is also a source for linear systems that are particularly challenging for iterative solvers and multigrid preconditioners due to the differing character of the structural equations and contact constraints. Both a condensed formulation and a saddle point formulation of the contact problem are considered with the corresponding differing matrix properties when developing adapted multigrid strategies within the proposed framework. First, in Chapter 6, the flexibility of the proposed AMG framework has been demonstrated by enabling multigrid preconditioners for problems arising from structural contact problems in condensed formulation. With the condensation of the Lagrange multipliers, one can avoid the saddle point structure of the linear systems. However, this comes along with other difficulties. Depending on the exact geometric configuration, the resulting linear systems are structurally non-symmetric and non-diagonally dominant due to the different coordinate systems used in the problem formulation. As it has been discussed in Chapter 6, only minor modifications of the aggregation-based AMG algorithms within the multigrid framework are necessary to solve large contact problems in condensed formulation with iterative methods using multigrid preconditioners. In particular, the usage of a cheap and efficient column permutation strategy has been proposed to overcome issues for the level smoothers caused by non-diagonally dominant problem matrices. Besides the permutation strategy, additional techniques have been introduced to improve the robustness. An observer mechanism keeps track of possibly problematic matrix properties allowing the algorithms to react on. The resulting methods have been found to be robust when applied to large numerical examples. In particular, the convergence behavior of the iterative solvers is independent from the exact geometric configuration when the proposed modifications are used within the AMG preconditioner. However, even though robust, one can observe a clear dependency of the linear iterations of the iterative solver from the number of active nodes that are currently in contact.

When solving the same structural contact problem using a different formulation, one obtains linear systems with different mathematical properties. In Chapter 7, saddle point AMG preconditioners have been introduced for the structural contact problems in the original saddle point formulation. Specifically, our contribution is a new interface aggregation method for Lagrange multipliers which perfectly fits into our flexible AMG framework. This interface aggregation strategy can be considered to be more natural, easier to implement and is shown to be cheaper than alternative methods proposed in the literature. This way, it is also shown that the presented framework can easily be extended to more general multigrid methods for block systems resulting from multiphysics problems. Even though the saddle point formulation is often considered to be hard to solve due to its characteristic block structure which requires special iterative methods for saddle point problems, it has the advantage that the meaning of the different primary and secondary variables in the block system is preserved over all multigrid levels. In this context, the correct choice of the block level smoothing method is essential for the overall performance of the saddle point multigrid method, since the block level smoother is the only place where the coupling between primary and secondary variables is performed. Numerical studies illustrate that the resulting saddle point AMG preconditioners are robust in the sense that all linear problems could be solved iteratively. However, depending on the choice of the level block smoother one finds the number of linear iterations to depend on the exact geometric configuration. From the results of the numerical experiments one finds cheap variants of saddle point smoothers based on Schur complement ideas to be a good choice as level smoothers, since they consistently consider the constraints within the iterative solving process. With SIMPLE based level smoothers the number of iterations of the linear solver shows a behavior independent of the geometric configuration and – in contrast to the condensed formulation – one cannot observe a clear dependency of the number of linear iterations from the number of active nodes in contact. With the saddle point formulation the user has the full control over the preconditioning quality by choosing appropriate level smoothers, but – on the other side – also has to deal with a large number of user parameters which may have some negative effect on the usability.

Flexibility also plays an important role in the context of the algorithmic design of AMG methods to meet today's requirements of state-of-the-art high-performance clusters. So, for the design of the software framework some special focus is put on addressing all requirements of a modern framework. Based on the TRILINOS libraries (cf. Heroux and Willenbring [89], Heroux et al. [90]) our AMG framework seems well prepared for the future to meet the challenges of new hardware platforms and software environments with a special focus on parallelization.

Altogether, different aspects of flexibility in an AMG framework have been highlighted with concrete examples arising both from computational contact mechanics and computational fluid mechanics dealing with different types of linear problems. The proposed methods in this thesis are a major step forward towards solving large problems that have hardly been accessible for iterative solvers until now.

Although substantial progress towards efficient iterative solvers and AMG preconditioners for specific problems such as contact problems in different formulations has been made, there is still room for improvements with regard to several aspects, which were only marginally covered or not addressed at all. In the following an outlook is given on selected promising research directions for the future.

First of all, one can think about many multigrid-specific enhancements. For example, a new aggregation concept can be considered which is based on a more hierarchical aggregation algorithm as, e.g., the aggregation routine described in Napov and Notay [140] leading to more aggressive coarsening. In context of aggressive coarsening the relation of transfer operator smoothing and level smoothing is an important research topic. It is intuitively clear that transfer operator smoothing in some sense incorporates the effect of level smoothers into the transfer operator basis functions. So, especially for large aggregates resulting from an aggressive coarsening an appropriate transfer operator smoothing may complement the level smoother (cf. Figure 4.4). Nevertheless, there are many open questions about reasonable transfer operator smoothing methods and the appropriate choice of complementary smoothers. In the context of non-symmetric problems the right choice of transfer operator smoothing strategies with the associated user-chosen parameters is still interesting for research (see also Appendix B).

Another interesting topic could be the development of application-specific sparsity patterns for transfer operators (see Section 4.4.4) in combination with the role of the near null space or mode preservation constraints. The right choice of near null space vectors is still an important problem and subject to further research. It is important to preserve a minimal set of all necessary low-frequency error modes on all multigrid levels which is particularly demanding for problems with some kind of discontinuities.

Of course, one can extend the framework in a straightforward way for other new applications, such as Thermo-Structure-Interaction (TSI) problems or Fluid-Structure-Contact-Interaction (FSCI) problems. Here, the goal should be to write general extensions independent of concrete applications, if possible. One should keep in mind, that the multigrid preconditioners serve as tool within an iterative solution process. So, a flexible multigrid framework is required for being able to quickly adapt and put together the corresponding block preconditioners.

Therefore, besides the purely multigrid-specific improvements there are more general research topics for the future. In this thesis the focus is on the design of multigrid preconditioners which are part of an iterative linear solver which again is part of an iterative nonlinear solution method. To improve the overall solution time one has to think of the solution process including nonlinear solver, linear solver and preconditioner as a whole. Here, the interplay of linear solver and nonlinear solver including sufficiently loose stopping criteria is essential. Whereas the extension of the multigrid preconditioners for general multiphysics problems is straightforward, the design of appropriate stopping criteria both for the linear and the nonlinear solver is highly challenging.

Appendix

A **B** **C** **D**

APPENDIX A

The preconditioned GMRES solver

The *Generalized Minimal Residual (GMRES)* method, originally introduced by Saad and Schultz [168], belongs to the class of Krylov subspace methods and is capable of solving non-symmetric problems. Details about current research on iterative solvers are far beyond the scope of this thesis. This appendix is not meant as replacement for a textbook on iterative solvers. The intention of this appendix is to briefly discuss the idea of the preconditioned GMRES method and give some literature for further reading. In between, the GMRES method is fairly popular for solving non-symmetric problems with quite a few ready-to-use implementations available. Even though aware of the extensions to the GMRES algorithm such as *flexible GMRES* (cf. Saad [169]) or *GMRESR* (cf. Van der Vorst and Vuik [189]) or alternative methods such as *BiCGstab* (cf. van der Vorst [188]), a classical (restarted) preconditioned GMRES method is used for all the numerical examples in this thesis. In this appendix, a rough outline of the GMRES algorithm is given following the explanations from Saad [170] which additionally considers many more aspects related to iterative methods and the GMRES algorithm in great detail.

A.1. The GMRES solver as Krylov subspace method

For introducing the GMRES method one first has to define the Krylov subspace $\mathcal{K}_k$.

Definition A.1.1 (Krylov subspace). A Krylov subspace of dimension k is defined as a subspace of the form

$$\mathcal{K}_k(\mathrm{A}, v) \equiv \operatorname{span}\{v, \mathrm{A}v, \mathrm{A}^2 v, \dots, \mathrm{A}^{k-1} v\} \tag{A.1}$$

for a regular matrix $\mathrm{A} \in \mathbb{R}^{n\times n}$ and a vector $v \in \mathbb{R}^n$.

With Definition A.1.1 one can briefly describe the idea of the GMRES method as follows: Given an initial solution vector x^0 and the right-hand side vector b in (1.1), the GMRES method iteratively builds solutions x^k by minimizing the residual norm $\left\|b - \mathrm{A}x^k\right\|_e$ over all vectors in the subspace $x^0 + \mathcal{K}_k(\mathrm{A}, r^0)$ where $r^0 = b - \mathrm{A}x^0$ denotes the initial residual vector. To find the approximate solution x^k from the affine subspace $x^0 + \mathcal{K}_k(\mathrm{A}, r^0)$, the condition

$$b - \mathrm{A}x^k \perp \mathrm{A}\mathcal{K}_k(\mathrm{A}, r^0) \tag{A.2}$$

is imposed. Assume $\{v_1, \ldots, v_k\}$ to be a set of orthonormal basis vectors spanning the Krylov subspace $\mathcal{K}_k = \mathrm{span}\{v_1, \mathrm{A}v_1, \ldots, \mathrm{A}^{k-1}v_1\}$ and let V_k denote the matrix with column vectors $v_1, \ldots, v_k$. Then, any vector x in the affine subspace $x^0 + \mathcal{K}_k(\mathrm{A}, r^0)$ can be written as $x = x^0 + V_k y$ with y a vector of size k. To minimize the residual norm over all vectors in $x^0 + \mathcal{K}_k(\mathrm{A}, r^0)$ one first needs the definition

$$J(y) := \|b - \mathrm{A}x\|_e = \left\|b - \mathrm{A}\big(x^0 + V_k y\big)\right\|_e . \tag{A.3}$$

Then, one can rewrite the residual as

$$\begin{aligned} b - \mathrm{A}x &= b - \mathrm{A}\big(x^0 + V_k y\big) \\ &= r^0 - \mathrm{A}V_k y \\ &= \beta v_1 - V_{k+1}\overline{H}_k y \\ &= V_{k+1}\big(\beta \boldsymbol{e}_1 - \overline{H}_k y\big), \end{aligned} \tag{A.4}$$

where $r^0 = \beta\, v_1$, $\overline{H}_k$ an (upper) Hessenberg matrix and $\boldsymbol{e}_1$ the first vector in the standard basis. A detailed definition of $\overline{H}_k$ can be found in Algorithm 15. Since the column vectors of V_{k+1} are orthonormal by definition, it follows

$$J(y) \equiv \left\|b - \mathrm{A}\big(x^0 + V_k y\big)\right\|_e = \left\|\beta \boldsymbol{e}_1 - \overline{H}_k y\right\|_e . \tag{A.5}$$

Minimizing (A.3) is equivalent to minimizing the rightmost expression in (A.5) such that the approximate x^k is given by $x^k = x^0 + V_k y_k$ with $y_k = \mathrm{argmin}_y \left\|\beta \boldsymbol{e}_1 - \overline{H}_k y\right\|_e$.

A.2. The preconditioned GMRES method

In this thesis, a GMRES method is used with right-preconditioning as an outer linear solver. The idea of preconditioning is briefly described in Section 1.2.3. Algorithm 15 gives a basic sketch of the GMRES algorithm with right-preconditioning. It is based on an Arnoldi procedure which generates an orthonormal set of basis vectors of the corresponding Krylov subspace $\mathcal{K}_k(\mathrm{A}, r^0)$. Note that the new variable $\widetilde{x}$ never needs to be invoked explicitly. Once the initial residual $r^0 = b - \mathrm{A}x^0 = b - \mathrm{A}W^{-1}\widetilde{x}^0$ is computed, all subsequent Krylov subspace vectors can be obtained without any reference to the new variable $\widetilde{x}$. With right-preconditioning, the Arnoldi process builds and orthogonal basis of the right-preconditioned Krylov subspace, which is spanned by the vector set $\left\{r^0, \mathrm{A}W^{-1}r^0, \ldots, \big(\mathrm{A}W^{-1}\big)^{k-1} r^0\right\}$.

The GMRES method as given in Algorithm 15 is by no means optimal. In practice one would replace the Modified Gram–Schmidt orthogonalization in the Arnoldi process by a more robust Householder algorithm. Furthermore, the algorithm does not provide the approximate x^k explicitly at each iteration step k which would be necessary to implement a convergence check (e.g., by testing the residual $r^k = b - \mathrm{A}x^k$). By smart internal transformations one can make use of the structure of the Hessenberg matrix $\overline{H}_k$ and introduce a convergence check at each linear iteration k. Details on a state-of-the-art implementation of the GMRES method can be found, e.g., in Saad [170]. Therein further improvements (such as restarting techniques as described in Remark A.2.1) are discussed, that are often used in practice. For the experiments in this thesis, the GMRES implementation from the AZTECOO package (Heroux [88], Tuminaro et al. [187]) is used which is contained in the Trilinos libraries (cf. Heroux and Willenbring [89], Heroux et al. [90]).

Algorithm 15: GMRES solver (cf. Saad [170, Algorithm 9.5])

```
Procedure GMRES (A, $x^0$, $b$)
    Calculate initial residual $r^0$
    $r^0 \leftarrow b - \mathrm{A}x^0$.
    Choose normalized initial vector for $\mathcal{K}_1$
    $\beta \leftarrow \|r^0\|_e$
    $v_1 \leftarrow \frac{r^0}{\beta}$
    Build orthonormal basis of Krylov subspace $\mathcal{K}_k$ using an Arnoldi procedure
    for $j \leftarrow 1$ to $k$ do
        $w_j \leftarrow \mathrm{A}W^{-1}v_j$
        for $i \leftarrow 1$ to $j$ do
            $h_{i,j} \leftarrow (w_j, v_i)_e$
            $w_j \leftarrow w_j - h_{i,j}v_i$
        end
        $h_{j+1,j} \leftarrow \|w_j\|_e$
        Check for Arnoldi procedure to stop
        if $h_{j+1,j} == 0$ then break
        Normalize basis vector $v_{j+1}$
        $v_{j+1} \leftarrow \frac{w_j}{h_{j+1,j}}$
    end
    Define $(k+1) \times k$ Hessenberg matrix $\overline{H}_k$
    $\overline{H}_k \leftarrow \{h_{i,j}\}_{1\leq i\leq k+1, 1\leq j\leq k}$
    Solve minimization problem
    $y_k \leftarrow \mathrm{argmin}_y \left\|\beta e_1 - \overline{H}_k y\right\|_e$
    Determine current approximate solution $x^k$
    $x^k \leftarrow x^0 + W^{-1}V_k y_k$
    return $x^k$
```

Remark A.2.1 (Restarted GMRES). The GMRES method is an iterative method which in contrast to other Krylov subspace methods also works for non-symmetric problems. However, the major drawback to GMRES is that the amount of work and storage required per iteration rises linearly with the iteration count. Since the costs of storing the Krylov subspace vectors will rapidly become prohibitive, the usual way to overcome this limitation is by restarting the iteration. There are different restarting strategies possible: the simplest strategy is to clear the accumulated number of basis vectors after a fixed number of iterations. This corresponds to the methods implemented in AztecOO and used for the examples in this work. In Gamnitzer [70, Appendix E.1] the corresponding algorithm for a right-preconditioned restarted GMRES algorithm is described in great detail. Alternatively, one can think of any other strategy to manage an appropriate limited number of basis vectors of the Krylov subspace (see, e.g., Saad [170] for some examples). Anyway, it is difficult to choose an appropriate number of basis vectors for the Krylov subspace to control the memory consumption without significantly disturbing the convergence. Unfortunately, there are no definite rules for a proper choice of the number of basis vectors.

Remark A.2.2 (Implementation of preconditioners). In practice, one does not explicitly build the preconditioning matrix W or its inverse. A close look at the Algorithm 15 reveals, that it is sufficient to implement the effect of applying the inverse of the preconditioner W to a vector. That is, one needs a routine that calculates y from $y = W^{-1}v$ for a given vector v.

APPENDIX B

1D prolongator smoothing

This section aims at throwing some light on the effect of different transfer operator smoothing strategies in the multigrid scheme. To keep things simple, only a small non-symmetric demonstration problem is considered that is solved using a two-level multigrid method.

B.1. Problem definition

Throughout the whole section one considers the 1D convection-diffusion equation

$$-\frac{\partial^2 u}{\partial x^2} + c\frac{\partial u}{\partial x} = b, \quad x \in \Omega, \tag{B.1}$$

for illustration. A finite difference discretization of (B.1) is used with central differences and proper boundary conditions, resulting in tridiagonal, non-symmetric linear systems. The convective parameter $c \in \mathbb{R}$ is assumed to be constant. Denoting the element length by h, the matrix stencil for the convection-diffusion problem is given in a short notation for the tridiagonal operator by $\frac{1}{h^2}\langle -1-\frac{ch}{2}, 2, -1+\frac{ch}{2}\rangle$ which corresponds to the Toeplitz matrix

$$\mathrm{A} = \frac{1}{h^2}\begin{pmatrix} \ddots & \ddots & \ddots & & & \\ & -1-\frac{ch}{2} & 2 & -1+\frac{ch}{2} & & \\ & & -1-\frac{ch}{2} & 2 & -1+\frac{ch}{2} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{B.2}$$

By dropping the scaling with the element length h and setting $\varepsilon := \frac{ch}{2} \in [0,1]$, one obtains the effective matrix stencil $\langle -1-\varepsilon, 2, -1+\varepsilon\rangle$.

In contrary to the convective parameter c, the parameter $\varepsilon \in [0,1]$ defines the *effective* convection. Choosing $\varepsilon = 0$ leads to the symmetric matrix stencil $\langle -1, 2, -1\rangle$, which is well known from the discretization of fully diffusive problems (of elliptic type). The choice $c = \frac{2}{h}$ would

	Coarse level matrix stencil		
Method	$\langle$ $a_{i,i-1}$	$a_{i,i}$	$a_{i,i+1}$ $\rangle$
SA-AMG	$\Big\langle$ $-1-\varepsilon+$ $2\omega-\frac{3}{2}\omega^2-\frac{1}{2}\omega^2\varepsilon^2,$	$2+$ $3\omega^2-4\omega+\omega^2\varepsilon^2,$	$-1+\varepsilon+$ $2\omega-\frac{3}{2}\omega^2-\frac{1}{2}\omega^2\varepsilon^2$ $\Big\rangle$
PG-AMG	$\Big\langle$ $-1-\varepsilon+$ $2\omega-\frac{3}{2}\omega^2+\frac{3}{2}\omega^2\varepsilon^2-2\omega\varepsilon^2,$	$2+$ $3\omega^2-4\omega-3\omega^2\varepsilon^2+4\omega\varepsilon^2,$	$-1+\varepsilon+$ $2\omega-\frac{3}{2}\omega^2+\frac{3}{2}\omega^2\varepsilon^2-2\omega\varepsilon^2$ $\Big\rangle$

Table B.1.: Matrix stencils for coarse level tridiagonal matrix $\mathrm{A}_{\ell+1} = (a_{i,j})_{\ell+1}$ for standard symmetric smoothed aggregation (SA-AMG) and Petrov–Galerkin smoothed aggregation (PG-AMG) approach. The ε parameter denotes the effective convection and ω the smoothing parameter for the prolongation and restriction operator.

yield pure upwinding. This corresponds to the parameter choice $\varepsilon = 1$, which describes the limit case of purely convective problems. Values larger than 1 for ε are not of physical interest. Without restriction of any kind, it is sufficient to consider $\varepsilon \geq 0$ only. For negative values of ε all considerations can easily be adapted by using A_ℓ^T instead of A_ℓ.

B.2. Simplified two-level method

For studying the effect of transfer operator smoothing strategies, it is sufficient to use a two-level method only.

B.2.1. Tentative transfer operators

Using optimal 3-point aggregates for the tridiagonal matrix the tentative transfer operators (see Section 3.4.1) are given by

$$\widehat{P} = \frac{1}{\sqrt{3}} \begin{pmatrix} \ddots & & & \\ & 1 & & \\ & 1 & & \\ & 1 & & \\ & & 1 & \\ & & 1 & \\ & & 1 & \\ & & & 1 \\ & & & 1 \\ & & & 1 \\ & & & & \ddots \end{pmatrix} \quad \text{and} \quad \widehat{R} = \widehat{P}^T. \tag{B.3}$$

As a result of the local QR-decomposition (cf. Figure 3.7), the transfer operator basis functions of $\widehat{P}$ are scaled by the square root of the aggregate size, which is 3 by construction in our example.

B.2.2. Smoothed aggregation transfer operators

The smoothed prolongation operator can be written as

$$P(\omega,\varepsilon) = \widehat{P} - \omega D^{-1}\mathrm{A}_\ell \widehat{P} = \frac{1}{\sqrt{3}} \begin{pmatrix} \ddots & & & & \\ & -\frac{\omega}{2}(-1+\varepsilon) & & & \\ & 1-\frac{\omega}{2}(1+\varepsilon) & & & \\ & 1 & & & \\ & 1-\frac{\omega}{2}(1-\varepsilon) & -\frac{\omega}{2}(-1+\varepsilon) & & \\ & -\frac{\omega}{2}(-1-\varepsilon) & 1-\frac{\omega}{2}(1+\varepsilon) & & \\ & & 1 & & \\ & & 1-\frac{\omega}{2}(1-\varepsilon) & -\frac{\omega}{2}(-1+\varepsilon) & \\ & & -\frac{\omega}{2}(-1-\varepsilon) & 1-\frac{\omega}{2}(1+\varepsilon) & \\ & & & 1 & \\ & & & 1-\frac{\omega}{2}(1-\varepsilon) & \\ & & & -\frac{\omega}{2}(-1-\varepsilon) & \\ & & & & \ddots \end{pmatrix}. \tag{B.4}$$

In (B.4) the $\omega \in [0,1]$ denotes the prolongation smoothing factor. The columns of (B.4) allow to derive the conditions

$$0 \le -\frac{\omega}{2}(-1+\varepsilon) \le 1 - \frac{\omega}{2}(1+\varepsilon) \le 1 \text{ and} \tag{B.5}$$

$$0 \le -\frac{\omega}{2}(-1-\varepsilon) \le 1 - \frac{\omega}{2}(1-\varepsilon) \le 1 \tag{B.6}$$

for smooth prolongation operator basis functions, which can be transformed to the effective conditions $0 \le \omega \le 1$ and $0 \le \varepsilon \le 1$. Note that $\omega = 0$ in (B.4) leads to the non-smoothed prolongation operator in (B.3).

B.2.3. Stencils of the coarse level operators

With the smoothed prolongation operator in (B.4) one can build the coarse level matrices and determine the corresponding matrix stencils. For restriction one can either use symmetric smoothed aggregation transfer operators with $R = P(\omega,\varepsilon)^T$ (cf. Section 4.2.1) or alternatively the Petrov–Galerkin smoothed aggregation approach for non-symmetric problems (see Section 4.2.2), where the transposed of A_ℓ is used for smoothing the restriction operator separately. In both cases the resulting coarse level operator $\mathrm{A}_{\ell+1} = R(\omega,\varepsilon)\mathrm{A}_\ell P(\omega,\varepsilon)$ is a tridiagonal matrix. The corresponding coarse level matrix stencils are listed in Table B.1. Figure B.1 shows the shape of the transfer operator basis functions both for the symmetric smoothed aggregation approach (SA-AMG from Section 3.5.1 with symmetric restriction from Section 4.2.1) and the Petrov–Galerkin smoothed aggregation approach (PG-AMG with non-symmetric restriction from Section 4.2.2) with varying transfer operator smoothing parameters ω and convection parameters ε. For the symmetric approach the restriction operator is just the transposed of P. Thus, the smoothed transfer operator basis functions for prolongation and restriction are identical. For the non-symmetric Petrov–Galerkin approach the transposed fine level matrix A_ℓ is used to generate the smoothed restriction operator basis functions. Therefore, the resulting basis functions for prolongator and

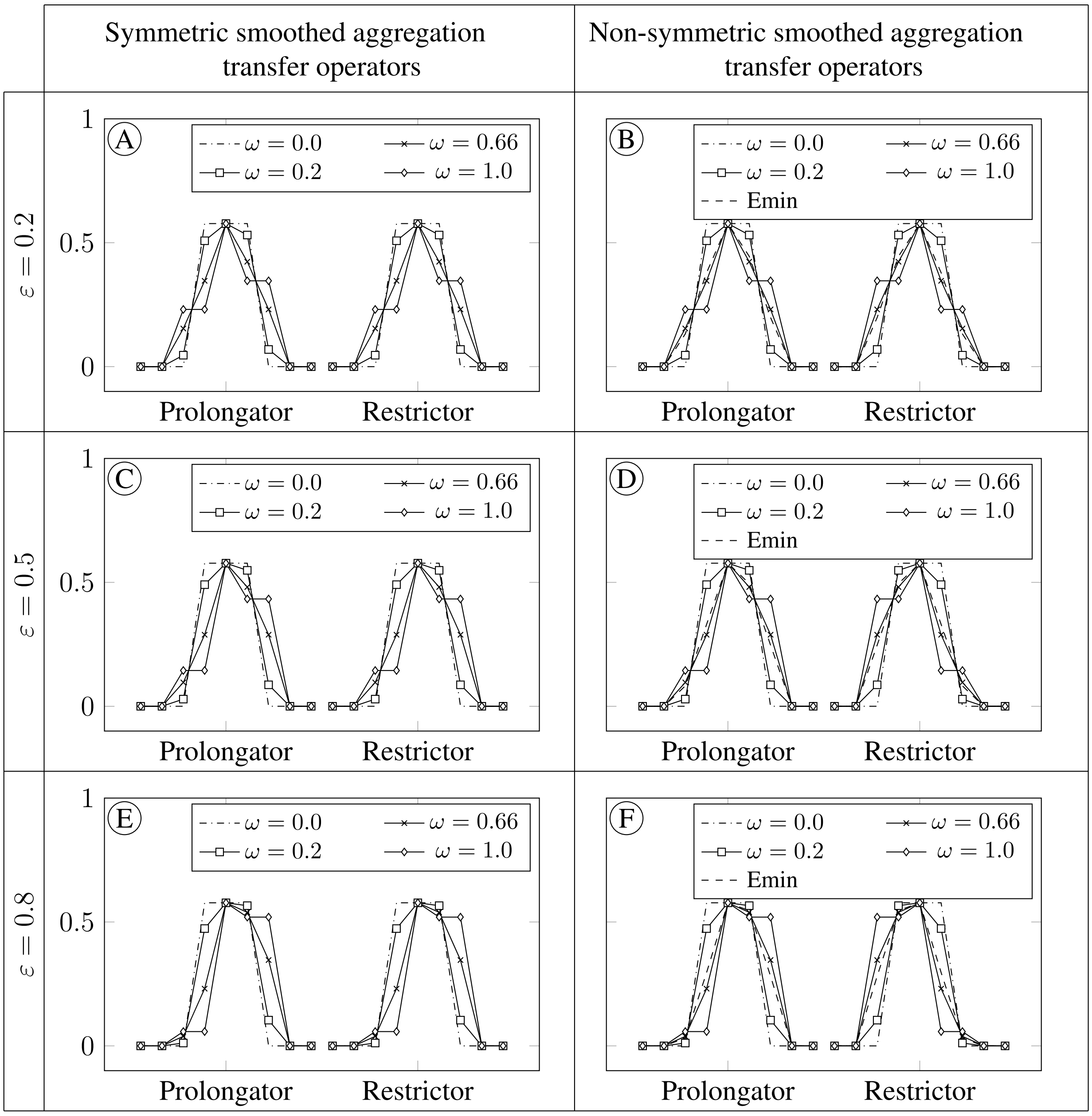


Figure B.1.: Transfer operator basis functions. The left column shows the symmetric smoothed aggregation transfer operator basis functions for different ω (see Figure B.2a). The right column shows the corresponding non-symmetric smoothed aggregation transfer operator basis functions (see Figure B.2b). The block rows allow a comparison of the transfer operator basis functions for an increasing convection parameter ε.

restrictor are not identical, as one can see from the given basis functions in the right column of Figure B.1.

Table B.1 shows the effect of the smoothed transfer operators on the coarse level matrix $\mathrm{A}_{\ell+1}$. The transfer operator smoothing method adds some symmetric values to the coarse matrix stencils. There are no additional non-symmetric additions to the matrix stencil. The symmetric smoothed aggregation method (SA-AMG) and the Petrov–Galerkin approach for non-symmetric problems (PG-AMG) exclusively differ in the mixed terms.

Remark B.2.1 ("Emin" method). The entry "Emin" in the right column represents the results for the transfer operator strategy introduced in Sala and Tuminaro [173], which is based on the Petrov–Galerkin approach for the restriction operators together with the local transfer operator smoothing strategy as briefly discussed in Section 3.5.2. However, in the case of Toeplitz matrices, it reduces to a special strategy of defining a global prolongation damping factor.

B.2.4. Diagonal dominance of the coarse level operators

From the matrix stencils in Table B.1, an expression for $\omega(\varepsilon)$ can be derived for which the coarse level matrix $\mathrm{A}_{\ell+1}$ remains diagonal dominant. The stencil for the interesting limit case is $\langle -2, 2, 0\rangle$. For the coarse level matrix generated by the SA-AMG approach this is equivalent to

$$-1+\varepsilon+2\omega-\frac{3}{2}\omega^2-\frac{1}{2}\omega^2\varepsilon^2 \overset{!}{=} 0 \quad\Leftrightarrow\quad \omega(\varepsilon)=\frac{2\pm\sqrt{-2+6\varepsilon-2\varepsilon^2+2\varepsilon^3}}{3+\varepsilon^2}. \tag{B.7}$$

A close analysis of the expression in (B.7) reveals that it has only a real solution for

$$\varepsilon \geq \frac{1}{3}+\frac{1}{3}\bigl(1+3\sqrt{57}\bigr)^{\frac{1}{3}}-\frac{8}{3\bigl(1+3\sqrt{57}\bigr)^{\frac{1}{3}}}\approx 0.36. \tag{B.8}$$

From (B.8) one finds that for $\varepsilon \lessapprox 0.36$ there is no $0 \leq \omega \leq 1$ which produces non-diagonally dominant coarse matrix stencils in $\mathrm{A}_{\ell+1}$. Figure B.2a shows all valid transfer operator smoothing parameters $\omega(\varepsilon)$ leading to diagonally dominant coarse level matrices.

For the PG-AMG approach one obtains the limit stencil $\langle -2, 2, 0\rangle$, if

$$-1+\varepsilon+2\omega-\frac{3}{2}\omega^2+\frac{3}{2}\omega^2\varepsilon^2-2\omega\varepsilon^2 \overset{!}{=} 0 \quad\Leftrightarrow\quad \omega(\varepsilon)=\frac{2+2\varepsilon\pm\sqrt{4\varepsilon^2+2\varepsilon-2}}{3(\varepsilon+1)}, \tag{B.9}$$

which exhibits real solutions for $\varepsilon \geq 0.5$. Figure B.2b shows the corresponding transfer operator smoothing parameters $\omega(\varepsilon)$ which lead to diagonally dominant coarse level matrices $\mathrm{A}_{\ell+1}$.

Note that the transition from the white to the gray area gives the transfer operator damping parameters $\omega(\varepsilon)$ which generate a coarse level operator with the limit stencil $\langle -2, 2, 0\rangle$ describing a purely convective problem.

For non-symmetric tridiagonal Toeplitz matrices $\mathrm{A}_\ell = (a_{i,j})_\ell$ as defined by the fine level matrix stencil $\langle -1-\varepsilon, 2, -1+\varepsilon\rangle$ as well as the coarse level stencils in Table B.1, it is very convenient to use the ratio of the off-diagonal and diagonal matrix entries to define a functional mapping s with

$$s : [0,1]^2 \to [0,0.5]\,,\ s(\omega,\varepsilon) = \left|\frac{a_{i,i+1}(\omega,\varepsilon)}{a_{i,i}(\omega,\varepsilon)}\right|. \tag{B.10}$$

A ratio of 0.5 corresponds to the symmetric stencil $\langle -1, 2, -1\rangle$ for $\varepsilon = 0$ and $\omega = 0$. In a similar way, the choice $s = 0$ defines the stencil $\langle -2, 2, 0\rangle$. Note that the value of the mapping s characterizes the degree of non-symmetry of the matrix. One can interpret the coarse matrix stencils from Table B.1 as disturbed matrix stencils $\langle -1 - \varepsilon_c, 2, -1 + \varepsilon_c\rangle$ with a disturbed effective convection parameter ε_c, which is different to ε. Consequently, the "disturbed" effective convection parameter ε_c corresponds to a disturbed convective parameter c_c which is not identical to the value of c in (B.1).

Figure B.3 illustrates how the different transfer operator smoothing strategies affect the ratio of the off-diagonal and diagonal entries of the coarse level matrix $\mathrm{A}_{\ell+1}$ depending on ω and ε. Depending on the damping parameter ω, the transfer operator damping strategy significantly increases or decreases the non-symmetry of the coarse level operator $\mathrm{A}_{\ell+1}$ compared to the fine level operator A_ℓ. The coarse level problem can therefore be interpreted as a discrete version of the coarse convection-diffusion equation with a convection parameter that is different to the convection parameter of the fine level problem. Looking at the Figure B.3 reveals that the convective part is less dominant on the coarse level, i.e., the level matrix $\mathrm{A}_{\ell+1}$ of the coarse problem is more symmetric, when an appropriate damping parameter ω is used. Note that for non-smoothed transfer operators with $\omega = 0$ the ratio of the off-diagonal and diagonal entries is preserved over all multigrid levels.

Of course, all these considerations are only valid for our chosen simplified example. But this example may provide some in-depth insight into the conceptual effect of transfer operator smoothing.

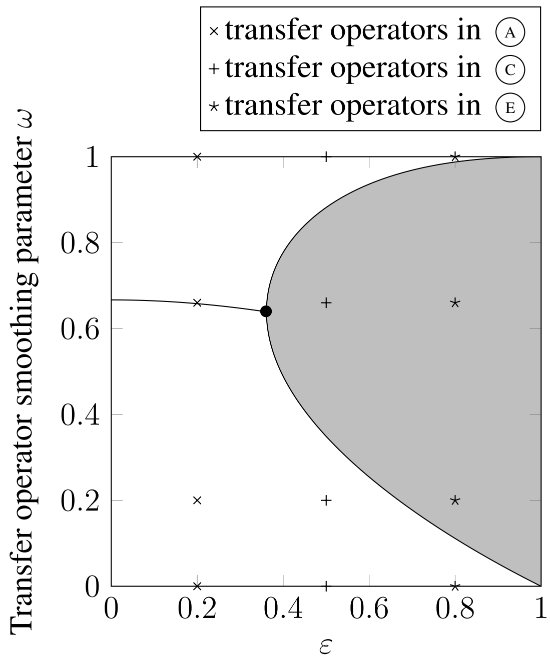
B.2.5. Optimal choice of transfer operator smoothing parameter

The next relevant question is which damping factor ω should be chosen for a given convection parameter ε. In the context of multigrid methods the coarse level matrix is used for some pre- and post-smoothing sweeps. A typical choice for multigrid level smoothers are relaxation-based methods, such as Jacobi iterations or Gauss–Seidel iterations (cf. Section 2.1). Therefore, an optimal choice of ω would try to generate optimal matrices for such level smoothing methods.

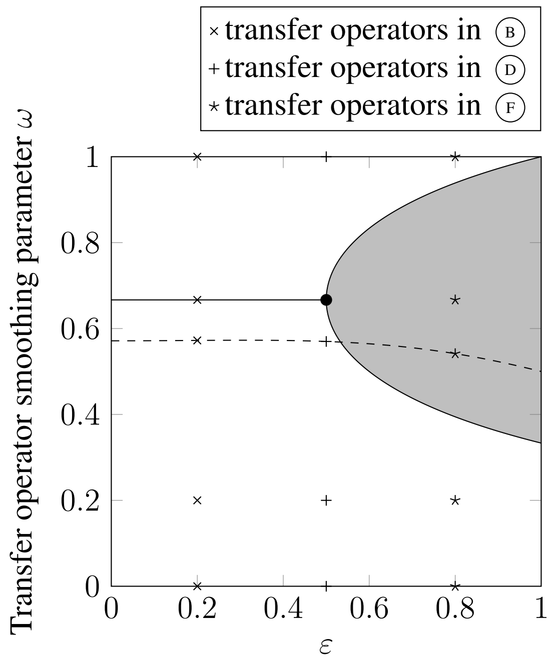
For reasons of simplicity the analysis is restricted to an undamped Jacobi iteration as level smoother. Since the coarse level matrices $\mathrm{A}_{\ell+1}$ with the matrix stencils from Table B.1 produce tridiagonal Toeplitz matrices, one can analytically determine the eigenvalue spectrum of the Jacobi iteration matrix $M := I - D^{-1}\mathrm{A}_{\ell+1}$ (see, e.g., Noschese et al. [141]).

Figure B.4 shows the maximum absolute eigenvalue of the Jacobi iteration matrix for given ω and ε for the coarse level operator $\mathrm{A}_{\ell+1}$.

Given a convection parameter ε, the choice $\omega(\varepsilon)$ is optimal for the Jacobi iteration, if the maximum absolute eigenvalue of the Jacobi iteration matrix M gets minimal. For the test example using the SA-AMG method the optimal choice would be $\omega_{opt} = \frac{2}{3+\varepsilon^2}$ for ε small enough to generate diagonally dominant matrices. Using the PG-AMG approach the optimal choice is $\omega_{opt} = \frac{2}{3}$ for $\varepsilon < 0.5$ independent of ε. The optimal choice for ω is also shown in the Figures B.2 by the solid lines.

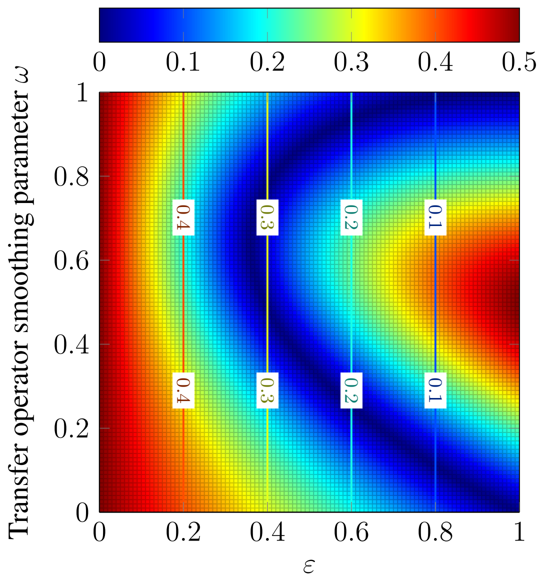


(a) Symmetric smoothed aggregation transfer operators

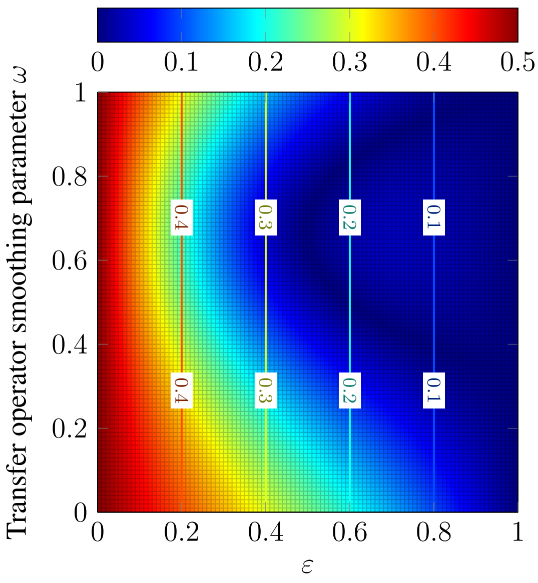


(b) Non-symmetric smoothed aggregation transfer operators

Figure B.2.: The gray area comprises all transfer operator smoothing parameters $\omega(\varepsilon)$ which lead to non-diagonally dominant coarse level matrices $\mathrm{A}_{\ell+1}$. The solid line describes the optimal $\omega(\varepsilon)$ which minimize the maximum absolute eigenvalue of the undamped Jacobi iteration matrix $M := I - D^{-1}\mathrm{A}_{\ell+1}$ of the coarse level matrix (see Section B.2.5 and Figure B.4). The dashed line indicates the $\omega(\varepsilon)$ chosen by the local transfer operator smoothing strategy from Section 3.5.2 with a Petrov–Galerkin approach for restriction. The different marks represent the $\omega(\varepsilon)$ variants shown in the Figure B.1 as well as Figures B.5 and B.6.



(a) Symmetric smoothed aggregation transfer operators



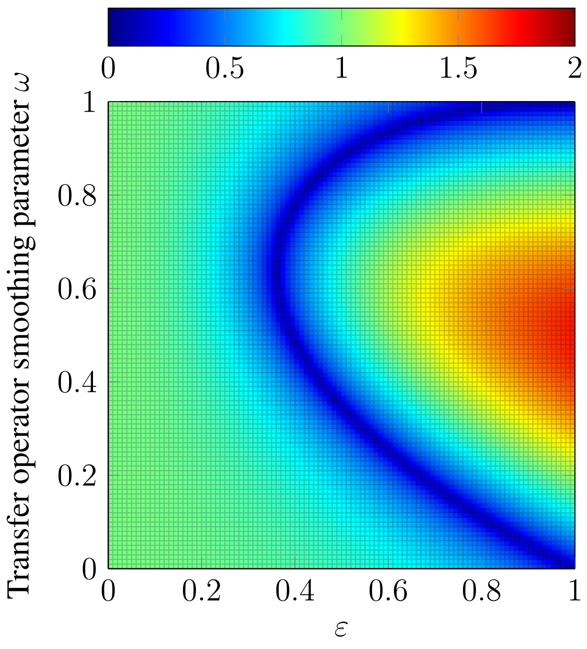
(b) Non-symmetric smoothed aggregation transfer operators

Figure B.3.: The colors illustrate the ratio of the coarse level matrix entries $|a_{i,i+1}| / |a_{i,i}|$ as a function of ε and ω. The ratio is in $[0, 0.5]$, where 0.5 represents the symmetric coarse level matrix stencil $\langle -1, 2, -1 \rangle$ and 0 the matrix stencil $\langle -2, 2, 0 \rangle$. The vertical contour plot lines denote the ratio of the fine level matrix entries $|a_{i,i+1}| / |a_{i,i}|$.

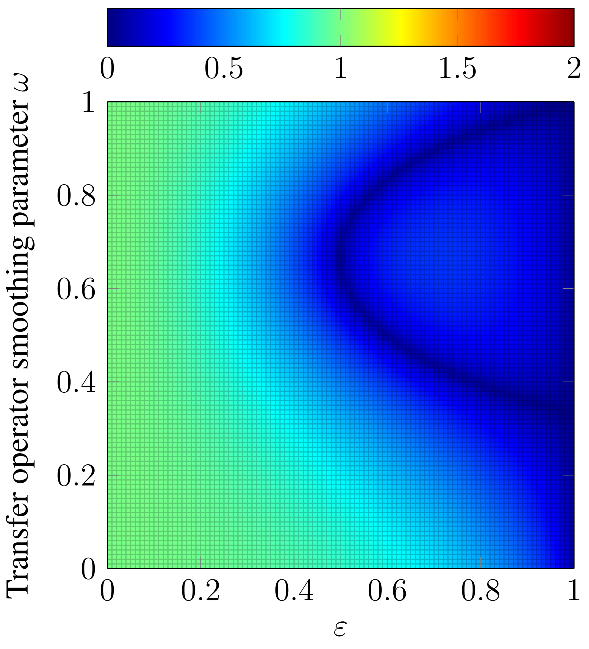
In Figure B.2b one can also find the prolongation damping parameters ω generated by the Emin method, that is introduced by Sala and Tuminaro [173] and often performs very well for practical non-symmetric examples using the Petrov–Galerkin approach together with local damping parameters (cf. Section 3.5.2).

B.2.6. Demonstration of a two-level multigrid method

To study the effect of the transfer operator smoothing strategies in a simplified multigrid context the 1D convection-diffusion problem in (B.1) is used on the domain $\Omega = [0, 1]$ with homogeneous Dirichlet boundary conditions. The discretization of the domain is based on a mesh with 30 fine level nodes (mesh size $h = \frac{1}{29}$). Furthermore, the convection parameter c in (B.1) is chosen such that the resulting linear operator A_ℓ has the effective matrix stencil $\langle -1 - \varepsilon, 2, -1 + \varepsilon \rangle$ away from the boundary nodes. By building optimal aggregates with three nodes per aggregate for the tridiagonal fine level matrix A_ℓ one obtains a two-level multigrid method with a tridiagonal 10×10 coarse level matrix $\mathrm{A}_{\ell+1}$. No pre- and post-smoothing is applied on the finest level. Figure B.5 shows the prolongated coarse level solution, if a direct solver is used on the coarse level. In the plots, one can easily find the 10 steps produced by the aggregates which are smoothed with an increasing transfer operator smoothing parameter ω. Comparing the symmetric smoothed aggregation approach with the Petrov–Galerkin approach for non-symmetric problems the results turn out to be very similar. Even if the convection gets more dominant for bigger ε, the Petrov–Galerkin approach seems to be only slightly better for this very simple example.



(a) Symmetric smoothed aggregation transfer operators



(b) Non-symmetric smoothed aggregation transfer operators

Figure B.4.: The colors indicate the maximum absolute eigenvalue of an undamped Jacobi iteration matrix $M := I - D^{-1}A_c$ as a function of ε and ω.

Things are different, if one solves the coarse level problem inexactly using 10 undamped Jacobi sweeps, as one can see in Figure B.6. For small convection parameters ($\varepsilon = 0.2$) both the symmetric as well as the non-symmetric smoothed aggregation method produce similar results for all tested transfer operator smoothing parameters ω. If the convection gets more dominant, the Petrov–Galerkin approach for non-symmetric systems performs much better compared to the symmetric case. Especially for the "optimal" smoothing parameter $\omega = 0.66$, the exact solution is well recovered by the prolongation of the inexact coarse level solution. In this simple case this even works for $\varepsilon > 0.5$, where the symmetric smoothed aggregation approach develops significant oscillations.

This example shows that the Petrov–Galerkin approach not only gives better results for non-symmetric problems, but is also more robust over a wider range of parameter values. The coarse level problems generated by the classical symmetric smoothed aggregation method applied to non-symmetric problems may severely disturb the multigrid solution process, if a wrong damping parameter ω is chosen. Even though one cannot generalize the findings for this example to more general problems, it might help to understand how the transfer operator smoothing method affects the coarse level operators in a multigrid setting. This may also help to choose appropriate damping parameters especially for highly convective problems. Non-smoothed aggregation (i.e., using non-smoothed transfer operators) turns out to be a safe fallback strategy, which has the interesting property of a constant ratio of the off-diagonal and diagonal entries in the matrix stencil that preserves the underlying physics (convection).

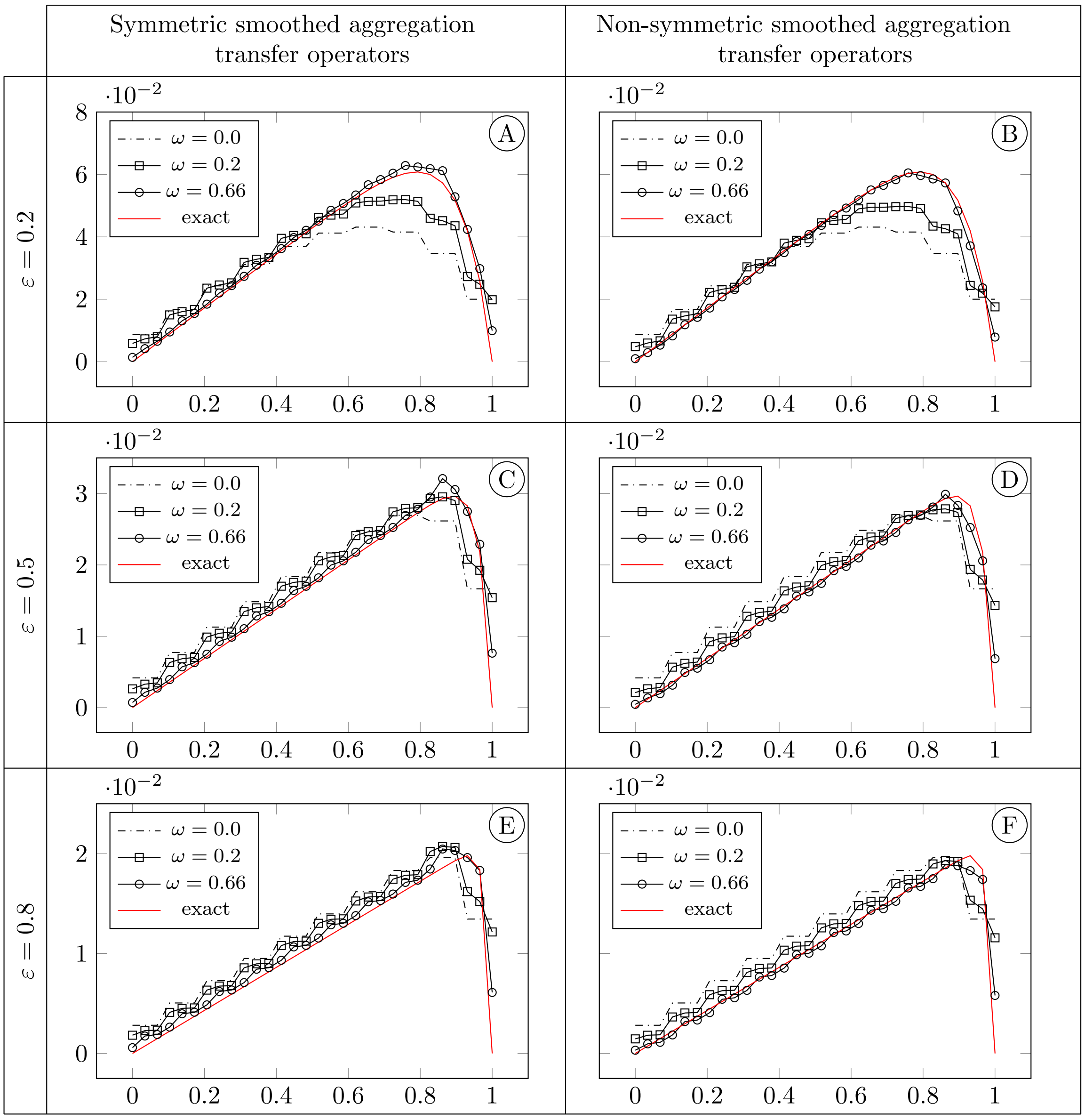


Figure B.5.: The plots visualize the prolongated coarse level solution of the linear system from Section B.2.6, when using a two-level solver. No pre- or postsmoothing is applied on the finest level and the coarse level problem is solved exactly. The left column shows the effect of symmetric smoothed aggregation transfer operators with different ω (see Figure B.2a). The right column shows the results for the corresponding non-symmetric smoothed aggregation transfer operators (see Figure B.2b). The block rows allow a comparison of the transfer operator basis functions for an increasing convection parameter ε.

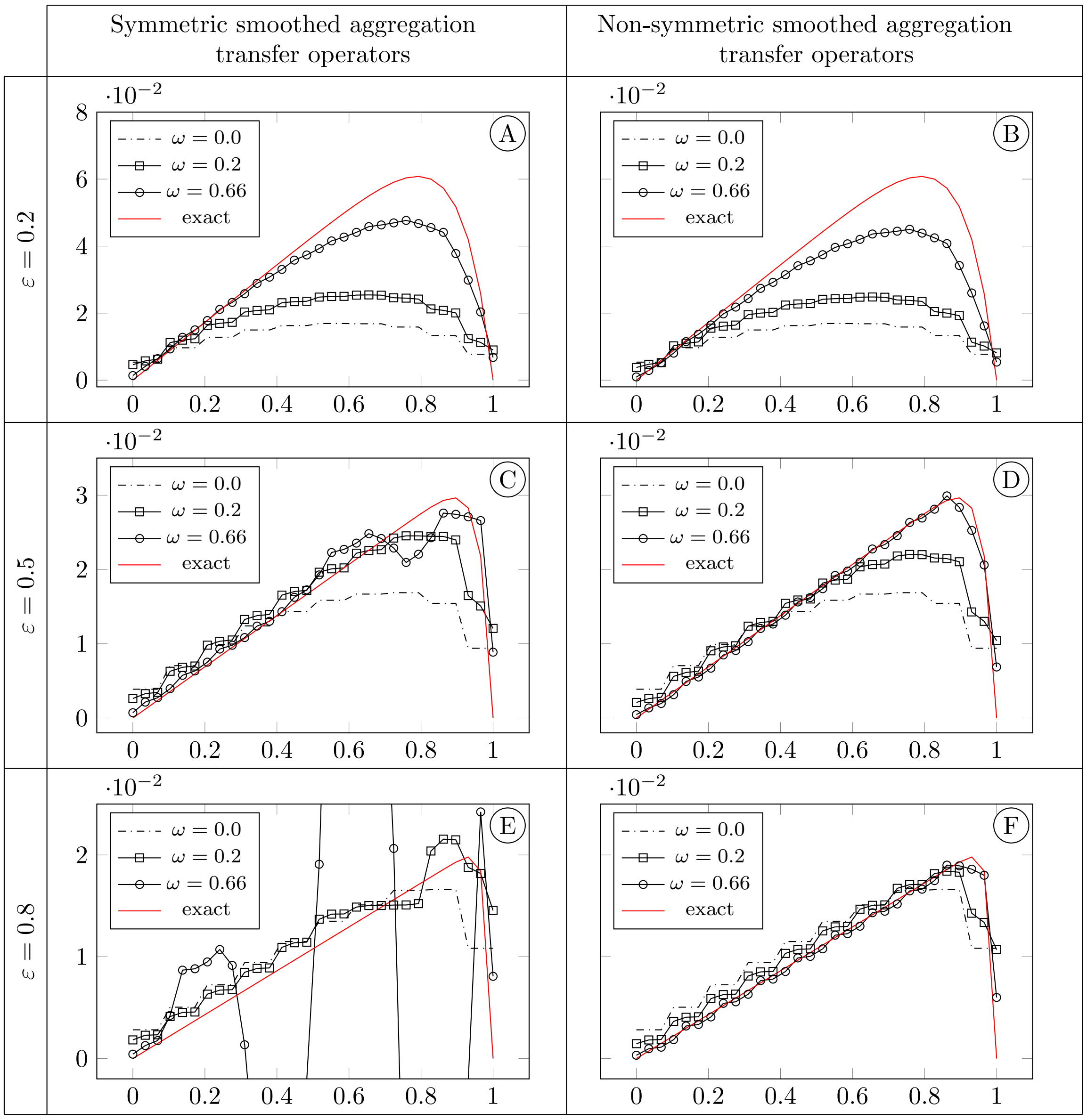


Figure B.6.: The plots visualize the prolongated coarse level solution of the linear system from Section B.2.6, when using a two-level solver. No pre- or postsmoothing is applied on the finest level. The coarse level problem is solved inexactly with 10 undamped Jacobi sweeps. The left column shows the effect for symmetric smoothed aggregation transfer operators with different ω (see Figure B.2a). The right column shows the results for the corresponding non-symmetric smoothed aggregation transfer operators (see Figure B.2b). The block rows allow a comparison of the transfer operator basis functions for an increasing convection parameter ε.

APPENDIX C

AMG & Dirichlet boundary conditions

To our knowledge there is no publication dealing with the peculiar details of Dirichlet boundary conditions (DBC) in context of iterative solvers and multigrid methods. Admittedly this is a topic, which is somewhat application-specific and may also often be considered to be “trivial”. However, in our opinion, there are some details that are more tricky that one might think in the first moment and therefore it is worth to be discussed in the following sections.

C.1. Algebraic representation of Dirichlet boundaries

There are different ways to consider Dirichlet boundary conditions in a linear system $\mathrm{A}x = b$. Of course, one can condense out Dirichlet boundary information from the linear system changing the size of the linear system. However, in practice, the condensation of the Dirichlet information is not very attractive, since changing the size of the linear system would be too expensive especially in a parallel implementation. In many standard software packages the Dirichlet information is not condensed out, but contained in the system matrices as zero rows with only one nonzero entry on the diagonal.

As a consequence, the Dirichlet information is contained in the linear system and affects the graph of the linear operator A. For general non-scalar problems one can distinguish different types of Dirichlet boundary conditions.

Definition C.1.1 (Partial Dirichlet node)**.** Assume that the linear operator A in (1.1) is defined by the entries a_{kl} where $k, l \in \{1, \ldots, n_\ell\}$. The node $i \in \big\{1, \ldots, m_\ell\big\}$ is a *partial* Dirichlet node, if there is at least one $k \in \big\{1, \ldots, n_\ell\big\}$ with $\mathrm{n}(k) = i$ and

$$a_{kl} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$

for $l = 1, \ldots, n_\ell$. The right-hand side vector b contains the Dirichlet values in the corresponding rows.

In a similar way one can define a full Dirichlet node, which can be understood as a special case of partial Dirichlet nodes.

Definition C.1.2 (Full Dirichlet node)**.** The node $i \in \{1, \ldots, m_\ell\}$ is a *full* Dirichlet node if and only if for all $k \in \{1, \ldots, n_\ell\}$ with $\mathrm{n}(k) = i$ the expression (C.1) holds for all $l = 1, \ldots, n_\ell$.

Note that for scalar problems with only one degree of freedom per node all Dirichlet boundary nodes are per definition *full* Dirichlet nodes. With the knowledge of the aggregation algorithms from Section 3.3 it is clear that such a representation of Dirichlet information also affects the aggregates within an aggregation-based multigrid method.

C.2. Dirichlet boundary conditions and relaxation-based methods

For simplicity here only the Jacobi method as introduced in Section 2.1.1 is exemplarily discussed, which is given by

$$x^{k+1} := x^k - \omega D^{-1}\big(\mathrm{A}x^k - b\big) = \big(I - \omega D^{-1}\mathrm{A}\big)x^k + \omega D^{-1}b \tag{C.2}$$

for the linear system $\mathrm{A}x = b$ from (1.1). Let the subscript $(\cdot)_\mathsf{D}$ denote the row indices with Dirichlet boundary conditions. One has to distinguish homogeneous and non-homogeneous Dirichlet boundary conditions.

Non-homogeneous DBC: For the more general case of non-homogeneous Dirichlet boundary conditions one finds: If $\omega = 1.0$, the Dirichlet rows of the iteration matrix $M := \big(I - \omega D^{-1}\mathrm{A}\big)$ are empty and it is $x_\mathsf{D}^{k+1} = b_\mathsf{D}$ for all k. However, if $k \neq 1.0$, this is not longer true. For appropriately chosen damping parameters ω only $x_\mathsf{D}^k \to b_\mathsf{D}$ holds for $k \to \infty$. So, for non-homogeneous Dirichlet boundary conditions one needs $\omega = 1.0$ to satisfy the Dirichlet boundary conditions exactly. Then, independent of the initial guess x_D^0 for the Dirichlet values, the Dirichlet information is correct after 1 sweep with a non-damped Jacobi iteration.

Homogeneous DBC: For the special case of homogeneous Dirichlet boundary conditions (that is $b_\mathsf{D} = 0$) one obtains some non-zero entries on the diagonal of $M_\mathsf{D} := \big(I - \omega D^{-1}\mathrm{A}\big)_\mathsf{D}$ only. As long as $x_\mathsf{D}^0 = 0$, i.e., the correct Dirichlet information is stored in the solution vector, the relaxation-based method does not disturb the correct Dirichlet values in the solution vector independent of ω.

Similar considerations can be made for other relaxation-based methods, which are quite often used as fine level smoothers for multigrid methods.

C.3. Dirichlet boundary conditions and Krylov subspace methods

Krylov subspace methods such as the CG method or the GMRES method that are often used as iterative solvers for linear systems, are known to fulfill Dirichlet conditions exactly, if the initial solution vector x^0 is already initialized with the exact Dirichlet values in the corresponding rows. Otherwise, Dirichlet conditions are only fulfilled up to the user-prescribed solver tolerance. Therefore, preconditioning methods within a Krylov subspace solver are not supposed to disturb correct Dirichlet information in the initial solution vector.

In practice, one does not solve the original linear system $\mathrm{A}x = b$ as defined in (1.1), but uses an incremental formulation. With the definition $r^k = \mathrm{A}x^k - b$ one can rewrite (1.1) as

$$\mathrm{A}\Delta x^{k+1} = \mathrm{A}\big(x^{k+1} - x^k\big) \overset{!}{=} b - \mathrm{A}x^k = -r^k. \tag{C.3}$$

Thus, one solves for the solution increment $\Delta x^{k+1} = x^{k+1} - x^k$.

Remark C.3.1 (Incremental formulation and Dirichlet boundary conditions). Let the superscript $(\cdot)_\mathsf{D}$ denote the indices with Dirichlet boundary conditions. Assuming that the initial guess x^0 for the solution vector contains the correct Dirichlet information, it follows $r^k_\mathsf{D} = 0$. If the initial guess in the Krylov iteration is chosen to be $\Delta x^0 = 0$, it is $\Delta x^0_\mathsf{D} = 0$, such that the correct Dirichlet information is not disturbed. The incremental formulations has the advantage that one has to deal only with homogeneous Dirichlet boundaries with all its consequences for preconditioners and smoothers (see, e.g., Section C.2).

C.4. Dirichlet boundary conditions and multigrid methods

In practice, when a multigrid method is used as preconditioner within an outer Krylov iterative solver together with an incremental formulation of the linear system, Dirichlet boundaries are usually not an issue. The outer linear solver will always satisfy (full and partial) Dirichlet boundary conditions at least up to the user-chosen solver tolerance, even if there is no special detection and handling of Dirichlet boundary nodes. Most often, it is fine to aggregate partial Dirichlet nodes together with usual inner nodes and the level smoother correctly handles all Dirichlet boundary information.

However, in rare situations with unusual solver settings and a weak solver tolerance, one might find the Dirichlet information to be not fulfilled exactly, which may be unacceptable or at least puzzling for the user. Especially when using a multigrid method as standalone solver or for special configurations of boundary conditions (e.g., 3D problems modeling pseudo 2D examples) or application examples (modeling of boundary layers in turbulent flows with low solver tolerance), it is important to put some more attention in the handling of boundary conditions. In the following different strategies for a proper handling of Dirichlet information is briefly discussed in context of multigrid methods.

C.4.1. Handling of full Dirichlet nodes

Full Dirichlet nodes can be understood as special case of partial Dirichlet nodes. In contrary to partial Dirichlet nodes, full Dirichlet nodes are detected by the aggregation algorithms from Section 3.3 as isolated nodes. Basically, one has two options to handle full Dirichlet nodes within a multigrid method. One can either drop them completely in the multigrid setup or keep them as single node aggregates transferring the Dirichlet information to the coarse levels. More advanced hybrid strategies are discussed in context of partial Dirichlet boundary conditions in the next sections.

Assuming that the solution vector x_ℓ contains the exact Dirichlet values in the corresponding rows after ν_1 pre-smoothing sweeps with the pre-smoother $\mathscr{S}$ (see, e.g., Algorithm 1), the best is to drop the Dirichlet nodes from the aggregation process. Then, the corresponding prolongation operator just contains zero rows in the rows associated with the Dirichlet boundary nodes, such that the the coarse grid correction has absolutely no influence to the corresponding entries in the solution vector x_ℓ. In Algorithm 1, the corresponding rows in e_ℓ are zero, such that the Dirichlet rows in x_ℓ are not disturbed. Now, it is up to the post-smoother to preserve the Dirichlet information correctly (see Section C.2).

Under certain circumstances, the pre-smoother might not be able to satisfy Dirichlet boundary conditions exactly (cf. Section C.2). Then, it could make sense in some cases not to drop the Dirichlet nodes from aggregation, but to consider the Dirichlet information in the coarse grid correction. To preserve the Dirichlet information on all multigrid levels one can use special single node aggregates. This way, the Dirichlet information can be transferred to the coarsest level and solved exactly using a direct solver. The disadvantage is that the coarsening rate may suffer from the additional small aggregates for the Dirichlet nodes.

C.4.2. Handling of partial Dirichlet nodes

It is more complicated for partial Dirichlet nodes, since it is not possible to detect them using the graph of the matrix A_ℓ. As long as the partial Dirichlet nodes are handled by the usual aggregation algorithms as described in Section 3.3, the (partial) Dirichlet information is lost on the coarser levels. The effect can be illustrated using the following minimal example for a two-level method (cf. Algorithm 1):

Example C.4.1. Let's have a look at the linear system

$$\left(\begin{array}{cc:cc} 1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ \hdashline 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{array}\right) x = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \text{ where } \widehat{x} = \begin{bmatrix} 0 \\ \frac{5}{2} \\ 4 \\ \frac{7}{2} \end{bmatrix} \tag{C.4}$$

is the exact solution. As indicated by the dashed lines in (C.4), 2 degrees of freedom per node are assumed, such that the first row represents a (partial) Dirichlet boundary condition. The initial guess for the solution vector is set to $x^0 = 0$, which contains the correct solution for the (homogeneous) Dirichlet row. For simplicity one neglects pre-smoothing on the finest level and uses the non-smoothed transfer operators from Section 3.4, that are given by

$$P := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } R = P^\mathsf{T}. \tag{C.5}$$

With the zero initial guess and without pre-smoothing, the right-hand side vector on the coarse level is calculated as $r_1 = Rr = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 \\ 4 \end{bmatrix}$. The coarse level system $P^\mathsf{T} \mathrm{A} P e_1 = r_1$ then is given by

$$\frac{1}{2}\begin{pmatrix} 3 & -2 \\ -3 & 4 \end{pmatrix} e_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 2 \\ 4 \end{bmatrix} \text{ with the exact solution } e_1 = \sqrt{2}\begin{bmatrix} \frac{8}{3} \\ 3 \end{bmatrix}. \tag{C.6}$$

Thus, the prolongated coarse level correction e_0 before post-smoothing can be calculated as

$$e_0 = Pe_1 = \begin{bmatrix} \frac{8}{3} \\ 3 \\ \frac{8}{3} \\ 3 \end{bmatrix}. \tag{C.7}$$

Note that the first entry in e_0 from (C.7) would disturb the Dirichlet information in the updated solution vector x_0. Obviously, the (partial) Dirichlet information is lost on the coarse level. In a multigrid method, again the post-smoother would be responsible to correct the wrong Dirichlet data (cf. Section C.2). Note that in this case a damped relaxation method would fail in correcting the Dirichlet data.

One should not drop (partial) Dirichlet nodes from aggregation, especially if there is no outer Krylov solver, which can still find a solution for the variables that are ignored by the multigrid preconditioner. To develop special strategies for handling (partial) Dirichlet information, the (partial) Dirichlet nodes have to be detected first. For extracting the Dirichlet node information one cannot use the (amalgamated) graph that is used for the aggregation, but has to use the full matrix A before amalgamation and mark the corresponding nodes to contain partial Dirichlet information. Then, one can think of different strategies:

C.4.2.1. Build single node aggregates for Dirichlet nodes

The first idea is to mark partial Dirichlet nodes and build single-node aggregates that are transferred one-to-one to the coarsest level. This way, the Dirichlet information is considered on all multigrid levels. However, the method has the significant disadvantage that the coarse level size is increased by the number of Dirichlet nodes, which deteriorates the coarsening rate. Therefore, this approach is not so interesting from the practical point of view. Furthermore, one has to keep in mind that single node aggregates need some special handling for problems with the number of near null space vectors larger than the number of degrees of freedom per node (cf. Remark 6.3.2 in Section 6.3.1).

C.4.2.2. Build aggregates from Dirichlet nodes

Instead of building single node aggregates, one can aggregate Dirichlet nodes of “same type” with the same right-hand side value into special aggregates. Two (partial) Dirichlet nodes are called to be of the “same type”, if they have Dirichlet boundary conditions for the same degrees of freedom (e.g., for the displacements in x-direction only). This is a good option for the incremental formulation with homogeneous Dirichlet boundaries only, since the number of Dirichlet aggregates is naturally limited to 2 in 2D and 6 in 3D (representing all possible combinations of Dirichlet and non-Dirichlet rows in a partial Dirichlet node, excluding full Dirichlet nodes).

Example C.4.2. Let's assume that one has again 2 degrees of freedom per node and the linear system is given by

$$\left(\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 3 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ \hline 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{array}\right) x = \left[\begin{array}{c} 0 \\ 1 \\ \hline 0 \\ 2 \\ \hline 2 \\ 2 \end{array}\right]. \tag{C.8}$$

As one can see from (C.8), rows 1 and 2 as well as rows 3 and 4 represent two partial Dirichlet nodes of same type, where the first degree of freedom per node (rows 1 and 3) are fixed with a (homogeneous) Dirichlet condition, whereas the second degree of freedom (rows 2 and 4) are free. Now these two partial Dirichlet nodes can be put together into one aggregate using the non-smoothed transfer operators from Section 3.4 given by

$$P = \left(\begin{array}{cc|cc} \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \text{ and } R = P^{\mathsf{T}}. \tag{C.9}$$

Using a zero initial guess, a simple calculation gives the coarse level problem $P^{\mathsf{T}}\mathrm{A}Pe_1 = P^{\mathsf{T}}r_0$ as

$$\left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 1 & 3 & -\frac{1}{\sqrt{2}} & 0 \\ \hline 0 & -\frac{1}{\sqrt{2}} & 2 & -1 \\ 0 & 0 & -1 & 1 \end{array}\right) e_1 = \left[\begin{array}{c} 0 \\ \frac{3}{\sqrt{2}} \\ \hline 2 \\ 2 \end{array}\right]. \tag{C.10}$$

The homogeneous Dirichlet boundary is preserved in the coarse level system (C.10) and consequently also in the prolongated coarse level correction $e_0 = Pe_1$ and the fine level solution before post-smoothing, as one can easily verify. Standard smoothing methods should be able to preserve the Dirichlet values in the solution vector.

C.4.2.3. Remove Dirichlet rows from prolongator

At least for homogeneous (partial) Dirichlet boundary nodes, one can also adapt the prolongation operator to fulfill the partial Dirichlet boundaries exactly, instead of building extra aggregates for (partial) Dirichlet nodes. The partial Dirichlet nodes are put into standard aggregates (together wit non-Dirichlet nodes), while the corresponding rows in the prolongator are zeroed out.

Example C.4.3. Revisiting the linear system (C.8) from Example C.4.2 using the transfer operators

$$P = \frac{1}{\sqrt{3}} \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } R = P^{\mathsf{T}}, \tag{C.11}$$

the coarse level problem is given by

$$\frac{1}{3} \begin{pmatrix} 2 & -2 \\ -2 & 7 \end{pmatrix} e_1 = \frac{1}{\sqrt{3}} \begin{bmatrix} 2 \\ 5 \end{bmatrix}. \tag{C.12}$$

Independent of the coarse grid correction e_1, the design of the prolongation operator automatically makes sure that the Dirichlet rows in the prolongated coarse grid correction e_0 remain zero and therefore cannot disturb the fine level solution vector.

Using a transfer operator smoothing strategy, which allows to describe a pattern for the transfer operators (such as the methods described in Section 4.4), makes it easy to implement this Dirichlet boundary node strategy.

Remark C.4.4 (Robustness). If the aggregation strategy builds a purely Dirichlet node aggregate adapting the resulting prolongation operator, one obtains zero coarse level basis functions, which cause the coarse level problem $\mathrm{A}_{\ell+1}$ to be singular. Therefore, it is highly important for the robustness of the multigrid method that each aggregate contains at lest one non-Dirichlet node to avoid singular coarse level matrices. However, one can state that for typical realistic examples this is not a common problem.

Appendix D

Comments on parallelization of SA-AMG

In Chapters 6 and 7 it has been demonstrated how to enable aggregation-based AMG methods as preconditioners for large contact mechanics problems making use of the flexibility of the proposed multigrid framework. The methods therein have been proven to be robust and efficient for several large problems that could not be solved up to now with iterative methods.

However, it is one thing to enable the usage of aggregation-based multigrid preconditioners for problems resulting from certain applications such as computational contact mechanics. It is quite another thing to apply aggregation-based multigrid preconditioners to really large and challenging problems resulting from industrial applications. Especially with an increasing problem size the scalability of the algorithms in a parallel environment gets more and more important. There is no doubt that the problems will grow in size in the future. Therefore, one should not completely ignore the scalability properties of the used methods.

However, in engineering one usually has very concrete problems with a rather fixed problem size to solve. So, engineers are more interested in fast solution algorithms rather than (slow, but perfectly) scalable methods. In Douglas et al. [53, Remark 3.28] the authors find clear words:

> "Actually, only wall-clock time counts on a parallel code. If you do not want the solution **now**, why bother with the nuisance of parallel programming?"

In other words: a good parallelization is worthless for a numerically inefficient algorithm. Therefore, the primary focus of this work is to find numerically efficient algorithms that work for concrete large problems that could not be solved before. Nevertheless, this chapter introduces some basic terms in context of parallelization and closes this work with some small scalability studies.

D.1. Theoretical considerations and definitions

Before discussing the parallelization capabilities of aggregation-based multigrid methods, first some basic notions and definitions are introduced.

In general, the term *Scalability* describes the property of a program to adapt automatically to a given number of processors p and therefore is highly important in context of parallel computing. Furthermore, *Granularity* is a measure for the size of program sections which are executed without communication with other processes. One can distinguish *fine grain* and *coarse grain* algorithms. Here, a fine grain algorithm has lots of synchronization points where communication is necessary whereas coarse grain algorithms consist of fewer and larger components that work most of the time independently without inter-processor communication. The finer the granularity, the greater the potential for parallelism and hence speed-up, but the greater the overheads of synchronization and communication. If no (or nearly no) communication at all is needed, the problem is often called *perfectly parallel* or *embarrassingly parallel*.

The effect of an increasing number of processors to the algorithm can be measured using the *Speedup* S.

Definition D.1.1 (Speedup)**.** The speedup S is a measure of the performance gain of a parallel code running on p processors compared to the sequential version with only one processor when keeping the problem size constant. Quantitatively, the speedup of p processors is given by

$$S = \frac{T(1)}{T(p)}.$$

Remark D.1.2 (Scaled speedup and weak scaling)*.* Whereas the speedup as given in Definition D.1.1 describes the strong scaling of an algorithm with an increasing number of processors p for a problem of constant size, there is also a *scaled speedup* which measures the parallel performance gain when increasing the global problem size with the number of processors used (weak scaling).

Definition D.1.3 (Amdahl's law)**.** Let p denote the number of processors (threads of execution) and let $f \in [0, 1]$ describe the fraction of the algorithm that is strictly serial. Then, the time $T(p)$ an algorithm takes to finish when being executed on p threads of execution is given by

$$T(p) = T(1)\Big(f + \frac{1}{p}(1-f)\Big). \tag{D.1}$$

Thus, the theoretical speedup S by running the algorithm on p threads of execution can be calculated with

$$S(p) := \frac{T(1)}{T(p)} = \frac{1}{f + \frac{1}{p}(1-f)}. \tag{D.2}$$

Amdahl's law describes a theoretical upper bound for the maximum possible speedup without considering overhead through increasing communication with an increasing number of processors p. It is based on the fraction f of the algorithm which is strictly serial. Often it is helpful to have a small example to illustrate the meaning of Definition D.1.3.

Example D.1.4 (Amdahl's law)**.** Let the serial fraction f of an algorithm be $f = 0.1$, i.e., 10% of the algorithm are strictly serial and 90% can be (perfectly) parallelized. Assuming 16 processors, the theoretical upper bound of the speedup is calculated with 6.4. Note that the maximum possible (theoretical) speedup with an infinite number of processors is only 10.

Obviously it is important to keep the strictly serial fraction of an algorithm small in order to obtain a good (theoretical) speedup. Another important point is to keep the overhead for the communication low.

D.2. Parallel multigrid

Some general comments on the parallelization of algebraic multigrid preconditioners can be found, e.g., in Haase et al. [83]. In this section, the different ingredients of an aggregation-based AMG algorithm are briefly analyzed with regard to parallelization.

Aggregation algorithm: Using the so-called *uncoupled aggregation* routine has the advantage that there are no communication costs at all. Each processor only aggregates local node information. Aggregates are not allowed to cross processor boundaries. Note that the uncoupled aggregation algorithm may not produce nice aggregates if there are not enough non-aggregated nodes left on the coarser levels. Thus, it makes sense to combine the uncoupled aggregation algorithm with a re-partitioning strategy that uses a smaller number of processors on the coarser levels.

Remark D.2.1 (Coupled aggregation). As an alternative one could also use a *coupled aggregation* algorithm which per definition allows aggregates to cross processor boundaries. However, the implementation is very complicated and the performance might not be satisfactory due to the higher communication costs (especially if the transfer operator basis functions are later smoothed by a smoothed aggregation method).

Tentative transfer operators: When using uncoupled aggregates it is possible to build the tentative prolongation operators without any inter-process communication.

Transfer operator smoothing: For the transfer operator smoothing some communication is necessary when calculating $\mathrm{A}_\ell \widehat{P}_{\ell+1}$, e.g., in (3.11). Further communication may be necessary for example to determine damping parameters ω based on some eigenvalue estimations (cf. Remark 3.5.2) or for more advanced calculations (such as local damping factors as in (3.14)). Usually, transfer operator smoothing pays off the additional communication costs resulting in a significantly reduced number of linear iterations.

Galerkin product: The Galerkin product (2.6) with its sparse triple matrix product is definitely one of the most expensive parts in a multigrid method, since it comes along with some communication even if the transfer operators can be considered to be very sparse. Since one cannot avoid the sparse matrix-matrix product it is highly important to have an efficient parallel matrix-matrix multiplication routine. Parallel matrix algorithms and parallel matrix-matrix multiplication algorithms in particular are still an active field of research (cf. Buluc and Gilbert [46], Gustavson [82]).

Level smoother: Typical level smoothers such as Gauss–Seidel methods are inherently sequential. Therefore, relaxation-based smoothers are usually combined with outer Schwarz methods. The performance of the resulting smoothers is sufficient for our purposes. An overview on other ultra-parallel multigrid smoothers can be found in Baker et al. [12].

Coarse level solver: Often, a direct solver is used on the coarsest level which can handle the low-energy error modes that are not tackled by the iterative level smoothers. A direct solver is a strictly serial algorithm. Performing the LU-decomposition during the setup and the back- and forward substitution during the iteration phase may – depending on the problem size on the coarsest level – dominate the overall timings in a multigrid method. Multigrid cycles other than the V-cycle may suffer from a rather high number of coarse level solves. For example, the work by Notay and Napov [146] discusses the issues of the K-cycle (cf. Notay [143]) for the parallel performance due to the high number of coarse level solves. Similar problems may occur for a W-cycle. Only with the V-cycle the number of coarse level solves scales linearly with the number of (multigrid) iterations.

D.3. Rebalancing and coarse grid correction

Load balancing is extremely important especially for coarse grained algorithms such as the aggregation and tentative prolongation methods to avoid processors waiting for other processors with a too high load. Especially for higher coarsening rates it makes sense to reduce the number of involved processors on the coarser levels. When using the uncoupled aggregation routine this might even be necessary to obtain reasonable aggregates on the coarse levels. Furthermore it allows to reduce the communication costs, if less processors are involved. Since many "parallel" implementations of direct coarse solvers run a sequential algorithm on a single processor only and communicate the data back and forth from and to the other processors, it makes sense to use only one processor on the coarsest level any way (see Example 3.6).

In order to reduce the communication overhead one can also use completely different ideas, such as additive AMG schemes as presented in, e.g., in Vassilevski and Yang [202]. However, in this thesis these kinds of methods are not further pursued.

D.4. Scaling studies

D.4.1. Two solid bodies example

In contrary to the examples in Section 6.5.1, now a discretization with $60 \times 60 \times 60$ nodes per solid body is used which corresponds to 432000 nodes and 1296000 degrees of freedom. For the scaling experiments the rotation angles are fixed to $\alpha_y = \alpha_z = 0$.

D.4.1.1. Strong scaling study

First of all, a small strong scaling study using the two solid bodies example in condensed formulation is performed. In each Newton iteration, the resulting linear system is solved using GMRES with a 3 level Contact PA-AMG preconditioner. Again, the nonlinear iteration stops if $\|\mathbf{r}^{\boldsymbol{u}}\|_e < 10^{-8}$ and $\|\Delta \boldsymbol{u}\|_e < 10^{-8}$ holds. On each level ℓ, 3 sweeps with a damped symmetric Gauss–Seidel method (damping parameter: 0.7) are applied. The coarse solver is chosen to be KLU. The coarsening stops if the coarse level matrix has less than 8000 rows. For a rapid coarsening the minimum size for an aggregate is set to 18 nodes.

For the strong scaling experiment the problem size is fixed but the number of processors is increased starting from 32 to 128. Table D.1 gives the corresponding numbers and timings exemplarily for time step 25. Obviously, the number of aggregates is influenced by the number of processors due to the uncoupled aggregation strategy. It is hard to predict how the number

Distribution		Aggregates		Setup		Solver phase (fixed tol)			Solver phase (adapt. tol)		
#PROCS		$\ell = 1$	$\ell = 2$	Time	EFF	#It.	Time	EFF	#It.	Time	EFF
128	8×16	4544	908	4.8	0.58	483	23.6	0.85	409	18.5	0.99
64	4×16	5000	950	7.2	0.77	495	40.7	0.97	428	37.4	0.99
64	8×8	5000	950	7.9	0.70	495	41.6	0.96	428	35.1	1.05
32	2×16	2860	806	10.5	1.05	519	79.9	1.00	439	75.3	0.98
32	8×4	2860	806	11.1	1.00	519	79.2	1.00	439	73.8	1.00

Table D.1.: Two solid bodies example – Strong scaling results for time step 25. The ‘Setup’ timings denote the wall-clock time in $[s]$ for generating one multigrid hierarchy in time step 25. The ‘Solver phase’ contains the summed up number of iterations and timings in $[s]$ for all linear systems within time step 25. When a fixed solver tolerance is used, this corresponds to 6 linear systems per time step, for the adaptive solver tolerance one needs 7 linear systems per time step. The efficiency ‘EFF’ is calculated by $EFF := \frac{T(32)_{8\times 4}}{\frac{p}{32}T(p)}$.

of aggregates really corresponds to the number of processors. But in general one can state that it is important that there are enough nodes on each processor to find appropriate aggregates, i.e., one should not use too many processors to allow for a reasonable number of aggregates on each processor. Since the active set is not changing over time, the multigrid hierarchy is only built once in the beginning of every time step and reused for all linear solves within the Newton iteration.

The results for a fixed stopping criterion for the linear solver are compared with an adaptive stopping criterion as already introduced in Section 6.5.4, viz.

$$\left\| \frac{\mathbf{r}_i^k}{\mathbf{r}_i^0} \right\|_e < 10^{-8} \quad \text{versus} \quad \left\| \frac{\mathbf{r}_i^k}{\mathbf{r}_i^0} \right\|_e < \varepsilon_a = \begin{cases} 10^{-8} & \text{if } i = 0, \\ 0.001 \cdot \frac{10^{-5}}{\left\|\mathbf{r}_i^{\boldsymbol{u}}\right\|_e} & \text{if } i > 0 \ \wedge \ \varepsilon_a < \frac{10^{-5}}{\left\|\mathbf{r}_i^{\boldsymbol{u}}\right\|_e} \end{cases}, \tag{D.3}$$

where $\mathbf{r}_i^k$ denotes the linear residual after k GMRES iterations in Newton step i and $\mathbf{r}_i^{\boldsymbol{u}}$ denotes the nonlinear residual after i Newton iterations.

Note that the number of iterations in Table D.1 denotes the accumulated number of linear iterations for the full time step 25. In case of the fixed tolerance this includes the linear iterations from 6 Newton steps and for the adaptive solver tolerance it includes 7 Newton steps. Even though the number of Newton steps is slightly increasing when using an adaptive weakened linear solver tolerance, one can save a significant number of iterations (and computational time). Note that the given timings also represent the accumulated solving time (iteration phase) for all linear systems in the time step 25.

When using a fixed tolerance, one is loosing efficiency due to the increasing communication. It is interesting to see that an adaptive solver tolerance compensates the higher communication costs with decreasing iteration timings in this example.

Remark D.4.1 (Weak scaling study). A reasonable weak scaling study is hard to perform as the ratio of the challenging slave contact nodes and the inner nodes is decreasing with increasing problem size.

	#Aggregates	SIMPLEC		Uzawa	
ℓ	disp/Lagr	It.	Time	It.	Time
2	$11429/297$	129.3	24.6	128.6	24.5
3	$7241/44$	102.2	36.1	233.5	49.2
4	$2933/14$	74.8	16.3	222.5	49.3
5	$66/13$	69.5	16.8	217.6	50.1

Table D.2.: Two solid bodies example – Dependency of linear solver from number of multigrid levels. '#Aggregates' denotes the number of coarse level aggregates using ℓ multigrid levels both for the displacement degrees of freedom and Lagrange multipliers. 'It' gives the number of average linear iterations for all linear systems in time step 25 and 'Time' is the corresponding average time in $[s]$.

D.4.1.2. Influence of multigrid levels

Next, the influence of an increasing number of multigrid levels is studied. The two solid bodies contact example is now formulated as saddle point problem. The problem size on the finest level is fixed to be 1296000 degrees of freedom for the displacement variables and 10800 Lagrange multipliers corresponding to 3600 nodes at the contact interface. For the linear solver a fixed convergence criterion is used with $\left\|\frac{\mathbf{r}_i^k}{\mathbf{r}_i^0}\right\|_e < 10^{-8}$. The number of multigrid levels is prescribed between 2 and 5. For coarsening non-smoothed transfer operators are used with the methods as described in Chapter 7. As level smoothers the CheapSIMPLEC and CheapUzawa methods are compared. On each multigrid level 3 CheapSIMPLEC(0.6) or 3 CheapUzawa(0.6) iterations are applied which internally use 2 sweeps with symmetric Gauss–Seidel (0.6) and 1 sweep with symmetric Gauss–Seidel (0.4) for approximately inverting the displacement or Schur complement block in the SIMPLE iteration. Table D.2 shows the number of coarse level aggregates both for the displacement variables and Lagrange multipliers. Obviously, the number of iterations is decreasing with the increasing number of multigrid levels when a CheapSIMPLEC smoother is applied. The CheapSIMPLEC smoother turns out to provide the better approximation and clearly outperforms the weaker CheapUzawa smoother both in iterations and timings. As one can see from Table D.2 one should choose the number of multigrid levels not too small such that the level smoothers can work effectively. With only two multigrid levels, the level smoother has nearly no effect.

D.4.2. Channel flow

Here, a strong scaling study for a steady-state simple channel flow example is performed. A finite element discretization is used with stabilized linear finite elements for the incompressible Navier–Stokes equations on a $128 \times 128 \times 128$ discretization with altogether 8388608 degrees of freedom. The channel is 6 units in length and 2 units in width and height. A quadratic inflow profile is prescribed, i.e., $(1 - y^2)(1 - z^2)$ and homogeneous Dirichlet boundaries on the side walls. The dynamic viscosity is set to 0.01 and the density is chosen to be 1. To reduce the norm of the absolute nonlinear residual (with both velocity and pressure degrees of freedom) by 8 orders of magnitude, one needs 4 Newton iterations. Within each Newton step a GMRES solver is used together with a 3 level AMG preconditioner. In each Newton step i the GMRES method

#PROCS		#DOFs pp	# Lin. It.	Time	Speedup	EFF
6	(6×1)	1398101	106.3	993.3	1.00	1.00
12	(6×2)	699051	99.8	445.5	2.23	1.11
24	(6×4)	349525	103.5	249.0	3.99	1.00
48	(6×8)	174763	105.3	192.9	5.15	0.64
96	(6×16)	87381	110.5	105.8	9.39	0.59

Table D.3.: Channel flow example – Strong scaling results for steady-state channel flow. The 'PROCS' denotes the number of processors. 'DOFS pp' gives an approximate number of degrees of freedom per processor. 'Lin. It.' denotes the average number of linear iterations needed to solve on linear system with GMRES. 'Time' gives the corresponding iteration time in $[s]$.

is supposed to be converged if $\frac{\|\mathbf{r}_i^k\|_e}{\|\mathbf{r}_i^0\|_e} < 10^{-8}$ holds where $\mathbf{r}^k$ denotes the linear residual after k GMRES iterations.

D.4.2.1. Strong scaling study

For our strong scaling study the number of processors is increased from 6 to 96 (see Table D.3). For level smoothing 2 sweeps with a CheapSIMPLE (0.6) are applied on all multigrid levels (including the coarsest level). For the prediction smoother 3 sweeps with symmetric Gauss–Seidel (0.8) have been chosen together with 1 sweep with a symmetric Gauss–Seidel (0.7) smoother for the Schur complement solver. The size of the aggregates is postulated to be within 27 and 32 nodes per aggregate. The aggregates for the velocity degrees of freedom are reused for the pressure degrees of freedom. To avoid further communication costs non-smoothed transfer operators are used.

In Table D.3 one can see that the average number of linear iterations is rather constant and only slightly increasing with the number of processors. One also finds the average number of degrees of freedom per processor to decrease, which may lead to worse aggregates on the coarser levels and therefore explain the number of linear iterations to increase. Note that always a fixed number of 6 computational nodes is used and only the number of processors per node is increased starting from 1 to 16. Each node has two Intel Xeon E5-2670 Octocore CPUs which share 32 Gb memory. In the timings, there is a notable drop in performance between 24 and 48 processors. This has probably technical reasons when switching from using half of a CPU to a full CPU per node.

D.4.2.2. Influence of multigrid levels

Next, the effect of an increasing number of multigrid levels to the performance of the linear solver is studied. The number of multigrid levels is prescribed between 2 and 5 (as long as the coarsest problem does not get too small). For the experiments in Table D.4 the level smoother is chosen as 1 sweep with CheapSIMPLE (0.4) either on all multigrid levels or with a direct solver on the coarsest level. Internally, for the prediction smoother in the CheapSIMPLE level smoother 3 sweeps with symmetric Gauss–Seidel (0.6) are used with 1 sweep with a symmetric Gauss–Seidel (0.6) smoother for the Schur complement solver. The size of the aggregates is prescribed to be in the range of 27 and 32 nodes. Then, the effect of different transfer operator strategies is compared. 'PA-AMG' serves as reference with non-smoothed transfer operators (cf. Section

					Coarse solver			
					Ch.SIMPLE		Direct	
		ℓ	Coarse size	OC	It.	Time	It.	Time
Transfers	PA-AMG	2	280552	1.02	934	614.1	–	–
		3	8084	1.02	339	226.5	176	119.4
		4	1340	1.02	211	143.1	175	118.5
		5	444	1.02	188	129.3	175	121.3
	Emin	2	280552	1.05	338	227.9	–	–
		3	7060	1.05	172	119.8	104	75.3
		4	460	1.05	115	82.1	121	85.3
	Schur	2	280552	1.05	511	344.9	–	–
		3	7056	1.05	126	88.0	95	70.2
		4	260	1.05	90	64.8	92	65.2

Table D.4.: Channel flow example – Dependency of linear solver from number of multigrid levels and transfer operators. ‘PA-AMG’ denotes non-smoothed tentative transfer operators (cf. Section 3.4). ‘Emin’ represents the transfer operators smoothing strategy with local damping factors (cf. Section 3.5.2) combined with the Petrov–Galerkin approach from Section 4.2.2. ‘Schur’ denotes SchurComp(1, 2.0) from Chapter 4. ‘Coarse size’ is the number of degrees of freedom on the coarsest level using ℓ multigrid levels. ‘OC’ denotes the operator complexity. ‘It’ gives the number of linear iterations for the first linear system and ‘Time’ is the corresponding time in $[s]$.

3.4). ‘Emin’ denotes the smoothed transfers with local damping factors (cf. Section 3.5.2) together with the Petrov–Galerkin approach from Section 4.2.2. ‘Schur’ denotes the SchurComp approach from Chapter 4 with only one smoothing sweep. All simulations run on 64 processors spread over 4 nodes with 2 Intel Xeon E5-2670 Octocore CPUs each.

For using a direct solver on the coarsest level one needs at least 3 multigrid levels for that example in order to have a reasonably small coarse level problem. As expected, when using CheapSIMPLE as coarse solver one has benefit from a larger number of multigrid levels which corresponds to more calls to the smoothing methods (on different levels). When using a direct solver on the coarsest level, the number of multigrid levels is not so important. Depending on the chosen transfer operator strategy (and the resulting coarsening) the number of linear iterations is nearly independent from the number of multigrid levels ℓ. The example also demonstrates that the usage of more advanced smoothed transfer operators can drastically reduce the number of linear iterations and the corresponding solver timings. The different behavior of the aggregation routine depending on the transfer operator smoothing strategy is also notable. The SchurComp methods leads to a significantly better coarsening on the coarser levels in this example.

Bibliography

[1] M. F. Adams. Algebraic multrigrid methods for constrained linear systems with applications to contact problems in solid mechanics. *Numerical Linear Algebra with Applications*, 11(2-3):141–153, 2004.

[2] M. Adams. A parallel maximal independent set algorithm. Technical Report UCB/CSD-98-993, EECS Department, University of California, Berkeley, Jan 1998.

[3] P. Alart and A. Curnier. A mixed formulation for frictional contact problems prone to Newton like solution methods. *Computer Methods in Applied Mechanics and Engineering*, 92(3):353–375, 1991.

[4] D. M. Alber and L. N. Olson. Parallel coarse-grid selection. *Numerical Linear Algebra with Applications*, 14(8):611–643, 2007.

[5] D. M. Alber. *Efficient setup algorithms for parallel algebraic multigrid.* PhD thesis, University of Illinois at Urbana-Champaign, 2007.

[6] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1996. ISBN 9780521555692.

[7] O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods, i. *Numerische Mathematik*, 56:157–177, 1989.

[8] O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods, ii. *SIAM Journal Numerical Analysis*, 27:1569–1590, 1990.

[9] O. Axelsson and M. Neytcheva. A general approach to analyse preconditioners for two-by-two block matrices. *Numerical Linear Algebra with Applications*, 20(5):723–742, 2013.

[10] I. Babuška. The finite element method with Lagrangian multipliers. *Numerische Mathematik*, 20(3):179–192, 1973.

[11] A. H. Baker, T. V. Kolev, and U. M. Yang. Improving algebraic multigrid interpolation operators for linear elasticity problems. *Numerical Linear Algebra with Applications*, 17 (2-3):495–517, 2010.

[12] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Multigrid smoothers for ultra-parallel computing. *SIAM Journal on Scientific Computing*, 33(5):2864–2887, October 2011.

[13] R. E. Bank, J. W. L. Wan, and Z. Qu. Kernel preserving multigrid methods for convection-diffusion equations. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1150–1171, December 2005.

[14] R. Bank, B. Welfert, and H. Yserentant. A class of iterative methods for solving saddle point problems. *Numerische Mathematik*, 56(7):645–666, 1989.

[15] K.-J. Bathe. The inf–sup condition and its evaluation for mixed finite element methods. *Computers & structures*, 79(2):243–252, 2001.

[16] F. Bauer. Optimally scaled matrices. *Numerische Mathematik*, 5(1):73–87, 1963.

[17] F. Belgacem and Y. Maday. The mortar element method for three dimensional finite elements. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 31(2):289–302, 1997.

[18] F. B. Belgacem. The mortar finite element method with Lagrange multipliers. *Numerische Mathematik*, 84(2):173–197, 1999.

[19] F. Belgacem, P. Hild, and P. Laborde. The mortar finite element method for contact problems. *Mathematical and Computer Modelling*, 28(4–8):263 – 271, 1998. Recent Advances in Contact Mechanics.

[20] W. N. Bell, L. N. Olson, and J. Schroder. PyAMG: Algebraic multigrid solvers in python, 2008. Version 1.1.

[21] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[22] M. Benzi and V. Simoncini. On the eigenvalues of a class of saddle point matrices. *Numerische Mathematik*, 103(2):173–196, 2006.

[23] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[24] C. Bernardi, Y. Maday, and A. T. Patera. A new nonconforming approach to domain decomposition: The mortar element method. In H. Brezis and J. Lions, editors, *Nonlinear partial differential equations and their applications*, pages 13–51. Pitman/Wiley: London/New York, 1994.

[25] R. Blaheta. Algebraic multilevel methods with aggregations: An overview. In I. Lirkov, S. Margenov, and J. Waśniewski, editors, *Large-Scale Scientific Computing*, volume 3743 of *Lecture Notes in Computer Science*, pages 3–14. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-31994-8.

[26] J. Bonet and R. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997. ISBN 9780521572729.

[27] A. M. Bradley. *Algorithms for the Equilibration of Matrices and their Application to Limited-Memory Quasi-Newton Methods*. PhD thesis, Institute for computational and mathematical engineering at Standford University, 2010.

[28] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *SIAM Journal on Numerical Analysis*, 20(5):pp. 967–975, 1983.

[29] D. Braess and R. Sarazin. An efficient smoother for the stokes problem. *Applied Numerical Mathematics*, 23:3–19, 1997.

[30] D. Braess, M. Dryja, and W. Hackbush. A multigrid method for nonconforming FE-discretisations with application to non-matching grids. *Computing*, 63(1):1–25, 1999.

[31] D. Braess. *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*, chapter The convergence rate of a multigrid method with Gauss-Seidel relaxation for the Poisson equation, pages 368–386. Springer, 1982.

[32] D. Braess and W. Dahmen. Stability estimates of the mortar finite element method for 3-dimensional problems. *East-West Journal of Numerical Mathematics*, 6:249–264, 1998.

[33] D. Braess, W. Dahmen, and C. Wieners. A multigrid algorithm for the mortar finite element method. *SIAM Journal on Numerical Analysis*, 37(1):pp. 48–69, 2000.

[34] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev. Analysis of the inexact uzawa algorithm for saddle point problems. *SIAM Journal on Numerical Analysis*, 34(3):pp. 1072–1092, 1997.

[35] A. Brandt and O. Livne. *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2011. ISBN 9781611970746.

[36] A. Brandt and I. Yavneh. Inadequacy of first-order upwind difference schemes for some recirculating flows. *Journal of Computational Physics*, 93(1):128–143, 1991.

[37] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, pages 257–284. Cambridge Univ. Press, Cambridge, 1984.

[38] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56, 1986.

[39] A. Brandt. General highly accurate algebraic coarsening. *Electronic Transactions on Numerical Analysis*, 10:1–20, 2000.

[40] A. Brandt and I. Yavneh. Accelerated multigrid convergence and high-Reynolds recirculating flows. *SIAM Journal on Scientific Computing*, 14(3):607–626, 1993.

[41] J. Brannick and L. Zikatanov. Algebraic multigrid methods based on compatible relaxation ad energy minimization. *PennState report No AM304*, 2006.

[42] J. J. Brannick and R. D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM Journal on Scientific Computing*, 32:1393–1416, 2010.

[43] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. Adaptive smoothed aggregation (αSA) multigrid. *SIAM Review*, 47(2):317–346, 2005.

[44] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders. Towards adaptive smoothed aggregation (αSA) for nonsymmetric problems. *SIAM Journal on Scientific Computing*, 32(1):14–39, 2010.

[45] S. Brunßen, F. Schmid, M. Schäfer, and B. Wohlmuth. A fast and robust method for contact problems by combining a primal-dual active set strategy and algebraic multigrid. *International Journal for Numerical Methods in Engineering*, 69:524–543, 2007.

[46] A. Buluc and J. R. Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. *SIAM Journal on Scientific Computing*, 34(4):C170–C191, 2012.

[47] P. W. Christensen. A semi-smooth Newton method for elasto-plastic contact problems. *International Journal of Solids and Structures*, 39(8):2323 – 2341, 2002.

[48] P. Christensen and J.-S. Pang. Frictional contact algorithms based on semismooth Newton methods. In M. Fukushima and L. Qi, editors, *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, volume 22 of *Applied Optimization*, pages 81–116. Springer US, 1999. ISBN 978-1-4419-4805-2.

[49] A. Cleary, R. Falgout, V. Henson, and J. Jones. Coarse-grid selection for parallel algebraic multigrid. In A. Ferreira, J. Rolim, H. Simon, and S.-H. Teng, editors, *Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 104–115. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64809-3.

[50] E. de Sturler and J. Liesen. Block-diagonal and constraint preconditioners for nonsymmetric indefinite linear systems. part i: Theory. *SIAM Journal on Scientific Computing*, 26(5):1598–1619, 2005.

[51] J. E. Dendy. Black box multigrid for nonsymmetric problems. *Applied Mathematics and Computation*, 13:261–283, 1983.

[52] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2003.

[53] C. C. Douglas, G. Haase, and U. Langer. *A Tutorial on Elliptic PDE Solvers and Their Parallelization*. Society for Industrial and Applied Mathematics, 2003.

[54] I. Duff and J. Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM Journal on Matrix Analysis and Applications*, 22(4):973–996, 2001.

[55] I. S. Duff and J. Koster. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 20: 889–901, 1999.

[56] I. Duff, A. Erisman, and J. Reid. *Direct Methods for Sparse Matrices*. Monographs on numerical analysis. Clarendon Press, 1986. ISBN 9780198534211.

[57] L. Dutto, W. Habashi, and M. Fortin. An algebraic multilevel parallelizable preconditioner for large-scale CFD problems. *Computer Methods in Applied Mechanics and Engineering*, 149:303–318, 1997.

[58] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Numerical mathematics and scientific computation. Oxford University Press, Great Clarendon Street, Oxford OX2 6DP, 2005.

[59] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 227(3):1790–1808, January 2008.

[60] H. C. Elman and G. H. Golub. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM Journal on Numerical Analysis*, 31(6):pp. 1645–1661, 1994.

[61] M. Emans. Performance of parallel AMG-preconditioners in CFD-codes for weakly compressible flows. *Parallel Computing*, 36(5–6):326 – 338, 2010. Parallel Matrix Algorithms and Applications.

[62] L. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2010. ISBN 9780821849743.

[63] R. Falgout and J. Schroder. Non-galerkin coarse grids for algebraic multigrid. *SIAM Journal on Scientific Computing*, 36(3):C309–C334, 2014.

[64] R. D. Falgout and P. S. Vassilevski. On generalizing the algebraic multigrid framework. *SIAM Journal on Numerical Analysis*, 4:1669–1693, 2004.

[65] R. D. Falgout and U. M. Yang. hypre: a library of high performance preconditioners. In *Preconditioners, Lecture Notes in Computer Science*, pages 632–641, 2002.

[66] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov. On two-grid convergence estimates. *Numerical Linear Algebra with Applications*, 12(5-6):471–494, 2005.

[67] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2013. ISBN 9781118723210.

[68] A. Francavilla and O. C. Zienkiewicz. A note on numerical computation of elastic contact problems. *International Journal for Numerical Methods in Engineering*, 9(4):913–924, 1975.

[69] J. Gaidamour, J. J. Hu, C. M. Siefert, and R. S. Tuminaro. Design considerations for a flexible multigrid preconditioning library. *Scientific Programming*, 20(3):223–239, 2012.

[70] P. Gamnitzer. *Residual-based variational multiscale methods for turbulent flows and fluid-structure interaction.* PhD thesis, Technische Universität München, 2010.

[71] M. W. Gee, J. J. Hu, and R. S. Tuminaro. A new smoothed aggregation multigrid method for anisotropic problems. *Numerical Linear Algebra with Applications*, 16:19–37, 2009.

[72] M. W. Gee, U. Küttler, and W. A. Wall. Truly monolithic algebraic multigrid for fluid–structure interaction. *International Journal for Numerical Methods in Engineering*, 85(8): 987–1016, 2011.

[73] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. ML 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[74] T. E. Giddings and J. Fish. An algebraic two-level preconditioner for asymmetric, positive-definite systems. *International Journal for Numerical Methods in Engineering*, 52(12):1443, 2001.

[75] M. Gitterle. *A dual mortar formulation for finite deformation frictional contact problems including wear and thermal coupling*. PhD thesis, Technische Universität München, 2012.

[76] M. Gitterle, A. Popp, M. W. Gee, and W. A. Wall. Finite deformation frictional mortar contact using a semi-smooth Newton method with consistent linearization. *International Journal for Numerical Methods in Engineering*, 84(5):543–571, 2010.

[77] G. H. Golub and H. A. van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1):35–65, 2000.

[78] J. Gopalakrishnan and J. E. Pasciak. Multigrid for the mortar finite element method. *SIAM Journal on Numerical Analysis*, 37(3):1029–1052, 2000.

[79] V. Gravemeier, M. W. Gee, and W. A. Wall. An algebraic variational multiscale–multigrid method based on plain aggregation for convection–diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 198(47-48):3821, 2009.

[80] M. Griebel, T. Neunhoeffer, and H. Regler. Algebraic multigrid methods for the solution of the Navier–Stokes equations in complicated geometries. *International Journal for Numerical Methods in Fluids*, 26(3):281–301, 1998.

[81] H. Guillard and P. Vaněk. An aggregation multigrid solver for convection-diffusion problems on unstructured meshes. Tech Rep. UCD/CCM 130, Univ. of Col. - Denver, 1998.

[82] F. G. Gustavson. Two fast algorithms for sparse matrices: Multiplication and permuted transposition. *ACM Transactions on Mathematical Software (TOMS)*, 4(3):250–269, 1978.

[83] G. Haase, M. Kuhn, and S. Reitzinger. Parallel algebraic multigrid methods on distributed memory computers. *SIAM Journal on Scientific Computing*, 24(2):410–427, 2002.

[84] G. Haase and U. Langer. Multigrid methods: from geometrical to algebraic versions. In *Modern methods in scientific computing and applications*, pages 103–153. Springer, 2002.

[85] W. Hackbusch. *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*, chapter Multi-grid convergence theory, pages 177–219. Springer, 1982.

[86] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*, volume 95 of *Applied Mathematical Sciences*. Springer, 1994.

[87] W. Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Series in Computational Mathematics*. Springer, 1st edition edition, 1985.

[88] M. A. Heroux. *AztecOO User Guide*. Sandia National Laboratories, Albuquerque, NM 87185, sand report edition, August 2007.

[89] M. A. Heroux and J. M. Willenbring. A new overview of the trilinos project. *Scientific Programming*, pages 83–88, 2012.

[90] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. An overview of the trilinos project. *ACM Transactions on Mathematical Software*, 31(3):397–423, 2005.

[91] C. Hesch and P. Betsch. Transient 3D contact problems - NTS method: mixed methods and conserving integration. *Computational Mechanics*, 48:437–449, 2011.

[92] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–436, 1952.

[93] M. Hintermüller, K. Ito, and K. Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM Journal on Optimization*, 13(3):865–888, August 2002.

[94] M. Hintermüller, V. Kovtunenko, and K. Kunisch. Semismooth Newton methods for a class of unilaterally constrained variational problems. *Advances in Mathematical Sciences and Applications*, 14(2):513–535, 2004.

[95] G. Holzapfel. *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*. John Wiley & Sons, 2000. ISBN 9780471823193.

[96] S. Hüeber and B. Wohlmuth. A primal-dual active set strategy for non-linear multibody contact problems. *Computer Methods in Applied Mechanics and Engineering*, 194:3147–3155, 2005.

[97] S. Hüeber, A. Matei, and B. Wohlmuth. Efficient algorithms for problems with friction. *SIAM Journal on Scientific Computing*, 29:70–92, 2007.

[98] S. Hüeber, G. Stadler, and B. Wohlmuth. A primal-dual active set algorithm for three-dimensional contact problems with coulomb friction. *SIAM Journal on Scientific Computing*, 30:572–596, 2008.

[99] S. Hüeber. *Discretization techniques and efficient algorithms for contact problems*. PhD thesis, Universität Stuttgart, 2008.

[100] K. James. Convergence of matrix iterations subject to diagonal dominance. *SIAM Journal on Numerical Analysis*, 10(3):478–484, 1973.

[101] A. Janka. Smoothed aggregation multigrid for a Stokes problem. Institute of Analysis and Scientific Comuting, Ecole Polytechnique Federale de Lausanne, 8 2006.

[102] C. Keller, N. I. M. Gould, and A. J. Wathen. Constrained preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis & Applications*, 21(4):1300 – 1317, 2000.

[103] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, et al. Multiphysics simulations challenges and opportunities. *International Journal of High Performance Computing Applications*, 27(1):4–83, 2013.

[104] N. Kikuchi and J. T. Oden. *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*. SIAM studies in applied mathematics. Society for Industrial and Applied Mathematics, 1988. ISBN 9780898714685.

[105] C. Kim, R. D. Lazarov, J. E. Pasciak, and P. S. Vassilevski. Multiplier spaces for the mortar finite element method in three dimensions. *SIAM Journal on Numerical Analysis*, 39:519–538, 2000.

[106] H. Kim, J. Xu, and L. Zikatanov. A multigrid method based on graph matching for convection–diffusion equations. *Numerical Linear Algebra with Applications*, 10(1-2):181–195, 2003.

[107] T. V. Kolev and P. S. Vassilevski. AMG by element agglomeration and constrained energy minimization interpolation. *Numerical Linear Algebra with Applications*, 13(9):771–788, 2006.

[108] J. Kraus and S. Margenov. *Robust Algebraic Multilevel Methods and Algorithms*. Radon series on computational and applied mathematics. Walter De Gruyter, 2009. ISBN 9783110193657.

[109] R. Krause and B. Wohlmuth. Nonconforming domain decomposition techniques for linear elasticity. *East-West Journal of Numerical Mathematics*, 8:177–206, 2000.

[110] R. H. Krause and B. I. Wohlmuth. Multigrid methods for mortar finite elements. In *Multigrid methods, VI (Gent, 1999)*, volume 14 of *Lecture Notes in Computational Science and Engineering*, pages 136–142. Springer, Berlin, 2000.

[111] K. Kunisch and G. Stadler. Generalized Newton methods for the 2D-Signorini contact problem with friction in function space. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(4):827–854, 3 2005.

[112] U. Küttler. *Effiziente Lösungsverfahren für Fluid-Struktur-Interaktions-Probleme*. PhD thesis, Technische Universität München, 2009.

[113] O. A. Ladyzhenskaya and R. A. Silverman. *The mathematical theory of viscous incompressible flow*, volume 76. Gordon and Breach New York, 1969.

[114] M. Lallemand, H. Steve, and A. Dervieux. Unstructured multigridding by volume agglomeration: Current status. *Computer and Fluids*, 21:397–433, 1992.

[115] U. Langer and H. Yang. Numerical simulation of fluid-structure interaction problems with hyperelastic models: A monolithic approach. *arXiv preprint arXiv:1408.3737*, 2014.

[116] T. A. Laursen. *Computational contact and impact mechanics*. Springer-Verlag Berlin Heidelberg, 2002.

[117] T. A. Laursen and J. C. Simo. A continuum-based finite element formulation for the implicit solution of multibody, large deformation-frictional contact problems. *International Journal for Numerical Methods in Engineering*, 36(20):3451–3485, 1993.

[118] T. A. Laursen. *Formulation and treatment of frictional contact problems using finite elements*. PhD thesis, Standford University, 1992.

[119] J. Liesen and Z. Strakos. *Krylov Subspace Methods: Principles and Analysis*. Numerical Mathematics and Scientific Computation. OUP Oxford, 2012. ISBN 9780191630323.

[120] O. E. Livne. Coarsening by compatible relaxation. *Numerical Linear Algebra with Applications*, 11:205–227, 2004.

[121] R. Lonsdale. *An algebraic multigrid scheme for solving the Navier-Stokes equations on unstructured meshes*, volume 7 of *Numerical Methods in Laminar and Turbulent Flow*, pages 1432–1442. Pineridge press, Swansea, U.K., 1991.

[122] S. P. MacLachlan and L. N. Olson. Theoretical bounds for algebraic multigrid performance: review and analysis. *Numerical Linear Algebra with Applications*, 21(2):194–220, 2014.

[123] J. Mandel, S. McCormick, and J. Ruge. An algebraic theory for multigrid methods for variational problems. *SIAM Journal on Numerical Analysis*, 25(1):91–110, 1988.

[124] J. Mandel, M. Brezina, and P. Vaněk. Energy optimization of algebraic multigrid bases. *Computing*, 62:205–228, 1999.

[125] J. Mandel. *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, chapter Hierarchical Preconditioning and Partial Orthogonalization for the p-Version Finite Element Method, pages 141–156. Society for Industrial and Applied Mathematics, 1989.

[126] J. Mandel. On block diagonal and Schur complement preconditioning. *Numerische Mathematik*, 58(1):79–93, 1990.

[127] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[128] A. Maslow. *The Psychology of Science: A Reconnaissance*. Maurice Bassett, 2004. ISBN 9780976040231.

[129] D. Mavriplis and V. Venkatakfrishnan. Agglomeration multigrid for viscous turbulent flows. *AIAA Journal*, 1994:2332, 1994.

[130] D. J. Mavriplis. Directional agglomeration multigrid techniques for high Reynolds number viscous flow solvers. *AIAA Journal*, 37:393–415, 1999.

[131] S. McCormick. Multigrid methods for variational problems: General theory for the V-cycle. *SIAM Journal on Numerical Analysis*, 22(4):634–643, 1985.

[132] C. Mense and R. Nabben. On algebraic multilevel methods for non-symmetric systems - convergence results. *Electronic Transactions on Numerical Analysis*, 30:323–345, 2008.

[133] B. Metsch. *Algebraic Multigrid (AMG) for Saddle Point Systems*. PhD thesis, Universitäts-und Landesbibliothek Bonn, 2013.

[134] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[135] A. C. Muresan and Y. Notay. Analysis of aggregation-based multigrid. *SIAM Journal on Scientific Computing*, 30:1082–1103, 2008.

[136] M. Murphy, G. Golub, and A. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972, 2000.

[137] A. Napov. *Algebraic analysis of V-cycle multigrid and aggregation-based two-grid methods*. PhD thesis, Faculté des sciences appliquées, 2010.

[138] A. Napov and Y. Notay. When does two-grid optimality carry over to the V-cycle? *Numerical Linear Algebra with Applications*, 17(2-3):273–290, 2010.

[139] A. Napov and Y. Notay. Algebraic analysis of aggregation-based multigrid. *Numerical Linear Algebra with Applications*, 18(3):539–564, 2011.

[140] A. Napov and Y. Notay. An algebraic multigrid method with guaranteed convergence rate. *SIAM Journal on Scientific Computing*, 34(2):1079–1109, April 2012.

[141] S. Noschese, L. Pasquini, and L. Reichel. Tridiagonal Toeplitz matrices: properties and novel applications. *Numerical Linear Algebra with Applications*, 20(2):302–326, 2013.

[142] Y. Notay. Algebraic multigrid and algebraic multilevel methods: a theoretical comparison. *Numerical Linear Algebra with Applications*, 12:419–451, 2005.

[143] Y. Notay. An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37:123–146, 2010.

[144] Y. Notay. Aggregation-based algebraic multigrid for convection-diffusion equations. *SIAM Journal on Scientific Computing*, 34(4):A2288–A2316, 2012.

[145] Y. Notay. A new analysis of block preconditioners for saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 35(1):143–173, 2014.

[146] Y. Notay and A. Napov. A massively parallel solver for discrete Poisson-like problems. Technical Report GANMN 14–01 (Revised version), Université Libre de Bruxelles, Brussels, Belgium, 2014.

[147] Y. Notay. AGMG software and documentation.

[148] Y. Notay. Algebraic analysis of two-grid methods: the nonsymmetric case. *Numerical Linear Algebra with Applications*, 17(1):73–96, 2010.

[149] M. Olschowka and A. Neumaier. A new pivoting strategy for gaussian elimination. *Linear Algebra and its Applications*, 240(0):131 – 151, 1996.

[150] L. N. Olson and J. B. Schroder. Smoothed aggregation multigrid solvers for high-order discontinuous Galerkin methods for elliptic problems. *Journal of Computational Physics*, 230(18):6959–6976, 2011.

[151] L. N. Olson, B. Hiriyur, J. Gaidamour, R. S. Tuminaro, C. Siefert, J. J. Hu, M. W. Gee, H. Waisman, A. Gerstenberger, T. A. Wiesner, J. B. Schroder, and D. E. Keyes. A flexible amg framework centered on energy minimization. Technical report, Sandia National Laboratories, 2010.

[152] L. N. Olson, J. Schroder, and R. S. Tuminaro. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications*, 17(4):713–733, 2010.

[153] L. N. Olson, J. B. Schroder, and R. S. Tuminaro. A general interpolation strategy for algebraic multigrid using energy minimization. *SIAM Journal on Scientific Computing*, 33(2):966–991, 2011.

[154] S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15:1787–1972, 1972.

[155] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation, New York, 1980.

[156] A. Popp. *Mortar Methods for Computational Contact Mechanics and General interface Problems*. PhD thesis, Technische Universität München, 2012.

[157] A. Popp, B. I. Wohlmuth, M. W. Gee, and W. A. Wall. Dual quadratic mortar finite element methods for 3D finite deformation contact. *SIAM Journal on Scientific Computing*, 34(4): B421–B446, 2012.

[158] A. Popp, M. W. Gee, and W. A. Wall. A finite deformation mortar contact formulation using a primal–dual active set strategy. *International Journal for Numerical Methods in Engineering*, 79(11):1354–1391, 2009.

[159] A. Popp, M. Gitterle, M. W. Gee, and W. A. Wall. A dual mortar approach for 3D finite deformation contact with consistent linearization. *International Journal for Numerical Methods in Engineering*, 83(11):1428–1465, 2010.

[160] A. Popp, A. Seitz, M. W. Gee, and W. A. Wall. Improved robustness and consistency of 3D contact algorithms based on a dual mortar approach. *Computer Methods in Applied Mechanics and Engineering*, 264:67–80, 2013.

[161] A. Prokopenko, J. J. Hu, T. A. Wiesner, C. M. Siefert, and R. S. Tuminaro. Muelu user's guide 1.0. Technical Report SAND2014-18874, Sandia National Labs, 2014.

[162] M. A. Puso. A 3D mortar method for solid mechanics. *International Journal for Numerical Methods in Engineering*, 59(3):315–336, 2004.

[163] M. A. Puso and T. A. Laursen. A mortar segment-to-segment frictional contact method for large deformations. *Computer Methods in Applied Mechanics and Engineering*, 193 (45-47):4891–4913, November 2004.

[164] M. A. Puso and T. A. Laursen. A mortar segment-to-segment contact method for large deformation solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 193(6–8):601 – 629, 2004.

[165] L. Qi and J. Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, 58(1-3):353–367, 1993.

[166] J. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[167] J. Ruge and K. Stüben. *Multigrid Methods*, chapter Algebraic Multigrid, pages 73–131. Frontiers in Applied Mathematics. SIAM, 1987.

[168] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7 (3):856–869, 1986.

[169] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, March 1993.

[170] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition edition, 2003.

[171] Y. Saad and H. A. van der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):1–33, November 2000.

[172] M. Sala, K. Stanley, and M. Heroux. Amesos: A set of general interfaces to sparse direct solver libraries. In *Proceedings of PARA'06 Conference, Umea, Sweden*, 2006.

[173] M. Sala and R. S. Tuminaro. A new Petrov-Galerkin smoothed aggregation preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 31(1):143–166, 2008.

[174] J. B. Schroder. *Generalizing smoothed aggregation-based algebraic multigrid.* PhD thesis, University of Illinois at Urbana-Champaign, 2010.

[175] P. Seshaiyer and M. Suri. hp submeshing via non-conforming finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 189(3):1011 – 1030, 2000.

[176] J. C. Simo and T. J. R. Hughes. *Computational inelasticity.* Springer, 1998.

[177] J. C. Simo, P. Wriggers, and R. L. Taylor. A perturbed Lagrangian formulation for the finite element solution of contact problems. *Computer Methods in Applied Mechanics and Engineering*, 50(2):163–180, August 1985.

[178] V. Simoncini. Block triangular preconditioners for symmetric saddle-point problems. *Applied Numerical Mathematics*, 49(1):63 – 80, 2004. Numerical Algorithms, Parallelism and Applications.

[179] A. Sluis. Condition numbers and equilibration of matrices. *Numerische Mathematik*, 14 (1):14–23, 1969.

[180] G. Stadler. Semismooth newton and augmented lagrangian methods for a simplified friction problem. *SIAM Journal on Optimization*, 15(1):39–62, January 2005.

[181] K. Stüben. *Multigrid*, Appendix A: An introduction to algebraic multigrid, pages 413–533. U. Trottenberg and C.W. Oosterlee and A. Schüller, 2001.

[182] K. Stüben and U. Trottenberg. *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*, chapter Multigrid methods: Fundamental algorithms, model problem analysis and applications, pages 1–176. Springer, 1982.

[183] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1–2):281 – 309, 2001. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[184] O. Taussky. A recurring theorem on determinants. *The American Mathematical Monthly*, 56(10):pp. 672–676, 1949.

[185] J. Thomas, B. Diskin, and A. Brandt. Distributed relaxation multigrid and defect correction applied to the compressible Navier-Stokes equations. Technical report, NASA Langley Research Center, 1999.

[186] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid.* Academic Press, 2001. ISBN 9780127010700.

[187] R. S. Tuminaro, M. A. Heroux, S. A. Hutchinson, and J. N. Shadid. *Official Aztec User's Guide, Version 2.1*. Sandia National Laboratories, Albuquerque, NM 87185, 1999.

[188] H. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[189] H. A. Van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, 1994.

[190] J. Van Doormaal and G. Raithby. Enhancements of the simple method for predicting incompressible fluid flows. *Numerical heat transfer*, 7(2):147–163, 1984.

[191] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65(1):138–158, 1986.

[192] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56:179–196, 1996.

[193] P. Vaněk. Acceleration of convergence of a two-level algorithm by smoothing transfer operators. *Applications of Mathematics*, 37:265–274, 1992.

[194] P. Vaněk. Fast multigrid solver. *Applications of Mathematics*, 40:1–20, 1995.

[195] P. Vaněk. Smoothed prolongation multigrid with rapid coarsening and massive smoothing. *Applications of Mathematics*, 57:1–10, 2012.

[196] P. Vaněk and M. Brezina. Nearly optimal convergence result for multigrid with aggressive coarsening and polynomial smoothing. *Applications of Mathematics*, 58:369–388, 2013.

[197] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid on unstructured meshes. Technical report, 1994.

[198] P. Vaněk, A. Janka, and H. Guillard. Convergence of algebraic multigrid based on smoothed aggregation ii: Extension to a Petrov-Galerkin method. *INRIA Technical Report 3683*, 1999.

[199] P. Vaněk, M. Brezina, and J. Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88:559–579, 2001.

[200] R. Varga. *Matrix Iterative Analysis*. Springer Series in Computational Mathematics. Springer, 2009. ISBN 9783642051548.

[201] P. S. Vassilevski. *Multilevel Block Factorization Preconditioners*. Springer, New York, NY, 2008.

[202] P. S. Vassilevski and U. M. Yang. Reducing communication in algebraic multigrid using additive variants. *Numerical Linear Algebra with Applications*, 21(2):275–296, 2014.

[203] P. S. Vassilevski. General constrained energy minimization interpolation mappings for AMG. *SIAM Journal on Scientific Computing*, 32(1):1–13, 2010.

[204] R. Verfürth. The contraction number of a multigrid method with mesh ratio 2 for solving the Poisson's equation. *Linear Algebra and its Applications*, 60:332–348, 1984.

[205] C. Vuik, A. Saghir, and G. P. Boerstoel. The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces. *International Journal for Numerical Methods in Fluids*, 33(7):1027–1040, 2000.

[206] M. Wabro. Coupled algebraic multigrid methods for the Oseen problem. *Computing and Visualization in Science*, 7:141–151, 2004.

[207] C. Wagner. On the algebraic construction of multilevel transfer operators. *Computing*, 65:73–95, August 2000.

[208] W. Wall and M. Gee. Baci—a multiphysics simulation environment. Technical report, Technische Universität München, 2014.

[209] W. Wan, T. Chan, and B. Smith. An energy-minimizing interpolation for robust multigrid methods. *SIAM Journal on Scientific Computing*, 21(4):1632–1649, 1999.

[210] R. Webster. An algebraic multigrid solver for Navier-Stokes problems. *International Journal for Numerical Methods in Fluids*, 18(8):761–780, 1994.

[211] P. Wesseling. *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*, chapter A robust and efficient multigrid method, pages 614–630. Springer, 1982.

[212] P. Wesseling. *An introduction to multigrid methods*. Pure and applied mathematics. John Wiley & Sons Australia, Limited, 1992. ISBN 9780471930839.

[213] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer, Heidelberg, 2000.

[214] P. Wesseling and C. W. Oosterlee. Geometric multigrid with applications to computational fluid dynamics. *Journal of Computational and Applied Mathematics*, 128(1-2):311–334, 2001.

[215] C. Wieners and B. Wohlmuth. Duality estimates and multigrid analysis for saddle point problems arising from mortar discretizations. *SIAM Journal on Scientific Computing*, 24: 2163–2184, 2003.

[216] C. Wieners and B. Wohlmuth. A general framework for multigrid methods for mortar finite elements. Technical Report 415, Universität Augsburg, Institut für Mathematik, 1999.

[217] T. A. Wiesner, R. S. Tuminaro, W. A. Wall, and M. W. Gee. Multigrid transfers for nonsymmetric systems based on Schur complements and Galerkin projections. *Numerical Linear Algebra with Applications*, 21:415–438, 2013.

[218] G. Wittum. Linear iterations as smoothers in multigrid methods: theory with applications to incomplete decompositions. *IMPACT of Computing in Science and Engineering*, 1(2): 180–215, 1989.

[219] G. Wittum. On the robustness of ILU smoothing. *SIAM Journal on Scientific and Statistical Computing*, 10(4):699–717, 1989.

[220] H. Wobker and S. Turek. Numerical studies of Vanka–type smoothers in computational solid mechanics. *Advances in Applied Mathematics and Mechanics*, 1(1):29–55, 2009.

[221] B. Wohlmuth. Hierarchical a posteriori error estimators for mortar finite element methods with Lagrange multipliers. *SIAM Journal on Numerical Analysis*, 36:1636–1658, 1999.

[222] B. Wohlmuth. Multigrid methods for saddlepoint problems arising from mortar finite element discretizations. *ETNA. Electronic Transactions on Numerical Analysis*, 11:43–54, 2000.

[223] B. Wohlmuth. *Discretization Techniques and Iterative Solvers Based on Domain Decomposition*, volume 17. Springer, Heidelberg, 2001.

[224] B. Wohlmuth. Variationally consistent discretization schemes and numerical algorithms for contact problems. *Acta Numerica*, 20:569–734, 5 2011.

[225] B. Wohlmuth. A mortar finite element method using dual spaces for the Lagrange multiplier. *SIAM Journal on Numerical Analysis*, 38:989–1012, 2000.

[226] B. Wohlmuth. A V-cycle multigrid approach for mortar finite elements. *SIAM Journal on Numerical Analysis*, 42:2476–2495, 2005.

[227] B. Wohlmuth and R. Krause. Multigrid methods based on the unconstrained product space for mortar finite element discretizations. *SIAM Journal on Numerical Analysis*, 39: 192–213, 2001.

[228] P. Wriggers. *Computational contact mechanics*. Springer-Verlag Berlin Heidelberg, 2006.

[229] P. Wriggers, T. Vu Van, and E. Stein. Finite element formulation of large deformation impact-contact problems with friction. *Computers & Structures*, 37(3):319–331, 1990.

[230] B. Yang and T. Laursen. A contact searching algorithm including bounding volume trees applied to finite sliding mortar formulations. *Computational Mechanics*, 41(2):189–205, 2008.

[231] H. Yang and W. Zulehner. Numerical simulation of fluid–structure interaction problems on hybrid meshes with algebraic multigrid methods. *Journal of Computational and Applied Mathematics*, 235(18):5367–5379, 2011.

[232] G. Zavarise and P. Wriggers. A segment-to-segment contact strategy. *Mathematical and Computer Modelling*, 28(4–8):497 – 515, 1998. Recent Advances in Contact Mechanics.

[233] P. D. Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *Journal of Computational and Applied Mathematics*, 33(1):1 – 27, 1990.

[234] W. Zulehner. A class of smoothers for saddle point problems. *Computing*, 65:227–246, 2000.

[235] W. Zulehner. Analysis of iterative methods for saddle point problems: A unified approach. *Mathematics of Computation*, 71:479–505, 2002.

Reports of the Institute for Computational Mechanics at Technische Universität München

24 (2014)	**Jakob Huemer:** Einfluss instationärer aerodynamischer Kräfte auf die Fahrdynamik von Personenkraftwagen.
23 (2014)	**Robert Metzke:** Modeling and experimental investigation of the mechanobiological environment associated with alveolar pneumocytes.
22 (2014)	**Shadan Shahmiri:** A hybrid ALE-fixed-grid approach for fluid-structure interaction.
21 (2014)	**Caroline Danowski:** Computational modelling of thermo-structure interaction with application to rocket nozzles.
20 (2014)	**Kei Müller:** Simulation of self-assembly and mechanics of transiently crosslinked, semiflexible biopolymer networks.
19 (2014)	**Mahmoud Ismail:** Reduced dimensional modeling of the entire human lung.
18 (2013)	**Florian Henke:** An extended finite element method for turbulent premixed combustion.
17 (2012)	**Markus Gitterle:** A dual mortar formulation for finite deformation frictional contact problems including wear and thermal coupling.
16 (2012)	**Andreas Maier:** Computational modeling of rupture risk in abdominal aortic aneurysms.
15 (2012)	**Georg Bauer:** A coupled finite element approach for electrochemical systems.
14 (2012)	**Alexander Popp:** Mortar methods for computational contact mechanics and general interface problems.
13 (2012)	**Thomas Klöppel:** A finite element model for the human red blood cell.
12 (2012)	**Sophie Rausch:** Computational and experimental modeling of lung parenchyma.

11 (2011)	**Christian Cyron:** Micromechanical continuum approach for the analysis of biopolymer networks.
10 (2011)	**Lena Wiechert:** Computational modeling of multi-field and multi-scale phenomena in respiratory mechanics.
9 (2010)	**Peter Gamnitzer:** Residual-based variational multiscale methods for turbulent flows and fluid-structure interaction.
8 (2010)	**Axel Gerstenberger:** An XFEM based fixed grid approach to fluid-structure interaction.
7 (2009)	**Ulrich Küttler:** Effiziente Lösungsverfahren für Fluid-Struktur-Interaktions-Probleme.
6 (2009)	**Moritz Frenzel:** Advanced structural finite element modeling of arterial walls for patient-specific geometries.
5 (2007)[1]	**Christiane Förster:** Robust methods for fluid-structure interaction with stabilised finite elements.
4 (2004)[1]	**Tobias Erhart:** Strategien zur Numerischen Modellierung transienter Impaktvorgänge bei nichtlinearem Materialverhalten.
3 (2004)[1]	**Michael Gee:** Effiziente Lösungsstrategien in der nichtlinearen Schalenmechanik.
2 (2003)[1]	**Volker Gravemeier:** The variational multiscale method for laminar and turbulent incompressible flow.
1 (2001)[1]	**Daniel Mok:** Partitionierte Lösungsverfahren in der Strukturdynamik und der Fluid-Struktur-Interaktion.

[1]This dissertation was supervised by Prof. Dr.-Ing. Wolfgang A. Wall at the Institute for Structural Mechanics at the University of Stuttgart and is published in the respective report series.