AMG Strategies for PDE Systems with Applications in Industrial Semiconductor Simulation

I n a u g u r a l - D i s s e r t a t i o n
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Tanja Clees
aus Aachen

2005

Berichterstatter:	Prof. Dr. Ulrich Trottenberg Prof. Dr. Rüdiger Seydel
Tag der mündlichen Prüfung:	30. November 2004

Vorwort des Institutsleiters

Die Simulation der Eigenschaften von Materialien und der Dynamik von Systemen spielt für die Industrie bei der Entwicklung neuer Prozesse und Produkte bereits heute eine entscheidende und nach wie vor wachsende Rolle. Dies zeigt sich etwa in so unterschiedlichen Bereichen wie der Mikrochip-Produktion, der Ölförderung oder dem Flugzeugbau. Durch Simulationen lassen sich zwar teure Experimente und Prototypen in der Regel nicht vollständig ersetzen, jedoch ermöglichen sie, prinzipielle Designentscheidungen schneller und sicherer zu fällen und so Entwicklungszeit und -kosten zu reduzieren. Dies kann zu entscheidenden Wettbewerbsvorteilen führen.

Oftmals liegen stark gekoppelte Systeme partieller Differentialgleichungen (Systeme von PDGln.) den Simulationen zu Grunde. Die bei ihrer numerischen Lösung auftretenden großen dünnbesetzten Gleichungssysteme verbrauchen häufig den größten Teil der Gesamtrechenzeit einer Simulation. Daher ist die Entwicklung schneller Löser für diese Gleichungssysteme meist von größter praktischer Bedeutung. Standardlöser sind aber nicht effizient genug für sehr große Matrizen. Ihr Rechenaufwand skaliert für viele wichtige Anwendungen nicht mit der Anzahl der Variablen des Gleichungssystems.

Für viele in der Praxis relevante Problemklassen stellen algebraische Mehrgitterverfahren (AMG) robuste und effiziente *skalierbare* Löser (oder Vorkonditionierer) dar. Allerdings kann AMG ohne umfassende Erweiterungen stark gekoppelte Systeme von PDGln. nicht effizient lösen. Für viele wichtige Systeme von PDGln. wurden bisher noch keine geeigneten AMG-Verfahren entwickelt. Überdies fehlte bisher ein Löser-Softwarepaket, welches industriellen Ansprüchen genügt.

Diese Dissertation liefert hierzu wichtige Beiträge. Sie entstand am Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen (SCAI) in der Abteilung “Numerische Software” sowie am Mathematischen Institut der Universität zu Köln. Das Fraunhofer-Institut SCAI zeichnet sich durch eine über zwanzigjährige Expertise auf dem Gebiet der geometrischen und algebraischen Mehrgitterverfahren aus. Insbesondere wurde hier die erste öffentlich verfügbare algebraische Mehrgitter-Software (AMG1R5) entwickelt, welche weltweit verbreitet und - obwohl längst veraltet und nicht für sehr große Gleichungssysteme entwickelt - auch heute noch tausendfach im Einsatz ist. Das Institut hat überdies in den letzten Jahren eine neue AMG-Software, die Löserbibliothek “SAMG” entwickelt, welche ganz auf die Ansprüche der Industrie ausgerichtet ist.

Ein Hauptbeitrag der vorliegenden Dissertation ist die Erweiterung von SAMG zur hocheffizienten numerischen Lösung praktisch relevanter, diskreter Systeme von PDGln. Insbesondere für drei wichtige Anwendungsklassen aus der industriellen Halbleitersimulation, die große numerische Herausforderungen darstellen, werden effiziente AMG-Verfahren entwickelt. Zwei der drei Klassen wurden bisher noch nicht erfolgreich mit AMG-Verfahren behandelt. Dank der im Rahmen der Dissertation realisierten Erweiterungen ist SAMG bereits heute für viele Systeme von PDGln. den industriellen Anforderungen gewachsen und bei Kunden im Einsatz.

Ulrich Trottenberg

Abstract

The numerical solution of strongly coupled systems of partial differential equations (PDE systems) is commonplace in many simulation codes. Typically, large sparse matrix equations arise in the corresponding simulation runs. A serious bottleneck in performing realistic, large-scale simulations is the speed by which these matrix equations can be solved. If they exceed a certain size, they can no longer be solved efficiently with standard numerical solvers, simply because these solvers are not scalable.

Classical algebraic multigrid (AMG) approaches are known to provide robust and efficient, *scalable* solvers or preconditioners for large classes of matrices as those typically arising from *scalar* PDE systems. However, because classical AMG is based on a so-called variable-based approach which does not distinguish between physical unknowns, extensions of classical AMG are required to efficiently solve *systems* of PDEs. In general, many important types of PDE systems have not been tackled by any AMG approach yet. Moreover, an "AMG software" suitable for many industrially relevant problems has been missing so far. This PhD thesis makes the following important contributions.

We develop a general AMG methodology which is suitable for important classes of industrially relevant PDE systems. Our AMG methodology extends classical AMG by a straightforward unknown- and a particularly powerful point-based strategy. In particular, a general concept for point-based approaches is introduced, which employs a primary matrix to construct a point-based coarsening. Several possibilities for selecting a primary matrix and for constructing the interpolation are discussed from a theoretical and, with special emphasis, a practical point of view.

We realize our AMG methodology within the product-quality solver library SAMG. In particular, we demonstrate that, in practice, all (accelerated) AMG approaches being part of SAMG are scalable if applied to proper classes of applications. Memory requirements are reasonable compared to the requirements of standard one-level preconditioners such as ILU(0). SAMG can easily be plugged into existing simulation codes and provides a rich environment allowing for many different AMG approaches.

We demonstrate the generality and flexibility of the proposed AMG methodology as well as the efficiency of concrete SAMG approaches for a variety of PDE systems. In particular, three important classes of applications arising in industrial semiconductor process and device simulation are discussed, namely stress analysis (linear elasticity problems), reaction-diffusion and drift-diffusion simulation. Reaction-diffusion and, in particular, drift-diffusion systems are numerically very challenging applications which have not been solved before by any AMG approach. For each application, it is shown by means of both heuristical justifications as well as numerical results that SAMG allows to construct robust and efficient AMG approaches even for cases, where state-of-the-art one-level solvers employed in standard simulation codes exhibit bad convergence or even fail.

Key words: algebraic multigrid (AMG), systems of partial differential equations (PDE systems), unknown-based approach, framework of point-based approaches, semiconductor process and device simulation, linear elasticity, stress analysis, reaction-diffusion systems, drift-diffusion systems.

Zusammenfassung

Die numerische Lösung stark gekoppelter Systeme partieller Differentialgleichungen (Systeme von PDGln.) ist allgegenwärtig in vielen Simulationscodes. Üblicherweise müssen sehr große dünnbesetzte Matrixgleichungen in den entsprechenden Simulationsläufen gelöst werden. Die Geschwindigkeit, mit der diese Gleichungen gelöst werden können, entscheidet wesentlich über die Größe und damit Wirklichkeitsnähe der Simulationen. Da Standardlöser nicht skalieren, sind sie aber nicht effizient für sehr große Matrizen.

Klassische algebraische Mehrgitterverfahren sind bekanntermaßen robuste und effiziente *skalierbare* Löser oder Vorkonditionierer für große Matrizenklassen, wie sie typischerweise von *skalaren* PDGln. herrühren. Da klassisches AMG auf einem sogenannten variablenbasierten Ansatz beruht, welcher nicht zwischen physikalischen Unbekannten unterscheidet, sind Erweiterungen nötig, um auch *Systeme* von PDGln. effizient lösen zu können. Generell sind AMG-Ansätze für viele wichtige Systeme von PDGln. noch nicht erfolgreich gewesen. Überdies fehlte bisher eine “AMG-Software”, die sich für viele industriell relevante Probleme eignet. Diese Dissertation liefert folgende wichtige Beiträge.

Wir entwickeln eine generelle AMG-Methodologie, die sich auf viele relevante Systeme von PDGln. anwenden lässt. Unsere Methodologie erweitert klassisches AMG durch eine naheliegende unbekannten- sowie eine besonders wirkungsvolle punktbasierte Strategie. Insbesondere wird ein generelles Konzept für punktbasierte Ansätze eingeführt, welches eine primäre Matrix zur Konstruktion einer punktbasierten Vergröberung verwendet. Mehrere Möglichkeiten für die Auswahl der primären Matrix und der Interpolation werden sowohl von einem theoretischen als auch - schwerpunktsmäßig - einem praktischen Standpunkt diskutiert.

Wir realisieren unsere AMG-Methodologie in der marktreifen Löserbibliothek SAMG. Insbesondere demonstrieren wir, dass in der Praxis alle (beschleunigten) AMG-Verfahren für geeignete Problemklassen skalieren. Der Speicherverbrauch ist dabei sehr moderat im Vergleich zu Standard-Einlevel-Vorkonditionierern wie etwa ILU(0). SAMG lässt sich einfach in existierende Simulationsprogramme einbauen und bietet eine sehr variantenreiche AMG-Umgebung.

Wir demonstrieren die Allgemeinheit und Flexibilität der vorgeschlagenen AMG-Methodologie sowie die Effizienz konkreter SAMG-Verfahren für eine Vielzahl von Systemen von PDGln. Insbesondere werden drei wichtige Anwendungen aus der industriellen Halbleitersimulation diskutiert, nämlich Stress-Analyse (lineare Elastizität), Reaktions-Diffusions- und Drift-Diffusions-Systeme. Die letzten zwei und hierbei insbesondere die letzte Anwendung stellen große numerische Herausforderungen dar und wurden bisher noch nicht erfolgreich mit AMG-Verfahren gelöst. Für jede Anwendung zeigen wir anhand von Heuristika und numerischen Resultaten, dass SAMG die Konstruktion robuster und effizienter AMG-Verfahren erlaubt, und das sogar für Fälle, wo die typischerweise in Standard-Simulationsprogrammen eingesetzten Einlevel-Löser schlecht konvergieren oder sogar fehlschlagen.

Schlagwörter: algebraisches Mehrgitter (AMG), Systeme partieller Differentialgleichungen, unbekanntenbasierter Ansatz, Umgebung für punktbasierte Ansätze, Halbleiter-Prozess- und Device-Simulation, lineare Elastizität, Reaktions-Diffusions-Systeme, Drift-Diffusions-Systeme.

Contents

Chapter 1

Introduction

For the development of novel technologies, industry is increasingly relying on computer-aided engineering. This is true, for example, for such different areas as the design of microchips, efficient oil production or the construction of aircrafts. However, the complex process of designing, testing and optimizing new processes and products usually has to be carried out in an iterative and more experimental fashion by means of time-consuming trial-and-error steps with expensive prototypes. In order to reduce design time and production costs, computer simulation thus gains a growing importance. Today, simulation is not able to replace the experimental process but is able to assist it in such a way that principal design decisions can be made faster and less prototypes are needed.

In various application fields, simulation is used to analyze the structure and dynamics of material systems. The underlying models involve physical quantities, also called *unknowns* in the following, such as material displacements, concentrations, potentials or pressures, and usually consist of one or more partial differential equations (PDEs) which have to be solved numerically. This is done by discretizing and linearizing the PDEs and solving the arising sparse matrix equations by direct or iterative linear solvers.

With the growing complexity of fabrication technologies and resulting products, a larger effort has to be invested in the simulations to meet the requirements of increasing accuracy and to gain an ever deeper insight into the governing forces. On one hand, the demand for higher accuracy has led to the use of more accurate discretization schemes and more complex and finer discretization grids. Today, truly three-dimensional locally refined unstructured finite element or finite volume meshes with up to some millions of grid nodes are commonplace for many applications. Their grid resolution will significantly grow as soon as computer memory resources will allow this. Unfortunately, with an increasing grid resolution increasingly large matrix equations have to be solved. On the other hand, the higher accuracy requirements are more and more leading to physically complex models involving strongly coupled PDE systems, reflecting the fact that typically several physical unknowns strongly depend on each other and cannot be considered separately. As a consequence, such a PDE system has usually to be solved simultaneously for all unknowns involved which, together with fine grids, results in matrix equations with up to several millions of variables. To be more specific, if the PDE system is nonlinear and/or time-dependent, a whole series of such huge matrix equations has to be solved. This is frequently the case.

The solution of huge sparse matrix equations usually belongs to the computationally very expensive parts of a simulation. Often, it is even the by far most expensive part. Therefore, any reduction in the linear system solution time will result in a significant saving in the total simulation time. This is a strong motivation for the intensive research activities in the field of

linear solvers. However, due to the trends described above, the arising matrices are not only increasingly large. The more complex the physical models and corresponding PDE systems, the more difficult to solve them efficiently. It seems clear that for a linear solver to be efficient it has to address the physics of the underlying problem as well as the numerical properties of the PDE system appropriately. Each particular type of PDE system can have its own set of particular difficulties. This aggravates the demand for efficient linear solvers for a broad spectrum of applications even further.

A **scalable** solver is characterized by a complexity[1] of $O(N)$ regarding both memory requirements *and* computational work. Though scalability is the most important property an efficient matrix solver should have, its **practical relevance** is a user-defined measure weighing applicability to and robustness for one or more large matrix classes and optimal numerical complexity among each other. The "wish list" of practitioners is even longer. In particular, an optimally efficient and robust linear matrix solver would fulfill all of the following properties: it would

- exhibit a complexity of $O(N)$ in terms of both memory requirements and computational work with small "$O(N)$'s constants";
- be able to handle industrially relevant matrix equations with up to millions of variables, independent of dimension and type of the underlying grid etc.;
- be able to solve large classes of such applications robustly;
- be simple to use, most preferably as a "black box";
- be simple to plug into existing simulation codes.

Clearly, not all of the points can be fulfilled simultaneously. For example, the more generally applicable, the less efficient a solution approach has to be expected to be. This is especially the case for the standard linear solvers used today, of both direct and one-level iterative type, which can be applied to a wide range of problems but cannot keep up with the increasing size and/or numerical difficulties of the problems. On one hand, direct solvers (sparse Gaussian elimination) can be applied very generally but generally exhibit a complexity of up to $O(N^3)$ w.r.t. their computational work and a complexity of up to $O(N^2)$ w.r.t. their memory requirements, which makes their use prohibitive for large matrices. On the other hand, one-level iterative solvers are less generally applicable. They usually exhibit a linear complexity w.r.t. memory requirements, but strongly suffer from large condition numbers of the matrices. Typically, they exhibit a complexity of $O(N^\alpha)$ ($\alpha > 1$) w.r.t. computational work and are likely to fail for very ill-conditioned problems.

Since problem sizes are substantially growing, optimal complexity, $O(N)$, in particular regarding computational work, will be even more of a concern in the future. For the applications sketched above, this optimality can be reached by approaches employing numerical information resulting from a hierarchy of grids (levels, scales). Various optimal hierarchical approaches exist, called multigrid, multilevel or multiscale approaches, each of them suitable for a certain range of problems.

Classical algebraic multigrid (AMG) [71, 87] is known to provide very efficient and robust solvers or preconditioners for large classes of matrix problems, $Av = b$, an important

[1] A linear solver is said to exhibit a **(numerical) complexity of** $O(N^\alpha)$ w.r.t. computational work (w.r.t. memory requirements), if - *for a fixed relative residual reduction* - its computational work (its memory requirements) scale(s) proportionally with N^α where N is the number of variables of the linear system of equations.

one being the class of sparse linear systems with matrices A which are "close" to being M-matrices. Problems like this widely occur in connection with discretized *scalar* elliptic partial differential equations (PDEs). In such cases, classical AMG is very mature and can handle millions of variables much more efficiently than any one-level method, a main reason being its optimal complexity of $O(N)$. Since explicit information on the geometry (such as grid data) is not needed, AMG is especially suited for unstructured grids both in 2D and 3D. In fact, only the matrix A and its right-hand side b have to be passed to an AMG solver since the construction of a reasonable multilevel hierarchy is part of the AMG algorithm, automatically performed by exploiting easily accessible algebraic properties such as the size and sign of matrix entries. Consequently, such an AMG solver is as easy to plug into an existing simulation code as any standard one-level solver, it has (nearly) black-box quality, and - altogether - already fulfills many of the "wishes" mentioned above.

Contents and Contributions of this Thesis However, extensions of classical AMG are required to efficiently solve *systems* of PDEs involving two or more scalar physical unknowns. This is because it is based on a so-called variable-based approach which does not distinguish between different unknowns. Unless the coupling between these unknowns is very weak, such an approach cannot work efficiently for PDE systems where, in general, the corresponding matrix A is far from being an M-matrix.

In the past, several ways to generalize classical AMG or other AMG approaches have been investigated, and there is still an ongoing rapid development of new AMG and AMG-like approaches. Regarding PDE systems, development has predominantly focused on specialized solvers for narrow classes of applications as, for instance, certain CFD (computational fluid dynamics) and linear elasticity problems where promising progress has been made. However, there is no unique and best approach yet, and besides the developments mentioned, AMG approaches have not been investigated for PDE systems. Some of the existing approaches rely on incisive conditions and cannot be generalized to larger application classes, at least not in an obvious way. Others have the potential to be more generally applicable if generalized or extended appropriately. However, this has not been done so far. As a consequence, none of the existing approaches is really satisfactory in dealing with larger classes of practically relevant problems, and many industrially relevant problems have not been tackled at all yet. Moreover, a software (library) which realizes more generally applicable, efficient AMG approaches, which can easily be plugged into existing simulation codes, and which is easy to use would highly be appreciated by the industry but has not been available so far.

This thesis adresses these gaps and makes the following important contributions for filling them at least for many important applications:

- *the development of a general AMG methodology which is suitable for important classes of industrially relevant PDE systems*

  We generalize the efficient classical "scalar" AMG methodology by employing "natural" structural information on a discrete PDE system to be solved, namely relationships between so-called variables, unknowns and points. Rather than a single method this will give us a *flexible, general methodology* capable of providing very efficient precon-

ditioners *not for all but large classes* of practically relevant PDE systems. Among them will be systems which have not been successfully solved by any other AMG approach.

Our methodology systematically extends classical scalar AMG, representing what we call the *variable-based strategy* in the following, by a straightforward *unknown*- and a particularly powerful *point-based strategy*.

Unknown-based AMG (UAMG) [2] is very similar to variable-based AMG (VAMG) except that for each physical unknown its own hierarchy is created. UAMG can handle anisotropies which are different from unknown to unknown. The unknowns are allowed to live on a different grid each (staggered grids, for instance). Among the basic conditions for this strategy to work is that the coupling between the different unknowns is not "too strong". In this thesis, we will introduce a measure for the strength of unknown cross-couplings.

For many important PDE systems, the unknown cross-couplings are indeed too strong for UAMG. Since for many such systems the different unknowns are discretized on essentially the same "grid", it appears to be quite natural to create the same hierarchy for all unknowns. A strategy which allows for strong unknown cross-couplings and produces the same level hierarchy, is point-based AMG (PAMG) [3]. We develop a general framework for PAMG approaches, which is the most important part of our methodology. One key concept for PAMG is that point-coarsening is performed by means of an auxiliary so-called *primary matrix*. A necessary condition for the PAMG framework to be applicable is that a primary matrix can be defined wich reflects the point couplings in a reasonable sense. We introduce and discuss various ways to define concrete primary matrices as well as three general types of interpolation approaches. Our focus is on the development of practical variants, that is variants which are computationally cheap ("$O(N)$ with a small constant"), efficient and applicable to relevant and sufficiently large classes of PDE systems.

Convergence of our AMG approaches is proved under the assumption that A is symmetric positive definite. We want to emphasize here that, in practice, this is not a necessary condition. For instance, the application of PAMG to very asymmetric drift-diffusion systems impressively demonstrates that AMG can efficiently work for considerable deviations from the "ideal" case.

- *a realization of this methodolgy within the product-quality solver library SAMG*

  We realize our general AMG methodology within the solver library SAMG. The result is a rich AMG environment with various concrete components. SAMG provides highest flexibility for adaptations to very different situations arising in practice and, if necessary, can easily be extended by the user even further.

  We will demonstrate that SAMG can solve matrices arising from many different industrially relevant classes of PDEs and PDE systems efficiently and robustly. We will

[2] Unknown-based AMG has already been introduced in the early paper [71].

[3] The basic idea, that is a "simultaneous" coarsening (and interpolation) of the unknowns, has already been outlined in the early papers [71, 8].

also demonstrate that, for these problem classes, SAMG meets industrial needs of scalability, robustness, (nearly-)black-box usage and plug-in-type integration into existing codes, as indicated in the "wish list" above. Moreover, SAMG provides a far more efficient and robust behavior for many problem classes than the standard one-level solvers usually employed in industrial simulation codes.

- *a demonstration of SAMG's efficiency and robustness for challenging industrial applications*

  By means of real-life applications, we will show the generality and flexibility of the proposed overall AMG methodology as well as the efficiency of concrete SAMG approaches for a variety of PDE systems. To be more specific, as a demonstration of UAMG's and especially PAMG's "practical relevance", three important applications in industrial semiconductor process and device simulation are discussed, namely stress analysis (linear elasticity problems), reaction-diffusion and drift-diffusion simulation. For each application, it will be shown by means of both heuristical justifications as well as numerical results for relevant test cases that the usage of SAMG leads to a solution process which, compared to state-of-the-art one-level solvers employed in standard simulation codes, is very promising both in terms of robustness and efficiency. Reaction-diffusion and, in particular, drift-diffusion systems are numerically very challenging applications which have not been tackled before by any AMG approach successfully.

- *forging links between applied mathematics and industrial application*

  This thesis covers the whole process from the development of a general AMG methodology over its product-quality software realization to its application to industrially relevant problems. Our flexible, general methodology considerably extends AMG's applicability to practically relevant PDE systems, and its realization SAMG meets industrial needs. Especially SAMG is an important contribution to forging links between applied mathematics and industrial application.

The thesis is organized as follows. In **Chapter 2** we make a general characterization of AMG and give an overview of specific approaches, in particular with respect to a stocktaking of what has been achieved so far for solving PDE systems. In addition, basic notations and definitions which are frequently used throughout this thesis are summarized. In the following two chapters, our AMG methodology and its three general strategies, namely the variable-based, the unknown-based and the point-based one, are explained in detail. Whereas **Chapter 3** introduces this methodology and its three strategies from a more theoretical point of view, including a discussion of the range of applicability and limitations, **Chapter 4** gives an overview of the concrete realization of our AMG methodology within the Fortran90 library SAMG. In these two chapters, we discuss, in particular, the choice of suitable AMG components and the performance of resulting approaches for three different classes of model problems. To be more specific, we consider anisotropic vector Laplacians, reaction-diffusion-like models and drift-diffusion-like models. They represent, in particular, some important properties of the discrete PDE systems discussed in Chapter 5 in a simplified, "concentrated" way. In **Chapter 5**, SAMG's efficiency and robustness is demonstrated for stress analysis, reaction-diffusion systems and drift-diffusion systems arising in industrial semiconductor

process and device simulation. We conclude this thesis in **Chapter 6** and give an outlook on future research. Finally, **Appendix A** contains some additional aspects, outsourced for better readability.

Remark: Parts of this thesis, namely a short introduction into the point-based framework and some results for semiconductor process and device simulation, have been published in [27, 24, 25, 26, 19, 18].

Acknowledgments

I owe sincere thanks to Prof. Dr. Ulrich Trottenberg for promoting my work and research at the Fraunhofer Institute for Algorithms and Scientific Computing (SCAI) and for supervising this Ph.D. thesis. To work at SCAI gives me the chance to combine my strong interests in applied mathematics and industrial application in an ideal way.

Very special thanks go to Dr. Klaus Stüben for his continuous support of my work and especially of this thesis; for all the inspiring and fruitful discussions with one of *the* AMG experts, and also for the great effort he spent with proofreading my thesis.

I wish to thank Prof. Dr. Rudolph Lorentz and the Ebel family for proofreading parts of this thesis.

Prof. Dr. Rüdiger Seydel is gratefully acknowledged for beeing co-examiner of this thesis.

The Delft Institute of Microelectronics and Submicrontechnology (DIMES) at the TU Delft, the Integrated Systems Laboratory at the ETH Zürich, and the company Synopsys Inc. provided interfaces to their simulation software and suitable test examples. Their support has made the strong industrial focus of this thesis possible. I would like to thank, in particular, PD. Dr. Slobodan Mijalkovic, PD. Dr. Andreas Pomp, Dr. Bernhard Schmithüsen, Stefan Röllin, Dr. Norbert Strecker (special thanks!), Dr. Andrey Kucherov, the group of Prof. Dr. Siegfried Selberherr at the TU Wien (institute for microelectronics) and Stephan Wagner (special thanks!) for their open-mindedness for new approaches, their hospitality, their kind help, and for many explanations and fruitful discussions.

I wish to thank many current and former colleagues at SCAI for being such great colleagues and for their encouragement and support during all stages of this thesis. *Fortunately,* I cannot list all of them here.

I thank all members of the Graduiertenkolleg “Scientific Computing” for all the interesting talks and discussions.

Finally, I would like to thank my husband Uli, my parents Yvonne and Jo, our whole family and our friends for so much more than words can express.

This research was supported by a Grant from the G.I.F., the German-Israeli Foundation for Scientific Research and Development.

Chapter 2

Fundamentals, Approaches, Notation

The demand for optimally scaling, more robust and more generally applicable iterative linear solvers has been driving the development of multigrid approaches for more than 30 years now. Two general types of multigrid exist, geometric multigrid (GMG) and algebraic multigrid (AMG). Today, both represent large classes of approaches, and additionally there are many "hybrid" approaches which exhibit some characteristica of GMG or AMG or both.

Classical articles or books about geometric multigrid methods include [7, 91, 34]. For a comprehensive survey of modern geometric multigrid, the reader is referred to the monograph [94], also containing [87] in which an in-depth introduction to algebraic multigrid is given. [14] reviews "robust" multigrid and [88, 103] review AMG, including references to many important related multi-level approaches. A survey of iterative methods with a special emphasis on accelerators can be found in [76]. The very recent [3] gives a survey on preconditioners including (multi-level) ILU-type methods as well as sparse approximate inverses. We will briefly review important approaches and relationships to AMG in Section 3.2.6.

The purpose of this chapter is threefold. Firstly, Section 2.1 summarizes fundamental principles which form the basis of concrete AMG approaches and characterizes the current status of AMG on the general point of view of someone comparing iterative matrix solvers.

Secondly, Section 2.2 gives a survey of the two different AMG methodologies that can be found today, namely classical and aggregation-based AMG, as well as extensions, new developments and current research activities. In particular, we survey which AMG approaches - besides our general AMG methodology - are already available for discrete PDE systems.

In the third part of this chapter, Sections 2.3 and 2.4, we introduce important general notation and definitions used in the subsequent chapters. In particular, we define the notation of the formal components of each approach belonging to our AMG methodology.

Remark 2.1 The reader is assumed to be familiar with the basics of multigrid, in particular with the two fundamental principles, namely *smoothing* and *coarse-grid correction*, and with the general multigrid cycling (see [94], for instance). ▲

Remark 2.2 Note that, unless explicitly stated otherwise, the term "AMG" stands for "classical AMG" throughout this thesis. A general exception from this rule is Section 2.2. ▲

Remark 2.3 We only investigate PDE systems in real space and, accordingly, only real matrices. It is assumed that the given matrix A is nonsingular and all diagonal entries are positive unless explicitly stated otherwise. ▲

2.1 Fundamentals

Roughly speaking, two basic observations motivated the development of geometric multigrid. One the one hand, when applied to a discrete elliptic problem, classical iterative methods like Gauss-Seidel relaxation converge very slowly but smooth[1] the error quickly. On the other hand, a smooth error can be well represented on a coarser grid where a reduction of its low-frequency components, causing the slow convergence, can be performed with less computational effort. These two observations constitute the fundamental multigrid principles, *smoothing* and *coarse-grid correction*, which - translated or generalized appropriately - lay the foundation for all multigrid approaches, geometric as well as algebraic ones. The interplay of both principles as realized within a concrete approach determines the efficiency of this approach.

Each GMG and AMG is not a single method but a methodology, representing a whole class of different concrete approaches. Moreover, there are two general AMG concept, *classical AMG* and *aggregation-based AMG*, each constituting even a whole methodology on its own. Particularly these two methodologies are briefly explained and reviewed in Section 2.2.1. The aim of this section is to survey important principal aspects of *classical AMG* as are revealed to someone comparing GMG, AMG and one-level iterative solvers from a very general point of view. It should be noted that these aspects can principally be translated to the aggregation-based AMG methodology.

We start with a characterization of robust multigrid (Section 2.1.1) mainly in order to motivate why we develop AMG approaches. We explain why GMG cannot fulfill our goals formulated in Chapter 1, in particular the goal of developing plug-in matrix solvers. In the general characterization of AMG (Section 2.1.2), we point out similarities and differences to GMG, explain why AMG development is a reasonable way for fulfilling at least important parts of our goals, indicate open questions and perform a rough classification of AMG in the circle of iterative matrix solvers. The aspects briefly mentioned in Section 2.1.2 will be explained in more detail in the following chapters.

2.1.1 Robust Geometric Multigrid

Geometric multigrid interpretes the two principles, smoothing and coarse-grid correction, in their original, geometric sense. It aims at solving grid-based equations,

$$L_h v_h = b_h \quad \text{on } \Omega_h \,, \tag{2.1}$$

where L_h denotes a (finite-difference) operator and v_h and b_h functions defined on a grid or, synonymously, mesh[2] Ω_h. Typically, (2.1) represents an elliptic partial differential equation (with boundary conditions) discretized on Ω_h. Most typically, Ω_h is a structured grid.

The main property of all GMG approaches is that they operate on a predefined hierarchy of grids, in standard cases (rectangular meshes) obtained by a simple coarsening[3] process

[1] The term "smooth" is meant relatively to the underlying discretization grid here.

[2] h refers to a "grid parameter", related to a mesh size.

[3] We define "coarsening" as the process of constructing the next coarser grid (or level). The construction of the intergrid transfer operators is regarded as a separate process.

as, for example, by doubling the mesh size in each direction ("$h \rightarrow 2h$"). Other classical components are Gauss-Seidel relaxation for smoothing, straightforward geometric intergrid transfer operators (bi-/tri-linear interpolation and restriction by injection or full weighting) and coarse-grid operators which are analogs of the finest-grid difference operator L_h. With these components, multigrid cycles (typically V-, F- or W-cycles) are performed through this hierarchy in an iterative manner to obtain an approximation of the solution v_h.

Straightforward multigrid components of the above type make GMG most suitable for isotropic problems on structured grids, the classical and simplest model problem being Poisson's equation, discretized by the standard five-point stencil on the unit square. However, for more complex applications, these simple and purely geometry-based components have to be improved or replaced. Difficulties are caused by, for example, non-uniform smoothing (as for anisotropic equations on standard-coarsened grids) and insufficient correction of error components on coarser grids (as for diffusion equations with strongly varying coefficients).

Since smoothing and coarse-grid correction are required to interact efficiently, fixing the grid hierarchy means tuning the components listed above. Among the major steps towards increasing the robustness of geometric multigrid and extending the range of applicability were the development of

- operator-dependent interpolation in combination with a Galerkin-based[4] coarse-grid correction process, originally developed to treat diffusion equations with discontinous coefficients,
- more complex smoothers if simple coarsening strategies shall be employed, examples being ILU-type smoothers or alternating line relaxation for two-dimensional anisotropic equations, alternating plane relaxation for three-dimensional ones,
- sophisticated coarsening techniques if simple smoothers shall be employed, for instance semi-coarsening in multiple directions for anisotropic equations.

The applicability of such more sophisticated techniques is relatively straightforward in regular-grid applications. However, with the exception of the first point and the ILU smoothing, these techniques can hardly be realized on less structured meshes, in particular in 3D. This is also true for each multigrid component which is geometrically constructed, even if the PDE to be solved is "simple". In particular, the more complex a grid the more difficult is the definition of suitable coarser grids just by exploiting geometric considerations, and in case of unstructured finite element (FE) or finite volume (FV) meshes this is hardly ever feasible. Thus, predefining the grid hierarchy is one of the crucial points in applying GMG to "real-life" applications.

In summary, GMG approaches can be highly efficient iterative solvers for a variety of concrete cases. However, a GMG approach is not a plug-in solver. It has usually to be tailored to the specific simulation code and the class of problems to be solved and is not generally applicable to large problem classes in the sense of a robust "black-box" solver. Moreover, for practical applications on complex three-dimensional meshes, it can be extremely cumbersome - if possible at all - to construct an efficient geometric multigrid method. These are the

[4] For a definition, see Section 2.3.2.

limiting factors which have prevented GMG approaches from being widely integrated into industrial simulation codes.

2.1.2 A General Characterization of Algebraic Multigrid

Operator-dependent interpolation and Galerkin-based coarse-grid correction have an interesting property: both can - in principle - be constructed purely algebraically, based on the underlying matrix and without referring to a grid. Hence, their invention did not only lead to more robust geometric multigrid approaches but was also the first step towards algebraic multigrid. Although these ideas have already been incorporated to some extent in the first "black-box" multigrid solver [21], this was still a geometric multigrid approach with a geometry-oriented coarsening. A breakthrough in overcoming the limitations imposed by geometric principles as discussed above, was achieved by the observation that, for certain matrix classes, even reasonable coarse levels themselves could be computed solely based on the entries of the matrix describing the problem. This observation has initialized the development of algebraic multigrid methods in the early eighties[5].

Algebraic Analogy AMG extends the two fundamental multigrid principles - smoothing and coarse-grid correction - to a fully algebraic setting. All GMG components such as smoothing, coarsening, interpolation, restriction and the grid operators, have an algebraic analog and play a similar role[6] as in geometric multigrid. This begins with the fact that, instead of a grid-based formulation (2.1), AMG operates on linear algebraic equations,

$$Av = b \quad \text{or, equivalently,} \quad \sum_{j=1}^{n_v} a_{ij} v_j = b_i \quad (i = 1, \ldots, n_v) \tag{2.2}$$

with $A = (a_{ij}) \in \mathbb{R}^{n_v, n_v}$ being a real (sparse) matrix, $b, v \in \mathbb{R}^{n_v}$ the right-hand side and the solution vector, respectively, and $n_v \in \mathbb{N}$ the size of A. The components of the vector v are called **variables**, denoted by $v_1, \ldots, v_{n_v}$. The corresponding index set $\{1, \ldots, n_v\}$ is denoted by $\mathcal{V}$.

If we replace the terms *grid point*, *grid*, *coarser grid* and *hierarchy of grids* by their "algebraic analogs", *variable*, *set of variables* (constituting a particular *level*), *subset of variables* and *hierarchy of levels*, respectively, we can describe algebraic multigrid[7] in formally the same way as a geometric multigrid method. In particular, the set $\mathcal{V}$ of variables formally plays the same role as the set Ω_h of grid points, and coarse-grid discretizations used in geometric multigrid to reduce low-frequency error components now correspond to properly constructed matrix equations of reduced dimension, the Galerkin coarse-level matrix equations.

A Conceptual Difference between GMG and AMG In both GMG and AMG, error components which cannot be diminished by the coarse-level correction process must efficiently be reduced by the smoothing process and vice versa. However, the way in which an efficient

[5] References will be given in Section 2.2.

[6] Note again that we consider *classical* AMG here.

[7] We should actually use the term multi*level* instead of multi*grid*. However, due to historical reasons and to emphasize the analogy to geometric multigrid, we stick to the latter.

interplay between both processes is achieved, constitutes the conceptual difference between geometric and algebraic multigrid. In (classical) GMG, a hierarchy of grids is predefined; the coarsening process and the interpolation operators are fixed and kept as simple as possible. Consequently, for an efficient interplay between smoothing and coarse-grid correction, the smoothing process has to be adjusted to the pre-defined grid hierarchy to achieve good convergence. AMG, on the other hand, only knows (2.2), at least in a purely algebraic setting. In particular, a suitable multilevel hierarchy is not known a priori. In contrast to GMG, a preferably simple smoothing process (typically Gauss-Seidel relaxation) is fixed in AMG, and AMG's main task is then to build up a suitable, problem-dependent hierarchy of levels including all necessary transfer operators as well as coarse-level operators[8] automatically and "algebraically" by solely using information contained in the matrix A. Of course, this coarse-level correction process has to be adjusted to the smoother to yield an efficient approach. It should be noted that, for AMG, smoothing has a somewhat different meaning than for GMG. In Section 3.2.1, we will define and explain this *algebraic smoothness* in detail. For the moment, we simply think of an *algebraically smooth* function (vector) being principally unaffected by relaxation.

Constructing the Hierarchy AMG provides a methodology for solving certain matrix equations hierarchically. A necessary condition for each hierarchical approach to be successful is that for the concrete problem class a "physically" meaningful hierarchy exist. An additional necessary condition for the AMG methodology to be successful is that this hierarchy can be constructed algebraically.

As will be explained in Section 2.3.2, the degrees of freedom in constructing a coarse-level correction process are the definition of coarsening and interpolation. The crucial condition to obtain a robust and efficient AMG solver is then to define coarsening and interpolation such that the overall coarse-level correction supports the smoothing process chosen - if this is possible for the concrete matrix class under consideration. AMG attempts to coarsen only "in directions" in which relaxation really smoothes the error for the given matrix A. The guiding principle in constructing the operator-dependent interpolation is complementary to the above principle of coarsening. That principle is to *force* the range of interpolation to approximately contain those "functions" which are unaffected by relaxation, that is the algebraically smooth ones.

Flexibility For certain important matrix classes, the relevant information for constructing a suitable hierarchy is contained in the matrix itself, for instance, in terms of size and sign of the coefficients. AMG can then create the necessary operators fully automatically, and the resulting coarse-level correction process is *locally* adapted to the smoothing properties of the given smoother. The automatic adaptation to the specific requirements of the matrix at hand is the major reason that AMG's efficiency is not sensitively depending on the concrete matrix equation to be solved, within the matrix class considered, - *despite using simple smoothers*. This makes AMG very flexible, efficient and robust in solving certain large matrix classes of high practical importance.

[8] The operators correspond to matrices here.

The Setup Phase - A Price to be Paid The flexibility of AMG and its simplicity of use, of course, have a price: A *setup phase*, in which the given problem (2.2) is analyzed, the coarse levels are recursively constructed and all operators are assembled, has to be concluded before the actual *solution phase* can start. This overhead is one reason for the fact that AMG is usually less efficient than comparable GMG approaches - if applied to problems for which GMG *can* be applied efficiently. Another reason is that AMG's components can, in general, not be expected to be "optimal"[9] they will always be constructed on the basis of compromises between computational work, memory requirements and overall efficiency. Nevertheless, if applied to standard elliptic model problems, the computational cost of AMG's solution phase, ignoring the setup cost, is typically comparable with the solution cost of a *robust* GMG solver. In addition, since AMG's flexibility makes it much wider applicable than GMG, the cost of AMG's setup phase, enabling this flexibility, easily pays off.

Available Theory AMG has been developed based on a variational concept (see Section 3.1.1), and convergence theory available so far assumes the matrix A to be symmetric positive definite at least. AMG is guaranteed to converge then. However, in order to obtain reasonable statements on problem-size (h-) independent convergence rates, much stronger conditions have to hold (see Chapter 3). We summarize already here that, similar to GMG, AMG has been best-developed for large classes of discretized elliptic *scalar* PDEs. To be more specific, it is theoretically best-understood and very efficient for weakly diagonally dominant symmetric M-matrices[10], as often arising for such PDEs. The theory also covers certain deviations from this ideal case. In this thesis (Sections 3.3 and 3.4), the AMG theory is generalized to our strategies for discrete PDE systems, principally following the way theoretical results have been obtained for scalar applications.

Unfortunately, convergence theory of practically applicable AMG approaches for matrices A considerably deviating from the strong conditions of this theory is not available so far. However, it should be noted that, assuming A only to be non-singular, smoothing and coarse-level correction can always be defined - in an impractical way though - so that AMG degenerates to a direct solver (see [87]). Although such approaches are much too expensive in practice, they indicate that AMG can be applied to much more general matrix classes, and they also served to motivate the development of certain, more realistic, practically applicable algorithmical components, as has been discussed in [87] (cf. Section 3.2.6).

Practical Applicability Indeed, experience has shown that the conditions of weak diagonal dominance, symmetry and M-matrix-property are sufficient but not necessary for AMG to be applicable. In practice, AMG approaches can work efficiently even for certain matrix classes which are considerably far away from this ideal case.

However, for other important matrix classes, appropriate smoothers and techniques for the automatic construction of a reasonable coarsening and interpolation have not been developed so far, and it is an open question how far we can go with AMG in practice. This is in particular true for discrete PDE systems.

[9] in the sense that - though exhibiting an "optimal" complexity of $O(N)$ - the "magnitude of $O(N)$'s constant" might be rather large.

[10] For a definition of these and other matrix types mentioned in the following, see Section 2.4.4.

Concrete developments for scalar applications are reviewed in Section 2.2.1. The status-quo on developments for PDE systems is surveyed in Section 2.2.2. Chapters 3 to 5 are concerned with theoretical and, in particular, practical aspects of our flexible, general AMG methodology for PDE systems. Our methodology considerably broadens AMG's range of applicability to many important types of PDE systems. Among them are numerically challenging PDE systems arising in industrial semiconductor simulation.

AMG as a Preconditioner It is a general experience that, for most practical applications, AMG works most efficiently when applied as a preconditioner for CG, BiCGstab or GMRes, for instance, so-called accelerators (see Section 4.4). In fact, simplified AMG variants used as preconditioners are often considerably more efficient than more complex AMG variants applied as stand-alone solvers. Moreover, only due to acceleration AMG works successfully for many practical applications. This is especially true for PDE systems [11]. Acceleration belongs to the most important means known today to increase AMG's applicability, robustness and efficiency.

SAMG - a System for Experts As indicated above, smoothing, coarsening, interpolation and acceleration are the main components that have to be carefully chosen for each class of applications in order to obtain an efficient AMG approach - which is possible not for all but many important matrix classes. We will see in the course of this thesis that many different variants for each of these four components are available. The goal in choosing concrete variants is always to find a compromise between robustness within whole matrix classes and highest efficiency for individual matrices.

Rather than one generally applicable approach, AMG thus represents a whole methodology. Consequently, we say that the software realization of our AMG methodology, SAMG (see Chapter 4), is a system for experts. This means, in particular, that SAMG in its current form is not a black-box, but a library of different matrix solvers, and each individual solver exhibits black-box character within the problem class(es) it can be applied to.

SAMG - A Plug-In Library of Matrix Solvers AMG only needs a matrix A and right-hand side b to be given, and, in case of PDE systems, some additional information[12] which is easily available in each simulation code. This implies two things. Firstly, as long as the underlying matrices have suitable properties, AMG can be applied to problems on arbitrarily complex meshes in 2D or 3D and even to pure matrix equations without a geometric background at all. This is a great practical advantage over GMG and also certain AMG(-type) approaches (see Remark 2.7). Secondly, SAMG's interface can be and has been kept very simple and comparable with that of classical one-level plug-in solvers as typically used in industrial simulation codes. Therefore, the SAMG library itself can easily be plugged into these codes.

A Rough General Classification The aspects mentioned above give a first glance at the advantages and limits of the AMG methodology and our realization SAMG. If (S)AMG is

[11] See Sections 4.4 and 4.6, and Chapter 5 for numerical results.
[12] Details will be given in Section 4.1.1.2.

applicable to a certain matrix class, its strenghts are its optimal complexity ($O(N)$), its robustness and black-box character, its plug-in character and applicability for complex geometric problems and even problems with no geometric background at all. That is, (S)AMG provides an attractive multilevel variant whenever GMG is either too difficult to apply - for instance, on unstructured meshes - or cannot be used at all. Therefore, (S)AMG should not be regarded as a competitor of GMG, but as an efficient alternative to standard one-level preconditioners, such as ILU (see Section 4.4). If (S)AMG is applicable to a given matrix class, it fulfills also the remaining goals listed in Chapter 1. The main task of this thesis is the extension of (S)AMG's applicability to practically relevant PDE systems.

2.2 Specific Approaches

Algebraic multigrid is most mature in solving large classes of scalar elliptic PDEs. Only a few AMG approaches have been developed for particular *systems* of PDEs. In Section 2.2.1, we will briefly characterize the general types of scalar AMG approaches that currently exist. This section can be seen as an update to corresponding sections in [88, 87]. Afterwards, in Section 2.2.2, we give a survey on AMG approaches that are applicable to certain PDE systems. Since our general AMG methodology covers, in particular, most of the underlying ideas, we will refer to the respective sections in Chapter 3 where those AMG approaches are reviewed and relationships to approaches based on our methodology are explained.

2.2.1 AMG for Scalar Applications

2.2.1.1 Classical AMG

The development of AMG started at the beginning of the eighties for problems $Av = b$ with weakly diagonally dominant Stieltjes[13] matrices with first steps and results given in [10, 86, 11]. A theoretical basis has been given in [70] and especially in [8]. The methodology described in these papers has originally been developed as a close algebraic analog of robust geometric multigrid. It only needs a matrix and right-hand side to be given. The setup phase is performed fully automatically and produces coarse levels which are *subsets of the finest level(-set $\mathcal{V}$)*. The definition of interpolation formulas and coarsening strategies is based on the principle of *algebraic smoothness* (see Section 3.2.1) and the notion of *strong connectivity* (see Section 4.2.1.1) as reflected by the magnitude of the off-diagonal entries of the matrix. The class of approaches based on this methodology constitutes what we call *classical AMG*, that is, even classical AMG is not a fixed method, but rather a methodology with different concrete realizations.

In [70], the first realization of classical AMG was described. Its Fortran77 implementation, AMG1R5, was made publically available in the mid eighties and is widely being used even today. The quite recent [17], for instance, gives a study of robustness and scalability of (de facto) AMG1R5.

For several years then, the research activities on AMG have been sleeping. They were revitalized at the beginning of the nineties when particularly industry has started to feel the

[13] for a definition of this and other matrix types, see Section 2.4.4.

need for more efficient matrix solvers due to problem sizes bursting the capabilities of direct and iterative one-level methods. AMG has thus become a popular and fruitful area of research with a rapid development still ongoing.

During this time, classical AMG approaches have substantially been enhanced. Most of the new developments were initiated by the fact that both the AMG1R5 algorithm and its technical realization reached their limits when applied to ever larger and more complex industrial problems. In particular, industry enquired for less memory-consuming and more robust approaches. Driven by these and other industrial needs, more robust interpolation formulas, namely *standard interpolation* and *multi-pass interpolation*, and *aggressive coarsening* strategies, reducing memory requirements considerably, have been developed (see [48, 87] and Sections 4.2.1.3 and 4.3). Incorporating these new techniques, a completely new AMG code, called RAMG [87], was developed. RAMG is a very flexible, robust and efficient approach for large classes of discretized *scalar* elliptic PDEs and similar matrices. It is based on the same methodology than AMG1R5 but provides different algorithmical components, and several variants for coarsening and interpolation as the ones mentioned above.

The classical AMG methodology [87], reviewed in detail in Section 3.2, constitutes the basis for the general AMG methodology that will be investigated in this thesis. Our library SAMG (see Chapter 4) is a corresponding generalization of the library RAMG. Further pointers to the literature investigating classical AMG approaches are given in [87, 88], for instance.

2.2.1.2 Aggregation-Based AMG

In parallel to the improvement of classical AMG, a second type of AMG approaches was developed, *aggregation-based AMG*[14], see [4, 98, 5, 97, 52], for instance[15]. Corresponding methods define coarser levels consisting of so-called *aggregates* of variables (or *macro-variables*, *supernodes*), each of which a new coarse-level variable is associated with. Aggregates are disjoint subsets of variables. In the simplest case, interpolation from the new coarse-level variables to the associated aggregates is *piecewise constant*: all variables belonging to an aggregate receive the same interpolation formula. Due to their simplicity, in particular with respect to their implementation, aggregative AMG methods have gained a large attractivity.

Unfortunately, an immediate implementation of these simple components leads to rather inefficient, not robust AMG approaches, even if used as a preconditioner. This is particularly true for second order problems for which a mere piecewise constant interpolation is not sufficient. Consequently, the basic idea of aggregative AMG needs certain improvements in order to become practically applicable. One remedy - whose efficiency and robustness is however limited to some rather simple situations - is *overcorrection* [4], rediscovered in [5] as a rescaling of the Galerkin operator. Another remedy which accelerates aggregative AMG also in more general situations is an a-posteriori improvement of interpolation by employing a *smoothing process* (*smoothed aggregation*) before the Galerkin operator is computed. In [98, 99], one ω-Jacobi relaxation step[16] with piecewise constant interpolation serving as a first guess is proposed for this purpose. The resulting interpolation is typically much better

[14] also called *aggregative AMG*.

[15] The idea of aggregation is much older, see [4] for references.

[16] applied to a "filtered matrix" derived from the original matrix A by adding all weak connections to the diagonal.

than the piecewise constant one. For some "ideal" problems and suitable ω (depending on the problem and the aggregates), linear interpolation is approached. Resulting solvers still tend to be not robust when used stand-alone. Classical AMG approaches usually show a more stable behavior because more effort is invested in creating coarsening and interpolation. When used as preconditioners, however, efficiency is often substantially increased - for both aggregative as well as classical AMG (cf. Sections 3.2.5 and 4.4).

At first sight, the two AMG types differ substantially in the way coarsening and interpolation are constructed. However, to some extent, aggregation-based AMG (in its basic form as outlined above) can be regarded as a particularly simple limiting case of classical AMG with the aim to keep coarsening as fast and interpolation as simple as possible in order to arrive at very cheap methods (cf. [87]). Both classical and (smoothed) aggregative AMG thus try to find a good compromise regarding convergence, computational work and memory requirements - but approach this compromise starting from somewhat opposite points.

Remark 2.4 For some theoretical analysis of convergence of smoothed aggregation, see [97]. An extension of the concept of smoothed aggregation for convection-diffusion equations is studied in [31]. The authors' method tackles the (assumed to be known!) "convective" and "diffusive" parts of the matrix separately and propose for the convective part the use of non-symmetric "one-sided prolongator smoothers". ▲

Remark 2.5 It should be noted that for both classical and aggregative AMG parallel variants have been developed (see [49] and [68], for instance), and main research on this important topic is ongoing. ▲

2.2.1.3 Towards More Accurate Interpolation

As has been indicated above and will be discussed in detail in the next chapter, one of the most crucial points for the efficiency of an AMG algorithm is the accuracy of interpolation. There are still open questions regarding the best way to define coarsening and interpolation, for instance, if the matrix A is symmetric positive definite, contains relatively large positive off-diagonal entries, and is far from being weakly diagonally dominant. In such cases, the performance of both classical as well as aggregative AMG may be only suboptimal.

Motivated by the fact that classical AMG in its original form is mainly suitable for M-matrices, but finite element discretizations, for instance, can produce also non-M-matrices, several new ideas have been published in the last years. Among them are *element preconditioning*, *element interpolation* and *energy minimization*.

The element preconditioning technique [33] assumes A to stem from a finite-element discretization and its element-stiffness matrices to be accessible. Based on this information, it constructs an M-matrix B which is spectrally equivalent to A. The approach [71] (i.e. AMG1R5) applied to B is then used as a preconditioner for the original problem. In some situations, and if the element-stiffness matrices are "similar" to each other, this AMG-type approach based on AMG1R5 can yield a more efficient preconditioner than AMG1R5 itself.

It is a well-known fact that error components which are slow-to-converge w.r.t. the relaxation process correspond to the eigenvectors of A belonging to the smallest eigenvalues (see also Section 3.2.1). A rather new direction of AMG research makes direct use of this

fact and tries to define interpolation so that the smaller the associated eigenvalue is, the better the eigenvectors are interpolated. To satisfy this by explicitly computing the eigenvectors of A is, of course, too expensive. However, in case of finite-element methods - assuming the elements and their element-stiffness matrices to be accessible - two ways to determine *local* representations of slow-to-converge error components have been reported in the literature.

One way consists in deriving measures (related to measures used in classical multigrid theory) the minimization of which allows the determination of these local representations. This approach, the first in this field, is called *AMG based on element interpolation*, *AMGe* [12, 16]. A derivative, *AMGe based on element agglomeration* [40], exploits AMGe ideas not only for interpolation, but also in order to produce coarse grids and coarse elements by an agglomeration[17] approach.

The other way, *spectral AMGe* (ρAMGe) [16], is based on the spectral decomposition (i.e. eigenvector computations) of small collections of element-stiffness matrices. This approach is in its infancy, in particular because there are many open questions regarding a successful extension to a robust algebraic *multi*level algorithm.

That AMGe is more robust for certain model problems than a classical AMG approach (comparable to AMG1R5) has been demonstrated in [12, 40]. Hence, it is an interesting new approach which might have the potential of leading to more robust AMG-type methods. Unfortunately, the increased robustness is at the price of a more expensive setup phase, a limited applicability (FE discretizations) and the need for additional information (knowledge of the element-stiffness matrices). In particular, AMGe is not a "pure" algebraic approach any more and thus points in a somewhat different direction compared with the aims pursued in this thesis. It should be noted that *element-free AMGe* [35] tries to overcome the outlined limitations by algebraically imitating the AMGe-construction of interpolation weights by means of a so-called extension operator. However, this approach is still in its infancy.

Other algebraic approaches, designed for the solution of FE-discretized PDEs, have been considered in [53, 107, 15]. In these approaches, the coarse-space basis functions are defined so that their energy is minimized[18] in some sense. This does not require the element-stiffness matrices to be known, but leads to a *global* (constraint) minimization problem the solution of which would be very expensive. However, iterative solution processes are proposed in the three papers to obtain approximate solutions, indicating that the extra work to be invested is acceptable. It is interesting to see that for a particular situation, the first iteration of the process described in [53] results in the method [99] developed earlier. While [107, 15] concentrate on scalar PDEs, an extension to PDE systems from linear elasticity is one major aspect in [53] (see also the next section). The test examples presented in the three papers indicate that energy minimization can help convergence. However, this benefit is essentially offset by the expense of the minimization.

The approach [104, 102] uses certain local minimizations based on the Euclidean norm[19] to find, for each variable, pairs of variables which would allow for a good interpolation. For the minimizations, so-called test vectors (see [105]) have to be provided. They should

[17] Loosely speaking, aggregation and agglomeration differ in the "target" of grouping: whereas aggregates are built from variables and are disjoint subsets of them, an agglomeration process builds macro-elements from finite elements. Here, boundary variables of macro-elements can belong to more than one macro-element.

[18] In the FE context, it is natural to define interpolation implicitly by constructing the coarse-space basis functions.

[19] instead of the energy norm as the approaches before. These norms are defined in Section 2.4.5.

approximately represent eigenvectors corresponding to the smallest eigenvalues of A (if A is symmetric positive definite). A heuristic algorithm is used to minimize the total number of C-variables. In contrast to other AMG methods, interpolation and restriction are constructed separately in case of a non-symmetric A. The approach is also available in parallel.

Remark 2.6 In this context, it should be noted that, albeit classical and aggregative AMG have been developed in the variational context, both have successfully been applied to numerous non-symmetric problems without any modification. In particular, the transpose of interpolation is always used as restriction. A heuristical, but no theoretical justification is available at this time. ▲

Remark 2.7 In contrast to classical and aggregative AMG, the approaches described above need additional information which are not necessarily available. This is especially the case for those approaches which have been designed for FE discretizations and need access to the element-stiffness matrices. Such approaches cannot be plugged into existing simulation codes in a straightforward way any more. This is why we classify them here as AMG-type rather than AMG approaches. ▲

A new trend in general research for AMG approaches has been initiated by Brandt's paper on compatible relaxation [9]. [22] presents a theory for AMG that allows for general smoothing processes and general coarsening approaches. In particular, several compatible relaxation methods are introduced, and a theoretical justification is given for their use as tools for measuring the quality of coarse grids. Several research groups are investigating how the concept of compatible relaxation can be exploited in order to yield new efficient and practically applicable AMG methods.

Another vivid area of AMG-related research is concerned with multilevel ILU-type and reduction techniques. In Section 3.2.6, we will briefly review some of these methods and discuss relationships to AMG.

2.2.2 Available AMG Approaches for PDE Systems

In practical applications, a variety of PDE systems has to be solved the numerical properties of which can differ drastically. Relevant PDE systems often consist of diffusion equations with additional convection, drift or reaction terms[20]. The individual PDEs are often of first order in time (if time-dependent) and of second order in space. They can be nonlinear and/or strongly coupled, the latter normally enforcing a "fully coupled" solution approach, that is, a simultaneous solution for all physical unknowns involved.

Typical approaches implemented in modern (industrial) simulation packages consist of an implicit discretization in time and space [21], a Newton-type[22] method to treat the nonlinearities and a direct method and/or one or more iterative one-level methods to solve the

[20] **Diffusion** is to be understood as the movement of particles due to a concentration gradient, **convection** as the transport of particles with a flowing fluid, **reaction** as the transformation of species, **drift** as the movement of particles due to an external force, for example, the movement of electrically charged particles due to an electric field.

[21] possibly including time stepping control and regridding methods.

[22] possibly with a sophisticated damping scheme.

arising systems of linear equations. The corresponding matrices are large, sparse, frequently ill-conditioned, often not symmetric positive definite, and usually far from being M-matrices. Hence, these matrices do not exhibit the properties that the "scalar" AMG approaches mentioned above principally rely on.

A way to overcome this problem consists in developing appropriate generalizations or extensions of existing AMG approaches. Until now, this way has been trodden only for specific PDE systems, mainly linear elasticity and Navier-Stokes systems. Appropriate extensions of the agglomeration-based AMG methods, mentioned in Section 2.2.1.3, can be applied to FE-discretized linear elasticity problems. We will mention other progresses made for certain

- linear elasticity problems in Section 3.3.3.2 and Remark 3.31,
- CFD problems (Navier-Stokes systems) in Remark 3.19,
- oil reservoir simulation problems[23] in Remark 3.20.

As indicated in Chapter 1, besides the developments mentioned above, AMG approaches have not been investigated for PDE systems so far. This is in particular true for industrial reaction-diffusion[24] as well as drift-diffusion systems.

The approaches reviewed in the remarks listed above have several strong relationships to approaches belonging to our AMG methodology. Our general AMG methodology covers, among others, also the basic ideas of these approaches. This will be explained in more detail in the respective remarks.

2.3 Formal Algebraic Multigrid Components

Each AMG algorithm consists of two parts, namely the **setup phase** the purpose of which is the automatic construction of a hierarchy of levels and transfer operators and the **solution phase** in which a multigrid cycling process is performed. To describe an AMG approach, it is sufficient to specify the components of a two-level process. The recursive extension of a two-level to a multilevel process is formally straightforward then. In order to distinguish fine-level and coarse-level quantities, we use indices h and H, respectively. In particular,

$$A_h v^h = b^h \quad \text{or} \quad \sum_{j \in \mathcal{V}^h} a_{ij}^h v_j^h = b_i^h \quad (i \in \mathcal{V}^h) \ , \tag{2.3}$$

$$A_H v^H = b^H \quad \text{or} \quad \sum_{j \in \mathcal{V}^H} a_{ij}^H v_j^H = b_i^H \quad (i \in \mathcal{V}^H) \ . \tag{2.4}$$

h and H are chosen in order to indicate the formal similarity to geometric multigrid. However, in general, they are not related to a discretization parameter. Later on, in subsequent chapters, we will only make use of these level indices if it is necessary to distinguish two consecutive levels.

[23] featuring linear elasticity problems and multi-phase flow problems in porous media, the latter typically black-oil models consisting of a pressure equation and two continuity equations.

[24] For completeness, we want to mention that in [102] a very straightforward point-based extension of [104] has been introduced and applied to two simple reaction-diffusion models. Similar to the approach [104], it needs test vectors. The effort necessary to obtain such vectors has not been discussed, however.

Remember the following general assumptions: In this thesis, it is assumed that the given matrix $A = A_h$ is real, nonsingular and

$$\forall\, i \in \mathcal{V}^h \;:\; a_{ii}^h > 0 \;\;, \tag{2.5}$$

unless explicitly stated otherwise.

Remark 2.8 Results carry over to singular symmetric matrices A the nullspace of which consist of the constant vectors only (in particular, all row sums of such matrices equal zero). In this case, the singularity of the coarsest-level system has to be treated correctly by the selected coarsest-level solver (for example, sparse Gaussian elimination), and interpolation and restriction have to transfer constants exactly. Since the latter is ensured by all interpolation and restriction operators proposed in the remainder of this thesis, we will not discuss the zero-row-sum case explicitly. ▲

Remark 2.9 The case that some (exceptional) diagonal entries of the original, finest-level matrix A are zero will be discussed in Section 4.1.1. The case that nonpositive diagonals emerge on coarse levels, will be discussed in Appendix A.1. ▲

For each AMG two-level algorithm, the smoothing process and the coarse-level correction process have to be defined. It must especially be explained how the coarse-level set $\mathcal{V}^H$ of variables is obtained ("coarsening") and how the operators between the two levels, namely the **interpolation** (or **prolongation**) $\boldsymbol{I_H^h}$ and **restriction** $\boldsymbol{I_h^H}$, and the coarse-level operator A_H are computed.

We will explain in Sections 2.3.1 and 2.3.2 how smoothing and (Galerkin-based) coarse-level correction do formally look for the class of AMG approaches which are considered in the remainder of this thesis. In particular, we will see that [25] coarsening and interpolation are the degrees of freedom in the coarse-level correction process employed in our AMG methodology.

We will see in subsequent chapters that the processes of smoothing, coarsening and interpolation, even for the AMG class we restrict ourselves to, are not uniquely defined. Particularly, even though coarsening and interpolation are strongly related to each other - and, moreover, to the smoothing process - there exist many variants for them. Concrete processes for smoothing, coarsening, interpolation and their interplay will be discussed in Chapters 3 and 4.

2.3.1 The Smoothing Process

Given a linear **smoothing operator** $\boldsymbol{S_h}$, we denote a single smoothing step by

$$\begin{aligned} v^h \;&\longrightarrow \overline{v}^h \quad \text{with} \\ &\overline{v}^h := \mathrm{SMOOTH}(S_h, A_h, b^h, v^h) := S_h v^h + (I_h - S_h) A_h^{-1} b^h \;\;. \end{aligned} \tag{2.6}$$

[25] besides the coarsest-level solver which is not explicitly discussed in this thesis but assumed to be a (sparse) direct solver.

I_h denotes the identity operator. Analogously, ν smoothing steps are denoted by

$$\text{SMOOTH}^{\nu}(S_h, A_h, b^h, v^h) \ .$$

In terms of the current error, $e^h := v^h_\star - v^h$ with $v^h_\star$ denoting the exact solution of (2.3), the application of ν smoothing steps means

$$e^h \longrightarrow \overline{e}^h \quad \text{with} \quad \overline{e}^h := S^\nu_h e^h \ . \tag{2.7}$$

Convention: The letter v will normally indicate approximation or solution quantities and the letter e correction or error quantities.

As mentioned above, AMG usually employs rather simple smoothers. Often, a variable-wise Gauss-Seidel (GS) relaxation (i.e. $S_h = I_h - Q_h^{-1} A_h$ with Q_h being the lower triangular part of A_h, including the diagonal) is selected. Clearly, unless A_h is positive definite, such a simple variable-wise relaxation method is only reasonable if A_h fulfills additional requirements, in particular, its diagonal elements should be sufficiently large compared to the off-diagonal entries.

In this thesis, we will consider variable-wise, unknown-wise and point-wise GS relaxation processes[26], ω-Jacobi relaxation, i.e. $S_h = I_h - \omega D_h^{-1} A_h$ with $D_h = \text{diag}(A_h)$, and various ILU variants (see, in particular, Sections 3.4.1.2 and 4.4 and Chapter 5).

2.3.2 The Coarse-Level Correction Process

The first step in AMG's coarse-level correction process consists in the construction of the set of coarse-level variables $\mathcal{V}^H$. This is done by splitting $\mathcal{V}^h$ into two disjoint subsets C^h and F^h,

$$\mathcal{V}^h = C^h \dot{\cup} F^h \ , \tag{2.8}$$

- based on certain rules - with C^h representing those variables which are to be contained in the coarse level (***C*-variables**) and F^h being the complementary set of fine-level variables (***F*-variables**). This splitting is called a ***C*/*F*-splitting** of $\mathcal{V}^h$. Note that here the coarse-level variables $\mathcal{V}^H := C^h$ can be interpreted as a subset of the fine-level ones.

Then, an interpolation operator I^h_H is constructed, fitting to the C/F-splitting, and mapping coarse-level corrections to fine-level ones. We only consider interpolations $e^h = I^h_H e^H$ which are of the form

$$e^h_i = (I^h_H e^H)_i = \begin{cases} e^H_i & \text{for } i \in C^h \ , \\ \sum_{j \in P^h_i} w^h_{ij} e^h_j & \text{for } i \in F^h \ , \end{cases} \tag{2.9}$$

where $P^h_i \subseteq C^h$ is called the **set of interpolatory variables** (for the i-th variable).

The restriction operator I^H_h, mapping fine-level vectors to coarse-level ones, is always defined to be the transpose of interpolation,

$$I^H_h := (I^h_H)^T \ . \tag{2.10}$$

[26] see Sections 3.3.1.1 and 3.4.1 for the last two.

Obviously, both interpolation and restriction are full rank operators. In this thesis, the term transpose always refers to the Euclidean inner product (see Section 2.4.5), denoted by $(\cdot,\cdot)_E$ in the following.

Once $\mathcal{V}^H$ and I_H^h, I_h^H are given, the AMG coarse-level correction operator (Galerkin-operator) can be evaluated:

$$A_H := I_h^H A_h I_H^h \,. \tag{2.11}$$

2.3.3 The Two-Level Process

Smoothing and coarse-level correction are now combined as in geometric multigrid. One step of a two-level process, starting with the approximation v_{old}^h, is defined as depicted in Figure 2.1.

1. Perform ν_1 pre-smoothing steps: $\overline{v}^h := \mathrm{SMOOTH}^{\nu_1}(S_h, A_h, b^h, v_{old}^h)$.
2. Compute the *residual*: $r^h := b^h - A_h \overline{v}^h$.
3. Restrict the residual to the coarse level: $b^H := I_h^H r^h$.
4. Solve the coarse-level correction system: $A_H e^H = b^H$.
5. Transfer the correction e^H to the fine level and correct the old approximation $\overline{v}^h$:
$$\overline{\overline{v}}^h := \overline{v}^h + I_H^h e^H \,. \tag{2.12}$$
6. Perform ν_2 post-smoothing steps: $v_{new}^h := \mathrm{SMOOTH}^{\nu_2}(S_h, A_h, b^h, \overline{\overline{v}}^h)$.

Figure 2.1: One cycle of a two-level AMG method.

In terms of the error, one two-level cycle maps $e_{old}^h \to e_{new}^h$ where

$$e_{new}^h = K_{h,H} e_{old}^h \quad \text{with} \quad K := K_{h,H} := I_h - I_H^h A_H^{-1} I_h^H A_h \,, \tag{2.13}$$

if $\nu_1 = \nu_2 = 0$ (i.e. coarse-level correction without smoothing). If smoothing is performed,

$$e_{new}^h = M_{h,H} e_{old}^h \quad \text{with} \quad M_{h,H} = M_{h,H}(\nu_1, \nu_2) := S_h^{\nu_2} K_{h,H} S_h^{\nu_1} \,. \tag{2.14}$$

$K_{h,H}$ is called the **coarse-level correction operator**, $M_{h,H}$ the **two-level iteration operator**. Obviously, this AMG two-level process formally equals a geometric two-level process. As in geometric multigrid, the extension to a multilevel process (with e.g. V-, F- or W-cycles) is straightforward.

2.4 More Basic Definitions and Notation

2.4.1 Unknowns and Points

We usually assume A to result from the discretization and linearization of a system of partial differential equation (PDEs)

$$F(u_1, \ldots, u_{n_u}) = 0 \tag{2.15}$$

in the domain $\Omega \subseteq \mathbb{R}^d$ attached with reasonable boundary conditions. Here, the $u_1, \ldots, u_{n_u}$ denote $n_u \geq 1$ scalar functions $\mathbb{R}^d \supset \Omega \to \mathbb{R}$, called **unknowns** in the sequel. In the special case $n_u = 1$, (2.15) represents a scalar PDE.

The set of indices $\mathcal{V}$ can be split into *disjoint* subsets $\mathcal{U}_1, \ldots, \mathcal{U}_{n_u}$ with $\mathcal{V} = \dot{\cup}_{n=1}^{n_u} \mathcal{U}_n$, where the subset $\mathcal{U}_n$ represents the indices of variables belonging to the $n-th$ unknown u_n. The u_n and $\mathcal{U}_n$ are both called **unknowns**, which should not lead to any confusion.

We generally assume the i-th equation (the i-th matrix row) to "belong" to the i-th variable (cf. also Remark 2.10 below). For some theoretical considerations in this thesis, it is convenient to assume (2.2) to be reordered "unknown-wise". That is, assuming any given order of indices inside each $\mathcal{U}_n$, the system (2.2) then has the form

$$\begin{bmatrix} A_{[1,1]} & \cdots & A_{[1,n_u]} \\ \vdots & \ddots & \vdots \\ A_{[n_u,1]} & \cdots & A_{[n_u,n_u]} \end{bmatrix} \begin{bmatrix} v_{[1]} \\ \vdots \\ v_{[n_u]} \end{bmatrix} = \begin{bmatrix} b_{[1]} \\ \vdots \\ b_{[n_u]} \end{bmatrix}. \tag{2.16}$$

Here, $v_{[n]}$ denotes the vector of variables associated with the n-th unknown, $b_{[n]}$ the corresponding part of the vector b, and $A_{[m,n]}$ the submatrix of A which reflects the couplings of the m-th to the n-th unknown. The entries in the $A_{[m,n]}$ with $m \neq n$ are called **unknown cross-couplings**. If an order of the unknowns $\mathcal{U}_n$ and the variables within each $\mathcal{U}_n$ has been fixed, a mapping of the variable-indices to the unknown-indices is induced, the so-called **variable-to-unknown mapping (VU mapping)**.

If (2.15) is discretized so that *"the different unknowns are living on the same grid"*, we often assume the linear system (2.2) to be reordered "point-wise". To be more specific, assume that the set $\mathcal{V}$ is split into *disjoint* subsets $\mathcal{P}_1, \ldots, \mathcal{P}_{n_p}$ with $\mathcal{V} = \dot{\cup}_{k=1}^{n_p} \mathcal{P}_k$. Then, these subsets $\mathcal{P}_k$ are called **points**, where $n_p > 1$ denotes the number of mesh points, and $\mathcal{P}_k$ contains the indices of variables sitting at the k-th point. The set $\{1, \ldots, n_p\}$ is denoted by $\mathcal{V}^p$. Clearly, the variables associated with any fixed point, all belong to *different* unknowns. The point-wise reordered linear system (2.2) looks like (assuming any given order of indices inside each $\mathcal{P}_k$):

$$\begin{bmatrix} A_{(1,1)} & \cdots & A_{(1,n_p)} \\ \vdots & \ddots & \vdots \\ A_{(n_p,1)} & \cdots & A_{(n_p,n_p)} \end{bmatrix} \begin{bmatrix} v_{(1)} \\ \vdots \\ v_{(n_p)} \end{bmatrix} = \begin{bmatrix} b_{(1)} \\ \vdots \\ b_{(n_p)} \end{bmatrix}, \tag{2.17}$$

where $v_{(k)}$ denotes the vector of variables located at point $\mathcal{P}_k$, $b_{(k)}$ the corresponding part of the vector b, and $A_{(k,l)}$ the submatrix of A, which reflects the couplings of the k-th to

the l-th point. The $A_{(k,l)}$ are called **point-coupling matrices**. If an ordering of the points $\mathcal{P}_k$ and the variables within each $\mathcal{P}_k$ has been fixed, a mapping of the variable-indices to the point-indices is induced, the so-called **variable-to-point mapping (VP mapping)**.

For an illustration of variables, unknowns and points, see Fig. 2.2(a).

Remark 2.10 VU and VP mappings can be defined for every discretized PDE system. They are, however, not unique, but for many PDE systems there exists a "natural" structural description. The classical counterexample that a not even a "natural" relationship between variables and matrix rows exists, is given by the Cauchy-Riemann system. ▲

Remark 2.11 In the special case that (2.15) represents a scalar PDE (n_u=1), there is only one unknown ($\mathcal{U}_1 = \mathcal{V}$), and each variable corresponds to a point, i.e. $\mathcal{P}_k = \{k\}$ for all $k \in \mathcal{V}$. ▲

Remark 2.12 Often, all unknowns are represented at each point. But there are important cases, as can be seen in Chapter 5, where this is not true. In such cases, typically, not all functions u_n exist in the whole domain (as in Fig. 2.2(a)). ▲

Remark 2.13 Usually, we think of points as being grid nodes or element centers. However, the above definition does not involve coordinates and, hence, it is only important that a *clustering* (with *disjoint* clusters) of the variables makes sense. ▲

2.4.2 Couplings, Patterns and Graphs

For formal descriptions, matrix-vector terminology is used. However, the use of *graphs* is often very convenient to easily describe, motivate, and analyze AMG processes such as coarsening and interpolation strategies. The use of graphs is a way to easily see similarities and analogies to geometric multigrid processes.

All graphs used in this thesis are based on the **connectivity pattern** $\Sigma(A)$ of a matrix A which is defined as the distribution of the nonzero entries of A, i.e. the set of index pairs (i, j) for which $a_{ij} \neq 0$. Note that in practice the connectivity pattern is a subset of the **sparsity pattern** of A. We come back to this in Section 4.1.1.2.

The nodes of the graphs correspond to variables or points, depending on which type of relation level we want to investigate. The edges of such a graph, that is the connections between the nodes, are defined via the connectivity pattern in the following way. We call a variable v_i (directly) **coupled** (or **connected**) to variable v_j if $a_{ij} \neq 0$. Correspondingly, the (direct) **neighborhood** of a variable v_i is defined by

$$N_i := \{j \in \mathcal{V} \mid j \neq i \text{ and } v_i \text{ coupled to } v_j\}\,. \tag{2.18}$$

Analogously, a point $\mathcal{P}_k$ (an unknown $\mathcal{U}_n$) is said to be coupled to a point $\mathcal{P}_l$ (an unknown $\mathcal{U}_m$) if there exists a variable in $\mathcal{P}_k$ ($\mathcal{U}_n$) which is coupled to a variable in $\mathcal{P}_l$ ($\mathcal{U}_m$).

For each of the coupling types, graphs can be drawn. In this thesis, we will often make use of graphs representing couplings of variables or points. If it is necessary to visualize more

than one "species"- variables, unknowns or points - at the same time, different "colors" (to distinguish variables of different unknowns) and groupings of variables (to mark variables living at the same point) can be used (see also Fig. 2.2(a)). Examples of such graphs are given by the Figures 2.2(b) and 2.3.

2.4.3 More Specific AMG Notation

If it is necessary to distinguish between positive and negative off-diagonal entries of A, we use the notation

$$a_{ij}^- := \begin{cases} a_{ij} & (\text{if } a_{ij} < 0) \\ 0 & (\text{if } a_{ij} \geq 0) \end{cases} \quad \text{and} \quad a_{ij}^+ := \begin{cases} 0 & (\text{if } a_{ij} \leq 0) \\ a_{ij} & (\text{if } a_{ij} > 0)\,, \end{cases} \tag{2.19}$$

and correspondingly

$$N_i^- := \{j \in N_i \mid a_{ij} < 0\} \quad \text{and} \quad N_i^+ := \{j \in N_i \mid a_{ij} > 0\}\,. \tag{2.20}$$

For theoretical considerations, it is often convenient to assume vectors and matrices to be reordered according to a given C/F-splitting so that (2.3) can be written as

$$A_h v^h = \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} v_F \\ v_C \end{pmatrix} = \begin{pmatrix} b_F \\ b_C \end{pmatrix} = b^h\,. \tag{2.21}$$

Analogously, the interlevel transfer operators are written as

$$I_H^h = \begin{pmatrix} I_{FC} \\ I_{CC} \end{pmatrix}, \quad I_h^H = (I_{CF}, I_{CC}) \tag{2.22}$$

with $I_{CF} = (I_{FC})^T$ and I_{CC} being the identity operator. Instead of $e^h = I_H^h e^H$ and (2.9), we simply write

$$e_F = I_{FC} e_C \quad \text{and} \quad e_i = \sum_{j \in P_i} w_{ij} e_j \quad (i \in F)\,, \tag{2.23}$$

respectively, which should not lead to any confusion.

2.4.4 Basic Matrix Types, Eigenvalues

Important parts of the AMG theory for the scalar case deal especially with the model class of *Stieltjes matrices*. But also more general basic types of matrices are discussed, the most important ones being listed in the following. Let $(\cdot,\cdot) = (\cdot,\cdot)_E$ denote the Euclidean inner product. Then a square matrix $B = (b_{ij})_{i,j}$ is called

- **symmetric** if $b_{ij} = b_{ji}$ for all i, j.
- **weakly diagonally dominant** if

$$\forall\, i \;:\; b_{ii} \geq \sum_{j \neq i} |b_{ij}| \quad . \tag{2.24}$$

- **strongly diagonally dominant** if

$$\forall\, i \;:\; b_{ii} > \sum_{j \neq i} |b_{ij}| \quad . \tag{2.25}$$

Remark: Strong diagonal dominance is obviously equivalent to the existence of a $\delta = \delta_B > 1$ with

$$\forall\, i \;:\; b_{ii} - \sum_{j \neq i} |b_{ij}| \geq \delta b_{ii} \quad . \tag{2.26}$$

If for a class $\mathcal{B}$ of matrices B a δ can be found which is indepent of B, $\mathcal{B}$ is said to satisfy the property of strong diagonal dominance *uniformly*.

- **positive semi-definite** if for all $v \;:\; (Bv, v) \geq 0$. If B is additionally symmetric, we write $B \geq 0$. Correspondingly, $B_1 \geq B_2$ means $B_1 - B_2 \geq 0$.

- **positive definite** if B is positive semi-definite and

$$\forall\, v \;:\; \Big((Bv, v) = 0 \Rightarrow v = 0 \Big) \quad . \tag{2.27}$$

If B is additionally symmetric, we write $B > 0$. Correspondingly, $B_1 > B_2$ means $B_1 - B_2 > 0$. The class of *symmetric* positive definite matrices is denoted by $\mathcal{A}_{\text{spd}}$, the subclass of weakly diagonally dominant symmetric positive definite matrices by $\mathcal{A}_{\text{wdd}}$.

Examples: Well-known examples of matrices belonging to $\mathcal{A}_{\text{spd}}$ result from typical nine-point discretizations of the anisotropic Laplace operator[27] $-\epsilon u_{xx} - u_{yy}$,

$$\frac{1}{h^2} \begin{bmatrix} -(1+\epsilon)\alpha & 2\alpha\epsilon - 1 & -(1+\epsilon)\alpha \\ 2\alpha - \epsilon & 2(1+\epsilon) & 2\alpha - \epsilon \\ -(1+\epsilon)\alpha & 2\alpha\epsilon - 1 & -(1+\epsilon)\alpha \end{bmatrix}_h \quad , \tag{2.28}$$

with $-1/2 < \alpha \leq 1/2$. For $\alpha = 0$, the standard anisotropic five-point stencil arises (see 2.30 below), for $\alpha = 1/4$ the stencil for the standard FE discretization with bilinear finite elements. Note that the above stencil results from the following discretization of $-\epsilon u_{xx}$ (for the stencil notation, see [94], for instance)

$$\frac{1}{1+2\alpha} \frac{1}{h^2} \begin{bmatrix} -\epsilon & 2\epsilon & -\epsilon \end{bmatrix}_h \begin{bmatrix} \alpha \\ 1 \\ \alpha \end{bmatrix}_h \tag{2.29}$$

and a corresponding discretization of $-u_{yy}$.

- an **M-matrix** (see [85]) if its off-diagonal entries are nonpositive, B is nonsingular and all entries of its inverse B^{-1} are nonnegative.

[27] with Dirichlet conditions, for instance. In case of periodic boundary conditions, for instance, the resulting matrices would be symmetric positive semi-definite.

- a **Stieltjes matrix** if $B > 0$ and off-diagonally nonpositive. The class of Stieltjes matrices is denoted by $\mathcal{A}_{\text{St}}$. Every Stieltjes matrix is an M-matrix.

  As perhaps the most known example, consider Poisson's equation with Dirichlet boundary conditions, discretized by means of the standard five-point Poisson stencil,

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h . \tag{2.30}$$

  Remark 2.14 Every symmetric weakly diagonally dominant matrix with positive diagonals, nonpositive off-diagonals and with $b_{ii} > \sum_{j\neq i} |b_{ij}|$ for at least one i is a Stieltjes matrix (for a proof see [85]). ▲

- **of essentially positive type** [8] if B is positive definite and if there exists a constant $c > 0$ such that

$$\forall\, v \,:\, \sum_{i,j}(-b_{ij})(v_i - v_j)^2 \geq c \sum_{i,j}(-b_{ij}^-)(v_i - v_j)^2 \ . \tag{2.31}$$

  The class of *symmetric* matrices of essentially positive type is denoted by $\mathcal{A}_{\text{ess}}$.

  Remark: This condition implies that each row containing off-diagonal elements has at least one negative off-diagonal entry. For the k-th row, this can easily be seen by applying the above inequality to the vector $v = (v_i)$ with $v_i = \delta_{ik}$ (Kronecker symbol).

  Examples: Higher order discretizations of second order elliptic problems or problems involving mixed derivatives often lead to symmetric essentially positive type matrices. For instance, consider the stencil

$$\frac{1}{12h^2}\begin{bmatrix} & & 1 & & \\ & -16 & & -16 & \\ 1 & & 60 & & 1 \\ & -16 & & -16 & \\ & & 1 & & \end{bmatrix}_h \tag{2.32}$$

  which corresponds to a fourth order dicretization of $-\Delta u$ (ignoring boundary conditions). Here, (2.31) is fulfilled with $c = 3/4$. As another example, the nine-point discretization of $-\Delta u + u_{xy}$,

$$\frac{1}{h^2}\begin{bmatrix} -\frac{1}{4} & -1 & \frac{1}{4} \\ -1 & 4 & -1 \\ \frac{1}{4} & -1 & -\frac{1}{4} \end{bmatrix}_h \tag{2.33}$$

  satisfies (2.31) with $c = 1/2$. In such cases, the "essential positiveness"of the matrices is due to the fact that, for each $b_{ij} > 0$, there exist **strong negative paths** of at least length two from i to j, i.e., there exist $b_{ik} < 0$ and $b_{kj} < 0$ with $|b_{ik}|$, $|b_{kj}|$ being sufficiently large compared with b_{ij}. ▲

- **of essentially block-positive type** if B is positive definite, all $B_{(k,l)}$ symmetric, and if there exists a constant $c > 0$ such that

$$\begin{aligned} \forall\, v \;:\quad & \sum_{k,l} \big(-B_{(k,l)}(v_{(k)} - v_{(l)}), v_{(k)} - v_{(l)}\big) \\ \geq c & \sum_{k,l,-B_{(k,l)}>0} \big(-B_{(k,l)}(v_{(k)} - v_{(l)}), v_{(k)} - v_{(l)}\big) \quad . \end{aligned} \tag{2.34}$$

  Remark: We use the above definition of the term "essentially block-positive type" as an analog of "essentially positive type" for the point-oriented case.

With $\lambda(B)$ we denote an *eigenvalue* of B. $\lambda_{\min}(B)$ and $\lambda_{\max}(B)$ denote the minimal and maximal eigenvalue of B, respectively. Analogously, $|\lambda|_{\min}(B)$ and $|\lambda|_{\max}(B)$ are defined. The following lemma holds

Lemma 2.1 (cf. [106]) *Let $B \in \mathbb{R}^{n,n}$ be any symmetric matrix. Then each eigenvalue $\lambda(B)$ is real, and there exists an orthonormal basis of $\mathbb{R}^n$ consisting of eigenvectors of B. Moreover,*

$$\forall\, v \in \mathbb{R}^n \;:\; (Bv, v)_E \leq \lambda_{\max}(B)(v, v)_E \quad , \tag{2.35}$$

$$\forall\, v \in \mathbb{R}^n \;:\; (Bv, v)_E \geq \lambda_{\min}(B)(v, v)_E \quad , \tag{2.36}$$

and $\lambda_{\min}(-B) = -\lambda_{\max}(B)$. For two symmetric matrices B_1 and B_2, we have

$$\lambda_{\max}(B_1 + B_2) \leq \lambda_{\max}(B_1) + \lambda_{\max}(B_2) \quad , \tag{2.37}$$

$$\lambda_{\min}(B_1 + B_2) \geq \lambda_{\min}(B_1) + \lambda_{\min}(B_2) \quad . \tag{2.38}$$

Of course, the sum of positive definite matrices (positive semi-definite matrices) is positive definite (at least positive semi-definite).

2.4.5 Inner Products and Norms

In addition to the Euclidean inner product $(\cdot, \cdot)_E$, we will use the following three inner products if $A > 0$:

$$(v, w)_0 := (Dv, w)_E \quad , \tag{2.39}$$

$$(v, w)_1 := (Av, w)_E \quad , \tag{2.40}$$

$$(v, w)_2 := (D^{-1}Av, Aw)_E \quad . \tag{2.41}$$

with D being the diagonal of A, $D := \operatorname{diag}(A)$ (which is also positive definite). The associated norms are denoted by $||\cdot||_i$ ($i \in \{E, 0, 1, 2\}$). $(\cdot, \cdot)_1$ is called the **energy inner product**, and $||\cdot||_1$ the **energy norm**.

Given any C/F-splitting, note that with $A > 0$ the matrices A_{FF} and $D_{FF} = \operatorname{diag}(A_{FF})$ are also positive definite. Then we can define the analogs of the above inner products applied to A_{FF} instead of A_h. We will use

$$(v_F, w_F)_{0,F} := (D_{FF}v_F, w_F)_E \text{ and } (v_F, w_F)_{1,F} := (A_{FF}v_F, w_F)_E \quad , \tag{2.42}$$

and their associated norms $||\cdot||_{i,F}$ ($i \in \{0,1\}$).

Given any VP mapping (and (2.17), see Section 2.4.1), we can define analogs of (2.39) - (2.41) by replacing D by its block-diagonal analog,

$$(v,w)_{P,0} := (D_P v, w)_E \ , \tag{2.43}$$

$$(v,w)_{P,1} := (A_P v, w)_E = \ (Av,w)_E \ , \tag{2.44}$$

$$(v,w)_{P,2} := (D_P^{-1} Av, Aw)_E \tag{2.45}$$

with D_P being the block-diagonal matrix of A, that is, the k-th diagonal block of D_P equals $A_{(k,k)}$. Their associated norms are defined accordingly. Of course, A and its "blocked"version A_P are identical (besides ordering!). Analogously, $(\cdot,\cdot)_{P,0,F}$ is defined.

In the following, we summarize some standard facts about matrix norms (see [84, 100]). We only consider matrix norms on $\mathbb{R}^{n,n}$ in the following. We call a matrix norm $|||\cdot|||$ **compatible** to a vector norm $||\cdot||$ (on $\mathbb{R}^n$) if for all matrices $B \in \mathbb{R}^{n,n}$ and vectors $v \in \mathbb{R}^n$

$$||Bv|| \leq |||B||| \, ||v|| \ . \tag{2.46}$$

With

$$||B|| := \max_{v \neq 0} \frac{||Bv||}{||v||} \tag{2.47}$$

for square matrices B a matrix norm is defined, the so-called **operator norm** induced by the vector norm $||\cdot||$. Every operator norm is submultiplicative:

$$||B_1 B_2|| \leq ||B_1|| \, ||B_2|| \ , \tag{2.48}$$

and we have $||I|| = 1$. Obviously, the operator norm is compatible to the vector norm it is induced by, and $||B|| \leq |||B|||$ holds for all matrix norms $|||\cdot|||$ which are compatible to the vector norm $||\cdot||$.

The **spectral radius** $\rho(B)$ of any matrix $B \in \mathbb{R}^{n,n}$ is defined by

$$\rho(B) := |\lambda|_{\max}(B) \, . \tag{2.49}$$

The spectral radius of B is the infimum of all operator norms of B:

$$\rho(B) = \inf_{||\cdot|| \text{ operator norm on } \mathbb{R}^{n,n}} ||B|| \ . \tag{2.50}$$

For all matrices B_1, B_2 the following holds:

$$\rho(B_1 B_2) = \rho(B_2 B_1) \ . \tag{2.51}$$

The following matrix norms are used in this thesis:

$$||B||_E := \sqrt{\rho(B^T B)} \quad \text{Euclidean norm} , \tag{2.52}$$

$$||B||_{max} := \max_{i,j} |b_{ij}| \quad \text{maximum norm} , \tag{2.53}$$

$$||B||_{rs} := \max_i \sum_j |b_{ij}| \quad \text{row sum norm} , \tag{2.54}$$

$$||B||_{sch} := \sqrt{\sum_{ij} b_{ij}^2} \quad \text{Schur norm} . \tag{2.55}$$

$||\cdot||_E$ is the operator norm and $||\cdot||_{sch}$ a compatible matrix norm to the Euclidean vector norm. The row sum norm is the operator norm induced by the maximum vector norm. $||\cdot||_E$, $||\cdot||_{rs}$ and $||\cdot||_{sch}$ are submultiplicative, $||\cdot||_{max}$ not. If B is symmetric, we have $||B||_E = \rho(B)$. If $B > 0$, we obtain

$$||B^{-1}||_E = \rho(B^{-1}) = \frac{1}{\lambda_{\min}(B)} . \tag{2.56}$$

For a regular matrix B, its **condition number** w.r.t. $||\cdot||_E$ is defined by

$$\mathrm{cond}_E(B) := ||B||_E \, ||B^{-1}||_E \geq 1 . \tag{2.57}$$

If $B > 0$, we obtain

$$\mathrm{cond}_E(B) = \frac{\lambda_{\max}(B)}{\lambda_{\min}(B)} . \tag{2.58}$$

For all $B_1 > 0$, $B_2 > 0$ and constants $c > 0$ the following equivalence holds:

$$\Big(\forall\, e \quad (B_1 e, e)_E \leq c (B_2 e, e)_E \Big) \iff \rho(B_2^{-1} B_1) \leq c \tag{2.59}$$

which follows from $B_2 = B_2^{1/2} B_2^{1/2}$ with $B_2^{1/2} > 0$, $B_2^{-1/2} B_1 B_2^{-1/2} > 0$ and

$$\begin{aligned}
& \forall\, e : \quad (B_1 e, e)_E \leq c (B_2 e, e)_E \\
\iff \quad & \forall\, e : \quad (B_1 B_2^{-1/2} e, B_2^{-1/2} e)_E \leq c (B_2 B_2^{-1/2} e, B_2^{-1/2} e)_E \\
\iff \quad & \forall\, e : \quad (B_2^{-1/2} B_1 B_2^{-1/2} e, e)_E \leq c (e, e)_E \\
\underset{(2.49)}{\iff} \quad & \rho(B_2^{-1/2} B_1 B_2^{-1/2}) \leq c \\
\underset{(2.51)}{\iff} \quad & \rho(B_2^{-1} B_1) \leq c
\end{aligned}$$

2.4.6 Further Notation

For an arbitrary matrix B, $B = 0$ means that B contains only zero entries.

The components of a vector w are denoted by w_i. $w > 0$ means that all of its components are positive.

$\mathcal{R}(Q)$ and $\mathcal{N}(Q)$ denote the *range* and *nullspace* of a given operator Q, respectively.

Finite difference discretization is abbreviated by **FD discretization**, analogously, finite element (discretization) by **FE (discretization)**, and finite volume (discretization) by **FV (discretization)**.

The following terms are used for discussing numerical results. We often make use of scientific notation in the form Me-P for $M \cdot 10^P$, for example 3.5e-9$= 3.5 \cdot 10^{-9}$. Analogously, Me+P is defined. The **average residual reduction factor (ARF) corresponding to n_{it} iterations** is defined as

$$\text{ARF} \;=\; \left(\frac{||r^{(n_{\text{it}})}||}{||r^{(0)}||} \right)^{1/n_{\text{it}}} \tag{2.60}$$

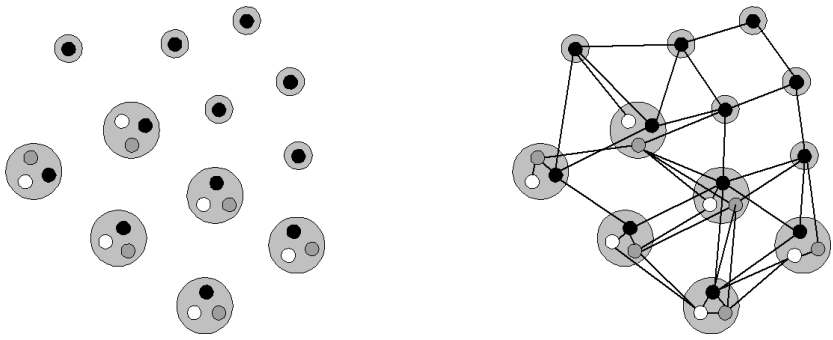
where n_{it} denotes the number of iterations (cycles) performed, $r^{(0)}$ the residual of the first guess (assumed to be nonzero), and $r^{(i)}$ ($i > 0$) the residual after the i-th iteration. We always use $||\cdot|| = ||\cdot||_E$ here. In addition, (in order to obtain an ARF closer to the spectral radius ρ of the iteration matrix M in case of stand-alone AMG, see also Remark 4.25) we often skip the first two iterations and define

$$\text{ARF}_2 \;=\; \left(\frac{||r^{(n_{\text{it}})}||}{||r^{(2)}||} \right)^{1/(n_{\text{it}}-2)} \tag{2.61}$$

For all solvers discussed, the iterations are stopped if an iteration (cycle) n with

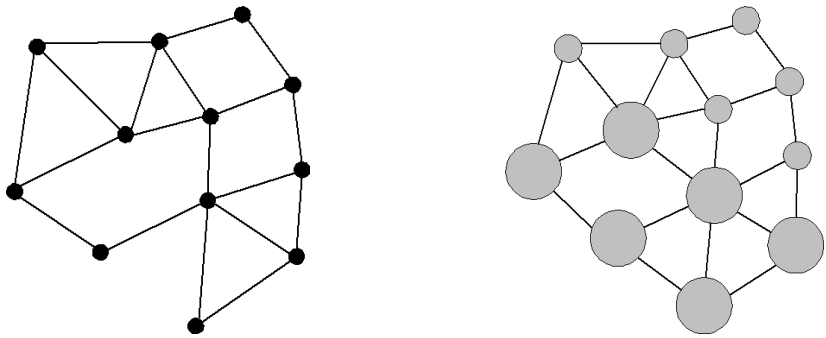
$$\frac{||r^{(n)}||}{||r^{(0)}||} \leq \epsilon_{\text{it}} \tag{2.62}$$

is reached, where $\epsilon_{\text{it}} \in]0;1]$ denotes the **(relative) residual reduction factor** demanded. $\epsilon_{\text{it}} =$ 1e-10 is used throughout this thesis unless explicitly stated otherwise. “log. residual” and “log. error” stand for $\log_{10}||r||$ and $\log_{10}||e||$, respectively. Timings given are always wall-clock timings.



(a) (b)

Figure 2.2: (a) A visualization of variables, unknowns and points. Variables are depicted by small circles. Variables belonging to the same unknown are marked by the same "color" (i.e. grey scale here), which then represents this unknown, and points are marked by large grey circles surrounding the variables living at that point. In this example, the number of variables per point varies, and only one unknown, the "black" one, lives on the whole domain. Here, $n_u = 3$, $n_v = 24$, and $n_p = 12$.
(b) Couplings between variables. In this example, the couplings are assumed to be symmetric (i.e. A is symmetric). In case of an asymmetric A, a *directed* graph would be a natural way for the visualization of couplings.



(a) (b)

Figure 2.3: (a) Couplings between the variables of the "black" unknown for the example in Fig. 2.2. (b) Point couplings for the example in Fig. 2.2.

Chapter 3

A General AMG Methodology for PDE Systems

3.1 Overview of Strategies and Model Problems

As we have discussed in the last chapter, AMG has originally been designed for solving scalar PDE applications. Extensions which have been developed so far can handle particular types of PDE systems such as those arising in linear elasticity. Important other types, for instance reaction-diffusion and drift-diffusion applications, have not been tackled successfully yet. It has turned out that different AMG strategies are suitable for different types of applications. We here describe AMG strategies for various types of applications. The description covers strategies which have already been mentioned in [71] as well as new strategies. Altogether, these strategies have led to a general AMG methodology which has systematically been realized in the AMG software SAMG. Whereas in this chapter our general methodology is discussed from a more theoretical point of view, software aspects, in particular the systematic implementation of our methodology within SAMG, are discussed in the next chapter.

The efficient solution of PDE systems without exploiting any structural information seems unrealistic. Hence, a natural starting point to create extensions of AMG for PDE systems is the exploitation of structural information, such as *variable-to-unknown (VU) mappings* and, if available, *variable-to-point (VP) mappings*, as introduced in Section 2.4.1. Such mappings provide the basis of our AMG methodology for PDE systems explained in detail in this chapter.

Our AMG methodology is developed based on a variational concept. In Section 3.1.1, we summarize the basic properties of Galerkin-based coarse-level correction processes and show, in particular, that the coarse-level correction operator $K_{h,H}$ fulfills a variational principle. Section 3.1.2 characterizes and surveys the three basic strategies of our methodology, namely variable-based, unknown-based and point-based AMG, and gives an outline of the remainder of this chapter (Sections 3.1.3 to 3.4).

Remark 3.1 For all investigations in this chapter, A is assumed to be symmetric positive definite ($A > 0$). The principal handling of symmetric positive semi-definite matrices the nullspace $\mathcal{N}(A)$ of which only consists of constant vectors, has already been described in Remark 2.8. Remark 2.9 has already pointed to sections which describe *practical* ways to treat nonpositive diagonal entries, $a_{ii} \leq 0$, arising on either the finest level or coarser levels or both.

We will extensively make use of the definitions in Sections 2.3 and 2.4. Level indices h, H are used, if necessary, to distinguish two consecutive levels. Lemmas, theorems and corollaries which can already be found in [87] are marked accordingly. ▲

3.1.1 Variational Principle

In this section, given any $A = A_h > 0$, we summarize the basic properties of Galerkin-based coarse-level correction processes. In particular, a variational principle for the coarse-level correction operator $K_{h,H}$ holds which simplifies further theoretical investigations substantially. All our AMG approaches are based on the same formulation of $K_{h,H}$, so that the following statements hold for all of them. For proofs, see [87].

We first state that the Galerkin coarse-level operator $A_H = I_h^H A_h I_H^h$ is also symmetric positive definite[1] (w.r.t. the Euclidean inner product):

Lemma 3.1 [87] *Let $A_h > 0$ hold, and let I_H^h have full rank. Then A_H is also symmetric positive definite.*

Proof. This is an immediate consequence of (2.10) and

$$\begin{aligned}(A_H v^H, w^H)_E &= (I_h^H A_h I_H^h v^H, w^H)_E = (A_h I_H^h v^H, I_H^h w^H)_E \\ &= (v^H, I_h^H A_h I_H^h w^H)_E = (v^H, A_H w^H)_E \,. \quad ▲\end{aligned}$$

The coarse-level correction operator $K_{h,H} = I_h - I_H^h A_H^{-1} I_h^H A_h$ can easily be seen to fulfill $K_{h,H}^2 = K_{h,H}$ and to be symmetric with respect to the energy inner product $(\cdot,\cdot)_1$, that is, $\forall\, v, w \;:\; (Kv, w)_1 = (v, Kw)_1$. Having these facts and $A_h, A_H > 0$ in mind, it can be proved that $K_{h,H}$ fulfills a variational principle. To be more specific, the following theorem holds:

Theorem 3.1 [87] *Let $A_h > 0$ and let any C/F-splitting and any full rank interpolation I_H^h be given. Then $K_{h,H}$ is an orthogonal projector w.r.t. the energy inner product $(\cdot,\cdot)_1$. The followings equalities hold:*

(1) $\forall\, e^h \;:\; ||K_{h,H} e^h||_1 = \min_{e^H} ||e^h - I_H^h e^H||_1 \,,$

(2) $||K_{h,H}||_1 = 1 \,.$

Statement (1), the variational principle of $K_{h,H}$, shows that Galerkin-based coarse-level corrections minimize the *energy* norm of the error w.r.t. all variations in the range of interpolation, $\mathcal{R}(I_H^h)$. Moreover, because of equality (2), a two-level method can never diverge if the smoother satisfies $||S_h||_1 \leq 1$. That this result carries over to complete V-cycles, can be shown by a recursive application of the following lemma with a zero initial guess, $\widetilde{e}^H = 0$, on each coarse level:

Lemma 3.2 [87] *Let the exact coarse-level correction e^H in* (2.12) *be replaced by any approximation $\widetilde{e}^H$ satisfying $||e^H - \widetilde{e}^H||_1 \leq ||e^H||_1$ where $||\cdot||_1$ is taken w.r.t. A_H. Then the approximate two-level correction operator still satisfies $||\widetilde{K}_{h,H}||_1 \leq 1$.*

[1] For a generalization to positive definite A_h, see Corollary A.1.

This lemma ensures a minimal robustness of each approach of our methodology: (V-)cycling will not diverge if $||S_h|| \leq 1$. However, this gives us neither information on what influences convergence concretely nor conditions under which A-independent convergence can be proved. In Sections 3.2, 3.3.2 and 3.4.4, we will investigate two-level processes for variable-, unknown- and point-based AMG, respectively, more closely in order to obtain concrete criteria for assessing smoothing, the C/F-splitting, interpolation and their interplay and then to obtain concrete statements on *two*-level convergence for certain subclasses of $\mathcal{A}_{\text{spd}}$. For reasons explained in [71, 87], A-independent *multi*level convergence cannot strictly be proved. However, it turns out that multilevel approaches converge for many discrete PDEs and PDE systems arising in practice (see Chapter 5, for instance).

3.1.2 Three Principal Strategies

Goals and Characterization Since increasingly large problems have to be solved in practical applications, our main concern is an optimal complexity[2] of $O(N)$ of our AMG approaches. In addition, we seek a reasonable compromise between robustness and black-box quality within sufficiently large classes of PDE systems, small computational work ("the magnitude of $O(N)$'s constant") and memory requirements, plug-in character, flexibility for adaptations to specific problems and extendability for covering more and more problem classes.

Clearly, there is no single algorithm satisfying all our goals for solving general PDE systems. Instead, we have developed a flexible, general AMG methodology which is based on the scalar AMG approach [87]. This methodology allows to tailor AMG components to concrete problem classes. Especially an exploitation of the VU and/or VP mapping plays a key role in our overall strategy. Concrete approaches differ "only" in the amount of information[3] they employ, and in the concrete choice of the three main components smoothing, coarsening and interpolation.

What has been achieved regarding numerical complexity can be summarized as follows. In the course of this and the following chapter, we will see that the computational work and memory requirements for the setup phase and one cycle of each concrete algorithm scale[4] with the problem size N. Therefore, if the convergence rate does not depend on N (within a problem class given), this will also be true for the overall approach for a fixed residual reduction and our main aim be fulfilled. Whereas this independence cannot be proved in a strict mathematical sense, numerical results indicate that it holds for a large number of relevant model problems (defined in Section 3.1.3) as well as practical application classes, as discussed in this chapter and Chapter 5.

Instead of more elaborate stand-alone solvers, we focus on relatively inexpensive AMG algorithms which can efficiently be used as preconditioners. In practice, this has been proved to be an efficient means to reduce computational efforts and memory requirements.

Before going into details of our methodology, an outline of its three principal strategies and its range of applicability is given in the following paragraphs.

[2] See the definition of the term **complexity** in Chapter 1.

[3] i.e., besides A and b, the VU and/or VP mapping, and/or coordinates.

[4] Note that we always assume A as well as all coarser-level matrices to be sparse. Indeed, this is observed in practice for all problem classes which are in the scope of this thesis.

Variable-Based AMG (VAMG) Our considerations begin with AMG as described in [87] since generalizations of its coarsening and interpolation concepts are used in each of our AMG approaches. AMG has been developed based on a variational concept (see Section 3.1.1) and is most mature for weakly diagonally dominant Stieltjes matrices as frequently arising in the context of *scalar* PDEs. Within our methodology, it constitutes what we call *variable-based AMG* since it solves $Av = b$ "as is", that means by ignoring unknowns and points totally. It can thus only cope with very weakly coupled PDE systems. In Section 3.2, we recall the variable-based methodology and the main results on convergence as far as important for our strategy.

Unknown-Based AMG (UAMG) Variable-based AMG can be extended in several ways in order to cope with more strongly coupled PDE systems. The most straightforward extension, the so-called *unknown-based AMG*, performs coarsening and interpolation separately for the individual unknowns by means of variable-based approaches applied to the diagonal blocks $A_{[n,n]}$ ($n = 1, \ldots, n_u$). Quite a lot of experience has been gained with this simple extension since its introduction in the early paper [71].

Unknown-based AMG and its range of applicability are discussed in detail in Section 3.3. In particular, we will see that necessary conditions for UAMG to work are that VAMG efficiently works for all $A_{[n,n]}$, and that smoothing produces an error which is smooth for each unknown separately. The last point will be seen to be closely related to the "strength" of unknown cross-couplings which are completely ignored by UAMG for constructing coarsening and interpolation. If these couplings are too "strong", this simple AMG approach may become inefficient or even fail. A new measure for the strength of unknown cross-couplings will be introduced in Section 3.3.

Point-Based AMG (PAMG) Since, for some practically very important PDE systems, the unknown cross-couplings are indeed too strong for UAMG, one of our goals is to develop approaches capable of handling such strong couplings. One important step in this direction is the observation that for many PDE systems *the different unknowns are discretized on (principally) the same grid* so that it appears to be quite natural to create the same hierarchy for all unknowns. This is in contrast to what UAMG does. A concept which addresses both issues, that is it allows for strong unknown cross-couplings and produces the same level hierarchy, is the so-called *point-based AMG*. We speak of a point-based approach if coarsening takes place on the level of points (*point-coarsening*[5]) rather than variables as before.

Section 3.4 details a *general framework for PAMG approaches* which is a main contribution of this thesis. One key concept for PAMG is that point-coarsening is performed by means of an auxiliary so-called *primary matrix* which is required to reflect the point couplings in a reasonable sense. We discuss in Section 3.4.2 various ways to define concrete primary matrices, some of which lead to "known" approaches, others to new ones. In Section 3.4.3, three general types of interpolation approaches are introduced and discussed, namely block-interpolation, multiple-unknown-interpolation and single-unknown-interpolation. For

[5] To be more specific, all variables belonging to the same point either become C or F so that, as a result, the same level hierarchy is assigned to all unknowns. For details including a discussion of some exceptions, see Section 3.4. Note that the basic idea of point-based coarsening can already be found in the early papers [71, 8].

both primary matrices and interpolation schemes, we especially focalize on the development of practical variants, that is variants which are computationally cheap ("$O(N)$ with a small constant"), efficient and applicable to relevant and sufficiently large classes of PDE systems. All our PAMG approaches make use of matrix entries and the VU and VP mappings. Only for some approaches, additional information is needed, namely information on coordinates of the grid nodes.

In Section 3.4.4, convergence of several typical PAMG approaches is proved under the assumption $A > 0$ (i.e. $A \in \mathcal{A}_{\text{spd}}$). Although many important PDE systems do not lead to matrices in $\mathcal{A}_{\text{spd}}$, in numerical tests a similar observation as in the "scalar AMG case" is made: $A > 0$ is not a necessary condition. The application of PAMG to very asymmetric drift-diffusion systems (see Section 5.3) impressively demonstrates that PAMG can efficiently work for considerable deviations from the "ideal" case.

Range of Applicability It seems clear that there exists no unique AMG algorithm which will work satisfactorily for all systems of PDEs. Instead, major work is required for developing concrete approaches for relevant classes of applications. Our methodology with its various components offers a great variety of approaches, which makes it very flexible for an application to different specific problem classes.

UAMG can efficiently solve, for instance, certain typical applications arising in the field of linear elasticity, at least if the simulation domain has a sufficiently large part of its boundary fixed by Dirichlet conditions. In such cases, UAMG works usually more efficiently than PAMG approaches. In Section 3.3, UAMG is discussed for appropriate model problems. The model problems itself are defined in Section 3.1.3. Numerical results for linear elasticity problems occurring in industrial semiconductor simulation are presented and discussed in Section 5.2.1.

The flexible point-based strategy can cope with a variety of different PDE systems. This is demonstrated for two important classes, namely reaction-diffusion and drift-diffusion problems. Throughout Section 3.4, a proper choice of primary matrices and interpolation schemes is investigated especially for these two problem classes by means of suitable model problems (defined in Section 3.1.3). We will see that for each of these two problem classes a different PAMG approach is required. Numerical tests demonstrate the effectiveness of our point-based strategy for reaction-diffusion and drift-diffusion problems arising in industrial semiconductor process and device simulation (see Sections 5.2.2 and 5.3).

In this chapter, we also consider some limits of our strategy. Problems for which none of the AMG methods discussed in this thesis works efficiently include cases where the unknowns are too strongly coupled for an application of UAMG and, at the same time, exhibit strong anisotropies in directions which are different for each of the unknowns so that a treatment by PAMG is not appropriate either.

We also draw some comparisons to other AMG approaches for solving PDE systems revealing that our general methodology covers (but is not limited to) many of the underlying ideas.

A synopsis of our AMG strategies, their main features, their range of applicability, their systematic realization and their efficiency for solving problems arising in industrial semiconductor simulation will be given in the concluding Chapter 6.

3.1.3 Model Problems

The following three model classes are used in the remainder of this chapter to illustrate several aspects of unknown- and point-based AMG. The performance of different AMG components applied to these classes is discussed, in particular, in Sections 3.3.3, 3.4.1.2, 3.4.2, 3.4.3.4, and 4.6.

For all three model classes, we assume the unit square $[0,1]^2$ to be the test domain with a uniform grid with mesh size $h = 1/2^p$ with $p \in \mathbb{N}$. We do not explicitly discuss boundary conditions, and assume, for simplicity, Dirichlet conditions everywhere.

Let ϵ with $0 \leq \epsilon \leq 1$ be given. With $\mathcal{L}_{x:\epsilon,h}$ we denote the standard 5-point stencil discretization operator (cf. (2.30)), multiplied with h^2,

$$\mathcal{L}_{x:\epsilon,h} u_h(x,y) := \left[\begin{array}{ccc} & -1 & \\ -\epsilon & 2+2\epsilon & -\epsilon \\ & -1 & \end{array} \right]_h u_h(x,y)$$

of the two-dimensional anisotropic Laplacian $-\epsilon u_{xx} - u_{yy}$ with anisotropy in the x-direction. Let $\mathcal{L}_{y:\epsilon,h}$ denote the corresponding discretization with anisotropy in the y-direction.

Let $L_{x:\epsilon} = L_{x:\epsilon,h}$ and $L_{y:\epsilon} = L_{y:\epsilon,h}$ denote the corresponding matrices emerging from a standard lexicographic numbering of the grid points. Let the abbreviation $L = L_h$ denote the corresponding matrix in the isotropic case $\epsilon = 1$. Note that

$$\lambda_{\min}(L) = 4\left(1 - \cos(\pi h)\right) \quad \text{and} \quad \lambda_{\max}(L) = 4\left(1 + \cos(\pi h)\right) \ .$$

With these basic stencils and matrices we "compose" model problems now. We make use of an unknown-wise ordering (2.16) of the variables to define the models and switch to a point-wise ordering (2.17), whenever more suitable for discussing specific aspects.

3.1.3.1 Anisotropic Vector Laplacians

The matrices $L_{x:\epsilon,h}$ and $L_{y:\epsilon,h}$ are used to define the following three PDE system model problems which comprise our first model class. We generally assume for this and the following models that a, b, c are real constants and a, b positive. The discrete PDE system

$$L_A * \left[\begin{array}{c} v_{[1]} \\ v_{[2]} \end{array} \right] = \left[\begin{array}{c} b_{[1]} \\ b_{[2]} \end{array} \right]$$

$$\text{with } L_A = L_S := L_{S,h}(\epsilon,a,b,c) := \left[\begin{array}{cc} a & c \\ c & b \end{array} \right] * \left[\begin{array}{cc} L_{x:\epsilon,h} & 0 \\ 0 & L_{x:\epsilon,h} \end{array} \right] \tag{3.1}$$

represents the anisotropic vector Laplacian where the anisotropy is in the *same* direction for the two unknowns [6]. Analogously, the above discrete PDE system with

$$L_A = L_D := L_{D,h}(\epsilon,a,b,c) := \left[\begin{array}{cc} a & c \\ c & b \end{array} \right] * \left[\begin{array}{cc} L_{x:\epsilon,h} & 0 \\ 0 & L_{y:\epsilon,h} \end{array} \right] \tag{3.2}$$

represents the anisotropic vector Laplacian where the anisotropy of the first unknown u_1 is again in the *opposite* direction than the one of the second unknown [7]. Similarly, the above

[6] The index "S" stands for "same direction" of the anisotropy of the first and second unknown.

[7] The index "D" stands for "different direction" of the anisotropy of the first and second unknown.

discrete PDE system with

$$L_A = L_X := L_{X,h}(\epsilon, a, b, c) := \begin{bmatrix} aL_{x:\epsilon,h} & cL_h \\ cL_h & bL_{y:\epsilon,h} \end{bmatrix} \tag{3.3}$$

represents another anisotropic vector Laplacian where the anisotropy of the first unknown u_1 is in the *opposite* direction than the one of the second unknown. We call the model problems in this class **anisotropic vector Laplacians** or **AVL models** throughout this thesis. More precisely, model (3.1) is called AVLS model, model, (3.2) AVLD model, and (3.3) AVLX model.

Note that, unless $\epsilon = 1$ or $c = 0$, the matrices L_D are not symmetric whereas the matrices L_S and L_X are always symmetric. Ignoring boundary conditions, we obviously get the following stencils - a pointwise ordering assumed now -

$$L_S = \begin{bmatrix} 0 & (-1) * \begin{bmatrix} a & c \\ c & b \end{bmatrix} & 0 \\ (-\epsilon) * \begin{bmatrix} a & c \\ c & b \end{bmatrix} & (2+2\epsilon) * \begin{bmatrix} a & c \\ c & b \end{bmatrix} & (-\epsilon) * \begin{bmatrix} a & c \\ c & b \end{bmatrix} \\ 0 & (-1) * \begin{bmatrix} a & c \\ c & b \end{bmatrix} & 0 \end{bmatrix}_h ,$$

$$L_D = \begin{bmatrix} 0 & \begin{bmatrix} -a & -\epsilon c \\ -c & -\epsilon b \end{bmatrix} & 0 \\ \begin{bmatrix} -\epsilon a & -c \\ -\epsilon c & -b \end{bmatrix} & (2+2\epsilon) \begin{bmatrix} a & c \\ c & b \end{bmatrix} & \begin{bmatrix} -\epsilon a & -c \\ -\epsilon c & -b \end{bmatrix} \\ 0 & \begin{bmatrix} -a & -\epsilon c \\ -c & -\epsilon b \end{bmatrix} & 0 \end{bmatrix}_h ,$$

$$L_X = \begin{bmatrix} 0 & \begin{bmatrix} -a & -c \\ -c & -\epsilon b \end{bmatrix} & 0 \\ \begin{bmatrix} -\epsilon a & -c \\ -c & -b \end{bmatrix} & \begin{bmatrix} (2+2\epsilon)a & 4c \\ 4c & (2+2\epsilon)b \end{bmatrix} & \begin{bmatrix} -\epsilon a & -c \\ -c & -b \end{bmatrix} \\ 0 & \begin{bmatrix} -a & -c \\ -c & -\epsilon b \end{bmatrix} & 0 \end{bmatrix}_h .$$

The $(L_X)_{(kk)}$ are symmetric positive definite if and only if $ab(1+\epsilon)^2 > 4c^2$. The nonvanishing $(L_X)_{(kl)}$ $(k \neq l)$ are symmetric positive definite if and only if $\epsilon ab > c^2$.

The $(L_S)_{(kk)}$ are symmetric positive definite if and only if $ab > c^2$. If the latter holds, we have $(L_S)_{(kl)} < 0$ for all non-vanishing $(L_S)_{(kl)}$ with $k \neq l$. Since each eigenvalue of L_S is of the form eigenvalue of $L_{x:\epsilon,h}$ times eigenvalue of $A_{(k,k)}$, L_S is symmetric positive definite then, and in addition of essentially block-positive type (2.34) (with the constant c of (2.34) being 1).

3.1.3.2 A Class of Reaction-Diffusion Models

As a simple model exhibiting some important properties of the reaction-diffusion systems investigated in Section 5.2.2, consider the reaction-diffusion operator

$$\begin{bmatrix} -\Delta u_1 & 0 \\ 0 & -\Delta u_2 \end{bmatrix} + \begin{bmatrix} g_1(u_1, u_2) \\ g_2(u_1, u_2) \end{bmatrix} .$$

For our model class, let $g_1 = f(x, y)u_2$ and $g_2 = f(x, y)u_1$ with a positive function f. With $-\Delta$ being discretized by means of the standard 5-point stencil, we arrive at the following matrices:

$$A = \begin{bmatrix} L & D \\ D & L \end{bmatrix} \tag{3.4}$$

where $D = (d_{ii})$ denotes a diagonal matrix with $d_{ii} = h^2 f(x_i, y_i)$. For our model class, we restrict ourselves to the following:

$$d_{ii} = \begin{cases} c := h^2 f & \text{for } 1 \le i \le n_z \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \le n_z \le n_p$ and a constant (function) $f > 0$. The model problems in this class are called **RD models** throughout this thesis.

Obviously, all "problematic" couplings are located on the block diagonal, to be more specific, on the first n_z point matrices $A_{(k,k)} = \begin{bmatrix} 4 & d_{kk} \\ d_{kk} & 4 \end{bmatrix}$ where $d_{kk} = c = h^2 f > 0$. Besides these point matrices, the unknowns are decoupled. For all k with $d_{kk} = c$, we have $A_{(k,k)} > 0$ if and only if $c < 4$. For $n_z = n_p$, the eigenvalues of A can easily be seen to be $\lambda(L) \pm c$. In this case, we arrive at $\lambda_{\min}(A) = 4\,(1 - \cos(\pi h)) - c$. Since the smallest eigenvalue of A is, for all n_z, larger or equal to the smallest eigenvalue of L minus c (due to Lemma 2.1), $A \in \mathcal{A}_{\text{spd}}$ holds if $h^2 f = c < 4\,(1 - \cos(\pi h))$. Note that $1 - \cos(\pi h) = O(h^2)$. The parameters c and n_z will be used later on to adjust the strength and number of unknown cross-couplings, respectively.

3.1.3.3 A Drift-Diffusion-Like Model Class

We consider the following class of matrix equations, mimicing some important properties of the drift-diffusion systems discussed in Section 5.3,

$$\begin{bmatrix} \lambda L & I & -I \\ -f_n * L_{x:\epsilon} & L & 0 \\ f_p * L & 0 & L \end{bmatrix} * \begin{bmatrix} v_{[1]} \\ v_{[2]} \\ v_{[3]} \end{bmatrix} = \begin{bmatrix} b_{[1]} \\ b_{[2]} \\ b_{[3]} \end{bmatrix} \tag{3.5}$$

where the vector $v_{[1]}$ shall reflect the potential ψ, $v_{[2]}$ the electron concentration n, and $v_{[3]}$ the hole concentration p. Let λ, c be positive constants, I denote the identity, and the functions f_n, f_p be defined as

$$f_n(x, y) := c \exp(10xy) \ , \quad f_p(x, y) := 1 \ .$$

We will discuss the cases

$$\epsilon \in \{\text{1e-3, 1}\}\ , \quad (\lambda, c) \in \{(1,1), (\text{1e-3,1e3}), (\text{1e-9,1e9})\}\ ,$$

so that among n and p the electron concentration n is always the "dominant" species here (see Example 3.5). The system's matrix A is not symmetric. Moreover, the larger c, the stronger the asymmetry of A. Throughout this thesis, we call the model problems in this class **DD models**.

3.2 Variable-Based AMG

In this section, we recall the essential aspects of classical, variable-based AMG (VAMG) as described in [87]. We consider the derivation of variable-based criteria and techniques for smoothing, coarsening and interpolation as well as results on convergence of two-level methods. We will follow, in principle, the considerations made in [87]. Additions are provided by Sections 3.2.3.4, 3.2.5 and 3.2.6.

This section mainly serves as a preparation for the subsequent sections since the components of VAMG, suitably modified or generalized, provide the basis for each approach which belongs to our AMG methodology for PDE systems. Under suitable conditions, most of the convergence statements will be shown to carry over to "analogous" unknown- and point-based approaches, as will be discussed in Sections 3.3.2, 3.4.1.1 and 3.4.4.

VAMG has been developed in the variational context. Hence, the theoretical investigation of convergence is most naturally done w.r.t. the energy norm, provided that $A > 0$ holds for the matrix A given. We have already seen in Section 3.1.1 that the Galerkin coarse-level operator then fulfills a variational principle. Because of this, and under natural additional assumptions, convergence of all approaches of our AMG methodology can be proved, which ensures a minimal robustness. However, since we have the solution of problem *classes* in mind - for instance the class of matrices representing a PDE discretized on increasingly finer grids and/or with varying "physical parameters" - convergence results are of practical importance only if convergence can be proved *uniformly* for relevant matrix classes $\mathcal{A}$. In the remainder of this section, we will hence discuss what has been achieved so far on that score for VAMG.

In Section 3.2.1, we will explain the central concept of algebraic smoothness and obtain a concrete measure for the *smoothing property* of a relaxation operator. In particular, it will be proved that Gauss-Seidel fulfills the smoothing property *uniformly* within the important class $\mathcal{A}_{\text{St}}$ of Stieltjes matrices, for instance. An interpretation of the smoothing property for certain important situations gives us several guidelines for designing C/F-splittings and interpolations later on.

The interplay between smoothing and coarse-level correction is studied for the two cases post- and pre-smoothing in Sections 3.2.2 and 3.2.4, respectively. In the case of post-smoothing, the focus is on criteria for suitable C/F-splittings and interpolations and on the derivation of concrete interpolation formulas, provided suitable coarser levels to be given[8]. In the case of pre-smoothing, we concentrate more on ways to improve the "quality" of smoothing

[8] Note that concrete coarsening procedures will not be discussed before Chapter 4.

and interpolation by "brute-force methods", so-called *F-smoothing* and *Jacobi interpolation*, which will prove their usefulness for "tough" problems. Post- and pre-smoothing are tightly coupled but provide a different point of view. For both cases, we can prove two-level convergence theorems with concrete upper bounds[9] for the convergence rate. Moreover, *uniform* two-level convergence can be proved, in particular, for weakly diagonally dominant Stieltjes matrices.

Remarks: If a theorem or lemma is only a special case of results for more general, point-based approaches, its proof is postponed to Section 3.4. Proofs which can already be found in [87] are mostly not repeated. Also far more examples than discussed in this section can be found in [87].

3.2.1 Algebraic Smoothness

As indicated in Section 2.3, smoothing and coarse-level correction play formally the same role as in geometric multigrid, although the term "smoothing"refers to different notions: In geometric multigrid, "smoothness"of an error is always related to two consecutive grids: An error e_h on a given fine grid is said to be **geometrically smooth** if it can be well approximated on the next coarser grid (of a predefined grid hierarchy). This notion of smoothness does not make sense in the context of algebraic multilevel approaches with automatic level construction because there is no predefined level hierarchy and, eventually, the problem to be solved has not even a geometric background at all. However, if we interprete the given smoother S as basic solver for $Av = b$, the purpose of the coarse-level correction is to accelerate this solver. Therefore, an error e is defined to be **algebraically smooth** if the smoother stalls, i.e. if $||Se|| \approx ||e||$ in some appropriate norm. In other words, an error is called algebraically smooth if something *has to be* done to speed up convergence - in the AMG context, by means of a properly constructed coarser level.

For symmetric positive definite matrices A, a concrete definition of the "smoothing property"of the relaxation operator S is introduced next. We follow the concept introduced in [8] and interprete it for certain model classes which will give us a basis for constructing AMG coarsening and interpolation.

3.2.1.1 Smoothing Property of Relaxation

The eigenvalues and -vectors of $D^{-1}A$ play a special role in investigating classical smoothers such as ω-Jacobi ($S = I - \omega D^{-1}A$) with a proper underrelaxation parameter ω and Gauss-Seidel ($S = I - Q^{-1}A$). For both schemes, eigenvectors corresponding to the *smallest* eigenvalues of $D^{-1}A$ typically determine the convergence of S interpreted as a solver of $Av = b$. The smaller the smallest eigenvalues, the slower the convergence and, hence, the "smoother"the corresponding eigenvectors (at least) in the algebraic sense.

Clearly, the smallest eigenvalues of $D^{-1}A$ can be assumed to approach zero for all relevant applications which can profit from multi-level improvements, because otherwise classical relaxation schemes would converge rapidly on their own. As an example, consider a standard elliptic PDE of second order, standard second-order discretized on a square grid

[9] The upper bounds are usually far too pessimistic, though. See Remark 3.4.

with mesh size h: The smallest eigenvalues of $D^{-1}A$ can be shown to satisfy $\lambda = O(h^2)$ and the largest ones $\lambda = O(1)$.

Remark 3.2 If the PDE mentioned previously is isotropic (e.g. Poisson-like), the smallest eigenvalues correspond to just those eigenvectors which are very smooth geometrically, and the large eigenvalues to geometrically non-smooth eigenvectors. Here, geometric coincides with algebraic smoothness. But this is not always the case and may be (for extreme non-PDE examples) even the other way around (see [87] for an example). Consequently, instead of "algebraically smooth"one should better speak of **slow-to-converge**, say. However, we stick to the term "algebraically smooth" for historical reasons. ▲

We will now make use of inner products and norms defined in Section 2.4.5 for a classification of eigenvectors of $D^{-1}A$. First, we summarize some basic relations.

Lemma 3.3 [87] *Let $A > 0$. Then the following inequalities hold for all e:*

$$||e||_1^2 \leq ||e||_0||e||_2\,, \quad ||e||_2^2 \leq \rho(D^{-1}A)||e||_1^2\,, \quad ||e||_1^2 \leq \rho(D^{-1}A)||e||_0^2 \tag{3.6}$$

The application of these norms to the eigenvectors ϕ of $D^{-1}A$ yields obviously

$$D^{-1}A\phi = \lambda\phi \Longrightarrow (\ ||\phi||_2^2 = \lambda||\phi||_1^2 \text{ and } ||\phi||_1^2 = \lambda||\phi||_0^2\ )\,. \tag{3.7}$$

Proof. The first inequality follows from Schwarz' inequality, the other two from (2.59). ■

If applied to an algebraically smooth error $e = \phi$ (λ close to zero), the above three norms are largely different in size:

$$||\phi||_2 \ll ||\phi||_1 \quad \text{and} \quad ||\phi||_1 \ll ||\phi||_0\,. \tag{3.8}$$

On the other hand, if applied to algebraically non-smooth error, the three norms are comparable in size. This observation motivates the significance of these norms: by comparing e.g. the size of the 1- (energy-) and the 2-norm it is possible to identify slow-to-converge error, giving rise to the following, central definition:

A smoothing operator S is said to satify the **smoothing property** w.r.t. a matrix $A > 0$ if for all e

$$||Se||_1^2 \leq ||e||_1^2 - \sigma||e||_2^2 \quad (\sigma > 0) \tag{3.9}$$

holds with σ being independent of e. S is said to satisfy the smoothing property w.r.t. a class $\mathcal{A}$ of matrices $A > 0$ if (3.9) holds uniformly for all $A \in \mathcal{A}$, i.e. with the same σ.

As a consequence and in accordance to the motivation of (3.9), an operator S which satisfies the smoothing property efficiently reduces an error e as long as $||e||_2$ is relatively large compared with $||e||_1$. If $||e||_2 \ll ||e||_1$, i.e. if e is algebraically smooth, S stalls.

Obviously, $\sigma||e||_2^2 \leq ||e||_1^2$ is necessary for (3.9) to hold which, because of (2.59), is equivalent to $\rho(D^{-1}A) \leq 1/\sigma$. Consequently, a necessary condition for (3.9) to hold uniformly for all $A \in \mathcal{A}$ is the uniform boundedness of $\rho(D^{-1}A)$ in $\mathcal{A}$. This is indeed satisfied

for all important subclasses of $\mathcal{A}_{\text{spd}}$ under consideration here. That especially Gauss-Seidel relaxation fulfills the smoothing property (3.9) uniformly within important matrix classes will be shown below. Let us first make the following remark.

Remark 3.3 In other applications, there might not exist an algebraic smooth error at all. For example, if $A > 0$ is strongly diagonally dominant, i.e. if it fulfills $a_{ii} - \sum_{j \neq i} |a_{ij}| \geq \delta a_{ii}$ with $\delta > 0$, we have $\rho(A^{-1}D) \leq 1/\delta$ which is equivalent to $||e||_2^2 \geq \delta ||e||_1^2$ for all e. If e was algebraically smooth, we would hence arrive at $\delta ||e||_1^2 \leq \frac{1}{\sigma} ||e||_1^2$ due to the necessary condition $\sigma ||e||_2^2 \leq ||e||_1^2$ stated above. This cannot be satisfied for sufficiently large δ. But because multi-level methods are not required to speed-up one-level methods for such matrices, we tacitly exclude such cases. ▲

In the following, important results on the smoothing properties of the classical smoothers Gauss-Seidel and Jacobi are collected. More detailed discussions of different relaxation schemes can be found in [8, 71, 87].

Theorem 3.2 [87] *Let $A > 0$ and define with any vector $w = (w_i) > 0$*

$$\gamma_- := \max_i \left\{ \frac{1}{w_i a_{ii}} \sum_{j<i} w_j |a_{ij}| \right\}, \quad \gamma_+ := \max_i \left\{ \frac{1}{w_i a_{ii}} \sum_{j>i} w_j |a_{ij}| \right\}.$$

Then Gauss-Seidel relaxation satisfies (3.9) *with $\sigma = \frac{1}{(1+\gamma_-)(1+\gamma_+)}$.*

For a **proof**, see [87]. This theorem also emerges as a special case of Theorem 3.10. ■

Gauss-Seidel satisfies the smoothing property (3.9) not only for all $A > 0$ but *uniformly* within all important classes $\mathcal{A}$ of matrices under consideration here:

- For all Stieltjes matrices, (3.9) is satisfied with $\sigma = 1/4$. This is because there exists a vector $z > 0$ with $Az > 0$. By choosing $w = z$ in Theorem 3.2, we obtain

  $$\gamma_- := \max_i \left\{ \frac{1}{z_i a_{ii}} \sum_{j<i} z_j |a_{ij}| \right\} = \max_i \left\{ 1 - \frac{1}{z_i a_{ii}} \sum_{j \leq i} z_j a_{ij} \right\} < 1 .$$

  Similarly, we obtain $\gamma_+ < 1$.
- This result carries over to each $A > 0$ which is obtained from a Stieltjes matrix by symmetrically flipping some or all off-diagonal signs.
- For each $A > 0$ with $\leq l$ nonvanishing entries per row, (3.9) is satisfied with $\sigma = 1/l^2$.
- In practice, usually $\sum_{j \neq i} |a_{ij}| \approx a_{ii}$ holds. Therefore, with $w_i \equiv 1$, γ_- and γ_+ can be expected to be close to or even less than 1. Hence, $\sigma \approx 1/4$ is a typical value for many applications.

Also ω-Jacobi with a suitable relaxation parameter ω fulfills (3.9), with $\sigma \approx 1/2$ being a typical value (see [87]).

3.2.1.2 Interpretation of Algebraically Smooth Error

The exploitation of what smooth error "looks like" is the key for constructing C/F-splittings and interpolation schemes suitable to speed up the convergence behavior of S. Therefore, in this section, algebraically smooth error is heuristically characterized for a typical smoother, namely Gauss-Seidel relaxation, and for important subclasses of $\mathcal{A}_{\text{spd}}$. Both characterizations start from the fact that, if S fulfills the smoothing property (3.9), an algebraically smooth error ($Se \approx e$) is characterized by $||e||_2 \ll ||e||_1$. With the results obtained here we will later on motivate interpolation schemes and criteria for constructing C/F-splittings.

Gauss-Seidel Relaxation In terms of the residual $r = Ae$, the inequality $||e||_2 \ll ||e||_1$ evaluates to

$$(D^{-1}r, r)_E \ll (e, r)_E \,. \tag{3.10}$$

This states an important characterization of algebraically smooth error: corresponding scaled residuals are much smaller than the errors themselves. Let us investigate this further for a typical AMG smoother. Gauss-Seidel relaxation, performed for variable v_i, corresponds to replacing v_i by $\overline{v}_i$ where

$$\overline{v}_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}v_j\right) = \frac{1}{a_{ii}}\left(a_{ii}v_i + b_i - \sum_{j} a_{ij}v_j\right) = v_i + \frac{r_i}{a_{ii}} \tag{3.11}$$

or, in terms of the corresponding error:

$$\overline{e}_i = e_i - \frac{r_i}{a_{ii}} \quad . \tag{3.12}$$

Here, r_i denotes the residual *before* the relaxation of variable v_i. For an algebraically smooth error, i.e. $\overline{e}_i \approx e_i$, we can heuristically conclude that $|r_i| \ll a_{ii}|e_i|$ and thus

$$\left|a_{ii}e_i + \sum_{j\in N_i} a_{ij}e_j\right| \ll a_{ii}|e_i| \quad . \tag{3.13}$$

That means, although the error may still be large globally, locally it can be approximated by a function of its neighboring error values e_j :

$$a_{ii}e_i + \sum_{j\in N_i} a_{ij}e_j = 0 \,. \tag{3.14}$$

In this sense, an algebraically smooth error provides some rough approximation to the solution of the *homogeneous F-equation* $A_{FF}e_F + A_{FC}e_C = 0$. In Section 3.2.3, the derivation of interpolation will directly be based on this equation.

Interpretation for Subclasses of $\mathcal{A}_{\text{spd}}$ Another way of interpreting smooth error is based on the fact that, because of Lemma 3.3, the inequality $||e||_2 \ll ||e||_1$ implies $||e||_1 \ll ||e||_0$. For all matrices A, $||e||_1^2$ can be split as follows:

$$||e||_1^2 = \frac{1}{2}\sum_{i,j}(-a_{ij})(e_i - e_j)^2 + \frac{1}{2}\sum_{i,j} a_{ij}(e_i^2 + e_j^2)\,. \tag{3.15}$$

Since A is assumed to be symmetric, this can further be simplified to

$$||e||_1^2 = \frac{1}{2}\sum_{i,j}(-a_{ij})(e_i - e_j)^2 + \sum_i s_i e_i^2 \tag{3.16}$$

with s_i being the i-th row sum of A,

$$s_i := s_i(A) := \sum_j a_{ij}\,. \tag{3.17}$$

The inequality $||e||_1 \ll ||e||_0$ is now reformulated for the class $\mathcal{A}_{\text{ess}}$. A matrix $A > 0$ of essentially positive type has to fulfill (2.31). Due to this and the splitting (3.16), the inequality $||e||_1 \ll ||e||_0$ is equivalent to

$$\frac{c}{2}\sum_{i,j}(-a_{ij}^-)(e_i - e_j)^2 + \sum_i s_i e_i^2 \ll \sum_i a_{ii} e_i^2 \quad .$$

In the most important case of $s_i \geq 0$, we thus have, on the average for each i,

$$\frac{c}{2}\sum_{j\neq i}\frac{-a_{ij}^-}{a_{ii}}\frac{(e_i - e_j)^2}{e_i^2} \ll 1$$

which can be interpreted as follows: An algebraically smooth error varies slowly in the direction of large negative connections, i.e. from e_i to e_j if $|a_{ij}|/a_{ii}$ is relatively large. Even if there are positive couplings in the matrix, an algebraically smooth error changes slowly in their direction, as long as they are not too large - i.e. as long as there exist strong negative paths for them (see Section 2.4.4 for examples). Since a subclass of $\mathcal{A}_{\text{ess}}$ is the class $\mathcal{A}_{\text{St}}$ of Stieljes matrices, the above interpretation directly carries over to $\mathcal{A}_{\text{St}}$ but with the simplification that there are no positive off-diagonal entries.

In the general case, however, if $a_{ij} > 0$ exceeds a certain size, the above cannot be expected to be true any more. This can be seen by transforming the equality (3.16) further. The value,

$$t_i := a_{ii} - \sum_{j\neq i}|a_{ij}|\,,$$

which can be used as a measure of diagonal dominance (cf. (2.26)), is related to s_i by $s_i = t_i + 2\sum_{j\neq i} a_{ij}^+$. Therefore, we arrive at

$$||e||_1^2 = \frac{1}{2}\sum_i\left(\sum_{j\neq i}|a_{ij}^-|(e_i - e_j)^2 + \sum_{j\neq i} a_{ij}^+(e_i + e_j)^2\right) + \sum_i t_i e_i^2\,. \tag{3.18}$$

Assuming $t_i \geq 0$ (weak diagonal dominance), $||e||_1 \ll ||e||_0$ now means, on the average for each i,

$$\sum_{j \neq i} \frac{|a_{ij}^-|}{a_{ii}} \frac{(e_i - e_j)^2}{e_i^2} + \sum_{j \neq i} \frac{|a_{ij}^+|}{a_{ii}} \frac{(e_i + e_j)^2}{e_i^2} \ll 1 \ . \tag{3.19}$$

Consequently, if a_{ij} is *positive* and a_{ij}/a_{ii} relatively large, e_j tends to approximate $-e_i$ relatively to the size of e_i. This leads to the following general conclusion:

For all (approximatively) weakly diagonally dominant matrices $A > 0$, relaxation schemes which satisfy the smoothing property (3.9) *smooth the error along large* negative *connections, but tend to oscillate along large* positive *connections.*

3.2.2 Post-smoothing and Two-Level Convergence

We now combine the already gained results on the variational principle of $K = K_h^H$ and the smoothing property to obtain a two-level convergence estimate which is stated in Theorem 3.3 below. Afterwards, we discuss the so-called τ-condition of interpolation which represents a sufficient condition for the convergence estimate to hold and will serve as a measure to assess the quality of interpolation. Together with the above interpretations of algebraic smoothness, it will be used to derive concrete interpolation formulas.

We adopt, as before, the theoretical approach introduced in [8] and further developed in [71, 87] and investigate the case of post-smoothing. We assume one smoothing step per cycle, that is, the two-level operator to be considered is SK.

3.2.2.1 A Theorem on Two-Level Convergence

Our goal now is to have SKe as small as possible. Hence, for a good interplay between smoothing and coarse-level correction, the error after the coarse-level correction step, Ke, needs to be "relaxable"so that the smoother S can effectively work again. Assuming the smoothing property (3.9) to be fulfilled, Ke should therefore be far away from being algebraically smooth, that is we want $||Ke||_2$ to be bounded from below by $||Ke||_1$. That this condition ensures convergence of the two-level method SK is stated in the following theorem:

Theorem 3.3 [87] *Let $A > 0$ and let S satisfy the smoothing property* (3.9)*. Furthermore, assume the C/F-splitting and interpolation to be such that*

$$\forall\, e \ : \ ||Ke||_1^2 \leq \tau ||Ke||_2^2 \tag{3.20}$$

with some $\tau > 0$ being independent of e. Then $\tau \geq \sigma$ and $||SK||_1 \leq \sqrt{1 - \sigma/\tau}$.

Proof. Combine (3.9), (3.20) and $||K||_1 = 1$. ■

Remark 3.4 We want to stress already here that Theorem 3.3 and the following theorems can be used to investigate *uniform* two-level convergence and to obtain principal guidelines.

The concrete upper bounds obtained, however, are usually far too pessimistic. For instance, for the matrices L, i.e. the Poisson equation discretized by means of the standard five-point stencil on the unit square (for a definition of L, see Section 3.1.3), $\tau = 2$ (for an explanation, see Remark 3.8), and $\sigma = 4/9$ (see Section 3.3.3.1), we obtain $\sqrt{1 - \sigma/\tau} = \sqrt{1 - \frac{4}{9} \cdot \frac{1}{2}} \approx 0.88$. This upper bound cannot explain the efficiency of AMG, observed in practice, for the ideal model case of a well-discretized Poisson equation. ▲

3.2.2.2 The τ-Condition of Interpolation

Condition (3.20) is difficult to examine in practice and does not really embody what we need: practical measures which help to judge directly the quality of our C/F-splitting and interpolation and which help to create suitable splittings and interpolation operators. The next theorem on a sufficient condition for (3.20) is an important step in this direction:

Theorem 3.4 [87] *If $A > 0$ and if the C/F-splitting and interpolation I_{FC} are such that*

$$\forall\, e \,:\, ||e_F - I_{FC} e_C||^2_{0,F} \leq \tau ||e||^2_1 \tag{3.21}$$

with τ being independent of e, then (3.20) *is satisfied.*

Henceforth, (3.21) is called the **τ-condition of interpolation**. It will serve as a measure for the quality of the C/F-splitting and interpolation, not only for variable-based but - generalized in a straightforward way - also for unknown- and point-based approaches.

The τ-condition (3.21) is non-trivial only for an algebraically smooth error e. This can easily be seen if we apply (3.21) to the eigenpairs (ϕ, λ) of $D^{-1}A$ and use Lemma 3.3:

$$||\phi_F - I_{FC}\phi_C||^2_{0,F} \leq \lambda\tau ||\phi||^2_0 \,.$$

This implies a non-trivial condition only for those ϕ which correspond to the small eigenvalues of $D^{-1}A$, i.e. the algebraically smooth eigenvectors. Critical for a uniform fulfillment of the τ-condition is therefore the accurate interpolation of those eigenvectors the eigenvalues of which tend to zero if A varies in $\mathcal{A}$.

Remark 3.5 As mentioned in Section 3.2.1.1, in case that $\mathcal{A}$ consists of the matrices $A = A_h$ emerging from the standard second-order discretization of an isotropic elliptic problem on grids with mesh size h, algebraically smooth eigenvectors of $D^{-1}A$ are also geometrically smooth and their eigenvalues satisfy $\lambda = O(h^2)$. In such cases, (3.21) is closely related to the requirement of first-order interpolation. ▲

As already pointed out in discussing the smoothing property, we are interested in uniform convergence within reasonable classes $\mathcal{A}$ of matrices A, representing e.g. similar problems on differently accurate grids. In the last section, it was already shown that for important subclasses of $\mathcal{A}_{\text{spd}}$ the standard Gauss-Seidel relaxation fulfills the smoothing property uniformly. Hence, to have also uniform two-level convergence in the sense of Theorem 3.3 within a class $\mathcal{A}$, it is sufficient to develop criteria for suitable C/F-splittings and for the construction of concrete interpolations which satisfy the τ-condition (3.21) uniformly within $\mathcal{A}$.

3.2.3 Interpolation Schemes

In the following, we explain the general variable-based approach for constructing interpolation and state theorems on the quality of concrete basic interpolation schemes measured in terms of two-level convergence. In particular, we explain the relationships of interpolation to both smoothing and C/F-splitting since the interplay of these three processes determines the efficiency of the overall approach. For our concrete procedure in defining these processes, this means that we discuss how our interpolation schemes are derived from properties of the smoothing property, how the C/F-splitting and interpolation influence each other, and which conditions on the C/F-splitting can be derived[10]. Theorems from [87] on two-level convergence will be stated for concrete basic interpolation schemes. In particular, we will see that uniform convergence for important subclasses of $\mathcal{A}_{\text{spd}}$ can be guaranteed. The "ideal" case will be seen to be the class of weakly diagonally dominant Stieltjes matrices ($\mathcal{A}_{\text{wdd}} \cap \mathcal{A}_{\text{St}}$). We will also make remarks on handling matrices deviating from this ideal case. See especially Sections 3.2.3.3 and 3.2.3.4.

In order to motivate the general approach in constructing interpolation, we recall that (3.21) is a nontrivial condition only for algebraically smooth error. For such error, however, we have seen in Section 3.2.1.2 that the homogeneous F-equations (3.14),

$$a_{ii}e_i + \sum_{j \in N_i} a_{ij}e_j = 0 \quad (i \in F) \ , \tag{3.22}$$

are approximately satisfied. Consequently, the definition of interpolation will also be based on these equations. This means that the definition of the interpolation weights w_{ij} in

$$e_i = \sum_{j \in P_i} w_{ij}e_j \quad (i \in F) \tag{3.23}$$

has to be such that (3.23) approximates (3.22) for all $i \in F$.

Remark 3.6 Variables which are (nearly) not coupled to any other variable, corresponding to matrix rows with all off-diagonal entries being (nearly) zero, do not require any interpolation. Often, they arise from Dirichlet boundary conditions and are called **essentially isolated variables**. Of course, such variables i will always become F-variables with "empty"interpolation formulas (3.23): $w_{ij} \equiv 0$. For simplicity, they are *tacitly excluded* in the following. ▲

We always assume in this section that we have already determined a C/F-splitting and sets of interpolatory variables, P_i $(i \in F)$. The C/F-splitting and the P_i are, however, very important for the quality of the interpolation itself. Generally, in order to allow the F-variables to be interpolated from C-variables, *the splitting has to be such that each F-variable has a "sufficiently strong connection" to the set of C-variables* (see also the conditions a proper C/F-splitting should fulfill as discussed in Section 4.2.1.1). Although this connection does not necessarily have to be via *direct* couplings, in the following we only consider, for ease of description, the so-called **direct interpolation** where the P_i are subsets of $C \cap N_i$, that is an

[10] Concrete algorithms for splitting a set Ω into C and F, however, will be explained in Section 4.2.

F-variable i is interpolated from a subset of the C-variables which belong to the neighborhood N_i of i. In practice, however, *indirect interpolation* schemes, as *standard interpolation* and *multipass-interpolation*, play an important role. Some remarks on these are made in Section 3.2.3.5, and detailed explanations of corresponding algorithms can be found in Section 4.3. There, also other types of interpolation weights[11] are discussed.

3.2.3.1 Stieltjes Matrices

We have seen in Section 3.2.1.2 that for weakly diagonally dominant Stieljes matrices algebraically smooth error varies slowly in the direction of strong *negative* couplings. To be more specific, we define a variable i to be **strongly (negatively) coupled** to a variable j ($i \neq j$) if

$$-a_{ij} \geq \epsilon_{\mathrm{str}} \max_{j \neq i} |a_{ij}^{-}| \ , \tag{3.24}$$

with some $\epsilon_{\mathrm{str}} > 0$, a typical value being 0.25 [12].

An error at a variable i is thus essentially determined by the weighted average of the error at the variables j it is strongly coupled to, i.e. its "strong neighbors". Consequently, assuming $\emptyset \neq P_i \subseteq C \cap N_i$, the more strong couplings of any F-variable i are contained in P_i, the better will

$$\frac{1}{\sum_{j \in P_i} a_{ij}} \sum_{j \in P_i} a_{ij} e_j \approx \frac{1}{\sum_{j \in N_i} a_{ij}} \sum_{j \in N_i} a_{ij} e_j \tag{3.25}$$

be satisfied for smooth error. This suggests approximating (3.22) by

$$a_{ii} e_i + \alpha_i \sum_{j \in P_i} a_{ij} e_j = 0 \quad \text{with } \alpha_i = \frac{\sum_{j \in N_i} a_{ij}}{\sum_{j \in P_i} a_{ij}} \tag{3.26}$$

which results in an interpolation scheme (3.23) with matrix-dependent, positive weights

$$w_{ij} = -\alpha_i a_{ij} / a_{ii} \quad (i \in F,\ j \in P_i) \ . \tag{3.27}$$

Theorem 3.5 below states that this interpolation scheme fulfills a τ-condition (3.21). Note that the row sums of (3.22) and (3.26) are equal and we have

$$a_{ii} \left(1 - \sum_{j \in P_i} w_{ij} \right) = s_i \tag{3.28}$$

which shows that $\sum_{j \in P_i} w_{ij} = 1$ if $s_i = 0$. Hence, constants are interpolated exactly in the limit case of a zero row sum matrix[13].

[11] i.e. except of weights based on entries of A as considered here, an example for an alternative being weights based on coordinates.

[12] We will use this measure of strong connectivity in Section 4.2 to construct the C/F-splitting.

[13] For regular matrices, this is not the case. Instead, the weights are chosen so that $I_{FC} I_C$ equals the result of one Jacobi step applied to (3.22) with the vector $e = 1_C$ (i.e. vector with all components being ones) as the starting vector. Cf. also Section 3.2.4 for *relaxation of interpolation*.

3.2.3.2 Essentially Positive Type Matrices

Also for matrices $A \in \mathcal{A}_{\text{ess}}$ with $s_i \geq 0$ for all i, algebraically smooth error varies slowly in the direction of strong *negative* couplings (see Section 3.2.1.2). For them, we could use the same interpolation scheme as described above. Remember that $a_{ij} < 0$ for all $j \in P_i$, and that, for all $A \in \mathcal{A}_{\text{ess}}$, each row containing off-diagonal entries has at least one negative off-diagonal entry (see Section 2.4.4). The weights (3.27) are positive again.

However, in practice, we want to implement an interpolation which - at least formally - can be employed for matrices which do not strictly fulfill the conditions $A \in \mathcal{A}_{\text{ess}}$ and $s_i \geq 0$ for all i, for instance, because they contain some particularly large positive off-diagonal entries. Other typical examples are provided by the discretizations (2.32) or (2.33) with Dirichlet boundary conditions. The resulting matrices A are in $\mathcal{A}_{\text{ess}}$ but do not fulfill $s_i \geq 0$ near *boundaries*. In such cases, $\sum_{j\in N_i} a_{ij}$ might become zero or even positive for certain $i \in F$, and we would obtain zero or even negative interpolation weights. According to the heuristic considerations in Section 3.2.1.2, it can be assumed for matrices $A \in \mathcal{A}_{\text{ess}}$ with $s_i \geq 0$ for all i that an algebraically smooth error satisfies

$$\sum_j a_{ij}^+ e_j \approx \sum_j a_{ij}^+ e_i \quad (i \in F) \tag{3.29}$$

which, for $j \neq i$, requires $e_j \approx e_i$ or a_{ij}^+ to be small relatively to a_{ii}. This suggests adding all positive entries to the diagonal. We use

$$\tilde{a}_{ii} e_i + \alpha_i \sum_{j\in P_i} a_{ij}^- e_j = 0 \quad \text{with } \tilde{a}_{ii} = a_{ii} + \sum_{j\in N_i} a_{ij}^+ \,,\; \alpha_i = \frac{\sum_{j\in N_i} a_{ij}^-}{\sum_{j\in P_i} a_{ij}^-} \tag{3.30}$$

instead of (3.26), which yields in all cases positive weights

$$w_{ij} = -\alpha_i a_{ij}^- / \tilde{a}_{ii} \quad (i \in F,\, j \in P_i) \;. \tag{3.31}$$

The row sums of (3.30) and (3.22) are equal, and

$$\tilde{a}_{ii} \left(1 - \sum_{j\in P_i} w_{ij} \right) = s_i \tag{3.32}$$

so that constants are interpolated exactly in the limit case of a zero row sum matrix.

The above interpolation can formally be applied to any matrix $A > 0$ and any C/F-splitting provided that $P_i \subseteq C \cap N_i^-$ and $P_i \neq \emptyset$ for each $i \in F$. The following theorem holds:

Theorem 3.5 [87] *Let $A \in \mathcal{A}_{\text{ess}}$ with all $s_i \geq 0$. With fixed $\tau \geq 1$ select a C/F-splitting so that, for each $i \in F$, there is a set $P_i \subseteq C \cap N_i^-$ satisfying*

$$\sum_{j\in P_i} |a_{ij}^-| \geq \frac{1}{\widetilde{\tau}} \sum_{j\in N_i} |a_{ij}^-| \,. \tag{3.33}$$

Then the interpolation (3.23) *with weights* (3.31) *satisfies the τ-condition* (3.21) *with $\tau = \widetilde{\tau}/c$ and the c of* (2.31).

Proof. A direct proof can be found in [87]. Additionally, this theorem emerges as a special case of Corollary 3.1 (see Corollary 3.2 in Section 3.4.4.2). ■

This theorem shows that the τ-condition (3.21) *can be satisfied uniformly, for instance, within the class of weakly diagonally dominant Stieltjes matrices whenever the sets P_i ($\subseteq C \cap N_i^-$) are reasonably large.*

Remark 3.7 Criterion (3.33) shows that *strong* couplings enhance convergence, but *weak* ones only increase computational efforts. Therefore, to satisfy (3.33) with as few C-variables as possible, the splitting should be arranged such that C-variables are only chosen from the strongest couplings of every F-variable. This just means coarsening in the direction of smoothness. See also the discussion in Section 4.2.1.1. ▲

Remark 3.8 Obviously, the concrete choice of τ is crucial: on one hand, the larger τ, the weaker is assumption (3.33) and the faster the coarsening can be, but the two-level convergence will be very slow. On the other hand, $\tau = 1$ gives best convergence, but forces *all* neighbors of $i \in F$ into C. As discussed in [87], this approach, applied recursively, will result in an extremely inefficient (direct) solver. A reasonable compromise is $\tau = 2$ which means, for $A \in \mathcal{A}_{\text{St}}$, that about 50% of the total strength of connections of every F-variable has to be represented on the next coarser level. In practice, however, coarsening may still be too slow, especially for matrices which have many row entries of similar size. ▲

Remark 3.9 Other variants of interpolation weights, which are usually less efficient or sometimes even not defined, are discussed in [87]. ▲

The requirement of $s_i \geq 0$ for all i in the previous theorem is sufficient but not necessary as the following theorem shows. However, the two-level convergence rate suffers from negative row sums:

Theorem 3.6 [87] *Let A be an essentially positive-type matrix with $s_i \geq -\kappa$ with some $\kappa \geq 0$. Assume $(Ae, e)_E \geq \epsilon(e, e)_E$ for all e with some $\epsilon > 0$. With fixed $\tau \geq 1$, select a C/F-splitting as in Theorem 3.5. Then the interpolation* (3.23) *with weights* (3.31) *satisfies the τ-condition* (3.21) *with τ replaced by some $\tilde{\tau} = \tilde{\tau}(\epsilon, \kappa, c, \tau)$. As a function of ϵ and κ, we have $\tilde{\tau} \to \infty$ if $\kappa \to \infty$ or $\epsilon \to 0$.*

Example: This theorem can be applied, for instance, to matrices A corresponding to the stencils (2.32) or (2.33) with Dirichlet boundary conditions, since they are in $\mathcal{A}_{\text{ess}}$ but $s_i \geq 0$ does not hold near boundaries. ▲

Remark 3.10 While uniform smoothing is guaranteed in $\mathcal{A}_{\text{St}}$, the above theorem shows that the τ-condition of interpolation cannot be expected to be fulfilled uniformly in this complete class, only in subclasses, as for instance the class $\mathcal{A}_{\text{St}} \cap \mathcal{A}_{\text{wdd}}$ (see box above).

A counterexample is the subclass of matrices $A = A_c$ defined by discretizing the Helmholtz operator $-\Delta - cI$ on the unit square with Dirichlet boundary conditions and with fixed meshsize h. For discretizing $-\Delta$, the standard five-point stencil is used. In lexicographic

numbering, we obtain $A_c = \frac{1}{h^2} L_h - cI_h$. Let λ_0 be the smallest eigenvalue of $\frac{1}{h^2} L_h$ and e_0 be a corresponding eigenfunction (normalized so that $||e_0||_E = 1$). We then have $||e_0||_1^2 = \lambda_0 - c$. Therefore, for the τ-condition (3.21) to hold uniformly, its left hand side has to approach zero if c approaches λ_0. This means that the first eigenfunction (e_0) of the Laplace operator has to be approximated with increasing accuracy if $c \to \lambda_0$, which is normally not true unless special interpolation techniques are used. ▲

Remark 3.11 Not in all cases, the first eigenfunctions produce problems as in the example above. For instance, if A is a zero row sum matrix (a limit case since we consider $A > 0$ in this chapter), the constant functions are eigenfunctions. They, however, do not produce such problems since they are interpolated exactly by our interpolation schemes. ▲

3.2.3.3 General Case

As has been shown in [87] for an example, the approximation (3.30) becomes less accurate in the sense of (3.21) if (3.29) is strongly violated. This indicates that the treatment of positive couplings in interpolation is very critical in general. For instance, we have seen in Section 3.2.1.2 for matrices $A \in \mathcal{A}_{\text{spd}}$ which are approximately weakly diagonally dominant that we have to expect an oscillatory behavior of algebraically smooth error if $a_{ij} > 0$. The oscillations are the stronger the larger a_{ij} is relative to a_{ii} (see (3.19)). However, we can expect that those e_j corresponding to positive couplings $a_{ij} > 0$ change slowly *among each other* unless a_{ij} is so small that it can be neglected. This gives rise to the following generalization of the interpolation scheme (3.27) which is completely symmetric in handling negative and positive couplings.

If a variable i has both negative and positive couplings and the C/F-splitting is such that both $C \cap N_i^- \supseteq P_i^- \neq \emptyset$ and $C \cap N_i^+ \supseteq P_i^+ \neq \emptyset$ hold, we can use the approximation

$$a_{ii} e_i + \alpha_i \sum_{j \in P_i^-} a_{ij} e_j + \beta_i \sum_{j \in P_i^+} a_{ij} e_j = 0 \tag{3.34}$$

$$\text{with} \quad \alpha_i = \frac{\sum_{j \in N_i^-} a_{ij}}{\sum_{j \in P_i^-} a_{ij}} \quad \text{and} \quad \beta_i = \frac{\sum_{j \in N_i^+} a_{ij}}{\sum_{j \in P_i^+} a_{ij}} \tag{3.35}$$

to define the interpolation scheme. The following interpolation weights emerge:

$$w_{ij} = \begin{cases} -\alpha_i a_{ij}/a_{ii} & (j \in P_i^-) \ , \\ -\beta_i a_{ij}/a_{ii} & (j \in P_i^+) \ . \end{cases} \tag{3.36}$$

We have $w_{ij} > 0$ $(j \in P_i^-)$ and $w_{ij} < 0$ $(j \in P_i^+)$. If either $N_i^+ = \emptyset$ or $N_i^- = \emptyset$, these definitions are to be modified in a straightforward way by setting $P_i^+ = \emptyset$, $\beta_i = 0$ or $P_i^- = \emptyset$, $\alpha_i = 0$, respectively. In particular, for Stieltjes matrices, the above interpolation is identical to (3.27). The row sums of (3.34) and (3.22) are equal and

$$a_{ii} \left(1 - \sum_{j \in P_i} w_{ij} \right) = s_i \ , \tag{3.37}$$

which shows that also here constants are interpolated exactly if all row sums are zero. Analogously to Theorem 3.5, we obtain

Theorem 3.7 [87] *Let $A > 0$ and $t_i = a_{ii} - \sum_{j\in N_i} |a_{ij}| \geq 0$. With fixed $\tau \geq 1$ select a C/F-splitting such that the following holds for each $i \in F$: If $N_i^- \neq \emptyset$, there is a set $P_i^- \subseteq C \cap N_i^-$ satisfying*

$$\sum_{j\in P_i^-} |a_{ij}| \geq \frac{1}{\tau} \sum_{j\in N_i^-} |a_{ij}| \tag{3.38}$$

and, if $N_i^+ \neq \emptyset$, there is a set $P_i^+ \subseteq C \cap N_i^+$ satisfying

$$\sum_{j\in P_i^+} a_{ij} \geq \frac{1}{\tau} \sum_{j\in N_i^+} a_{ij} \,. \tag{3.39}$$

Then the interpolation (3.23) *with weights* (3.36) *satisfies the τ-condition* (3.21)*.*

Proof. A direct proof can be found in [87]. Additionally, this theorem emerges as a special case of Corollary 3.1 (see Corollary 3.2 in in Section 3.4.4.2). ■

Analogously to Theorem 3.6, a straightforward extension of the above theorem to the case $t_i \geq -\kappa$ with some $\kappa \geq 0$ can also be proved.

The above interpolation scheme, which has been developed for matrices $A \in \mathcal{A}_{\text{wdd}}$, can be used for all matrices, at least technically. However, its "quality" will suffer considerably, in particular, from a violation of weak diagonal dominance. In general, the treatment of large positive off-diagonal entries is critical for AMG's efficiency.

3.2.3.4 Elimination of Positive Couplings

The question arises if and how we can handle matrices $A \in \mathcal{A}_{\text{spd}} \setminus \mathcal{A}_{\text{wdd}}$ appropriately. In particular, it is an open question in general how we should treat large positive entries when constructing coarsening and interpolation. Although a general answer might not exist, a remedy which we call *elimination of positive couplings* often helps in practice increasing the robustness of AMG for matrices with large positive entries. In contrast to incorporating such entries explicitly, as done in the last section for constructing interpolation, this remedy tries to "get rid" of (large) positive entries. It can heuristically be motivated as follows.

An example for a practically relevant matrix $A \in \mathcal{A}_{\text{spd}}$ which is *not* in $\mathcal{A}_{\text{wdd}}$ is the matrix which corresponds to the stencil (2.28) with $\epsilon = 0$ and $\alpha = 1/4$,

$$\frac{1}{h^2} \begin{bmatrix} -1/4 & -1 & -1/4 \\ 1/2 & 2 & 1/2 \\ -1/4 & -1 & -1/4 \end{bmatrix}_h \,. \tag{3.40}$$

For instance, Gauss-Seidel relaxation for A produces errors which are (geometrically and algebraically) smooth in y-direction. However, a coarsening process based on (3.24) does not

detect the direction of smoothness. Although it ignores the positive entries $1/2$, as is appropriate for the above matrix, it regards the entries $-1/4$ as being strong due to the standard choice $\epsilon_{\rm str} = 1/4$. A simple remedy for the above matrix would be an $\epsilon_{\rm str} > 0.25$. This workaround, however, does not help in general since it does not address an appropriate handling of large positive matrix entries. Often, a better way to decide on strength of connectivity is to *"eliminate" (large) positive entries by inserting their corresponding stencils* (only for and) prior to deciding on the strength of *negative* couplings. For the above stencil, one such elimination step produces the stencil

$$\frac{1}{h^2}\begin{bmatrix} 1/14 & 0 & -1 & 0 & 1/14 \\ -2/14 & 0 & 2 & 0 & -2/14 \\ 1/14 & 0 & -1 & 0 & 1/14 \end{bmatrix}_h , \tag{3.41}$$

The positive off-diagonals have become rather small, so have the negative off-diagonals in x-direction so that a coarsening process based on the standard criterion (3.24) correctly identifies the real direction of smoothness, the y-direction.

The above elimination can be regarded as a first step in employing geometry implicitly. It tries to find the "real" negative couplings reflecting the directions of smoothness.

Eliminations of positive couplings can also improve interpolation and can be incorporated into "indirect" interpolation schemes discussed next. Variants implemented within SAMG are mentioned in Section 4.3.

3.2.3.5 Indirect Interpolation

So far, we have constructed interpolation based on *direct* connections, that is, an F-variable i is interpolated only from C-variables in its direct neighborhood N_i. Correspondingly, C/F-splittings have to be such that each $i \in F$ is sufficiently strongly connected to the set of C-variables via *direct* connections.

Although a strong F-to-C connectivity is indeed crucial, it does not necessarily have to be via direct connections. In fact, this may limit the quality of interpolation and, closely related, the speed of coarsening. Whereas, on one hand, a too slow coarsening will result in high memory requirements, a faster coarsening, on the other hand, typically implies a slower convergence. However, advantages in terms of less memory requirements and less computational work for the setup and per cycle often outweigh the disadvantage of slower convergence. In many cases, it pays to employ a computationally cheaper AMG variant as a preconditioner rather than a more expensive AMG approach - regardless if the latter is used stand-alone or as a preconditioner (see also the discussion in Section 4.4).

For an illustration, consider the following situation, the typical geometric scenario of isotropic five-point discretizations on regular meshes. Interpolation based only on direct connections would not allow for the $h \to 2h$ coarsening which is typically used in GMG methods, the reason being that those F-variables i sitting in the center of a coarse-grid cell have no direct connection to the C-variables (graphs illustrating the situation can be found in [87]). However, all their direct F-neighbors, j, *do have* strong connections to the C-variables and can thus be interpolated directly.

A straightforward generalization of interpolation is obtained by interpolating the j-variables first and then, via the resulting interpolation formulas, the i-variable. This approach can

be applied in several variants. It can be used solely for F-variables lacking a strong connection to C, or it can be used as a means to increase the quality of interpolation by "enlarging" the formulas of direct interpolation. Depending on the application, such variants increase the robustness of interpolation substantially.

In a straightforward way, elementary but technically rather involved, we obtain corresponding theorems on the fulfillment of the τ-condition (3.21) which are analogous to the above stated ones. Practical indirect interpolation schemes, the so-called (extended) standard and multipass interpolation, will be considered in Section 4.3.

3.2.4 Pre-smoothing and Two-Level Convergence

The last three Sections (3.2.1 to 3.2.3) were mainly concerned with minimizing the energy norm of the two-level iteration operator SK in case of (one) post-smoothing (step). The starting point was the direct implication of the *smoothing property* on the interpolation which led to the τ-condition (3.21). Criteria for the fulfillment of this condition were then investigated for concrete interpolation formulas the development of which was inspired by the exploitation of algebraic smoothness. The interplay of smoothing and interpolation was thus analyzed based on the concept of algebraic smoothness.

In this section, we summarize a very different approach, considered in [48, 87], for the investigation of this interplay and for the construction of rapidly converging AMG methods. This approach aims at *forcing* the right hand side of the variational principle,

$$||K_{h,H} S_h^\nu e^h||_1 = \min_{e^H} ||S_h^\nu e^h - I_H^h e^H||_1 \tag{3.42}$$

(see Theorem 3.1) to become small. For this purpose, two "brute-force" methods, the so-called *F-smoothing* and *Jacobi interpolation*, are employed. Theorem 3.8 below proves an upper bound for the two-level convergence of the resulting "brute-force" AMG approach. Moreover, we will see that the condition on Jacobi interpolation employed in this theorem is closely related to the τ-condition (3.21) discussed in Section 3.2.2. The convergence of the "brute-force" AMG approach is *uniform* for the same matrix classes for which the approach described in Section 3.2.2 converges uniformly. In addition, we discuss in brief advantages in terms of convergence if full smoothing instead of mere F-smoothing is used and state that Gauss-Seidel relaxation with "CF-ordering" may be preferable to a "lexicographic ordering".

The main reason for discussing the two "brute-force" methods for smoothing and interpolation here is that each of them can help to improve convergence for some "tough" applications. Properly applied to the respective AMG strategy - based on variables, unknowns or points -, they provide us with tools to enhance our "conventional" AMG approaches, as will be demonstrated in Section 5.3.2 for a class of very ill-conditioned matrices.

3.2.4.1 Convergence Using Mere F-relaxation

The basic idea behind the approach considered here is the fact that, for all (scalar) applications we have in mind here, the submatrix A_{FF} can easily be forced to be very well conditioned,

for instance strongly diagonally dominant,

$$a_{ii} - \sum_{j \in F, j \neq i} |a_{ij}| \geq \delta a_{ii} \quad (i \in F) \tag{3.43}$$

with some fixed, predefined $\delta = \delta(A_{FF}) > 0$ (see (2.26)) - if we choose the C/F-splitting accordingly. Assuming (3.43) to hold, the solution of the *F-equation*,

$$A_{FF} e_F + A_{FC} e_C = 0 \tag{3.44}$$

(with frozen e_C), can efficiently be approximated, for instance, by relaxation, in the following called **F-relaxation**. Using this as the basis for both smoothing *and* interpolation, the right hand side of (3.42) can be forced to become arbitrarily small. This is shown in the following.

One **F-smoothing** step is defined to be a mapping $v \to \bar{v}$ where

$$Q_{FF} \bar{v}_F + (A_{FF} - Q_{FF}) v_F + A_{FC} v_C = b_F, \; \bar{v}_C = v_C \, . \tag{3.45}$$

In the case of Gauss-Seidel relaxation, Q_{FF} denotes the lower triangular part of A_{FF} including the diagonal, in the case of Jacobi relaxation, we would have $Q_{FF} = D_{FF}$. However, we only use Gauss-Seidel in practice. The corresponding smoothing operator S_h, mapping the corresponding error quantities $e \to \bar{e}$, reads

$$S_h^\nu e = \begin{pmatrix} S_{FF}^\nu (e_F - \widehat{e}_F) + \widehat{e}_F \\ e_C \end{pmatrix} \text{ where } S_{FF} = I_{FF} - Q_{FF}^{-1} A_{FF} \, . \tag{3.46}$$

For any given $e = (e_F, e_C)^T$, we have rapid convergence $S_h^\nu e \to \widehat{e} \; (\nu \to \infty)$ assuming (3.43) to hold. Here, $\hat{e} := (\hat{e}_F, e_C)^T$ with $\hat{e}_F := -A_{FF}^{-1} A_{FC} e_C$ denotes the solution of (3.44).

Remark 3.12 F-relaxation does not satisfy the smoothing property. The last equation shows that $Se = e$ for all $e \in \mathcal{E} := \{e \mid e_F = -A_{FF}^{-1} A_{FC} e_C\}$. Hence, (3.9) cannot hold. ▲

We define interpolation by applying μ F-relaxation steps to solve the F-equations (3.44) approximately. In contrast to F-smoothing, we use Jacobi relaxation in order to keep the resulting operator as local as possible. The resulting interpolation process is thus called **Jacobi interpolation**. To be more specific, given any e_C, we iteratively define a sequence of operators,

$$I_{FC}^{(\mu)} = P_{FF} I_{FC}^{(\mu-1)} - D_{FF}^{-1} A_{FC} \text{ with } P_{FF} = I_{FF} - D_{FF}^{-1} A_{FF} \tag{3.47}$$

starting with some reasonable first guess interpolation operator $I_{FC}^{(0)}$. Because of (3.43), we have rapid convergence $(I_H^h)^{(\mu)} e_C \to \widehat{e} \; (\mu \to \infty)$ at a rate which depends only on δ.

Remark: In contrast to F-smoothing, there is no "natural" first guess $I_{FC}^{(0)}$ available. As will be shown in the theorem below, the choice of $I_{FC}^{(0)}$ is crucial for uniform two-level convergence. ▲

The following theorem states a condition, (3.48), on the first guess interpolation $I_{FC}^{(0)}$ which is sufficient to imply two-level convergence of the AMG approach resulting from combining F-smoothing and Jacobi interpolation.

Theorem 3.8 [87] *Let $A > 0$ and assume the C/F-splitting to be such that A_{FF} is strongly diagonally dominant* (3.43) *with fixed $\delta > 0$. Let smoothing be performed by $\nu \geq 1$ F-relaxation steps* (3.45)*. Finally, let the interpolation be defined by $I_{FC} := I_{FC}^{(\mu)}$* (3.47) *with some $\mu \geq 0$ and assume that the first guess interpolation, $I_{FC}^{(0)}$, satisfies*

$$||(\widehat{I}_{FC} - I_{FC}^{(0)})e_C||_{1,F} \leq \tau ||e||_1 \tag{3.48}$$

for all e with some $\tau \geq 0$ being independent of e and with $\widehat{I}_{FC} := -A_{FF}^{-1}A_{FC}$. Then the following estimate holds:

$$||KS^{\nu}e||_1 \leq (||S_{FF}||_{1,F}^{\nu} + \tau ||P_{FF}||_{1,F}^{\mu})||e||_1 \,. \tag{3.49}$$

Clearly, the norms of S_{FF} and P_{FF} in (3.49) are less than 1 and depend only on δ. In particular, the larger δ, the faster the convergence. Consequently, the previous theorem shows that we can enforce arbitrarily fast two-level convergence by selecting ν and μ accordingly. Moreover, the convergence is *uniform* for $A \in \mathcal{A}$ if we can construct the first guess interpolation, $I_{FC}^{(0)}$, such that (3.48) is uniformly satisfied for all such A. Lemma 3.4 below shows that this can be achieved for the same classes of matrices for which the related τ-condition (3.21) can be satisfied uniformly (see Section 3.2.3).

Lemma 3.4 [87] *The conditions* (3.48) *and* (3.21) *are essentially equivalent. More precisely, consider the two estimates*

$$\text{(a)} \quad ||e_F - I_{FC}e_C||_{0,F}^2 \leq \tau_1 ||e||_1^2 \,, \quad \text{(b)} \quad ||(\widehat{I}_{FC} - I_{FC})e_C||_{1,F}^2 \leq \tau_2 ||e||_1^2 \,. \tag{3.50}$$

If (a) holds for all e and if $\eta \geq \rho(D^{-1}A)$, then (b) holds for all e with $\tau_2 = \eta\tau_1$. If (b) holds for all e and if A_{FF} is strongly diagonally dominant (3.43)*, then (a) holds for all e with $\tau_1 = (1 + \sqrt{\tau_2})^2/\delta$.*

Of course, not each choice for the first guess interpolation $I_{FC}^{(0)}$ can work. For $I_{FC}^{(0)} = 0$, for instance, we cannot expect (3.48) to hold (see Remark 5.6 in [87]). Generally, $I_{FC}^{(0)}$ has to be chosen such that the corresponding Galerkin operator is spectrally equivalent to the Schur complement, $A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}$, w.r.t. all matrices in the class under consideration. For more details see [87].

However, the requirement of strong diagonal dominance (3.43) and the condition (3.48) on the first guess $I_{FC}^{(0)}$ can easily be satisfied by constructing the C/F-splitting and the interpolation according to Theorem 3.7. In particular, $I_{FC}^{(0)}$ is then chosen to be the direct interpolation (3.36). Therefore, not only the conditions on interpolation are essentially equivalent, as stated in Lemma 3.4 above, but the interpolation can be defined on the same basis.

Remark 3.13 Remarks regarding the practical employment of F-smoothing and Jacobi relaxation can be found in Sections 3.2.5 and 4.3.1.5. Various numerical experiments employing F-smoothing and Jacobi interpolation can be found in [48, 87]. ▲

3.2.4.2 Convergence Using Full Relaxation

The analyses of post- and pre-smoothing, respectively, lead to similar results. However, the approach discussed here is not really in the spirit of standard multigrid since smoothing in the usual sense is not exploited. The role of F-smoothing is merely to force $S^\nu e \approx \widehat{e}$ rather than to smooth the error of the full system. This, together with Jacobi interpolation is a "brute-force" approach to make $||S^\nu e - I_H^h e_C||_1$ small for all $e = (e_F, e_C)^T$.

> Practice has shown, however, that the use of *full* relaxation steps instead of mere F-relaxation usually leads to more efficient AMG approaches in terms of computational work. Moreover, we want to note here that Gauss-Seidel in CF-ordering (related to red-black Gauss-Seidel relaxation in geometric multigrid) has turned out to be a very efficient smoother in practice. In particular, for positive definite problems, it performs usually more efficient than Gauss-Seidel without a specific ordering.

For general smoothing processes based on full relaxation, the results of Theorem 3.8 do not carry over. Simply by ignoring the C-part of the relaxation, they do carry over in a trivial way to Gauss-Seidel relaxation in CF-ordering: estimate (3.49) with $\nu = 1$ is obtained. This, however, cannot explain the better performance observed for full smoothing.

Heuristically, the better performance of full instead of mere F-smoothing can be explained as follows. In case of full smoothing - assuming S to satisfy the smoothing property - rather cheap interpolation schemes, based on simple assumptions as (3.25), are usually sufficient to approximate algebraically smooth error. In case of mere F-smoothing, however, such "basic" interpolation schemes have to be "improved" by Jacobi interpolation in order to treat all error components not affected[14] by F-smoothing. Note that one step of Jacobi interpolation is more expensive than replacing mere F- by full smoothing.

3.2.5 Discussion

Section 3.2 was concerned with the basic principles of AMG for scalar applications and the two-level convergence theory. These principles as well as the convergence theory will be extended to unknown- and point-based AMG in the following sections. We will see that the essential aspects will carry over to our whole AMG methodology. In this section, we make some remarks on limits of the two-level convergence theory and on practical means to improve VAMG for "tough" problems.

The AMG theory presented here applies only to symmetric positive definite matrices. For certain "ideal" subclasses of $\mathcal{A}_{\text{spd}}$, the most prominent example being $\mathcal{A}_{\text{St}} \cap \mathcal{A}_{\text{wdd}}$ [15], but also subclasses of $\mathcal{A}_{\text{ess}}$ [16] and some perturbations thereof, *uniform* two-level convergence can be proved. However, the upper bounds are often far too pessimistic as has been seen in Remark 3.4. Other limits of the theory, for instance, w.r.t. multi-level convergence, are discussed in [87].

Of practical importance are two other points. Firstly, even if not strictly provable, we can expect uniform two-level convergence for certain larger matrix classes, being not too far

[14] Recall, in particular, that an error $e \in \mathcal{E}$ is not reduced at all by F-smoothing (see Remark 3.12).

[15] weakly diagonally dominant Stieltjes matrices.

[16] namely matrices in $\mathcal{A}_{\text{ess}}$ (symmetric essentially positive type matrices) with $s_i \geq 0$ for all i.

off from the ideal case. In practice, VAMG's performance turns out to be fairly insensitive to deviations from the ideal case, and V-cycle convergence is, to a large extent, independent of the size of the problem. Secondly, two-level convergence investigations already reveal (at least some) important guidelines in obtaining efficient multi-level approaches. Among those guidelines are

- Find a smoother which fulfills the smoothing property uniformly for the matrix class under consideration with a large σ (see Section 3.2.1.1).
- Find a reasonable compromise for τ, in particular, by coarsening in the direction of smoothness (see Remarks 3.7 and 3.8).
- Handle large positive off-diagonal entries carefully (see Sections 3.2.3.3 and 3.2.3.4).
- Consider indirect interpolation (see Section 3.2.3.5).

We will come back to these topics when discussing the practical realization of C/F-splitting and interpolation schemes (see Sections 4.2 and 4.3).

A "classical" means to improve convergence which has not been mentioned so far is to replace V-cycling by F- or W-cycling. However, in contrast to their importance for GMG, in an AMG approach F- or W-cycles do often not pay: they improve convergence rates but are quite expensive compared with a V-cycle. They reach at best the two-level convergence rate and are therefore not able to cure possible problems with the accuracy of the coarse-level correction. Hence, they provide only a possibility of second choice.

A fitting of interpolation based on some "test vector(s)", as has already been mentioned in [71], is another possibility to improve convergence. However, such a "sophisticated" technique tends to be computationally quite expensive. We come back to this topic in Section 3.3.3.2.

Of more practical importance for "tough" problems are the following means:

- the usage of AMG as a preconditioner, which is one of the most important ways (if not even the most important one) known today to increase AMG's robustness and efficiency as already indicated in Section 2.1.2. This will be demonstrated, for instance, in Sections 4.4 and 4.6, and in Chapter 5.
- ILU(T-type) smoothing which can often fulfill the demand of stronger smoothing for "tough" problems (cf. also investigations in [59, 110] of ILU smoothing for GMG).
- the employment of F-smoothing and Jacobi relaxation, as discussed in Section 3.2.4. These provide purely algebraic means to improve convergence. They should be used *in addition* to full smoothing and the (indirect) interpolation schemes discussed in 3.2.3, respectively. To be more specific, for many "tough" matrices arising in practical applications, a good compromise to obtain an efficient AMG approach is to employ full smoothing in CF-ordering, maybe with one or more additional F-smoothing steps, to use one of the (indirect) interpolation schemes of Section 3.2.3, and to improve interpolation - if necessary - by one or more Jacobi relaxation steps.

For example, the applications in Chapter 5 profit from all these means (even used in combination for drift-diffusion systems) to a large extent.

3.2.6 Complement: Towards Even More Robust *and* Efficient Multilevel Preconditioners

For many years, multigrid methods, general-purpose Krylov subspace methods, one-level iterative preconditioners, and direct methods have been developed rather independently from each other. Particularly in the last few years, research groups - particularly practitioners - have incorporated ideas from other areas into their world or even started to combine different areas. Today, the development of robust and efficient preconditioners is in the focus of research. The overall goal is to develop preconditioners which combine the use of multilevel ideas for (nearly) optimal efficiency with the use of incomplete factorizations or approximate inverses for extreme robustness also for difficult applications. Promising developments include

(a) a transfer of techniques from sparse direct solvers, such as reorderings, scalings, and pivoting, to ILU-type preconditioners,
(b) the development of new one-level preconditioners based on sparse approximate inverses (for instance, SPAI by Grote and Huckle, AINV by Benzi and Tuma),
(c) ILU and SPAI as smoothers for AMG methods,
(d) the usage of multigrid methods, in particular AMG, as preconditioners for Krylov subspace methods,
(e) the incorporation of reduction techniques into algebraic multilevel methods and, in particular, the development of multilevel ILU methods,
(f) parallelizations of the methods [17] .

Due to the vast amount of literature, we do not try to give a complete survey, but refer the interested reader to the survey by Saad and van der Vorst [76] and Benzi's recent survey [3] and the references given therein. Points (c) and (d) are also discussed in this thesis.
Especially the combination of ILU and multilevel techniques draws much attention. Incomplete factorizations and AMG-type approaches are closely related. All these algorithms can be interpreted as approximate Schur complement methods (see paragraph below and articles by Axelsson and Vassilevski (especially AMLI), Dahmen and Elsner cited in [3]). To reach the overall goal mentioned above, a very promising approach to develop "general-purpose" preconditioners is thus based on a combination of algebraic multilevel techniques and incomplete factorizations with an additional incorporation of (some of) the other techniques mentioned above. In the following, we briefly mention important corresponding directions of research and their relationship to AMG.

Schur-Complement Approach Assuming any C/F-splitting given, remember (2.21), a convenient form for $Av = b$ for theoretical investigations:

$$Av = \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} v_F \\ v_C \end{pmatrix} = \begin{pmatrix} b_F \\ b_C \end{pmatrix} = b \ . \tag{3.51}$$

[17] At least briefly, we want to mention the very interesting and popular FETI methods, a family of (nearly) scalable algorithms, developed by Farhad et al. for solving huge problems in structural mechanics and other applications of finite element analysis on massively parallel computers by a domain decomposition approach. [46, 45] discusses particular interesting modern developments. The FETI methods make use of knowledge of the underlying application to achieve a nearly optimal complexity and cannot be considered purely algebraic methods. The basic idea, however, has the potential to be adapted to more general problem classes.

Assuming A_{FF}^{-1} to exist, a way to solve $Av = b$ consists in a block-elimination of A_{CF}, followed by solving the resulting equation

$$Cv_C = b_C - A_{CF}A_{FF}^{-1}b_F \tag{3.52}$$

involving the Schur complement $C := A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}$, and finally solving for the F-variables:

$$v_F = A_{FF}^{-1}\left(b_F - A_{FC}v_C\right) \ . \tag{3.53}$$

AMG as an approximate Schur complement approach If we start with a zero first guess $v^{(0)}$ and define, with I_{CC} being the identity,

$$I_H^h := \begin{pmatrix} I_{FC} \\ I_{CC} \end{pmatrix} \ , I_{FC} := -A_{FF}^{-1}A_{FC} \ , I_h^H := (I_{CF}, I_{CC}) \ ,$$

we obtain the Schur complement as a coarse-level matrix (independent of the concrete I_{CF}):

$$\begin{aligned} A_H = I_h^H A_h I_H^h &= (I_{CF}, I_{CC}) \begin{pmatrix} A_{FF}I_{FC} + A_{FC} \\ A_{CF}I_{FC} + A_{CC} \end{pmatrix} \\ &= (I_{CF}, I_{CC}) \begin{pmatrix} -A_{FF}A_{FF}^{-1}A_{FC} + A_{FC} \\ -A_{CF}A_{FF}^{-1}A_{FC} + A_{CC} \end{pmatrix} = C \ . \end{aligned}$$

With $I_{CF} := -A_{CF}A_{FF}^{-1}$ in addition, the same right-hand side as in (3.52) emerges:

$$I_h^H(b - Av^{(0)}) = -A_{CF}A_{FF}^{-1}b_F + b_C \ .$$

Let $v^\star$ denote the exact solution of $Av = b$. Solving (3.52) yields $v_C^\star$ then. With a smoothing operator S defined as

$$S := \begin{pmatrix} 0 & -A_{FF}^{-1}A_{FC} \\ 0 & I_{CC} \end{pmatrix} \ ,$$

(pure F-smoothing!) we obtain in a post-smoothing step the exact solution of $Av = b$:

$$\begin{aligned} v^{(1)} &= S(v^{(0)} + I_H^h v_C^\star) + (I - S)A^{-1}b = SI_H^h v_C^\star + (I - S)v^\star \\ &= \begin{pmatrix} 0 & -A_{FF}^{-1}A_{FC} \\ 0 & I_{CC} \end{pmatrix} \begin{pmatrix} -A_{FF}^{-1}A_{FC} \\ I_{CC} \end{pmatrix} v_C^\star + \begin{pmatrix} I_{FF} & A_{FF}^{-1}A_{FC} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v_F^\star \\ v_C^\star \end{pmatrix} \\ &= \begin{pmatrix} -A_{FF}^{-1}A_{FC}v_C^\star \\ v_C^\star \end{pmatrix} + \begin{pmatrix} v_F^\star + A_{FF}^{-1}A_{FC}v_C^\star \\ 0 \end{pmatrix} = \begin{pmatrix} v_F^\star \\ v_C^\star \end{pmatrix} \ . \end{aligned}$$

Therefore, the resulting two-level AMG method SK (i.e. coarse-level correction followed by one smoothing step) is a direct solver, which can naturally be extended to a multi-level method. [18]

[18] These statements also hold for the method KS, see Section A.2.3 ("Limit case of direct solvers") in [87].

Both the smoother S and interpolation I_{FC} of the direct solver described above are directly related to solving the (homogeneous) F-equation (3.44), $A_{FF}e_F + A_{FC}e_C = 0$, exactly. The "brute-force" AMG methods described in Section 3.2.4 use F-smoothing and interpolation operators which are formed based on approximations of this F-equation. Also the interpolation operator of standard AMG is based on it. Hence, AMG and, in particular, the "brute-force" methods discussed in Section 3.2.4 can be interpreted as approximate Schur complement methods.

Several other known methods also approximate the two-level Schur complement method (3.52)-(3.53) and can also be extended to multi-level methods.

Reduction Methods For instance, choosing a C/F-splitting which results in a diagonal - or at least diagonally dominant - matrix A_{FF} leads to a "reduction method".

A diagonally dominant A_{FF} can be constructed, for instance, by first applying an AMG-type C/F-splitting[19] method to the matrix graph of strong couplings and afterwards, if necessary at all, shifting variables to C (till A_{FF} is as strongly diagonally dominant as desired).

Obviously, for A_{FF} to become diagonal, the corresponding set of F-variables has to be an independent set. This can be achieved, for instance, by applying an AMG-type C/F-splitting method with the following changes: it is applied to the whole matrix graph, not only to the matrix graph of strong couplings as done in AMG, and the roles of C and F are "interchanged afterwards".

An obvious advantage of forcing A_{FF} to be diagonal(ly dominant) is that its inverse (if existing) and, hence, the Schur complement C are easy to compute. A strong disadvantage is, however, that the density of the coarse-level matrices C usually increases exponentially unless a cut-off strategy is applied. Due to cut-off, such a method is not a direct solver anymore.

A well-known method of this type is the method of total reduction (TR; see, in particular, [80, 81]). TR has mainly been developed for solving $Av = b$ with A corresponding to a scalar elliptic PDE on a rectangle, discretized by means of a 5-point or 9-point stencil as, for instance, (2.30) [20]. For such applications, the iterative variant of the method is numerically stable (see [80, 81, 95]), has optimal complexity O(N) and can be implemented very efficiently which makes it a "perfect" iterative solver. The only but strong disadvantage is its small range of applications.

Remark 3.14 (Relationship of standard AMG and TR) Consider any standard 5-point discretization on a rectangular mesh. Then, obviously, A_{FF} becomes diagonal if we select the C/F-splitting such that, for each $i \in F$, all of its neighbors are in C (red-black coarsening) The coarse-level operator of the standard VAMG method (VAMG(std,A), see Chapter 4) on the second level consisting of the black points, say, can be seen to correspond to ("long") 9-point stencils. By straightforward calculations[21], one can see that, applied to $Av = b$ with A corresponding to (2.30), the two-level standard AMG method VAMG(std,A), the two-level direct solver described above and the two-level total reduction method are equivalent.

[19] Readers not familiar with classical AMG should also read Chapter 4 before reading this section.

[20] together with Dirichlet, Neumann or periodic boundary conditions. It can also be applied to certain more general situations w.r.t. equation, dimension, geometry of the domain.

[21] e.g. using stencil calculus described in [80]. Cf. also Example A.2.1 in [94].

Applied to the second-level matrix, TR and the direct AMG solver are still equivalent. A "usual" AMG method, however, differs from them. This is because the strategies of AMG and TR are quite different. As already mentioned, C/F-splittings are created and employed in a different way; TR solves for (subsets of) the original variables on all levels instead of smoothing errors and computing corrections as done in AMG; and TR uses a cut-off to retain sparsity of the coarse-level matrices, whereas AMG aims at keeping the coarse-level matrices sparse *and* at the same time creating good interpolation and coarse-level operators by a proper coarsening process (max-MIS property, see Section 4.2.1). ▲

Remark 3.15 The MGR method combines geometric multigrid and reduction by means of so-called "intermediate grids" (see [67], [111]). ▲

Approximate cyclic reduction methods as the one described in [66] make also use of C/F-splittings to produce easy-to-invert A_{FF}. The resulting matrices A_{FF} are usually not forced to be diagonal. A C/F-splitting is applied to a "reduced graph", i.e. a graph of strong couplings (AMG-type C/F-splitting, optionally with C and F interchanged), and an approximation of the Schur complement based on a sequence of "point-Gauss" elimination steps is used. The method aims at constructing approximate Schur complements C with a similar sparsity structure than that of A.

Multi-Level Incomplete Factorizations The two-level approach (3.52)-(3.53) is also the basis for "algebraic recursive" or "multi-level ILU" methods (for surveys, see [103, 76, 3]). Among them are

- NGILU and MRILU by Botta, Wubs and van der Ploeg (cf. [103]),
- the hierarchical basis multigraph algorithm by Bank and Smith (see [103]),
- multilevel ILU (MLILU) by Bank and Wagner (cf. [103]; for some comparative notes of this and the previous method, see [66]),
- multilevel ILU (ILUM), block ILUM (BILUM), BILUTM, (parallel) algebraic recursive multilevel solvers ((p)ARMS) and ARMS-C by Saad and co-workers (see, for instance, [51, 73] and the references given therein),

and more. They combine incomplete factorization techniques, AMG-type C/F-splittings, permutations or reorderings (for instance, MC64 by Duff and Koster, part of the Harwell Subroutine Library HSL), rescalings, partial pivoting, block complete pivoting, and dropping (cut-off) strategies to obtain robust *and* efficient preconditioners.

It seems clear that none of the mentioned AMG or multilevel ILU methods alone will be the "holy grale" for all applications - and such a "holy grale" is unlikely to show up. It is the clever combination of techniques for the concrete application considered which makes an efficient preconditioner. Research for "hybrid" linear solvers with characteristics from both iterative and direct methods will be ongoing with "AMG-like" multilevel techniques playing an important role.

3.3 Unknown-Based AMG

We now recall a rather popular AMG strategy to solve systems of PDEs, the so-called **unknown-based AMG (UAMG)** strategy which is very similar to the variable-based strategy except that all unknowns are treated separately. Compared with the variable-based strategy, the only additional information required is information about the VU mapping as defined in Section 2.4.1.

To be more specific, let us assume, for ease of description, the variables to be ordered by unknowns, that is, $Av = b$ has the form (2.16) (see Section 2.4.1):

$$\begin{bmatrix} A_{[1,1]} & \cdots & A_{[1,n_u]} \\ \vdots & \ddots & \vdots \\ A_{[n_u,1]} & \cdots & A_{[n_u,n_u]} \end{bmatrix} \begin{bmatrix} v_{[1]} \\ \vdots \\ v_{[n_u]} \end{bmatrix} = \begin{bmatrix} b_{[1]} \\ \vdots \\ b_{[n_u]} \end{bmatrix} . \tag{3.54}$$

The unknown-based strategy applies variable-based methods to the $A_{[n,n]}$ $(n = 1, \ldots, n_u)$ for coarsening and interpolation. Especially due to the separate coarsening this can even lead to a decoupling of the individual discrete PDE systems in extreme cases.

A detailed description of the components smoothing, coarsening, interpolation and coarse-level operators defining each UAMG approach is given in Section 3.3.1. Section 3.3.2 then discusses two-level convergence of UAMG. This discussion will indicate that essential conditions for the unknown-based strategy to work are that each $A_{[n,n]}$ can successfully be treated by variable-based AMG, that smoothing the individual unknowns separately is sufficient to cause the resulting error to be smooth separately for each unknown, and that the unknown cross-couplings are not too "strong". In particular, as a new contribution, a measure for this *strength of unknown cross-couplings* is introduced. Finally, in Section 3.3.3, we indicate the range of applicability and the limits of the unknown-based strategy. In particular, we discuss the conditions and the measure mentioned above for the model problems introduced in 3.1.3 and for linear elasticity problems. We also summarize experiences compiled from the literature on the application of AMG to linear elasticity.

3.3.1 Components

3.3.1.1 Unknown-Wise Smoothing

For UAMG, we usually employ **unknown-wise Gauss-Seidel (UGS)** smoothing, that is, VGS smoothing but with an unknown-wise ordering: first all variables belonging to $\mathcal{U}_1$ are relaxed, then all variables belonging to $\mathcal{U}_2$ and so on. Both VGS and UGS satisfy the smoothing property (3.9) [22] for all $A \in \mathcal{A}_{\text{St}}$ and other important subclasses of $\mathcal{A}_{\text{spd}}$ (see Section 3.2.1.1). In practice, if UGS smoothing is not efficient, often ILU(0) or ILUT smoothing helps.

[22] In fact, (3.9) cannot distinguish different orderings.

3.3.1.2 Unknown-Wise Coarsening and Interpolation

In UAMG, coarsening and interpolation only deal with the $A_{[n,n]}$. To be more specific, coarsening the set of variables corresponding to the n-th unknown is strictly based on the connectivity structure reflected by the submatrix $A_{[n,n]}$. A variable-based coarsening method is applied to each $A_{[n,n]}$. Also interpolation for a particular unknown $\mathcal{U}_n$ is solely based on the corresponding matrix entries of A, that is, only on $A_{[n,n]}$. In particular, interpolation to any variable v_i involves only coarse-level variables corresponding to the same unknown, i.e. if $v_i \in \mathcal{U}_n$, then its set of interpolatory variables P_i is a subset of $\mathcal{U}_n \cap C$. Obviously, the following form emerges for the interpolation operator I_H^h:

$$I_H^h = \begin{bmatrix} I_{H,[1,1]}^h & & 0 \\ & \ddots & \\ 0 & & I_{H,[n_u,n_u]}^h \end{bmatrix} \tag{3.55}$$

where each $I_{H,[n,n]}^h$ represents an interpolation operator constructed for $A_{[n,n]}$ by means of a variable-based method as discussed in Section 3.2.3. Such an interpolation is called **multiple-unknown-interpolation (MU-interpolation)**.

3.3.1.3 The Coarse-Level Matrices

The Galerkin coarse-level matrices are usually computed w.r.t. all unknowns. That means, as for VAMG, we define

$$A_H := (I_H^h)^T A_h I_H^h \ . \tag{3.56}$$

In some cases, however, it might be more feasible in terms of computational efforts to compute even the Galerkin operator unknown-wise, that is

$$A_H := (I_H^h)^T A_{u,h} I_H^h \tag{3.57}$$

where A_u is defined as the block-diagonal matrix consisting of the $A_{[n,n]}$, that is

$$A_u := \begin{bmatrix} A_{[1,1]} & & 0 \\ & \ddots & \\ 0 & & A_{[n_u,n_u]} \end{bmatrix} \ . \tag{3.58}$$

In case of an unknown-based approach, we call type (3.56) **full Galerkin** and type (3.57) **block-Galerkin**.

3.3.2 Two-Level Convergence

In the following, we assume that the full Galerkin coarse-level operator (3.56) is employed. The subsequent investigations of two-level convergence give us direct generalizations of statements that have been developed for VAMG. We discuss essential conditions on the efficiency of UAMG and quantify that the "stronger" the unknown cross-couplings the worse the convergence of an unknown-based approach has to be expected to be.

For $A > 0$, also $A_u > 0$ holds. We then define the **unknown-wise inner energy product**:

$$(v, w)_{u,1} := (A_u v, w)_E \ . \tag{3.59}$$

The **unknown-wise energy norm**, $||\cdot||_{u,1}$, is defined accordingly.

Lemma 3.5 *The following inequality holds for all $A > 0$:*

$$\forall\, e \ : \ (A_u e, e) \leq \rho(A^{-1}A_u)(Ae, e) \ . \tag{3.60}$$

Proof. This follows from (2.59) and (2.51) used to obtain a valid constant c in

$$(A_u e, e) = (A_u A^{-1} Ae, e) \leq c(Ae, e) \ .$$

■

Since in UAMG a variable-based coarsening and interpolation strategy is applied to each $A_{[n,n]}$, it is natural to demand VAMG to work for each $A_{[n,n]}$ and a τ-condition (3.21) to be fulfilled with some $\tau = \tau(A_{[n,n]})$. As has to be expected, the maximum τ_u of these $\tau(A_{[n,n]})$ comes into the upper bound for two-level convergence. We first note that the following ***τ*-condition of MU-interpolation**,

$$\forall\, e \ : \ ||e_F - I_{FC} e_C||^2_{0,F} \leq \tau_u ||e||^2_{u,1} \ , \tag{3.61}$$

holds under the conditions mentioned above. We can prove the following analogs of Theorems 3.3 and 3.4 now:

Lemma 3.6 *Let $A > 0$ and a VU mapping be given. If the C/F-splitting and interpolation I_{FC} are such that the τ-condition* (3.61) *of MU-interpolation is fulfilled with τ_u being independent of e, then* (3.20) *is satisfied with $\tau = \tau_u \rho\big(A^{-1}A_u\big)$.*

Proof. Due to Lemma 3.5, we obtain $||e||^2_{u,1} \leq \rho\big(A^{-1}A_u\big)||e||^2_1$. The remainder follows from Theorem 3.4. ■

Lemma 3.7 *Let $A > 0$ and a VU mapping be given. Let S satisfy the smoothing property* (3.9)*. Furthermore, assume the C/F-splitting and interpolation to be such that the condition* (3.61) *is fulfilled with some τ_u being independent of e. Then $||SK||_1 \leq \sqrt{1 - \sigma/\tau}$ is satisfied with $\tau = \tau_u \rho\big(A^{-1}A_u\big) \geq \sigma$.*

Proof. Combine Theorems 3.6 and 3.3. ■

The crucial points here are the fulfillment of the τ-condition of MU-interpolation and the smoothing property (3.9). Since we employ variable-based methods for coarsening and interpolation, the smoother S has to provide error which is algebraically smooth separately for the different unknowns. In principle, we thus demand an "unknown-smoothing property" to hold and define: A smoothing operator S is said to satify the **unknown-smoothing property** w.r.t. a matrix $A > 0$ and a VU mapping given if, for all e, the following inequality holds with a $\sigma_u > 0$ being independent of e:

$$||Se||^2_{u,1} \leq ||e||^2_{u,1} - \sigma_u ||e||^2_{u,2} \quad (\sigma_u > 0) \ . \tag{3.62}$$

The following theorem, a straightforward analog of Theorem 3.3 combined with Theorem 3.4, can be proved now. As for Lemma 3.6, we make use of Lemma 3.5 in the proof, here even on several places. Lemma 3.5 provides a simple means to compare the energy norm based on A with that based on A_u. The upper bound of two-level convergence obtained below is thus not *optimal* (and might even be far away from that). However, it at least indicates which terms influence UAMG's performance.

Theorem 3.9 *Let $A > 0$ and let S satisfy the unknown-smoothing property* (3.62)*. Furthermore, assume the C/F-splitting and interpolation to be such that the τ-condition* (3.61) *of MU-interpolation is fulfilled with τ_u being independent of e. Then*

$$||SK||_{u,1} \leq \sqrt{\rho\big(A^{-1}A_u\big)\rho\big(A_u^{-1}A\big)}\sqrt{1-\sigma_u/\widetilde{\tau}} \tag{3.63}$$

with $\widetilde{\tau} = \tau_u\rho\big(A^{-1}A_u\big)^2\rho\big((A_u^{-1}A)^2\big)$.

Proof. Due to Corollary 3.6, the τ-condition of MU-interpolation implies

$$\forall\, e \,:\, ||Ke||_1^2 \leq \tau_u\rho\big(A^{-1}A_u\big)||Ke||_2^2 \tag{3.64}$$

with some $\tau_u > 0$ being independent of e. Due to (2.59) and (2.51), we obtain

$$\begin{aligned} ||Ke||_2^2 \;&=\; (D^{-1}AKe, AKe)_E \;=\; (AD^{-1}AKe, Ke)_E \\ &\leq\; \rho\big(A_u^{-1}DA_u^{-1}AD^{-1}A\big)(A_uD^{-1}A_uKe, Ke)_E \;. \\ &=\; \rho\big((A_u^{-1}A)^2\big)||Ke||_{u,2}^2 \end{aligned}$$

With Lemma 3.6 and (3.64), we can then estimate

$$\begin{aligned} \rho\big(A^{-1}A_u\big)^{-1}||Ke||_{u,1}^2 \;&\leq\; ||Ke||_1^2 \;\leq\; \tau_u\rho\big(A^{-1}A_u\big)||Ke||_2^2 \\ &\leq\; \tau_u\rho\big(A^{-1}A_u\big)\rho\big((A_u^{-1}A)^2\big)||Ke||_{u,2}^2 \;. \end{aligned}$$

Together with (3.62), we now obtain

$$||SKe||_{u,1}^2 \;\leq\; ||Ke||_{u,1}^2 - \sigma_u||Ke||_{u,2}^2 \;\leq\; \left(1 - \frac{\sigma_u}{\tau_u\rho\big(A^{-1}A_u\big)^2\rho\big((A_u^{-1}A)^2\big)}\right)||Ke||_{u,1}^2 \;.$$

We know from Theorem 3.1 that $||K||_1 = 1$. Again due to (2.59), we can estimate

$$||Ke||_{u,1}^2 \;\leq\; \rho\big(A^{-1}A_u\big)||Ke||_1^2 \;\leq\; \rho\big(A^{-1}A_u\big)||e||_1^2 \;\leq\; \rho\big(A^{-1}A_u\big)\rho\big(A_u^{-1}A\big)||e||_{u,1}^2 \;.$$

In summary, the theorem is proved. ■

Therefore, we can - in principle - measure the strength of unknown cross-couplings by

$$\rho_u \;:=\; \rho(A^{-1}A_u)\,\rho(A_u^{-1}A) \;. \tag{3.65}$$

Remark 3.16 Due to (2.51), $\rho\big(A^{-1}A_u\big) = \rho\big(A_uA^{-1}\big)$ and $\rho\big(A_u^{-1}A\big) = \rho\big(AA_u^{-1}\big)$ hold. ▲

Remark 3.17 The above upper bound for two-level convergence can be larger than one. Due to Theorem 3.1, unknown-based AMG will nevertheless not diverge for $A > 0$ if the smoother satisfies $||S||_1 \leq 1$ (see also Lemma 3.7). ▲

3.3.3 Discussion

The investigations made above indicate which conditions are necessary for an unknown-based approach to be efficient. First, smoothing should cause the error to become algebraically smooth separately for each unknown. This will be the case if the unknown cross-couplings are not too strong in the sense of a small ρ_u. Second, a similar statement can be made for the variable-based coarsening and interpolation strategies applied to the $A_{[n,n]}$. Not only should these strategies be appropriate for the $A_{[n,n]}$ but the resulting operator K also for the whole matrix A. Again, this will be the case if the unknown cross-couplings are weak in the sense of a small ρ_u. In summary, ρ_u provides a measure of the **strength of unknown cross-couplings** in the following sense: the larger this value, the "stronger", that is the worse, has the influence of unknown cross-couplings on the convergence of an unknown-based approach to be expected.

We are going to investigate the strength of unknown cross-couplings and the applicability of UAMG for our three model classes now. Afterwards, in Section 3.3.3.2, we will discuss UAMG for matrices arising in linear elasticity. Also other AMG approaches for elasticity problems will be reviewed.

3.3.3.1 Theoretical and Numerical Investigations of UAMG for the Model Problems

Investigations of the Smoothing Property The smoothing property of variable-wise Gauss-Seidel (VGS) and, hence, also unknown-based Gauss-Seidel (UGS) is now investigated for the three model classes defined in Section 3.1.3. We start with the AVL models.

For matrices of the form L_S defined by (3.1), we can compute the parameter σ in (3.9) by means of Theorem 3.2 - as long as $|c| < \sqrt{ab}$. For $w = 1$ we obtain

$$\gamma_+ = \max\left\{ \frac{(1+\epsilon)a + (3+3\epsilon)|c|}{(2+2\epsilon)a}, \frac{(1+\epsilon)b + (1+\epsilon)|c|}{(2+2\epsilon)b} \right\} ,$$
$$\gamma_- = \max\left\{ \frac{(1+\epsilon)a + (1+\epsilon)|c|}{(2+2\epsilon)a}, \frac{(1+\epsilon)b + (3+3\epsilon)|c|}{(2+2\epsilon)b} \right\} .$$

Obviously, this evaluates to

$$\gamma_+ = \max\left\{ \frac{a+3|c|}{2a}, \frac{b+|c|}{2b} \right\} = \frac{1}{2} + \frac{1}{2}\max\left\{ 3\frac{|c|}{a}, \frac{|c|}{b} \right\} ,$$
$$\gamma_- = \max\left\{ \frac{a+|c|}{2a}, \frac{b+3|c|}{2b} \right\} = \frac{1}{2} + \frac{1}{2}\max\left\{ \frac{|c|}{a}, 3\frac{|c|}{b} \right\} ,$$
$$\sigma = \frac{1}{(1+\gamma_+)(1+\gamma_-)} = \frac{4}{(3+|c|\max\{\frac{3}{a},\frac{1}{b}\})(3+|c|\max\{\frac{1}{a},\frac{3}{b}\})}$$

independent of ϵ. We can follow now that (for $w = 1$) $\sigma \in]0, 4/9]$. For the case $A \in \mathcal{A}_{\text{spd}}$, i.e. $|c| < \sqrt{ab}$, we obtain $\sigma \in]1/9, 4/9]$. For instance, for $a = b = 2|c|$, we obtain $\sigma = 16/81$ as a value for the smoothing property of VGS for matrices L_S. As expected, these results for σ indicate that large unknown cross-couplings, which correspond to a large $|c|$, destroy the good smoothing property of VGS obtained for the decoupled case ($\sigma = 4/9$).

For matrices of the form L_X defined by (3.3), we can compute the parameter σ analogously. For $w = 1$ we obtain

$$\gamma_+ = \max\left\{ \frac{(1+\epsilon)a + 6|c|}{2(1+\epsilon)a}, \frac{(1+\epsilon)b + 2|c|}{2(1+\epsilon)b} \right\} .$$

If we replace a with b, we obtain γ_-. In contrast to the AVLS model, σ depends on ϵ here. For instance, for $a = b = 2|c|$, we arrive at $\sigma = \frac{4(1+\epsilon)^2}{9(2+\epsilon)^2}$, which results in $\sigma = 1/9$ for $\epsilon = 0$ and (as for L_S) $\sigma = 16/81$ for $\epsilon = 1$. The obtained values for σ are the same or (a bit) worse than those obtained for L_S. Numerical results for different L_X matrices will be given in Section 3.4.1.2.

Also for the reaction-diffusion-like class defined in Section 3.1.3.2, Theorem 3.2 helps in determining σ. Here, $\max_k |d_{kk}| < 4\,(1 - \cos(\pi h))$ ensures $A > 0$ and, hence, Theorem 3.2 to be applicable. For $w = 1$ we obtain

$$\gamma_+ = \gamma_- = \max_k \left(\frac{|d_{kk}| + 2}{4} \right) \geq \frac{1}{2} \quad \text{and thus} \quad \sigma = \left(\frac{16}{(\max_k |d_{kk}| + 6)^2} \right) \leq \frac{4}{9} .$$

For the case $A > 0$, $\max_k |d_{kk}| < 4\,(1 - \cos(\pi h))$, we obtain

$$\sigma \in \left] \frac{16}{(10 - 4\,\cos(\pi h))^2}, \frac{4}{9} \right] .$$

Of course, since $|c| = h^2|f|$ is very small for small h, the lower bound is very close to the upper bound $4/9$. Note, however, that the unknown cross-couplings can in practice be much larger than diagonal elements. The same conclusion as for the AVL case can thus be drawn. With increasing magnitude of $\max_k |d_{kk}|$ we can expect VGS and UGS to be less efficient smoothers. Numerical results proving this will be shown in Section 3.4.1.2.

The matrices belonging to the drift-diffusion-like class (Section 3.1.3.3) are not symmetric so that Theorem 3.2 cannot be employed. Numerically, it can be seen that VGS strongly diverges. This has to be expected since the matrices are far from being diagonally dominant: a whole off-diagonal block, $A_{[1,2]}$, dominates the diagonal blocks $A_{[n,n]}$ (see Example 3.5). Numerical results will be shown in Section 3.4.1.2.

Investigations of ρ_u As pointed out above, the factors $\rho\big(AA_u^{-1}\big)$ and $\rho\big(A_uA^{-1}\big)$ can be expected to have a decisive impact on UAMG's convergence. For the AVLS model (3.1) with $a = b$ and $|c| < a$, it is easy to see that

$$AA_u^{-1} = \begin{bmatrix} I & \frac{c}{a}I \\ \frac{c}{a}I & I \end{bmatrix} ,$$

and thus $\rho(AA_u^{-1}) = 1 + |c|/a$. Additionally, AA_u^{-1} is symmetric positive definite, and we obtain

$$\rho(A_uA^{-1}) = (\lambda_{\min}(AA_u^{-1}))^{-1} = (1 - |c|/a)^{-1} .$$

Obviously, for $|c|$ approaching a (from below to retain $A > 0$), A approaches singularity and $\rho_u = \frac{1+|c|/a}{1-|c|/a}$ tends to infinity.

For the AVLX model (3.3), the matrices $A_u A^{-1}$ and $A A_u^{-1}$ are (except of trivial cases) not symmetric and have complex eigenvalues. For $a = b = 1$, $c = 10$, $\epsilon = 1e-3$ we obtain[23]

$$|\lambda|_{\max}(A_u A^{-1}) \approx \text{5.3e-2} \ , \quad |\lambda|_{\max}(A A_u^{-1}) \approx \text{1.7e+2} \ .$$

For $a = b = 10$, $c = 1$, $\epsilon = 1e - 3$ we obtain

$$|\lambda|_{\max}(A_u A^{-1}) \approx \text{2.8e+2} \ , \quad |\lambda|_{\max}(A A_u^{-1}) \approx \text{2.7e+0} \ .$$

Although with the above parameter settings the corresponding matrices A are not > 0, the ρ-values indeed indicate that UAMG is not appropriate here.

Numerical results for the AVL models will be shown in Section 4.6. They will demonstrate that for UAMG applied to AVL models, it is not so important whether the anisotropies for the different unknowns are in the same or in different directions. However, it is important how large the unknown cross-couplings are, as indicated by the above considerations of ρ_u. For the ideal situation of nearly decoupled Laplacians ($\rho_u \approx 1$), Remark 5.2 will give an example.

For the RD model with $n_z = 1$, $c = 1e + 3$, we obtain[24]

$$\{\lambda(A_u A^{-1})\} \approx \{ \text{-3.3e-3, 3.3e-3, 1} \} \ , \quad \{\lambda(A A_u^{-1})\} \approx \{ \text{-3.0e+2, 3.0e+2, 1} \}$$

and, hence, $\rho_u \approx$ 3.0e+2. For the same RD model but with $c = 1e + 9$, we obtain

$$\{\lambda(A_u A^{-1})\} \approx \{ \text{-3.3e-9, 3.3e-9, 1} \} \ , \quad \{\lambda(A A_u^{-1})\} \approx \{ \text{-3.0e+8, 3.0e+8, 1} \}$$

and, hence, $\rho_u \approx$ 3.0e+8. Note that, for increasing n_z, $|\lambda|_{\max}(A A_u^{-1})$ and thus ρ_u grow substantially.

The situation is comparable with the AVL models. With increasing c, VAMG's efficiency decreases because diagonal dominance is increasingly strongly violated for the $2n_z$ rows containing a large positive off-diagonal entry. Due to the same reasons, also UAMG's convergence will break down [25]. Even worse for the DD models with moderate (λ, c). Neither VAMG nor UAMG works here.

Concrete numerical results for both RD and DD models will be shown in Section 3.4.1.2 (investigations of smoothing properties) and Section 4.6 (performance of different VAMG, UAMG and PAMG approaches).

3.3.3.2 Linear Elasticity

Linear elasticity or plasticity problems, as for instance material stress calculations, are of great practical importance. For typical FE discretizations and boundary conditions, the arising matrices are symmetric positive definite. However, the numerical solution of these matrices by means of one-level or hierarchical iterative solvers faces several problems, among

[23] Computations have been performed for the case $h = 1/32$, employing LAPACK's [1] direct eigensolver.
[24] Computations have been performed for the case $h = 1/32$, employing LAPACK's [1] direct eigensolver.
[25] regardless whether UGS or the "stronger" ILU(0) is used as smoother.

the severe ones being locking effects (due to unproper discretizations), higher order finite elements, shell elements, anisotropies and large positive off-diagonal matrix entries, multi- or single-point constraints and rigid body modes. Consequences can be very ill-conditioned, nearly singular matrices. In addition, higher order elements or shell elements pose severe problems for (A)MG approaches.

We concentrate here on AMG approaches for matrices emerging from second-order standard FE discretizations. After describing the PDE system to be solved, we briefly go into the problems for AMG - anisotropies, large positive off-diagonals and rigid bodes - and review the different approaches in the literature. In particular, we explain under which conditions UAMG can work and prepare subsequent discussions in Section 5.2.1 where AMG for applications in semiconductor stress simulation is investigated.

Lamé Equations The classical Lamé equations modeling 3D linear elasticity problems (see, for instance, [6] for more detailed information) formulated with the Lamé coefficients λ, μ read as follows

$$\begin{bmatrix} -(2\mu+\lambda)\,u_{1,xx} - \mu u_{1,yy} - \mu u_{1,zz} - (\mu+\lambda)\,u_{2,xy} - (\mu+\lambda)\,u_{3,xz} \\ -\mu u_{2,xx} - (2\mu+\lambda)\,u_{2,yy} - \mu u_{2,zz} - (\mu+\lambda)\,u_{1,xy} - (\mu+\lambda)\,u_{3,yz} \\ -\mu u_{3,xx} - \mu u_{3,yy} - (2\mu+\lambda)\,u_{3,zz} - (\mu+\lambda)\,u_{1,xz} - (\mu+\lambda)\,u_{2,yz} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} . \quad (3.66)$$

The scalar functions u_1, u_2, u_3 denote the displacements in x-, y- and z-direction, respectively. Typical boundary conditions are Dirichlet conditions for some parts of the boundary and $\frac{\partial T_\sigma(u)}{\partial \nu} = g$ for the remainder, where T_σ denotes the stress tensor, $\frac{\partial}{\partial \nu}$ the derivative normal to the boundary, f the volume force and g the surface force (cf. [6]).

If we assume that external forces depend only on x and y with vanishing z-component, and that there are no z-components of the strain[26], the plane-strain problem emerges[27]:

$$\begin{bmatrix} -(2\mu+\lambda)\,u_{1,xx} - \mu u_{1,yy} - (\mu+\lambda)\,u_{2,xy} \\ -\mu u_{2,xx} - (2\mu+\lambda)\,u_{2,yy} - (\mu+\lambda)\,u_{1,xy} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} . \quad (3.67)$$

The problem class considered in Chapter 5.2.1 will be of the plane-strain type.

Generally, the Lamé coefficients are related to Young's modulus E of elasticity and the Poisson ratio ν by

$$\nu = \frac{\lambda}{2(\lambda+\mu)} , \quad E = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu} , \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} , \quad \mu = \frac{E}{2(1+\nu)} . \quad (3.68)$$

The Lamé system is elliptic in the sense that each individual equation is elliptic. Additionally, each individual equation exhibits anisotropy, in particular for the case of nearly incompressible material, $\lambda \gg \mu$, and the anisotropy for each equation is obviously in a different direction than for the respective other equation(s). For $\lambda \gg \mu$, the problem is very ill-conditioned[28].

[26] i.e. a deformation in z-direction is not allowed.

[27] If we allowed for strain in z-direction, we would arrive at the plane-stress problem.

[28] see discussions of locking effects in [6], for example. Here, a weak formulation of the system with the pressure p as an additional unknown function, resulting in a system with a penalty term, would help the situation. However, (u, p)-systems are not discussed here.

The arising stiffness matrices A inherit the anisotropy and ill-conditioning of the continuous PDE system. Due to the ill-conditioning, classical one-level solvers have severe problems in solving the matrix equations efficiently.

Handling Anisotropies and Large Positive Off-Diagonals Despite the facts that the equations are elliptic and the stiffness matrices A usually symmetric positive definite, AMG approaches might not be efficient. One problem being that even standard bilinear finite elements lead to large positive off-diagonals in the submatrices $A_{[n,n]}$ here (see Section (2.4.4), stencil (2.28) with $\alpha = 1/4$). A modification of coarsening, efficiently curing the problem here, has already been discussed in Section 3.2.3.4. This way, VAMG can handle the $A_{[n,n]}$ correctly also in case of strong anisotropies. The PDE system, however, is too strongly coupled for VAMG.

Calculations of ρ_u have been performed for (3.67), on the unit square, discretized by bilinear finite elements, $h = 1/32$, $\nu \in \{0.20, 0.45\}$, with Dirichlet conditions on two sides of the unit square and homogeneous Neumann boundaries on the remaining two. Calculations have also been performed for three stress matrices arising in semiconductor simulation, see Section 5.2.1.3. Values for ρ_u between 3 and 10 have been obtained. These "small" ρ_u values, indicating a "moderate" unknown cross-coupling, correspond to the experience that UAMG works quite efficiently for stress matrices - at least if they are not nearly singular. This topic will be discussed next. Before, we note that, due to a more or less strong anisotropy in different directions for the different unknowns, unknown-based AMG should be more appropriate than a point-based strategy.

Rigid Body Modes However, the application of any AMG approach to the stiffness matrices A is challenging mainly due to the so-called *rigid body modes* forming the non-trivial kernel of an elasticity operator without essential boundary conditions. In case of 2D elasticity, this kernel consists of three rigid body modes, namely two translations and one rotation; in case of 3D elasticity, it consists of six rigid bodes, namely three translations and three rotations. Since the "quality" of an AMG interpolation depends on how well eigenfunctions belonging to the smallest eigenvalues are reproduced, these eigenfunctions, which are equal to the rigid body modes in the singular case, have to be interpolated as exactly as possible if A is singular or nearly singular. The latter is the case if only on a small part of the domain Dirichlet conditions are imposed.

In the singular case (i.e. A has only zero row sums), the translations, being unknown-wise constant functions, pose no problems for AMG since their exact interpolation can simply be achieved by requiring the row sums of the interpolation matrix I_{FC} to be one unknown-wise. Our interpolation schemes do fulfill this condition for zero-row-sum matrices. However, the rotations are linear functions, and their exact interpolation cannot be guaranteed by an AMG approach without special input and/or modifications. Moreover, in the nearly singular case, the rigid body modes are only approximations of the eigenfunctions corresponding to the smallest eigenvalues. The smaller the smallest eigenvalue of A, the worse the performance of unmodified AMG can be expected to be. The magnitude of the smallest eigenvalue of A is strongly related to the rate r_{Dir} which is defined as the number of Dirichlet variables divided by the total number of boundary variables. r_{Dir} is thus an important factor determining the performance of UAMG for the stiffness matrices A in case of linear elasticity.

AMG Approaches for Linear Elasticity That the unknown-based approach can successfully be applied to certain linear elasticity problems has already been demonstrated in [69, 71]. As discussed above, the convergence of this simple and cheap approach deteriorates if the problem is (nearly) singular since the eigenfunctions corresponding to the rotations in the singular case cannot be exactly reproduced by the interpolation schemes employed. The same is true for aggregative approaches.

In principle, if a set of independent eigenvectors corresponding to the smallest eigenvalues of A is known and is passed to an AMG solver as a set of "test vectors", interpolation can be forced to reproduce them correctly. Whereas in general situations, of course, explicitly computing these eigenvectors is too expensive, the rigid body modes of linear elasticity (being good approximations of these eigenfunctions in the nearly singular case) can be determined via the coordinates of the grid nodes. For classical AMG, a procedure to fit interpolation to a set of submitted vectors (such as rigid body modes) has been outlined already in [71]. Nevertheless, the development of robust and efficient approaches, based on classical AMG, is still subject to further research. For AMG based on smoothed aggregation, a proper incorporation of submitted eigenvectors has been discussed in [99, 53], for instance. Since these approaches are point-based, we will review them and other point-based approaches for linear elasticity in Section 3.4 in Remark 3.31.

To interpolate eigenvectors belonging to small eigenvalues particularly well is also the expressed goal of AMGe. Instead of using a set of test vectors, local approximations to the smallest eigenvalues are constructed via the element-stiffness matrices. As has been demonstrated in [12, 16] for a thin body elasticity model, AMGe indeed shows better convergence properties than unknown-based AMG [71]. However, AMGe's efficiency in terms of computational cost has not been discussed so far.

3.4 A General Framework for Point-Based AMG

In this section, we outline a general and flexible framework for constructing AMG approaches to solve various types of PDE systems. In contrast to the previous approaches, that is variable- and unknown-based ones, all of the new ones operate on the level of points rather than variables. This is to be understood as follows. Whereas in an unknown-based approach each unknown is associated with its own level hierarchy and own transfer operators, it often makes more sense to coarsen the unknowns *simultaneously* if they live on the same grid. This leads to what we call point-based approaches: we talk about a **point-based AMG approach (PAMG)** if coarsening takes place on the level of points (rather than variables as before) and all unknowns are defined on the same hierarchy. Such a coarsening is called **point-coarsening**.

Since we primarily have the solution of PDEs in mind, we think of points as corresponding to real physical grid nodes in space, each one associated with a block of variables, $v_{(k)}$. However, we want to point out that, from AMG's point of view, it is not important whether points really correspond to physical grid nodes. Instead, one may think of the nodes of a graph representing the connectivity structure of A. Regarding a point-based approach, it is only relevant for AMG to know whether there are blocks $v_{(k)}$ of variables (corresponding to different unknowns) which may be treated (at least coarsened, but often also interpolated)

simultaneously. In the context of PAMG approaches, for simplicity, we assume the variables to be ordered point-wise (cf. Section 2.4.1):

$$\begin{bmatrix} A_{(1,1)} & \cdots & A_{(1,n_p)} \\ \vdots & \ddots & \vdots \\ A_{(n_p,1)} & \cdots & A_{(n_p,n_p)} \end{bmatrix} \begin{bmatrix} v_{(1)} \\ \vdots \\ v_{(n_p)} \end{bmatrix} = \begin{bmatrix} b_{(1)} \\ \vdots \\ b_{(n_p)} \end{bmatrix}, \tag{3.69}$$

where $A_{(k,l)}$ represents the couplings of the k-th to the l-th point, that is of $v_{(k)}$ to $v_{(l)}$. The $A_{(k,l)}$ are also called **point-coupling matrices**. In case that all unknowns are defined at all points, all $A_{(k,l)}$ are $(n_u \times n_u)$-matrices. Generally, however, $A_{(k,l)}$ is a $(|\mathcal{P}_k| \times |\mathcal{P}_l|)$-matrix[29] and, hence, not necessarily square.

As VAMG and UAMG, also PAMG follows the scheme outlined in Section 2.3. In addition to the information VAMG and UAMG can exploit, namely A, b, and (for UAMG) the VU mapping, PAMG makes use of the VP mapping, too, in order to define the three main components smoothing, coarsening, and interpolation. A main difference to VAMG and UAMG is the **point-coarsening scheme** which is uniformly obeyed by all our PAMG approaches: To obtain a C/F-splitting on a given level with A being the current level-matrix the following three steps are performed:

1. Define an auxiliary (sparse) $(n_p \times n_p)$-matrix $\boldsymbol{P} = (p_{kl})$, called the **primary matrix**, which reflects the point-couplings of the current level-matrix A *in some reasonable sense*. In particular, the employed primary matrix should reflect the physical connectivity (the general structure as well as the strength of connections) of neighboring variables reasonably well, *simultaneously for all unknowns*.

2. Perform a scalar AMG coarsening process applied to $\boldsymbol{P}$ to obtain a splitting of the set of points into coarse- and fine-level points. The resulting index sets of C- and F-points are denoted by C^p and F^p, respectively, and the splitting is called the $\boldsymbol{C^p/F^p}$**-splitting**.

3. Assign (in principle) the same new coarse level to all unknowns by copying this splitting to the set of variables.

Remark 3.18 The above scheme can be generalized; we could allow more than one primary matrix. In principle, each unknown could have its own primary matrix. Possible realizations are, however, one topic of future research. ▲

Due to the various information PAMG can exploit to solve a discrete PDE system, one can expect that we have much more freedom compared to VAMG and UAMG in defining concrete components. However, as for all AMG approaches, we have to bear in mind that smoothing, coarsening and interpolation are strongly related processes, and that their interplay determines the efficiency of the overall appproach.

In the following sections, we describe a **general framework** to set up concrete point-based approaches. We focalize on concrete ways to construct each of the components of our scheme for relevant applications. Since usually not only one problem but rather a whole class

[29] $|S|$ denotes the number of elements of a set S.

of problems shall be solved efficiently, we especially seek definitions of the PAMG components which can be used "uniformly" for the whole class considered. The considerations of the next sections will reveal that for many problem classes which exhibit a point structure efficient PAMG approaches can be defined by means of our framework. Whereas the components can often be defined uniformly for one problem class, they might vary substantially between the classes.

Outline of the Following Sections Section 3.4.1 investigates point-oriented smoothing, especially the block-variant of Gauss-Seidel and its application to our model problems. This is followed by a discussion of possible coarse-level correction processes. Section 3.4.2 contains a detailed description of possible primary matrices. It also explains their recursive definition in case of a multilevel method and discusses the resulting point-coarsening processes for the types of primary matrices introduced. Section 3.4.3 then explains the possibilities within our framework of how to base the interpolation strategy on a given primary matrix. Three general types of interpolation and, for each type, concrete variants are introduced. In both Sections 3.4.2 and 3.4.3, point-based coarsening and interpolation are discussed especially for the two models problems 3.1.3.2 and 3.1.3.3 for which unknown-based AMG is inefficient or even fails. We will see, in particular, that the applicability of concrete components vary substantially between the models and that two main directions of point-based AMG cristallize out: oriented on norms on one hand, oriented on coordinates on the other. These results will later be strengthened also for practical problems arising in semiconductor simulation: Chapter 5 will reveal that, whereas a strategy based on coordinates is effective for reaction-diffusion problems, a norm-based strategy is effective for drift-diffusion problems. In Section 3.4.4, we develop generalizations of VAMG's two-level convergence theory and prove - under suitable conditions - some corresponding generalized theorems on convergence of PAMG approaches.

3.4.1 Smoothing

Though, at least formally, any relaxation method could be used as a smoother in a point-based approach, a distinctly point-oriented smoothing is often a prerequisite for the success of a point-based approach. The reason being that a smoothing which treats variables belonging to the same point simultaneously is often most appropriate for handling strong unknown cross-couplings and for producing algebraically smooth error which allows for a point-coarsening. This should especially be true for applications where the decisive unknown cross-couplings are located mostly in the $A_{(k,k)}$ on the block-diagonal of A. An obvious example for such a matrix class is given by the RD models (3.4).

A standard block-wise smoother is **block-wise Gauss-Seidel (BGS)**. Its linear smoothing operator S_{BGS} equals

$$S_{BGS} = I - Q_P^{-1} A \tag{3.70}$$

with Q_P being the lower block-triangular part of A including the diagonal blocks:

$$Q_P = (Q_{(k,l)})_{k,l=1,\dots,n_p} \quad \text{with} \quad Q_{(k,l)} = \begin{cases} A_{(k,l)} & \text{if } l \leq k\,, \\ 0 & \text{else.} \end{cases}$$

Note that Q_P is non-singular for arbitrary $A > 0$.

In the next Section, 3.4.1.1, we prove that S_{BGS} satisfies the so-called point-smoothing property. Afterwards, BGS' smoothing properties are examined for our model problems. Numerical results are also presented and discussed for ILU(0) smoothing. Another remark concerning ILU-type smoothing is made at the end of Section 3.4.1.2.

3.4.1.1 Point-Smoothing Property

Generalizations of the variable-based smoothing property (3.9), Theorem 3.2 (see Section 3.2.1.1) and Lemma 3.2 in [87] to the point-based case can be obtained in a straightforward way. We start with the smoothing property and define:

A smoothing operator S is said to satify the **point-smoothing property** w.r.t. a matrix $A > 0$ if for all e

$$||Se||_1^2 \leq ||e||_1^2 - \sigma||e||_{P,2}^2 \quad (\sigma > 0) \tag{3.71}$$

holds with σ being independent of e.

The following lemma gives us a straightforward generalization of Lemma 3.2 [87] to the point case.

Lemma 3.8 *Let $A > 0$ and let the smoothing operator be of the form $S = I - Q_P^{-1}A$ with a nonsingular matrix Q_P. Then the smoothing property* (3.71) *is equivalent to*

$$\sigma Q_P^T D_P^{-1} Q_P \leq Q_P + Q_P^T - A\,.$$

Proof. A straightforward calculation shows that

$$||Se||_1^2 = ||e||_1^2 - ((Q_P + Q_P^T - A)Q_P^{-1}Ae, Q_P^{-1}Ae)_E\,.$$

Therefore, (3.71) is equivalent to

$$\sigma||e||_{P,2}^2 \leq ((Q_P + Q_P^T - A)Q_P^{-1}Ae, Q_P^{-1}Ae)_E$$

which, in turn, is equivalent to

$$\sigma(D_P^{-1}Q_P e, Q_P e)_E^2 \leq ((Q_P + Q_P^T - A)e, e)_E\,. \qquad \blacksquare$$

With this lemma, we can now prove a straightforward generalization of Theorem 3.2 to the point-based case.

Theorem 3.10 *Let $A > 0$ and define with any vector $w = (w_k)_{k=1,\ldots,n_p} > 0$*

$$\gamma_-^p := \max_k \left\{ \frac{1}{w_k} \sum_{l<k} w_l ||A_{(k,k)}^{-1} A_{(k,l)}||_E \right\},$$

$$\gamma_+^p := \max_k \left\{ \frac{1}{w_k} \sum_{l>k} w_l ||A_{(k,k)}^{-1} A_{(k,l)}||_E \right\}.$$

Then Block-Gauss-Seidel relaxation (3.70) *satisfies* (3.71) *with $\sigma = \frac{1}{(1+\gamma_-^p)(1+\gamma_+^p)}$.*

Proof. We proceed analogously to [87]. S_{BGS} satisfies the assumptions of Lemma 3.8, and we have $Q_P + Q_P^T - A = D_P$. Hence, (3.71) is equivalent to $\sigma(Q_P^T D_P^{-1} Q_P e, e)_E \leq (D_P e, e)_E$ which, because of $D_P > 0$, $Q_P^T D_P^{-1} Q_P > 0$ and (2.59), is equivalent to

$$\sigma \leq 1/\rho(D_P^{-1} Q_P^T D_P^{-1} Q_P) \ .$$

A sufficient condition for the latter inequality is given by

$$\sigma \leq 1/|D_P^{-1} Q_P^T| \, |D_P^{-1} Q_P|$$

for the operator norm $|\cdot|$ induced by an arbitrary vector norm (cf. (2.50) and (2.48)). We have

$$(D_P^{-1} Q_P)_{(k,l)} = \begin{cases} A_{(k,k)}^{-1} A_{(k,l)} & \text{for } l \leq k \, , \\ 0 & \text{else,} \end{cases}$$

$$(D_P^{-1} Q_P^T)_{(k,l)} = \begin{cases} A_{(k,k)}^{-1} A_{(l,k)}^T = A_{(k,k)}^{-1} A_{(k,l)} & \text{for } l \geq k \, , \\ 0 & \text{else.} \end{cases}$$

For a vector $w = (w_k)_{k=1,\ldots,np} > 0$, the matrix norm $||\cdot||_w$ defined by

$$||B||_w := \max_k \left\{ \frac{1}{w_k} \sum_l w_l ||B_{(k,l)}||_E \right\}$$

is the operator norm to the vector norm $||v||_w := \max_k \left\{ \frac{1}{w_k} ||v_{(k)}||_E \right\}$. For this special choice, we get

$$||D_P^{-1} Q_P||_w = \max_k \left\{ \frac{1}{w_k} \sum_{l \leq k} w_l ||A_{(k,k)}^{-1} A_{(k,l)}||_E \right\} = 1 + \gamma_-^p \, ,$$

$$||D_P^{-1} Q_P^T||_w = \max_k \left\{ \frac{1}{w_k} \sum_{l \geq k} w_l ||A_{(k,k)}^{-1} A_{(k,l)}||_E \right\} = 1 + \gamma_+^p \, ,$$

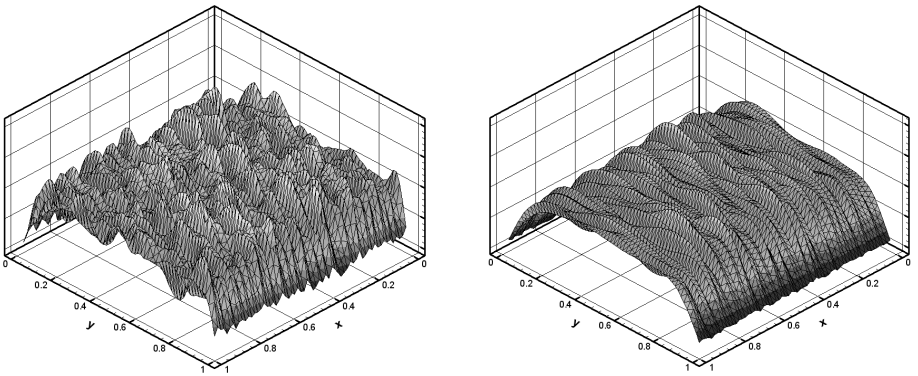
which proves the theorem. ■

3.4.1.2 Investigations of Point Smoothing for the Model Problems

In this section, the point-smoothing properties of BGS are examined for the three model classes defined in Section 3.1.3. In particular, the results for the RD and DD model classes are also important for subsequent discussions.

Consider $A = L_S$ defined by the **AVL model** (3.1). For each of the two cases $l < k$ and $l > k$, two matrices $A_{(k,k)}^{-1} A_{(k,l)}$ are nonzero and of the form (ignoring boundary conditions)

$$A_{(k,k)}^{-1} A_{(k,l)} = -\frac{d}{2 + 2\epsilon} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \ ,$$



(a) (b)

Figure 3.1: BGS relaxation for the DD model with h=1/64 (n_v=11 907), (λ, c, ϵ)=(1e-3,1e3,1e-3): Error of unknown u_1 after (a) 1 iteration, (b) 5 (and qualitatively all following) iterations. The errors of u_1, u_2 and u_3 are similarly smooth. For and (λ, c, ϵ) =(1e-9,1e9,1e-3) the plots are very similar.

one matrix with $d = 1$, and one with $d = \epsilon$. For $w \equiv 1$, we thus obtain

$$\gamma_+^p = \gamma_-^p = \frac{1}{2+2\epsilon} + \frac{\epsilon}{2+2\epsilon} = \frac{1}{2} \ ,$$

independent of a, b, c. Although the matrix $A = L_D$ defined by the **AVLD model** (3.2) is not symmetric so that (3.10) can strictly speaking not be applied, the resulting σ can serve as a hint how BGS works for this case. We obtain

$$A_{(k,k)}^{-1} A_{(k,l)} = \begin{cases} -\frac{1}{2+2\epsilon} \begin{bmatrix} \epsilon & 0 \\ 0 & 1 \end{bmatrix} & \text{for one } l < k,\ l > k \text{ each} \\ & (\text{“on the } x\text{-axis of the stencil”}), \\ -\frac{1}{2+2\epsilon} \begin{bmatrix} 1 & 0 \\ 0 & \epsilon \end{bmatrix} & \text{for one } l < k,\ l > k \text{ each} \\ & (\text{“on the } y\text{-axis of the stencil”}), \end{cases}$$

and thus arrive at the same $\gamma_+^p = \gamma_-^p$ as for L_S. Therefore, for the AVLS and AVLD models, we arrive at $\sigma = 4/9$ - *completely independent of* h, ϵ, a, b, c. This is in accordance with numerical results: even for the asymmetric L_D, BGS is an efficient smoother. Applied to L_S, the smoothing effect is as for VGS applied to $L_{x:\epsilon}$: the error is smooth in y-direction, but, unless ϵ approaches 1, not in x-direction. For both unknowns, smooth error is thus qualitatively identical to Fig. 3.1. In contrast to this, applied to L_D, the error of unknown u_1 is smooth in the contrary direction than the error of unknown u_2: the smoothing effect on unknown u_1 (u_2) is similar to that of VGS applied to $L_{x:\epsilon}$ ($L_{y:\epsilon}$).

We can also explain these results heuristically by observing that, for both L_S and L_D, the error $e_{(k)}$ is the approximate sum of the four summands $A_{(k,k)}^{-1} A_{(k,l)} e_{(l)}$ ($l \neq k$) with

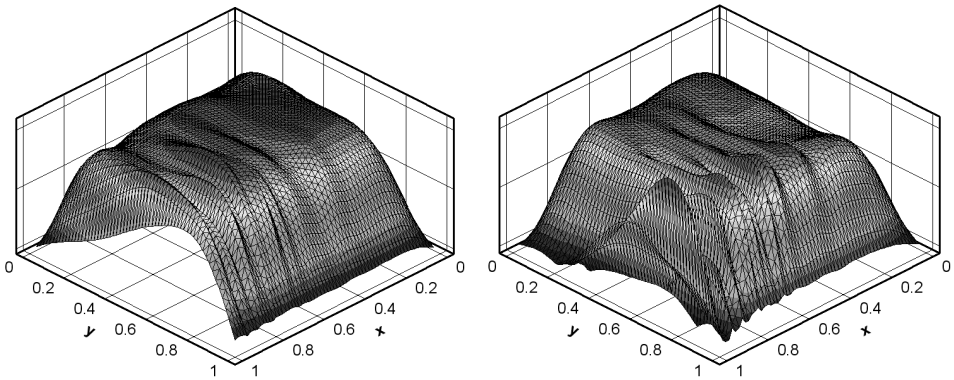


Figure 3.2: BGS relaxation for the DD model with h=1/64 (n_v=11 907), (λ, c, ϵ)=(1,1,1e-3): Error of (a) unknown u_1, (b) unknown u_2 after 10 (and qualitatively all following) iterations. The smoothness of the error of u_3 is similar to that of u_2. The errors after 1 iteration look similar to Fig. 3.1(a).

diagonal[30] matrices $A_{(k,k)}^{-1}A_{(k,l)}$ as shown above. In case of L_S, by construction, the point-couplings exhibit the same anisotropy as $L_{x:\epsilon}$. That means, for each variable i, the contribution of the two nearest neighbors, belonging to the same unknown than i, in x-direction to the sum is smaller by a factor of ϵ than the contribution of the two nearest neighbors in y-direction. For both unknowns, BGS thus reflects the anisotropy in x-direction. In case of L_D, however, the $A_{(k,k)}^{-1}A_{(k,l)}$ are identical besides the fact that for two of them the diagonal entries are permuted compared to the other two. This has the effect that, for small ϵ, the error of a variable belonging to u_1 (u_2) is the approximate sum of the errors of the two nearest neighbors in y-direction (x-direction) belonging to u_1 (u_2).

For the **AVLX model** (3.3) and $\delta := 4c^2 - ab(1+\epsilon)^2 \neq 0$, we obtain

$$A_{(k,k)}^{-1}A_{(k,l)} = \begin{cases} \frac{1}{2\delta}\begin{bmatrix} ab\epsilon(1+\epsilon)-2c^2 & -bc(1-\epsilon) \\ ac(1-\epsilon) & ab(1+\epsilon)-2c^2 \end{bmatrix} & \text{for one } l<k,\ l>k \text{ each} \\ & (\text{“on the } x\text{-axis of the stencil”}), \\ \frac{1}{2\delta}\begin{bmatrix} ab(1+\epsilon)-2c^2 & bc(1-\epsilon) \\ -ac(1-\epsilon) & ab\epsilon(1+\epsilon)-2c^2 \end{bmatrix} & \text{for one } l<k,\ l>k \text{ each} \\ & (\text{“on the } y\text{-axis of the stencil”}), \end{cases}$$

For $w \equiv 1$ and $\delta > 0$, we obtain[31]

$$\gamma_+^p = \gamma_-^p = \frac{1}{4\delta}\left(\delta + \sqrt{ab(1-\epsilon)^2\delta}\right) \ .$$

[30] Therefore, only variables belonging to the same unknown than a variable i belongs to contribute to e_i.

[31] The software Maple [108] has been used to calculate the $A_{(k,k)}^{-1}A_{(k,l)}$ and their eigenvalues.

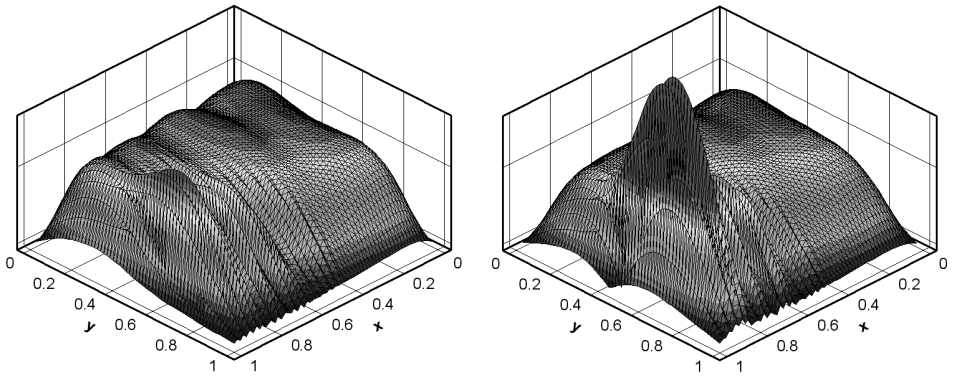


Figure 3.3: ILU relaxation for the DD model with h=1/64 (n_v=11 907), (λ, c, ϵ)=(1,1,1e-3): Error of unknown u_1 after (a) 4, (b) 6 iterations.

In contrast to L_S and L_D, the $A_{(k,k)}^{-1}A_{(k,l)}$ are not diagonal; they are not even symmetric here. For $a = b$, we have $(A_{(k,k)}^{-1}A_{(k,l)})_{12} = -(A_{(k,k)}^{-1}A_{(k,l)})_{21}$. Moreover, γ_+^p, γ_-^p and thus σ strongly depend on a, b, c, ϵ. The larger $|c|$ compared to a, b, the more dominating the unknown cross-couplings, cL. Therefore, for large $|c|$, the smoothing effect of BGS applied to L_X is similar to the effect of VGS applied to L. If a and b are much larger than $|c|$, BGS produces an error for unknown u_1 (u_2) which is as smooth as the error VGS applied to $L_{x:\epsilon}$ ($L_{y:\epsilon}$) produces. However, if ϵ is small and $a = b$ is only moderately larger than $|c|$, for instance $a = b \approx 2|c|$, BGS is not an appropriate smoother any more (see also Table 3.1).

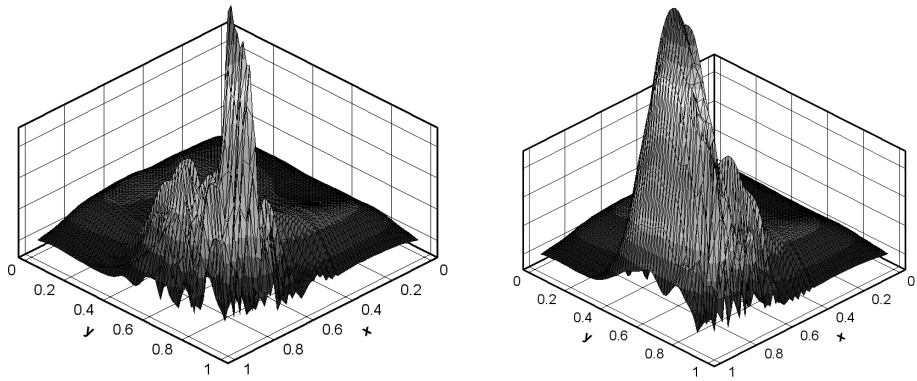
From Section 3.3.3.1 and the numerical results shown in Table 3.2 we know that VGS and UGS are not appropriate for the **RD models** as soon as $c = h^2 f$ or n_z are quite large. However, they work - at least for small n_z - also for $c = 1$ which is considerably larger than $4\,(1 - \cos(\pi h)) = O(h^2)$ being the limit for $A \in \mathcal{A}_{\text{spd}}$ [32]. In the following, we show that BGS is an appropriate smoother even for large n_z and c. Assume $n_z = n_p$ and $c^2 \neq 16$. Then observe that, for each k, there exist four indices l for which

$$A_{(k,k)}^{-1}A_{(k,l)} = \frac{1}{c^2 - 16}\begin{bmatrix} 4 & -c \\ -c & 4 \end{bmatrix} ,$$

two l are smaller than k, and two larger. The remaining $A_{(k,k)}^{-1}A_{(k,l)}$ $(k \neq l)$ vanish. Since $c > 0$ and

$$\left|\left|\frac{1}{c^2 - 16}\begin{bmatrix} 4 & -c \\ -c & 4 \end{bmatrix}\right|\right|_E = \max\left\{\frac{|4 + c|}{|c^2 - 16|}, \frac{|4 - c|}{|c^2 - 16|}\right\} = \max\left\{\frac{1}{|c - 4|}, \frac{1}{|c + 4|}\right\} = \frac{1}{|c - 4|} ,$$

[32] In Table 3.2, numerical results for the quite small $h = 1/512$ are presented.



(a) (b)

Figure 3.4: ILU relaxation for the DD model with h=1/64 (n_v=11 907), (λ, c, ϵ)=(1,1,1e-3): Error of unknown u_2 after (a) 4, (b) 6 iterations. The smoothness of the error of u_3 is similar to that of u_2.

we obtain, for $w \equiv 1$, $\gamma_+^p = \gamma_-^p = \frac{2}{|c-4|}$, and therefore

$$\sigma = \left(\frac{|c-4|}{|c-4|+2} \right)^2 .$$

The system matrix A is symmetric positive definite if and only if $h^2 f = c < 4\,(1 - \cos(\pi h)) = O(h^2)$. Of course, we then arrive at $\sigma \lessapprox 4/9$. With growing c, we obtain $\sigma = 1$ in the limit. Although Theorem 3.8 cannot be applied any more due to $A \notin \mathcal{A}_{\text{spd}}$, this large σ reflects that BGS approaches a direct solver for growing c and $n_z \approx n_p$. This is also true for ILU(0) which (at least in terms of convergence) is sometimes more efficient than BGS here. The above considerations are confirmed by the numerical results shown in Table 3.2. Summarizing, BGS and ILU(0) are appropriate smoothers for the RD models for all c and n_z whereas VGS and UGS are not as soon as c or n_z exceed a certain limit.

The **DD models** clearly leave the range of matrices covered by theory. Hence, in particular Theorem 3.10 cannot be applied any more nor can we expect that AMG stand-alone converges in all cases. Numerical results for the DD models are given in Table 3.3 and Figs. 3.1, 3.2, 3.3 and 3.4. Both the ARFs and the resulting smooth errors after some BGS relaxation steps indicate that BGS is an appropriate smoother for the DD models.

In contrast to BGS, the methods VGS, UGS and also ILU(0) diverge. *The divergence alone is, however, not the decisive criterion whether an approach can efficiently be used as a smoother. A smoother can help improving the convergence of the overall approach even if it diverges stand-alone. This can be the case if the divergence of the smoother is only due to some error frequencies which can efficiently be treated by the AMG approach or, finally, the accelerator employed.* Indeed, this can be observed for ILU(0) smoothing. Figs. 3.3 and 3.4 show that divergence occurs only in a small area. Outside, ILU(0) produces smooth errors. In

contrast to this, VGS and UGS diverge everywhere and "without a pattern" in the simulation domain. The divergence is even too strong to be plottable.

For $\epsilon = 1$, the errors of u_1,u_2,u_3 are smooth in all directions, regardless of the concrete (λ, c). If $\epsilon \ll 1$, the strong anisotropy of $A_{[2,1]}$ can only be seen in one corner[33] of the domain in case of (λ, c)=(1,1), but it is the stronger reflected, the larger c and smaller λ are. This has to be taken into account when constructing coarsening and interpolation later on. Note also that for large c (and small λ), the smoothness of the error of u_1 (see Fig. 3.2(a)) one one hand, and the smoothness of the errors of u_2 and u_3 (see Figs. 3.2(b)) on the other hand is somewhat different.

a	c	ARF_2(VGS)	ARF_2(UGS)	ARF_2(BGS)	ARF_2(ILU(0))
10	1	div (0.107e+1)	div (0.107e+1)	div (0.108e+1)	div (0.126e+6)
2	1	div (0.456e+1)	div (0.455e+1)	div (overflow)	div (overflow)
1	2	div (overflow)	div (0.760e+3)	0.998e+0	0.998e+0
1	10	div (overflow)	div (0.493e+6)	0.998e+0	0.998e+0

[a] ϵ_{it}=1e-10 is reached in only 4 cycles. ARF = 0.316e-02.

Table 3.1: ARF_2 for VGS, UGS, BGS, ILU(0) applied to different AVLX models. $a = b$, $\epsilon =$ 1e-3, $h =$1/512 (n_v=522 242). See Section 2.4.6 for the definition of ARF_2.

n_z	c	ARF_2(VGS)	ARF_2(UGS)	ARF_2(BGS)	ARF_2(ILU(0))
1	1e0	0.965	0.965	0.965	0.968
	1e3	div (0.153e+20)	div (0.153e+20)	0.965	0.931
	1e9	div (0.153e+68)	div (0.153e+68)	0.965	0.826
100	1e0	0.965	0.965	0.965	0.968
	1e3	div (0.150e+21)	div (0.150e+21)	0.965	0.865
	1e9	div (0.147e+69)	div (0.147e+69)	0.864	0.867
1000	1e0	div (0.188e+01)	div (0.188e+01)	div (0.216e+1)	div (0.262e+01)
	1e3	div (0.176e+21)	div (0.176e+21)	0.965	0.853
	1e9	div (0.255e+69)	div (0.255e+69)	0.659[a]	0.858

[a] ϵ_{it}=1e-10 is reached in only 4 cycles. ARF = 0.316e-02.

Table 3.2: ARF_2 for VGS, UGS, BGS, ILU(0) applied to different RD models. $h =$1/512.

λ	c	ARF_2(VGS)	ARF_2(UGS)	ARF_2(BGS)	ARF_2(ILU(0))
1e0	1e0	div (0.221e+16)	div (0.382e+15)	0.964	div (0.162e+01)
1e-3	1e3	div (0.435e+40)	div (0.363e+39)	0.964	div (overflow)
1e-9	1e9	div (0.948e+73)	div (0.327e+73)	0.964	div (overflow)

Table 3.3: ARF_2 for VGS, UGS, BGS, ILU(0) applied to different DD models. $h =$1/512 ($n_v = 783\,363$). Shown are results for ϵ=1. Results for $\epsilon =$ 1e-3 are slightly better.

[33] namely the corner where x and y approach 1. This is due to the definition of f_n in Section 3.1.3.3.

3.4.2 Primary Matrices and Point-Coarsening

A reasonable primary matrix $\boldsymbol{P} = (p_{kl})$ has to reflect the couplings $A_{(k,l)}$ between the points reasonably well. Depending on the type of application, there are many possibilities for defining a primary matrix - at least from a technical point of view: Not all of them lead to an efficient overall approach (if any). Often, the construction of $\boldsymbol{P}$ can be done automatically as part of AMG's setup phase. In other cases, it may be better to let the user of (S)AMG provide a reasonable matrix himself [34], based on his knowledge of the underlying physics of the given problem. In all cases, a primary matrix can usually be interpreted as describing the connectivity structure of some auxiliary (scalar) unknown. Clearly, this unknown should represent the physical connectivity structure of all "real" unknowns in the given system of PDEs reasonably well, i.e. simultaneously for all of them.

We now describe different processes for defining a primary matrix. In Section 3.4.2.1, we start with a general discussion of the possibilities to represent the couplings between the points by means of a $\boldsymbol{P}$. Concrete possibilities for defining the entries of $\boldsymbol{P}$ are described in Sections 3.4.2.2 to 3.4.2.4. We discuss primary matrices which equal one of the $A_{[n,n]}$, and we discuss several variants of primary matrices based on norms of the $A_{(k,l)}$ or coordinates of the grid nodes. In Section 3.4.2.5, some other possibilities are indicated. Afterwards (Section 3.4.2.6), two general possibilities for the recursive definition of primary matrices in case of a *multi*-level method are discussed. Finally, in Section 3.4.2.7, the point-coarsening process is generally explained and, for the concrete types of primary matrices introduced, further discussed.

3.4.2.1 Representation of Point-Couplings

For setting up a primary matrix, we need to define the connectivity pattern[35] of $\boldsymbol{P}$ as well as the concrete values p_{kl}. Three general approaches are imaginable:

- based directly on information contained in the matrix A and its VU and VP mappings. The simplest variant is the employment of a submatrix of A, for example, one of the $A_{[n,n]}$ if possible (see Section 3.4.2.2). The most natural variant here is the usage of norms of the $A_{(k,l)}$ (see Section 3.4.2.3). The latter can always be employed, at least technically. In practice, an *automatic construction* of corresponding types of primary matrices is possible.

- in addition to the above information, based on coordinates (see Section 3.4.2.4). It depends on the application whether such information is available. Then, an *automatic construction* of corresponding types of primary matrices is possible again.

- based on an auxiliary scalar problem (different from the above) representing the physics of the PDE system in a suitable way (see Section 3.4.2.5). An example might be a discrete Laplacian. In practice, a corresponding $\boldsymbol{P}$ needs to be provided externally.

[34] This will be discussed in Section 4.2.3.1.

[35] Recall from Section 2.4.2 that the connectivity pattern Σ of a matrix is defined to be the distribution of its nonzero entries.

Clearly, since A is a sparse matrix in all applications considered in this thesis, $\boldsymbol{P}$ has also to be sparse to yield an efficient approach. This directly leads to the general question of an appropriate connectivity pattern Σ of $\boldsymbol{P}$, which is considered now and prior to the discussion of the three ways mentioned above to define the concrete values of the p_{kl}.

If norms of the $A_{(k,l)}$ are used to define the p_{kl}, or if $\boldsymbol{P} := A_{[n,n]}$ (if possible; see also the next section), this automatically leads to a sparse $\boldsymbol{P}$ ("inherited sparsity"). In case of coordinates, however, the connectivity pattern of $\boldsymbol{P}$ must be determined in addition. It seems reasonable to assign a zero value to a p_{kl} if $\mathcal{P}_k$ is not coupled to $\mathcal{P}_l$, as in case of norms. This should also serve as a guideline for most cases where another measure is used to define $\boldsymbol{P}$.

Whenever $\boldsymbol{P}$ is defined automatically within the setup phase of an approach belonging to our framework, the connectivity pattern of A is the basis for that of $\boldsymbol{P}$. Two different types of patterns are considered, the so-called **unknown-pattern (u-pattern)** and **maximal pattern**, respectively:

- **unknown-pattern (u-pattern):** One possibility to define a connectivity pattern for $\boldsymbol{P}$ is copying the connectivity pattern of $A_{[n,n]}$ for any $1 \leq n \leq n_u$. In this case, we call the n-th unknown the **primary unknown** and the corresponding pattern the n-**th unknown-pattern**.

  If we want to stress that the n-th unknown-pattern is used, the primary matrix is denoted by $\boldsymbol{P}_n$ and its pattern by Σ_n.

  Note that the n-th unknown may serve as a primary unknown only if the connectivity pattern of $A_{[n,n]}$ is **complete** or **valid**, that is, the n-th unknown has to be represented at all points. In practice, it often happens that not all unknowns are represented at a point, that is, the number of variables may vary from point to point (cf. the applications discussed in Chapter 5). But a reasonable primary matrix is always required to represent *all* points and is thus not allowed to contain empty matrix rows. Therefore, if the connectivity pattern of an $A_{[n,n]}$ is not complete, $\mathcal{U}_n$ cannot serve as a primary unknown and the n-th u-pattern is said to be **not valid**.

- **maximal pattern:** We call the connectivity pattern of $\boldsymbol{P}$ maximal if it reflects the full point-coupling structure of A. To be more specific, a (k,l) is in the connectivity pattern of $\boldsymbol{P}$ if $\mathcal{P}_k$ is coupled to $\mathcal{P}_l$. Obviously, this pattern is valid in all cases.

  If the usage of the maximal pattern shall be emphasized, we denote the primary matrix by $\boldsymbol{P}_{max}$ and its pattern by Σ_{max}.

Examples of different patterns are shown in Figures 3.5 and 3.6. Another possibiliy to visualize the pattern of the primary matrix is to use graphs, if necessary directed. For a coupling structure as depicted in Figure 2.2, the maximal pattern is shown in Figure 2.3(b) and the first u-pattern, the only valid one for this example, in Figure 2.3(a).

In the following sections, we describe several possibilities to define the non-vanishing entries of $\boldsymbol{P}$.

3.4.2.2 Unknown-Based Primary Matrix

An unknown-based primary matrix simply means $\boldsymbol{P} = A_{[n,n]}$ for any $1 \leq n \leq n_u$. This selection requires the n-th unknown to be complete ($|\mathcal{U}_n| = n_p$). Being natural in this case,

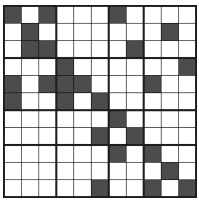


Figure 3.5: Connectivity pattern of a matrix A corresponding to eleven variables, three unknowns, and four points. The eleven variables are assumed to be sorted point-wise, and at each of the four points with increasing unknown number. At the third point, the third unknown does not exist. Hence, one column and one row are missing.

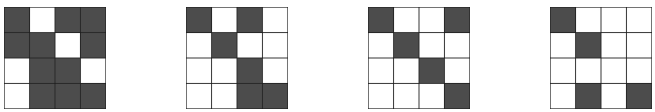


Figure 3.6: Corresponding connectivity patterns $\Sigma_{max}, \Sigma_1, \ldots, \Sigma_3$. The pattern Σ_3 is not valid because the third unknown does not exist at the third point (empty row!).

only $\Sigma = \Sigma_n$ is taken into account here. Whether or not this choice of $\boldsymbol{P}$ makes sense, depends on the application, in particular, whether the connectivity structure of the n-th unknown is also representative for the other unknowns.

Note that employing some externally defined matrix as a primary matrix may be realized via an unknown-based primary matrix. Externally defined primary matrices will be discussed in Section 3.4.2.5.

In the following two examples and two remarks, we discuss applications of this simple variant. We start with an example on how certain Navier-Stokes systems might be solved by a PAMG approach using an unknown-based $\boldsymbol{P}$. In the following two remarks, we review point-oriented aggregative AMG approaches that are used for solving certain CFD problems. Relationships to our framework are indicated, revealing that the approaches proposed in the literature correspond to a point-based approach with an unknown-based primary matrix. In addition, some recent research activities are mentioned. We conclude with Example 3.2 on an unknown-based $\boldsymbol{P}$ for drift-diffusion systems.

Example 3.1 An unknown-based primary matrix might be reasonable for Navier-Stokes systems, at least if discretized on non-staggered grids. One possibility might be the following. If the "pressure" equation in the system (i.e. the equation not corresponding to a velocity) contains (at least) an artificial viscosity or penalty term ("$h^2\Delta$") as an artificial, weak contribution of the pressure, the corresponding $A_{[3,3]}$ might be chosen as the primary unknown. An investigation of corresponding AMG approaches will be a topic of future research. See also the following two remarks. Another possibility arises if a pressure-correction equation is already available or can be created in addition to the Navier-Stokes system to be solved.

See Section 3.4.2.5. ▲

Remark 3.19 (Aggregative AMG for Navier-Stokes Equations) For the system of Navier-Stokes equations, we can find both segregated as well as coupled solution approaches in the literature. In a segregated approach, scalar PDEs are solved. AMG for this application is discussed, for example, in [29, 87] and references given therein.

More and more, the Navier-Stokes system is solved fully coupled in CFD[36] simulators. For instance, the point-based approach [63, 64] can be characterized as an AMG/ACM[37] method suitable for the coupled, linearized, discrete equations arising from an implicit FV discretization of the 3D Navier-Stokes equations. It employs an ILU(0) factorization as a smoother. The coarsening of the finite volumes is done by an agglomeration method with a strategy based on the strength of pressure coefficients. This measure of strength of couplings is similar to the one that would be employed by a point-based approach of our framework with the pressure being the primary unknown (see also Example 3.1 above). However, restriction and interpolation are simply piecewise constant, classifying this AMG solver to be of the nonsmoothed aggregation type.

Other nonsmoothed point-based aggregation-type AMG approaches for the solution of the finite-volume discretized Navier-Stokes systems can be found in [109] and references therein. Main differences to the approach described before are that an augmented Navier-Stokes system, consisting of five equations for the 3D case, is solved, and that these approaches employ block-Gauss-Seidel smoothers.

Recently, in [101] a new AMG-like concept for the coupled solution of the Oseen problem[38], discretized by a mixed finite element method, has been introduced. The concept employs techniques known from GMG for saddle point problems such as Braess-Sarazin- or Vanka-type smoothing and, for two mixed FE discretizations, investigates possibilites for a(n alternate or "shifted") coarsening of the velocity and pressure components. This approach is still in its infancy. ▲

Remark 3.20 (Oil Reservoir Simulation) Very recently, a straightforward point-based extension of smoothed aggregative AMG for the solution of FV-discretized oil reservoir simulation systems is discussed in [60]. Except for the fact that this approach uses smoothing of aggregation (only applied to the pressure coefficients), it resembles the aggregative AMG approaches mentioned above for the Navier-Stokes equations. In particular, aggregation of cells based on a "strong graph of the scalar pressure equation" [39] has been found to be robust and efficient also for the oil reservoir applications considered. Again, this graph of strong connections resembles the one employed in a PAMG approach of our framework with the pressure being the primary unknown.

UAMG and certain PAMG approaches with an unknown-based or norm-based $\boldsymbol{P}$ can be applied successfully to certain PDE systems arising in oil reservoir simulation. Since this is subject to recent research activities, detailed results will be published somewhere else. ▲

[36] CFD = computational fluid dynamics.

[37] ACM = additive correction multigrid, see references in [63, 64].

[38] arising from a fixed-point iteration for the nonlinear incompressible Navier-Stokes system.

[39] In a preprocessing step, an equation that resembles the standard pressure equation is determined.

Example 3.2 Numerical results for drift-diffusion systems (see Section 5.3.2) have shown that, in some cases, the selection of the potential as the primary unknown yields an efficient approach. However, this approach has not proved to be robust. ▲

3.4.2.3 Norm-Based Primary Matrix

A special point-based approach, sometimes called **block approach**, has already been sketched in the very early paper [71][40]. This approach has to some extent been further investigated, for instance, in [58, 30][41]. In this thesis, we generalize the ideas of [71] and, compared with [30] and others, develop more practically applicable variants. This will be particularly true for interpolation, as discussed in Section 3.4.3, but is also true for the variants we propose to construct a norm-based primary matrix.

In the discussion of norm-based primary matrices, the representation (3.69) is a natural starting point. The entries of $\boldsymbol{P}$ are then defined based on some norm of the point-coupling matrices $A_{(k,l)}$ describing the point-wise connectivity. Especially in the norm-based approach, the primary matrix can thus be interpreted as a "condensation" of A in the sense that each point-coupling matrix $A_{(k,l)}$ is reduced to a scalar entry p_{kl} of $\boldsymbol{P}$. Besides the concrete selection of a norm, which will be discussed below, this can be done in several ways. In this thesis, we consider four different variants. Two straightforward possibilities are the following: for all $(k,l) \in \Sigma$, $k \neq l$, define

$$(1) \quad p_{kl} = -||A_{(k,l)}|| \quad \text{and} \quad p_{kk} = ||A_{(k,k)}|| \tag{3.72}$$

$$(2) \quad p_{kl} = -||A_{(k,l)}|| \quad \text{and} \quad p_{kk} = -\sum_{l \neq k} p_{kl} \tag{3.73}$$

with $||\cdot||$ denoting a suitable norm. Both variants are simple to compute and, hence, suitable for practical use. However, in contrast to scalar AMG, there is no sign-condition here: all off-diagonal entries are simply defined to be negative, regardless of the sign distribution of A. Therefore, one might be attempted to extend these definitions of $\boldsymbol{P}$, for instance, by assigning positive entries, $||A_{(k,l)}||$, to those p_{kl}, for which $A_{(k,l)} \geq 0$. This leads to the following two variants: for all $(k,l) \in \Sigma$, $k \neq l$, define

$$(3) \quad p_{kl} = \begin{cases} ||A_{(k,l)}|| & \text{if } A_{(k,l)} \geq 0, \\ -||A_{(k,l)}|| & \text{otherwise,} \end{cases} \quad \text{and} \quad p_{kk} = ||A_{(k,k)}|| \tag{3.74}$$

$$(4) \quad p_{kl} = \begin{cases} ||A_{(k,l)}|| & \text{if } A_{(k,l)} \geq 0, \\ -||A_{(k,l)}|| & \text{otherwise,} \end{cases} \quad \text{and} \quad p_{kk} = \sum_{l \neq k} |p_{kl}| \tag{3.75}$$

with $||\cdot||$ denoting a suitable norm. Not unexpectedly, one can observe that these two approaches are a bit simpler to handle for theoretical considerations as performed in Section 3.4.4. However, at least the question on the correct handling of indefinite $A_{(k,l)}$ remains. Anyway, variants (3.72) and (3.73) are much cheaper to compute than (3.74) and (3.75).

[40] In [71], the idea of a point-wise coarsening and block-interpolation (see Section 3.4.3.1) was outlined, and the result of a preliminary test was given.

[41] cf. also Remarks 3.27 and 3.38 for some comments.

Therefore, only the variants (3.72) and (3.73) are used in practice, even if some $A_{(k,l)}$ ($k \neq l$) are positive definite or indefinite.

Remark 3.21 *For instance, due to the problems in handling positive semi-definite and indefinite $A_{(k,l)}$ ($k \neq l$) or the problems in handling anisotropies which are different from unknown to unknown, it is* not *true that a norm-based point approach is a very straightforward or even* the *natural extension of AMG to systems of PDEs, an opinion which is nevertheless sometimes advocated in the scientific community. This topic will be discussed a bit further in Section 3.4.2.7.* ▲

Remark 3.22 A norm-based primary matrix principally inherits the sparsity of A. ▲

Remark 3.23 Note that the p_{kk} are always greater than or equal to zero in each of the four variants. $p_{kk} > 0$ is true for (3.74) and (3.72) because of $A > 0$. In (3.75) and (3.73), p_{kk} can only be zero if all off-diagonal point-coupling matrices $A_{(k,l)}$ of the row are zero. But because of $A_{(k,k)} \neq 0$, $\mathcal{P}_k$ is then an **isolated point**, and we *define* p_{kk} to be equal to 1. As in case of Dirichlet variables, such points become **forced F-points** in the coarsening process and obtain "empty"interpolation formulas as a consequence. *As always, such "trivial" points are tacitly excluded in the discussion of interpolation formulas.* ▲

Norms Regarding the concrete choice of the norm, the *Euclidean* matrix norm, $||B||_E = \sqrt{\max|\lambda(B^T B)|}$ ($= \rho(B)$ for $B > 0$), is convenient for theoretical considerations (see Section 3.4.4), because it is the operator norm induced by the Euclidean vector norm. But since $||B||_E$ is too expensive to evaluate, various "cheaper"norms have been considered for *practical* applications, such as the maximum, row sum and Schur norm (see Section 2.4.5). The Schur norm is compatible to the Euclidean vector norm and often yields similar results than the Euclidean matrix norm. However, it is case-dependent which norm gives the best results, i.e. leads to a good representation of the point-coupling structure of A within $\boldsymbol{P}$.

Remark 3.24 It turns out that, in many applications and especially the ones considered in this thesis, AMG convergence rates do not significantly depend on the concrete choice of the norm. This is because often the point-coupling matrices $A_{(k,l)}$ are dominated by one or more equally large entries a_{ij} each so that the different norms considered above essentially coincide. Therefore, in order to minimize computational cost, we usually select the maximum norm in practice. ▲

Example 3.3 A norm-based $\boldsymbol{P}$ computed for the AVL model L_S always equals $L_{x:\epsilon}$, regardless which of the above norms has been chosen. For the AVL model L_D, an isotropic L is obtained. In both cases, it is reasonable to base coarsening on a norm-based $\boldsymbol{P}$ since it reflects the direction(s) of smoothness resulting from BGS relaxation (see Section 3.4.1.2) correctly. Therefore, we expect point-based AMG with a norm-based $\boldsymbol{P}$ to yield an appropriate solver here. This is confirmed by the numerical results shown in Section 4.6. ▲

Example 3.4 For the RD models, the situation is comparable[42]: $\boldsymbol{P} \hat{=} L$. However, we should have in mind here that in practical applications as presented in Section 5.2.2 large off-diagonal entries arising from reaction terms disturb the scenario. As discussed in Example 3.10, coarsening should still be based on an isotropic L which, however, cannot be obtained from a norm-based $\boldsymbol{P}$ any more. A way to obtain an "appropriate L" is discussed in Example 3.10. ▲

Example 3.5 The matrices corresponding to the DD models are dominated by $A_{[2,1]}$ in the sense that in the majority of cases the largest entry of an $A_{(k,l)}$ belongs to $A_{[2,1]}$. For (ϵ, λ, c)=(1e-3,1,1), for instance, about 60% of the dominating entries are in $A_{[2,1]}$. The remainder corresponds to entries in $A_{[n,n]}$ all of which correspond to the discrete isotropic Laplacian L. The larger c, that share tends to 100%. Therefore, chosing a norm-based primary matrix results in a "mixture" of $A_{[2,1]} = f_n(c,x,y)\, L_{x:\epsilon}$ and L reflecting that share. For sufficiently large c, we arrive at $\boldsymbol{P} = A_{[2,1]}$, regardless of the concrete norm. As long as $\epsilon \approx 1$, an unknown-based $\boldsymbol{P}$ equals - besides scaling - the norm-based $\boldsymbol{P}$ so that both types of primary matrices also yield the same coarsening process here: $\boldsymbol{P} \hat{=} L$.

For $\epsilon \ll 1$, however, only the norm-based variant correctly reflects the direction(s) of smoothness resulting from BGS relaxation (see Section 3.4.1.2). We therefore expect a norm-based $\boldsymbol{P}$ to be the best choice here. Numerical results are presented in Section 4.6. We will see in Section 5.3 that a comparable situation arises for the drift-diffusion systems. ▲

3.4.2.4 Coordinates-based Primary Matrices

The original AMG did not exploit any information on the given problem apart from the matrix A itself. In many PDE applications, this unnecessarily limits the possibilities for an efficient coarsening and interpolation. As a matter of fact, geometric information such as the coordinates of grid nodes is usually available and simply accessible, and could thus be exploited in AMG's setup phase. Note that this does not put any restrictions on the grid's *shape*.

If points correspond to real physical grid nodes in space, and if we assume their coordinates to be known, $\boldsymbol{P}$ may be constructed easily and automatically based on distances of points, leading to coarsening processes which are closely related to geometric coarsening (see also Section 3.4.2.7). A simple definition would be

$$p_{kl} = -1/\delta_{kl}^2 \quad (\forall\, (k,l) \in \Sigma,\ k \neq l) \quad \text{and} \quad p_{kk} = -\sum_{l \neq k} p_{kl} \tag{3.76}$$

where δ_{kl} denotes the distance between points $\mathcal{P}_k$ and $\mathcal{P}_l$, i.e.

$$\delta_{kl} := ||\pi_k - \pi_l||_E\,. \tag{3.77}$$

with π_i being the vector of coordinates corresponding to the i-th point.

As has been mentioned above, unlike Σ_{max} for a norm-based $\boldsymbol{P}$, no "natural" maximal Σ is imposed for coordinates-based $\boldsymbol{P}$ so that the choice of a suitable Σ is particularly important

[42] for (3.73) and (3.75). In case of (3.72) and (3.74), p_{kk} with $1 \leq k \leq n_z$ can be different from the corresponding diagonal entry of L.

in order to ensure the sparsity of $\boldsymbol{P}$. Since, usually, there is no reason to define an artificial coupling between points which are not coupled by matrix entries a_{ij}, the maximal pattern, Σ_{max}, or one of the unknown-patterns, depending on the application, should be chosen.

Remark 3.25 It would also be possible to orient Σ on some reasonable pattern of "strong" point-couplings, for instance, defined via norms of the point-coupling matrices and a suitable threshold. This would reduce the sparsity of $\boldsymbol{P}$ further and prevent "weak point-couplings" from getting too much influence on $\boldsymbol{P}$ and thus the coarsening. However, it depends on the case whether this is advantageous or not. ▲

The resulting $\boldsymbol{P}$ is called **distance-based** primary matrix. Compared with $1/\delta_{kl}$, the above variant strengthens the coupling between two nearby points. If Σ corresponds to the connectivity pattern of a Laplacian second-order discretized by means of, for instance, the five-point stencil in case of a regular grid or a standard FV discretization on a Delaunay grid (see Section 5.3.1.5) or a similar discretization, the employment of a distance-based $\boldsymbol{P}$ would result in a coarsening strongly related to the one employed in standard geometric multigrid. This and possible applications of a distance-based $\boldsymbol{P}$ are further discussed in Section 3.4.2.7.

The above distance-based primary matrix does not take positions of the points into account. Problems may hence arise if a point $\mathcal{P}_k$ is coupled to other points not surrounding $\mathcal{P}_k$. To overcome this limitation of a mere distance-based approach, we might construct a primary matrix which punishes non-uniformly distributed couplings by appropriate scalings of the distance-based p_{kl}. For a two-dimensional simulation domain, such a **position-based** primary matrix can be constructed in the following way: for each row k of $\boldsymbol{P}$ do

1. construct the distance-based entries p_{kl} ($k, l \in \Sigma$) of this row,
2. if the matrix row k has more than 2 nonzero entries, scale each off-diagonal entry p_{kl} by a factor f_{kl} computed as follows:
   (a) compute the signed angle α ($|\alpha| \in [0; 180°[$) between $\pi_k\pi_l$ and $\pi_k\pi_m$ for all $m \neq l, k$,
   (b) determine the smallest angle α_1 counterclockwise and the smallest angle α_2 clockwise,
   (c) $f_{kl} = (\alpha_1 + \alpha_2)/180°$, $p_{kl}^{\text{new}} = f_{kl}\, p_{kl}^{\text{old}}$.

   After having scaled all entries, compute $p_{kk} = -\sum_{l \neq k} p_{kl}$.

This "two-dimensional cake-scaling" rewards those off-diagonal entries p_{kl} which are far[43] from their next neighbors and punishes especially those which lie in between two near neigbors. Due to multiplying the distance-based p_{kl}^{old} by f_{kl}, distances as well as angles are taken into account here. Note that this scaling does not punish all cases of non-uniformly distributed couplings in the three-dimensional case. However, non-uniformly distributed couplings mainly occur near boundaries or in case of a "strange" grid. In the first case, the position-based $\boldsymbol{P}$ has not performed more efficiently than the distance-based $\boldsymbol{P}$ in practice so far. In the second case, which is likely to be disadvantageous for a discretization of the problems we have in mind here, it should make more sense to "repair" the grid instead of punishing its "failures" afterwards. Therefore, we have not investigated this topic further.

[43] assuming equal distances to π_k.

3.4.2.5 Other Possibilities

Not in all cases is it possible to construct an (or the most) appropriate primary matrix by extracting it automatically only from the matrix A. In such a case, if an auxiliary scalar problem can be defined representing the point couplings in a reasonable way, the corresponding matrix of the problem would be a natural primary matrix. We now give three examples where this way of defining $\boldsymbol{P}$ is promising.

Example 3.6 If anisotropies in a given problem are mainly due to non-uniform mesh spacings, a suitable primary matrix might be given by a discretization of the Laplace operator. If coordinates are available, this might be mimiced by a coordinates-based primary matrix. If, however, coordinates are not available, an approximate discrete Laplacian might externally be defined to AMG. ▲

Example 3.7 In Section 3.2.3.4, we have discussed two possibilities to modify standard coarsening in order to treat the anisotropic Laplacian corresponding to the nine-point stencil (2.28) with $\epsilon = 0$ and $\alpha = 1/4$ correctly. Since an appropriate coarsening corresponds to the standard coarsening process for the strongly anisotropic five-point stencil $\mathcal{L}_{x:0,h}$, another possibility is given by a point-based[44] approach with $\boldsymbol{P} = L_{x:0,h}$. If a similar problem is posed on an unstructured grid, a suitable primary matrix might be obtained from a "suitable"[45] discretization of the anisotropic Laplacian $-\epsilon u_{xx} - u_{yy}$ on that grid. Normally, it should be possible to set up such a primary matrix in the same part of a simulation code in which the matrix of the original problem is set up. ▲

Example 3.8 One can also imagine cases where it makes sense to define a primary matrix based on some natural physical quantity for which there is no reasonable equation contained in the original system of PDEs. An example for such a case might be the pressure in the context of the Navier-Stokes equations. Instead of choosing a $\boldsymbol{P}$ representing an artificial viscosity term ("$h^2\Delta$", see Example 3.1), another possibility could be a (separate) pressure-correction equation. An advantage might be that this equation typically reflects also anisotropies and discontinuities ("material contrasts") which are physically contained in the system. ▲

Remark 3.26 (Practical Realization) As has been mentioned in Section 3.4.2.2, employing some externally defined matrix as a primary matrix may be realized via an unknown-based primary matrix. This could be done by augmenting the original matrix A by this primary matrix $\boldsymbol{P}$, interpreting $\boldsymbol{P}$ as the matrix of a new, artificial unknown in the system and selecting this unknown as the primary unknown. Implications of this approach and another possibility will be discussed in the next section and in Section 4.2.3.1. As has been stated above, externally defined primary matrices will be one direction of future research. ▲

[44] Remember that in the scalar case, each point corresponds to one variable.

[45] producing $a_{ij} \leq 0$ for all $i \neq j$. For instance, a finite-volume discretization might be a candidate.

3.4.2.6 Recursive Definition of Primary Matrices

So far we have described possibilities to set up a primary matrix on a given level. For a multi-level method, there are essentially two ways for the recursive definition of primary matrices:

- **separate definition:** Here, on each level, $\boldsymbol{P}$ is computed "from scratch". This is natural for norm-based as well as coordinates-based primary matrices.
- **full integration:** Here, a coarse-level primary matrix is computed via a Galerkin process. This is especially reasonable if on the finest level a submatrix of $A = A_h$ (one of the $A_{[n,n]}$) has been chosen as primary matrix. On the next coarser level, the corresponding part of A_H can then be chosen as the primary matrix on that level.

For externally defined (user-supplied) primary matrices, both variants might be considered. We will come back to this in Section 4.2.3.1.

3.4.2.7 Point-Coarsening Strategy

In our framework, point-coarsening is always performed based on a primary matrix $\boldsymbol{P}$. After having investigated possibilities to set up concrete primary matrices in the last sections, we now explain the process of coarsening the set of variables by means of a given primary matrix and discuss the impact of the concrete type of primary matrix chosen on the point-coarsening process.

As indicated at the beginning of Section 3.4, once we have defined a primary matrix $\boldsymbol{P}$, a "classical" variable-based coarsening process is applied to $\boldsymbol{P}$ to obtain a C^p/F^p-splitting of the set of points. To be more specific, the basic criterion (3.24) for the strength of a coupling is applied to the set $\mathcal{P}$ of points and the primary matrix $\boldsymbol{P}$ chosen. That is, we define the point $\mathcal{P}_k$ to be *strongly (negatively) coupled* to the point $\mathcal{P}_l$ ($k \neq l$) if

$$-p_{kl} \geq \epsilon_{\mathrm{str}} \max_{j \neq k} |p_{kj}^{-}| \,. \tag{3.78}$$

Remember that criterion (3.24) and variants are used in concrete variable-based coarsening schemes[46] in order to distinguish strong and weak couplings. Based on this, the final C/F-splitting is computed. Accordingly, in a point-based approach, the C^p/F^p-splitting emerges. To obtain the desired C/F-splitting in terms of variables, the C^p/F^p-splitting is then "copied" in a straightforward way to the variables: if $\mathcal{P}_k$ has been selected to be an F- or C-point, all variables attached to $\mathcal{P}_k$ will become F- or C-variables respectively (at least in the first step). Details of the implementation and special cases are described in Section 4.2.3.2.

Formally, the point-coarsening process as described above is a simple extension of the scalar coarsening process. However, whether or not this process is finally reasonable, i.e. whether it allows good AMG interpolation and, through this, results in efficient AMG solvers, strongly depends on the application. Moreover, the concrete selection of suitable primary matrices is very crucial for the success of the resulting AMG solver. It seems that there is no

[46] The variable-based coarsening schemes implemented in our library SAMG, namely standard and aggressive coarsening, will be described and discussed in Section 4.2.1.

general best strategy so that, in practice, one has to make compromises. A few aspects are discussed in this section. In particular, since (3.78) is central to the resulting C^p/F^p-splitting, we investigate this criterion for two important types, namely norm-based and distance-based primary matrices as introduced above.

Norm Coarsening In the scientific community, the block approach is often considered as the natural extension of AMG to systems of PDEs. However, this is the case only up to some extent. There are also cases for which the block approach is not convenient or even fails. As has been noted in Remark 3.21, questions arise on an appropriate definition of the p_{kl} if not all $A_{(k,l)} \leq 0$ ($k \neq l$), for instance. Ill-conditioned $A_{(k,l)}$ constitute another field of problems. We want to discuss this a bit further with the help of criterion (3.78).

The condition number of a matrix is defined by (2.57). The larger the condition number, the more ill-conditioned the matrix. In particular, a symmetric $A_{(k,l)}$ is ill-conditioned if $|\lambda|_{\min}(A_{(k,l)})$ is small compared with $|\lambda|_{\max}(A_{(k,l)})$. If a norm-based primary matrix has been chosen, based on (3.72) or (3.73) without restriction of generality, criterion (3.78) reads

$$||A_{(k,l)}|| \geq \epsilon_{\text{str}} \max_{j \neq k,\ (k,j) \in \Sigma} ||A_{(k,j)}|| \tag{3.79}$$

for a $(k,l) \in \Sigma$, $k \neq l$. Without additional requirements, it may happen now that a $\mathcal{P}_k$ is strongly coupled to a $\mathcal{P}_l$ according to (3.79) even if $|\lambda|_{\min}(A_{(k,l)})$ is small compared with the $||A_{(k,j)}||$ ($j \neq k,\ (k,j) \in \Sigma$). Since, depending on the application, this may give rise to a bad interpolation later on and thus an inefficient[47] overall approach, the goal should be to avoid situations where

$$||A_{(k,l)}|| \geq \epsilon_{\text{str}} \max_{j \neq k, (k,j) \in \Sigma} ||A_{(k,j)}|| \quad \wedge \quad |\lambda|_{\min}(A_{(k,l)}) \ll \epsilon_{\text{str}} \max_{j \neq k,\ (k,j) \in \Sigma} ||A_{(k,j)}|| \ . \tag{3.80}$$

This is illustrated with the following example.

Example 3.9 The condition number of an $A_{(k,l)}$ is strongly influenced by scalings of this matrix. Correspondingly, a norm-based approach reacts very sensitive to scalings of A. The largest impact is felt if the rows corresponding to the variables belonging to the same point are not scaled simultaneously and "uniformly".

For some matrix classes, norm-based primary matrices are principally invariant to such scalings, an example being the AVL models (3.1). Not invariant are matrices where off-diagonal couplings determine the efficiency of the approach as, for instance, in case of the DD models. In particular, if we scale every row corresponding to the second unknown by a small factor s here, we destroy the dominance of $A_{[2,1]}$ over the other $A_{[m,n]}$. Then, a norm-based $\boldsymbol{P}$ might not reflect the direction of smoothness any more (for small ϵ; see also Example 3.5). ▲

In case of any norm-based $\boldsymbol{P}$, the question arises if an incorporation of approximations of minimal eigenvalues in addition to or instead of norms of the $A_{(k,l)}$ (approximations of maximal eigenvalues) could yield better coarsenings and interpolations later on. Clearly, there

[47] Correspondingly, worse upper bounds for the two-level convergence rate emerge, see Section 3.4.4.

are ways to avoid the situation (3.80). For instance, in [58] a "min-max-coupling" criterion,

$$\min_{||x||=1} ||A_{(k,l)}x|| \geq \epsilon_{\text{str}} \max_{j \neq k} ||A_{(k,j)}|| \,, \tag{3.81}$$

(sic!) which intends comparing minimal with maximal eigenvalues was proposed. This criterion prevents the problem (3.80) mentioned above, but introduces a new one: It can happen that there is no strong coupling at all. For example, if the k-th block row of A contains many (nearly) singular block matrices with some large entries, then this criterion would define all point couplings to be weak in the worst case. Such a point $\mathcal{P}_k$ would then become a forced F-point - despite the large entries contained in the row. Therefore, this criterion cannot be expected to generally give good results or even better ones than a norm coarsening. Additionally, it is more expensive to evaluate and should therefore hardly ever pay (as the convergence rates for the simple test cases in [58] indicate, too).

In such critical cases as (3.80), the real question is if a norm-based or, in general, a point-based strategy is a good way *at all* to solve the given system. If too many block matrices are ill-conditioned, we must rather expect that it is at least not the optimal strategy. Therefore, criteria such as the "min-max-coupling" cannot be expected to help because they do not remove the real problem.

Up to now, there seems to be no final answer what kind of block approach (in particular, which norm) is "optimal". There are more possible approaches to perform the coarsening. Since not all of them are covered by our point-based framework, we want to list some typical ones in the following remark and comment on their practical usage.

Remark 3.27 In [58], five coarsening criteria have been investigated. Two of them are equivalent to criteria emerging from suitable primary matrices of our framework:

- The "norm-coupling" criterion corresponds to variant (3.72) with the row sum norm and a maximal pattern.
- Another criterion corresponds to the same variant but with the maximum norm. However, no numerical results for this criterion are given in [58].

Besides the "min-max-coupling" criterion discussed above, for another two criteria which cannot be obtained from a norm-based primary matrix and (3.78), experiments were performed in [58] for some simple model problems.

- One is the "min-coupling" criterion which must read

  $$\min_{||x||=1} ||A_{(k,l)}x|| \geq \epsilon_{\text{str}} \max_{j \neq k} \min_{||x||=1} ||A_{(k,j)}x||$$

  with the row sum norm and a maximal pattern. Due to the special structure of the simple model problems in [58], for them it behaves usually very similar to the "min-max-coupling" criterion. But in general it is problematic because $\min_{||x||=1} ||A_{(k,l)}x||$ cannot be used to define a norm. Additionally, it is usually much more expensive to evaluate.
- The other, "classical point-coupling" criterion defines a point coupling to be strong if a coupling of one of the corresponding variables is strong in the sense of the unknown-based approach. It completely neglects cross-couplings between the different unknowns

and should therefore only very seldom be useful in a point-based approach, as simple tests in [58] show. It leads to a similar behavior compared with a coarsening process based on a primary unknown if the coupling structures of all $A_{[n,n]}$ are similar to each other. But in this case, coarsening based on a primary unknown - i.e. based on *one* $A_{[n,n]}$ - is of course much cheaper.

Since there is no evidence that these criteria and similar ones provide particular advantages, we stick to *our* point-coarsening strategy based on a primary matrix. ▲

Coordinates-Based Coarsening In case of a distance-based primary matrix, (3.78) is equivalent to

$$\sqrt{\epsilon_{\mathrm{str}}}\delta_{kl} \leq \min_{j\neq k,\,(k,j)\in\Sigma} \delta_{kj}\,. \tag{3.82}$$

We have stated in 3.4.2.4 that, under certain conditions on the discretization, this leads to a "geometric" coarsening process. This is simple to see if the underlying grid is regular and Σ corresponds to the connectivity pattern of a Laplacian FD-discretized by means of the standard five-point stencil. For instance, a typical value of $\epsilon_{\mathrm{str}} = 1/4$ then leads to a "red-black" coarsening. Such a "uniform" coarsening also arises if Σ represents the connectivity structure of a FV-discretization on a Delaunay mesh (see Section 5.3.1.5). If the grid causes anisotropies, a distance-based primary matrix reflects them accordingly. For an illustration, see Fig. 3.7.

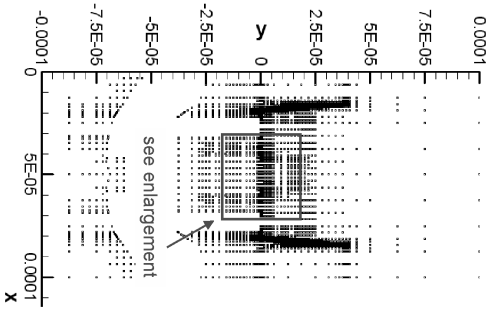


Figure 3.7: Grid of the STI test case for device simulation (see also Section 5.3.2.3). An enlargement of the marked part is shown in Fig. 3.8.

Example 3.10 A natural area of application of a distance-based $\boldsymbol{P}$ is given by problems the anisotropies of which (if any) are mainly due to non-uniform mesh spacings, but for which the matrix itself does not reflect this accordingly. An important practical example where this happens are reaction-diffusion equations in which the underlying continuous diffusion operator is isotropic but "disturbed" or "superposed" by reaction equations. Such a situation is posed by the problem class defined in 3.4, being a model for the semiconductor reaction-diffusion applications discussed in Section 5.2.2. We have seen that the disturbances caused by the reaction terms are in principle removed by the smoothing operator chosen (for instance, block

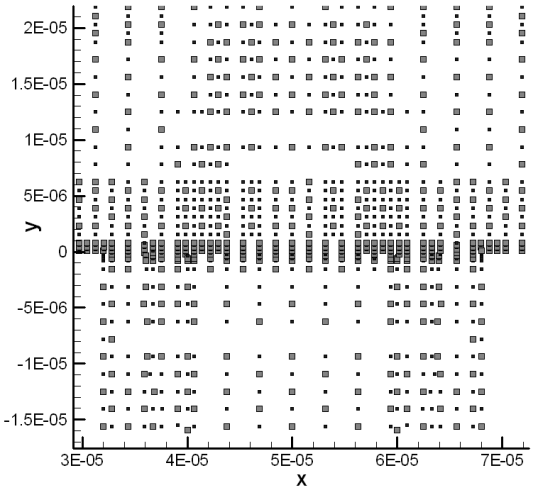


Figure 3.8: Exemplary point-coarsening with a distance-based $\boldsymbol{P}$. Shown are the finest (small) and the next coarse level (large box) for the part of the grid marked in Fig. 3.7.

Gauss-Seidel), see Section 3.4.1. Hence, a coarsening oriented on node-distances tackles the diffusion problem which essentially remains to be solved [48]. In contrast to this, the efficiency of an approach which makes only use of matrix entries in order to define coarsening and interpolation would be destroyed by large reaction terms. ▲

3.4.3 Interpolation Strategies for Point-Based Approaches

Given a fine and a coarse level - the latter obtained by one of the point-coarsening strategies - one has many possibilities to define interpolation now. Which of them may be a good choice in practice depends, as usually, on the application, the interplay between smoothing and coarsening, and computational costs.

For ease of description, we simply assume the interpolation operator, I_{FC}, to be ordered point-wise, that is

$$I_{FC} = (W_{(k,l)}) \tag{3.83}$$

with "weights" $W_{(k,l)}$ with $k \in F^p$ and $l \in P_k^p$ where P_k^p are the **sets of interpolatory points**. Each P_k^p contains all the C-points the F-point $\mathcal{P}_k$ is interpolating from. Hence, it is a subset of C^p and, in case of direct interpolation, a subset of $N_k^p \cap C^p$. We only consider *direct* interpolation here, the corresponding *indirect* schemes are obtained as described in Sections 3.2.3.5 and 4.3.

[48] An "ideal" approach for such a PDE system would be a splitting of the corresponding matrix into its "diffusive" and its "reactive" part. The "diffusive" part could then directly be used for coarsening instead of an approximation which can "a-posteriori" obtained by a distance-based $\boldsymbol{P}$. In practice, however, such a splitting is usually not available.

Analogously to the fact that, in a variable-based approach, an interpolation formula shall approximate (3.22) for an algebraic smooth error, we here seek suitable $W_{(k,l)}$ for

$$e_{(k)} = \sum_{l \in P_k^p} W_{(k,l)} e_{(l)} \tag{3.84}$$

so that (3.84) is an approximation of the equation

$$A_{(k,k)} e_{(k)} + \sum_{l \neq k} A_{(k,l)} e_{(l)} = 0 \ . \tag{3.85}$$

This equation is equivalent to (3.22) and characterizes an algebraically smooth error as created by a smoother obeying both the smoothing property (3.9) as well as the point-smoothing property (3.71).

For each application, it is important to consider the following questions prior to a decision on the *type of interpolation* finally employed: Is it important to take *unknown cross-couplings* into account? *If not*, that is if each unknown should be interpolated only from variables corresponding to the same unknown, do we need to consider each unknown separately, or is it sufficient to use the same formulas for all unknowns? *If yes*, shall the interpolation be block-wise or not?

The combination "take unknown cross-couplings into account and do not interpolate block-wise" does not fit to a point-based strategy but suggests a variable-based approach instead. Hence, we consider the following three **general types of interpolation** in our framework for point-based approaches:

- **block-interpolation (B-interpolation)**: couplings between different unknowns are taken into account, and the interpolation formulas are computed block-wise.
- **multiple-unknown-interpolation (MU-interpolation)**: the interpolation formulas are computed variable-wise and separately for each unknown; in particular, F-variables are only interpolated from variables of the same unknown.
- **a single-unknown-interpolation (SU-interpolation)**: the interpolation formulas are computed variable-wise, but are "identical" for the variables belonging to the same point, and F-variables are only interpolated from variables of the same unknown.

In order to completely define any of the above interpolation operators, we need to decide on its **pattern**, i.e. the nonzero structure of the interpolation matrix I_{FC}, and then compute the concrete **weights**.

In Sections 3.4.3.1-3.4.3.3, we describe details of the different interpolation processes, assuming a fine and a coarse level to be given. The coarse level is assumed to be the result of a point-coarsening process with some primary matrix $\boldsymbol{P}$. However, as for all AMG approaches, it is important to stress that the construction of coarsening and interpolation are closely related to each other. That is, coarsening and interpolation must "fit together". In particular, the pattern of I_{FC} should be related in a "natural" way to the pattern of $\boldsymbol{P}$ since the definition of the C/F-splitting has been based on the primary matrix $\boldsymbol{P}$.

In Section 3.4.3.4, we discuss the choice of interpolation for the RD and DD models and corresponding applications considered in Section 5.

Remark 3.28 In Sections 3.4.3.1 and 3.4.3.2, we only consider weights based on entries of A. In all interpolations, however, weights can be based on a part of A, on $\boldsymbol{P}$, or on coordinates. The technical possibilities for each interpolation type are discussed in Section 4.3.2. Again, note that this choice has to fit to the remaining components chosen. ▲

Recall from above that we assume $P_k^p \subseteq N_k^p \cap C^p$. We make use of the following definitions being generalizations of the ones used for the discussion of variable-based approaches:

$$N_k^p := \{l \in \mathcal{V}^p \mid l \neq k \wedge A_{(k,l)} \neq 0\} \ , \tag{3.86}$$

$$N_k^{p,+} := \{l \in N_k^p \mid A_{(k,l)} \geq 0\} \ , \qquad P_k^{p,+} := N_k^{p,+} \cap P_k^p \ , \tag{3.87}$$

$$N_k^{p,-} := \{l \in N_k^p \mid (-A_{(k,l)}) \geq 0\} \ , \qquad P_k^{p,-} := N_k^{p,-} \cap P_k^p \ , \tag{3.88}$$

$$N_k^{p,0} := N_k^p \setminus (N_k^{p,-} \cup N_k^{p,+}) \ , \qquad P_k^{p,0} := N_k^{p,0} \cap P_k^p \ . \tag{3.89}$$

3.4.3.1 Block-Interpolation (B-Interpolation)

Generally speaking, with block-interpolation we mean any interpolation obtained by a block-wise approximation of (3.85) in a way which is analogous to the classical "scalar" approaches to define interpolation. Unknown cross-couplings are taken into account here. At first sight, the use of a block-interpolation seems most natural, especially if $\boldsymbol{P}$ is defined based on norms, that is, according to one of (3.72)-(3.75).

Variant (1) One way to approximate (3.85) and a straightforward analog of (3.25) is

$$\Big(\sum_{j\in\widehat{N}^p} A_{(k,j)}\Big)^{-1}\Big(\sum_{j\in\widehat{N}^p} A_{(k,j)}e_{(j)}\Big) \approx \Big(\sum_{j\in\widehat{P}^p} A_{(k,j)}\Big)^{-1}\Big(\sum_{j\in\widehat{P}^p} A_{(k,j)}e_{(j)}\Big) \tag{3.90}$$

with $\widehat{N}^p \subseteq N_k^p$ and $\widehat{P}^p := \widehat{N}^p \cap P_k^p$.

Remark 3.29 Note that (3.90) is well-defined only if the number of variables per point is constant (equal to n_u) and if the inverses exist. ▲

Applied to both $\widehat{N}^p = N_k^p \setminus N_k^{p,+}$ and $\widehat{N}^p = N_k^{p,+}$, we obtain the following analog of (3.36): for all $k \in F^p$ define

$$\begin{aligned}
&\forall\, l \in P_k^p \setminus P_k^{p,+} :\ W_{(k,l)} := -A_{(k,k)}^{-1}\Big(\sum_{j\in N_k^p\setminus N_k^{p,+}} A_{(k,j)}\Big)\Big(\sum_{j\in P_k^p\setminus P_k^{p,+}} A_{(k,j)}\Big)^{-1} A_{(k,l)} \ , \\
&\forall\, l \in P_k^{p,+} :\ W_{(k,l)} := -A_{(k,k)}^{-1}\Big(\sum_{j\in N_k^{p,+}} A_{(k,j)}\Big)\Big(\sum_{j\in P_k^{p,+}} A_{(k,j)}\Big)^{-1} A_{(k,l)} \ ,
\end{aligned} \tag{3.91}$$

provided again that the number of variables per point is constant and the inverses exist. For $A > 0$, the positive-definiteness of all $A_{(k,k)}$ is ensured so that they are invertible, in particular. Unfortunately, the $\sum_{j\in\widehat{N}^p} A_{(k,j)}$ cannot be expected to be invertible in general.

In addition, it is not clear how to handle indefinite $A_{(k,j)}$. Both gives rise to problems, if not anyway for the efficiency of the approach, at least for its implementation since inverting (numerically) singular matrices has to be avoided. Moreover, this type of interpolation is very expensive because of the necessary inversions and eigenvalue calculations, and it is inflexible according to Remark 3.29.

Remark 3.30 With I denoting the identity operator, observe that

$$I - \sum_{l \in P_k^p} W_{(k,l)} = A_{(k,k)}^{-1} S_{(k)} \quad \text{with} \quad S_{(k)} := A_{(k,k)} + \sum_{l \in N_k^p} A_{(k,l)} \tag{3.92}$$

Therefore, if all $S_{(k)} = 0$, each block-row sum of the interpolation operator equals the identity operator I which means that constant vectors are interpolated exactly. ▲

Remark 3.31 (Straightforward point-based extensions applied to linear elasticity)
The straightforward variant (1) as well as a straightforward generalization of the basic interpolation formula of [71] have been considered in [58, 30]. Results of tests with anisotropic Laplacians as well as linear elasticity problems have been reported there. They confirm that the problems indicated in the discussion of variant (1) above are likely to arise even for rather simple model problems on uniform grids. Depending on the coarsening and concrete matrix, one or both of the variants discussed in [58, 30] fail because required inversions cannot be performed.

In [58, 30], a kernel-preserving property of variant (1) has been proved for linear elasticity problems with homogeneous Neumann boundary conditions on simple regular meshes. Unfortunately, this property can hardly be exploited in more complex geometric situations (unstructured grids).

Recently, similar straightforward approaches have been suggested in [32]. Whereas [58, 30] advocate straightforward generalizations of the interpolation schemes discussed in [87], an interpolation based on the "old" version [71] is used in [32]. In addition, simple aggregation-like components are discussed there. These investigations are still at the beginning.

A straightforward **point-based variant of smoothed aggregative AMG** is proposed in [99] for linear elasticity. Scalar operations are simply replaced by block-counterparts (among them spectral radii of certain matrix products). For a set of "real-life" linear elasticity problems, the variant which takes all rigid body modes into account has been shown to be more robust and faster than the variant which takes only the translations into account. Recently, [62] has demonstrated the efficiency of a point-wise smoothed aggregative AMG approach. The advocated method corresponds to [99, 53] with some modifications (local approximation estimators [52]). ▲

Variant (2) (3.91) can be simplified in a straightforward way if we do not distinguish between the two classes of point-coupling matrices, namely $A_{(k,l)} \geq 0$ and $A_{(k,l)} \not\geq 0$. This leads to

$$\forall\, l \in P_k^p \;:\; W_{(k,l)} := -A_{(k,k)}^{-1} \Big(\sum_{j \in N_k^p} A_{(k,j)} \Big) \Big(\sum_{j \in P_k^p} A_{(k,j)} \Big)^{-1} A_{(k,l)}\,, \tag{3.93}$$

for all $k \in F^p$ (i.e. (3.90) applied to N_k^p) - again provided that the number of variables per point is constant and that the inverses exist (see Remark 3.29). Also here, inverses have to be calculated, but eigenvalue computations are not necessary which considerably decreases the computational cost for the setup of the interpolation weights. As above,

$$I - \sum_{l \in P_k^p} W_{(k,l)} = A_{(k,k)}^{-1} S_{(k)} \,. \tag{3.94}$$

Variant (3) A less critical analog of (3.25) emerges if we replace the inverses in (3.90) by diagonal matrices the i-th diagonal of which consists of the inverse of the sum of the entries of the i-th row of all $A_{(k,j)}$ with $j \in \widehat{N}^p$ or $j \in \widehat{P}^p$, respectively. To be more specific, define

$$\widehat{N} := \{j \in \mathcal{V} \mid \exists k \in \widehat{N}^p \,:\, j \in \mathcal{P}_k\} \quad , \quad \widehat{P} := \{j \in \mathcal{V} \mid \exists k \in \widehat{P}^p \,:\, j \in \mathcal{P}_k\} \,. \tag{3.95}$$

Then replace (3.90) by

$$R_{\widehat{N},k}^{-1} \Big(\sum_{j \in \widehat{N}^p} A_{(k,j)} e_{(j)} \Big) \quad \approx \quad R_{\widehat{P},k}^{-1} \Big(\sum_{j \in \widehat{P}^p} A_{(k,j)} e_{(j)} \Big) \tag{3.96}$$

with $R_{\widehat{N},k} = (r_{\widehat{N},kij})_{i,j}$ and $R_{\widehat{P},k} = (r_{\widehat{P},kij})_{i,j}$ being $|\mathcal{P}_k| \times |\mathcal{P}_k|$-matrices with all off-diagonal entries being zero and, for each $i \in \mathcal{P}_k$,

$$r_{\widehat{N},kii} := \sum_{j \in \widehat{N}} a_{ij} \quad \text{and} \quad r_{\widehat{P},kii} := \sum_{j \in \widehat{P}} a_{ij} \,. \tag{3.97}$$

If one of the $r_{\widehat{N},kii}$ or $r_{\widehat{P},kii}$ is zero, it is replaced by 1.

As above, the inverse matrices serve as scaling factors, but here the factors are chosen so that the row sums of the matrices[49] $\sum_{j \in \widehat{N}^p} A_{(k,j)}$ and $\sum_{j \in \widehat{P}^p} A_{(k,j)}$ - if not equal to zero - are scaled to one. In contrast to the above variants, these inverses always exist due to the modification mentioned above. Moreover, the emerging interpolation formulas are cheaper to evaluate than (3.93). Another great practical advantage of this variant is that it can easily handle the case of varying number of variables per point. Applied to $\widehat{N}^p = N_k^p$, we obtain

$$\forall\, l \in P_k^{p,+} \,:\, W_{(k,l)} := -A_{(k,k)}^{-1} R_{N_k^p} R_{P_k^p}^{-1} A_{(k,l)} \,. \tag{3.98}$$

Note that, in contrast to variants (1) and (2), $I - \sum_{l \in P_k^p} W_{(k,l)}$ is in general not equal to one if all $S_{(k)} = 0$.

In practice, even this scheme is very expensive so that simpler types of interpolation often lead to more efficient AMG processes. Thus, besides the above block-interpolations, we consider variable-wise defined interpolation formulas which make use of multiple unknowns (Section 3.4.3.2) or a single unknown (Section 3.4.3.3). These approaches are described in the following two sections.

[49] In the following two terms, the $A_{(k,j)}$ are suitably extended and considered as being $|\mathcal{P}_k| \times |n_u|$-matrices.

3.4.3.2 Multiple-Unknown-Interpolation (MU-Interpolation)

A multiple-unknown-interpolation is formally identical to the interpolation used in UAMG. The only difference lies in the level hierarchy, that is the C/F-splitting, the interpolatory sets P_i are based on. Whereas in UAMG the level hierarchies are computed separately for the different unknowns, they are (principally) identical in PAMG. In the last case, it is depending on the concrete matrix considered whether the computed C/F-splitting fits to the $A_{[n,n]}$.

An MU-interpolation may be used in cases where only the coarsening should be computed point-wise. The employment of this type of interpolation then leads to an AMG approach which can be regarded as a compromise between an unknown-based and a point-based one.

An application for which a special MU-interpolation works quite efficiently is given by the reaction-diffusion equations considered in Section 5.2.2. This will be explained in Example 3.11 below. Also for DD models and the drift-diffusion matrices considered in Section 5.3.2, an MU-interpolation has advantages depending on the concrete application. See Example 3.12 below.

3.4.3.3 Single-Unknown-Interpolation (SU-Interpolation)

We speak of a single-unknown-interpolation if an interpolation is computed for the set of points $\mathcal{V}^p$ and then transferred point-wise to the variables $\mathcal{V}$ of the target system $Av = b$ such that the interpolation formulas are (essentially) the same for all unknowns.

To compute an interpolation for the set of points, a variable-based interpolation scheme (see Section 3.2.3) is applied to the primary matrix $\boldsymbol{P}$. Recall that we can choose between interpolation weights based on entries of the matrix - here $\boldsymbol{P}$ - or coordinates, depending on the application. In addition, another possibility is offered for SU-interpolations. If an unknown-pattern has been chosen with $\mathcal{U}_n$ being the primary unknown, we might choose the entries of $A_{[n,n]}$ as basis for computing the interpolation weights.

Being equivalent to choosing a specific kind of interpolation weights in computing the interpolation formulas for the points, we can directly apply a "classical" VAMG interpolation to an $(n_p \times n_p)$-matrix $\widetilde{\boldsymbol{P}} = (\widetilde{p}_{kl})_{k,l=1,\ldots,n_p}$ with $\Sigma(\widetilde{\boldsymbol{P}}) \subseteq \Sigma(\boldsymbol{P})$. Depending on the interpolation weights chosen, this matrix then equals $\boldsymbol{P}$ or $A_{[n,n]}$ or is coordinates-based [50]. We call this a **$\widetilde{\boldsymbol{P}}$-interpolation** in the following. Recall that we consider only *direct* interpolation here, i.e. $P_k^p \subseteq N_k^p \cap C^p$. Analogously to (3.86), we define:

$$\widetilde{N}_k^p := \{l \in \mathcal{V}^p \mid l \neq k \wedge \widetilde{p}_{kl} \neq 0\}, \quad \widetilde{P}_k^p := \widetilde{N}_k^p \cap P_k^p, \tag{3.99}$$

$$\widetilde{N}_k^{p,-} := \{l \in \widetilde{N}_k^p \mid \widetilde{p}_{kl} < 0\}, \qquad \widetilde{P}_k^{p,-} := \widetilde{N}_k^{p,-} \cap P_k^p, \tag{3.100}$$

$$\widetilde{N}_k^{p,+} := \{l \in \widetilde{N}_k^p \mid \widetilde{p}_{kl} > 0\}, \qquad \widetilde{P}_k^{p,+} := \widetilde{N}_k^{p,+} \cap P_k^p. \tag{3.101}$$

Note that $\widetilde{N}_k^p$ can be different from N_k^p. Analogously for $\widetilde{N}_k^{p,+}$ and $\widetilde{N}_k^{p,-}$. Observe that $\widetilde{P}_k^p = P_k^p \cap \widetilde{N}_k^p = \widetilde{P}_k^{p,-} \dot{\cup} \widetilde{P}_k^{p,+}$. Furthermore, since $\widetilde{\boldsymbol{P}}$ has a subset of or even the same connectivity pattern of $\boldsymbol{P}$, $\widetilde{N}_k^p \subseteq N_k^p$ holds. This means that only those $\widetilde{p}_{kl}$ can be nonzero whose corresponding $A_{(k,l)}$ are nonzero.

[50] see also Section 4.3.2.

Since the interpolation formulas are constructed with the help of a variable-based approach, the following formulas emerge for direct interpolation: $\forall\, k \in F^p$

$$\begin{aligned} \forall\, l \in \widetilde{P}_k^{p,-}: \quad & w_{kl}^p := -\frac{\widetilde{p}_{kl}}{\widetilde{p}_{kk}}\alpha_k^p > 0 \quad \text{with} \quad \alpha_k^p := \frac{\sum_{j\in\widetilde{N}_k^{p,-}}\widetilde{p}_{kj}}{\sum_{j\in\widetilde{P}_k^{p,-}}\widetilde{p}_{kj}} > 0 \\ \forall\, l \in \widetilde{P}_k^{p,+}: \quad & w_{kl}^p := -\frac{\widetilde{p}_{kl}}{\widetilde{p}_{kk}}\beta_k^p < 0 \quad \text{with} \quad \beta_k^p := \frac{\sum_{j\in\widetilde{N}_k^{p,+}}\widetilde{p}_{kj}}{\sum_{j\in\widetilde{P}_k^{p,+}}\widetilde{p}_{kj}} > 0 \end{aligned} \tag{3.102}$$

Remark 3.32 A ***P*-interpolation** is defined to be an SU-interpolation fully[51] based on the primary matrix $\boldsymbol{P}$. Analogously, an ***A*$_{[n,n]}$-interpolation** is defined. For the latter, note that the chosen $\boldsymbol{P}$ and $A_{[n,n]}$ can be different. ▲

Transfer of the $\widetilde{P}$-Interpolation For the SU-interpolation - an interpolation which is "the same"for all unknowns - the definition of the interpolation formulas for the variables of a point $\mathcal{P}_k$ corresponds to the transfer of the interpolation weights w_{kl}^p ($l \in P_k^p$) to the "diagonal elements"of $W_{(k,l)}$. That means, they are transferred to the w_{ij} with $i \in \mathcal{P}_k$ and $j \in \mathcal{P}_l$ such that i and j belong to the same unknown - as long as such variable index pairs (i, j) exist. We have to distinguish between two cases here:

- **Case 1:** Each (nonempty) point is attached to all unknowns.
- **Case 2:** There are (nonempty) points which are not attached to all unknowns.

In the first, "ideal"case, two nonempty points $\mathcal{P}_k$ and $\mathcal{P}_l$ are always attached to the same set of unknowns. Hence, each submatrix $W_{(k,l)}$ is square and obtains the following form:

$$W_{(k,l)} = \begin{bmatrix} w_{kl}^p & & 0 \\ & \ddots & \\ 0 & & w_{kl}^p \end{bmatrix}. \tag{3.103}$$

In the second case, not all unknowns are living on the whole domain. Two algorithms are implemented in SAMG for the definition of the $W_{(k,l)}$ in this case. A more detailed, technical discussion is postponed to Section 4.3.2.2. However, we want to indicate the probably crucial point here: the occurrence of an *unknown-overlapping interpolation* in the neighborhood of the "interface" of regions with different kind and/or number of variables. Usually, the impact of such overlaps on the convergence is small. However, for the case that they do disturb, we want to mention that both of the two algorithms mentioned above provide the possibility to "skip" unknown cross-couplings. In Example 3.13 below, we will illustrate a typical resulting interpolation structure for case (2) and the occurrence of an unknown-overlapping interpolation.

[51] That is, the interpolation for the variables of A is based on the SW-pattern (see Section 4.2.1) and the entries p_{kl} of $\boldsymbol{P}$.

Remark 3.33 Another possibility to base the interpolation weights on $\boldsymbol{P}_{max}$ but avoid possible problems with unknown cross-interpolations (as indicated above) is the employment of an MU-interpolation. Here, instead of the entries of the $A_{[n,n]}$, the corresponding entries of $\boldsymbol{P}_{max}$ are used to compute the *weights* of I_{FC}. ▲

3.4.3.4 Examples

In the following, we discuss the choice of interpolation for the RD and DD models and corresponding applications considered in Section 5.

Example 3.11 (RD models and semiconductor reaction-diffusion equations)
For the semiconductor reaction-diffusion equations considered in Section 5.2.2, a special MU-interpolation works more efficiently than the other two interpolation strategies, that is block- or SU-interpolations. Heuristically, this might be explained as follows.

We have already seen in Section 3.4.1.2 and in Example 3.10 that BGS smoothing handles the unknown cross-couplings of the RD models appropriately and that a coordinates-based $\boldsymbol{P}$ mimicing a discrete Laplacian allows for an appropriate coarsening. However, in the RD models, unknown cross-couplings can only be found in the diagonal blocks $A_{(k,k)}$ whereas in general cases unknown cross-couplings can also be located somewhere else. Indeed, this is the case for the matrices considered in Section 5.2.2.

BGS smoothing is not always able to "remove" the influence of all large couplings which correspond to reaction terms and are located outside the diagonal blocks $A_{(k,k)}$ so that the error might locally not be smooth in all directions. Hence, an SU-interpolation which is "identical" for all unknowns [52] has numerically proved to be not robust. This is also the case for block-interpolation.

However, an MU-interpolation with weights being based on coordinates[53] is often able to address the described "local problems" of BGS smoothing. Strong couplings in the $A_{[n,n]}$ which are due to reaction terms and not due to the underlying diffusion operator are also reflected, and the choice of interpolatory variables is adapted to them. However, the magnitude of the couplings is determined by distances which prevents the destruction of interpolation by exceptionally large reaction terms. The resulting interpolation thus resembles the interpolation for a diffusion operator but on the pattern of strong connectivity as reflected by both the diffusive and the reactive terms. By construction, (at least) constant functions are interpolated exactly here.

It should be noted that the resulting point-based approach might not always converge stand-alone. This is mostly due to large unknown cross-couplings which are neither correctly treated by smoothing nor interpolation. In such cases, the described PAMG approach often profits substantially from acceleration by BiCGstab or GMRes. Concrete numerical results for semiconductor reaction-diffusion systems will be discussed in Section 5.2.2. ▲

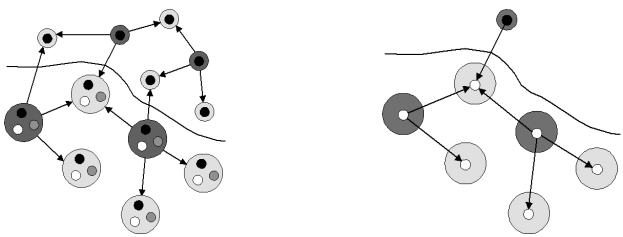
Example 3.12 (DD models and semiconductor drift-diffusion equations)
For the DD models, both MU- and SU-interpolation can have advantages and disadvantages, depending on the concrete problem (for numerical results, see Section 4.6). We know that the

[52] and can be based on coordinates or matrix entries here, see also Section 4.3.2.
[53] see also Section 4.3.2.

system's smoothing behavior is controlled by two influences. In particular, the larger c, the stronger it is dominated by $A_{[2,1]}$ (see Fig. 3.1). In addition, a certain isotropic part plays a role which is the larger the smaller c is (see Fig. 3.2). In Example 3.5, we have explained why coarsening based on a norm-based $\boldsymbol{P}$ can be expected to be most appropriate here. Hence, interpolation should also be based on this $\boldsymbol{P}$ as long as smoothing and coarsening go hand in hand - simultaneously for all three unknowns.

However, as indicated by Figs. 3.2 (a) and (b), for small ϵ and $c \approx 1$, $\boldsymbol{P}$ does not fully reflect the local direction(s) of smoothness simultaneously for all unknowns. Hence, the more different smooth error looks like for the unknowns, the less efficient an SU-interpolation will be. We will see in Section 4.6 that indeed the SU-interpolation suffers - but only slightly - from small ϵ together with $c \approx 1$. If not all error frequencies are sufficiently reduced, using the resulting PAMG approach as a preconditioner for BiCGstab or GMRes considerably improves convergence and robustness in many cases and has turned out to be a must for practical applications here: As will be explained in Section 5.3.2, the situation for semiconductor drift-diffusion matrices is comparable to the case (λ, c)=(1,1) with anisotropies. ▲

Example 3.13 (Unknown-overlapping interpolation)
Figures 3.9(a) and (b) show an example of the construction of an SU-interpolation as can happen in device simulation in a similar manner (see Section 5.3). In this example, the domain Ω consists of two parts (the interface is indicated by a line): only in one part, the system consists of all three unknowns. Fig. 3.9(b) depicts the resulting structure of interpolation: One variable of the grey unknown does also interpolate from a black one, a variable of a different unknown. In general, if not all unknowns are living in the whole domain, the occurrence of an *unknown-overlapping interpolation* in the neighborhood of the boundary is very probable.



(a) (b)

Figure 3.9: (a) Exemplary $\widetilde{\boldsymbol{P}}$-interpolation structure. C-points are dark grey, F-points light grey. (b) Resulting interpolation structure for the grey unknown (analogous for the white unknown) after transfer. Here, one variable of the white (grey) unknown does not only interpolate from variables of the same unknown but also from a black one.

3.4.4 Two-Level Convergence Analysis

In the following, we generalize the two-level convergence theory for variable-based and unknown-based AMG to point-based approaches. In addition, the proofs of theorems for

the variable-based case are given, as far as still pending. To be more specific, generalizations of the τ-condition (3.21) and of Theorem 3.7 are developed. For the proofs, strong conditions must be fulfilled, in particular, all unknowns must be attached to all points, A must be symmetric positive definite, and the off-diagonal point matrices must at least be symmetric. These assumptions are generally made in the following.

The obtained estimates will turn out to be at least structurally similar to the ones of Theorem 3.7. But, as one might expect, the condition of the point matrices will come into, for example. The general statements will then be transferred to or itself generalized for concrete cases, as for instance the block- and $\boldsymbol{P}$-interpolations, which will allow for comparing the properties of both interpolation types further.

As for VAMG and UAMG, very rough upper bounds for two-level convergence will emerge which should not be used for quantitative assessments of specific approaches. Their main importance lies in the fact that they provide valuable insights into what influences the convergence qualitatively.

Throughout this Section, we only consider *direct* interpolations, i.e. $P_k^p \subseteq N_k^p \cap C^p$.

3.4.4.1 τ-Conditions

We proceed in a way similar to the proof of Theorem 3.7, as performed in [87]. Completely analogously to the proof of Theorem 3.3 we obtain

Theorem 3.11 *Let $A > 0$ and let S satisfy the point-smoothing property* (3.71)*. Furthermore, assume the C^p/F^p-splitting and interpolation to be such that*

$$\forall\, e \;:\; ||Ke||_1^2 \leq \tau ||Ke||_{P,2}^2 \tag{3.104}$$

with some $\tau > 0$ being independent of e. Then $\tau \geq \sigma$ and $||SK||_1 \leq \sqrt{1-\sigma/\tau}$.

The next theorem (the direct analog of Theorem 3.4) provides us with a sufficient condition for (3.104).

Theorem 3.12 *If $A > 0$ and the C^p/F^p-splitting and interpolation I_{FC} are such that*

$$\forall\, e \in \mathcal{R}(K) \;:\; ||e_F - I_{FC} e_C||_{P,0,F}^2 \leq \tau ||e||_1^2 \tag{3.105}$$

with τ being independent of e, then (3.104) *is satisfied.*

The next step is the generalization of some lemmata to the point-based case. We start with a variant of Cauchy-Schwarz's inequality.

Lemma 3.9 *Let $i, n \in \mathbb{N}$, v_l $(l = 1, \ldots, i)$ be vectors in $\mathbb{R}^n$, W_l $(l = 1, \ldots, i)$ $(n \times n)$-matrices, $||\cdot||$ for vectors a norm and for matrices the operator norm compatible to this vector norm, and $\mu := \sum_{l=1}^{i} ||W_l||$. Then the following inequality holds:*

$$\left|\left| \sum_{l=1}^{i} W_l v_l \right|\right|^2 \leq \mu \sum_{l=1}^{i} ||W_l|| \cdot ||v_l||^2 \,. \tag{3.106}$$

The proof of this lemma can be found in Section A.2 of the appendix.
The next lemma provides a splitting of $(Ae,e)_E$ under certain assumptions. In particular, the off-diagonal point-coupling matrices, $A_{(k,l)}$ $(k \neq l)$, have to be symmetric. Define a generalization of the t_i to the point-case:

$$T_{(k)} = A_{(k,k)} + \sum_{l \in N_k^p \setminus N_k^{p,+}} A_{(k,l)} - \sum_{l \in N_k^{p,+}} A_{(k,l)} \ . \tag{3.107}$$

Lemma 3.10 (a) *Let A be symmetric, and $N_k^p = N_k^{p,+} \cup N_k^{p,-}$ for all $k \in \mathcal{V}^p$. Then the following holds:*

$$\begin{aligned}(Ae,e)_E =& \frac{1}{2} \sum_{k,l,l \in N_k^{p,-}} (-A_{(k,l)}(e_{(k)} - e_{(l)}), e_{(k)} - e_{(l)})_E \\ &+ \frac{1}{2} \sum_{k,l,l \in N_k^{p,+}} (A_{(k,l)}(e_{(k)} + e_{(l)}), e_{(k)} + e_{(l)})_E + \sum_k (T_{(k)} e_{(k)}, e_{(k)})_E \, .\end{aligned}$$

(b) *Let A be of* **essentially block-positive type**, *see* (2.34). *Recall $S_{(k)} = \sum_l A_{(k,l)}$ for all $k \in \mathcal{V}^p$. Then the following holds:*

$$(Ae,e)_E \geq \frac{c}{2} \sum_{k,l,l \in N_k^{p,-}} (-A_{(k,l)}(e_{(k)} - e_{(l)}), e_{(k)} - e_{(l)})_E + \sum_k (S_{(k)} e_{(k)}, e_{(k)})_E \, .$$

Proof of (a). The assumption $N_k^p = N_k^{p,+} \cup N_k^{p,-}$ implies that each $A_{(k,l)}$ $(k \neq l)$ is either ≥ 0 or ≤ 0 and in particular symmetric. Since A is symmetric, too, we have $A_{(l,k)} = A_{(k,l)}^T = A_{(k,l)}$, and

$$\begin{aligned}&\sum_{k,l,l \in N_k^{p,-}} (-A_{(k,l)}(e_{(k)} - e_{(l)}), e_{(k)} - e_{(l)})_E \\ &+ \sum_{k,l,l \in N_k^{p,+}} (A_{(k,l)}(e_{(k)} + e_{(l)}), e_{(k)} + e_{(l)})_E \\ = \ & -2 \sum_{k,l,l \in N_k^{p,-}} (A_{(k,l)} e_{(k)}, e_{(k)})_E + 2 \sum_{k,l,l \in N_k^{p,-}} (A_{(k,l)} e_{(l)}, e_{(k)})_E \\ &+2 \sum_{k,l,l \in N_k^{p,+}} (A_{(k,l)} e_{(k)}, e_{(k)})_E + 2 \sum_{k,l,l \in N_k^{p,+}} (A_{(k,l)} e_{(l)}, e_{(k)})_E \\ = \ & 2(Ae,e)_E - 2 \sum_k (A_{(k,k)} e_{(k)}, e_{(k)})_E \\ &-2 \sum_k \sum_{l \in N_k^{p,-}} (A_{(k,l)} e_{(k)}, e_{(k)})_E + 2 \sum_k \sum_{l \in N_k^{p,+}} (A_{(k,l)} e_{(k)}, e_{(k)})_E \\ = \ & 2(Ae,e)_E - 2 \sum_k (T_{(k)} e_{(k)}, e_{(k)})_E\end{aligned}$$

which proves (a).

Proof of (b). We make use of the definition (2.31) and Lemma 2.1 and proceed analogously to (a):

$$\begin{aligned} & c \sum_{k,l,l\in N_k^{p,-}} (-A_{(k,l)}(e_{(k)}-e_{(l)}),e_{(k)}-e_{(l)})_E \\ \leq & \sum_{k,l,k\neq l} (-A_{(k,l)}(e_{(k)}-e_{(l)}),e_{(k)}-e_{(l)})_E \\ = & \; 2\sum_{k,l,k\neq l} (A_{(k,l)}e_{(k)},e_{(l)})_E + 2\sum_{k,l,k\neq l} (-A_{(k,l)}e_{(k)},e_{(k)})_E \\ = & \; 2(Ae,e)_E - 2\sum_k (S_{(k)}e_{(k)},e_{(k)})_E \;\; . \end{aligned}$$

This proves (b). ■

The following lemma gives us lower bounds for $(Ae,e)_E$ in terms of the $A_{(k,l)}$ under certain assumptions. These will be needed in the proof of Theorem 3.13 later on.

Lemma 3.11 (a) *Let A be symmetric, $N_k^p = N_k^{p,+} \cup N_k^{p,-}$ and $T_{(k)} \geq 0$ for all $k \in \mathcal{V}^p$. Then*

$$\begin{aligned} (Ae,e)_E \geq & \sum_{k\in F^p} \sum_{l\in P_k^{p,-}} (-A_{(k,l)}(e_{(k)}-e_{(l)}),e_{(k)}-e_{(l)})_E \\ & \sum_{k\in F^p} \sum_{l\in P_k^{p,+}} (A_{(k,l)}(e_{(k)}+e_{(l)}),e_{(k)}+e_{(l)})_E + \sum_{k\in F^p} (T_{(k)}e_{(k)},e_{(k)})_E \,. \end{aligned}$$

(b) *Let A be of essentially block-positive type and $S_{(k)} \geq 0$ for all $k \in \mathcal{V}^p$. Then*

$$(Ae,e)_E \geq c \sum_{k\in F^p} \sum_{l\in P_k^{p,-}} (-A_{(k,l)}(e_{(k)}-e_{(l)}),e_{(k)}-e_{(l)})_E + \sum_{k\in F^p} (S_{(k)}e_{(k)},e_{(k)})_E \,.$$

Proof of (a). Because of Lemma 2.1, Lemma 3.10, $\mathcal{V}^p = C^p \dot\cup F^p$, $P_k^{p,-} \subseteq C^p$ and $P_k^{p,+} \subseteq C^p$, we can estimate

$$\begin{aligned} (Ae,e)_E = & \frac{1}{2} \sum_{k,l\in\mathcal{V}^p, l\in N_k^{p,-}} (-A_{(k,l)}(e_{(k)}-e_{(l)}),e_{(k)}-e_{(l)})_E \\ & + \frac{1}{2} \sum_{k,l\in\mathcal{V}^p, l\in N_k^{p,+}} (A_{(k,l)}(e_{(k)}+e_{(l)}),e_{(k)}+e_{(l)})_E + \sum_{k\in\mathcal{V}^p} (T_{(k)}e_{(k)},e_{(k)})_E \\ \geq & \sum_{k\in F^p} \sum_{l\in P_k^{p,-}} (-A_{(k,l)}(e_{(k)}-e_{(l)}),e_{(k)}-e_{(l)})_E \\ & + \sum_{k\in F^p} \sum_{l\in P_k^{p,+}} (A_{(k,l)}(e_{(k)}+e_{(l)}),e_{(k)}+e_{(l)})_E + \sum_{k\in F^p} (T_{(k)}e_{(k)},e_{(k)})_E \,. \end{aligned}$$

The **proof of (b)** is analogous. ■

Now we can prove generalized versions of Theorems 3.7 and 3.5 for point-coupling matrices $A_{(k,l)}$ instead of matrix entries a_{ij}:

Theorem 3.13 (a) *Let $A > 0$, $N_k^p = N_k^{p,+} \cup N_k^{p,-}$, and $T_{(k)} \geq 0$ for all $k \in \mathcal{V}^p$. Select a C^p/F^p-splitting, a set P_k^p for each $k \in F^p$, and an interpolation I_{FC}. Define*

$$\mu_k = \sum_{l \in P_k^p} ||W_{(k,l)}||_E + \left|\left| I - \sum_{l \in P_k^{p,-}} W_{(k,l)} + \sum_{l \in P_k^{p,+}} W_{(k,l)} \right|\right|_E .$$

If for all $k \in F^p$ the inequalities

$$\tau\, \lambda_{\min}(T_{(k)}) \geq \mu_k ||A_{(k,k)}||_E \left|\left| I - \sum_{l \in P_k^{p,-}} W_{(k,l)} + \sum_{l \in P_k^{p,+}} W_{(k,l)} \right|\right|_E \tag{3.108}$$

$$\wedge \quad \forall\, l \in P_k^{p,-} \,:\, \tau\, \lambda_{\min}(-A_{(k,l)}) \geq \mu_k ||A_{(k,k)}||_E ||W_{(k,l)}||_E \tag{3.109}$$

$$\wedge \quad \forall\, l \in P_k^{p,+} \,:\, \tau\, \lambda_{\min}(A_{(k,l)}) \quad \geq \mu_k ||A_{(k,k)}||_E ||W_{(k,l)}||_E \tag{3.110}$$

hold with a $\tau \geq 1$ not depending on k, l, the τ-condition (3.105) *is fulfilled, i.e.*

$$||e_F - I_{FC} e_C||_{P,0,F}^2 \leq \tau ||e||_1^2 .$$

(b) *Let A be of essentially block-positive type, and $S_{(k)} \geq 0$ for all $k \in \mathcal{V}^p$. Select a C^p/F^p-splitting, a set P_k^p for each $k \in F^p$, and an interpolation I_{FC} with $P_k^p = P_k^{p,-}$ for all $k \in F^p$. Define*

$$\mu_k = \sum_{l \in P_k^p} ||W_{(k,l)}||_E + \left|\left| I - \sum_{l \in P_k^p} W_{(k,l)} \right|\right|_E .$$

If for all $k \in F^p$ the inequalities

$$\tau\, \lambda_{\min}(S_{(k)}) \geq \mu_k ||A_{(k,k)}||_E \left|\left| I - \sum_{l \in P_k^p} W_{(k,l)} \right|\right|_E \tag{3.111}$$

$$\wedge \quad \forall\, l \in P_k^p \,:\, \tau\, \lambda_{\min}(-A_{(k,l)}) \quad \geq \mu_k ||A_{(k,k)}||_E ||W_{(k,l)}||_E \tag{3.112}$$

$$\tag{3.113}$$

hold with a $\tau \geq 1$ not depending on k, l, the τ-condition (3.105) *is fulfilled with τ/c instead of τ, i.e.*

$$||e_F - I_{FC} e_C||_{P,0,F}^2 \leq \frac{\tau}{c} ||e||_1^2 .$$

Proof of (a). Let $(\cdot,\cdot)_{1:B}$ denote the energy inner product corresponding to a matrix $B > 0$, $||\cdot||_{1:B}$ the corresponding norm. Since $A > 0$, the submatrices $A_{(k,k)}$ are symmetric positive definite, too. Because of $N_k^p = N_k^{p,+} \cup N_k^{p,-}$, we have $P_k^p = P_k^{p,+} \cup P_k^{p,-}$ for all k. A straightforward calculation shows that

$$||e_F - I_{FC} e_C||_{P,0,F}^2$$

$$
\begin{aligned}
&= \sum_{k\in F^p} \left(A_{(k,k)} \left(e_{(k)} - \sum_{l\in P_k^p} W_{(k,l)} e_{(l)} \right),\ e_{(k)} - \sum_{l\in P_k^p} W_{(k,l)} e_{(l)} \right)_E \\
&= \sum_{k\in F^p} \Bigg|\Bigg| \sum_{l\in P_k^{p,-}} W_{(k,l)}(e_{(k)} - e_{(l)}) - \sum_{l\in P_k^{p,+}} W_{(k,l)}(e_{(k)} + e_{(l)}) \\
&\qquad\qquad + \left(I - \sum_{l\in P_k^{p,-}} W_{(k,l)} + \sum_{l\in P_k^{p,+}} W_{(k,l)} \right) e_{(k)} \Bigg|\Bigg|_{1:A_{(k,k)}} \\
&\leq \sum_{k\in F^p} ||A_{(k,k)}||_E \Bigg|\Bigg| \sum_{l\in P_k^{p,-}} W_{(k,l)}(e_{(k)} - e_{(l)}) + \sum_{l\in P_k^{p,+}} (-W_{(k,l)})(e_{(k)} + e_{(l)}) \\
&\qquad\qquad + \left(I - \sum_{l\in P_k^{p,-}} W_{(k,l)} + \sum_{l\in P_k^{p,+}} W_{(k,l)} \right) e_{(k)} \Bigg|\Bigg|_E^2 .
\end{aligned}
$$

Lemma 3.9 yields now

$$
\begin{aligned}
||e_F - I_{FC} e_C||_{P,0,F}^2 \ \leq \sum_{k\in F^p} ||A_{(k,k)}||_E\, \mu_k \Bigg(& \sum_{l\in P_k^{p,-}} ||W_{(k,l)}||_E ||e_{(k)} - e_{(l)}||_E^2 \\
& + \sum_{l\in P_k^{p,+}} ||W_{(k,l)}||_E ||e_{(k)} + e_{(l)}||_E^2 \\
& + \left|\left| I - \sum_{l\in P_k^{p,-}} W_{(k,l)} + \sum_{l\in P_k^{p,+}} W_{(k,l)} \right|\right|_E ||e_{(k)}||_E^2 \Bigg) .
\end{aligned}
$$

Because all $A_{(k,l)}$ and thus all $T_{(k)}$ are symmetric, their eigenvalues are real (cf. Lemma 2.1). Then, $\forall\ k \in F^p$, assumptions (3.108), (3.109) and (3.110) are equivalent to

$$
\begin{aligned}
& \tau\, T_{(k)} \geq \mu_k ||A_{(k,k)}||_E \quad \left|\left| I - \sum_{l\in P_k^{p,-}} W_{(k,l)} + \sum_{l\in P_k^{p,+}} W_{(k,l)} \right|\right|_E I \\
\wedge \quad & \forall\, l \in P_k^{p,-} \ :\ \tau\,(-A_{(k,l)}) \geq \mu_k ||A_{(k,k)}||_E ||W_{(k,l)}||_E I\,, \\
\wedge \quad & \forall\, l \in P_k^{p,+} \ :\ \tau\,(A_{(k,l)}) \quad \geq \mu_k ||A_{(k,k)}||_E ||W_{(k,l)}||_E I\,.
\end{aligned}
$$

Therefore, we can estimate

$$
\begin{aligned}
||e_F - I_{FC} e_C||_{P,0,F}^2 \leq & \sum_{k\in F^p} \sum_{l\in P_k^{p,-}} \tau(-A_{(k,l)}(e_{(k)} - e_{(l)}), e_{(k)} - e_{(l)})_E \\
& + \sum_{k\in F^p} \sum_{l\in P_k^{p,+}} \tau(A_{(k,l)}(e_{(k)} + e_{(l)}), e_{(k)} + e_{(l)})_E \\
& + \sum_{k\in F^p} \tau(T_{(k)} e_{(k)}, e_{(k)})_E\,.
\end{aligned}
$$

Finally, Lemma 3.11 shows that $||e_F - I_{FC}e_C||^2_{P,0,F} \leq \tau(Ae,e)_E = \tau||e||^2_1$, which proves (a). For the **proof of (b)** proceed analogously. ■

Remark 3.34 It is easy to see that $1 \leq \mu_k \leq 1 + 2\sum_{l\in P_k^p} ||W_{(k,l)}||_E$ holds. ▲

Remark 3.35 Because of $A > 0$, all $D_{(k,k)} > 0$. If we now assume the conditions of Theorem 3.13(a) to hold with each $A_{(k,k)}$ replaced by $D_{(k,k)}$, we obtain analogously that the τ-condition (3.21) is fullfilled, i.e. $||e_F - I_{FC}e_C||^2_{0,F} \leq \tau||e||^2_1$. Analogously for Theorem 3.13(b). ▲

Generally, Theorem 3.13(a) cannot be applied if there are $T_{(k)} \notin \mathcal{A}_{\text{spd}}$ or if there are point-coupling matrices, $A_{(k,l)}$, which are symmetric but neither positive nor negative semi-definite. Analogously for Theorem 3.13(b).

Remark 3.36 There are ways to generalize Theorem 3.13 such that the generalizations qualitatively correspond to Theorem 3.6 and similar theorems discussed in Section 3.2.3. The resulting qualitative statements say that a violation of the conditions of Theorem 3.13 leads to an enlarged τ. ▲

Even if all $T_{(k)} \leq 0$ (or $S_{(k)} \leq 0$) and all $A_{(k,l)}$ are semi-definite, problems can occur: if $A_{(k,l)}$ is singular, the inequalities in (3.109) and (3.110) are equivalent to $W_{(k,l)} = 0$, even if $A_{(k,l)} \neq 0$. Similarly, if $\lambda_{\min}(T_{(k)}) = 0$ (which means that $T_{(k)}$ is singular), (3.108) is only fulfilled if

$$I - \sum_{l\in P_k^{p,-}} W_{(k,l)} + \sum_{l\in P_k^{p,+}} W_{(k,l)} = 0 \ .$$

In case of $\lambda_{\min}(S_{(k)}) = 0$, the more natural condition $I - \sum_{l\in P_k^p} W_{(k,l)} = 0$ results which means that constants must be interpolated exactly. As for all interpolations we investigate in this thesis, this can be enforced.

3.4.4.2 Application to Different Interpolations

In the following sections, we discuss the application of Theorem 3.13 to various concrete interpolations to get an impression what factors determine the magnitude of τ.

Block-Interpolations For direct block-interpolations (3.91) and (3.93) the following corollary can be derived from Theorem 3.13:

Corollary 3.1 *Let $A > 0$, $N_k^p = N_k^{p,+} \cup N_k^{p,-}$ and $T_{(k)} \geq 0$ for all $k \in \mathcal{V}^p$. Select a C^p/F^p-splitting and a set P_k^p for each $k \in F^p$.*

(a) *If I_{FC} can be defined by* (3.91)*, then*

$$\mu_k = \sum_{l\in P_k^p} ||W_{(k,l)}||_E + ||A^{-1}_{(k,k)}T_{(k)}||_E$$

with $T_{(k)}$ defined by (3.107)*. If for all $k \in F^p$ the inequalities*

$$\tau\,\lambda_{\min}(T_{(k)}) \geq \mu_k \mathrm{cond}_E(A_{(k,k)})||T_{(k)}||_E \tag{3.114}$$

$$\wedge \quad \forall\, l \in P_k^{p,-} : \quad \tau\,\lambda_{\min}(-A_{(k,l)})$$

$$\geq \mu_k \mathrm{cond}_E(A_{(k,k)})\,||A_{(k,l)}||_E \rho\Bigg(\Bigg(\sum_{j\in N_k^{p,-}} A_{(k,j)}\Bigg)\Bigg(\sum_{j\in P_k^{p,-}} A_{(k,j)}\Bigg)^{-1}\Bigg), \tag{3.115}$$

$$\wedge \quad \forall\, l \in P_k^{p,+} : \quad \tau\,\lambda_{\min}(A_{(k,l)})$$

$$\geq \mu_k \mathrm{cond}_E(A_{(k,k)})\,||A_{(k,l)}||_E \rho\Bigg(\Bigg(\sum_{j\in N_k^{p,+}} A_{(k,j)}\Bigg)\Bigg(\sum_{j\in P_k^{p,+}} A_{(k,j)}\Bigg)^{-1}\Bigg), \tag{3.116}$$

$$\tag{3.117}$$

hold with a $\tau \geq 1$ not depending on k, l, the τ-condition (3.105) *is fulfilled.*

(b) *If I_{FC} can be defined by* (3.93)*, then*

$$\mu_k = \sum_{l\in P_k^p} ||W_{(k,l)}||_E + \Bigg|\Bigg|I + A_{(k,k)}^{-1}\Bigg(\sum_{j\in N_k^p} A_{(k,j)}\Bigg)\Phi_{(k)}\Bigg|\Bigg|_E$$

with

$$\Phi_{(k)} := \Bigg(\sum_{j\in P_k^p} A_{(k,j)}\Bigg)^{-1}\Bigg(\sum_{j\in P_k^{p,-}} A_{(k,j)} - \sum_{j\in P_k^{p,+}} A_{(k,j)}\Bigg).$$

If for all $k \in F^p$ the inequalities

$$\tau\,\lambda_{\min}(T_{(k)}) \geq \mu_k \mathrm{cond}_E(A_{(k,k)})\Bigg|\Bigg|A_{(k,k)} + \Bigg(\sum_{j\in N_k^p} A_{(k,j)}\Bigg)\Phi_{(k)}\Bigg|\Bigg|_E \tag{3.118}$$

$$\wedge \quad \forall\, l \in P_k^{p,-} :$$

$$\tau\,\lambda_{\min}(-A_{(k,l)}) \geq \mu_k \mathrm{cond}_E(A_{(k,k)})\,||A_{(k,l)}||_E\Bigg|\Bigg|\Bigg(\sum_{j\in N_k^p} A_{(k,j)}\Bigg)\Bigg(\sum_{j\in P_k^p} A_{(k,j)}\Bigg)^{-1}\Bigg|\Bigg|_E, \tag{3.119}$$

$$\wedge \quad \forall\, l \in P_k^{p,+} :$$

$$\tau\,\lambda_{\min}(A_{(k,l)}) \geq \mu_k \mathrm{cond}_E(A_{(k,k)})\,||A_{(k,l)}||_E\Bigg|\Bigg|\Bigg(\sum_{j\in N_k^p} A_{(k,j)}\Bigg)\Bigg(\sum_{j\in P_k^p} A_{(k,j)}\Bigg)^{-1}\Bigg|\Bigg|_E, \tag{3.120}$$

hold with a $\tau \geq 1$ not depending on k, l, the τ-condition (3.105) *is fulfilled.*

Proof of (a). Because $A > 0$, all $A_{(k,k)} > 0$, and we have

$$\begin{aligned}\mathrm{cond}_E(A_{(k,k)})||T_{(k)}||_E &= \rho(A_{(k,k)})\rho(A_{(k,k)}^{-1})||T_{(k)}||_E \\ &= ||A_{(k,k)}||_E\,||A_{(k,k)}^{-1}||_E\,||T_{(k)}||_E \\ &\geq ||A_{(k,k)}||_E\,||A_{(k,k)}^{-1}T_{(k)}||_E\end{aligned}$$

$$= ||A_{(k,k)}||_E \left|\left| I - \sum_{l \in P_k^{p,-}} W_{(k,l)} + \sum_{l \in P_k^{p,+}} W_{(k,l)} \right|\right|_E$$

because of

$$I - \sum_{l \in P_k^p \setminus P_k^{p,+}} W_{(k,l)} + \sum_{l \in P_k^{p,+}} W_{(k,l)} = A_{(k,k)}^{-1} T_{(k)} \ . \tag{3.121}$$

Hence, (3.114) implies (3.108). Analogously,

$$\begin{aligned}
& \mathrm{cond}_E(A_{(k,k)}) ||A_{(k,l)}||_E \rho\Big(\Big(\sum_{j \in N_k^{p,-}} A_{(k,j)} \Big) \Big(\sum_{j \in P_k^{p,-}} A_{(k,j)} \Big)^{-1} \Big) \\
= \ & ||A_{(k,k)}||_E \, ||A_{(k,k)}^{-1}||_E \, || - A_{(k,l)}||_E \\
& \qquad \cdot \rho\Big(\Big(\sum_{j \in N_k^{p,-}} A_{(k,j)} \Big) \Big(\sum_{j \in P_k^{p,-}} A_{(k,j)} \Big)^{-1} \Big) \\
\geq \ & ||A_{(k,k)}||_E \left|\left| -A_{(k,k)}^{-1} \Big(\sum_{j \in N_k^{p,-}} A_{(k,j)} \Big) \Big(\sum_{j \in P_k^{p,-}} A_{(k,j)} \Big)^{-1} A_{(k,l)} \right|\right|_E \\
= \ & ||A_{(k,k)}||_E \, ||W_{(k,l)}||_E
\end{aligned}$$

which shows that (3.115) implies (3.109). Analogously, (3.116) implies (3.110). The proof of (b) is straightforward now. ■

Remark 3.37 If $T_{(k)} > 0$, (3.114) is equivalent to $\tau \geq \mu_k \mathrm{cond}_E(A_{(k,k)}) \ \mathrm{cond}_E(T_{(k)})$ according to (2.57) and (2.58). If $-A_{(k,l)} > 0$ holds, (3.115) is equivalent to

$$\tau \geq \mu_k \mathrm{cond}_E(A_{(k,k)}) \ \mathrm{cond}_E(-A_{(k,l)}) \rho\Big(\Big(\sum_{j \in N_k^{p,-}} A_{(k,j)} \Big) \Big(\sum_{j \in P_k^{p,-}} A_{(k,j)} \Big)^{-1} \Big), \tag{3.122}$$

analogously for the other inequalities. ▲

Obviously, the obtained estimates for τ are similar to the ones for the variable-wise interpolation which can be found in Theorem 3.7.

Remark 3.38 If we additionally demand the $A_{(k,l)}$, which are in the same row of A, to commutate pairwise, i.e. $A_{(k,l)} A_{(k,j)} = A_{(k,j)} A_{(k,l)}$, we can drop (3.114) and replace conditions (3.115) and (3.116) by

$$\tau \geq n_u \rho\Big(\Big(\sum_{j \in N_k^{p,-}} (-A_{(k,j)}) \Big) \Big(\sum_{j \in P_k^{p,-}} (-A_{(k,j)}) \Big)^{-1} \Big), \tag{3.123}$$

$$\tau \geq n_u \rho\Big(\Big(\sum_{j \in N_k^{p,+}} A_{(k,j)} \Big) \Big(\sum_{j \in P_k^{p,+}} A_{(k,j)} \Big)^{-1} \Big). \tag{3.124}$$

Ideas of a proof (for the case that all $A_{(k,l)} \leq 0$ for $k \neq l$) can be found in [58]. However, these apparently simpler conditions, which are formally nearly the analogs to (3.38) and

(3.39), are obtained by paying too high a price because the assumption of commutating matrices does *usually not hold*, not even for symmetric matrices. In addition, the above estimates with n_u need not to be better. ▲

Remark 3.39 In contrast to the estimates obtained for variable-based AMG, the setting $P_k^p = N_k^p$ does not automatically result in $\tau = 1$ as one whould expect. In the inequalities (3.123) and (3.124), $\tau = 1$ can only emerge for $n_u = 1$. This, however, is in contrast to the fact that we investigate coupled PDE systems here. For conditions (3.115) and (3.116), the situation is comparable since $\tau = 1$ implies $\mathrm{cond}_E(A_{(k,l)}) = 1$ for all k, l here. However, if only the symmetry of $A_{(k,l)}$ ($k \neq l$) is demanded, also coupled systems with $A_{(k,k)} = c_1 I$ ($c_1 > 0$) and, for instance, $A_{(k,l)} = \begin{bmatrix} 0 & c_2 \\ c_2 & 0 \end{bmatrix}$ (with a c_2 so that $A \in \mathcal{A}_{\text{spd}}$ is still satisfied) would result in $\mathrm{cond}_E(A_{(k,l)}) = 1$ for all k, l. ▲

Variable-Based AMG In case of variable-based AMG where point-coupling matrices $A_{(k,l)}$ degenerate to matrix entries a_{kl}, we can directly derive Theorems 3.7 and 3.5 from Theorem 3.1:

Corollary 3.2 (a) *Let $A > 0$ and $t_i = a_{ii} - \sum_{j \in N_i} |a_{ij}| \geq 0$ for all i. With fixed $\tau \geq 1$ select a C/F-splitting such that the following holds for each $i \in F$: If $N_i^- \neq \emptyset$, there is a set $P_i^- \subseteq C \cap N_i^-$ satisfying*

$$\sum_{j \in P_i^-} |a_{ij}| \geq \frac{1}{\tau} \sum_{j \in N_i^-} |a_{ij}| \tag{3.125}$$

and, if $N_i^+ \neq \emptyset$, there is a set $P_i^+ \subseteq C \cap N_i^+$ satisfying

$$\sum_{j \in P_i^+} a_{ij} \geq \frac{1}{\tau} \sum_{j \in N_i^+} a_{ij} \,. \tag{3.126}$$

Then the interpolation (3.23) *with weights* (3.36) *satisfies the τ-condition* (3.21).
(b) *Let A be a weakly diagonally dominant, essentially positive type matrix. With fixed $\tau \geq 1$ select a C/F-splitting so that, for each $i \in F$, there is a set $P_i \subseteq C \cap N_i^-$ satisfying*

$$\sum_{j \in P_i} |a_{ij}^-| \geq \frac{1}{\tau} \sum_{j \in N_i} |a_{ij}^-| \,. \tag{3.127}$$

Then, interpolation (3.23) *with weights* (3.31) *satisfies the τ-condition* (3.21) *with τ/c rather than τ.*

Proof of (a): According to (3.36) and the assumptions, we have $w_{ij} > 0$ for all $j \in P_i^-$, $w_{ij} < 0$ for all $j \in P_i^+$ and

$$\sum_{j \in P_i} |w_{ij}| = \sum_{j \in P_i^-} w_{ij} - \sum_{j \in P_i^+} w_{ij} = \frac{1}{a_{ii}} \Big(- \sum_{j \in N_i^-} a_{ij} + \sum_{j \in N_i^+} a_{ij} \Big) = \frac{a_{ii} - t_i}{a_{ii}} \,.$$

Hence, we have $\mu_i = \sum_{j \in P_i} |w_{ij}| + \frac{t_i}{a_{ii}} = 1$. The remainder follows from the obvious reduction of Theorem 3.1(a) to the variable-based case. The **proof of (b)** is analogous. ■

Different variable-wise interpolations, especially the direct interpolation with weights (3.27), were already discussed in Section 3.2.3.

$\widetilde{\boldsymbol{P}}$-Interpolations For $\widetilde{\boldsymbol{P}}$-interpolations (3.103) the following corollary can be derived from Theorem 3.13:

Theorem 3.14 *Let $A > 0$, $N_k^p = N_k^{p,+} \cup N_k^{p,-}$ and $T_{(k)} \geq 0$ for all $k \in \mathcal{V}^p$. Select a $\tau \geq 1$, a C^p/F^p-splitting and a set P_k^p for each $k \in F^p$. Let*

$$\forall\, k \in F^p \;:\quad 1 - \sum_{l \in \widetilde{P}_k^{p,-}} w_{kl}^p + \sum_{l \in \widetilde{P}_k^{p,+}} w_{kl}^p \geq 0\,. \tag{3.128}$$

(a) *If $P_k^{p,-} = \widetilde{P}_k^{p,-}$ and $P_k^{p,+} = \widetilde{P}_k^{p,+}$ and I_{FC} is defined by* (3.103)*, then, for all $k \in F^p$, the inequalities*

$$\tau\, \lambda_{\min}(T_{(k)}) \quad \geq \frac{||A_{(k,k)}||_E}{\widetilde{p}_{kk}} t_k^p \tag{3.129}$$

$$\wedge \quad \forall\, l \in P_k^{p,-} \;:$$

$$\tau\, \lambda_{\min}(-A_{(k,l)}) \geq \frac{||A_{(k,k)}||_E}{\widetilde{p}_{kk}} \left| \frac{\sum_{j \in \widetilde{N}_k^{p,-}} \widetilde{p}_{kj}}{\sum_{j \in P_k^{p,-}} \widetilde{p}_{kj}} \right| |\widetilde{p}_{kl}| \tag{3.130}$$

$$\wedge \quad \forall\, l \in P_k^{p,+} \;:$$

$$\tau\, \lambda_{\min}(A_{(k,l)}) \quad \geq \frac{||A_{(k,k)}||_E}{\widetilde{p}_{kk}} \left| \frac{\sum_{j \in \widetilde{N}_k^{p,+}} \widetilde{p}_{kj}}{\sum_{j \in P_k^{p,+}} \widetilde{p}_{kj}} \right| |\widetilde{p}_{kl}| \tag{3.131}$$

imply the τ-condition (3.105)*. If the $A_{(k,k)}$ are replaced by $D_{(k,k)}$, the above inequalities imply the τ-condition* (3.21)*.*

(b) *If $\widetilde{N}_k^{p,+} = \emptyset$ and I_{FC} is defined by* (3.103)*, then $\widetilde{N}_k^p = \widetilde{N}_k^{p,-}$, $\widetilde{P}_k^{p,-} = \widetilde{P}_k^p = P_k^p$ and, for all $k \in F^p$, the inequalities*

$$\tau\, \lambda_{\min}(T_{(k)}) \quad \geq ||A_{(k,k)}||_E \left(1 - \sum_{l \in P_k^{p,-}} w_{kl}^p + \sum_{l \in P_k^{p,+}} w_{kl}^p\right) \tag{3.132}$$

$$\wedge \quad \forall\, l \in P_k^{p,-} \;:$$

$$\tau\, \lambda_{\min}(-A_{(k,l)}) \geq \frac{||A_{(k,k)}||_E}{\widetilde{p}_{kk}} \left| \frac{\sum_{j \in \widetilde{N}_k^p} \widetilde{p}_{kj}}{\sum_{j \in P_k^p} \widetilde{p}_{kj}} \right| |\widetilde{p}_{kl}| \tag{3.133}$$

$$\wedge \quad \forall\, l \in P_k^{p,+} \;:$$

$$\tau\, \lambda_{\min}(A_{(k,l)}) \quad \geq \frac{||A_{(k,k)}||_E}{\widetilde{p}_{kk}} \left| \frac{\sum_{j \in \widetilde{N}_k^p} \widetilde{p}_{kj}}{\sum_{j \in P_k^p} \widetilde{p}_{kj}} \right| |\widetilde{p}_{kl}| \tag{3.134}$$

imply the τ-condition (3.105)*.*

The **proofs of (a) and (b)** are analogous. Hence, we only prove (a) explicitly. Since (3.103) holds for all $k \in F^p$ and all $l \in P_k^p$,

$$W_{(k,l)}(e_{(k)} - e_{(l)}) = w_{kl}^p (e_{(k)} - e_{(l)})$$

holds, and we can evaluate

$$||W_{(k,l)}||_E = |w_{kl}^p| = \begin{cases} \frac{|\widetilde{p}_{kl}|}{\widetilde{p}_{kk}} \left| \frac{\sum_{j\in\widetilde{N}_k^{p,-}} \widetilde{p}_{kj}}{\sum_{j\in\widetilde{P}_k^{p,-}} \widetilde{p}_{kj}} \right|, & l \in \widetilde{P}_k^{p,-}, \\ \frac{|\widetilde{p}_{kl}|}{\widetilde{p}_{kk}} \left| \frac{\sum_{j\in\widetilde{N}_k^{p,+}} \widetilde{p}_{kj}}{\sum_{j\in\widetilde{P}_k^{p,+}} \widetilde{p}_{kj}} \right|, & l \in \widetilde{P}_k^{p,+}, \end{cases} \tag{3.135}$$

$$\left|\left| I - \sum_{l\in\widetilde{P}_k^{p,-}} W_{(k,l)} + \sum_{l\in\widetilde{P}_k^{p,+}} W_{(k,l)} \right|\right|_E = \left| 1 - \sum_{l\in\widetilde{P}_k^{p,-}} w_{kl}^p + \sum_{l\in\widetilde{P}_k^{p,+}} w_{kl}^p \right| = \frac{|t_k^p|}{\widetilde{p}_{kk}}$$

$$\text{with } t_k^p := \widetilde{p}_{kk} + \sum_{l\in\widetilde{N}_k^{p,-}} \widetilde{p}_{kl} - \sum_{l\in\widetilde{N}_k^{p,+}} \widetilde{p}_{kl} = \widetilde{p}_{kk} - \sum_{l\in\widetilde{N}_k^p} |\widetilde{p}_{kl}| \ .$$

According to (3.102) and the assumptions, we have

$$1 - \sum_{l\in P_k^{p,-}} w_{kl}^p + \sum_{l\in P_k^{p,+}} w_{kl}^p \geq 0\,, \tag{3.136}$$

$$\forall\, l \in P_k^{p,-} \ : \ w_{kl}^p \geq 0 \quad \text{and} \quad \forall\, l \in P_k^{p,+} \ : \ w_{kl}^p \leq 0\,, \tag{3.137}$$

and therefore

$$\mu_k = \frac{\sum_{l\in\widetilde{N}_k^p} |\widetilde{p}_{kl}|}{\widetilde{p}_{kk}} + \frac{t_k^p}{\widetilde{p}_{kk}} = \frac{\sum_{l\in\widetilde{N}_k^p} |\widetilde{p}_{kl}|}{\widetilde{p}_{kk}} + \frac{\widetilde{p}_{kk} - \sum_{l\in\widetilde{N}_k^p} |\widetilde{p}_{kl}|}{\widetilde{p}_{kk}} = 1 \ . \tag{3.138}$$

The remainder follows from (3.135). ■

For all types of $\widetilde{\boldsymbol{P}}$, the diagonal entries $\widetilde{p}_{kk}$ should be as large as possible and the sums (for instance, $|\sum_{j\in\widetilde{N}_k^{p,-}} \widetilde{p}_{kj}|$ and $|\sum_{j\in P_k^{p,-}} \widetilde{p}_{kj}|$) as similar in magnitude as possible in order to obtain a small τ. This corresponds to the results we have obtained for variable-based AMG.

Remark 3.40 A difference, however, lies in the fact that in the estimates above even for $P_k^{p,-} = \widetilde{N}_k^{p,-}$ (etc.) not automatically $\tau = 1$ emerges. $\tau = 1$ would result if, for instance, $\widetilde{p}_{kl} = |\lambda|_{\min}(A_{(k,l)})$ for $k \neq l$ and $\widetilde{p}_{kk} = ||A_{(k,k)}||_E$ and $t_k^p = \lambda_{\min}(T_{(k)})$. However, as we have discussed in Section 3.4.2.7, such a definition of the $\widetilde{p}_{kl}$ is not advantageous since this might prevent coarsening totally. ▲

For $\boldsymbol{P}$-interpolations with a norm-based $\boldsymbol{P}$, we can concretize the above results further.

Corollary 3.3 (a) *If $\boldsymbol{P}$ is defined by* (3.74) *with the Euclidean norm,*

$$\forall\,(k,l) \in \Sigma \ : \ p_{kl} = \begin{cases} ||A_{(k,l)}||_E & \text{for } A_{(k,l)} \geq 0 \quad (k \neq l) \\ -||A_{(k,l)}||_E & \text{else} \quad (k \neq l) \end{cases} \quad \text{and} \quad p_{kk} = ||A_{(k,k)}||_E\,,$$

then $P_k^{p,-} = \widetilde{P}_k^{p,-}$ and $P_k^{p,+} = \widetilde{P}_k^{p,+}$, which shows that we can apply Theorem 3.14(a) if the remaining conditions are fulfilled. The condition (3.128) *is equivalent to*

$$\forall\, k \in F^p \ : \quad t_k^p = ||A_{(k,k)}||_E - \sum_{j\in\widetilde{N}_k^p} ||A_{(k,j)}||_E \geq 0\,, \tag{3.139}$$

and the inequalities (3.129), (3.130) *and* (3.131) *are equivalent to*

$$\tau\,\lambda_{\min}(T_{(k)}) \quad \geq t_k^p \tag{3.140}$$

$$\wedge \quad \forall\, l \in P_k^{p,-} \;:$$

$$\tau\,\lambda_{\min}(-A_{(k,l)}) \geq ||A_{(k,l)}||_E \frac{\sum_{j\in\widetilde{N}_k^{p,-}} ||A_{(k,j)}||_E}{\sum_{j\in P_k^{p,-}} ||A_{(k,j)}||_E} \tag{3.141}$$

$$\wedge \quad \forall\, l \in P_k^{p,+} \;:$$

$$\tau\,\lambda_{\min}(A_{(k,l)}) \quad \geq ||A_{(k,l)}||_E \frac{\sum_{j\in\widetilde{N}_k^{p,+}} ||A_{(k,j)}||_E}{\sum_{j\in P_k^{p,+}} ||A_{(k,j)}||_E} \;. \tag{3.142}$$

(b) *If* $\boldsymbol{P}$ *is defined by* (3.72) *with the Euclidean norm,*

$$\forall\,(k,l)\in\Sigma \;:\; p_{kl} = -||A_{(k,l)}||_E \quad (k\neq l) \quad \text{and} \quad p_{kk} = ||A_{(k,k)}||_E\,,$$

then $\widetilde{N}_k^{p,+} = \emptyset$*, which shows that we can apply Theorem 3.14(b) if the remaining conditions are fulfilled. Define*

$$\phi_k := \frac{\sum_{j\in P_k^{p,-}} ||A_{(k,j)}||_E - \sum_{j\in P_k^{p,+}} ||A_{(k,j)}||_E}{\sum_{j\in P_k^p} ||A_{(k,j)}||_E} \;.$$

Then condition (3.128) *is equivalent to*

$$\forall\, k\in F^p \;:\quad ||A_{(k,k)}||_E - \phi_k \sum_{j\in\widetilde{N}_k^p} ||A_{(k,j)}||_E \geq 0\,,$$

and the inequalities (3.132) *to* (3.134) *are equivalent to*

$$\tau\,\lambda_{\min}(T_{(k)}) \quad \geq ||A_{(k,k)}||_E - \phi_k \sum_{j\in\widetilde{N}_k^p} ||A_{(k,j)}||_E \tag{3.143}$$

$$\wedge \quad \forall\, l \in P_k^{p,-} \;:$$

$$\tau\,\lambda_{\min}(-A_{(k,l)}) \geq ||A_{(k,l)}||_E \frac{\sum_{j\in\widetilde{N}_k^p} ||A_{(k,j)}||_E}{\sum_{j\in P_k^p} ||A_{(k,j)}||_E} \tag{3.144}$$

$$\wedge \quad \forall\, l \in P_k^{p,+} \;:$$

$$\tau\,\lambda_{\min}(A_{(k,l)}) \quad \geq ||A_{(k,l)}||_E \frac{\sum_{j\in\widetilde{N}_k^p} ||A_{(k,j)}||_E}{\sum_{j\in P_k^p} ||A_{(k,j)}||_E} \;. \tag{3.145}$$

(c) *If* $\boldsymbol{P}$ *is defined by* (3.75) *with the Euclidean norm,*

$$\forall\,(k,l)\in\Sigma \;:\; p_{kl} = \begin{cases} ||A_{(k,l)}||_E & \textit{for } A_{(k,l)} \geq 0 \quad (k\neq l) \\ -||A_{(k,l)}||_E & \textit{else} \quad (k\neq l) \end{cases} \quad \text{and} \quad p_{kk} = \sum_{l\neq k} |p_{kl}|\,,$$

then $P_k^{p,-} = \widetilde{P}_k^{p,-}$, $P_k^{p,+} = \widetilde{P}_k^{p,+}$*. The conditions* (3.128) *and* (3.129) *are fulfilled because of* $t_k^p = 0$*, and the inequalities* (3.130) *and* (3.131) *are equivalent to*

$$\forall\, l \in P_k^{p,-} \;:$$

$$\tau\,\lambda_{\min}(-A_{(k,l)}) \geq \frac{||A_{(k,l)}||_E\,||A_{(k,k)}||_E}{\sum_{j\in\widetilde{N}_k^p}||A_{(k,j)}||_E}\,\frac{\sum_{j\in\widetilde{N}_k^{p,-}}||A_{(k,j)}||_E}{\sum_{j\in P_k^{p,-}}||A_{(k,j)}||_E} \tag{3.146}$$

$$\wedge \quad \forall\, l \in P_k^{p,+}\ :$$

$$\tau\,\lambda_{\min}(A_{(k,l)}) \quad \geq \frac{||A_{(k,l)}||_E\,||A_{(k,k)}||_E}{\sum_{j\in\widetilde{N}_k^p}||A_{(k,j)}||_E}\,\frac{\sum_{j\in\widetilde{N}_k^{p,+}}||A_{(k,j)}||_E}{\sum_{j\in P_k^{p,+}}||A_{(k,j)}||_E}\,. \tag{3.147}$$

(d) *If* $\boldsymbol{P}$ *is defined by* (3.73) *with the Euclidean norm,*

$$\forall\,(k,l)\in\Sigma\ :\ p_{kl} = -||A_{(k,l)}||_E \quad (k\neq l) \quad \text{and} \quad p_{kk} = -\sum_{l\neq k} p_{kl}\,,$$

then $\widetilde{N}_k^{p,+} = \emptyset$. *Define* ϕ_k *as in (b). Then condition* (3.128) *is equivalent to*

$$\forall\, k\in F^p\ :\quad 1-\phi_k \geq 0\,,$$

and the inequalities (3.132) *to* (3.134) *are equivalent to*

$$\tau\,\lambda_{\min}(T_{(k)}) \quad \geq (1-\phi_k)||A_{(k,k)}||_E \tag{3.148}$$

$$\wedge \quad \forall\, l \in P_k^{p,-}\ :$$

$$\tau\,\lambda_{\min}(-A_{(k,l)}) \geq \frac{||A_{(k,l)}||_E\,||A_{(k,k)}||_E}{\sum_{j\in P_k^p}||A_{(k,j)}||_E} \tag{3.149}$$

$$\wedge \quad \forall\, l \in P_k^{p,+}\ :$$

$$\tau\,\lambda_{\min}(A_{(k,l)}) \quad \geq \frac{||A_{(k,l)}||_E\,||A_{(k,k)}||_E}{\sum_{j\in P_k^p}||A_{(k,j)}||_E}\,. \tag{3.150}$$

Proof. The statements follow from (3.73) to (3.74), (3.136), (3.138) and (2.57)ff. by straightforward calculations. ■

Remark 3.41 If $(-A_{(k,l)}) > 0$ holds, inequality (3.141) is equivalent to

$$\tau\ \geq \mathrm{cond}(-A_{(k,l)})\,\frac{\sum_{j\in\widetilde{N}_k^{p,-}}||A_{(k,j)}||_E}{\sum_{j\in P_k^{p,-}}||A_{(k,j)}||_E}\,, \tag{3.151}$$

analogously for the other inequalities. ▲

3.4.4.3 Comparison of the Block- and $\boldsymbol{P}$-Interpolations

As can be expected, the "best" conditions emerge for the block-interpolation (3.91) and the $\boldsymbol{P}$-interpolation (3.103) with $\boldsymbol{P}$ being defined by (3.74) or by (3.75). This is because negative and positive definite point-coupling matrices $A_{(k,l)}$ are distinguished there. However, for the variants which treat all $A_{(k,l)}$ equally, the lower bounds for τ are comparable. Only these cheaper variants are used for practical applications.

Regarding the $\boldsymbol{P}$-interpolations, Corollary 3.3 reveals that variants (3.74) and (3.72), in particular $p_{kk} = ||A_{(k,k)}||$, should be preferred if $||A_{(k,k)}||$ is larger than $\sum_{j\in\widetilde{N}_k^p}||A_{(k,j)}||_E$.

Otherwise, variants (3.75) or (3.73) should yield better upper bounds. Since, in addition, the last two variants ensure $\boldsymbol{P}$ being an M-matrix, but (3.73) is cheaper, this variant should be chosen as a default for practical applications.

By comparing Corollaries 3.1 and 3.3, we can conclude that the conditions on the block-interpolations and on the "corresponding" $\boldsymbol{P}$-interpolations are quite similar. Hence, the upper bounds for the two-level convergence rates should be similar for both interpolation types, and it is problem-, that means A-, dependent which set of conditions yields a smaller lower bound for τ. However, keep in mind that all these convergence estimates are only very rough upper bounds and rather "worst case estimates" since, for instance, geometric information is not taken into account. In practice, the convergence is usually much faster than predicted. It can be observed that block-interpolation and $\boldsymbol{P}$-interpolation behave often similarly. In addition, a $\boldsymbol{P}$-interpolation is usually much cheaper to compute than a block-interpolation and does not face technical problems. Therefore, we usually prefer the $\boldsymbol{P}$-interpolation.

Chapter 4

Software Issues - The SAMG Library

In the preceding chapters, we have described our AMG strategies for solving PDE systems from a methodical point of view. In particular, the range of applicability of the strategies and their robustness have been discussed. This discussion will be continued in Chapter 5 for three concrete practical applications. In this Chapter 4, we explain the realization of our strategies within the Fortran90 library **SAMG** [89] and finalize the discussion of the model problems.

SAMG has two faces. One one hand, it is a product-quality library of efficient AMG approaches for solving matrices arising from different industrially relevant PDEs and PDE systems. On the other hand, it provides a user-extensible, rich AMG environment for a flexible testing of combinations of various different modules and for tailoring AMG approaches to even more applications than those already handled efficiently.

The "standard" AMG components are discussed as well as several variants serving the goal to increase their robustness, or to decrease their computational work or memory requirements. We especially show that SAMG provides highest flexibility for adaptations to various situations arising in practice. We analyze the computational work and memory requirements of the setup phase and one cycle and show that, in practice, they are usually $O(N)$. We also demonstrate that our AMG solvers exhibit reasonable "magnitude of $O(N)$'s constants" regarding computational work *and* memory requirements. AMG approaches with *aggressive* coarsening and GS smoothing used as preconditioners often turn out as reasonable compromises for the overall efficiency. Typically, their memory requirements are approximately equal to or even lower than 1.5 times the requirements for the standard one-level preconditioner ILU(0) [1].

We start with an overview of SAMG. Section 4.1 outlines its key features and the basic course of each AMG approach. In Sections 4.2 to 4.4, we concentrate on the three main AMG components, i.e. coarsening, interpolation and smoothing. Section 4.4 also comments on the usage of AMG as a preconditioner, and which one-level solvers are incorporated in SAMG. For each of the general AMG strategies, the characteristic factors (complexities) dominating the computational cost are discussed in Section 4.5. Finally, Section 4.6 presents the performance of different AMG approaches applied to the model problems.

Remark 4.1 We are not going to describe the "real" implementation of SAMG on Fortran90 level. Due to the rich possibilities SAMG provides, only the most important features are described. We concentrate on the principal methods and work out which "routines" the VAMG,

[1] This means that c_{prec} defined in Section 4.1.3 is approximately equal to or even smaller than 1.5.

UAMG and PAMG approaches can use in common, and where different "routines" are necessary. Detailed information on the installation and usage of the SAMG library and an explanation of all of SAMG's user parameters are given in the "SAMG user's manual" [89]. ▲

4.1 Overview

SAMG is a modern, modular Fortran90 library of algebraic multigrid approaches [2]. It is the successor of the Fortran90 library RAMG which, in turn, is the successor of the old Fortran77 code AMG1R5. Whereas the AMG approach described in the classical paper [71] has been implemented in AMG1R5, RAMG is basically the realization of AMG described in [87] (see also Section 3.2). RAMG's extension to our current library SAMG incorporates our whole AMG methodology, in particular our point-based strategy described in Chapter 3.4 with a variety of different concrete primary matrices and interpolations.

Our overview of SAMG starts with a summary of its key features in Section 4.1.1. At the end of Section 4.1.1, we emphasize the corresponding new features compared with SAMG's predecessor RAMG.

All our AMG approaches consist of two phases, a setup phase and a solution phase. In the **setup phase**, the level hierarchy is constructed, the main parts of which are the construction of coarsening and interpolation. In the **solution phase**, standard multigrid cycles are performed by means of a smoothing process and the constructed coarse level and interlevel transfer operators. These two phases are outlined in Sections 4.1.2.1 and 4.1.2.2, respectively. In Section 4.1.3, some additional definitions are made.

4.1.1 Key Features

4.1.1.1 Modularity, Adaptability, Flexibility, Special Features

SAMG's realization is based on a modular concept. Each of the main components of an AMG algorithm, that is smoothing, coarsening, interpolation, computation of the Galerkin operator, and coarsest-level solution, represents a separate "module". The coarsening module, in turn, consists of two or three modules, namely the definition of $\boldsymbol{P}$ (only in case of a PAMG approach), a sorting and a splitting process.

All modules are completely separated from each other, and each has a fixed task (for instance the construction of interpolation) and a fixed type of output (for instance, an interpolation operator I_{FC}), but consists of several concrete variants (for instance a direct interpolation based on coordinates) of the respective AMG component. Moreover, in each module, a variant may be released only for one or two of our strategies (variable-, unknown- or point-based, respectively) or may require additional data such as coordinates to be available.

The separation and clear interfacing has the important advantage that new variants can be added quickly to a module when necessary, without touching the other modules. As a

[2] Clearly, by forcing the number of levels in the hierarchy to be one, also a variety of one-level solvers can be selected. For details, see Section 4.4. Consequently, the SAMG library can even be regarded as a complete library containing also classical one-level solvers.

consequence, we can offer user interfaces for each of SAMG's main parts to implement their own variants and to adapt SAMG further to specific situations.

By combining proper modules, very different concrete approaches can be selected, with or without the employment of additional data such as VU or VP mappings or coordinates. This makes SAMG a complete and very flexible AMG environment, corresponding to the general, flexible AMG methodology discussed in the previous chapter. However, it should be stressed again that the variants have to be selected carefully: the natural technical limits as well as the dependence on the concrete application have always to be borne in mind.

Special Features If a series of matrix equations with similar or even identical matrices shall be solved, for instance for time-dependent and/or nonlinear problems, SAMG can be advised to reuse parts of or even the whole setup. An example is given in Section 5.2.2.

In general, SAMG can be forced to exclude particular variables from the coarsening process. For each variable, it can be specified whether it is forced to remain on the finest level or stay in all levels. (see also vector `ic_set` described below). Such an intervention into SAMG's setup phase is often useful. For instance, it provides simple and powerful workarounds for cases where a PDE system with a small number of additional (algebraic) constraints is to be solved, and the algebraic constraints can and shall be handled by (ILU-type) smoothing and (BiCGstab or GMRes) acceleration only. An example is discussed in Section 5.3.2. The last row of the respective matrices corresponds to an algebraic constraint and has a zero diagonal. In the case of zero diagonals, it is usually necessary to force corresponding variables to remain on the finest level. If necessary, they can also be excluded from smoothing[3].

If the input matrix A is positive definite, so are the coarser-level matrices (cf. Lemma 3.1), at least up to round-off. Practically, however, it might happen that some coarse-level diagonals become (numerically) zero or even negative. For instance, this happens for the semiconductor drift-diffusion systems discussed in Section 5.3.2. Besides the technical problems such exceptional matrix rows produce, AMG's convergence usually suffers from their occurrence. Ways to handle or avoid nonpositive diagonals are discussed in Appendix A.1.

SAMG features several user-interfaces. Already explicitly provided are `samg_user_coo` (see below), `ic_set`, a new type of primary matrices, another coarsest-level solver and an acces to SAMG's message handler. For user-defined smoothing, coarsening or interpolation, simply the corresponding controlling parameters and calling sequences would have to be extended.

4.1.1.2 Data Structure

The matrix data are transmitted to SAMG in the so-called modified **compressed sparse row format**[4] which consists of three vectors, `ia(1:`n_v`+1)`, `ja(1:`n_A`)`, `a(1:`n_A`)`. n_v denotes the number of variables, n_A the number of matrix entries stored. In `a` and `ja` the entries of A and their column numbers are stored. The entries of the first row come first, followed

[3] This is, in particular, necessary in case of Jacobi or VGS smoothing. Depending on the concrete matrix and its "arrangement", BGS- or (M)ILU(T)-smoothing might be able handle them.

[4] CSR format, also called Harwell-Boeing format.

by the entries of the second row, and so on. For each `i`, `ia(i)` stores the beginning of row `i` in `a` and `ja`, and we define `ia(`n_v`+1)=`n_A`+1`. It is a *modified* CSR format since diagonals are stored first within a row (i.e. `ja(ia(i))=i`). Except of the diagonal entries, the off-diagonal entries can be stored in any order within a row.

For UAMG approaches, a VU mapping needs to be passed to SAMG, for PAMG approaches, a VU and a VP mapping. This can be done via vectors `iu(1:`n_v`)` and `ip(1:`n_v`)` containing the mapping of the variable indices to the unknown and point indices, respectively. For VAMG approaches, both `iu` and `ip` can be submitted as dummy vectors of length 1. For UAMG approaches, `ip(1:`n_v`)` can be dummy. The order of the variables can be arbitrary. In particular, an unknown-wise ordering is not necessary. When using a point-based approach, both `iu(1:`n_v`)` and `ip(1:`n_v`)` have to be submitted and have to follow the following rule. Within `ip`, the values are not allowed to decrease. This means, the variables have to be sorted pointwise with an increasing numbering of points. To avoid a renumbering of the concrete point data, passed to SAMG by a simulation code, empty points, i.e. $\mathcal{P}_k = \emptyset$ for some k, are allowed though. The order of the variables of a point $\mathcal{P}_k$ can be *arbitrary*. Moreover, the number of variables may vary from point to point. This is very important since it often occurs in practical applications (for an example, see Section 5.3).

Optionally, if (2D or 3D) coordinates shall be used, the user can provide a subroutine called
`samg_user_coo` which returns, for each *variable* i, the coordinates of the corresponding grid node.

As mentioned above, SAMG provides an interface for the user to explicitly force some variables into F or C. For this purpose, the user has to call an allocation routine for a vector called `ic_set`, and then to mark the corresponding variables by a set routine.

The five vectors mentioned, optionally the user-defined function `samg_user_coo`, optionally the vector `ic_set`, and, of course, control parameters (see the SAMG User's Manual [89]) are the maximum amount of data that have to be provided to SAMG.

Remark 4.2 With its easy-to-use interface, SAMG can simply be plugged into existing simulation codes regardless if they are written in Fortran95, Fortran77, C or C++. SAMG runs on all platforms used today, ranging from Unix systems (tested on Compaq Alpha, IBM, Sun, SGI, HP) over Linux (on Alpha as well as PC) to Windows systems, and is compatible to all state-of-the-art compilers. ▲

Remark 4.3 Compared with RAMG, in particular the whole point-based strategy, the user interfaces and the support of C and C++ have been added. Most of these features have been added or substantially extended during the work on this thesis. New features of coarsening, interpolation and smoothing will be mentioned in Sections 4.2 to 4.4. ▲

4.1.2 SAMG's Two Phases

4.1.2.1 The Setup Phase

Fig. 4.1 outlines the general steps which are performed within SAMG's setup phase. The algorithm starts with a preparation of some auxiliary quantities and an initialization of pa-

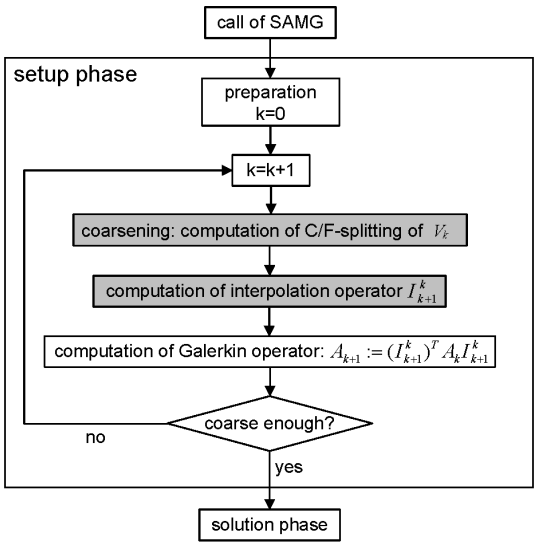


Figure 4.1: The main steps of SAMG's setup phase.

rameters which control the different parts of the whole SAMG run. Then[5] the recursive process of constructing the level hierarchy begins, starting on the finest level, numbered by index 1:

$$k = 1 \ , \ \ A_1 := A \ , \ \ \mathcal{V}_1 := \mathcal{V} \ .$$

For the current level k, the coarsening is then constructed, i.e. the C/F-splitting of the set $\mathcal{V}_k$ is computed. If a reasonable splitting cannot be determined successfully, the setup phase is terminated, the current level k is defined to be the coarsest level, and the solution phase is entered, in which multigrid cycling between the finest level 1 and the coarsest level k is performed.

Otherwise, if a new C/F-splitting for $\mathcal{V}_k$ has been constructed successfully, the set of variables on the next coarser level $k+1$ is defined to be the set $C_k = C$ of coarse level variables on level k:

$$\mathcal{V}_{k+1} := C_k \ .$$

The algorithm continues with the computation of the interpolation operator I_{k+1}^k and, afterwards, the computation of the Galerkin operator on level $k+1$:

$$A_{k+1} := I_k^{k+1} A_k I_{k+1}^k \ . \tag{4.1}$$

Remark 4.4 In SAMG, the restriction operator I_k^{k+1} is always defined to be the transpose of interpolation, i.e.

$$I_k^{k+1} := \left(I_{k+1}^k\right)^T \ .$$

[5] unless a one-level method has been selected by the user.

However, it is constructed only temporarily when needed. ▲

Remark 4.5 There is one exception concerning the definition (4.1) of the coarse-level matrix, namely UAMG's variant (3.57). This variant is denoted by block-UAMG. ▲

The coarsening process and the construction of interpolation contain many degrees of freedom. However, their interplay as well as the properties of the employed smoother strongly affect the performance of the coarse-level correction process. Moreover, the differences between variable-, unknown- and point-based approaches lie in the concrete forms of coarsening and interpolation. Hence, in Sections 4.2 and 4.3 , we explain coarsening and interpolation in more detail, in each case starting with the methods implemented for the variable-based approach because they form the basis for the unknown- and point-based approaches.

4.1.2.2 The Solution Phase

In SAMG's second phase, the solution phase, standard multigrid cycles of V-, F- or W-type are performed. Details on these cycling types can be found in [94]. We have already mentioned in Section 3.2.5 that, although uniform V-cycle convergence cannot strictly be proved, V-cycles are typically more efficient than the more expensive F- and W-cycles. Therefore, in practice, we usually select the V-cycle.

The important degrees of freedom in defining the solution phase are the choice of

- the smoother,
- the type of cycling,
- the coarsest-level solver[6],
- the accelerator.

In Section 4.4 all smoothers and accelerators available within SAMG are listed.

4.1.3 Additional Notation

In all graphs of this chapter, items in light grey mark features of SAMG which - compared to RAMG - are new or substantially different, as has already been done in Fig. 4.1. We usually omit level indices. The notation introduced in Chapter 2 is used, extended by the following definitions.

Variable-based AMG is abbreviated by VAMG, analogously are UAMG and PAMG defined. The notation for the concrete variants for coarsening, interpolation, smoothing and acceleration are explained in Remarks 4.17, 4.21, 4.24, and 4.26.

For any matrix A, we will distinguish its sparsity pattern and its connectivity pattern. The **sparsity pattern** $\Sigma_s(A)$ is defined to be the set of index pairs (i, j) for which entries a_{ij} of A are stored. The sparsity pattern represents a superset of the **connectivity pattern** $\Sigma(A) = \Sigma_c(A)$, that is, the distribution of its nonzero entries.

[6] Variants are not explicitly discussed here. Sparse Gaussian elimination is used as a default and has been used for all numerical tests discussed in this thesis.

The **cardinality** of a set S is defined to be the number of elements of S and denoted by $|S|$. The **A-complexity** c_A, the **grid complexity** c_g and the **average "stencil size"** (i.e. row length) s_a over all levels are defined as

$$c_A := \frac{\sum_{k=1}^{n_{\rm lev}} |\Sigma_s(A_k)|}{|\Sigma_s(A_1)|} \quad , \quad c_g := \frac{\sum_{k=1}^{n_{\rm lev}} |\mathcal{V}_k|}{n_v} \quad , \quad s_a := \frac{\sum_{k=1}^{n_{\rm lev}} |\Sigma_s(A_k)|}{\sum_{k=1}^{n_{\rm lev}} |\mathcal{V}_k|} = \frac{c_A}{c_g} \frac{|\Sigma_s(A_1)|}{n_v} \quad ,$$

where $n_{\rm lev}$ denotes the number of levels in the AMG hierarchy including the finest level. Analogously, we define the $\boldsymbol{P}$**-complexity** $c_{\boldsymbol{P}}$ and the **point complexity** c_p. s_p denotes the **average number of interpolatory variables per F-variable**. Note that $|\Sigma_s(A_1)| = O(n_v)$ for all application classes we have in mind here. In case of PAMG, the computational costs will be seen to be (principally) proportional to the sum of the memory needed for the A_k and (unless $\boldsymbol{P}$ is a part of A) the $\boldsymbol{P}_k$. Hence, we define the **AMG-complexity** as

$$c_{\rm AMG} := \frac{\sum_{k=1}^{n_{\rm lev}} \big(|\Sigma_s(A_k)| + |\Sigma_s(\boldsymbol{P}_k)|\big)}{|\Sigma_s(A_1)|} \quad .$$

In order to compare the memory requirements of an AMG method with the standard one-level preconditioner ILU(0), we define the **preconditioner's complexity** as

$$c_{\rm prec} := {\rm mem}_{\rm AMG}/{\rm mem}_{\rm ILU}$$

where ${\rm mem}_{\rm AMG}$ (${\rm mem}_{\rm ILU}$) denotes the memory necessary for the AMG-method (the ILU(0) method) stand-alone including the memory necessary for storing A.

4.2 Coarsening

The first setup step in creating a new level is the coarsening process. The corresponding C/F-splitting constructed in the coarsening process should be suitable for the interpolation to be constructed in the next step.

In Section 4.2.1, we demonstrate that a basic, so-called *standard coarsening* algorithm [71] for VAMG can heuristically be derived from Theorem 3.5 (which indicates the quality of direct variable-based interpolation for weakly diagonally-dominant Stieltjes matrices).

Standard coarsening as described in Section 4.2.1.1 is the classical coarsening for the variable-based case. It is particularly based on the assumption that the matrix does not contain large *positive* off-diagonal entries. However, for many matrices arising in practice, this does not hold. Section 4.2.1.2 indicates appropriate extensions of standard coarsening to deal with such situations.

For typical applications, standard coarsening gives ratios $\frac{|C|}{|\mathcal{V}|}$ between 0.25 and 0.5. Regarding the efficiency of the overall approach, it is often worthwhile to reduce this further. For this purpose, *aggressive coarsening* has been introduced in [48][7]. The explanation of this accelerated coarsening strategy completes Section 4.2.1.

Also in the core part of the coarsening phases for the unknown- and point-based approaches, variable-based coarsening algorithms are used, applied to suitable matrices. This is discussed in Sections 4.2.2 and 4.2.3.

[7] based on an idea in [71].

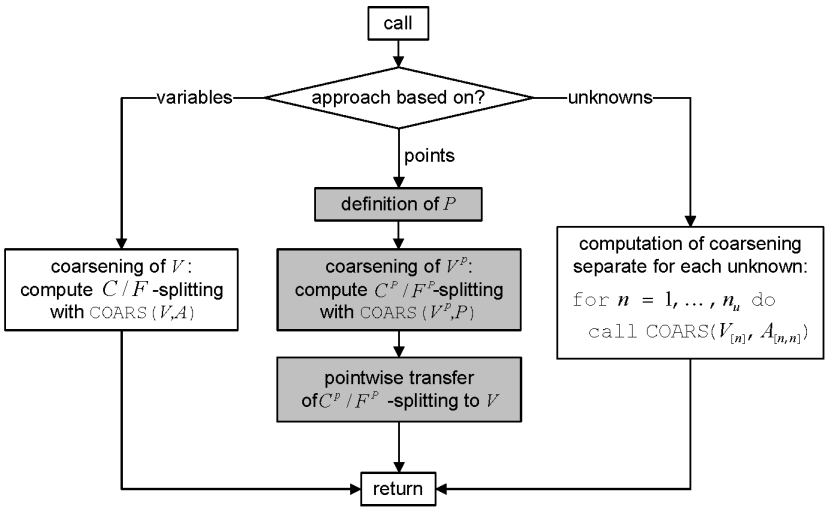


Figure 4.2: Overview of SAMG's coarsening process.

Remark 4.6 Note that for coordinates-based coarsening a point-based approach has to be chosen. Point-based coarsenings for scalar problems are discussed in Section 4.2.3.2 ▲

The overall coarsening phase of SAMG can be depicted as shown in Fig. 4.2. The variable-based coarsening scheme - including all features mentioned above and discussed in Section 4.2.1 - is written in function-like style as COARS$(\mathcal{V}, M)$, where $\mathcal{V}$ denotes the set of variables which shall be split and M the matrix the coarsening algorithm is applied to.

4.2.1 Variable-Based Coarsening

We now recall heuristic criteria for the definition of C/F-splittings and appropriate algorithms for their computation. Since the quality of a C/F-splitting cannot be seen independently of the interpolation, our criteria are motivated by "consequences" of conditions on interpolation such as (3.33).

We start with the "ideal case" of symmetric essentially positive type matrices and their special case Stieltjes matrices since the development of the *standard coarsening algorithm* [71] has been oriented on the last case.

4.2.1.1 The Standard Coarsening Algorithm

Recall from Theorem 3.5 the τ-property of *direct* interpolation for symmetric essentially positive type matrices:

$$\sum_{j \in P_i} |a_{ij}^-| \geq \frac{1}{\tau} \sum_{j \in N_i} |a_{ij}^-| \tag{4.2}$$

for each $i \in F$. Note that *direct* interpolation means $P_i \subseteq C \cap N_i^-$.

The magnitude of τ directly determines the upper bound $\sqrt{1-\sigma/\tau}$ for convergence of the two-level method SK (see Theorem 3.3). Obviously, the smaller $\tau(>1)$, the smaller is this upper bound. On the other hand, the smaller τ, the larger the number of variables $j \in P_i$ for each $i \in F$ has to be in order to fulfill the inequality above. Because of $P_i \subseteq C$, this is directly related to the complexities c_g and c_A.

One extreme case, namely $P_i = N_i$, can be seen to lead to a direct solver (see [87]). However, this approach is very expensive and thus not practical at all. The other extremum, $|P_i| = 1$ for all i, essentially leads to piecewise-constant interpolation[8]. Generally, this is too inaccurate to be used directly. We seek for a compromise between both extrema which yields both a sufficient interpolation and an acceptable complexity.

The sum on the left-hand side of inequality (4.2) "profits" from matrix entries $a_{ij} < 0$ the absolute value of which are large compared with the other off-diagonal entries $a_{ij} < 0$. Such couplings are said to be *strong*. To be more concrete, we recall from (3.24) that a variable v_i is **strongly (negatively) coupled** (**strongly n-coupled**) to a variable v_j if the following holds

$$-a_{ij} \geq \epsilon_{\mathrm{str}} \max |a_{ik}^-| \tag{4.3}$$

with a $\epsilon_{\mathrm{str}} \in [0;1]$. Note that the relation of being strongly coupled is not symmetric.

Remark 4.7 A standard value for ϵ_{str} is 0.25. Sometimes different values may make sense. An example where a larger ϵ_{str} is one possibility to find the correct direction of smoothness has been discussed in Section 3.2.3.4. However, this is not typical. ▲

Remark 4.8 In general, if coarsening is based on this notion of strong connectivity, positive off-diagonal entries, if any, should be small since we declare all positive connections, regardless of size, as weak here. ▲

Obviously, we can efficiently decrease τ only by putting variables j corresponding to strong couplings in P_i. Since errors are algebraically smooth in the direction of strong negative couplings (for $A \in \mathcal{A}_{\mathrm{wdd}}$), this essentially means that coarsening is in the direction of smoothness then.

SAMG's first step in the coarsening phase is the computation of the set S_i of strong couplings for each variable $i \in \mathcal{V}$. Given a concrete definition of strong coupling, the definition of S_i is:

$$S_i := \{ j \in \mathcal{V} |\, i \text{ is strongly coupled to } j \} \ . \tag{4.4}$$

The set of all (i, S_i) defines a **pattern of strong and weak couplings (SW-pattern)** of A. This SW-pattern is not only needed for constructing the set C but also for computing the interpolation, as will be discussed in Section 4.3. Computing the SW-pattern is performed by "sorting" the entries within each matrix row. A corresponding **SW-sort algorithm**, applied to a matrix M, is denoted in function-like style as $\mathsf{SORT}(M)$ and returns the SW-pattern of M. If the sorting is based on (4.3), we obtain the **standard SW-sort algorithm**, denoted by $\mathsf{SORT}_{\mathrm{std}}(M)$.

[8] and corresponds to what is done in aggregation-based AMG.

Remark 4.9 During this algorithm, the following two degenerate cases have to be handled. Variables with couplings only to *positive* off-diagonal entries are marked as **forced C-variables (FC-variables)** in the standard SW-sort algorithm. As already mentioned in Remark 3.6, variables corresponding to strongly diagonally dominant[9] rows will always become F-variables with "empty"interpolation formulas (3.23): $w_{ij} \equiv 0$. In general, such variables are called **forced F-variables (FF-variables)**. Obviously, all other potential F-variables will have at least one strong n-coupling so that their $S_i \neq \emptyset$. ▲

According to the considerations made above and corresponding discussions in Section 3.2.3, the sets P_i of interpolatory variables *should* have the following properties:

P1: Necessary for *direct* interpolation: $|P_i| > 0$, that is, for each $i \in F$, there exists at least one variable from which it can be interpolated.

P2: Each $P_i \cap S_i$ should be reasonably large to ensure a small τ.

P3: The variables in P_i should "surround" the variable v_i in order to provide the basis for a reasonably good interpolation. Geometrically speaking, one-sided interpolation should be avoided as much as possible [10].

P4: The $|P_i|$ should be as small as possible in order to reduce computational cost and to preserve sparsity.

Recalling that all $P_i \subseteq C$ [11], the following set of heuristic criteria for constructing the C/F-splitting "corresponds" to the set P1-P4:

C1: (For *direct* interpolation:) Each F-variable i should have a strong connection to at least one C-variable: $S_i \cap C \neq \emptyset$.

C2: Each $C \cap S_i$ should be reasonably large.

C3: C-variables should "surround" the F-variables which interpolate from them.

C4: $|C|$ should be as small as possible.

In particular, criteria C2 and C4 cannot both be satisfied at the same time so that a suitable compromise is necessary. This can be achieved by the following criterion: *Prefer variables with many variables strongly coupled to* them *to be put into* C. A measure of the importance of a variable i being in C is therefore the cardinality of the **set** S_i^T **of its strong transpose couplings** defined as

$$S_i^T := \{ j \in \mathcal{V} |\, i \in S_j \} \ . \tag{4.5}$$

Note that $j \in S_i^T$ does not imply $i \in S_j^T$ in general. The above criterion can now be formulated as follows:

C5: Variables with large $|S_i^T|$ should be put into C.

[9] defined via one of SAMG's user parameters corresponding to the δ in (2.26), applied to one row at a time.
[10] for illustrations, see [87].
[11] Implications of this and other cases are discussed in Sections 4.2.1.3 and 4.3.1.

In practice, the accuracy of interpolation (in the sense of C3, in particular) and thus the convergence and the h-independence of complete V-cycles can often substantially be improved - without sacrificing complexity - by arranging the C/F-splitting carefully. As a rule of thumb, the set of C-variables should approximately build a set with the following properties:

- The set of C-variables should build a **maximally independent set (MIS)**. "Independent" means that the C-variables are not strongly coupled among each other:

  C6: For variables $i, j \in C$ we should have $i \notin S_j$ and $j \notin S_i$: C shall be an **independent set**.

  "Maximally independent" means that, if any of the F-variables was made a C-variable, the independence of the set would be violated.

- Among the sets with the MIS-property, choose a **maximal** one. Such a set is said to have the **max-MIS property**.

This can be ensured to a large extent by using the coarsening algorithm presented now.

The **standard splitting algorithm**[12] is a reasonable compromise which emphasizes the "quality" of interpolation more than a reduction of the grid complexity. The algorithm is depicted in Fig. 4.3. In function-like style, it is denoted by $\texttt{SPLIT}_{\mathrm{std}}(\mathcal{V}, \texttt{SORT}(M))$ and returns the sets C and F. Here, $\mathcal{V}$ is the index set which will be split into C and F, and $\texttt{SORT}(M)$ is the SW-pattern of the matrix M the splitting is based on. This is A in VAMG, each of the $A_{[n,n]}$ in UAMG, and $\boldsymbol{P}$ in PAMG.

The splitting process is based on the following *measure of importance* of a variable to become a member of C at the current stage of the algorithm:

$$\lambda_i := |S_i^T \cap U| + 2|S_i^T \cap F| \tag{4.6}$$

where U denotes the set of variables which are "undecided" yet. Note that we have $\mathcal{V} = U \dot\cup C \dot\cup F$ at any time. Obviously, at the beginning, λ_i is largest for variables with many U-variables strongly coupled to them [13]. Dependent variables (dependent in the sense of criterion C6) are put in F. At later stages of the coarsening process, variables are favored with many already decided F-variables strongly coupled to them [14]. Note that only at the beginning λ has to be computed "globally". In subsequent steps, only local updates are necessary. That is, for instance, if the status of a variable has changed from U to F, its own λ-value is reset to zero and, for all $j \in S_i$, the λ_j-value is raised by 1.

Standard splitting thus favors variables with a large $|S_i^T|$ to become C-variables according to criterion C5. At the same time, it aims at arranging the C/F-splitting in such a way that the C-variables are independent from each other (criterion C6) and even approximately build a max-MIS set. Hence, standard coarsening aims at building C/F-splittings with the **max-MIS-property**. Practice has shown that this often fulfills criterion C3, at least approximately.

Remark 4.10 None of the C-variables is strongly coupled to any of those C-variables created prior to itself in the coarsening process described above. However, since the relation of being

[12] introduced in [71] as "preliminary C-variable choice" and called "standard coarsening process" in [87].

[13] towards criterion C5 and, indirectly, criteria C2 and C4 with a preference of C4.

[14] again towards criterion C5 and, indirectly, criteria C2 and C4, but with a preference of C2 now.

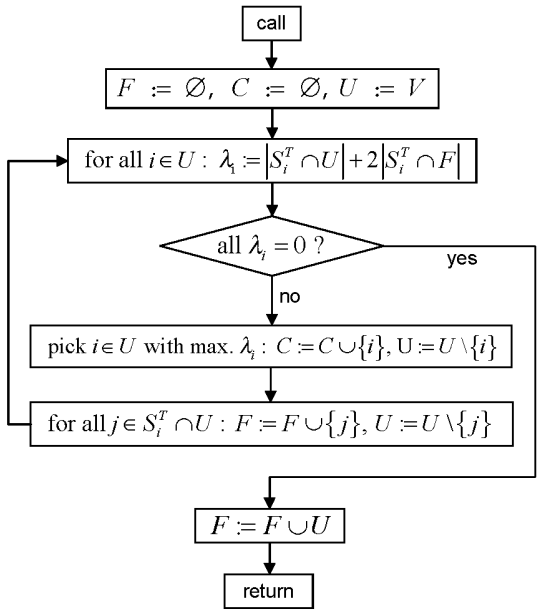


Figure 4.3: The standard splitting algorithm $\texttt{SPLIT}_{\text{std}}$.

strongly coupled is not necessarily symmetric, criterion C6 does not necessarily hold in a strict sense. ▲

Remark 4.11 Before the last step, namely $F := F \cup U$, of $\texttt{SPLIT}_{\text{std}}$ (see Fig. 4.3) is performed, it may indeed happen that the status of some variables remains "undecided", that is the set U is not empty. It is easy to see that such remaining variables i have all the following properties:

- $\lambda_i = 0$. Therefore, no F-variable is strongly coupled to i, and the undecided U-variables are not strongly coupled among each other.
- i is not strongly coupled to a C-variable since i would have become an F-variable otherwise.
- i is strongly coupled to at least one variable since it would have become an FF-variable otherwise.

It follows that i can have strong connections only to F-variables and is strongly coupled to at least one of them. We have at least the following two possibilities now. We can put each of the remaining $i \in U$ into C, or we can put each of them into F and interpolate it from the F-variables to which it is strongly coupled. The latter is done in $\texttt{SPLIT}_{\text{std}}$. How interpolation formulas for such variables are created, is discussed in Section 4.3.1.1. ▲

We now define the **standard coarsening algorithm** for a matrix M as follows:

$$\texttt{COARS}_{\text{std}} := \texttt{COARS}_{\text{std}}(\mathcal{V}, M) := \texttt{SPLIT}_{\text{std}}(\mathcal{V}, \texttt{SORT}(M))$$

SORT$_{\mathrm{std}}$ is one possibility for an SW-sort algorithm SORT. An important extension, used as a default in SAMG, is discussed in the following section. Important other variants for COARS, implemented in SAMG, are discussed thereafter.

4.2.1.2 Treatment of Positive Couplings

In practice, we often have to deal with matrices which are not in $\mathcal{A}_{\text{ess}}$. In fact, they are often not symmetric, and in many cases at least some large positive couplings occur. Since the standard SW-sort algorithm explained above defines the the strength of connectivity according to (4.3), *all* positive couplings, regardless of their size, are defined as weak. Convergence might suffer considerably from completely ignoring positive off-diagonal entries, in particular if they are of comparable size or even larger than the negative ones.

One reason may be that it is no longer sure that error will be smooth in the direction of large negative connections. A simple remedy for this important case (for an example, see Section 3.2.3.4) is as follows: SAMG defines a positive coupling as large if

$$a_{ij} > \epsilon_{\mathrm{lpos}} \max |a_{ik}^-| \ . \tag{4.7}$$

The threshold value ϵ_{lpos} can be defined by the user, a typical value being 0.2 which is used in SAMG as a default. Now, before a decision on the strength of connectivity is made, large positive couplings are eliminated (see Section 3.2.3.4). That is, for each i, we (locally) eliminate all strong positive couplings a_{ij} by means of the j-th equation:

$$e_j \rightarrow -\sum_{k \in N_j} a_{jk} e_k / a_{jj} \ . \tag{4.8}$$

A new equation for e_i results:

$$\widehat{a}_{ii} e_i + \sum_{j \in \widehat{N}_i} \widehat{a}_{ij} e_j = 0 \quad \text{with} \quad \widehat{N}_i = \{ j \neq i \,|\, \widehat{a}_{ij} \neq 0 \} \ .$$

After this elimination,

$$-\widehat{a}_{ij} \geq \epsilon_{\mathrm{str}} \max |\widehat{a}_{ik}^-| \tag{4.9}$$

is used to decide which couplings are strong. Other approaches, realized in SAMG, taking positive couplings into account are described in [89].

4.2.1.3 Aggressive Coarsening

For many (scalar) PDE applications, only a few nonzero entries per matrix row are typical. Unfortunately, in such cases, standard coarsening and its modifications, as discussed above, might produce C/F-splittings with a grid size reduction of only $|C|/|\mathcal{V}| \approx 0.5$. Usually, this results in Galerkin matrices for the second level which have more entries than the finest-level A. For instance, as shown in [87], standard coarsening for the isotropic 7-point stencil on a regular 3D mesh produces a first-level C which corresponds to the "black" points of a "red-black" grid, so that the Galerkin operator on the second level corresponds to a 19-point

stencil, and the second-level matrix contains approximately 1.36 times more entries than the finest-level matrix A. Although subsequent coarsening will typically become faster, because the enlarged matrix typically contains "more" strong connections, the first coarsening step significantly influences the overall complexities c_g and c_A.

The main reason for such an insufficient coarsening is that the algorithm discussed above relies on *direct* strong connections, that is strong connections of F- to C-variables. However, an incorporation of the *indirect* strong connections, that is strong F-to-F connections, can substantially reduce the complexity. For a demonstration, consider the standard 5-point stencil (2D-Laplace). Standard geometric $h \rightarrow 2h$ coarsening would give a reasonable grid size reduction of $|C|/|\mathcal{V}| \approx 0.25$. In the resulting grid, half of the F-points do have strong couplings only to other F-points. However, all F-points can be interpolated if we allow bilinear interpolation. This means, first all F-points with a strong connection to C will be interpolated, afterwards the remaining F-points from their strong connections to F.

An algebraic analog and extension of this process is the reduction of complexity by means of so-called **aggressive coarsening**[15]. As indicated above, we extend the definition of strong connectivity to also include variables which are not directly coupled. In SAMG, the *concept of long-range strong connections* [71] is used: A variable i is said to be **strongly coupled to a variable j along a path of length l** if there exists a sequence of variables $i_0, i_1, \ldots, i_l$ with $i = i_0$ and $j = i_l$ such that $i_{k+1} \in S_{i_k}$ for $k = 0, 1, \ldots, l-1$. With given values $p \geq 1$ and $l \geq 1$, we then define a variable i to be **strongly coupled to a variable j w.r.t. (p, l)** if at least p paths of length $\leq l$ exist such that i is strongly coupled to j along each of these paths (in the above sense). However, it usually does not pay to exploit strong connectivity in this generality. The cases $(p, l) = (2, 2)$ and $(p, l) = (1, 2)$ have turned out to be the most efficient so that only these two variants are considered here.

For the implementation, the **set of strong couplings w.r.t. (p, l)**,

$$S_i^{p,l} := \{j \in \mathcal{V} \mid i \text{ strongly coupled to } j \text{ w.r.t. } (p, l)\} \tag{4.10}$$

might directly be used instead of S_i in $\texttt{SPLIT}_{\text{std}}$. But even for $S_i^{2,2}$ and $S_i^{1,2}$, this would require a substantial extra overhead because the computation and storage of the complete connectivity information contained in $S_i^{p,l}$ and also its transpose $(S_i^{p,l})^T$ are necessary for each i. However, basically the same coarsening can be achieved by applying the standard coarsening algorithm $\texttt{COARS}_{\text{std}}$ twice (for $l = 2$) instead. That is, aggressive coarsening $\texttt{COARS}_{\text{agg}}$ proceeds as follows:

1. Carry out $\texttt{SPLIT}_{\text{std}}(\mathcal{V}, \texttt{SORT}(A))$.
2. Only the resulting set of C-variables is thinned out further. For that purpose, we define strong n-connectivity only between the C-variables (via neighboring F-variables), that is, for each $i \in C$, (4.10) is replaced by

   $$\hat{S}_i^{p,l} := \{j \in C \mid i \text{ strongly coupled to } j \text{ w.r.t. } (p, l)\} \ .$$

   With these sets constituting the new SW-pattern $\texttt{SORT}_C$ for the "old" C-variables the procedure $\texttt{SPLIT}_{\text{std}}(\texttt{C}, \texttt{SORT}_C)$ is applied now. The resulting set of "new" C-variables will then be used as the set of variables on the next coarser level.

[15] together with the so-called **multi-pass interpolation** which is explained in Section 4.3.1.3.

In the following, we refer to the aggressive coarsening strategies based on $\hat{S}_i^{1,2}$ and $\hat{S}_i^{2,2}$ as **A1-** and **A2-coarsening**, respectively.

Remark 4.12 It hardly ever pays to employ aggressive coarsening on more than the finest level since on coarser levels standard coarsening is usually fast enough. Clearly, A1- is faster than A2-coarsening. Illustrations of both types are shown in [87]. Note that generally A2-coarsening is effective only in (at least) "plane-wise" isotropic areas while A1-coarsening is also effective in strongly anisotropic parts of the problem. For the numerical tests reported in this thesis, always the A1-variant has been used. ▲

Remark 4.13 In order to show the differences between standard ("std.") and A1-coarsening ("A1"; only on the finest level), we summarize values for c_g and c_A as presented in [87] for several typical scalar applications (1 = 2D diffusion problems on structured grids, 2 = an industrial test case from 2D CFD simulation (pressure-correction method), 3 = two industrial test cases from 3D CFD simulation (pressure-correction method), 4 = 3D diffusion problems on unstructured grids with strongly discontinuous coefficients[16]):

	(std.)		**(A1)**	
application class	c_A	c_g	c_A	c_g
1	≈ 2.4	≈ 1.7	≈ 1.5	≈ 1.2
2	≈ 2.4	≈ 1.7	≈ 1.4	≈ 1.2
3	$\in [\,2.8,\ 3.4]$	$\in [\,1.5,\ 1.6]$	$\in [\,1.4,\ 1.5]$	$\in [\,1.1,\ 1.2]$
4	$\in [\,2.5,\ 2.9]$	$\in [\,1.6,\ 1.8]$	$\in [\,1.4,\ 1.8]$	$\in [\,1.1,\ 1.3]$

4.2.2 Unknown-Based Coarsening

The coarsening scheme `COARS` for the variable-based case can be employed for the unknown-based case as well, with all of its features. However, as discussed in Section 3.3.1.2, the unknowns are coarsened separately. Within SAMG, this means that `COARS(`$\mathcal{V}_{[n]}$`,` $A_{[n,n]}$`)` is performed for each $n = 1, \ldots, n_u$ separately (see Fig. 4.2).

An important characteristic of this procedure is that during the coarsening phase all unknown cross-couplings are completely ignored, regardless of sign and size. The range of applicability of UAMG has been discussed in Section 3.3.3. If applicable, an advantage is that unknown-based coarsening is among the cheapest coarsenings for the system's case.

Remark 4.14 It should be noted that the submatrices $A_{[n,n]}$ are never explicitly set up in SAMG. Instead, the vector `iu` is used to identify the corresponding matrix entries. ▲

4.2.3 Point-Based Coarsening

We recall from Section 3.4.2 and Fig. 4.2 the basic steps of coarsening in case of PAMG approaches, namely the setup of the primary matrix $\boldsymbol{P}$ and point-coarsening. All variants implemented in SAMG for the setup of $\boldsymbol{P}$ are described in Section 4.2.3.1. Important remarks on SAMG's implementation of point-coarsening are made in Section 4.2.3.2. Also the special case of point-coarsening for scalar problems is discussed there.

[16] [87] presents results, e.g., for a test case from industrial oil reservoir simulation based on a streamline method.

4.2.3.1 Definition of the Primary Matrix

Connectivity and Sparsity Pattern of $\boldsymbol{P}$ A primary matrix suitable for the problem which shall be solved should reflect two things, namely, which points are connected, and how strong these connections are. Which points are connected is described by the *connectivity pattern* of $\boldsymbol{P}$. As discussed in Section 3.4.2.1, several such patterns can be considered. In SAMG, the *maximal* or an *unknown-pattern* can be chosen. Formally, the n-th u-pattern can be selected only if it is *complete* in the sense that the n-th unknown is represented at all points.

It has to be noted that, in SAMG, patterns are actually derived from the *sparsity pattern* of the matrix A and are only sparsity patterns, $\Sigma_s(\boldsymbol{P})$, themselves. That is, they might contain zeros and are thus only supersets of the actually desired connectivity patterns $\Sigma_c(\boldsymbol{P})$. An obvious drawback is that zero entries stored in A or $\boldsymbol{P}$ waste memory. However, searching and expunging zeros from A would mean extra computational effort and an interference in the data structure and is therefore left to the user. Ideally, the two types of patterns coincide.

Implemented Types of Primary Matrices $\boldsymbol{P}$ In Section 3.4.2, several types of primary matrices have been introduced, and areas of applicability have been discussed for each of them. The following list shows which types of primary matrices can automatically be constructed by SAMG.

- **norm-based primary matrix:** According to the discussions in Sections 3.4.2.3 and 3.4.4, the variants (3.72) and (3.73),

$$p_{kl} = -||A_{(k,l)}|| \quad \text{and} \quad p_{kk} = ||A_{(k,k)}|| \tag{4.11}$$

$$\text{and} \quad p_{kl} = -||A_{(k,l)}|| \quad \text{and} \quad p_{kk} = -\sum_{l \neq k} p_{kl} \tag{4.12}$$

  respectively, have been implemented. Selectable are the maximum, row sum or Schur norm. These variants and norms are favored because the computation of the corresponding entries p_{kl} of $\boldsymbol{P}$ is cheap and much cheaper than for variants (3.74) and (3.75) or for more expensive matrix norms as, for example, the Euclidean matrix norm. In addition, possible problems with non-square $A_{(k,l)}$ are avoided.

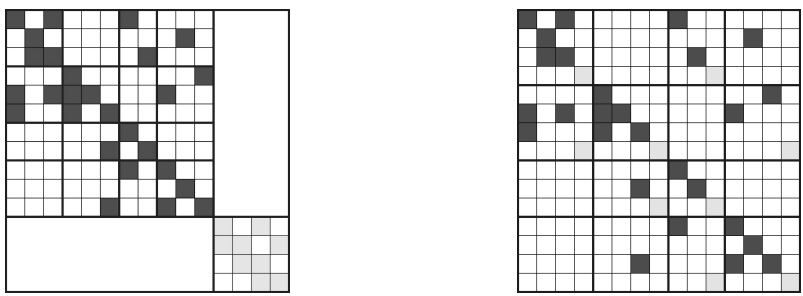
- **coordinates-based primary matrix:** Two variants are implemented. The entries can be computed based on distances or positions of the points. The corresponding formulas have been described in Section 3.4.2.4. The only prerequisites for both variants are that (two- or three-dimensional) coordinates are available for all variables, and that one of SAMG's user-accessible subroutines, `samg_user_coo`, provides these data accordingly.

- **unknown-based primary matrix:** This is the simplest type possible. Since $\boldsymbol{P} = A_{[n,n]}$ (for an $1 \leq 1 \leq n_u$), $\boldsymbol{P}$ needs not to be computed or stored additionally (see Section 3.4.2.2). The only technical requirements for this $\boldsymbol{P}$ are that the n-th unknown-pattern has been chosen, (i.e. n has been chosen to be the primary unknown), and that the underlying connectivity pattern is complete. It depends on the application whether such a simple primary matrix makes sense.

The next paragraph describes how the user can provide a primary matrix to SAMG himself.

User-Provided Primary Matrix As has been mentioned in Section 3.4.2.5, the user himself can provide any reasonable primary matrix to SAMG. There are two ways to submit the corresponding data.

- One way is to augment the original matrix A by an $(n_p \times n_p)$-matrix $A_{[n_u+1,n_u+1]}$ (see Fig. 4.4), and the vectors v and b by vectors $v_{[n_u+1]}$ and (an arbitrary) $b_{[n_u+1]}$ of length n_p. Additionally, the k-th new variable must be attached to point $\mathcal{P}_k$ (by augmenting the vector `ip` correspondingly), and the new variables must be attached to an artificial unknown with number $n_u + 1$ (by augmenting the vector `iu`), so that the resulting $n + 1$-th u-pattern is complete. The new unknown can then obviously serve as a primary unknown and is called the **dummy unknown**.



(a) (b)

Figure 4.4: An augmented system: The "black part" represents the original matrix A (already point-wise ordered), the "part in light grey" the submatrix $A_{[n_u+1,n_u+1]}$ belonging to the dummy unknown. Shown are the augmented system (a) before and (b) after a point-wise ordering.

In the current SAMG realization, the augmented system must be reordered point-wise, as in Fig. 4.4(b), before a point-based SAMG approach can be applied. Then, we can simply choose $A_{[n_u+1,n_u+1]}$ to be the primary matrix.

Remark 4.15 Since the dummy unknown does not have couplings to other unknowns, it is possible and natural to exclude it from the solution phase and, in particular, from the computation of residuals in order to avoid misleading results. This corresponds to applying the solution phase only to the original matrix A. To use this exclusion process, we just have to declare the unknown $n_u + 1$ to be "dummy" by setting an SAMG parameter properly. ▲

- The second possibility to provide a primary matrix to SAMG is to implement a new setup routine for $\boldsymbol{P}$. SAMG provides a corresponding user-interface. In this case, not only $A_{[n_u+1,n_u+1]}$ must be defined, but also all coarse-level primary matrices.

Remark 4.16 Note that internally defined primary matrices (e.g. norm- or distance-based) are usually constructed "from scratch" on each level. Therefore, such coarse-level primary matrices $\boldsymbol{P}_k$ are hardly ever identical to the corresponding Galerkin operators which depend on $\boldsymbol{P}_1$ and the interpolation. In general, this is the main difference between an SAMG approach with internally defined $\boldsymbol{P}_k$ and an approach where SAMG is applied to a system which has "physically" been augmented by the finest-level matrix $\boldsymbol{P}_1$. ▲

4.2.3.2 Point-Coarsening

Point-coarsening consists of coarsening the set of points $\mathcal{V}^p$ by means of `COARS(`$\mathcal{V}^p$`,`$\boldsymbol{P}$`)` and transferring the resulting C^p/F^p-splitting to the set $\mathcal{V}$ to obtain its C/F-splitting. All variants implemented for variable-based coarsening (see Section 4.2.1) can also be employed for `COARS(`$\mathcal{V}^p$`,`$\boldsymbol{P}$`)`.

The **transfer** then proceeds in two steps. In the first step, all variables belonging to points in C^p (F^p) are put into C (F). In the second step, it is checked whether some variables shall become (additional) forced F- or forced C-variables. This can be the case if `ic_set` has been set correspondingly by the user, or if an MU-interpolation has been selected. In the latter case, an SW-pattern for A has been computed, and the `SORT(`A`)` algorithm may have returned (additional) forced C- or F-variables. If, according to such additional information, a variable shall be forced into F, this variable - but not automatically the whole point - is forced into F. However, if a variable shall be forced into C, the whole point is forced into C. This takes precedence over possible forced F-variables.

Also in the special case of **scalar problems**, a distance-based or position-based $\boldsymbol{P}$ can be chosen. Possible applications include reaction-diffusion equations (cf. Section 3.10). It should be noted that a norm-based $\boldsymbol{P}$ does not make sense since it would only result in flipping signs of all positive off-diagonal entries. However, special variants sorting out or modifying positive off-diagonals might be appropriate for special applications and can be implemented via the user interface.

Remark 4.17 (Notation): The type of coarsening used - "std" or "agg" - is added as first parameter in the list defining the AMG approach, e.g. VAMG(std,...) for variable-based AMG with standard[17] coarsening, UAMG(agg,...) for an unknown-based variant with aggressive coarsening. In the case of a point-based approach, we add the type of primary matrix before the "type" of coarsening. We write "$A_{[n,n]}$" for $\boldsymbol{P} = A_{[n,n]}$, "ns" for norms (3.73), "nf" for norms (3.72), "dist" for distances, and "pos" for positions [18] . Example: we write PAMG(ns,std,...) for PAMG with a primary matrix based on norms (3.73) and standard coarsening. ▲

[17] Note that positive off-diagonals are treated as described in Section 4.2.1.2.

[18] In the numerical tests presented, we only use variant A1 in case of aggressive coarsening and only on the finest level and always choose the maximal pattern for all $\boldsymbol{P}$ except of $\boldsymbol{P} = A_{[n,n]}$.

4.3 Interpolation

After having computed the C/F-splitting for level k successfully, the algorithm proceeds with constructing the interpolation formulas for all F-variables. An overview of the interpolation process is depicted in Fig. 4.5. It shows that the main branching is due to the overall type of AMG approach employed, that is variable-, unknown- or point-based. Figs. 4.5 and 4.6 indicate that in all three branches the same "scheme" `VINT(`F`,`C`,`$\mathtt{t}_{\mathrm{pat}}$`,`$\mathtt{t}_{\mathrm{weights}}$`)` is employed[19]. `VINT` computes interpolation formulas for a given C/F-splitting, based on an SW-pattern `SORT(`M`)` for a matrix M (specified by $\mathtt{t}_{\mathrm{pat}}$=$M$), and with interpolation weights of the type $\mathtt{t}_{\mathrm{weights}}$.

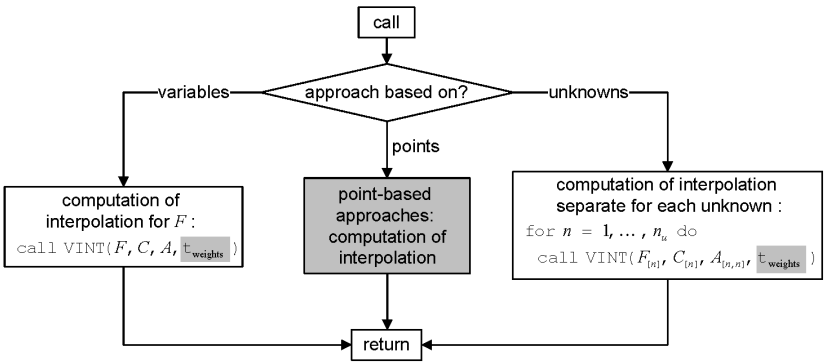


Figure 4.5: Overview of interpolation.

In the following section, we explain interpolation schemes for VAMG and thus directly the main features of `VINT`. Afterwards, only the interpolation schemes for PAMG need to be discussed in more detail.

4.3.1 Variable-Based Interpolation

In this section, we assume for simplicity that we want to compute an interpolation operator for our basic system $Av = b$ in case of a variable-based approach. For other matrices, for example $A_{[n,n]}$ or $\boldsymbol{P}$, all possibilities discussed here work analogously.

All variable-based interpolation schemes considered in this thesis are derived from the following basic formulas for the interpolation weights,

$$w_{ij} = \begin{cases} -\alpha_i a_{ij}/a_{ii} & (j \in P_i^-) \\ -\beta_i a_{ij}/a_{ii} & (j \in P_i^+) \end{cases} \tag{4.13}$$

$$\text{with} \quad \alpha_i = \frac{\sum_{j \in N_i^-} a_{ij}}{\sum_{j \in P_i^-} a_{ij}} \quad \text{and} \quad \beta_i = \frac{\sum_{j \in N_i^+} a_{ij}}{\sum_{j \in P_i^+} a_{ij}} \,. \tag{4.14}$$

[19] `VINT` stands for variable-based interpolation.

These formulas have been derived from the τ-properties (3.38) and (3.39) in Section 3.2.3. The different schemes mainly differ in the way strong F-to-F couplings are treated. In Sections 4.3.1.1 to 4.3.1.3, the variants implemented in SAMG are explained, starting with the simplest variant, the so-called *direct interpolation*, continuing with *(extended) standard interpolation* and concluding with *multipass interpolation*. Afterwards, possibilities to incorporate other kinds of interpolation weights, for example based on coordinates, are explained. The realization of different means to improve efficiency, namely smoothing of interpolation, scaling and truncation, is explained in Section 4.3.1.5.

Remark 4.18 It should be noted in advance that in case of aggressive coarsening only multipass interpolation can be used. ▲

4.3.1.1 Direct Interpolation

Direct interpolation assumes that the C/F-splitting has been constructed by means of standard coarsening. At the beginning, the sets P_i of interpolatory variables are set to

$$P_i = P_i^- \cup P_i^+ \quad \text{with} \quad S_i^- = S_i \cap N_i^- \quad , \qquad S_i^+ = S_i \cap N_i^+ \quad , \tag{4.15}$$

$$P_i^- = S_i^- \cap C \quad , \qquad P_i^+ = S_i^+ \cap C \quad , \tag{4.16}$$

based on the sets S_i constructed by SORT. Then, the interpolation weights are directly computed by means of the equations above. Due to the definition of the sets P_i^- and P_i^+, the interpolation formulas (4.13) are well-defined if $P_i \neq \emptyset$. However, two exceptional cases can arise for particular F-variables i (see also Remarks 4.9 and 4.11 in Section 4.2.1.1):

- $P_i = \emptyset$ and i is an FF-variable. Forced F-variables receive an empty interpolation formula.
- $P_i = \emptyset$ and i is not an FF-variable. Such a variable is called **exceptional F-variable (XF-variable)**. It has at least one strong F-to-F coupling. These are used for interpolation. Hence, the computation of the interpolation formula is postponed until all regular F-variables have been treated. Then, analogously to (4.8), we (approximately) eliminate all e_j with $j \in S_i \subseteq F$ in the equation

$$a_{ii} e_i + \sum_{j \in N_i} a_{ij} e_j = 0 \tag{4.17}$$

by means of the corresponding j-th equations. By applying (4.13) to the resulting equation, but with $P_i^- = S_i^- \subseteq F$ and $P_i^+ = S_i^+ \subseteq F$, the interpolation weights for the XF-variables are then computed.

Remark 4.19 If an XF-variable has strong connections only to FF-variables, an empty interpolation formula would result. Such variables are forced into C. This is also the case for XF-variables which have strong connections only to F-variables for which no interpolation formula has been computed at the current stage of the algorithm. ▲

4.3.1.2 (Extended) Standard Interpolation

A straightforward enhancement of direct interpolation is the inclusion of all *strong* F-to-F couplings in the construction of the interpolation operator. This is called **standard interpolation** - "standard" because it is more robust and usually more efficient than direct interpolation. Standard interpolation is used as a default in SAMG (unless aggressive coarsening has been selected). It assumes that the C/F-splitting is obtained from standard coarsening and is defined in the following way. Analogously to the treatment of XF-variables in direct interpolation, we first (approximately) eliminate all e_j with $j \in F_i^s := F \cap S_i$ in the equation (4.17). By defining P_i as the union of $C_i^s := C \cap S_i$ and all C_j^s ($j \in F_i^s$), we now define interpolation analogously to (4.13).

Extended standard interpolation then denotes the variant where *all* F-to-F couplings are taken into account, regardless if strong or weak. Advantages of these *indirect* interpolations have already been discussed in Section 3.2.3.5. The incorporation of couplings to F-variables increases the quality of the approximation of (4.17). In addition, it contributes to the objective of having F-variables nicely "surrounded" by interpolatory variables.

4.3.1.3 Multi-Pass Interpolation

The previous types of interpolations require that the C/F-splitting has been obtained from standard coarsening. In particular, they require that each F-variable which is not an FF-variable has at least one strong connection. Moreover, for computing the interpolation formula for an F-variable i which has strong connections only to F-variables, they require that at least for one of these F-variables a "regular" interpolation formula has been constructed before. Otherwise, this F-variable i is forced into C (see also Remark 4.19 above). Even in case of standard coarsening, depending on the concrete C/F-splitting (including user-forced F- or C-variables) and the order of the variables, such situations can happen.

Multi-pass interpolation has been designed to "release" the requirements on the C/F-splitting and the order of the variables. If aggressive coarsening has been chosen, SAMG enforces the use of multi-pass interpolation. It proceeds as follows:

1. **First pass:** For all $i \in F$ for which $C \cap S_i \neq \emptyset$ holds, use direct interpolation and define the set F^* to contain all these variables. If $F = F^*$, stop the process.

2. **Next pass:** For all $i \in F \setminus F^*$ for which $F_i^{*,s} := S_i \cap F^*$ is not empty, we base their interpolation formulas on those already computed for $j \in F_i^{*,s}$. For this purpose, we replace in the i-th equation (4.17) all e_j with $j \in F_i^{*,s}$ by the interpolatory term $\sum_{k \in P_j} w_{jk} e_k$. This leads to a new equation for e_i. Defining P_i as the union of all P_j for $j \in F_i^{*,s}$, the interpolation formula for e_i is then computed as in case of standard interpolation. Update F^* by all variables i which have obtained an interpolation formula now.

3. If $F = F^*$ stop, otherwise go back to step 2.

Note that, in order to preserve the locality of interpolation, the update of F^* in each pass is done in a Jacobi and not a Gauss-Seidel fashion.

In principle, multi-pass interpolation can always be selected, but has been designed for a use in connection with A1- and A2-coarsening [20]. If coarsening and interpolation are based on the same matrix[21] the multi-pass process can be seen to terminate after at most 4 passes (besides some very exceptional cases).

Remark 4.20 If multi-pass interpolation has not finished after 4 passes or has even run into a deadlock, the remaining variables without an interpolation formula are forced into C, and (by default) the complete process is started from scratch based on the new C/F-splitting. If again a deadlocks occurs, the remaining variables are again forced into C, but - by default - no new process is initiated. Instead, SAMG continues with the interpolation formulas computed. ▲

4.3.1.4 Types of Weights

So far, we have only considered $\texttt{VINT}(F, C, A, \texttt{t}_{\text{weights}})$ with $\texttt{t}_{\text{weights}}$="A" (see Remark 4.21 below). That means, in particular, that the interpolation *weights* have been based on the entries of A.

Alternatively, if for each variable coordinates are given, then interpolation weights might also be computed via the distances or positions of these grid nodes. Another possibility is to utilize the entries of a suitable matrix B instead of those of A, simply by exchanging a_{ij} by b_{ij} when computing the interpolation weights. If $\mathcal{U}_n$ has been chosen to be the primary unknown in a PAMG approach (but $\boldsymbol{P} \neq A_{[n,n]}$), this matrix B might equal $A_{[n,n]}$. This option is built into SAMG, other choices might be supplied by the user via an interface.

What remains to be explained here is the way interpolation weights are computed based on coordinates. In the distance-based case, during the construction of a particular interpolation weight, the squared reciprocal Euclidean distance $||\pi_i - \pi_j||_E^{-2}$ is used instead of a matrix entry a_{ij}. Instead of scaling by α_i/a_{ii} and β_i/a_{ii} according to (4.13), each weight of a row is scaled by the row sum. In the position-based case, first the interpolation for the distance-based case is computed (without performing a scaling). Afterwards the resulting weights are multiplied by penalty factors as explained in Section 3.4.2.4, and finally each weight of a row is scaled by the row sum. Note that coordinates-based interpolation interpolates constants exactly.

Remark 4.21 (Notation): The type of weights is denoted by the parameter $\texttt{t}_{\text{weights}}$ and appended to VAMG's and UAMG's "parameter list" as follows: we write VAMG(·,$\texttt{t}_{\text{weights}}$) and UAMG(·,$\texttt{t}_{\text{weights}}$). In case of VAMG and UAMG, $\texttt{t}_{\text{weights}}$ can be equal to "A" (entries of matrix A; default), "dist"(ances), or "pos"(itions). For instance, VAMG(agg,dist) denotes a VAMG approach with aggressive coarsening and multi-pass interpolation[22] with weights being based on distances. We make use of the abbreviations VAMG(·) for VAMG(·,A) and UAMG(·) for UAMG(·,A). See Remark 4.24 for the general type of interpolation (direct, (extended) standard, multi-pass) being used. ▲

[20] Corresponding illustrations for a 5-point Poisson stencil can be found in [87].
[21] This is the case for VAMG, UAMG and PAMG with $\boldsymbol{P}$-interpolation.
[22] see also Remark 4.18 above.

4.3.1.5 Improving the Interpolation

(Extended) Standard interpolation is a means to improve direct interpolation. Different possibilities to "improve" any of our interpolation schemes are Jacobi interpolation, scaling, or truncation. All of these three means have a different objective. They can be combined.

Jacobi interpolation[23] can be seen as an a-posteriori improvement of direct, (extended) standard or multi-pass interpolation: first, one of these interpolation schemes is computed. Afterwards μ Jacobi interpolation steps are performed with the interpolation weights constructed in the first step as an initial guess.

In practice, $\mu \leq 2$ is usually enough to improve the convergence without sacrificing the overall performance. Note that per Jacobi interpolation step not only the computing time of the setup phase but also the radius of interpolation is increased which is likely to blow up the complexities. However, since usually a lot of small entries are created, a **truncation**[24] **of interpolation** limits the increase of complexity considerably so that the method works feasibly again. Hence, Jacobi interpolation should always be used in combination with truncation.

Not only Jacobi interpolation, but also standard or multi-pass interpolation tend to increase the *radius* of interpolation and thus the A-complexity. Again, truncation is a feasible remedy in all these cases.

Remark 4.22 Note that truncation of interpolation is a "safe" process whereas, for instance, a truncation of the final Galerkin operator is not. This is due to the fact that the variational principle[25] for K is usually destroyed by modifying the Galerkin operator. ▲

AMG's interpolation schemes interpolate constant functions exactly if A is a zero row sum matrix (unknown-wise). However, for matrices with non-vanishing row sums (near boundaries for example), this is not necessarily the case (unless coordinates-based weights are used). In fact, forcing the constants to be always interpolated exactly might or might not result in a less accurate approximation of algebraically smooth error and thus reduced efficiency.

For matrices with some rows strongly *violating* diagonal dominance, an example being reaction-diffusion equations (cf. Sections 3.1.3.2 and 5.2.2), care has to be taken. In such cases, the interpolation schemes might suffer considerably from "critical" matrix rows. A cheap means to bring back a reasonable convergence in many such cases is **scaling the interpolation**: the original interpolation weights are scaled so that all row sums of I_H^h equal one. Constants are always interpolated exactly then. Experience shows, however, that more robust coarsening and interpolation schemes (for instance elimination of positive entries or Jacobi interpolation) should be tried first.

4.3.2 Point-Based Interpolations

Fig. 4.6 outlines SAMG's realization of interpolation for PAMG. MU-interpolation is completely defined since `VINT` is used, applied to each unknown separately. All features available

[23] For a definition and discussion, see Section 3.2.4.

[24] Truncation is here defined as a dropping of weights which are below a certain user-definable threshold.

[25] holding in case of $A > 0$, see Section 3.1.1.

in the variable-based case can be employed here and, in addition, the interpolation weights can also be based on the entries of $\boldsymbol{P}$ if the maximal pattern has been chosen. A possible application for the latter has been mentioned in Remark 3.33.

In case of the block-interpolation, we still have to define `PINT`. This is done in Section 4.3.2.1. Afterwards, in Section 4.3.2.2, we explain variants for the concrete transfer of interpolation weights to the variables in case of an SU-interpolation. Finally, the special case of scalar applications is discussed in Section 4.3.2.3.

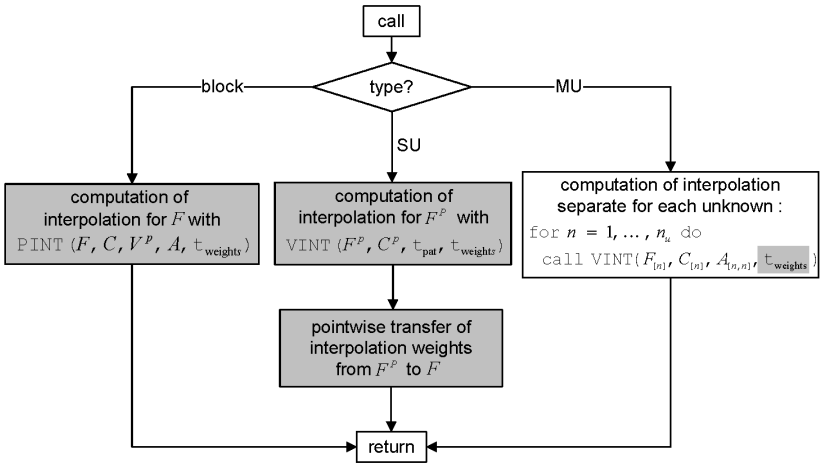


Figure 4.6: Overview of the three general types of interpolation for point-based AMG.

4.3.2.1 Block-Interpolation

Due to the fact that variant (3.93) is likely to produce problems in practice and is considerably more expensive than (3.98), the algorithm `PINT` is based on the variant (3.98). Great advantages of this algorithm are that it can easily handle the case of varying number of variables per point, and that only the $A_{(k,k)}$ have to be inverted. If an $A_{(k,k)}^{-1}$ cannot be inverted numerically, the variables of the k-th point are forced into C.

There are essentially two possibilies for the generalization of "elimination" processes used in `VINT` for an incorporation into block-interpolation. It could be done fully block-wise in a direct analogy to the scalar variants (if $|\mathcal{P}_k|$ is constant), or it could be performed variable-wise before the application of $A_{(k,k)}^{-1}$ (i.e. the solution of the corresponding block-systems) in (3.98). Only the variable-wise variants are implemented in SAMG to limit computational work and avoid problems with a varying number of variables per point.

4.3.2.2 Single-Unknown-Interpolation

In an SU-interpolation, the interpolation weights are most naturally based on the primary matrix $\boldsymbol{P}$ (default). Other possibilities are $A_{[n,n]}$ [26], or coordinates if available.

The SW-pattern, the SU-interpolation is based on, is denoted by VINT's parameter $\mathtt{t}_{\mathrm{pat}}$ (see Section 4.3). The choice $\mathtt{t}_{\mathrm{pat}}$=$\boldsymbol{P}$ is always possible and always used for the numerical tests with SU-interpolation presented in this thesis. Note that, if the n-th unknown-pattern has been chosen, SAMG also allows for $\mathtt{t}_{\mathrm{pat}}$=$A_{[n,n]}$.

The general way to transfer the interpolation weights resulting from VINT has already been explained in Section 3.4.3.3. The only open point here is the concrete transfer of weights in the case that in parts of the simulation domain not each unknown is defined on each point. The following methods are implemented in SAMG:

- "Cheap" method: For the transfer of a concrete w_{kl}^p to a variable v_i belonging to the point $\mathcal{P}_k$ a "position shifting"is attempted: if v_i belongs to the n-th unknown and is the t-th variable on point $\mathcal{P}_k$, check if the t-th variable (v_j) on point $\mathcal{P}_l$ also belongs to the n-th unknown. If this is true, define $w_{ij} := w_{kl}^p$. If not, simply use the first variable on point $\mathcal{P}_l$ (instead of v_j) regardless of its unknown-type. This is possible since C^p does not contain empty points.

- More expensive method: Proceed as in the cheaper variant but search for a suitable variable, i.e. one with the desired unknown-number, within the point. Only if this fails, use the first variable attached to the current point.

There is another option for both methods in case a suitable variable is not found at the point in question. Instead of using the first variable on that point, simply skip the weight (maybe with a rescaling of the remaining weights).

Remark 4.23 Note that skipping can result in empty interpolation formulas for some variables and should be used with care. In general, it is an open question whether unknown cross-couplings should be taken into account in interpolation. Whereas this is fairly natural in the context of block-interpolation, it depends in case of SU-interpolation on the concrete application [27] which of the above methods makes sense, and whether skipping should be allowed or not. ▲

Remark 4.24 (Notation): In all numerical tests, standard interpolation is used in connection with standard coarsening, and multipass-interpolation in connection with aggressive coarsening. This is not explicitly added to the "list" defining the concrete AMG approach.

The parameter $\mathtt{t}_{\mathrm{weights}}$, defined in Remark 4.21, is extended by the choice "P" (primary matrix $\boldsymbol{P}$) in case of a PAMG approach. For instance, in case of $\boldsymbol{P}$-interpolation, we have $\mathtt{t}_{\mathrm{pat}}$=$\boldsymbol{P}$ and $\mathtt{t}_{\mathrm{weights}}$="P", and in case of $A_{[n,n]}$-interpolation, we have $\mathtt{t}_{\mathrm{pat}}$=$A_{[n,n]}$ and $\mathtt{t}_{\mathrm{weights}}$="A" (see also Remark 3.32).

The type of interpolation ("B", "MU", or "SU"), and the value of $\mathtt{t}_{\mathrm{weights}}$ follow the type of coarsening in the parameter list for VAMG, UAMG, PAMG. For instance, we write

[26] possible only if the n-th u-pattern has been selected, different only if $\boldsymbol{P} \neq A_{[n,n]}$.

[27] cf. Section 3.4.3.3.

PAMG(ns,std,SU,P) for a point-based approach with a $\boldsymbol{P}$ based on norms (3.73), standard coarsening, and a $\boldsymbol{P}$-interpolation. Another example: we write PAMG(dist,agg,MU,dist) for a point-based approach with a distance-based $\boldsymbol{P}$, aggressive coarsening, and an MU-interpolation with weights being based on distances.

Note that we do not explicitly add $\mathtt{t}_{\mathrm{pat}}$ to the parameter list since in all numerical tests presented in this thesis SAMG's standard choice of $\mathtt{t}_{\mathrm{pat}}$ is used: $\mathtt{t}_{\mathrm{pat}}$=$A$ for VAMG, $\mathtt{t}_{\mathrm{pat}}$=$A_{[n,n]}$ for UAMG, $\mathtt{t}_{\mathrm{pat}}$=$\boldsymbol{P}$ for PAMG. ▲

4.3.2.3 Point-Based Interpolations for the Scalar Case

In the scalar case ($n_u = 1$), B-, MU- and SU-interpolation are reduced to one interpolation scheme, namely $\mathtt{VINT}(F,C,\mathtt{t}_{\mathrm{pat}},\mathtt{t}_{\mathrm{weights}})$. The parameter $\mathtt{t}_{\mathrm{weights}}$ can be set to the types "A", "P", "dist", or "pos", $\mathtt{t}_{\mathrm{pat}}$ to the types $\boldsymbol{P}$ and A.

4.4 Smoothing, Acceleration, One-Level Solvers

The following smoothers are available in SAMG:

- Jacobi relaxation.
- variable-, unknown, block-wise Gauss-Seidel (VGS, UGS, BGS) relaxation.
- (M)ILU(0): (modified) incomplete LU decomposition without fill-in outside the sparsity pattern of A, see [74]. For ILU(0), we also write ILU.
- ILUT($l_{\mathrm{fill}}, \tau_{\mathrm{droptol}}$) [72, 74]: ILU with a dual-dropping strategy controlled by two parameters, namely the parameter l_{fill}, determining the maximum level of (absolute!) fill-in per row of the incomplete inverse, and the dropping tolerance τ_{droptol}. Entries which are smaller than τ_{droptol} are dropped.
- (M)ILUTP($l_{\mathrm{fill}}, \tau_{\mathrm{droptol}}$) [74]: (modified) ILUT($l_{\mathrm{fill}}, \tau_{\mathrm{droptol}}$) with a column pivoting strategy.

By default, one pre- and one post-smoothing step is performed, both in CF-ordering in case of a Gauss-Seidel variant (see Section 3.2.4.2). Note that, for symmetric matrices, "symmetric" variants of Gauss-Seidel relaxations are employed: in post-smoothing, the variables are passed through in the reverse order of what has been selected for pre-smoothing. This is done in order to obtain a symmetric iteration matrix.

Compared with RAMG, ILU(0) has substantially been accelerated[28], and MILU(0) and (M)ILUTP have been added. The "M" variants add positive values to the diagonal entries of the matrix U of the incomplete LU decomposition in order to prevent ILU(TP) from failing due to zero diagonals in U, see also [59, 74]. This stabilizes ILU(TP) and can be useful, for instance, for matrices containing some (nearly) zero diagonals (see also Section 4.1.1). In Remarks 5.15 and 5.16 we mention planned extensions which might increase the robustness and efficiency of smoothing further for certain applications.

[28] for the price of only one additional working vector with length n_v.

AMG as a Preconditioner For many applications, the convergence of stand-alone AMG approaches is slowed down by just a few eigenvalues of the iteration matrix M which are close to or even larger than 1. Typical eigenvalue distributions for such cases are depicted in Figs. 4.7 (a) and (b) [29]. It is provenly efficient for such situations to employ the AMG method not stand-alone but as a preconditioner. This is particularly true for matrices considerably deviating from the ideal weakly diagonally-dominant Stieltjes-matrix case, as is the case for many industrially relevant PDE systems. But even VAMG applied to a discrete Poisson's equation might profit considerably, in particular, if aggressive coarsening is used.

In SAMG, right-preconditioning is used (see [74], for instance). The one-level iterative method which is preconditioned that way is called **accelerator**. The following accelerators are implemented in SAMG:

- CG: conjugate gradient method (cf. [74]).
- BiCGstab [96]: stabilized biconjugate gradient method. Note that during one BiCGstab iteration *two* AMG cycles are performed.
- GMRES(k) [75, 74]: restarted method of generalized minimal residuals with a Krylov space of dimension k.

For the examples mentioned in Figs. 4.7 (a) and (b), a drastic improvement of convergence is achieved by acceleration (see Table 5.5). Noteworthily, the stand-alone variant with ILU(0) smoothing diverges in contrast to the one with GS smoothing for this case but, when accelerated, the variant with ILU exhibits a better convergence rate than the variant with GS.

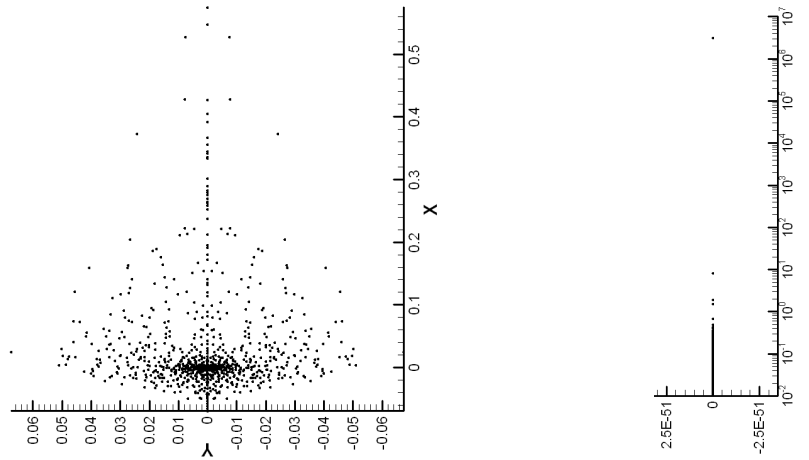
Remark 4.25 The iteration matrix M for a stand-alone AMG method can numerically be constructed by applying, for all $i = 1 \dots n_v$, one cycle of the respective AMG method to the system $Ae^{(i)} = 0$ with $e^{(i)}$ being the i-th standard unit vector ($e_i^{(i)} = 1$, all other components being 0). Combining the resulting approximation vectors $v^{(i)}$ yields the iteration matrix $M = [\, Me^{(1)} \ \dots \ Me^{(n_v)} \,] = [\, v^{(1)} \ \dots \ v^{(n_v)} \,]$. ▲

One-Level Solvers By limiting the number of levels to 1, that is, preventing SAMG from creating a hierarchy, only smoothing and acceleration are performed. This way, several standard iterative methods, namely CG, BiCGstab and GMRES(k), preconditioned by any[30] one of the "smoothers" mentioned above, can be selected. Therefore, SAMG does not only provide a whole AMG environment but also a broad collection of classical one-level solvers.

Remark 4.26 (Notation): The selected smoother precedes, the selected accelerator follows the AMG approach chosen. For instance, ILU-PAMG(dist,std,MU,A)-BiCGstab denotes a PAMG approach with a distance-based $\boldsymbol{P}$, standard coarsening, an MU-interpolation with weights being based on A, ILU(0) smoothing and BiCGstab acceleration. ▲

[29] The corresponding iteration matrices M have numerically been constructed according to Remark 4.25.

[30] Among the reasonable combinations are CG with Jacobi, symmetric VGS/UGS/BGS or (M)ILU(0), and BiCGstab or GMRES(k) with Jacobi, VGS, UGS, BGS, (M)ILU(0) or (M)ILUT(P).



(a) (b)

Figure 4.7: Eigenvalue distributions of exemplary iteration matrices for the SILO2 example (stress simulation, see Section 5.2.1). SAMG approaches employed: (a) GS-UAMG(std), $\rho \approx ARF = 0.532$ for $n_{\rm it} = 36$. (b) ILU(0)-UAMG(std), $\rho = ARF_2 =$ 0.315e+7 (strong divergence!).

4.5 Computational Cost

The computational cost, that is memory requirements and computational work, of AMG1R5's setup phase has been discussed in [71]. That SAMG usually needs *considerably less memory* than AMG1R5, due to reduced complexities, has been demonstrated in [90]. However, the estimates for the computational work remain qualitatively valid so that the work for constructing $\texttt{COARS}_{\rm std}$ and `VINT(`F`,`C`,std,A)` sums up to $n_v\, p_2(s_a, s_p)$ and for the Galerkin operators to $n_v\, p_3(s_a, s_p)$ where $p_m(s_a, s_p)$ denotes a polynomial of in total m-th order in s_a, s_p with small coefficients. Note that typically $s_a = O(c_A/c_g)$ and $s_p < 3$ holds. For aggressive coarsening, more work has to be invested first, but the size ($|A_k|$ and $|\mathcal{V}_k|$) of the following levels is usually considerably reduced. See also Section 4.2.1.3.

The work for UAMG's coarsening and MU-interpolation depends on c_g and the $A_{[n,n]}$-complexities. For the PAMG components, c_p and $c_{\boldsymbol{P}}$ come into in addition to c_g and c_A so that PAMG's memory requirements and computational work are essentially proportional to $c_{\rm AMG}$. The more physical unknowns the system contains and the larger the number of entries in $A_{[m,n]}$ ($m \neq n$) compared to $A_{[n,n]}$, the smaller are the $\Sigma(A_{[n,n]})$ compared to $\Sigma(A)$, and the less matters the additional memory for $\boldsymbol{P}$ - if it has to be stored - relatively to the overall memory requirements. Whereas $\Sigma(A_{[n,n]})$ typically equals $\Sigma(A)$ multiplied by 0.25 up to 0.5 in case of two unknowns, for three unknowns this factor typically lies between $0.\overline{1}$ and $0.\overline{3}$. A similar estimate holds for $\Sigma(\boldsymbol{P}_k)/\Sigma(A_k)$ so that we typically obtain $c_{\rm AMG} = f_A c_A$ with $f_A \in [1.1,\, 1.5]$ (see Table 4.1).

Due to the inversions and incorporation of *all* entries of A_k, B-interpolation is the most

expensive variant. MU-interpolation needs principally the sum of the costs of `VINT` applied to all $A_{[n,n]}$, SU-interpolation is even cheaper. Often, if reasonably working, a cheap PAMG approach is PAMG($A_{[n,n]}$,agg,SU,P) both in terms of memory requirements and computational work. In general, the incorporation of coordinates in one or more components of the setup can add a considerable amount of computing time.

Since usually only (at most) one of VAMG, UAMG, and the PAMG types is reasonable for a fixed problem class, we skip a more detailed comparison of the different approaches.

The work for each cycle is dominated by smoothing and acceleration. The latter only affects the finest-level matrix so that the work for each SAMG accelerator is proportional to $|\Sigma_s(A_1)|$. The total work for smoothing is in principle proportional to the number of entries stored in all matrices, $\sum_{k=1}^{n_{\text{lev}}} |\Sigma_s(A_k)|$. Therefore, $c_A + 1$ is a convenient approximation of the ratio of the total work per V-cycle including the accelerator to the relaxation work on the finest level.

The complexities strongly depend on the nature of the problem and the concrete AMG approach. For a fixed application class and AMG approach, however, they should principally be constant. The costs of both the setup phase and each cycle are $O(n_v)$ then.

As a general rule of thumb, the setup phase needs approximately the same time as 3-10 cycles. This strongly depends on the application class and the concrete approach chosen. For instance, a coordinates-based $\boldsymbol{P}$ adds a considerable amount of work to the setup phase, ILU(0)-smoothing or acceleration a considerable amount of work to each cycle.

Concrete complexity values for SAMG are given in Remark 4.13 for typical scalar applications, in Table 4.1 for our model problems and in the next chapter for industrial test cases.

In all cases, the cost *and* convergence rates for AMG approaches with aggressive coarsening are very reasonable so that very efficient preconditioners result. For (B)GS smoothing, a considerable speedup for the price of 1.0 up to 1.5 times more memory compared with the standard one-level preconditioner ILU(0) (i.e. $c_{\text{prec}} \in [1.0, 1.5]$) is typically obtained. For ILU(0) smoothing, we typically obtain $c_{\text{prec}} \in [1.5, 2.0]$.

4.6 Numerical Results for the Model Problems

In this section, we complete our discussion of the models with numerical results[31] of runs with different AMG approaches. Memory requirements have already been shown in Table 4.1. We concentrate on robustness now. The investigations made in Sections 3.3.3.1 and 3.4.1.2 and Examples 3.3, 3.4, 3.5 and 3.10 are confirmed by the results presented here.

The **AVLS models** are very simple to solve - in principle all PAMG variants and (if $c^2 < ab$) even VAMG and UAMG work here very efficiently stand-alone[32]. In contrast to this, the **AVLD models** need acceleration. As long as $c^2 < ab$, a variety of AMG approaches

[31] Notation: GS: VGS,UGS, or BGS. I: ILU. Bi: BiCGstab. V: VAMG, U: UAMG. Remainder: PAMG with a,n,d: $\boldsymbol{P} = A_{[1,1]}$, "ns", "dist". m,s,b: MU-,SU-,B-interpolation (with default type of weights). In all cases, standard coarsening and interpolation have been used. In all tables, the numbers of cycles to reach ϵ_{it}=1e-10 are shown. div = divergence, stag = stagnation, >h = more than 100 iterations.

[32] PAMG(dist,·,MU, ·) needs the least memory. However, VAMG, UAMG and norm-based PAMG are faster.

model	parms.	crs.	c_g	c_A	f_A	s_p	c_{prec}
AVLS	ϵ=1e-1	std.[a,b]	1.87, 1.67	3.49, 2.20	1.06, 1.41	1.94, 2.39	2.56, 1.69
AVLS	ϵ=1	std.	1.66	2.16	1.25	2.32	2.60
AVLD	ϵ=1e-3	std.	1.67	2.81	1.36	2.89	2.39
RD	all n_z, c	std.[c]	1.67, 1.67	2.20, 2.20	1.71, 1.71	2.39, 2.39	2.11, 2.11
DD	ϵ=1e-3	std.	2.00	2.88	1.27	1.52	2.31
DD	ϵ=1	std.	1.67	2.58	1.29	2.40	2.06
AVLS	ϵ=1e-1	agg.[a]	1.28, 1.13	1.51, 1.24	1.15, 1.43	1.33, 1.61	0.97, 1.03
AVLD	ϵ=1e-3	agg.	1.17	1.30	1.43	1.72	1.06
RD	all n_z, c	agg.[c]	1.17, 1.67	1.30, 2.20	1.74, 1.71	1.71, 2.39	1.31, 2.15
DD	ϵ=1e-3	agg.	1.48	1.96	1.29	1.35	1.51
DD	ϵ=1	agg.	1.21	1.74	1.28	1.76	1.32

[a] First value: (nearly) identical for VAMG(·), UAMG(·), PAMG(ns/nf/$A_{[1,1]}$,·,MU/B,A) and PAMG(ns/nf/$A_{[1,1]}$,·,SU,P); second value: PAMG(dist,·,SU,P).

[b] The *worst* values are a bit less than the ones presented in [58] for comparable tests.

[c] First value: PAMG(dist,·,MU,A) and PAMG(dist,·,SU,P); second value: PAMG(dist,·,B,A).

Table 4.1: Complexities for the model problems and the BGS-PAMG approaches marked bold-face in Tables 4.2-4.5 below (for AVLS, see footnote [a]; for RD, see footnote [c]). parms.=parameters, crs.=coarsening. $f_A := c_{\text{AMG}}/c_A$. Here always c_g=c_p and $c_{\boldsymbol{P}} \leq c_A$.

yields quite efficient preconditioners (see Table 4.2). With increasing $c^2/(ab)$, however, VGS and UGS are no appropriate smoothers and VAMG and UAMG no appropriate preconditioners any more. Only the PAMG(n,·,SU,P) preconditioners[33] show a stable convergence behavior. Due to the shape of algebraically smooth error, produced by BGS and also ILU (see Section 3.4.1.2), a coarsening in y-direction would be most suitable for u_1, and a coarsening in x-direction most suitable for u_2. PAMG(n,·,SU,P) performs a coarsening which yields a compromise of both. SU-interpolation fits best to the norm-based coarsening here so that, in total, this PAMG-approach is the most robust and efficient one for the AVLD models. ILU-PAMG(ns,·,SU,P)-BiCGstab turns out to be the best variant here.

As could be expected from the discussion in Section 3.3.3.1, VAMG and UAMG are not suitable for the **AVLX models** (see Table 4.2). Similarly to what we have observed for the AVLD models, PAMG(n,·,SU,P) as a preconditioner should also be suitable for AVLX models - in contrast to the AVLD models, however, only for "extreme" parameter settings such as $ab > c^2$ or $ab < c^2$ or $\epsilon = 1$ (in the last case, AVLX, AVLD and AVLS coincide, of course). This is indeed the case. To be more specific, for PAMG to be applicable, ab has to be much larger or at least moderately smaller than c^2.

Not unexpectedly, the picture for the **RD models** is more complicated (see Tables 4.3 and 4.4). GS-VAMG and GS-UAMG usually fail even with BiCGstab. Also with ILU smoothing, the situation remains unsatisfactory, and the results are quite unpredictable: BiCGstab can even make it worse! Using PAMG approaches with distance-based coarsening increases the robustness considerably. B-interpolation shows the most robust behavior here. In general, the performance of the variants with BGS smoothing increases with increasing c. Only the case of a large n_z together with $c \approx 1$ shows very slow convergence. For PAMG, the performance of

[33] Both variants for a norm-based $\boldsymbol{P}$, "ns" and "nf", give nearly the same results here.

a	c	GS-V	I-V	GS-U	I-U	GS-nm	I-nm	GS-ns	I-ns	GS-nb	I-nb
10	1	9	4	4	2	5	2	**84**	7	4	2
2	1	43	div	6	div	8	div	**82**	8	9	div
1	2	div	div	div	div	>h	div	**82**	8	div	div
1	10	div	div	div	div	18	div	**87**	8	div	stag
10	1	>h	stag	>h	stag	>h	stag	>h	div	div	stag
2	1	div	div	div	div	div	div	div	div	div	div
1	2	div	div	div	div	div	div	5	3	div	div
1	10	div	div	div	div	7	stag	4	3	div	div

Table 4.2: AMG-BiCGstab applied to AVLD (first) and AVLX models (second block of results). a=b, ϵ=1e-3, h=$1/512$, n_v=522 242, n_A=5 214 244. Error reduction approx. 1e-8 or better in all cases where the method converges.

the runs with BGS smoothing qualitatively "corresponds" to BGS' stand-alone performance (see Table 3.2). In contrast to this and to Table 3.2, ILU smoothing suffers from increasing n_z and c which might be due to the fact that a variable-based ILU is used. A corresponding block-variant can be expected to perform comparably to BGS (cf. also Remarks 5.15 and 5.16).

For the **DD models**, VAMG and UAMG diverge in most cases. Only ILU-UAMG-BiCGstab for $\lambda = c = 1$ works quite efficiently. As discussed in Example 3.5 and as can be seen from Table 4.5, PAMG with a norm-based coarsening yields a robust and efficient approach here - however, in case of $\epsilon \ll 1$, only with SU-interpolation (see also discussions in Example 3.12). Note that using B-interpolation leads to divergence here. ILU-smoothing does not increase PAMG's performance for the DD models. However, it can again be expected that block-variants of ILU-type smoothers do lead to better convergence rates (cf. also Section 5.3.2.2).

The results presented here together with the investigations made in Section 3.3.3.1 and 3.4.1.2 show that UAMG can handle strong anisotropies which are different from unknown to unknown but is likely to fail if the cross-unknown couplings are strong. In contrast to this, PAMG can handle strong cross-unknown couplings as long as the smoother employed (BGS or ILU, for instance) produces an algebraically smooth error which allows for a point-coarsening strategy.
The AVLX models show a limit of our AMG methodology: PDE systems with unknown cross-couplings which are too strong for UAMG and for which a point-coarsening and thus PAMG is not suitable as well cannot be handled by our AMG methodology.
The other model examples as well as real-life applications (see next chapter) show that, in particular, PAMG can handle very strongly coupled, practically very relevant PDE systems which exhibit very different numerical properties.

n_z	c	GS-V	I-V	GS-U	I-U	GS-dm	I-dm	GS-ds	I-ds	GS-db	I-db
1	1e0	10	7	10	7	10	7	26	13	10	7
	1e3	div	11	div	11	10	11	26	13	10	11
	1e9	div	18	div	18	4	18	7	16	4	18
100	1e0	>h	65	>h	65	98	64	26	13	10	7
	1e3	div	19	div	19	31	>h	32	71	21	17
	1e9	div	28	div	28	3	div	4	div	2	div
1000	1e0	div	div	div	div	div	div	div	div	div	div
	1e3	div	div	33	div	>h	div	75	div	32	28
	1e9	div	div	div	div	2^a	div	2^a	div	1^a	div

[a] Too less cycles: error reduction only one order of magnitude.
However, after 11 (13 for GS-db) iterations, the error is reduced by 1e-14.

Table 4.3: AMG applied to RD models. h=$1/512$, n_v=522 242, n_A=3 129 364. Error reduction approx. 1e-8 or better if the method converges, with the exception of case "[a]".

n_z	c	GS-V	I-V	GS-U	I-U	GS-dm	I-dm	GS-ds	I-ds	GS-db	I-db
1	1e0	div	3	div	3	**4**	3	**6**	4	**4**	3
	1e3	>h	4	>h	4	**4**	4	**6**	5	**4**	4
	1e9	>h	8	>h	8	**2**	9	**3**	10	**2**	8
100	1e0	14	10	11	8	**11**	8	**6**	4	**4**	3
	1e3	div	9	div	>h	**7**	>h	**7**	15	**5**	7
	1e9	>h	>h	div	div	$\mathbf{2}^a$	div	$\mathbf{2}^a$	>h	$\mathbf{1}^a$	>h
1000	1e0	>h	>h	>h	div	**>h**	div	**>h**	div	**div**	div
	1e3	div	13	div	div	**14**	>h	**11**	>h	**7**	11
	1e9	>h	>h	div	div	$\mathbf{2}^a$	div	$\mathbf{2}^a$	div	$\mathbf{1}^a$	>h

Table 4.4: AMG-BiCGstab applied to RD models. Parameters and "[a]" as in Table 4.3.

λ	c	ϵ	I-U-Bi	GS-am		GS-as		GS-nm		GS-ns	
1e+0	1e+0	1e+0	32	10	4	10	4	10	4	31	**7**
1e-3	1e+3	1e+0	div	10	4	10	4	10	4	10	**4**
1e-9	1e+9	1e+0	div	10	4	10	4	10	4	10	**4**
1e+0	1e+0	1e-3	stag	>h	53	>h	51	>h	56	>h	**25**
1e-3	1e+3	1e-3	div	>h	62	>h	66	>h	62	>h	**13**
1e-9	1e+9	1e-3	div	>h	68	>h	69	>h	67	>h	**10**

Table 4.5: AMG applied to DD models. h=$1/512$, n_v=783 363, n_A=7 562 289. For PAMG variants, left column: AMG stand-alone, right column: with BiCGstab. Again, error reduction approx. 1e-8 or better.

Chapter 5

Industrial Applications

The general AMG methodology described in Chapters 3 and 4 formally allows the definition of various concrete algorithms. It seems clear that there exists no unique AMG approach which will work satisfactorily for all systems of PDEs. Instead, major work still needs to be invested to compose and optimize concrete algorithms for certain classes of industrial applications. In this chapter, we consider the application of AMG to semiconductor simulation.

Semiconductor circuits play a central role in nearly all areas of our life. Popular examples of semiconductor circuits are microprocessors and memory chips. They can contain up to millions of single semiconductor devices, such as transistors, fabricated on the same die, a small piece of a semiconductor substrate (a wafer). The design and test of both single semiconductor devices and whole circuits is very expensive and due to the ever decreasing size very difficult. Therefore, great efforts are spent in replacing the iterative experimental process of constructing, testing and optimizing of hardware prototypes by computer simulations as far as possible. By now, semiconductor simulation can assist this experimental process and is able to reduce the number of expensive prototypes to an increasingly large extent.

Section 5.1 gives a short survey on semiconductor simulation. As discussed there and in more detail in the remaining sections of this chapter, two important and computationally expensive parts are process and device simulation. They mainly aim at the approximate computation of the final shape and doping profile of a single semiconductor device and its resulting electrodynamic behavior, respectively. Due to the complexity of the models and grids used, industrial process and device simulation are increasingly recognized as important and challenging areas for numerical simulation. Occurring PDE systems include *stress governing equations*, *reaction-diffusion equations* and *drift-diffusion equations*. Brief explanations of the terms diffusion, drift and reaction can be found in Section 2.2.2. All these systems exhibit different numerical properties and difficulties. As one consequence, different strategies are necessary to solve the arising linear systems efficiently by means of AMG.

That a simple unknown-based AMG approach as a preconditioner is suitable to speed up stress simulations will be shown in Section 5.2.1. For reaction-diffusion and drift-diffusion equations, the situation is more complicated. Where classical iterative solvers often converge only slowly (or even break down) and variable-based or straightforward unknown-based AMG are no sufficient preconditioners any more, suitable point-based AMG approaches, accelerated by BiCGstab or GMRes, can cause remarkable speedups. In Section 5.2.2, we will demonstrate that some typical reaction-diffusion problems can efficiently be solved by using a primary matrix based on geometric distances. As can be seen in Section 5.3, typical drift-diffusion problems, on the other hand, are efficiently solved by selecting a primary matrix based on norms.

We have performed concrete tests with the following simulation codes:

- FLOOPS [50] for stress analysis,
- DIOS [39] for reaction-diffusion simulation,
- TAURUS [92] for drift-diffusion simulation.

One should point out that the concrete industrial test cases, we had at our proposal, are rather small or only of medium size w.r.t. number of variables, compared to what industry can run today. They are too small to demonstrate "real" advantages of SAMG over one-level preconditioners in terms of computational speed. Nevertheless, compared with typical one-level solvers, a speedup of 2 is achieved for the largest stress analysis matrix. Compared with the iterative one-level solver used as a default in DIOS, a speedup of 2 is also achieved for the complete reaction-diffusion simulation run. For the largest drift-diffusion problem, SAMG is a bit faster than the iterative one-level solver used as a default in TAURUS. However, since SAMG clearly shows a robust behavior for all three applications and a considerably faster performance with increasing problem size, it can be expected that for larger problems than the ones presented here the observed trends will continue. Hence, we can expect that SAMG clearly outperforms the one-level solvers which are commonly used in commercial simulators in case of large(r) problem sizes.

Remark 5.1 We use the definitions and descriptions of our AMG approaches made in the last chapter. Descriptions of the standard one-level iterative approaches, our AMG approaches are compared with in this chapter, can be found in [74]. In Section 2.4.6, the definitions of ARFs, stopping criteria etc. can be found. Remember, in particular, that n_u denotes the number of physical unknowns, n_p the number of points, n_v the number of variables, and n_A the number of matrix elements stored ($n_A \geq$ number of nonzeros). ▲

5.1 Semiconductor Simulation

The main purposes of semiconductor simulation are twofold:

1. **circuit design:** the design of a suitable "logic" and layout of a circuit serving a specific application,
2. **circuit simulation:** the validation of the design and the computation of physical properties of a concrete circuit.

The knowledge about the properties of single semiconductor devices, such as resistors, capacitors, diodes, transistors, et cetera, which are used to construct whole circuits, forms the basis for both parts. Therefore, a simulation of the manufacturing process, the so-called **process simulation**, and afterwards a simulation of electrodynamic properties of the device, the so-called **device simulation**, have to be carried out for each type of device used in modern circuits before the process of circuit design and circuit simulation can be performed. The steps (re-)design and simulation have typically to be performed in an iterative fashion in order to optimize the circuit layout for a specific application.

An overview of semiconductor simulation is depicted in Figure 5.1. It shows the four main steps involved, namely *process*, *device* and *circuit simulation*, and *circuit design*, indicating their dependencies. Short characterizations of the four steps are given in the following.

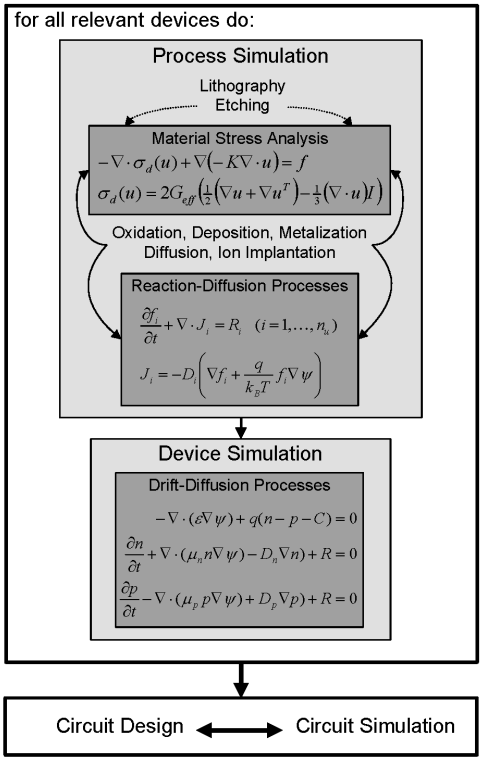
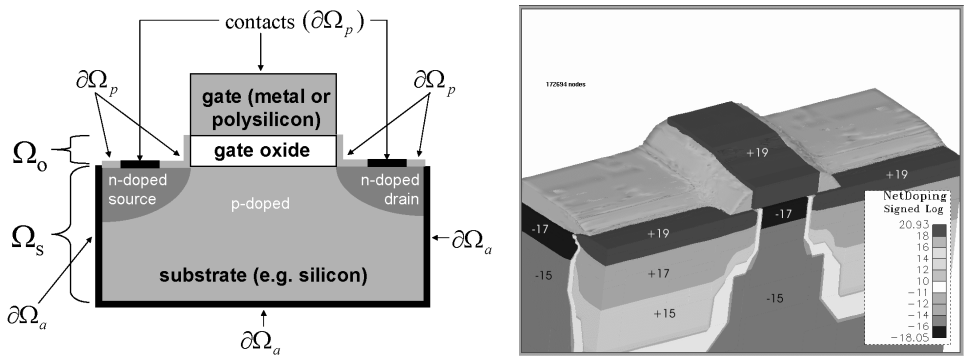


Figure 5.1: Overview of semiconductor simulation. The parts in dark grey indicate the problem classes discussed in this chapter.

Process Simulation Figure 5.2(a) shows a schematic view of an (n-)MOSFET[1]. In order to fabricate the different material layers, in the example the gate oxide and the contacts, on the semiconductor wafer, several steps of pattern definition (lithography), pattern transfer (etching), layer formation (oxidation, deposition, metalization), and layer modification (diffusion, ion implantation) have to be performed. Hence, during the manufacturing process of a device, several different configurations (i.e. the composition and geometry) of material layers develop until the final configuration is reached.

[1]MOSFET is an abbreviation of “metal oxide field effect transistor”. “n” stands for “with an n-channel”. A more detailed description of the different MOSFET regions can be found in Section 5.3.1.3.



(a) (b)

Figure 5.2: (a) Sketch of an n-MOSFET. For explanations, see Section 5.3.1.3. (b) Layout and doping profile of a p-MOSFET. Courtesy of Synopsys Inc.

The main tasks of a process simulation are the computation of these configurations, in particular the final configuration of the device, the stress and strain distributions inside the layers and across their interfaces, and the doping profile of the device (i.e. the distribution of electrons and holes within the device). The final shape and doping profile for an exemplary MOSFET are depicted in Figure 5.2(b). Such results serve as input for a device simulation.

As indicated in Figure 5.1, two problem classes play an important role in nearly all steps of a process simulation, namely the stress analysis and the simulation of reaction-diffusion processes. The corresponding models and their efficient numerical treatment are topics of Section 5.2.

Device Simulation A device simulation takes the above-mentioned results of a process simulation for a particular device as input and computes its electrodynamic properties. That means, based on the geometry and the doping profile, local functions including the electrodynamic potential and electron and hole concentrations are computed for each relevant bias applied to the contacts of the device. For this purpose, a series of drift-diffusion systems is solved. A detailed description of these systems can be found in Section 5.3. In particular, the difficulties in their numerical treatment will be discussed. The systems are strongly coupled, highly nonlinear and usually very ill-conditioned, and the efficient solution of the arising linear systems by means of iterative solvers is a great challenge.

From the local functions computed, the net currents of the device are determined for each bias applied, resulting in so-called IV-characteristics[2]. These are of particular importance for industrial purposes since they determine the behavior of a device within a circuit to a large extent. For instance in case of a transistor, the IV-characteristics yield information on its switching behavior. Hence, device simulations have to be performed for each relevant semiconductor device before the design and simulation of whole circuits can take place.

[2] "IV" stands for current-voltage (or current-bias) characteristics!

Circuit Design Having a specific application in mind, engineers decide on the components of a suitable circuit and its principal layout. The computation of an optimal placement of all devices and their interconnects within a three-dimensional structure which can be fabricated on a wafer is the objective of CAD[3] tools for (up to) ULSI[4] layout design.

Remark 5.2 In [71, 65], the application of AMG to certain two-dimensional placement problems in VLSI layout design is considered. The arising matrices are similar to a system consisting of two discrete Laplacians (with some additional algebraic constraints). In [65], the two parts are decoupled, in [71] they may be weakly coupled. In the first case, VAMG (with a suitable treatment of the constraints) can be applied, in the second case a straightforward UAMG approach yields an efficient solver.

Circuit Simulation Assuming a layout of a particular circuit to be given and its electrodynamic properties (and maybe other properties such as heat distributions) for each device involved to be computed, a simulation of the whole circuit can be performed. Circuit simulation aims at the validation of the layout design, i.e. answering the question “Does the manufactured circuit work as it has been designed to?”. Time-dependent, nonlinear systems of differential-algebraic equations have to be solved.

Remark 5.3 AMG is able to efficiently solve certain subclasses of matrices arising in circuit simulation. However, these problems are not in the scope of this thesis.

5.2 Process Simulation

Two principal classes of PDE systems are involved in process simulation[5]. The first class describes the mechanical deformation of fabricated multi-layer material structures. The corresponding **stress governing equations** account for the distribution of the stresses and strains for the constellations of material layers arising during the manufacturing process of a device. Systems of stress governing equations are essential for the analysis of native film growing processes (such as thermal oxidation and nitridation) as well as the analysis of mechanical properties of the wafer after deposition and etching processes.

The second class of PDE systems models the redistribution of dopants and point defects in thermal processes (for example, an annealing step after an implantation) and requires the solution of multi-species **reaction-diffusion systems**. Simulations of reaction-diffusion processes are of major importance for determining the final doping profile of the device, and their results are therefore an important input for a subsequent device simulation, as already pointed out.

We will see in the following chapters that both stress systems and reaction-diffusion systems - or at least important parts of them - exhibit an “elliptic nature” and are ill-conditioned.

[3] CAD = computer aided design.

[4] LSI = large scale integration. An integrated circuit that uses very-large-scale integration (VLSI) contains 100,000 up to 1,000,000 transistors, an integrated circuit that uses ultra-large-scale integration (ULSI) more than one million transistors.

[5] A general survey on semiconductor process modeling can be found in [42], for example.

Whereas this makes them critical for the application of classical, one-level iterative solvers, hierarchical approaches provide a possibility to speed up the computations. We will discuss suitable AMG approaches for these problem classes in Sections 5.2.1 and 5.2.2, respectively. In particular, we will demonstrate that efficient methods can be obtained by choosing an unknown-based strategy for stress governing equations and a point-based strategy with a distance-based primary matrix for reaction-diffusion equations.

For stress simulation, we concentrate on an analysis of ill-conditioning and convergence based on eigenvalue calculations. This is possible since the matrices investigated here are rather small and, hence, allow for accurate eigenvalue computations. For reaction-diffusion systems, the focus lies on the performance of SAMG compared with iterative one-level solvers used in a commercial process simulator. For this purpose, a typical simulation run with about 200 matrix solves for a 3D test layout is considered.

5.2.1 Stress Simulation

After a description of the system of stress governing equations in the following Section 5.2.1.1 and our test cases (Section 5.2.1.2), we will characterize the matrices stemming from such systems (Section 5.2.1.3) and discuss their efficient solution by means of a suitable AMG approach (Section 5.2.1.4). Numerical results for two selected problem classes are presented and discussed in Section 5.2.1.5. In particular, results on the efficiency and robustness of AMG in comparison with classical one-level linear solvers are presented.

5.2.1.1 Governing Equations

The stress analysis in process simulation is essentially based on the momentum equation

$$-\nabla \cdot \boldsymbol{\sigma}_{\mathbf{d}} + \nabla p = \mathbf{f} \quad \text{in} \quad \Omega \tag{5.1}$$

where Ω is a bounded domain with boundary Γ, $\boldsymbol{\sigma}_{\mathbf{d}}$ is the symmetric deviatoric stress tensor, p is the mean pressure and $\mathbf{f}$ is the body force. The boundary conditions are given by

$$(-p\mathbf{I} + \boldsymbol{\sigma}_{\mathbf{d}}) \cdot \mathbf{n} = \mathbf{g} \quad \text{on} \quad \Gamma_{\mathrm{g}} \tag{5.2}$$

$$\mathbf{u} = \mathbf{h} \quad \text{on} \quad \Gamma_{\mathrm{u}} \ , \tag{5.3}$$

where $\mathbf{g}$ is the surface traction of the boundary segment $\Gamma_{\mathrm{g}} \subset \Gamma$, $\mathbf{h}$ is the displacement of the boundary segment $\Gamma_{\mathrm{u}} \subset \Gamma$ ($\Gamma_{\mathrm{u}} \cap \Gamma_{\mathrm{g}} = \emptyset$), $\mathbf{n}$ is the outward unit normal vector on the boundary and $\mathbf{I}$ is the identity tensor.

Mechanical properties of the materials involved in the semiconductor fabrication vary from purely elastic solids to viscous fluids. Silicon and poly-silicon are assumed to be elastic materials whereas nitride Si_3N_4 and oxide SiO_2 are assumed to be viscoelastic compressible fluids. The mechanical properties are quite accurately modeled with the constitutive relationship of the Maxwell viscoelasticity (cf. [83]). The Maxwell viscoelasticity is commonly implemented in process simulation in its incremental form based on the constitutive relationships of linear elasticity

$$\boldsymbol{\sigma}_{\mathbf{d}} = 2G_{\mathrm{eff}} \left[\frac{1}{2} \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^{\mathrm{T}} \right) - \frac{1}{3} (\nabla \cdot \mathbf{u}) \mathbf{I} \right] \quad \text{in} \quad \bar{\Omega} \tag{5.4}$$

$$p = -K \nabla \cdot \mathbf{u} \quad \text{in} \quad \bar{\Omega} \tag{5.5}$$

where the viscoelastic material properties are introduced by an effective shear modulus (effective rigidity) G_{eff} given by

$$G_{\text{eff}} = G \frac{\tau}{\Delta t} \left(1 - \exp\left(-\frac{\Delta t}{\tau} \right) \right) . \tag{5.6}$$

Here, $\mathbf{u}$ is the incremental displacement vector, $G > 0$ and $K > 0$ are the shear and bulk moduli (rigidity and compressibility, respectively), Δt is the time step size, and τ is the Maxwellian relaxation time defined as $\tau = V/G$, where V is the material viscosity. G and K are assumed to be known constants.

G_{eff} provides a continuous modeling of the mechanical behavior of a material from purely elastic deformation to viscous flow. Namely, for $G \ll V$ it reduces to Hooke's law for elasticity ($G_{\text{eff}} = G$) while for $V \ll G$, we obtain Newton's law for viscous fluids. However, note that only the incremental deviatoric stress component exhibits stress relaxation, accounted for by G_{eff}, while the incremental dilatational stress component is assumed to be purely elastic in this model.

In general, $\mathbf{u}$ is a vector of three scalar-valued functions, namely the displacements in x-, y- and z-direction, in the following denoted by u_1, u_2 and u_3, respectively. By inserting equations (5.4) and (5.5) into (5.1), and by assuming the full elastic case $G_{\text{eff}} = G$, we arrive at the classical Lamé equations (3.66). The parameters E, ν, λ, μ from (3.66) and (3.68) are related to G and K as follows:

$$G = \mu = \frac{E}{2(1+\nu)} \ , \quad \frac{4G}{3} + K = 2\mu + \lambda \ , \quad \frac{G}{3} + K = \mu + \lambda \ . \tag{5.7}$$

For all test examples investigated here, the stress analysis is based on a plane-strain formulation (3.67). Hence, the following PDE system has to be solved for the unknowns u_1 and u_2:

$$\begin{bmatrix} -\left(\frac{4G}{3} + K\right) u_{1,xx} - G u_{1,yy} - \left(\frac{G}{3} + K\right) u_{2,xy} \\ -G u_{2,xx} - \left(\frac{4G}{3} + K\right) u_{2,yy} - \left(\frac{G}{3} + K\right) u_{1,xy} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \ . \tag{5.8}$$

Also the boundary conditions can be formulated in terms of the u_1, u_2 (and u_3 in 3D).

Table 5.1 gives values[6] for E and ν, reported in [83], and resulting λ and μ for the layer materials discussed here. In the viscoelastic case (oxide and nitride), one has to be aware of the fact that the parameter G has to be replaced by the function $G_{\text{eff}} = G_{\text{eff}}(G, V, \Delta t)$ defined in (5.6). V is temperature- and material-dependent[7] and decreases, for instance, for the oxide[8] from 9e5 GPa s at 800° C to 4e3 GPa s at 1100° C. Values for nitride are (a bit) higher. Since $G = \mu$ is (considerably) smaller than V here, we arrive at a $G_{\text{eff}} \approx G$ so that, in principle, the computed values for λ, μ can be used for the following discussions of numerical properties of the arising matrices.

[6] Throughout this chapter, values for physical constants are given in basic SI units (used are the following five out of seven: A,K,kg,m,s) or typical derived SI units (for example: 1 C = 1 A s, x°C = $(x + 273.15)$ K, 1 Pa = 1 kg m^{-1} s^{-2}, 1 V = 1 kg m^2 s^{-3} A).

[7] for concrete values, see [83].

[8] in the "wet processing" case. Values for the "dry processing" case are approximately 10 times higher.

material	E [GPa]	ν	λ [GPa]	μ [GPa]	$\mu+\lambda$ [GPa]	$2\mu+\lambda$ [GPa]	$\frac{\mu}{2\mu+\lambda}$
SiO_2	660	0.17	145.3	282.1	427.4	709.4	0.40
Si_3N_4	3890	0.30	2244.2	1496.2	3740.4	5236.5	0.29
Si and poly-Si	1870	0.28	929.7	730.5	1660.2	2390.6	0.31

Table 5.1: Material constants E, ν from [83] and resulting λ, μ etc.

5.2.1.2 Test Cases

The test problems for the numerical experiments have been generated by the process simulator FLOOPS [50]. For several material systems, stress simulations have been performed and typical global stiffness matrices been extracted. To be more specific, we have considered two general classes of problems very relevant for the manufacturing process:

- The first one corresponds to the "single full integration stress solving step" in the simulation of the "sealed interface local oxidation" (SILO) process. Originally, SILO is an evolutionary problem on a time-dependent domain $\Omega = \Omega(t)$. To be more specific, during this oxidation process, the SiO_2-layer is permanently growing at the expense of the silicon layer. This results in a permanent change of the layer topology and, in particular, moving layer interfaces. Since a SiO_2 molecule needs considerable more space than a Si atom, stresses and strains emerge in particular at the layer interfaces, resulting in material displacements. Due to the tight coupling of the two processes oxidation and displacement of material, they have to be solved in a coupled way, usually by an "incremental approach"[9]. We consider one particular stress analysis step here. The underlying grid structure of one of the two SILO examples chosen is shown in Figure 5.17(a).
- The second class of simulation examples, DEPO, is related to the stress distributions in multilayer material regions after thin film deposition processes. Origins of the stress are intrinsic stress distributions in the material films deposited. In order to test different problem scales, one particular DEPO problem, containing four different material layers, is formulated with four differently refined grids. Fig. 5.17(b) shows one of the corresponding grid structures. Inside and in the neighborhood of the very thin poly-Si layer contained in the concrete DEPO class chosen, the stresses are largest. This demands a very fine discretization grid in and around this layer.

As common for process simulators as, for instance, FLOOPS [50], the system of stress governing equations is discretized using standard piecewise linear finite elements on a triangulation of the domain Ω. Usually, an unstructured grid is employed, see Fig. 5.17. Table 5.2 compiles data on the magnitude of the six arising grids and matrices[10].

5.2.1.3 Characterization of the Arising Matrices

Important numerical properties of the matrices A arising for the plane-strain problem (5.8) to be solved have already been discussed in Section 3.3.3.2. From there we know that the

[9] For more details and an illustration of a SILO process, see [83], for instance.

[10] It was only possible to test rather small examples with less than 25.000 variables. This is because the version of FLOOPS used for the tests did not allow for a finer discretization of the test cases.

example	layers	n_p	n_v	n_A
SILO1	5	716	1 432	15 912
SILO2	5	905	1 810	23 079
DEPO1	4	528	1 056	13 577
DEPO2	4	4 477	8 954	121 868
DEPO3	4	8 418	16 836	230 896
DEPO4	4	10 897	21 794	299 593

Table 5.2: Description of the test matrices.

matrices A are ill-conditioned and, hence, classical one-level solvers face problems in solving the matrices efficiently. For instance, the larger[11] λ/μ or the smaller r_{Dir}, the less efficient they are. We recall that r_{Dir} is defined as the number of Dirichlet variables divided by the total number of boundary variables.

UAMG is a good preconditioner for linear elasticity matrices on appropriate FE grids if the rate r_{Dir} is large enough. Regarding the two problems classes SILO and DEPO, quite a large part of the boundary is fixed by Dirichlet conditions. To be more specific, for all test cases considered here, r_{Dir} is between 0.5 and 0.6 so that we expect the convergence of UAMG to be only a bit worse than for the full Dirichlet case, $r_{\mathrm{Dir}} = 1$.

In both problem classes, SILO and DEPO, we have to deal with at least one nitride layer, at least one oxide layer and a silicon layer. The DEPO class contains an additional poly-Si layer. From Table 5.1 it can be seen that, for all four layers, ν is far from the critical value 0.5. Therefore, the material properties lead to only a slight anisotropy of the two PDEs.

However, the parameters of neighboring layers can be quite different. This is particularly true for the oxide compared to the nitride, since here the values of E, λ, μ of the nitride are approximately one order of magnitude larger than those of the oxide (see Table 5.1). Hence, especially the SILO2 example (see Fig. 5.17(a)) suffers from this parameter "jump" and, physically interpreted, large stresses around its two oxide-nitride interfaces can occur. Together with the fact that the mesh around these interfaces is not particularly refined (see Fig. 5.17(a)), this might explain to a large extent the fact that SILO2 has by far the worst condition number here [12] (see Table 5.3).

As indicated above, the very thin poly-Si layer in the DEPO structure causes large stresses and could numerically be an origin of ill-conditioning, although poly-Si and Si have the same E and ν values. However, the DEPO meshes are refined in and around this layer (see Figs. 5.3 and 5.17(b)) so that the effect is lessened, and the condition number of the DEPO matrices is rather high but considerably lower than for SILO2 (see Table 5.3). The coarsening structure produced by UAMG(std) for DEPO2 is shown in Fig. 5.3. Also an enlargement of the area "between" the nitride and oxide and around the critical thin poly-Si layer is shown there. As can be seen from Table 5.4, the coarsening is equally fast in all four material layers so that, in particular, the critical thin poly-Si layer is not "neglected" but handled as the other layers.

We have computed the strength of unknown cross-couplings as described by $\rho(A_u A^{-1})$,

[11] This is equivalent to growing ν, approaching 0.5 from below.

[12] In contrast to SILO2, SILO1 is refined at the interfaces. The overall discretization grid of SILO1 is, however, rather course.

example	n_u	cond$_E$	$\rho(A_u A^{-1})$	$\rho(A^{-1} A_u)$	ρ_u	ρ_t	ρ_{11}	ρ_{22}
(3.67), ν=0.20	2046	1382	2.23	1.55	3.46	0.20	0.07	0.03
(3.67), ν=0.33	2046	1978	2.59	1.61	4.18	0.26	0.06	0.06
(3.67), ν=0.45	2046	5322	5.50	1.82	10.01	0.60	0.05	0.05
DEPO1	1056	4091	3.62	1.72	6.25	0.69	0.62	0.26
SILO1	1432	2137	2.63	1.62	4.27	0.25	0.15	0.04
SILO2	1810	49675	4.67	1.79	8.34	0.57	0.44	0.48

Table 5.3: ρ_u and asymptotic convergence rates (i.e. spectral radii of the corresponding iteration matrices) ρ_t = ρ[GS-UAMG(std)(A)], ρ_{11} = ρ[GS-VAMG(std)$(A_{[1,1]})$], ρ_{22} = ρ[GS-VAMG(std)$(A_{[2,2]})$] for (3.67) with three different ν and the three smallest stress matrices. Remark 4.25 explains how the iteration matrices M have been computed. All eigenvalue computations have been performed with LAPACK's [1] direct eigensolver.

layer	nodes	$c_{g,1}$	$c_{g,2}$
oxide	1013	1.55	1.49
nitride	799	1.54	1.49
poly-Si	587	1.56	1.47
silicon	2424	1.55	1.51
in total	4477	c_g=1.52	

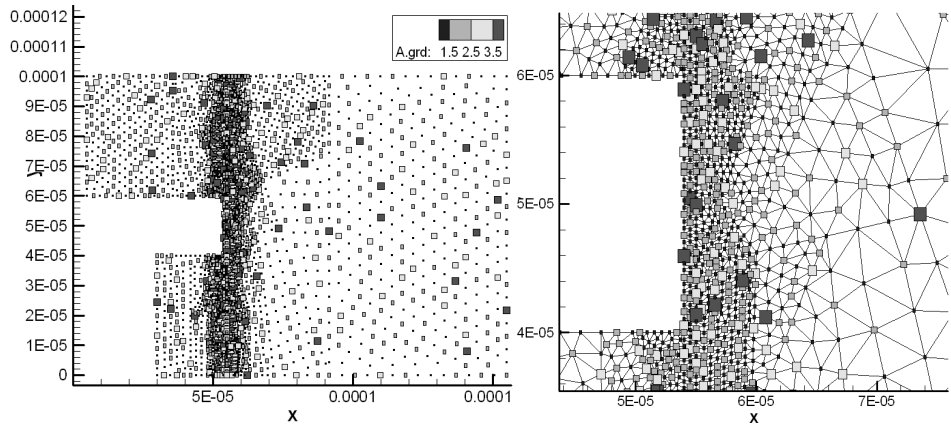
Table 5.4: Grid complexities for the DEPO2 example. For the current layer: nodes = number of nodes in layer including layer boundary nodes, $c_{g,n}$ = grid complexity w.r.t. the n-th unknown.

$\rho(A^{-1} A_u)$ and ρ_u (3.65), in order to show that the matrices are indeed not too strongly coupled for UAMG to work, and that they have properties similar to the plane-strain problem (3.67) (see Section 3.3.3.2) on standard grids with a standard FE discretization. Table 5.3 shows these values for the three smallest stress examples, and for model (3.67) on the unit square, discretized using bilinear finite elements, $h = 1/32$, with Dirichlet conditions on two sides of the unit square and with three different ν. The values for the stress matrices are indeed comparable with the ones for the linear elasticity models - this is also true for SILO2, by far the most ill-conditioned matrix here - so that UAMG should exhibit a similar convergence behavior. In particular, we expect UAMG to be a more efficient preconditioner than classical one-level solvers. However, since the SILO and DEPO problems exhibit only a slight anisotropy, we also expect PAMG to show similar convergence properties here. That these expectations hold is demonstrated below.

5.2.1.4 Efficient AMG Approaches

As mentioned in Section 3.3.3.2 and Remark 3.31, unknown-based as well as certain point-based AMG approaches with block-interpolation have been investigated in the literature for linear elasticity problems. We have thus tested both UAMG as well as PAMG variants with GS[13] as well as ILU(0) for smoothing. We have also compared standard coarsening with

[13] which means UGS for UAMG and BGS for PAMG.



(a) (b)

Figure 5.3: Coarsening for the x-displacement in case of the DEPO2 problem. (a) Full grid without FE edges, (b) enlargement of the critical area, with FE edges.

A1-coarsening. In addition and according to the following Remark 5.4, we have tested, for UAMG, both the block-Galerkin and the full-Galerkin variant.

Remark 5.4 In [56], AMG1R5 [71] as a block-diagonal preconditioner for BiCGstab or GMRes has been investigated for stress governing equations. This approach corresponds[14] to block-UAMG, that is an unknown-based AMG approach with block-Galerkin coarse level operators (3.57). The deterioration of convergence usually caused by using block-UAMG instead of full-UAMG is often compensated by the reduction of computational work and also memory requirements of the setup phase and each cycle. That this also holds for SAMG applied to our test cases will be demonstrated below. ▲

Table 5.3 also gives the spectral radii ρ of the iteration matrices, that is the asymptotic convergence rates, of GS-UAMG(std) applied to A as well as its two diagonal blocks $A_{[n,n]}$, $n = 1, 2$. These asymptotic convergence rates show that stand-alone UAMG converges with "acceptable"[15] but varying rates. The convergence of UAMG applied to A is always worse, sometimes considerably, than that of VAMG applied to the $A_{[n,n]}$. However, for the SILO and DEPO matrices, the performance of VAMG for the $A_{[n,n]}$ (the maximum of ρ(GS-VAMG(std))($A_{[n,n]}$)) is not too far from UAMG's behavior.

As a generally expected trend, for comparable problem sizes, ρ_t is increasing with ρ_u, though this does not hold strictly, most probably due to the different properties of the materials and grids. Nevertheless, two well-separated "levels" can be observed for the six matrices listed in Table 5.3: very roughly, $\rho_t \in [0.2, 0.3]$ corresponds to $\rho_u \in [3, 5]$, and $\rho_t \in [0.5, 0.7]$

[14] besides the fact, that in [56] also (Gauss-Seidel) smoothing is performed separately for each unknown, i.e. GS is applied to the $A_{[n,n]}$, whereas we use UGS smoothing.

[15] for a stand-alone AMG variant. Acceleration is discussed below.

to $\rho_u \in [6, 10]$ here. It is subject to further investigations whether such "level formation" can also be found in more general situations.

preconditioner	bd	it.	ARF	c_A	c_P	time	m_{tot}	c_{prec}	sp.
ILU(0)-UAMG(std)		17	0.25	2.3	0.0	1.53	19.44	2.38	1.8
ILU(0)-UAMG(std)	x	23	0.35	1.6	0.0	1.53	14.82	1.79	1.8
ILU(0)-PAMG(ns,std,t,P)		20	0.28	1.8	2.1	1.50	19.95	2.45	1.8
ILU(0)-UAMG(agg)		32	0.48	1.2	0.0	1.59	11.33	1.34	1.7
ILU(0)-UAMG(agg)	x	32	0.48	1.1	0.0	1.41	10.49	1.23	1.9
ILU(0)-PAMG(ns,agg,t,P)		33	0.48	1.2	1.3	1.63	14.95	1.81	1.7
GS-UAMG(std)		21	0.32	2.3	0.0	2.25	11.86	1.50	1.2
GS-UAMG(std)	x	27	0.39	1.6	0.0	2.25	8.86	1.11	1.2
GS-PAMG(ns,std,t,P)		24	0.34	1.8	2.1	1.94	13.55	1.54	1.4
GS-UAMG(agg)		42	0.58	1.2	0.0	2.28	6.73	0.77	1.2
GS-UAMG(agg)	x	46	0.58	1.1	0.0	2.34	6.30	0.68	1.2
GS-PAMG(ns,agg,t,P)		45	0.59	1.2	1.3	2.31	10.39	1.18	1.2

Table 5.5: Results for the DEPO4 example. Accelerator always BiCGstab. bd: block-diagonal Galerkin, it.: number of BiCGstab-cycles, time: wall-clock time [sec], m_{tot} [MBytes]: total memory needed, sp.: speed-up compared with fastest one-level solver.

For the larger matrices, namely DEPO2-4, stand-alone AMG variants stall or diverge after a few iterations. Eigenvalue distributions for the iteration matrices of GS-UAMG(std) and ILU(0)-UAMG(std) applied to the SILO2 matrix (see Figs. 4.7 (a) and (b), respectively), show that only a few eigenvalues are between 0.25 and 0.57 (UGS smoothing) or are even larger than 1 (ILU smoothing), preventing AMG stand-alone from being more efficient. We can assume a "similar picture" also for the other matrices. Hence, acceleration[16] by BiCGstab or GMRes should considerably enhance the convergence. Indeed, this is always the case as can exemplarily be seen for DEPO4 in Table 5.5 where results for the different AMG variants mentioned above, accelerated by BiCGstab[17], are collected: each of the accelerated approaches shows a reasonable ARF whereas stand-alone GS-UAMG(std) stalls after a few iterations.

As expected, UAMG- and PAMG-preconditioned BiCGstab yield similar convergence rates here, but due to the facts that PAMG's setup phase is more expensive and PAMG needs more memory, UAMG is the more efficient preconditioner. In all cases, approaches with standard coarsening yield a smaller, better ARF but are not faster than the corresponding variants with aggressive coarsening which are clearly preferable overall. Whereas ILU(0)-UAMG diverges (see Fig. 4.7(b)), ILU(0) smoothing helps improving the *preconditioning* properties considerably so that its use also pays in total computational time here. However, the memory requirements are of course higher as for the GS variants. The qualitative results for the other five examples are basically the same. An interesting result is that the block-AMG variants are more efficient than the corresponding full-AMG variants, confirming Remark 5.4.

[16] Note that the matrices are asymmetric due to the boundary conditions.

[17] For the matrices considered, BiCGstab has turned out to be a more efficient accelerator for AMG than GMRes(k).

Remark 5.5 For all six test cases, it has turned out that PAMG variants with B-, MU- or SU-interpolation, a norm-based or distance-based $\boldsymbol{P}$ converge with very similar ARFs here. This indicates that, for "real-life" grids, PAMG approaches even with block-interpolation do not tackle the rotational rigid body modes better than UAMG (as sometimes hoped in the literature). ▲

one-level solver	it.	ARF	time	m_{mat}	m_{tot}
ILUT(3;0.005)-BiCGstab	> 500	(stag)			
ILUT(9;0.005)-BiCGstab	135	0.84	2.7	3.9	10.1
ILUT(9;0.005)-GMRes(4)	> 500	0.98	> 5.3	3.9	10.7
ILUT(9;0.005)-GMRes(20)	> 500	0.96	> 5.9	3.9	16.1

Table 5.6: Results for the DEPO4 example. Fastest one-level solver (ILUT(9;0.005)-BiCGstab) and best GMRes-solvers. it.: number of cycles, time: wall-clock time [sec], m_{mat}: memory [MBytes] needed for original matrix A, m_{tot}: total memory needed. "stag" means stagnation of the approach. Note that the overall speed depends only very slightly on $\tau_{droptol}$ here.

5.2.1.5 Comparison with One-Level Solvers

Among the iterative one-level methods implemented in SAMG, ILUT(9;5e-3)-preconditioned BiCGstab has turned out to be the fastest one-level solver. Corresponding results for the DEPO4 example are shown in Table 5.6. Hence, our yardstick for comparisons with UAMG-BiCGstab approaches is ILUT(9;5e-3)-BiCGstab.

Figs. 5.4 and 5.5 show comparisons of the fastest AMG approach, ILU(0)-UAMG(std), with the best one-level solver, ILUT(9;5e-3)-BiCGstab. In addition, the performance of GS-UAMG(std) is shown. The results clearly demonstrate that UAMG-BiCGstab reduces both residuals and errors quickly and much more stable than ILUT(9;5e-3)-BiCGstab. Whereas the classical iterative solvers become inefficient or even stagnating with increasing problem size, in case of AMG-preconditioned BiCGstab and also GMRes(20) the ARFs are bounded from above by a constant substantially smaller than 1, and the computing times are (nearly) proportional to the number of variables. Therefore, they are efficient preconditioners and exhibit a nearly optimal behavior here. Even for DEPO4, the largest matrix in the set, but in absolute terms rather small, already a speedup of nearly 2 has been achieved in comparison with classical one-level solvers. It can be expected that the speedup considerably grows with increasing problem size.

5.2.2 Reaction-Diffusion Processes

Reaction-diffusion processes[18] play a key role in layer modification steps such as oxidation, nitridation, silicidation, ion implantation and the like (cf. Fig. 5.1), and in thermal annealing

[18] more precisely, drift-diffusion-reaction processes.

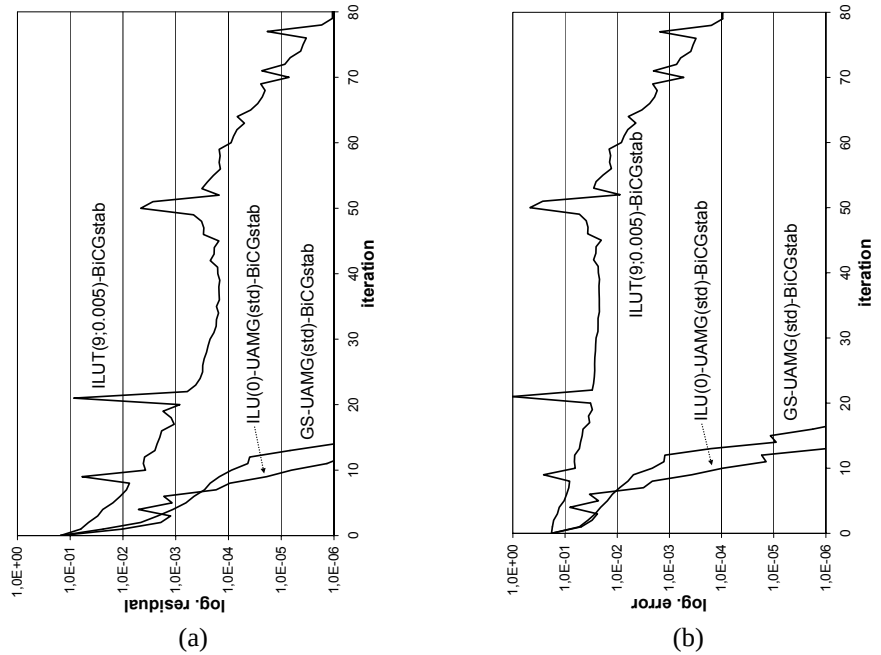


Figure 5.4: Reduction of (a) residuals and (b) errors for DEPO4.

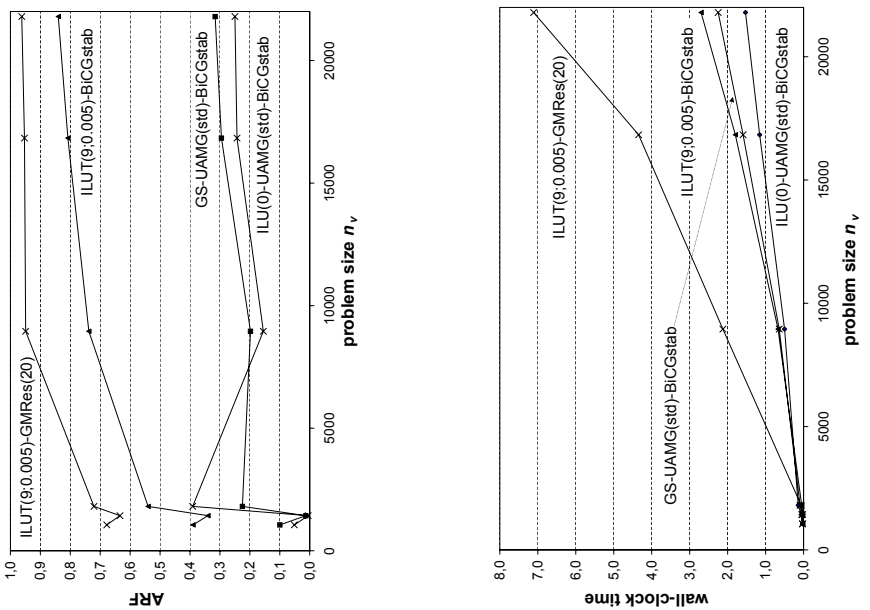
steps following the aforementioned steps. We restrict the subsequent discussions to thermal annealing steps which do alter the constitution but do not alter the shape of the layers.

According to [37, 36], the present understanding of reaction-diffusion processes of dopants and point defects in silicon is as follows. The interaction of substitutional dopants[19] and silicon point defects[20] is believed to be the elementary physical process that causes dopant redistribution. Dopants on lattice sites and silicon interstitials or vacancies react with each other. The created dopant-point defect pairs are assumed to diffuse, resulting in dopant redistribution. A large number of ionization and other chemical reactions is assumed to occur at the same time, involving both dopants and point defects. Usually, the electrical processes (charge transport and generation and ionization reactions) are assumed to be very fast in comparison with the chemical reactions and dopant transport phenomena so that equilibrium for the electronic reactions is assumed for process simulation runs.

The task of reaction-diffusion simulations is then to determine the concentration of each relevant species as a function of space, time and outer process conditions. The most important species which have to be treated are substitutional and interstitial impurities, intrinsic silicon point defects (interstitials and vacancies) in various charge states, impurity-point defect pairs in various charge states and immobile configurations of dopants and point defects.

[19] **Dopants** are species (atoms, ions, clusters) different from the basic wafer material (e.g. silicon). Examples are atoms of the third or fifth group of the periodic table as boron or phosphorus, respectively.

[20] Silicon **point defects** are interstitial silicon atoms or vacancies in the silicon crystal lattice.



(a) (b)

Figure 5.5: (a) Average residual reduction rates and (b) computing times for the 6 examples and 4 different solvers.

5.2.2.1 System of Reaction-Diffusion Equations

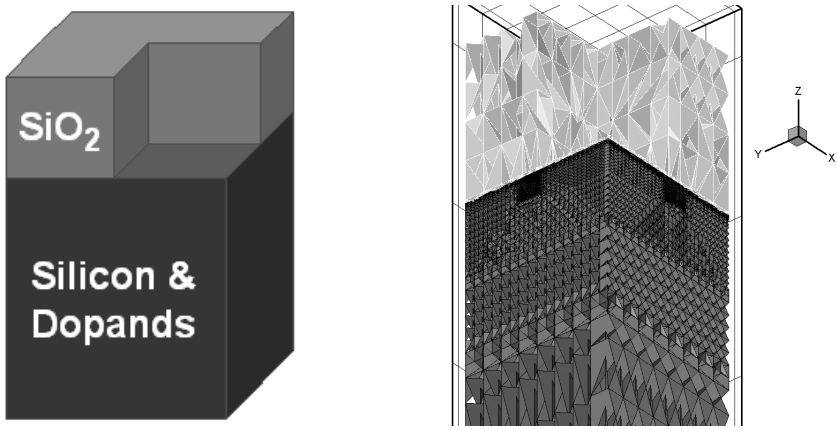
A system of reaction-diffusion equations consists of a sequence of balance equations of the form (see [37, 36])

$$\frac{\partial C_i}{\partial t} + \nabla \cdot J_i = R_i \quad (i = 1, \ldots, N) \tag{5.9}$$

where the J_i denote (diffusion and field driven) fluxes given by

$$J_i = -D_i \left(\nabla C_i + \frac{q}{k_B T} C_i \nabla \psi \right). \tag{5.10}$$

q denotes the elementary charge, k_B the Boltzmann constant, T the absolute temperature, ψ the electrostatic potential, and N the number of species with a magnitude of about 30 to 40 typically. For the i-th species, C_i denotes the concentration, $R_i = R_i(C_1, \ldots, C_N)$ the reaction term, and D_i the diffusivity. The reaction terms R_i are polynomials in $C_1, \ldots, C_N$ where the powers of the C_i are determined by stoichiometry, and the coefficients correspond to the reaction rates. The latter are constants depending on T and considered as known in the scope of the "direct problem" which is posed here. Note, however, that most of them are subject to "inverse problems", that is parameter extraction problems.



(a) (b)

Figure 5.6: (a) Layout and (b) grid of a 3D test example with SiO_2 on top of the wafer.

After inserting (5.10) into (5.9) for each i, the PDE system consists of N equations. By employing some equilibrium assumptions, a reduction to a system of typically 3 to 6 equations of a form similar to (5.9) can be performed (see [37]). To be more specific, the substitutions can be performed so that we end up with a system of n_u PDEs each of which of the form

$$\frac{\partial P_i}{\partial t} - \nabla \cdot \left(\sum_{k=1}^{n_u} Q_{ik} \nabla u_k \right) = \widetilde{R}_i \quad (i = 1, \ldots, n_u) \ . \tag{5.11}$$

The $u_1, \ldots, u_{n_u}$ serve as unknowns of the reduced system. Among typical examples are the total concentration of interstitials, I_{tot}, the total concentration of vacancies, V_{tot}, and concentrations of impurity dopants such as arsenic, boron or phosphorus. P_i and R_i are polynomials in $u_1, \ldots, u_{n_u}$, the Q_{ik} rational expressions in these unknowns. Initial conditions for the system (5.11) are obtained from an initial distribution of dopants which is obtained as output from a simulation of the previous process step. For the potential ψ, an additional Poisson equation can be solved, which could be coupled to the above system. We have only investigated the typical, uncoupled case.

5.2.2.2 Discretization

In common process simulators, such as DIOS being part of the GENESISe suite [39], the discretization of the time-dependent, nonlinear PDE system to be solved is performed as follows. An implicit approach is chosen for the time discretization. The spatial discretization is performed by the so-called “box method”on Delaunay grids (cf. [36] and Section 5.3.1.5), and the resulting nonlinear system is linearized by a modified Newton(-Raphson) method. ILU-preconditioned BiCGstab or GMRes are commonly used as solvers for the resulting linear systems. More precisely, modified ILUT or ILU(0) preconditioners are employed.

Since for each time step a nonlinear system and for each nonlinear system a series of linear systems has to be solved, we end up with a large series of linear systems to be solved in each reaction-diffusion simulation. Normally, the smaller the time step, the better converges the Newton iteration and the smaller the condition numbers of the corresponding matrices. In DIOS, a run-time evaluation of the necessary number of linear and nonlinear iterations is used for adaptation of the time step.

5.2.2.3 Exemplary Test Case

For the tests, the SAMG library has been integrated into DIOS. The performance of SAMG has then been investigated for the simulation of an annealing step after an ion implantation step for the 3D model shown in Fig. 5.6. The model consists of a silicon layer with a small silicon oxide layer on top and an inert gaseous N_2 phase as ambient. Therefore, no additional material[21] can enter the layer system. In the annealing process, only the already available silicon point defects and the implanted dopants arsenic and boron react with each other. In the concrete simulation considered, the temperature, starting from 750°C, is increased in each time step and reaches 925°C finally.

The physical unknowns to be determined by the simulation are the total concentrations of interstitials (I_{tot}), vacancies (V_{tot}), and the dopants arsenic (As_{tot}) and boron (B_{tot}). The Delaunay grids used for the simulation contain approximately 181 500 tetrahedra, 38 500 pyramids (with a base of four sides), 2 500 bricks and n_p=61 319 points. The arising matrices have n_v=245 276 rows and $n_A \in [\,7\,028\,004,\ 7\,046\,780\,]$ nonzero matrix entries.

5.2.2.4 Reaction Front and Matrix Properties

Of particular interest are the concentration profiles in and near the *reaction front*, a narrow region, moving from the "implantation surface" of the wafer towards the interior, where fast reactions occur due to large concentration gradients (see [61] for example). Hence, for each unknown u_n, there exists a subdomain with strong variations in space and time, whereas on the complementary subdomain only small changes are observed. Although these "subdomains of strong variations" do not coincide for the individual functions, they all lie inside one narrow reaction front for all applications we have in mind here. For our example (see Fig 5.6), the "implantation surface" is the part on top of the silicon layer which is not covered by the oxide layer. From this surface, the reaction front is moving downwards, in normal direction to the surface, and is more and more spreading out sideways under the oxide layer.

The large concentration gradients inside the reaction front are reflected on matrix level in the magnitude of off-diagonal entries: inside the reaction front, the reaction terms cause very large positive or negative off-diagonal entries in the corresponding rows of the matrices A. Hence, even for reasonable time steps, the matrices become ill-conditioned, leading to serious problems for the standard, one-level iterative solvers used in process simulators. It can be observed that these solvers become less efficient or even stagnate in an unpredictable way. The difficulties often increase during later time steps of a simulation. For our exemplary test case, this is demonstrated in Section 5.2.2.6.

[21] for instance, oxygen, if present, would enlarge the silicon oxide layer.

The individual PDEs (5.11) in the stationary case are "harmless" diffusion equations *in the total absence of the reaction front*. Outside the reaction front, the concentrations change only slowly so that a relatively "harmless" diffusion-dominated system remains to be solved. The reaction terms, however, destroy the dominance of the diffusion terms locally inside the reaction front due to the large concentration gradients. As a result, A and their submatrices $A_{[n,n]}$ are far from being M-matrices and, hence, VAMG or a straightforward UAMG approach do not make sense here.

5.2.2.5 Efficient AMG Approaches

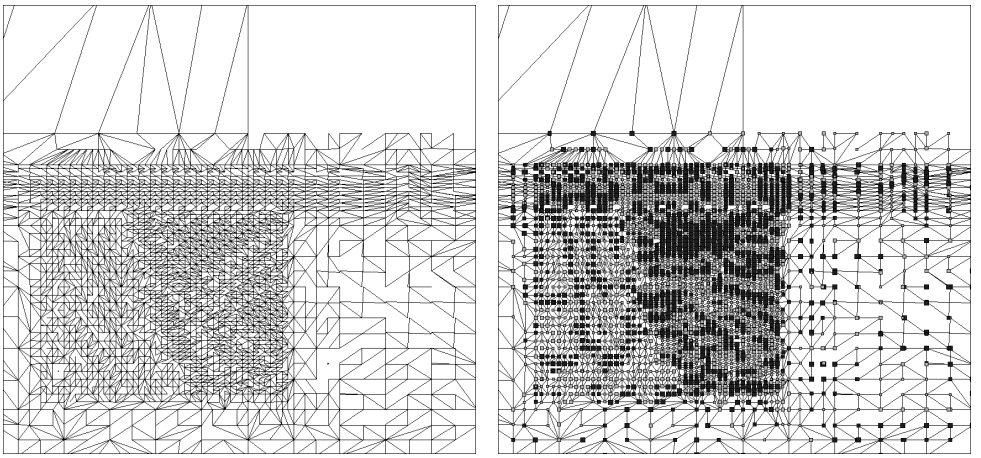
Numerical experiments have shown that, by introducing a very simple modification of AMG's coarsening process, one can make UAMG often work again. The only problem for UAMG is the narrow reaction front corresponding to just a small amount of matrix rows compared with n_v. Therefore, it is tempting to simply not coarsen at all inside this front. To demonstrate this, we have forced all those variables v_i to stay in the coarse levels, whose corresponding rows strongly violate diagonal dominance,

$$\sum_{j \neq i} |a_{ij}| > \sigma_{\text{vio}} |a_{ii}| \,. \tag{5.12}$$

Depending on the threshold parameter $\sigma_{\text{vio}} > 1$, the resulting UAMG approach, employed as a preconditioner, often converges quite reasonably. However, the choice of σ_{vio} is crucial and matrix-dependent, and the above criterion is not always suitable for reliably distinguishing the reaction-dominant from the slow-diffusion part. This is because violation of diagonal dominance can also be due to the Laplacian-type unknown cross-couplings which are also contained in the matrix. Numerical tests reported in the next section will confirm that this modified UAMG approach is not robust enough for a use in practice.

The reaction-diffusion matrices considered here and the RD models discussed in the preceding chapters (see Sections 3.1.3.2, 3.4.1.2 and 4.6 and Examples 3.4 and 3.11) have some important properties in common. Similarly as for the RD models, the fact that the dominance of diffusion is "only disturbed" due to reaction terms can be exploited in order to define efficient PAMG approaches. As can be seen in Fig. 5.7(a), the underlying grids are adaptively refined in the reaction front. Since, for the problem considered, anisotropies are due to non-uniform grid spacings only, a primary matrix based on distances yields an appropriate coarsening for the underlying diffusion problem, as has been discussed in Section 3.4.2.4. Fig. 5.7(b) depicts a typical coarse level created by distance-based coarsening.

It is promising now to use an accelerated PAMG approach in order to address all disturbances caused by the reaction terms. Similarly as for the RD models, BGS (or ILU) smoothing helps to handle the large unknown cross-couplings. However, an important difference to the RD models is that these couplings also occur outside the $A_{(k,k)}$. As has heuristically been explained in 3.11, a distance-based primary matrix together with an MU-interpolation with weights also based on distances makes PAMG an efficient preconditioner here. Numerical results, presented in Section 5.2.2.6, show that the proposed method treats the "pure" diffusion outside the reaction front as well as the fast reactions inside properly.



(a) (b)

Figure 5.7: (a) Part of a 2D cross section of the 3D grid of Fig. 5.6 which is adaptively refined near an edge of the interface oxide/wafer. Apparently missing edges are due to the fact that 3D elements have been cut. (b) Finest and, depicted by filled black boxes, the next coarser level for the grid on the left.

Since, particularly due to the Newton process, series of similar matrices have to be solved, it is feasible to reuse parts of SAMG's setup [22]. Therefore, a control mechanism has been integrated into the DIOS-SAMG interface. It performs a full setup at the beginning of a new Newton process. For the following matrices belonging to the same nonlinear problem, the level hierarchy is reused, and only the Galerkin operators are calculated from scratch. The control mechanism also checks if convergence problems arise. However, for the simulation run with the test case described above, no problems occur. Numerical results illustrating the effectiveness of the proposed overall approach are discussed in the next Section 5.2.2.6. Let's make a remark first.

Remark 5.6 It should be mentioned that in the past also geometric multigrid methods have been developed for certain problems involving diffusion processes. For instance, nonlinear geometric multigrid methods (FAS) with adaptive multilevel grid selection strategies for certain diffusion-oxidation evolution processes are the topic of [41]. Since the extension of such geometric multigrid techniques to the described *reaction*-diffusion problems on *unstructured* grids is not straightforward, if feasible at all, they are not discussed here. ▲

5.2.2.6 Numerical Results

We now demonstrate for the test case described in Section 5.2.2.3 that SAMG outperforms standard iterative preconditioners with respect to both stability of the convergence as well as

[22] This belongs to SAMG's features, see Section 4.1.1.

computing time. For this purpose, we investigate the performance for both the solution of single matrices and the whole DIOS simulation run.

Efficiency of AMG for Individual Matrices We start with investigations for individual matrices. They have been extracted from the simulation run performed with DIOS' standard iterative solver, an ILU(T)-GMRes method with certain parameter adaptations dynamically reacting on the solver performance during the run. DIOS uses ϵ=1e-4 as a default. Results are discussed in detail now for two matrices:

- A_{rd1} denotes the first matrix arising in the Newton process for time $t = 0.004$ (752.4°C).
- A_{rd2} denotes the last matrix of the Newton process for time $t = 0.058$ (785.0°C).

Whereas the DIOS solver needs 24 iterations for A_{rd1} to reach ϵ=1e-4, it needs 104 for A_{rd2}.

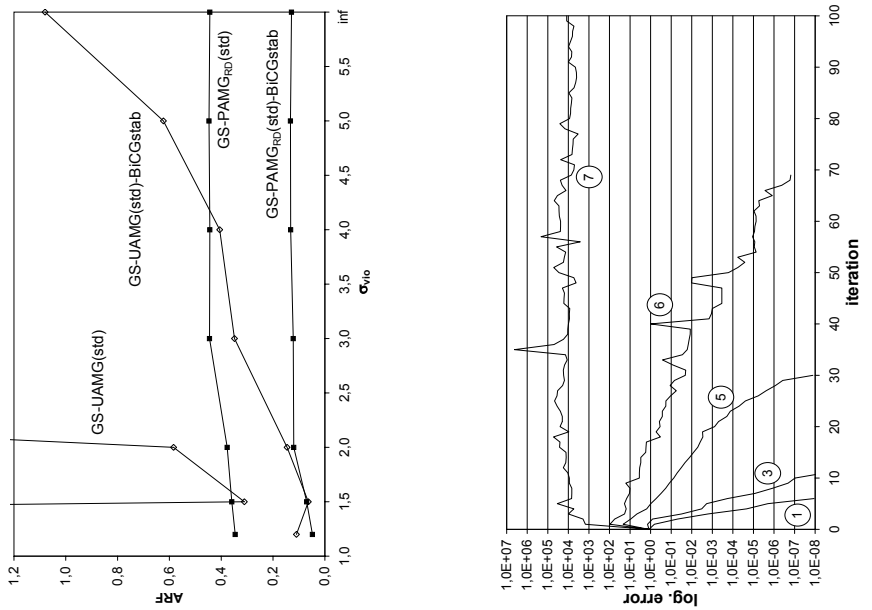
Remark 5.7 It does not make sense to choose matrices from late time steps t since the concrete linear solver chosen has a large impact on all matrices evolving in the current Newton process and also all Newton processes for subsequent time steps. ▲

preconditioner	c_g	$c_{\boldsymbol{P}}$	c_A	c_{prec}	$\mathrm{m}_{\mathrm{tot}}$	c_{tot}	time
ILU(0)-UAMG(agg)	1.21	0	1.44	1.58	283.63	3.26	21.7
ILU(0)-UAMG(std)	1.61	0	3.07	3.18	557.66	6.42	38.1
UGS-UAMG(agg)	1.21	0	1.44	0.90	163.77	1.88	(stag)
UGS-UAMG(std)	1.61	0	3.07	2.22	384.39	4.42	108.8
ILU(0)-PAMG$_{\mathrm{RD}}$(agg)	1.36	1.69	1.68	1.93	345.49	3.97	22.8
ILU(0)-PAMG$_{\mathrm{RD}}$(std)	1.86	3.13	3.23	3.63	638.72	7.35	39.2
BGS-PAMG$_{\mathrm{RD}}$(agg)	1.36	1.69	1.68	1.18	204.58	2.35	31.0
BGS-PAMG$_{\mathrm{RD}}$(std)	1.86	3.13	3.23	2.40	414.47	4.77	44.0
ILU(0)	1	0	1	1	184.98	2.10	31.1

Table 5.7: Complexities and timings for the A_{rd2} example. σ_{vio}=1.2 for UAMG. Accelerator always BiCGstab. $\mathrm{m}_{\mathrm{tot}}$ = overall memory requirements [MBytes] including mem(A) = memory needed for finest-level matrix = 86.93 MBytes, c_{tot} = $\mathrm{m}_{\mathrm{tot}}$/mem$(A)$, time = wall-clock time for whole run. "stag" means stagnation of the approach. In particular for comparing timings, see explanations in the text!

Corresponding to a general experience, also here the AMG approaches achieve their best performance in terms of computational time when used as preconditioners. It has turned out that the performance of both BiCGstab and GMRes(20) is comparable here. This is also true when used with ILU(0) or ILUT preconditioning. However, since BiCGstab needs less memory, this accelerator is more efficient and is thus chosen for the following tests.

Regarding PAMG variants, approaches with a norm-based $\boldsymbol{P}$ and scaling of interpolation (see Section 4.3.1.5) work sometimes, but they are less stable than approaches with a distance-based $\boldsymbol{P}$. Whereas approaches with a block-interpolation often diverge here, and approaches with an SU-interpolation show an inconsistent behavior, MU-interpolation with



(a) (b)

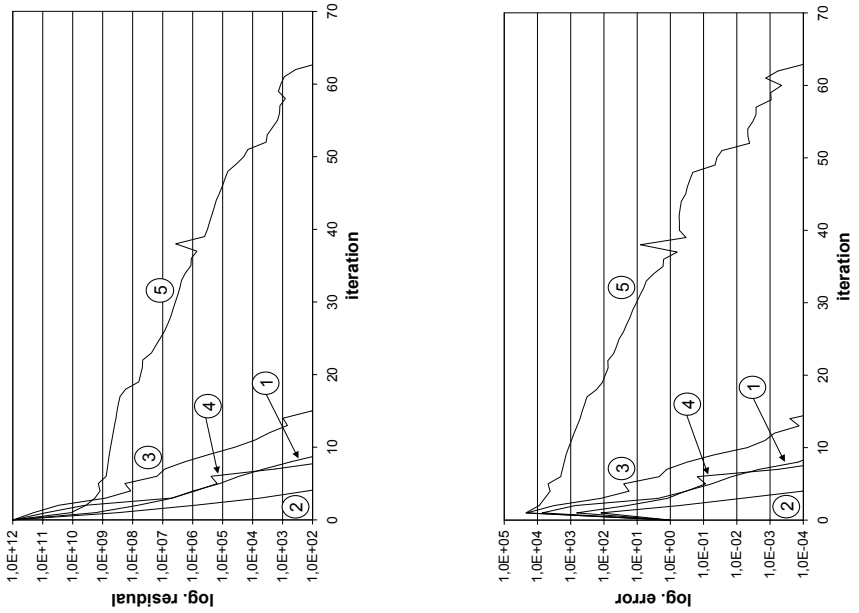
Figure 5.8: (a) Dependence of UAMG's and PAMG's convergence behavior on σ_{vio} for the A_{rd1} example. (b) Error histories of several solvers for the A_{rd1} example. BiCGstab with 1: ILU(0)-PAMG$_{\mathrm{RD}}$(agg), 3: BGS-PAMG$_{\mathrm{RD}}$(agg), 5: ILU(0), 6: ILUT(3), 7: ILUT(5).

weights based on distances has been found to be robust [23]. Therefore, among the point-based approaches, PAMG(dist,·,MU,dist) is the most robust solver and preconditioner. In the following, we refer to this approach as $\mathbf{PAMG_{RD}}(\cdot)$. Both ILU(0) and BGS yield efficient smoothers for this approach. ILU(0), however, needs much more memory. As can be seen in Table 5.7 for matrix A_{rd2}, aggressive coarsening considerably reduces memory requirements and often also the overall computing time compared to the respective variant with standard coarsening.

Exemplarily for matrix A_{rd1}, convergence rates of UAMG(std) and PAMG$_{\mathrm{RD}}$ as functions of σ_{vio} (5.12) are depicted in Fig. 5.8(a). Obviously, the choice of σ_{vio} is very crucial for UAMG. Whereas PAMG$_{\mathrm{RD}}$'s performance is stable for all σ_{vio} even without BiCGstab, UAMG in the stand-alone version converges only in a small intervall. BiCGstab is able to improve UAMG's convergence, however, the convergence quickly deteriorates with increasing σ_{vio}. Unfortunately, with decreasing σ_{vio}, the complexity[24] and thus the computing time suffers considerably. Qualitatively the same trends are obtained with ILU(0)-smoothing.

[23] In contrast to this, an interpolation with coordinates-based weights does not help improving *UAMG's* performance.

[24] For very small σ_{vio}, too many variables are forced into C.



(a) (b)

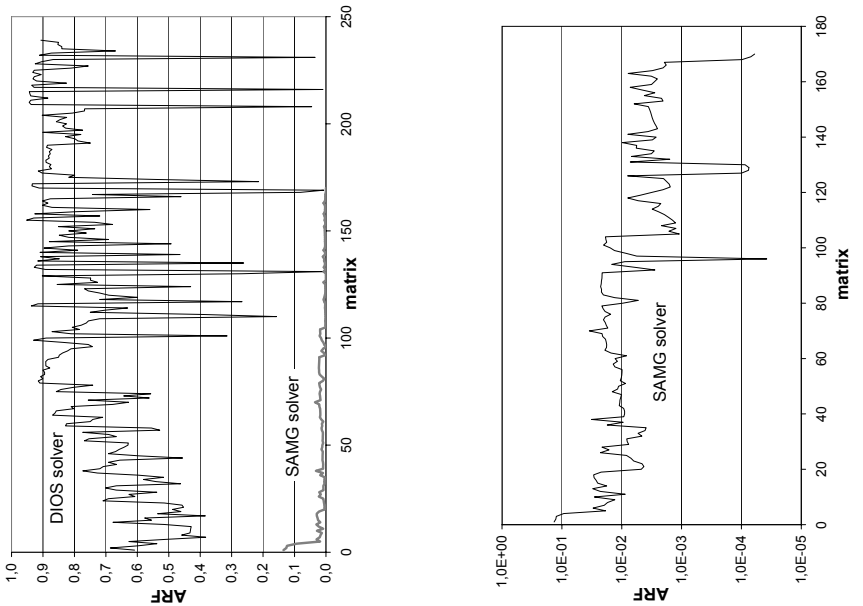
Figure 5.9: (a) Residual and (b) error histories for the A_{rd2} example. Preconditioners 1,3,5 as in Fig. 5.8(b), 2: ILU(0)-PAMG$_{\mathrm{RD}}$(std), 4: BGS-PAMG$_{\mathrm{RD}}$(std).

UAMG(-BiCGstab)'s performance strongly depends on both σ_{vio} and the concrete matrix. For instance, whereas the optimal[25] σ_{vio} for A_{rd1} is 1.5, it is 1.2 for A_{rd2}. Moreover, for A_{rd2} for instance, UGS-UAMG(std)-BiCGstab is slow and UGS-UAMG(agg)-BiCGstab even stagnates for this matrix (see Table 5.7). In contrast to this, PAMG$_{\mathrm{RD}}$-BiCGstab with both BGS and ILU(0) shows qualitatively the same stable behavior for all matrices tested.

Regarding computing times, ILU(0)-UAMG-BiCGstab for the optimal σ_{vio} is a bit faster than ILU(0)-PAMG$_{\mathrm{RD}}$-BiCGstab for the two matrices A_{rd1} and A_{rd2} (see Table 5.7). However, for only nearly-optimal σ_{vio}, the speed of ILU(0)-UAMG-BiCGstab considerably deteriorates due to either a higher complexity or a worse convergence so that PAMG$_{\mathrm{RD}}$ approaches outperform UAMG then. In particular, PAMG$_{\mathrm{RD}}$-BiCGstab with aggressive coarsening and BGS-smoothing leads to a cheap and quite fast variant, whereas the same approach but with ILU(0) smoothing needs more memory but needs considerably less overall computing time.

Comparison with One-Level Solvers In the following, we present a comparison of the four PAMG$_{\mathrm{RD}}$-BiCGstab variants mentioned above with classical one-level solvers. For all matrices tested, the performance of the PAMG$_{\mathrm{RD}}$-BiCGstab variants and of ILU(0)-

[25] in terms of the convergence of the stand-alone ILU(0)-UAMG(std) approach.



(a) (b)

Figure 5.10: Comparison of ARFs: (a) DIOS' standard solver and SAMG, (b) SAMG only.

BiCGstab essentially lies in the "range" defined by Figs. 5.8(b) and 5.9. That is, the results for the other matrices tested correspond to 5.8(b) or 5.9 or lie somewhere in between. It can be seen that all PAMG$_{\rm RD}$-BiCGstab variants tested here converge much faster than the ILU(T)-preconditioned ones[26].

Fig. 5.9 indicates[27] a typical behavior: It is a general observation that ILU(0)-BiCGstab does typically not reduce errors and residuals simultaneously. The error reduction is behind, in particular if only a residual reduction of ϵ=1e-4, say, is demanded, which is the default in DIOS. In contrast to this, PAMG$_{\rm RD}$-BiCGstab reduces residuals and errors nearly "simultaneously".

Complexities, total memory requirements and total timings for different AMG approaches and ILU(0) are shown in Table 5.7. The PAMG$_{\rm RD}$-BiCGstab variants with aggressive coarsening show reasonable complexities. In particular, we have $c_{\rm prec} < 2$ here, that is, both variants need only less than twice more memory than ILU(0) [28]. Both variants are faster than or as fast as ILU(0)-BiCGstab. Note that here the time required to reduce *residuals* is mea-

[26] ILUT-BiCGstab does not always converge. Interestingly, if it converges, the larger the level $l_{\rm fill}$ of fill-in, the worse the convergence of ILUT($l_{\rm fill}$)-BiCGstab - at least for small fill-ins. ILU(0) has turned out to be both the best one-level preconditioner as well as the best smoother for PAMG$_{\rm RD}$ in terms of convergence rates.

[27] even if Fig. 5.9 is still rather advantageous for ILU(0)-BiCGstab here. For matrices arising at later time steps we can expect ILU(0)-BiCGstab to perform worse, see also Fig. 5.10.

[28] Recall from Table 4.1 that $c_{\rm prec} < 2$ has also been obtained for our model problems.

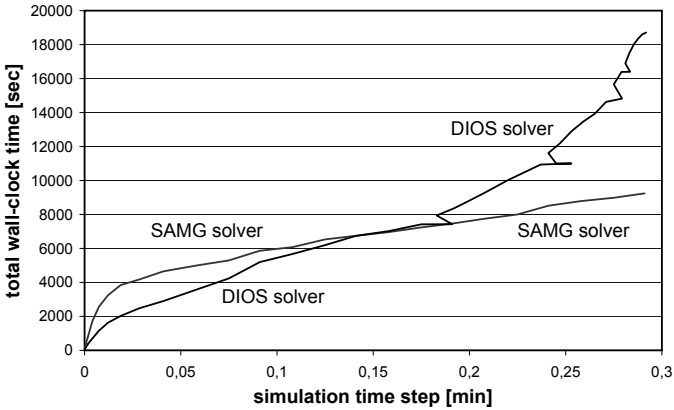


Figure 5.11: Total wall-clock computing time before computations for the current simulation time step have started *versus* simulation time step for the 3D reaction-diffusion example.

solver	SAMG	DIOS
number of time step reductions	0	5
number of nonlinear solves N_{ns}	23	41
number of linear solves N_{ls}	170	239
ratio N_{ls}/N_{ns}	7.39	5.83
number of new matrix structures	25	37
total computing time [sec]	9242.1	18907.4
ratio of computing times	1	2.05

Table 5.8: Results of a DIOS simulation run for the 3D reaction-diffusion example. Shown are performance data of the SAMG solver chosen, BGS-PAMG$_{\mathrm{RD}}$(agg)-BiCGstab, and the standard iterative DIOS solver, respectively.

sured. However, as mentioned above, DIOS stops a linear solve step if the residual is reduced by ϵ=1e-4. As can be seen in Fig. 5.9 for A_{rd2}, ILU(0)-BiCGstab has then reduced the error by approximately 5e-2 only, whereas the PAMG$_{\mathrm{RD}}$-BiCGstab variants have reduced the error by approximately 5e-4. Therefore, even if ILU(0)-BiCGstab is seemingly not slower than BGS-PAMG$_{\mathrm{RD}}$(agg)-BiCGstab and needs a bit less memory, BGS-PAMG$_{\mathrm{RD}}$(agg)-BiCGstab is more favorable due to the much better *error* reduction. In summary, the PAMG$_{\mathrm{RD}}$(agg)-BiCGstab approaches are clearly the favorable choices here. BGS-PAMG$_{\mathrm{RD}}$(agg)-BiCGstab is a good compromise of error reduction, speed and memory requirements.

Results of the Full DIOS Simulation Run The behavior described for single matrices is typical for a whole simulation: PAMG$_{\mathrm{RD}}$-BiCGstab yields stable and superior convergence rates for residuals *and* errors in all Newton iterations for *each* time step. The PAMG$_{\mathrm{RD}}$(agg)-

BiCGstab variants are faster[29], more robust, and thus considerably more efficient than ILU(0)- or ILUT-preconditioned one-level solvers. This is true even if only the slower, but less memory-consuming of both favorable PAMG_{RD}(agg)-BiCGstab approaches is chosen, namely the one with BGS smoothing. This is shown for the full DIOS simulation run now.

In Table 5.8 and Figs. 5.10 and 5.11, the performance of DIOS's original run is compared with a run where BGS-PAMG_{RD}(agg)-BiCGstab has been chosen as the linear solver for all matrices. The following conclusions can be drawn. Whereas in the original DIOS run 5 time-step rejections occur, each of which with a complete remeshing step, this does not happen when SAMG is chosen. Due to this and its generally better error-reduction properties, SAMG reduces the number of nonlinear iterations considerably. Since the average number of linear solves necessary per nonlinear step is just a bit higher for SAMG than for the DIOS solver, the total number of linear solves necessary is substantially reduced when using SAMG. Also the number of new matrix structures which have to be set up by DIOS is reduced. Altogether, these results demonstrate clearly that SAMG stabilizes the whole simulation. Whereas the performance of the DIOS solver substantially degrades in later steps of the simulation, SAMG's performance improves. As a consequence, SAMG clearly outperforms the DIOS solver in the second half of the simulation run, as can be seen, in particular, in Fig. 5.11. In total, SAMG performs the full simulation more than twice as fast as the DIOS solver.

5.3 Device Simulation

Semiconductor device simulation aims at the computation of the electrodynamic behavior of a self-contained semiconductor device under various operating conditions. These can be, for example, different voltages applied to contacts of the device. The results of a device simulation are time- and spatially dependent functions as well as net quantities. The former are the electrostatic potential, the concentration of electrons and holes (and, depending on these three functions, the electron and hole current densities), and - if not assumed to be an input to the simulator - the device temperature distribution. The latter are typically *current-voltage characteristics* (*IV-characteristics*[30]).

There exists an extended hierarchy of semiconductor models, ranging from quasi-hydrodynamic to kinetic and classical to quantum models. In [43] in form of a survey and in [55] in more detail, a hierarchy of the most important models is presented and discussed. On the highest level in this hierarchy are the kinetic models, i.e. the semi-classical semiconductor Boltzmann equation and the quantum Boltzmann equation. The numerical simulation of Boltzmann systems has been carried out by Monte-Carlo or deterministic particle methods. Since they are very expensive, simpler fluid dynamical models have been derived, one of them being the hydrodynamic equations. The quasi-hydrodynamic models range from (quantum) hydrodynamic and Schroedinger-Poisson over (quantum) energy-transport to (quantum) drift-diffusion models, which is the lowest level considered here.

The simplest quasi-hydrodynamic model is the standard drift-diffusion system. Compared with higher dimensional, more involved models, it provides less accurate local poten-

[29] if not only the residual reduction but also the error reduction is taken into account, as explained above.

[30] see Section 5.3.2.4 for an example.

tials and concentrations, but often predicts (current densities and) net quantities similarly reliably with much less computational effort. Although the local potentials and concentrations would give a more detailed insight into the functioning of the device considered, engineers are typically interested in current densities and global IV-characteristics[31]. In the engineering environment, the quite complex energy-transport equations are used mainly to compute data for model parameters in the drift-diffusion equations, whereas the simpler, "cheaper" drift-diffusion systems are commonly used to determine current densities and IV-characteristics.

It should be pointed out that, with increasing miniaturization of the devices and the use of novel device structures and other materials instead of silicon, drift-diffusion models reach the limit of their validity. However, since the microscopic effects not described by them occur only locally, in some small parts of the device, it might be feasible to use more sophisticated models also only locally. An example for such an extended model might be the consideration of quantum mechanical effects only in the channel of a MOSFET. Therefore, drift-diffusion systems will remain an important tool for investigating the behavior also of the coming generation of semiconductor devices.

This part of the thesis is concerned with the efficient application of AMG approaches to the linear systems arising in the numerical solution of drift-diffusion systems for semiconductor devices. The focus lies entirely on *stationary* simulations since these are typically performed in an industrial environment.

In the next section, the standard drift-diffusion model for semiconductor device simulation is described. In Section 5.3.2, we discuss properties and the efficient solution of the arising matrix equations. In particular, we will show that point-based AMG approaches employing a norm-based primary matrix can yield preconditioners which are more robust and often more efficient than the standard one-level preconditioners commonly used in (industrial) device simulation.

5.3.1 The Standard Drift-Diffusion Model

In the following, a description of the transient and the stationary drift-diffusion models for semiconductor devices, the spatial simulation domain and boundary conditions is given. We then concentrate on aspects with a large impact on the arising systems of linear equations. In particular, we discuss - in brief - appropriate scalings and the singular perturbation character of the system, layer behavior and conditioning. Finally, the commonly used discretization and linearization techniques are outlined. With the exception of Section 5.3.1.1, we consider the stationary case.

5.3.1.1 The Transient Basic Semiconductor Equations

The standard drift-diffusion model consists of a set of so-called *basic semiconductor equations*. They can be derived, for instance, from Maxwell's equations, several relations obtained from solid-state physics and some further (rather simplified) assumptions. Details on their derivation are given in [82, 55], for example. The basic semiconductor equations can be

[31] to use the latter as input for circuit simulations, for example.

written as follows:

$$-\nabla\cdot(\epsilon_s\nabla\psi)+q(n-p-C)=0 \qquad \text{Poisson(-type) equation,} \tag{5.13}$$

$$q\frac{\partial n}{\partial t}-\nabla\cdot J_n+qR(\psi,n,p)=0 \qquad \text{electron continuity equation,} \tag{5.14}$$

$$q\frac{\partial p}{\partial t}+\nabla\cdot J_p+qR(\psi,n,p)=0 \qquad \text{hole continuity equation,} \tag{5.15}$$

$$J_n=-q(\mu_n n\nabla\psi-D_n\nabla n) \qquad \text{electron current relation,} \tag{5.16}$$

$$J_p=-q(\mu_p p\nabla\psi+D_p\nabla p) \qquad \text{hole current relation.} \tag{5.17}$$

$\psi=\psi(x,t)$ denotes the electrostatic potential, and $n=n(x,t)$ and $p=p(x,t)$ the electron and hole carrier concentrations, respectively. x denotes the independent spatial variables (usually three-dimensional), and t the time. J_n and J_p are the densities of the electron and hole current, respectively. ϵ_s represents the permittivity[32]. q is the elementary charge[33], and $C=C(x)$ the net impurity concentration[34]. $R=R(\psi,n,p)$ denotes the recombination-generation term, μ_n and μ_p the mobilities, and D_n and D_p the diffusivities. Often, Einstein's relations are assumed to hold:

$$D_n=U_T\mu_n \quad\wedge\quad D_p=U_T\mu_p \tag{5.18}$$

where $U_T=k_BT/q$ is the thermal voltage, k_B the Boltzmann constant[35], and T the device temperature, which is often treated as a constant[36] (see below).

There are various models for the mobilities and the recombination-generation. Mobilities and diffusivities are always positive. In general, they are functions. Typically, we can assume μ_n to vary between 50 and 1500 $\text{cm}^2\ \text{V}^{-1}\ \text{s}^{-1}$ and μ_p between 50 and 500 $\text{cm}^2\ \text{V}^{-1}\ \text{s}^{-1}$ for silicon at room temperature. The most basic recombination-generation process, namely two-particle transition, is described by the Shockley-Read-Hall term,

$$R_{SRH}=\frac{np-n_{intr}^2}{\tau_p^l(n+n_{intr})+\tau_n^l(p+n_{intr})}\ , \tag{5.19}$$

where τ_n^l and τ_p^l denote the electron and hole life-times, respectively, and n_{intr} the intrinsic carrier concentration[37]. Three-particle transition is modeled by Auger recombination-generation[38],

$$R_{AU}=(C_n^{AU}n+C_p^{AU}p)(np-n_{intr}^2)\,, \tag{5.20}$$

[32] a three-dimensional tensor, but usually assumed to be a scalar constant whose approximate value in silicon is $1.594\cdot10^{-10}\ \text{As V}^{-1}\ \text{m}^{-1}$.

[33] $q=1.6021892\cdot10^{-19}$ As.

[34] i.e. the doping profile. C is defined to be the difference of the concentrations n_{dp} and p_{dp} of the electron and the hole carriers, respectively, contained in the device after its fabrication: $C(x)=n_{dp}(x)-p_{dp}(x)$.

[35] $k_B=1.380662\cdot10^{-23}\ \text{V As K}^{-1}$.

[36] Assuming an ambient temperature of $T=300\,\text{K}$, we obtain $U_T=0.025852\,\text{V}$ then.

[37] For silicon at room temperature, τ_n^l =1e-6 s, τ_p^l=1e-5 s, see [54]. The intrinsic carrier concentration is defined to be the geometric average $\sqrt{n_0p_0}$ of the carrier concentrations n_0, p_0 in a semiconductor in equilibrium. $n_{intr}\approx 9.65\cdot10^9\ \text{cm}^{-3}$ for silicon at room temperature.

[38] with C_n^{AU} =2.8e-31 cm^6s^{-1} and C_p^{AU} =9.9e-32 cm^6s^{-1} for silicon at room temperature [54].

and impact ionization, which is extremely significant at high electric fields, by the impact ionization (or avalanche) generation rate,

$$R_I = -\alpha_n \frac{|J_n|}{q} - \alpha_p \frac{|J_p|}{q} \tag{5.21}$$

with some constants α_n, α_p. If necessary, the three rates are linearly superimposed, that is $R = R_{SRH} + R_{AU} + R_I$. Often R_I and quite often R_{AU} are neglected. If only R_{SRH} or $R_{SRH} + R_{AU}$ are considered, we write $R = R_{SRH,AU}$.

By inserting equations (5.16) and (5.17) into (5.14) and (5.15), respectively, and multiplying the resulting equations by $1/q$ we obtain the following set of partial differential equations of second order which shall be solved for ψ, n and p:

$$-\nabla \cdot (\epsilon_s \nabla \psi) + q(n - p - C) = 0 \,, \tag{5.22}$$

$$\frac{\partial n}{\partial t} + \nabla \cdot (\mu_n n \nabla \psi - D_n \nabla n) + R = 0 \,, \tag{5.23}$$

$$\frac{\partial p}{\partial t} - \nabla \cdot (\mu_p p \nabla \psi + D_p \nabla p) + R = 0 \,. \tag{5.24}$$

If thermal effects (for instance, thermal breakdown phenomena) are to be investigated, the device temperature T (see (5.18)) cannot be assumed to be a known input to the device simulator any longer, but has to be computed. For this purpose, the above system is extended by a heat flow equation:

$$\rho c \frac{\partial T}{\partial t} - H - \nabla \cdot k(T) \cdot \nabla T = 0 \tag{5.25}$$

where ρ is the specific mass density, c the specific heat of the material, H the thermal generation (depending on J_n and J_p), and $k(T)$ the thermal conductivity (usually modeled as a rational function of T). However, since in most device simulations the device temperature is assumed to be known ($T = 300\,\mathrm{K}$), we will restrict ourselves to the system (5.22)-(5.24).

A detailed description of all physical models and parameters mentioned can be found in [82, 54, 55] where also mathematical analyses of the models are carried out, and some results on existence and (non-)uniqueness of solutions are given. We want to note here that (local) uniqueness of the solution of the drift-diffusion system (5.22)-(5.24) can be proved if $\mu_n, \mu_p > 0$ and $R = R_{SRH,AU}$, that is if avalanche phenomena are excluded, and if the potentials applied to the device are sufficiently small. However, there are physically relevant cases where the solution is not unique, a prominent example being the snap-back phenomenon (hysteresis) in thyristor technology.

5.3.1.2 The Stationary Basic Semiconductor Equations

In general, ψ, n and p are functions of position x *and* time t. Since the mobilities and diffusivities are positive, each of the three equations (5.22)-(5.24) is parabolic (under natural conditions on ψ, n, p and R). However, if all potentials which are externally applied to the device contacts are time-independent, that is if the boundary conditions for ψ (see also Section 5.3.1.3) are time-independent, the problem is reduced to the following set of stationary

basic semiconductor equations

$$-\nabla \cdot (\epsilon_s \nabla \psi) + q(n - p - C) = 0\,, \tag{5.26}$$
$$\nabla \cdot (\mu_n n \nabla \psi - D_n \nabla n) + R = 0\,, \tag{5.27}$$
$$\nabla \cdot (-\mu_p p \nabla \psi - D_p \nabla p) + R = 0\,. \tag{5.28}$$

Each individual equation is elliptic now (under natural conditions on ψ, n, p and R). In [54], a detailed analysis of the system (5.26)-(5.28) is performed.

Simulations of the stationary case are carried out to investigate steady states of devices sufficiently long after switching processes. *Since these are the typically performed simulations in industry, we concentrate only on them in the following.*

5.3.1.3 Simulation Domain and Boundary Conditions

The simulation domain $\Omega \subset \mathbb{R}^d$, which is normally three-dimensional ($d = 3$) and only in simple cases two-dimensional ($d = 2$), consists of two parts, Ω_s and Ω_o. Ω_s represents the union of all material layers where the above-described coupled system of stationary semiconductor equations shall be solved. This is typically the case in the semiconductor layers, in particular the wafer (often a doped silicon substrate). Ω_o is defined as the union of layers for which it is assumed that (nearly) no charge carrier currents can occur. In particular, insulating layers belong to Ω_o. In Ω_o, the above system degenerates to Laplace's equation,

$$-\nabla \cdot (\epsilon_o \nabla \psi) = 0\,, \tag{5.29}$$

where ϵ_o represents the permittivity of the corresponding material layers. In case of a MOSFET, Ω_o represents the gate oxide. Here, the interface between Ω_s and Ω_o is the semiconductor/oxide-interface. The simplified sketch in Figure 5.2(a) depicts different domains, interfaces and boundaries of an n-MOSFET. Some explanations follow.

An **n-domain** (or n-region) is defined as a subdomain of Ω_s in which $C(x) > 0$ holds. Analogously, a **p-domain** (or p-region) is defined as a subdomain in which $C(x) < 0$ holds. The interface between an n- and an adjacent p-domain is called **pn-junction**. The junction is called abrupt if C exhibits a jump across the interface.

The n- and p-domains determine the electrical behavior of the device to a large extent since they form local diodes at the pn-junctions. For instance in case of an n-MOSFET, a p-domain is located between two n-domains as shown in Fig. 5.2(a). Hence, two diodes (or a triode "NPN") are formed in such a way that a current cannot flow from one n-domain to the other. However, under appropriate conditions on the bias applied to the contacts of the device, a new n-domain connecting the two other n-domains is formed inside the p-domain. This n-domain is called a **channel** since it allows electrons to move from one (original) n-domain to the other. Basically, by means of the bias applied to gate/bulk, the "height" (measured perpendicular to the Ω_s/Ω_o-interface) of the channel is determined.

The boundary of Ω can be split into two disjoint parts, $\partial\Omega = \partial\Omega_p \,\dot\cup\, \partial\Omega_a$. $\partial\Omega_p$ represents those parts of $\partial\Omega$ which correspond to real "physical" boundary segments, that is interfaces with insulating material and contacts. $\partial\Omega_a$ consists of artificial boundary segments, which are introduced, for example, to reduce the simulation domain by cutting off the "bottom" part

of the bulk as much as possible - the wafer is rather thick -, and to obtain a "self-contained" device, that is to separate it from adjacent devices if it is is embedded in an integrated circuit, for instance.

Different types of boundary and interface conditions are prescribed on different parts of $\partial\Omega$ and the interfaces between Ω_s and Ω_o. Details can be found in [82], for instance. We want to mention only the conditions usually employed:

- Dirichlet conditions are defined on the parts of $\partial\Omega_s$ corresponding to purely voltage-controlled Ohmic contacts (for example, the source and drain contacts of a transistor).
- For voltage-driven Schottky contacts[39], a highly simplified model is often used, resulting in a Dirichlet condition for ψ and Neumann boundary conditions for J_n and J_p, which can be transformed into mixed boundary conditions for n and p, or Dirichlet conditions for n and p, too.
- On artificial and insulating boundaries, homogeneous Neumann boundary conditions for ψ, J_n and J_p are usually prescribed.
- On the interfaces between Ω_s and Ω_o, (homogeneous) Neumann conditions for J_n and J_p and a Neumann condition for $\epsilon_s\psi - \epsilon_o\psi$ are assumed.

5.3.1.4 Scaling, Layer Behavior and Conditioning

Since ψ, n and p are very different in magnitude and show a strongly different behavior, these three physical functions as well as the three equations should be scaled appropriately to allow for a structural analysis and an efficient numerical solution. Different scalings have been described in the literature and are used in simulation packages. One standard set of scaling factors was introduced by DeMari [20]. The factors are summarized in Table 5.9. The scaled system reads

$$-\Delta\psi + (n - p - C) = 0\,, \tag{5.30}$$

$$\nabla \cdot (\mu_n n \nabla\psi - \mu_n \nabla n) + R = 0\,, \tag{5.31}$$

$$\nabla \cdot (-\mu_p p \nabla\psi - \mu_p \nabla p) + R = 0\,. \tag{5.32}$$

All quantities are scaled here [40], the equations live on the correspondingly scaled domain Ω_{scal}, and the differential operators are taken with respect to the scaled independent variables. From a mathematical point of view, this scaling is unsatisfactory because the physical functions are still very different in magnitude. In spite of this, DeMari-type scalings are frequently used even today.

A "singular perturbation scaling" (Table 5.9) which has been regarded as being more appropriate for theoretical and numerical analysis was introduced by Vasiléva et al. and further investigated by Markowich, Selberherr et al. (see [82, 54] and the references given therein).

[39] The metal/semiconductor junction which is used, for example, as the MESFET (metal-semiconductor transistor) gate is called a Schottky(-barrier) contact. For more details, see [93].

[40] but denoted by the same symbols as before in order to simplify the notation.

quantity	symbol	DeMari factor	symbol	SPS factor
ψ	ψ_m	k_BT/q	ψ_0	k_BT/q
$n,\ p,\ C$	C_m	n_{intr}	C_0	$\max\{\|C(x)\|, x\ \in\ \Omega\ \}$
x	x_m	$\sqrt{\epsilon_s k_B T/(q^2 n_{intr})}$	x_0	$\max\{\|x-y\|, x, y\ \in\ \Omega\ \}$
$D_n,\ D_p$	D_m	$1\ \mathrm{cm}^2\mathrm{s}^{-1}$	D_0	$\max\{(D_n(x), D_p(x)), x\ \in\ \Omega\ \}$
$\mu_n,\ \mu_p$		D_m/ψ_m		D_0/ψ_0
R		$D_m C_m/x_m^2$		$D_0 C_0/x_0^2$
t		x_m^2/D_m		x_0^2/D_0

Table 5.9: DeMari scaling factors and "singular perturbation scaling" (SPS) factors.

The scaling factors can be very different in size compared with the DeMari factors. For example, the scaling factor for $n,\ p,\ C$ is approximately 10, for R approximately 12 magnitudes larger than the corresponding DeMari factors (for a standard situation as described in [82]). The scaled system now reads

$$\lambda^2 \Delta\psi - (n - p - C) = 0\ , \tag{5.33}$$

$$\nabla \cdot (\mu_n n \nabla\psi - \mu_n \nabla n) + R = 0\ , \tag{5.34}$$

$$\nabla \cdot (-\mu_p p \nabla\psi - \mu_p \nabla p) + R = 0 \tag{5.35}$$

with $\lambda^2 = \frac{\psi_0 \epsilon_s}{x_0^2 q C_0}$ denoting the squared scaled minimal normed Debye length of the device which is a very small parameter in practice[41]. The scaled continuity equations are formally identical to the ones obtained by DeMari scaling. Of course, however, $\psi,\ n,\ p$ are scaled by the "singular perturbation scaling" factors now and again denoted by the same symbols as before. In Poisson's equation, the very small factor λ^2 appears in front of the second-order derivatives, the highest ones here. Therefore, this scaling shows the *singular perturbation character* of the system and allows for a more rigorous mathematical analysis via a singular perturbation approach with λ^2 being the *singular perturbation parameter*. In the following, we summarize important results of this analysis, as performed in [54], for instance, on *layer behavior* and *conditioning*. We start with statements on the occurrence of layers[42].

The "transition" interval within which a solution (ψ, n or p) is not approximated to order $O(\lambda)$ by the solution of the corresponding *reduced problem*[43] is called a (zeroth-order) **layer**. Typically we have $\lambda \ll 1$ and $\lambda \approx \delta$ with $\delta^2 := \frac{n_{intr}}{C_0}$. Then the solutions of the stationary device problem (under "moderate injection", see [82]) exhibit the following features:

- There are thin layer strips at (abrupt) pn-junctions, Schottky contacts and semiconductor/oxide interfaces. Within these layers, ψ, n and p and generally the tangential components of J_n and J_p are rapidly varying functions.

[41] λ^2 is smaller than 10^{-7}:

$$\lambda^2 = \frac{k_B T \epsilon_s}{\max|x-y|^2 q^2 \max|C(x)|} \approx \frac{1.4 \cdot 10^{-23} \cdot 300 \cdot 1.6 \cdot 10^{-10}}{1.6^2 \cdot 10^{-38} \max|x-y|^2 \max|C(x)|}$$

[42] The term "layer" in the context of "layer behavior" is to be distinguished from "material layer"!

[43] The reduced problem emerges by setting λ=0 in the system (5.33)-(5.35) with boundary conditions.

- Outside the junction-, Schottky contact- and semiconductor/oxide-interface-layers, the solutions ψ, n, p, J_n and J_p are moderately varying functions.
- Zeroth-order layers do not occur at Ohmic contacts and insulating boundary segments.

Remark 5.8 It should be pointed out that these features reflect the device physics correctly: so-called depletion layers occur at pn-junctions and Schottky contacts, and inversion layers at semiconductor/oxide-interfaces. ▲

Singular perturbation analysis can also be used to investigate the **conditioning** of the semiconductor equations. A physical problem is **well-conditioned** (**ill-conditioned**) if small changes of the data cause small (large) changes of the solutions. The data are here the doping profile C, the recombination-generation rate R and the boundary conditions. The following statements on the conditioning of the stationary semiconductor equations (including appropriate boundary conditions) can be obtained if the system is sufficiently close to thermal equilibrium:

- Poisson's equation (5.33) is well-conditioned (with respect to ψ), at least for a moderate bias applied, independent of the singular perturbation parameter λ and the doping profile C.
- Both continuity equations (5.34) and (5.35) are well-conditioned (referring to n or p, respectively) independent of δ if every p- and n-domain has an Ohmic contact.
- If an n- or p-domain has no (Ohmic) contact, the continuity equation for the majority carrier concentration of this region is ill-conditioned. The errors can be amplified by a factor of the magnitude $O(\delta^{-4})$ and, therefore, domains without contacts can produce great numerical difficulties in computing carrier concentrations. Such domains are called **floating regions**.

Example 5.1 In case of a MOSFET, source and drain (both n- or both p-domains) always have Ohmic contacts. If the domain in which the channel is formed[44] has an Ohmic contact, too, the full system can be expected to be well-conditioned (sufficiently close to thermal equilibrium). However, if the "channel domain" is not contacted, i.e. if it is a floating region, the continuity equation corresponding to the majority carrier concentration is ill-conditioned. Such floating channel regions occur, for instance, in devices fabricated by silicon-on-insulator (SOI) technology (e.g. FinFETs). Since SOI is one of the standard technologies today, problematic floating regions occur quite often in device simulation. ▲

Remark 5.9 Normally, the effect of perturbations on the current densities J_n and J_p is less dramatic: small perturbations of the data cause only small perturbations of the current densities (at least under "moderate injection", see [82]), and numerical results for the important current densities and IV-characteristics can be quite accurate even if the perturbations of the carrier concentrations are large. ▲

[44] This happens in the bulk in case of a "conventional" MOSFET, see Section 5.3.1.3.

5.3.1.5 Meshing, Discretization and Linearization

While the discretization of the Poisson(-type) equation ((5.26), (5.30) or (5.33), respectively) is straightforward, the discretization of the continuity equations, which can be characterized as special diffusion-convection-reaction equations, is crucial for an efficient solution of the drift-diffusion system. In practice, the whole drift-diffusion system is discretized by the (mid-perpendicular) Scharfetter-Gummel box method (SG-BM) on boundary Delaunay meshes. This discretization method is briefly characterized in the following.

A triangulation $\mathcal{T}$ of a polygonally bounded domain Λ is a **Delaunay triangulation** if the interior of the circumcircle of each element T of the triangulation does not contain mesh vertices. Furthermore, a 2D Delaunay mesh[45] is **boundary Delaunay** (*box-method-conforming Delaunay*) if we have, for each boundary edge, $\alpha < \pi/2$ where α is the opposite angle in the element T this edge belongs to. The boundary requirements in 3D are formulated in terms of circumcircles for the boundary edges and faces in an analogous way.

The Delaunay mesh serves as a *primary* mesh from which a *secondary* (or *dual*) mesh of **Voronoi boxes** is derived. Given a Delaunay mesh with vertices x_k, the Voronoi box $\mathcal{V}_k$ associated to an x_k is bounded by the mid-perpendicular planes associated to each edge e_{kl} between vertices x_k and x_l. The aforementioned Delaunay conditions on the mesh and its boundary then guarantee that the (open domains) $\mathcal{V}_k$ do not overlap and are completely contained in the domain Λ.

The BM on a Delaunay mesh is nothing else than a finite volume approach (see [74], for instance). In the 2D case, it can also be interpreted as a disturbed FE method with piecewise linear trial functions on the primary grid and piecewise constant test functions on the boxes. In 3D, however, the BM and a standard FE discretization can exhibit drastically different properties as shown in [47]. The comparison of these discretizations presented there for diffusion simulation on 3D Delaunay meshes strongly advocates the use of the FV discretization since it is stable[46]. As a consequence, the solution does not contain any nonphysical negative concentrations.

The discretization of the continuity equations needs special care in order to gain stability. For this purpose, the so-called Scharfetter-Gummel discretization approach [77, 54] has been developed. The **Scharfetter-Gummel box method (SG-BM)** can be outlined as follows. We start with the assumption that, along mesh edges, the mobilities μ_n and μ_p are constant, and the electrostatic potential ψ behaves as a linear function. Approximations of J_n and J_p can then be obtained by solving a one-dimensional boundary value problem. This leads to an exponentially-fitted scheme for the current relations. The emerging approximations of the edge current densities are employed to obtain the final discretization of the continuity equations. For more details on the SG-BM, see [44], for instance. Inside the domain Ω_s, that is ignoring boundaries and interfaces, the resulting discretized system reads

$$(F_\psi, F_n, F_p)^T = 0 \tag{5.36}$$

[45] In the following, we only consider triangles or tetrahedrons, respectively. See also Remark 5.10.

[46] i.e. it fulfills a discrete maximum principle since the matrix corresponding to the BM-discretized stationary diffusion equation is a Stieltjes matrix, see Theorem 5.1 below.

with[47]

$$
\begin{aligned}
(F_\psi)_k &= \sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \epsilon_{k,l,T} \frac{s_{k,l,T}}{|x_k - x_l|} (\psi_k - \psi_l) \\
&\quad + \sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} V_{k,l,T} \left(n_k - p_k - C_k\right) = 0 \ , \qquad (5.37) \\
(F_n)_k &= \sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \mu_{n;k,l,T} \frac{s_{k,l,T}}{|x_k - x_l|} \left[B(\psi_k - \psi_l) n_k - B(\psi_l - \psi_k) n_l\right] \\
&\quad + \sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} V_{k,l,T} R_{k,l,T} = 0 \ , \qquad (5.38) \\
(F_p)_k &= \sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \mu_{p;k,l,T} \frac{s_{k,l,T}}{|x_k - x_l|} \left[B(\psi_l - \psi_k) p_k - B(\psi_k - \psi_l) p_l\right] \\
&\quad + \sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} V_{k,l,T} R_{k,l,T} = 0 \ , \qquad (5.39)
\end{aligned}
$$

where B denotes the Bernoulli function,

$$
B(0) := 1 \quad \text{and, for } x \neq 0, \quad B(x) := \frac{x}{\exp(x) - 1} \begin{cases} \in]0;1[& \text{for } x > 0 \ , \\ \approx \ -x + 1 & \text{for } x \leq 0 \ , \end{cases} \qquad (5.40)
$$

$\mathcal{T}_k$ the set of all elements T with vertex x_k, $l(\mathcal{T}_k)$ the set of vertices l of $\mathcal{T}_k$ connected to x_k by an edge, $V_{k,l,T}$ the sum of the volumes of the parts of $\mathcal{V}_k$ which belong to an element T having both x_k and x_l as vertices, and $s_{k,l,T}$ the volume of the facet $A_{k,l,T}$ which the Voronoi cells $\mathcal{V}_k$ and $\mathcal{V}_l$ share. For modifications in order to avoid negative $s_{k,l,T}$, see Remark 5.13. The values $\epsilon_{k,l,T}$, $\mu_{n;k,l,T}$, $\mu_{p;k,l,T}$ and $R_{k,l,T}$ are suitable element-edge approximations of the corresponding functions ϵ, μ_n, μ_p and R, respectively, on the edge e_{kl}.

Remark 5.10 In practice, mixed-element meshes are used as simulation meshes. In 2D, they consist of triangles and rectangles, in 3D, of prisms and pyramids with bases of three or four sides. If the simulation domain Ω is split by inner interfaces (see Section 5.3.1.3) into several domains, the mesh for each of these domains has to be boundary Delaunay. While the construction of boundary Delaunay meshes can automatically be performed in 2D, it is problematic in 3D. ▲

Main advantages of the SG-BM are that the discretization is stable[48] and that it inherits the so-called *local dissipativity*, a physically important property, of the continuous system (see [28]) in case of $R = R_{SRH,AU}$, for instance. However, the stability is paid for by a loss of convergence order[49]. No higher-order equivalents of the SG-BM are known.

[47] formulated for the system (5.26)-(5.28), analogous for (5.30)-(5.32) and (5.33)-(5.35).

[48] since, linearized, it fulfills a discrete maximum principle, i.e. the M-matrix property, see Theorem 5.1 below.

[49] Results in [57], for instance, indicate a convergence order of only $ch^{1/2}$ for a mesh-dependent norm and an ϵ-dependent constant c.

This highly nonlinear discrete system is linearized by a (modified) Newton(-Raphson) method[50]. Necessary for the Newton method is the solution of linear systems where the arising matrices A correspond to Jacobians of the above nonlinear system (5.36). We call these Jacobians A **drift-diffusion matrices**. In the following, we always assume the physical unknowns of the linear systems to be solved ordered as $u_1 := \psi, u_2 := n, u_3 := p$. The principal form of A, corresponding to this ordering, is then shown in Table 5.10. We use the following abbreviations:

$$K_{n,k,l} := \mu_{n;k,l,T} \frac{s_{k,l,T}}{|x_k - x_l|} \quad , \quad K_{p,k,l} := \mu_{p;k,l,T} \frac{s_{k,l,T}}{|x_k - x_l|} \quad .$$

Since the discretized and linearized systems inherit the layer behavior and conditioning of the

	$\frac{\partial}{\partial(\psi_k)}$	$\frac{\partial}{\partial(n_k)}$	$\frac{\partial}{\partial(p_k)}$
	$\frac{\partial}{\partial(\psi_l)}\ (l \neq k)$	$\frac{\partial}{\partial(n_l)}\ (l \neq k)$	$\frac{\partial}{\partial(p_l)}\ (l \neq k)$
$(F_\psi)_k$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \epsilon_{k,l,T} \frac{s_{k,l,T}}{\lvert x_k - x_l \rvert}$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} V_{k,l,T}$	$-\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} V_{k,l,T}$
	$-\epsilon_{k,l,T} \frac{s_{k,l,T}}{\lvert x_k - x_l \rvert}$		
$(F_n)_k$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \Big[K_{n,k,l} \big[n_k B'(\psi_k - \psi_l) + n_l B'(\psi_l - \psi_k) \big] + V_{k,l,T} \frac{\partial R_k}{\partial(\psi_k)} \Big]$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \Big[V_{k,l,T} \frac{\partial R_k}{\partial(n_k)} + K_{n,k,l} B(\psi_k - \psi_l) \Big]$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} V_{k,l,T} \frac{\partial R_k}{\partial(p_k)}$
	$K_{n,k,l} \big[-n_k B'(\psi_k - \psi_l) - n_l B'(\psi_l - \psi_k) \big] + V_{k,l,T} \frac{\partial R_k}{\partial(\psi_l)}$	$-K_{n,k,l} B(\psi_k - \psi_l) + V_{k,l,T} \frac{\partial R_k}{\partial(n_l)}$	$V_{k,l,T} \frac{\partial R_k}{\partial(p_l)}$
$(F_p)_k$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \Big[K_{p,k,l} \big[-p_k B'(\psi_l - \psi_k) - p_l B'(\psi_k - \psi_l) \big] + V_{k,l,T} \frac{\partial R_k}{\partial(\psi_k)} \Big]$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} V_{k,l,T} \frac{\partial R_k}{\partial(n_k)}$	$\sum_{\mathcal{T}_k} \sum_{l(\mathcal{T}_k)} \Big[V_{k,l,T} \frac{\partial R_k}{\partial(p_k)} + K_{p,k,l} B(\psi_l - \psi_k) \Big]$
	$K_{p,k,l} \big[p_k B'(\psi_l - \psi_k) + p_l B'(\psi_k - \psi_l) \big] + V_{k,l,T} \frac{\partial R_k}{\partial(\psi_l)}$	$V_{k,l,T} \frac{\partial R_k}{\partial(n_l)}$	$-K_{p,k,l} B(\psi_k - \psi_l) + V_{k,l,T} \frac{\partial R_k}{\partial(p_l)}$

Table 5.10: Jacobian of the discrete drift-diffusion system (5.36). Replace each R_k by $R_{k,l,T}$.

original equations, we have to expect layer behavior near pn-junctions, Schottky contacts and semiconductor/oxide interfaces and ill-conditioned continuity equations in floating regions. As shown in [2], even for simple diode examples which do not face critical regions, the condition numbers are quite high. Moreover, because of the fact that the original system (5.26)-(5.28) is usually scaled "only" by DeMari factors, ψ, n, p and their discrete analogs

[50] usually with a damping strategy, see [92], for instance. See also Remark 5.11 for an alternative method.

are still very different in magnitude. This leads, in particular, to large differences in the magnitude of the matrix entries. As the diagonal entries can vary several orders of magnitude among each other, so can the off-diagonal entries, both among each other and compared to the respective diagonal entries. Due to all these reasons, the arising drift-diffusion matrices are very ill-conditioned and often nearly singular even if the underlying problem is far from a possibly-existing bifurcation point. As a consequence, this leads to great difficulties in solving the matrix equations efficiently.

In commercial device simulators, the sequence of drift-diffusion matrix equations is solved by iterative[51] one-level methods, usually by an ILU- or ILUT-preconditioned CGS or BiCGstab. More precisely, some modified ILU(0)-approach is used in TAURUS [92] by Synopsys Inc. However, due to the above-mentioned properties of the matrices, iterative one-level solvers often exhibit an unsatisfactory performance.

5.3.2 Efficient Solution of the Linear Systems

In this section, it is demonstrated that one of SAMG's point-based AMG approaches with a norm-based primary matrix, accelerated by BiCGstab, works more robustly and often more efficiently for large drift-diffusion matrices than the standard one-level solvers commonly used in device simulation.

We start with deriving more numerical properties of the arising matrices. Based on the statements obtained in Sections 5.3.1.5 and 5.3.2.1, we discuss in Section 5.3.2.2 why VAMG and UAMG approaches do not work for drift-diffusion matrices whereas certain PAMG approaches are reasonable candidates. Section 5.3.2.2 also explains the concrete PAMG approach of our framework which has been turned out to be a suitable preconditioner. In Section 5.3.2.3, the exemplary devices and the concrete TAURUS simulation runs for the numerical tests are described. Numerical results of these simulation runs will be presented in Section 5.3.2.4.

Remark 5.11 By now, linear multigrid methods have been developed only for solving the three individual partial differential equations arising during a Gummel(-type) iteration. In Gummel's approach, an equation for ψ is solved first, then an equation for n and then for p. This way, the more expensive Newton approach is replaced by a Gauss-Seidel-type iteration of solving three single PDEs. The most difficult part there is the solution of discretized and linearized continuity equations. For example, in [23], a geometric multigrid method for this type of equation was investigated and successfully applied to some examples on structured grids. In contrast to this, in this thesis, we are interested in the solution of the matrix equations arising from the fully coupled approach. This approach is typically used in modern (commercial) device simulators because it is often more favorable, in particular near equilibrium, than a Gummel(-type) iteration. ▲

Remark 5.12 In particular until the early nineties, the application of nonlinear geometric multigrid methods (full approximation schemes (FAS)) was investigated as a further approach

[51] if the problem size exceeds the abilities of direct solvers.

to solve drift-diffusion systems (see the references given in [23]). Such approaches are, however, difficult to apply (if possible at all) to the more sophisticated models and unstructured grids used in modern commercial device simulators. ▲

5.3.2.1 Numerical Properties of the Matrices

According to the different modelling situations in the two parts Ω_s and Ω_o of Ω, the drift-diffusion matrices A consist of two parts, A_s and A_o, which are very different by nature. On the one hand, the part A_o of A which corresponds to Laplace's equation (on Ω_o) does not pose problems. Since the submatrix $A_{[1,1]}$ describing the ψ-to-ψ couplings is a Stieltjes matrix if the mesh is boundary Delaunay (see Theorem 5.1 below), and there are only a few couplings to n and p, namely across the interface to Ω_s, (V)AMG is appropriate for A_o.

On the other hand, the part A_s of A which corresponds to the coupled PDE system posed on Ω_s inherits the very tight coupling of the physical unknowns, in particular reflected by large entries in the submatrices $A_{[m,n]}$ ($m \neq n$). Properties of these submatrices which are important with respect to AMG are investigated in more detail now.

We know from Section 5.3.1.5 that the entries of A_s are given by Table 5.10 together with the discrete and linearized analogs of the boundary and interface conditions. For the following considerations, we restrict ourselves to the interior of Ω_s - that means to a discussion of matrix entries as presented in Table 5.10 - together with Dirichlet boundary conditions on $\partial\Omega_s$. Defining a matrix B being a **±M-matrix** if either B or $-B$ is an M-matrix, the following theorem can be proved.

Theorem 5.1 *For linearized SG-BM discretized drift-diffusion systems with $R = R_{SRH,AU}$, the submatrices $A_{[n,m]}$ of A_s are either diagonal matrices or ±M-matrices if the underlying mesh is boundary Delaunay and of acute type*[52]*. To be more specific, independent from the concrete R, $A_{[1,1]}$ is even a weakly diagonally dominant Stieltjes matrix, and $A_{[1,2]}$ and $A_{[1,3]}$ are diagonal matrices. For $R = R_{SRH,AU}$, $A_{[2,3]}$ and $A_{[3,2]}$ are diagonal matrices, and $A_{[2,1]}$, $A_{[2,2]}$, $A_{[3,1]}$, and $A_{[3,3]}$ are weakly diagonally dominant ±M-matrices.*

Proof. The statements on the diagonal-blocks $A_{[n,n]}$ can be found in [54], for instance, which also points to related literature. We add the proof for the remaining $A_{[m,n]}$ here. Obviously, $A_{[1,2]}$ and $A_{[1,3]}$ are always diagonal. Assuming $R = R_{SRH,AU}$, we obtain

$$\frac{\partial R_k}{\partial(n_l)} = \frac{\partial R_k}{\partial(p_l)} = 0 \quad \text{for } l \neq k \quad \text{and} \quad \frac{\partial R_k}{\partial(\psi_l)} = 0 \quad \text{for all } k, l \ .$$

Hence, all derivatives of R vanish for $l \neq k$ in case of $R = R_{SRH,AU}$. Obviously, $A_{[2,3]}$ and $A_{[3,2]}$ are diagonal then. For all x, the Bernoulli function B fulfills

$$B(x) > 0\,, \quad B'(0) = -\frac{1}{2} \quad \text{and, for } x \neq 0\,, \quad B'(x) = \frac{\exp(x) - 1 - x\exp x}{(\exp(x) - 1)^2} < 0 \ .$$

[52] We call a simplicial mesh of **acute type** if all interior angles of all triangles or, respectively, all interior angles between faces of tetrahedrons are not larger than $\pi/2$ (i.e. non-obtuse).

If the mesh is of acute type, all $K_{n,k,l}$ and $K_{p,k,l}$ are nonnegative. Due to all these facts, we obtain the following relations inside Ω_s

$$\frac{\partial(F_n)_k}{\partial(\psi_k)} < 0 \ , \quad \frac{\partial(F_n)_k}{\partial(\psi_l)} = \frac{\partial(F_n)_l}{\partial(\psi_k)} > 0 \quad (j \neq i) \ , \quad \frac{\partial(F_n)_k}{\partial(\psi_k)} = -\sum_{l \neq k} \frac{\partial(F_n)_k}{\partial\psi_l} \ , \tag{5.41}$$

$$\frac{\partial(F_p)_k}{\partial(\psi_k)} > 0 \ , \quad \frac{\partial(F_p)_k}{\partial(\psi_l)} = \frac{\partial(F_p)_l}{\partial(\psi_k)} < 0 \quad (j \neq i) \ , \quad \frac{\partial(F_p)_k}{\partial(\psi_k)} = -\sum_{l \neq k} \frac{\partial(F_p)_k}{\partial\psi_l} \ . \tag{5.42}$$

That $A_{[2,1]}$ and $A_{[3,1]}$ are $\pm$M-matrices follows from Remark 2.14 in Section 2.4.4. ■

Remark 5.13 In general, if the grid generator produces obtuse angles ($\alpha_i > \pi/2$), corresponding $s_{k,l,T}$ could be negative and the blocks $A_{[n,m]}$ of A will not be $\pm$M-matrices in the original SG-BM. In order to obtain $\pm$M-matrices again, the *compensated box method* is then used in practice, as described in [79], for instance. ▲

In many cases, either n or p strongly varies in the simulation domain and clearly dominates the other two physical unknowns. In particular, A_s is usually dominated by couplings *to* the potential ψ. To be more specific, either the submatrix $A_{[2,1]}$ or $A_{[3,1]}$ (of A_s), depending on the majority carrier concentration, contains a significant part of the largest couplings, measured by absolute value. This can be seen as follows. For $\psi_k \approx \psi_l$, $B(\psi_k - \psi_l)$ is $O(1)$, and we have $B'(\psi_k - \psi_l) \approx -0.5$ (cf. Fig. 5.12) so that the following estimate emerges:

$$n_k B'(\psi_k - \psi_l) + n_l B'(\psi_l - \psi_k) \lesssim -\frac{n_k + n_l}{2} < 0 \ .$$

Note that, very roughly,

$$V_{k,l,T} = O(|x_k - x_l|^3) \ , \quad K_{*,k,l} = \mu_{*;k,l,T} O(|x_k - x_l|) \text{ with } * \in \{\psi, n, p\}.$$

Without restriction of generality, we assume now that the concentration n clearly dominates p, i.e. n is several orders of magnitude larger than p. We then have

$$V_{k,l,T} \Big| \frac{\partial R_{SRH,AU}}{\partial(n)} \Big| \ll K_{n,k,l} \frac{n_k + n_l}{2}$$

as shown in Fig. 5.13. The same is true for $V_{k,l,T}|\partial R_{SRH,AU}/\partial(p)|$. The dominance becomes smaller, the more similar n and p are in size.

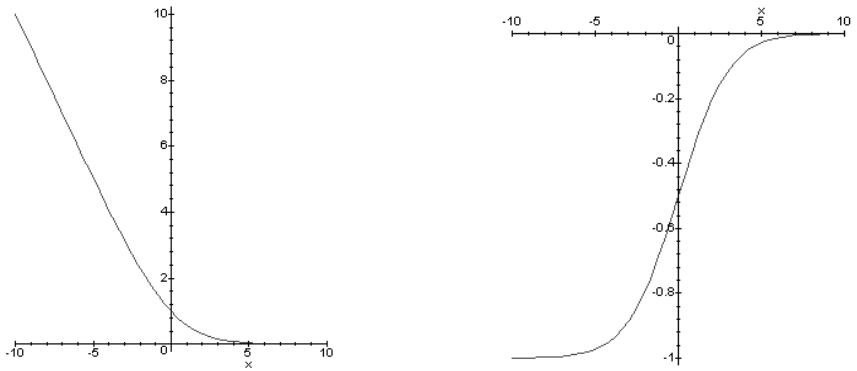
If we now compare the matrix entries of A_s, as listed in Table 5.10, we obtain the following result: The matrix $A_{[2,1]}$, reflecting the couplings of n to ψ, clearly dominates the other $A_{[m,n]}$ in the sense that most often

$$\frac{\partial(F_n)_k}{\partial(\psi_l)} \gg \max\Big\{ \frac{\partial(F_n)_k}{\partial(n_l)} \, , \frac{\partial(F_n)_k}{\partial(p_l)} \, , \frac{\partial(F_\psi)_i}{\partial(*_l)} \, , \frac{\partial(F_p)_k}{\partial(*_l)} \Big\}$$

with $* \in \{\psi, n, p\}$ holds. This means that the entry a_{ij} in a $A_{(k,l)}$ which couples n to ψ is most often the largest. The argumentation for clearly dominating p is analogous. These arguments qualitatively hold also for the discretized and linearized scaled systems (5.30)-(5.32) and (5.33)-(5.35), respectively, as can exemplary be seen in Fig. 5.18 depicting a

drift-diffusion matrix (DeMari scaling) arising in the simulation run for the EEPROM test case described in Section 5.3.2.3 below. For all examples tested so far, for either $\star = n$ or $\star = p$ the following statement holds: in around 80% of the cases, the entry a_{ij} in $A_{(k,l)}$ which couples unknown $\star$ to ψ is largest. That is, either $A_{[2,1]}$ or $A_{[3,1]}$ clearly dominates the other $A_{[m,n]}$.

Several consequences arise from this "under-representation" of the diagonal blocks $A_{[n,n]}$. Obviously, many rows of the drift-diffusion matrices strongly violate diagonal dominance. The discrete PDE system is strongly coupled in the sense that $A^{-1}A_u$ and $A_u^{-1}A$ are far from being the identity matrix [53]. Last but not least, for the reasons mentioned in Section 5.3.1.5, we often face very ill-conditioned or even nearly singular matrices. All these reasons complicate an accurate and efficient numerical solution in general.



(a) (b)

Figure 5.12: Plot of (a) $B(x)$ and (b) $B'(x)$ for $x \in [-10, 10]$.

5.3.2.2 The AMG Approach Employed

The properties of A discussed above prevent VAMG and UAMG from being reasonably applicable here. VAMG- and UAMG-preconditioned approaches usually diverge and are therefore not discussed in the remainder. Except of regions with a strong layer behavior, which is only present in the drift-diffusion systems, the slightly anisotropic DD models (with moderate (λ, c)) and the (DeMari-scaled) drift-diffusion matrices are similar to some extent. Without restriction of generality, assuming n to be the dominating species, both exhibit the same composition of $\pm$M-submatrices and diagonal submatrices $A_{[m,n]}$ [54]. For both, $A_{[2,1]}$ is the dominating submatrix in the sense discussed above, and $A_{[2,1]}$ can exhibit a slight anisotropy[55]

[53] i.e. ρ_u is large. Note that, strictly speaking, ρ_u defined in (3.65) is a measure for symmetric $A(> 0)$ only, but a large ρ_u should in general be a good hint that the discrete PDE system is too strongly coupled for UAMG.

[54] we neglect $A_{[2,3]}$ and $A_{[3,2]}$ since here their entries are zero or very small compared to the others for large n.

[55] B' is for regions with moderately varying ψ a moderately varying function. n and p are moderately varying functions outside the junction-, Schottky contact- and semiconductor/oxide-interface-layers, see Section 5.3.1.4.

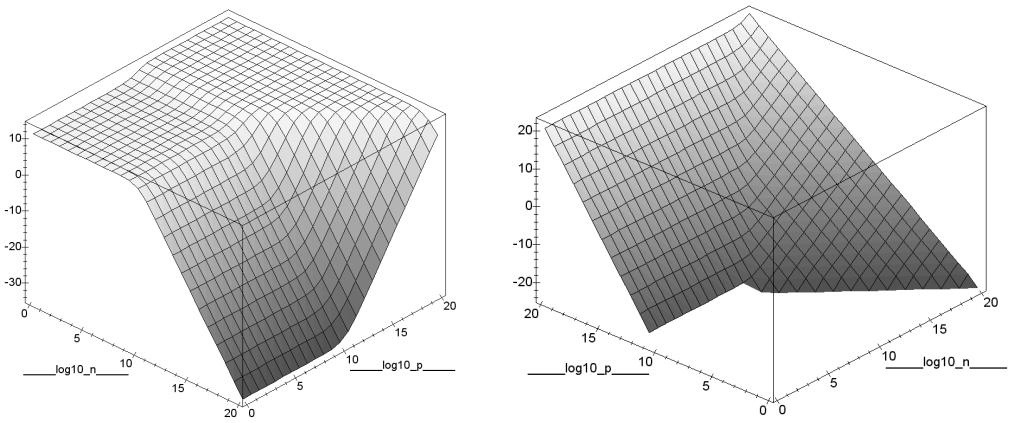


Figure 5.13: Plot of (a) $\log_{10}(\partial R_{SRH}/\partial(n))$ and (b) $\log_{10}(\partial R_{AU}/\partial(n))$.

also in case of the drift-diffusion matrices. Since certain PAMG approaches have been seen to yield efficient preconditioners for the anisotropic DD models, it seems promising to investigate their application to drift-diffusion matrices as well.

Indeed, in accordance to the investigations made in Section 3.4.1.2 and Examples 3.2, 3.5 and 3.12 for the DD models, it has turned out that PAMG yields a very robust and efficient preconditioner if the following components are chosen:

- Smoothing: for some simple matrices which are not too ill-conditioned and which do not violate diagonal dominance too strongly, BGS can be used for smoothing. Figure 5.14 depicts algebraically smooth error produces by BGS for such a "simple" device[56]. In general, however, an often much stronger, but also more expensive ILU smoother should be employed. For all numerical tests performed so far, ILU(0) has shown a robust behavior. This is in contrast to ILUT which does not work robustly as a smoother here. If not stated otherwise, results are presented for ILU(0) smoothing.
- Coarsening based on a primary matrix based on norms (3.73). A1-coarsening is used on the first level, standard coarsening else. For all drift-diffusion systems tested so far, the performance of the resulting approach was neither sensitively influenced by the concrete choice of the norm nor the concrete choice of p_{kk} (see Section 3.4.2, Remark 3.24).
- An SU-interpolation with weights being based on the entries of $\boldsymbol{P}$. In some cases, an MU-interpolation with scaling of the weights (see Section 4.3.1.5) has been found to yield a similar performance.
- One step of Jacobi-relaxation of interpolation is applied to the second-to-first-level interpolation operator since this variant has turned out to improve the robustness and efficiency of the respective PAMG approach considerably.

[56] the only one with a two-dimensional grid here.

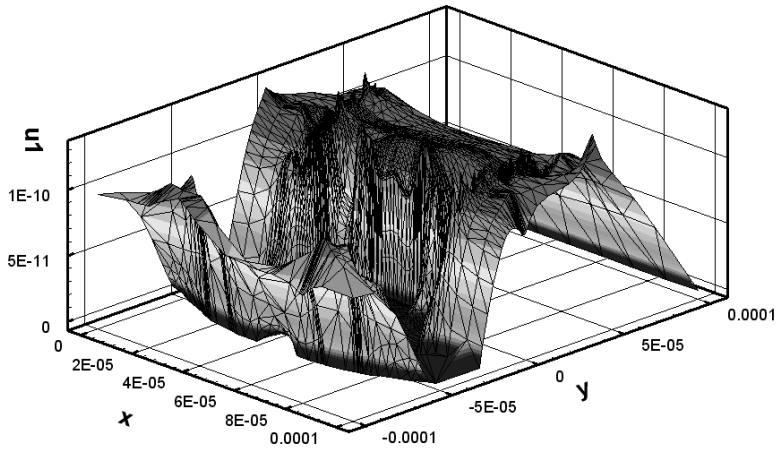


Figure 5.14: STI example: error after 10 BGS smoothing steps.

- As discussed in Section 4.1.1.1, rows of A with zero diagonal entries have to be treated in a special way in order to avoid possible problems with smoothing and nonpositive diagonals on coarser levels and to avoid "disturbed" coarsenings. The last row of the matrices A arising in the last part of the simulation for the EEPROM example (see next section), has a zero diagonal entry. Here, it has proved to be sufficient simply to force the corresponding variable to stay on the finest level (FF-variables) and thus to completely exclude it from the coarsening process.
- For many matrices, some nonpositive diagonal entries occur on coarser levels. They are handled as described in Appendix A.1.2.2.
- Accelerator: BiCGstab.

Remark 5.14 We have also tried to use one of the unknown-matrices $A_{[n,n]}$ as a primary matrix. $\boldsymbol{P} = A_{[1,1]}$ works in some cases, but is not as robust as a norm-based $\boldsymbol{P}$. ▲

Drift-diffusion matrices provide practically important examples where accelerated AMG approaches with BGS often diverge but ILU(0) helps PAMG to yield a robust preconditioner. On the first sight, this might be surprising since BGS works for the DD models (see Section 4.6) whereas ILU(0) often diverges there. However, as discussed in Sections 3.4.1.2 and Section 4.6, a more closer look reveals that ILU(0) smoothes the error everywhere except of a small area and, for the DD models with moderate (λ, c), ILU(0)-PAMG(ns,·,SU,P)-BiCGstab converges (however, the same approach with BGS smoothing works better for the DD models). In case of the drift-diffusion matrices, both BGS and ILU(0) diverge if used stand-alone. Also in combination with AMG approaches they diverge. However, if ILU(0) as a smoother for the PAMG approach described above is accelerated by BiCGstab, for instance, convergence is achieved, and the approach is quite efficient.

Remark 5.15 Block variants of ILU(0) or ILUT approaches can be expected to yield even better smoothers than the standard, variable-based ILU(0) employed here. Block variants will be one topic of future research. ▲

Remark 5.16 Recently, permutations of drift-diffusion matrices A with the aim to enhance the preconditioning properties of ILU(0) and SPAI variants have been considered in [78]. Presented numerical results indicate that this helps ILU(0) and makes it more stable. The SPAI variants with or without permutations of A, however, have been found to be much more expensive and less stable preconditioners than ILU(0) for this application class [57]. Instead of permutations as a preprocessing step, the investigation of appropriate pivoting strategies incorporated into the ILU(0) *smoother* used in our AMG approach will also be a topic of future research. A first step has already been taken by incorporating (M)ILUTP smoothing into SAMG (see Section 4.4). ▲

5.3.2.3 Description of the Test Cases

We have tested examples from several classes of industrially relevant semiconductor devices. To be more specific, we have considered

- a shallow trench isolated MOSFET[58] (STI),
- an electrically erasable programmable read-only memory cell (EEPROM),
- a metal-semiconductor transistor[59] (MESFET),
- a FinFET[60],

General information about most of these and other types of semiconductor devices can be found in [93], for instance. A description and analysis of a (particular) FinFET is given in [38], for example. Table 5.11 shows details on the concrete test cases and dimensions of the arising matrix problems. Layouts, doping profiles and grids of the STI and the FinFET are shown in Figs. 3.7 and 5.19.

In device simulators such a TAURUS, a simulation series for a given device consists of many individual simulations of drift-diffusion systems which differ, for example, in their respective boundary conditions. More precisely, each simulation run starts with the zero bias step, a step in which all voltages are set to zero. Afterwards, several *bias ramps* are applied to the device. For instance, Figs. 5.15 and 5.16 show the sequence of bias ramps applied in case of the EEPROM and the FinFET, respectively. Exemplarily, we briefly explain the bias ramps for the FinFET: In the first 21 simulation steps (the first ramp), the gate voltage is gradually increased from 0 to 1V, keeping the drain voltage fixed at 0.05V. During the next 10 bias steps (the second ramp), the gate voltage is fixed at 1V, and the drain voltage is increased step by step to a value of 1V. For each individual simulation step, i.e. for each bias applied, a Newton process is employed to solve the discretized problem, and, within

[57] Recently, AMG approaches which incorporate SPAI variants as smoothers and/or in various steps of the setup phase have been investigated in [13]. So far, it has not been investigated whether SPAI variants are, for instance, good smoothers also for drift-diffusion matrices.

[58] MOSFET = metal oxide semiconductor field effect transistor.

[59] Both source and drain are Schottky contacts.

[60] FinFET = a double-gate MOSFET structure in which a thin, fin-shaped body is straddled by the gate forming two self-aligned channels that run along the sides of the fin.

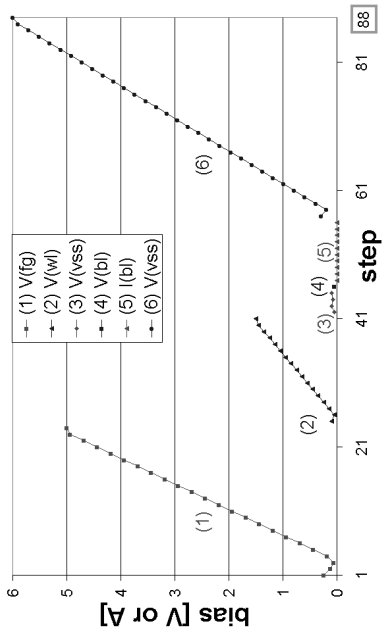


Figure 5.15: EEPROM example: bias steps.

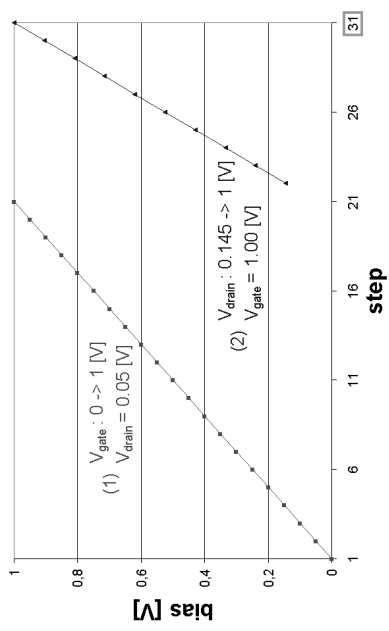


Figure 5.16: FinFET example: bias steps.

each Newton step, a few matrix solves are necessary. Therefore, during a whole simulation run, several hundred linear systems have to be solved. In (commercial) simulators such as TAURUS, sophisticated control mechanisms are integrated to detect and "repair" possibly occurring difficulties during the linear and nonlinear iterations. In the worst case, this means a bias step rejection and step size reduction.

In the following, we will present detailed results for two exemplary cases, namely the EEPROM and the FinFET. The FinFET case represents the simulation of a modern device on a moderately large grid. Since a FinFET is fabricated on an SOI wafer, numerical difficulties arise due to the occurring floating region (see also Example 5.1). The EEPROM example, which is rather small in terms of variables, was chosen because it exhibits an additional difficulty: for the sixth bias ramp, the system is extended by one equation (an algebraic condition) leading to a row with a zero diagonal. Rows with zero diagonals are likely to produce problems for all iterative solvers. During the AMG setup phase, this exceptional row is treated separately as described in Section 5.3.2.2 above. In addition, remarks on the numerical performance will be made for the small STI example and the middle-sized MESFET, which is very ill-conditioned, in particular due to its Schottky contacts.

Example	dim	n_s	n_o	n_p	n_v	n_A
STI	2D	1	7	5 516	9 212	125 516
EEPROM	3D	1	9	10 493	15 415	310 361
MESFET	3D	2	3	14 720	28 026	476 804
FinFET	3D	2	5	69 092	97 530	1 443 940

Table 5.11: Details on the four examples. dim = spatial dimension, n_s = number of material layers in Ω_s, n_o = number of material layers in Ω_o.

5.3.2.4 Numerical Results

Generally, to demonstrate the performance of SAMG for a given device, it is not sufficient to look at its performance in solving just a few selected linear systems arising as part of a whole simulation. In fact, as explained in Section 5.3.2.3, hundreds of linear systems have to be solved during a full simulation series, and the properties of the matrices change substantially depending on the bias step and the progress made in the Newton process. Consequently, to obtain a clear picture of the benefit of SAMG, one has to consider full simulation series.

In order to demonstrate the robustness and efficiency of the PAMG preconditioner chosen, we have created an interface in TAURUS to the SAMG library, performed tests with the mentioned PAMG-BiCGstab approach and compared the results with the results of the corresponding runs with TAURUS's default iterative solver (called "TAURUS solver" in the following), an ILU-CGS method. In addition, to demonstrate the effects of the coarse-level corrections, we also compare the performance of PAMG-BiCGstab with that of the corresponding one-level method, i.e. ILU(0)-BiCGstab.

For all examples tested so far, it can be observed that the TAURUS solver does not always fulfill the prescribed convergence criterion, i.e. a relative residual reduction ϵ (2.62) of at least 1e-3, measured in the Euclidean norm, within a maximum number of iterations. This is depicted exemplarily for the EEPROM and the FinFET in Figs. 5.20(a) and 5.21(b). For both the STI and MESFET, the TAURUS solver behaves similarly. In the figures, $||r_0||$ denotes the Euclidean norm of the first and $||r_e||$ the Euclidean norm of the last residual. A value $||r_e||/||r_0||$ above the lower line at 1e-3 then means a violation of the criterion, a value above the upper line at 1e0 means divergence of the TAURUS solver for the current matrix. Especially in the EEPROM case, the Euclidean norm of the last residual is often more than 10^5 times larger than the first residual. Note for all graphs that the matrices arising during a full simulation series are always numbered consecutively.

In contrast to this, PAMG-BiCGstab shows a stable and fast convergence behavior for the STI, the EEPROM and the FinFET. The convergence criterion is fulfilled for all SAMG runs, and hence, instead of $||r_e||/||r_0||$, average residual reduction factors (ARFs) are depicted in Figs. 5.20 and 5.21. The performance of PAMG-BiCGstab for the STI is similar to the FinFET case. The ARFs are usually lower than 0.5 and often much better, and less matrix solves were necessary during the Newton steps (see Table 5.12), especially in case of the larger example, i.e. the FinFET.

A comparison of PAMG-BiCGstab with the corresponding one-level solver, ILU(0)-

BiCGstab, demonstrates that this drastic improvement of robustness and convergence speed is to a large extent caused by employing a hierarchy. In contrast to PAMG-BiCGstab, ILU(0)-BiCGstab exhibits ARFs which are close to or sometimes even larger than 1 (divergence). Additionally, ILU(0)-BiCGstab needs (much) more matrix solves, even more than ILU-CGS (see Table 5.12).

Remark 5.17 It should be noted that, in case of the MESFET, the worst case in terms of conditioning here, PAMG-BiCGstab converges but considerably worse than for the other three examples. In fact, only for a part of the matrices, PAMG-BiCGstab performs better than ILU(0)-BiCGstab so that employing a hierarchy only partly helps here. Interestingly, ILU(0)-BiCGstab performs much better than the TAURUS solver here. A main reason for the bad performance of the TAURUS solver and the reduced efficiency of PAMG-BiCGstab seems to be the "second half" of the simulation: at a certain point in the bias ramping for the concrete simulation run, both the TAURUS solver and PAMG-BiCGstab have problems in solving the concrete nonlinear system. The TAURUS solver faces several step rejections, whereas in case of the PAMG-BiCGstab just a temporary switching to ILU(0)-BiCGstab helps to overcome the troubles. The development of better "intelligent" solver-switching strategies will be subject to future research. ▲

example	approach	ϵ	♯ matrices	total	SAMG	c_{prec}
STI	PAMG-BiCGstab	1e-6	260	0.97	0.75	0.93 [α]
	TAURUS solver	1e-3	260	0.32		—
EEPROM	PAMG-BiCGstab	1e-6	520	3.81	2.19	[1.41,1.44]
	TAURUS solver	1e-3	538	2.38		—
	ILU(0)-BiCGstab	1e-6	560	4.31	2.56	1
	PAMG-BiCGstab	1e-3	578	3.80	2.02	[1.41,1.44]
	ILU(0)-BiCGstab	1e-3	620	4.00	2.06	1
FinFET	PAMG-BiCGstab	1e-3	100	4.27	3.25	[1.66,1.71]
	TAURUS solver	1e-3	157	4.46		—
	ILU(0)-BiCGstab	1e-3	216	11.49	9.15	1

[α] with BGS smoothing. For ILU smoothing, c_{prec}=1.41.

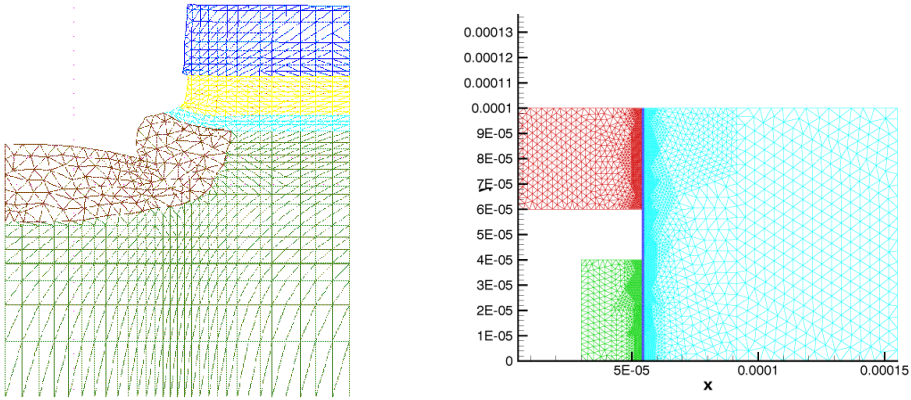
Table 5.12: Timings, number of necessary matrix solves and c_{prec} for the drift-diffusion simulations. ϵ denotes the residual reduction demanded. "total" is the total wall-clock time in hours needed for the whole simulation run. "SAMG" is the part of "total time" which SAMG needed to solve the matrices.

Table 5.12 also shows timings and c_{prec} for full simulation runs, including meshing and assembling of the matrices. It should be noted that the test character of TAURUS' interface to SAMG leads to extra overhead for the transfer of matrix data to SAMG. Whereas for the smallest example, the STI, the TAURUS solver (and also ILU(0)-BiCGstab) is much faster than PAMG-BiCGstab and for the "medium-sized" EEPROM example the TAURUS solver is considerably faster than PAMG-BiCGstab yet, the effort for employing the more

robust PAMG approach is paid off for the largest example here, the FinFET. In all cases, the preconditioner's complexity c_{prec} of PAMG is reasonable compared to ILU(0).

Remark 5.18 Recently, the ILU(0) method built into older versions of SAMG, including the one all tests for the drift-diffusion systems have been performed with, has been considerably speeded up, in particular for matrices containing more than ten entries per row. Only one additional vector of length n_v is needed in the faster variant. Based on timings conducted for single drift-diffusion matrices, it can be expected that due to the new ILU(0) the above timings for PAMG-BiCGstab would be reduced by a factor of 1.10 to 1.27. ▲

One should point out that the test cases are still rather small - and too small to demonstrate "real" advantages of PAMG over one-level preconditioners in terms of computational speed. However, since PAMG-BiCGstab clearly shows a robust behavior and considerably faster performance with increasing problem size, it can be expected that for even larger problems than the ones presented here the observed trends will be continued. The results clearly indicate that SAMG is often capable of solving the matrix equations more robustly, and we can expect that PAMG-BiCGstab clearly outperforms the one-level solvers which are commonly used in commercial device simulators in case of large(r) problem sizes.



(a) (b)

Figure 5.17: The grid structures for (a) the SILO2 example (green=silicon, yellow & brown=oxide, blue & cyan=nitride). (b) the DEPO2 problem (green=nitride, red=oxide, blue=polysilicon, cyan=silicon).

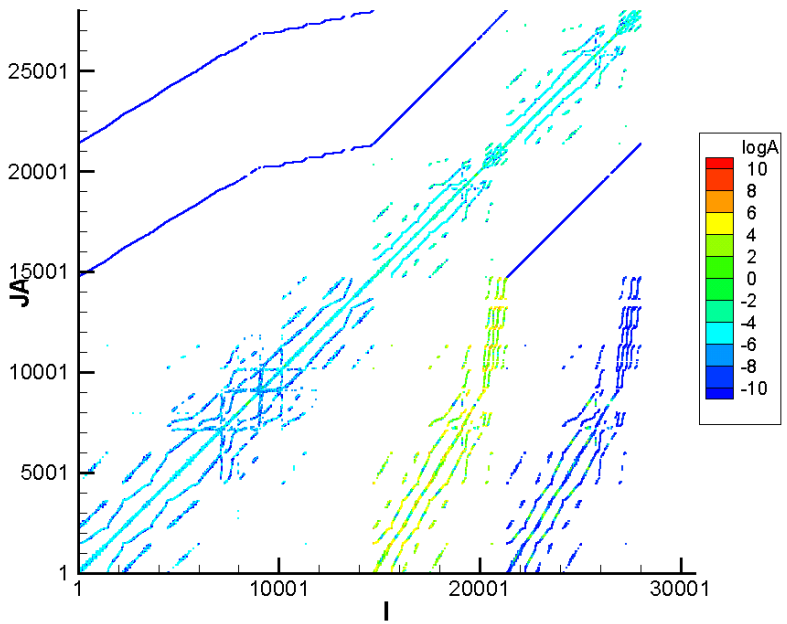
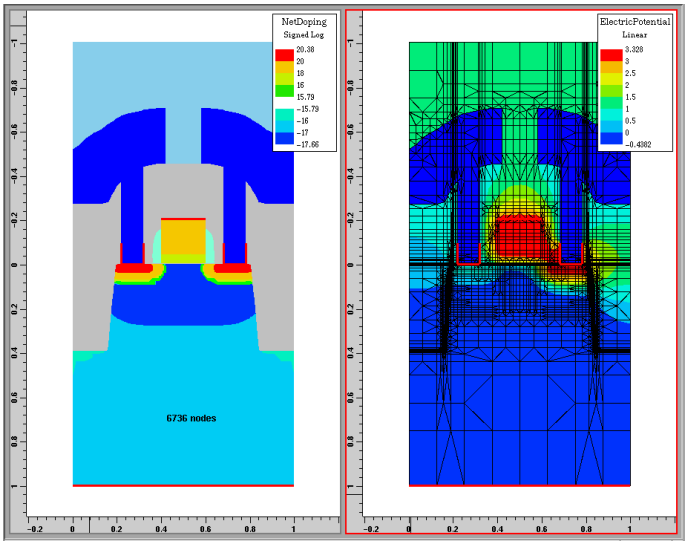
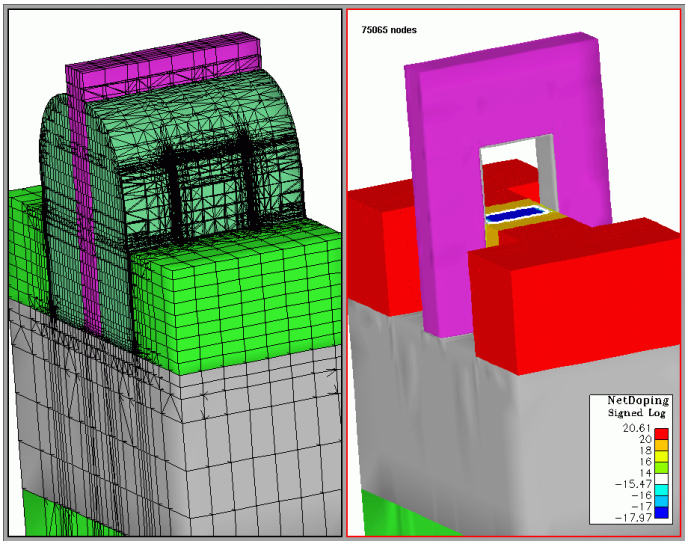


Figure 5.18: Plot of $\log_{10}|a_{ij}|$ of the entries a_{ij} of a typical matrix arising in the simulation of the EEPROM example. The variables are sorted unknown-wise here to make a comparison of the $A_{[m,n]}$ possible.



(a)



(b)

Figure 5.19: Doping profile of the wafer, layout and grid for two of the exemplary devices (see Table 5.11): (a) the STI, (b) the FinFET. Courtesy of Synopsys Inc.

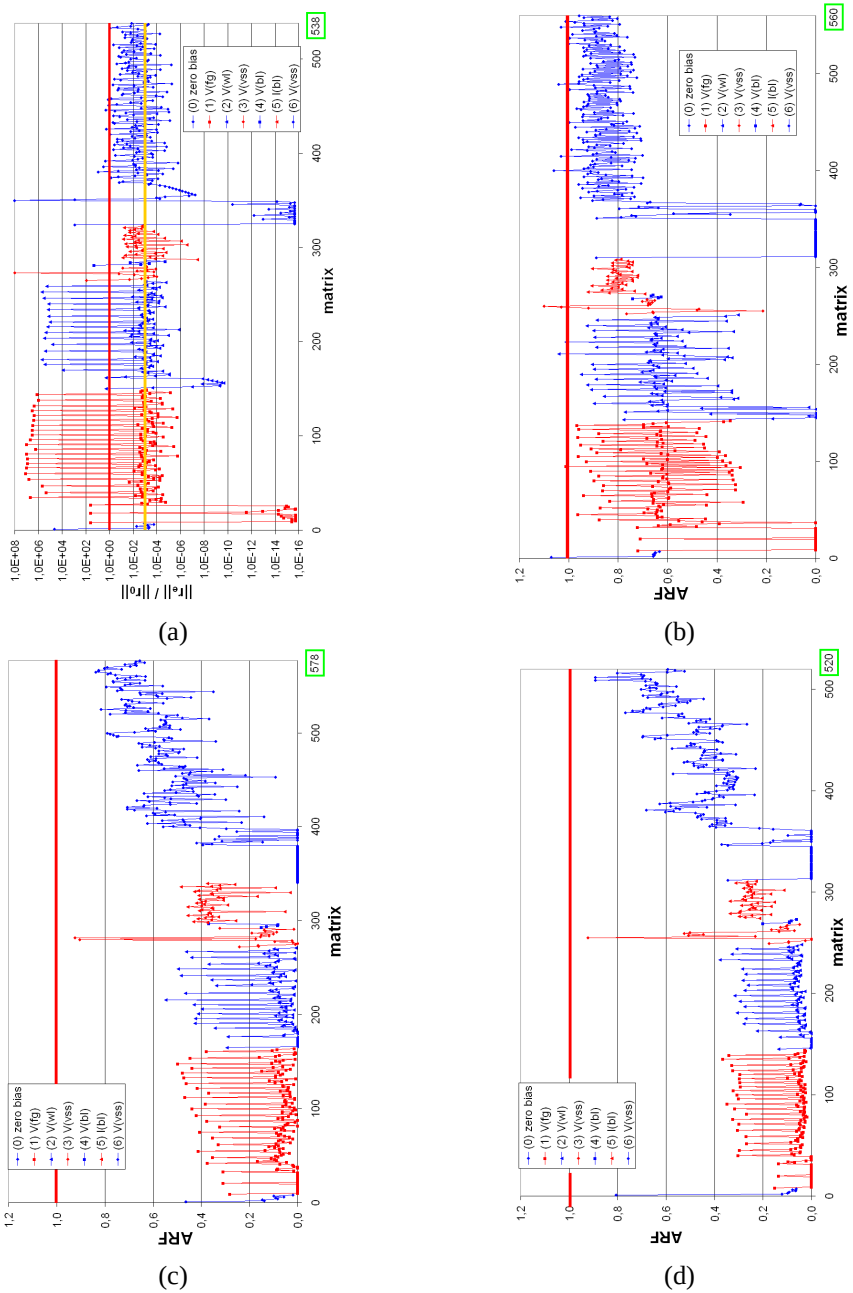


Figure 5.20: EEPROM example: results for (a) the TAURUS solver, (b) ILU-BiCGstab, $\epsilon = 10^{-6}$, (c) PAMG-BiCGstab, $\epsilon = 10^{-3}$, (d) PAMG-BiCGstab, $\epsilon = 10^{-6}$.

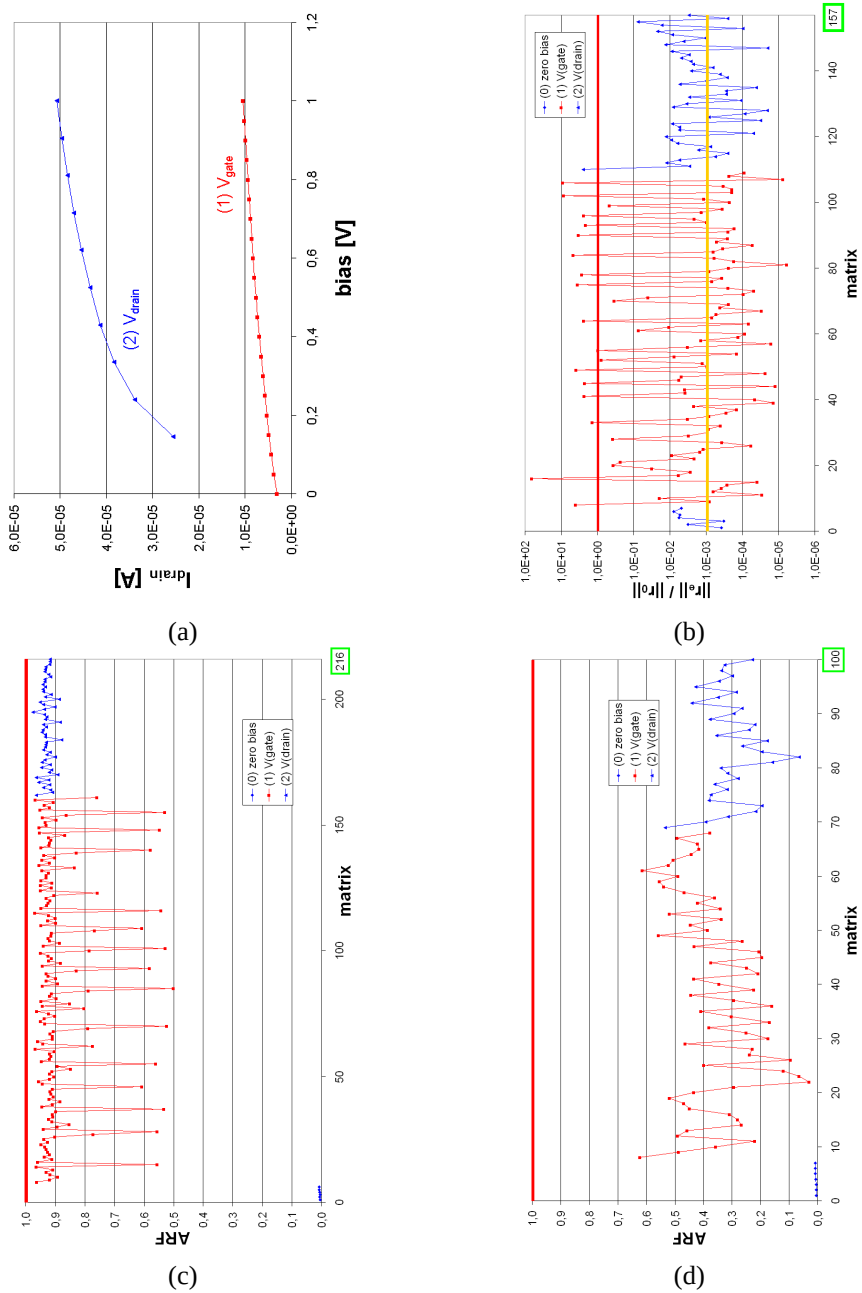


Figure 5.21: FinFET example: (a) current-voltage characteristics, (b) results for the TAURUS solver, (c) results for ILU-BiCGstab, (d) results for PAMG-BiCGstab.

Chapter 6

Conclusions and Outlook

In this thesis, a general AMG methodology for PDE systems has been developed, and its theory as well as its practical realization and (industrial) application have been investigated. Our AMG methodology extends classical AMG by a straightforward unknown- and a particularly powerful point-based strategy. Concrete approaches differ in the amount of information[1] they employ and in the concrete choice of smoothing, coarsening and interpolation. We summarize the main results and give an outlook on future research.

Unknown-based AMG (UAMG) is certainly the simplest strategy for solving PDE systems. Nevertheless, it is quite an efficient preconditioner for some practical applications. Essential conditions for this strategy to work are that smoothing causes the resulting error to be smooth separately for each unknown and that the unknown cross-couplings are not too strong in the sense of a small ρ_u, a new measure introduced in this thesis. Advantages of this strategy are that it can easily cope with anisotropies which are different between the different unknowns and that unknowns can virtually be distributed arbitrarily across mesh points. It can efficiently be applied, for instance, to certain linear elasticity problems as has been demonstrated for applications in industrial semiconductor stress analysis. However, we have also seen limits of UAMG, for instance for reaction-diffusion models with a large coupling between the different unknowns, even if this coupling exists only at one point.

As a main contribution of this thesis, a **general framework for point-based approaches** has been introduced, which employs a primary matrix to construct a point-based coarsening. A necessary condition for point-based AMG (PAMG) to make sense is that the unknowns are discretized on essentially the same - real or virtual - "grid", a condition which is often fulfilled in practice. This strategy is especially well suited for situations in which a point-oriented relaxation produces an error which is characterized by the same kind of algebraic smoothness for each of the unknowns, and in which a primary matrix $\boldsymbol{P}$ and an interpolation can be defined so that both of them reflect the directions of smoothness sufficiently well. Several possibilities for selecting a primary matrix and for the computation of the final interpolation weights have been discussed. As a result, our framework contains many degrees of freedom and thus allows many different concrete PAMG approaches. A special focus has been on the development of cheap primary matrices and interpolation schemes for practical applications. Although it seems clear that a suitable AMG approach (based on unknowns or points) for all types of PDE systems cannot be found, our point-based strategy provides a rich environment for defining efficient and robust approaches for a variety of relevant PDE systems. In general, as for UAMG, efficiency and robustness are drastically increased by applying

[1] The matrix A, the right-hand-side b, the variable-to-unknown mapping, the variable-to-point mapping, and coordinates are the *maximum* amount of information employed.

PAMG not stand-alone, but as a preconditioner. PAMG's efficiency has been demonstrated, in particular, for reaction-diffusion and, with special emphasis, to highly nonlinear, very ill-conditioned drift-diffusion systems, arising in industrial semiconductor simulation. Whereas the most efficient point-based preconditioner for the reaction diffusion systems is oriented on the distances of the grid nodes, the point-based preconditioner for the drift-diffusion systems is oriented on norms of the $A_{(k,l)}$. That AMG's applicability has been extended to such numerically challenging PDE systems is another main contribution of this thesis.

We have also extended the classical **AMG theory** for scalar PDEs to both unknown- and point-based strategies. The theory developed is applicable to essentially (block-)positive type matrices and certain variations thereof. Not unexpectedly, strong conditions have to be fulfilled to make the new theorems on two-level convergence applicable. However, as in the scalar case, the qualitative statements also hold in much more general situations. In particular, numerical results indeed confirm that SU-interpolation does often not perform worse than block-interpolation. Moreover, it is considerably cheaper and often even more robust.

The **solver library SAMG** features, in particular, the realization of our general AMG methodology. It fulfills the properties listed in the introduction in the following ways. All (accelerated) AMG approaches being part of SAMG are scalable if properly applied. For known reasons, this cannot be proved rigorously but is observed in practice. The approaches exhibit reasonable complexities[2]. SAMG can easily be plugged into existing simulation codes. It is a very flexible system for experts. Many different components can be selected for smoothing, coarsening, interpolation, and acceleration. Concrete AMG approaches can typically be applied as preconditioners to large problem classes without loosing robustness. Therefore, each concrete overall approach presents a black-box solver for the problem class(es) it can be applied to. Moreover, SAMG provides a far more efficient behavior for many problem classes than the standard one-level solvers usually employed in industrial simulation codes.

One topic of **future research** will be the investigation of SAMG's applicability to more problem classes, especially applications in industrial oil reservoir simulation (already in progress) and Navier-Stokes equations on non-staggered[3] grids. As indicated in this thesis, the palette (and maybe number) of primary matrices might have to be extended for some applications. In the case of Navier-Stokes, for instance, a physically reasonable primary matrix might be a discrete Laplacian or might arise from a pressure-correction equation. Another direction of future research will be "stronger" smoothers. In particular for drift-diffusion systems and oil reservoir simulation, we have just started investigating block-ILU/ILUT variants. We will also be concerned with the improvement of AMG for structural mechanics. Not only in this context would an appropriate treatment of (nearly) singular matrices be of high practical importance. Of particular relevance for industry are **parallelizations** of SAMG (already in progress), based on both OpenMP and MPI, as well as "self-learning" SAMG-parameter optimization algorithms (towards an **"intelligent solver"**; currently being investigated).

[2]Typically, AMG preconditioners employing aggressive coarsening and GS smoothing need $c_{\mathrm{prec}} \in [1.0, 1.5]$ times the memory needed for the standard one-level preconditioner ILU(0).

[3]For staggered grids, an extension of SAMG's data structure to allowing overlapping variable-clusters and the development of suitable AMG operators would be necessary. Whether and how the latter can be achieved, is an open question.

Appendix A

Auxiliary Results and Additional Proofs

A.1 Nonpositive Diagonal Entries

As has been proved in Lemma 3.1, if the input matrix A is symmetric positive definite, so are the coarser-level matrices, at least up to round-off. We can generalize this even further:

Corollary A.1 *Let A_h be positive definite, and let I_H^h have full rank. Then A_H is also positive definite.*

Proof. This is an immediate consequence of (2.10) and

$$(A_H v^H, v^H)_E = (I_h^H A_h I_H^h v^H, v^H)_E = (A_h I_H^h v^H, I_H^h v^H)_E \,. \quad \blacktriangle$$

If A_h is positive definite, its diagonal entries are positive, and the above Corollary proves that also the diagonal entries of A_H are positive then. Practically, however, it might happen that some coarse-level diagonal entries become (numerically) zero or even negative, in particular, if A_h positive definite is not strictly fulfilled. Very small coarse-level diagonal entries occur, for instance, for the drift-diffusion matrices discussed in Section 5.3.2. Besides the technical problems such exceptional matrix rows produce, AMG's convergence usually suffers from their occurrence. Ways to handle or avoid nonpositive diagonals occurring for a coarse-level matrix A_H the corresponding finer-level matrix of which has only positive diagonal entries are discussed in the following. In this section, we make the general assumption that

$$\forall\, i \,:\quad a_{ii}^h > 0 \,. \tag{A.1}$$

A.1.1 Problem Formulation

Let the indices in C be numbered $i_1, \ldots, i_c$ with $c := |C|$ being the number of C-variables. Recalling the definitions made in Section 2.4.3, we can write

$$I_{FC} = \begin{pmatrix} w^{(i_1)} & \ldots & w^{(i_c)} \end{pmatrix}$$

with, for all $i \in C$,

$$w^{(i)} := (w_j^{(i)})_{j \in F} := (w_{ji})_{j \in F}$$

being the i-th column of I_{FC} ($w_{ji} = 0$ for all $i \neq P_j$). Let e_i^H be the i-th unit vector (i.e. the i-th component of e_i^H being 1, zero the remainder) with length c. Then

$$\begin{aligned} a_{ii}^H &= (e_i^H)^T A_H e_i^H \\ &= (e_i^H)^T (I_H^h)^T A_h I_H^h e_i^H \\ &= (I_H^h e_i^H)^T A_h (I_H^h e_i^H) \end{aligned}$$

Due to (2.21), and because the vector $I_H^h e_i^H$ is the i-th column vector of $I_H^h = (I_{FC}, I_{CC})^T$, we obtain

$$\begin{aligned} a_{ii}^H &= \left(\begin{pmatrix} I_{FC} \\ I_{CC} \end{pmatrix} e_i^H \right)^T \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} I_{FC} \\ I_{CC} \end{pmatrix} e_i^H \\ &= a_{ii}^h + (w^{(i)})^T A_{FF} w^{(i)} + (e_i^H)^T A_{CF} w^{(i)} + (w^{(i)})^T A_{FC} e_i^H \end{aligned}$$

and finally

$$a_{ii}^H = a_{ii}^h + \sum_{j,k \in F} w_j^{(i)} w_k^{(i)} a_{jk}^h + \sum_{j \in F} w_j^{(i)} (a_{ij}^h + a_{ji}^h) \,. \tag{A.2}$$

In the following, our aim is to fulfill a condition analogous to (A.1) also for A_H, i.e.

$$\forall\, i \in C : \quad a_{ii}^H > 0 \,.$$

Because of (A.2), this is equivalent to

$$\forall\, i \in C : \quad a_{ii}^h + \sum_{j,k \in F} w_j^{(i)} w_k^{(i)} a_{jk}^h + \sum_{j \in F} w_j^{(i)} (a_{ij}^h + a_{ji}^h) > 0 \,. \tag{A.3}$$

To be more specific, with a given A_h we want to find conditions on I_{FC} or possibilities to modify I_{FC} so that (A.3) is fulfilled.

A.1.2 Different Workarounds

A.1.2.1 Brute-Force Method

Assume that $a_{ii}^H \leq 0$ for an $i \in C$. Due to our general assumption, we know that $a_{ii}^h > 0$. Therefore, by setting some or even all interpolatory weights $w_j^{(i)}$ ($j \in F$) to zero, we can always force also a_{ii}^H to become positive. This process is accompanied by a rescaling of the remaining weights in order to preserve row sums of the interpolation matrix.

However, in extreme cases, all interpolatory weights of a variable $j \in F$ are set to zero during this elimination process such that the interpolation formula for variable j is completely destroyed. Moreover, the Galerkin operator which is computed with the remaining, rescaled interpolatory weights might again have some nonpositive diagonal entries, and the procedure has to be repeated. Overall, this method is a “brute-force” method to arrive at (A.3) and does not always work.

A.1.2.2 A More Sophisticated Method

The following ansatz is used to modify the interpolatory weights as less as possible.

ANSATZ: Try to replace all original $w^{(i)}$ by $f^{(i)}w^{(i)}$ with a positive scaling factor $f^{(i)}$ as close to 1 as possible so that (A.3) is fulfilled.

Two questions arise:

1. Under the assumptions of Section A.1.1, do such $f^{(i)}$ always exist?
2. If they exist, how can they be computed?

For the remainder of Appendix A.1, we work only with the inequality (A.3) for a fixed $i \in C$. With

$$m := m^{(i)} := \sum_{j,k \in F} w_j^{(i)} w_k^{(i)} a_{jk}^h \,, \qquad f := f^{(i)} \,,$$
$$s := s^{(i)} := \sum_{j \in F} w_j^{(i)} (a_{ij}^h + a_{ji}^h) \,, \qquad a := a_{ii}^h \,,$$

condition A.3 can be reformulated as

$$a + f^2 m + f s > 0 \,. \tag{A.4}$$

In the following discussion, keep in mind that $a > 0$.

1. First, the simple case $m = 0$ is discussed.

$$\text{(A.4)} \quad \Leftrightarrow \quad \begin{cases} f > -\frac{a}{s} & \text{for } s > 0 \,, \\ f < -\frac{a}{s} & \text{for } s < 0 \,, \\ f \text{ arbitrary} & \text{for } s = 0 \,. \end{cases}$$

   Therefore, in each case, a positive f can be found which fulfills (A.4).

2. Now, be $m \neq 0$. Define $D := (\frac{s}{2m})^2 - \frac{a}{m}$. Then

$$\text{(A.4)} \quad \Leftrightarrow \quad \begin{cases} m\left(f + \frac{s}{2m} - \sqrt{D}\right)\left(f + \frac{s}{2m} + \sqrt{D}\right) > 0 & \text{for } D \geq 0 \,, \\ m\left((f + \frac{s}{2m})^2 + (\frac{a}{m} - (\frac{s}{2m})^2)\right) > 0 & \text{for } D < 0 \,. \end{cases}$$

   (a) Case $D \geq 0$ and $m > 0$ and $s > 0$:

$$\begin{aligned} \text{(A.4)} \quad \Leftrightarrow \quad & (f > -\frac{s}{2m} + \sqrt{D} \quad \text{and} \quad f > -\frac{s}{2m} - \sqrt{D}) \text{ or} \\ & (f < -\frac{s}{2m} + \sqrt{D} \quad \text{and} \quad f < -\frac{s}{2m} - \sqrt{D}) \end{aligned}$$

   Because of $a, m, s > 0$, $\sqrt{D} = \sqrt{(\frac{s}{2m})^2 - \frac{a}{m}} < \frac{s}{2m}$ and $-\frac{s}{2m} < 0$. Therefore, the right hand sides of the first two inequalities are negative, and each $f \geq 0$ fulfills the above (full) condition, in particular $f = 1$. Hence, (A.3) already holds without modification of I_{FC}.

(b) Case $D \geq 0$ and $m > 0$ and $s < 0$:

$$\text{(A.4)} \quad \Leftrightarrow \quad (f > -\frac{s}{2m} + \sqrt{D}) \text{ or } (f < -\frac{s}{2m} - \sqrt{D})$$

Both right hand sides of the inequalities are positive because of $-\frac{s}{2m} > 0$ and $\sqrt{D} < \frac{|s|}{2m}$. Therefore, a positive f can always be found which fulfills (A.4).

(c) Case $D \geq 0$ and $m > 0$ and $s = 0$:
not possible because of assumption $(\frac{s}{2m})^2 - \frac{a}{m} = D \geq 0$ and $a, m > 0$.

(d) Case $D \geq 0$ and $m < 0$:

$$\begin{aligned} \text{(A.4)} \quad \Leftrightarrow \quad & (f > -\frac{s}{2m} + \sqrt{D} \quad \text{and} \quad f < -\frac{s}{2m} - \sqrt{D}) \text{ or} \\ & (f < -\frac{s}{2m} + \sqrt{D} \quad \text{and} \quad f > -\frac{s}{2m} - \sqrt{D}) \\ \Leftrightarrow \quad & -\frac{s}{2m} - \sqrt{D} < f < -\frac{s}{2m} + \sqrt{D} \end{aligned}$$

Because of $m < 0$, $\sqrt{D} = \sqrt{(\frac{s}{2m})^2 - \frac{a}{m}} > |\frac{s}{2m}|$. Therefore the left hand side of the inequality for f is negative, the right hand side positive, and each $0 \leq f < -\frac{s}{2m} + \sqrt{D}$ fulfills (A.4).

(e) Case $D < 0$: Because of $(\frac{s}{2m})^2 - \frac{a}{m} = D < 0$, the inequalities $\frac{a}{m} - (\frac{s}{2m})^2 > 0$ and $m > 0$ hold, so that the full inequality is fulfilled regardless of the choice of f. In particular, $f = 1$ works, which means that (A.3) is already fulfilled without any modification of I_{FC}.

The discussion of all possible cases has shown that always a positive f can be found which fulfills (A.4). Depending on the case and the resulting bounds on f, an f should be chosen which is as close to 1 as possible in order to assure that the matrix data in I_{FC} - and thereby the interpolation operator I_H^h - are not fully destroyed. Always check if $f = 1$ fulfills the conditions. If not, proceed as follows.

- If only one bound (besides $0 < f$) is given, choose an $\epsilon \in]0, 1[$ and $(1 - \epsilon)$ times an upper bound or $(1 + \epsilon)$ times a lower bound for f, respectively.
- If $f < L$ or $f > U$ must be fulfilled, take either $f = (1 - \epsilon)L$ or $f = (1 + \epsilon)U$ depending on which one is closer to 1.
- If f must be contained in an interval $]L, U[$, take either $f = (1+\epsilon)L$ or $f = (1-\epsilon)U$, depending on which one is closer to 1, if both values are within $]L, U[$. Otherwise choose $f = (L + U)/2$.

To compute a suitable f, the values of m and s have to be computed. Because in an AMG code usually first A_H and therefore the values a_{ii}^H are computed, only one of them, m or s, has to be computed from scratch. The other value can be obtained from $a_{ii}^H = a_{ii}^h + m + s$.

In summary, we can answer both questions, posed above, positively and are therefore able to scale the columns of I_{FC} with suitable $f^{(i)} > 0$ in such a way that with this scaled I_{FC} (A.3) is fulfilled.

A.2 Proof of Lemma 3.9

Lemma A.1 *Let $i, n \in \mathbb{N}$, v_l $(l = 1, \ldots, i)$ vectors in $\mathbb{R}^n$, W_l $(l = 1, \ldots, i)$ $(n \times n)$-matrices, $||\cdot||$ for vectors a norm and for matrices the operator norm induced by this vector norm, and $\mu := \sum_{l=1}^{i} ||W_l||$. Then the following inequality holds:*

$$\left|\left|\sum_{l=1}^{i} W_l v_l\right|\right|^2 \leq \mu \sum_{l=1}^{i} ||W_l|| \cdot ||v_l||^2 .$$

Proof. First, let $\mu = \sum_{l=1}^{i} ||W_l|| = 1$. Induction over i:
$i = 1$: Because of $||W_1|| = 1$:

$$||W_1 v_1||^2 \leq ||W_1||^2 ||v_1||^2 \leq ||W_1|| \cdot ||v_1||^2 .$$

$i \to i+1$: Let $||W_i|| + ||W_{i+1}|| > 0$ (trivial otherwise!). Then

$$\begin{aligned}\left|\left|\sum_{l=1}^{i+1} W_l v_l\right|\right|^2 &= \left|\left|\sum_{l=1}^{i-1} W_l v_l + (||W_i|| + ||W_{i+1}||)\frac{W_i v_i + W_{i+1} v_{i+1}}{||W_i|| + ||W_{i+1}||}\right|\right|^2 \\ &\leq \sum_{l=1}^{i-1} ||W_l|| \cdot ||v_l||^2 + (||W_i|| + ||W_{i+1}||)\left(\frac{||W_i v_i + W_{i+1} v_{i+1}||}{||W_i|| + ||W_{i+1}||}\right)^2\end{aligned}$$

because $\sum_{l=1}^{i-1} ||W_l|| + (||W_i|| + ||W_{i+1}||) = 1$ (i summands). Obviously,

$$\begin{aligned} & ||v_i||^2 - 2||v_i|| \cdot ||v_{i+1}|| + ||v_{i+1}||^2 \geq 0 \\ \Leftrightarrow \quad & 2||W_i|| \cdot ||v_i|| \cdot ||W_{i+1}|| \cdot ||v_{i+1}|| \leq ||W_i|| \cdot ||W_{i+1}||(||v_i||^2 + ||v_{i+1}||^2) \\ \Rightarrow \quad & ||W_i v_i + W_{i+1} v_{i+1}||^2 \leq ||W_i||^2 ||v_i||^2 + ||W_{i+1}||^2 ||v_{i+1}||^2 \\ & \qquad + ||W_i|| \cdot ||W_{i+1}||(||v_i||^2 + ||v_{i+1}||^2) \\ \Rightarrow \quad & \frac{||W_i v_i + W_{i+1} v_{i+1}||^2}{||W_i|| + ||W_{i+1}||} \leq ||W_i|| \cdot ||v_i||^2 + ||W_{i+1}|| \cdot ||v_{i+1}||^2 . \end{aligned}$$

Therefore,

$$\left|\left|\sum_{l=1}^{i+1} W_l v_l\right|\right|^2 \leq \sum_{l=1}^{i+1} ||W_l|| \cdot ||v_l||^2$$

which proves the lemma for $\mu = 1$. Now, let $\mu = \sum_{l=1}^{i} ||W_l|| > 0$ be arbitrary. Because of

$$\sum_{l=1}^{i} \left|\left|\frac{W_l}{\mu}\right|\right| = \frac{1}{\mu} \sum_{l=1}^{i} ||W_l|| = 1$$

we now obtain

$$\left|\left|\sum_{l=1}^{i} \frac{W_l}{\mu} v_l\right|\right|^2 \leq \sum_{l=1}^{i} \left|\left|\frac{W_l}{\mu}\right|\right| \cdot ||v_l||^2$$

which shows

$$\left|\left|\sum_{l=1}^{i} W_l v_l\right|\right|^2 \leq \mu \sum_{l=1}^{i} ||W_l|| \cdot ||v_l||^2 . \qquad \blacksquare$$

Bibliography

[1] ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. *LAPACK Users' Guide*, 3d ed. SIAM, Philadelphia (PA), USA, 1999.

[2] ASCHER, U., MARKOWICH, P., SCHMEISER, C., STEINRÜCK, H., AND WEISS, R. Conditioning of the steady state semiconductor device problem. *SIAM J. Appl. Math. 49* (1989), 165–185.

[3] BENZI, M. Preconditioning techniques for large linear systems: a survey. *J. Comp. Physics 182* (2002), 418–477.

[4] BLAHETA, R. A multilevel method with overcorrection by aggregation for solving discrete elliptic problems. *J. Comp. Appl. Math. 24* (1988), 227–239.

[5] BRAESS, D. Towards algebraic multigrid for elliptic problems of second order. *Computing 55* (1995), 379–393.

[6] BRAESS, D. *Finite Elemente*. Springer, Berlin, 2nd ed., 1997.

[7] BRANDT, A. Multi-level adaptive solutions to boundary-value problems. *Math. Comput. 31* (1977), 333–390.

[8] BRANDT, A. Algebraic multigrid theory: the symmetric case. *Appl. Math. Comp. 19* (1986), 23–56.

[9] BRANDT, A. General highly accurate algebraic coarsening. *Electr. Trans. Numer. Anal. 10* (2000), 1–20.

[10] BRANDT, A., MCCORMICK, S., AND RUGE, J. *Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations*. Tech. rep., Institute for Computational Studies, POB 1852, Fort Collins (CO), USA, 1982.

[11] BRANDT, A., MCCORMICK, S., AND RUGE, J. Algebraic Multigrid (AMG) for sparse matrix equations. In *Sparsity and its Applications*, D. Evans, Ed. Cambridge University Press, Cambridge, UK, 1984, pp. 257–284.

[12] BREZINA, M., CLEARY, A., FALGOUT, R., HENSON, V., JONES, J., MANTEUFFEL, T., MCCORMICK, S., AND RUGE, J. Algebraic multigrid based on element interpolation (AMGe). *SIAM J. Sci. Comp. 22* (2000), 1570–1592.

[13] BRÖKER, O. *Parallel Multigrid Methods using Sparse Approximate Inverses*. PhD thesis, Swiss Fed. Inst. of Technology (ETH), Zürich, Switzerland, 2003.

[14] CHAN, T., AND WAN, W. Robust multigrid methods for nonsmooth coefficient elliptic linear systems. *J. Comp. Appl. Math. 123* (2000), 323–352.

[15] CHAN, T., XU, J., AND ZIKATANOV, L. An agglomeration multigrid method for unstructured grids. In *Proceedings of the Tenth International Conference on Domain Decomposition methods, Boulder (CO), USA, August 1997*, J. Mandel, C. Farhat, and X.-C. Cai, Eds., vol. 218 of *Contemporary Mathematics (1998)*. pp. 67–81.

[16] CHARTIER, T., FALGOUT, R., HENSON, V., JONES, J., MANTEUFFEL, T., MCCORMICK, S., RUGE, J., AND VASSILEVSKI, P. Spectral AMGe (ρAMGe). To appear in *SIAM J. Sci. Comp.* Also available as Lawrence Livermore Laboratory Tech. Rep. UCRL-JC-146369.

[17] CLEARY, A., FALGOUT, R., HENSON, V., JONES, J., MANTEUFFEL, T., MCCORMICK, S., MIRANDA, G., AND RUGE, J. Robustness and scalability of algebraic multigrid. *SIAM J. Sci. Comp. 21* (2000), 1886–1908.

[18] CLEES, T. Algebraic multigrid for selected systems of partial differential equations. *Abschlussbericht, Graduiertenkolleg Scientific Computing*, Universität zu Köln, Cologne, Germany, 2003. `http://www.informatik.uni-koeln.de/ls_speckenmeyer/NeuGK/berichte.shtml`.

[19] CLEES, T., AND STÜBEN, K. Algebraic multigrid for industrial semiconductor device simulation. In *Challenges in Scientific Computing - CISC 2002. Proceedings of the Conference "Challenges in Scientific Computing", Berlin, October 2-5, 2002*, E. Bänsch, Ed., vol. 35 of *Lecture Notes in Computational Science and Engineering*, Springer, Berlin, 2003, pp. 110–130.

[20] DEMARI, A. An accurate numerical steady-state one-dimensional solution of the p-n junction. *Solid-State Electron. 11* (1968), 33–58.

[21] DENDY, J. Black box multigrid. *J. Comp. Physics 48* (1982), 366–386.

[22] FALGOUT, R., AND VASSILEVSKI, P. On generalizing the AMG framework. Submitted to *SIAM J. Numer. Anal.* Also available as Lawrence Livermore National Laboratory Tech. Rep. UCRL-JC-150807.

[23] FUHRMANN, J., AND GÄRTNER, K. Incomplete factorization and linear multigrid algorithms for the semiconductor device equations. In *Proceedings of the IMACS International Symposium on Iterative Methods in Linear Algebra, Brussels, April 2-4, 1991*, R. Beauwens and P. de Groen, Eds., Elsevier, Amsterdam, 1992, pp. 493–503.

[24] FÜLLENBACH[1], T. Algebraische Mehrgitterverfahren für Systeme partieller Differentialgleichungen. *Forschungsbericht, Graduiertenkolleg Scientific Computing 4/2000 - 3/2001*, Universität zu Köln, Cologne, Germany, 2001. `http://www.informatik.uni-koeln.de/ls_speckenmeyer/NeuGK/berichte.shtml`.

[25] FÜLLENBACH[1], T. Algebraic multigrid for selected systems of partial differential equations. *Forschungsbericht, Graduiertenkolleg Scientific Computing 4/2001 - 3/2002*, Universität zu Köln, Cologne, Germany, 2002. `http://www.informatik.uni-koeln.de/ls_speckenmeyer/NeuGK/berichte.shtml`.

[26] FÜLLENBACH[1], T., AND STÜBEN, K. Algebraic multigrid for selected PDE systems. In *Elliptic and Parabolic Problems, Rolduc and Gaeta 2001. Proceedings of the 4th European Conference*, World Scientific, London, 2002, pp. 399–410.

[27] FÜLLENBACH[1], T., STÜBEN, K., AND MIJALKOVIĆ, S. Application of an algebraic multigrid solver to process simulation problems. In *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), Seattle (WA), USA, 2000*, IEEE, Catalog Number: 00TH8502, ISBN 0-7803-6279-9, pp. 225–228.

[28] GAJEWSKI, H., AND GÄRTNER, K. On the discretization of van Roosbroeck's equations with magnetic field. *Z. angew. Math. Mech. (ZAMM) 76* (1996), 247–264.

[29] GRIEBEL, M., NEUNHOEFFER, T., AND REGLER, H. Algebraic multigrid methods for the solution of the Navier-Stokes equations in complicated domains. *Int. J. Numer. Methods on Heat and Fluid Flow 26* (1998), 281–301.

[30] GRIEBEL, M., OELTZ, D., AND SCHWEITZER, M. An algebraic multigrid method for linear elasticity. *SIAM J. Sci. Comp.* (2002). Submitted.

[1] now: Clees

[31] GUILLARD, H., AND VANĚK, P. *An aggregation multigrid solver for convection-diffusion problems on unstructured meshes*. UCD/CCM Report 130. Center for Computational Mathematics, University of Colorado at Denver, Denver (CO), USA, 1998.

[32] HAASE, G., LANGER, U., REITZINGER, S., AND SCHÖBERL, J. A general approach to algebraic multigrid methods. *Universität Linz, Austria. SFB-Report No. 00-33* (2000).

[33] HAASE, G., LANGER, U., REITZINGER, S., AND SCHÖBERL, J. Algebraic multigrid methods based on element preconditioning. *Int. J. Comp. Math. 78* (2001), 575–598.

[34] HACKBUSCH, W. *Multigrid Methods and Applications*. Springer, Berlin, 1985.

[35] HENSON, V., AND VASSILEVSKI, S. Element-free AMGe: general algorithms for computing interpolation weights. *SIAM J. Sci. Comp. 23* (2001), 629–650.

[36] HÖFLER, A. *Development and Application of a Model Hierarchy for Silicon Process Simulation*. PhD thesis, Swiss Fed. Inst. of Technology (ETH), Zürich, Switzerland, 1997. Also available as Series in Microelectronics, Vol. 69, Hartung-Gorre, Konstanz, Germany, 1997.

[37] HÖFLER, A., AND STRECKER, N. *On the Coupled Diffusion of Dopants and Silicon Point Defects*. Tech. Rep. 94/11, Integrated Systems Laboratory, Swiss Fed. Inst. of Technology (ETH), Zürich, Switzerland, 1994.

[38] HUANG ET AL., X. Sub-50 nm P-Channel FinFET. *IEEE Trans. Electr. Dev. 48* (2001), 880–886.

[39] ISE INTEGRATED SYSTEMS ENGINEERING AG. *GENESISe, ISE TCAD Release 6.1*. Zürich, Switzerland, 2000.

[40] JONES, J., AND VASSILEVSKI, S. AMGe based on element agglomeration. *SIAM J. Sci. Comp. 23* (2001), 109–133.

[41] JOPPICH, W., AND MIJALKOVIĆ, S. *Multigrid Methods for Process Simulation*. Springer, Wien, 1993.

[42] JOPPICH, W., AND MIJALKOVIĆ, S. Semiconductor process modeling. In *Wiley Encyclopedia of Electrical and Electronics Engineering*, J. Webster, Ed., vol. 19. Wiley, New York, 1999, pp. 127–139.

[43] JÜNGEL, A. *Macroscopic models for semiconductor devices: a review*. Konstanzer Schriften in Mathematik und Informatik 106, Universität Konstanz, Germany, 2000.

[44] KERKHOVEN, T. On the Scharfetter-Gummel box method. In *Simulation of Semiconductor Devices and Processes, Vol. 5.*, S. Selberherr, H. Stippel, and E. Strasser, Eds. Wien, 1993, pp. 237–240.

[45] KLAWONN, A., AND RHEINBACH, O. *A parallel implementation of dual-primal FETI methods for three dimensional linear elasticity using a transformation of basis*. Tech. Rep. SM-E-601, Fachbereich Mathematik, Universität Duisburg-Essen, Germany, February 2005.

[46] KLAWONN, A., RHEINBACH, O., AND WIDLUND, O. Some computational results for dual-primal FETI methods for three dimensional elliptic problems. In *Domain Decomposition Methods in Science and Engineering*, R. Kornhuber, R. Hoppe, D. Keyes, J. Periaux, O. Pironneau, and J. Xu, Eds., vol. 40 of *Lecture Notes in Computational Science and Engineering*, Springer, Berlin, 2005, pp. 361–368.

[47] KOSIK, R., FLEISCHMANN, P., HAINDL, B., PIETRA, P., AND SELBERHERR, S. On the interplay between meshing and discretization in three-dimensional diffusion simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 19* (2000), 1233–1240.

[48] KRECHEL, A., AND STÜBEN, K. Operator dependent interpolation in algebraic multigrid. In *Proceedings of the Fifth European Multigrid Conference, Stuttgart, October 1-4, 1996*, vol. 3 of *Lecture Notes in Computational Science and Engineering*, Springer, Berlin, 1998.

[49] KRECHEL, A., AND STÜBEN, K. Parallel algebraic multigrid based on subdomain blocking. *Parallel Computing 27* (2001), 1009–1031. Also available as *GMD Report 71*, December 1999. Fraunhofer SCAI, Sankt Augustin, Germany.

[50] LAW, M. *Florida Object-Oriented Process Simulator, FLOOPS Manual*. University of Florida, Gainesville (FL), USA, 1993.

[51] LI, Z., Y. SAAD, Y., AND M. SOSONKINA, M. pARMS: a parallel version of the algebraic recursive multilevel solver. *Numer. Lin. Alg. Appl. 10* (2003), 485–509.

[52] MANDEL, J. Local approximation estimators for algebraic multigrid. *Electr. Trans. Numer. Anal. 15* (2003), 56–65.

[53] MANDEL, J., BREZINA, M., AND VANĚK, P. Energy optimization of algebraic multigrid bases. *Computing 62* (1999), 205–228.

[54] MARKOWICH, P. *The Stationary Semiconductor Device Equations*. Springer, Wien, 1986.

[55] MARKOWICH, P., RINGHOFER, C., AND SCHMEISER, C. *Semiconductor Equations*. Springer, Wien, 1990.

[56] MIJALKOVIĆ, S., AND MIHAJLOVIĆ, M. Component-wise algebraic multigrid preconditioning for the iterative solution of stress analysis problems from microfabrication technology. *Comm. Numer. Methods Engrg. 17* (2001), 737–747.

[57] MILLER, J., AND WANG, S. A new non-conforming Petrov-Galerkin finite-element method with triangular elements for a singularly perturbed advection-diffusion problem. *IMA J. Numer. Anal. 14* (1994), 257–276.

[58] OELTZ, D. *Algebraische Mehrgittermethoden für Systeme partieller Differentialgleichungen*. Master's thesis, Institut für Angewandte Mathematik, Rheinische Friedrich-Wilhelms-Universität Bonn, Germany, 2001.

[59] OERTEL, K.-D., AND STÜBEN, K. *Multigrid with ILU-smoothing: systematic tests and improvements*. Arbeitspapiere der GMD 306, Fraunhofer SCAI, Sankt Augustin, Germany, 1988.

[60] PAPADOPOULOS, A., AND TCHELEPI, H. Block smoothed aggregation AMG preconditioning for oil reservoir simulation systems. *Oxford University, Computing Laboratory Numerical Analysis Report No. 03/04* (Oxford, UK, 2003). `http://web.comlab.ox.ac.uk/oucl/publications/natr/`.

[61] POMP, A., SCHENK, O., AND FICHTNER, W. *An ILU Preconditioner Adapted to Diffusion Processes in Semiconductors*. Tech. Rep. 99/11, Integrated Systems Laboratory, Swiss Fed. Inst. of Technology (ETH), Zürich, Switzerland, 1999.

[62] POOLE, G., LIU, Y.-C., AND MANDEL, J. Advancing analysis capabilities in ANSYS through solver technology. *Electr. Trans. Numer. Anal. 15* (2003), 106–121.

[63] RAW, M. A coupled algebraic multigrid method for the 3D Navier-Stokes equations. In *Fast Solvers for Flow Problems, Proceedings of the 10th GAMM-Seminar, Kiel, January 14-16, 1994*, vol. 49 of *Notes on Numerical Fluid Mechanics*, Vieweg, Braunschweig, Germany, 1995, pp. 204–215.

[64] RAW, M. Robustness of coupled algebraic multigrid for the Navier-Stokes equations. In *34th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 15-18, 1996*, AIAA Paper 96-0297, 1996.

[65] REGLER, H. *Anwendungen von AMG auf das Plazierungsproblem beim Layoutentwurf und auf die numerische Simulation von Strömungen*. PhD thesis, Technische Universität München, Munich, Germany, 1997.

[66] REUSKEN, A. On the approximate cyclic reduction preconditioner. *SIAM J. Sci. Comp. 21* (2000), 565–590. Also available as preprint no. 144, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Germany, 1997.

[67] RIES, M., TROTTENBERG, U., AND WINTER, G. A note on MGR methods. *Lin. Alg. Appl. 49* (1983), 1–26.

[68] ROBINSON, G. Parallel computational fluid dynamics on unstructured meshes using algebraic multigrid. In *Parallel Computational Fluid Dynamics 92*, R. Pelz, A. Ecer, and J. Häuser, Eds., Elsevier, Amsterdam, 1993.

[69] RUGE, J. AMG for problems of elasticity. *Appl. Math. Comp. 19* (1986), 293–309.

[70] RUGE, J., AND STÜBEN, K. Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). In *Multigrid Methods for Integral and Differential Equations*, D. Paddon and H. Holstein, Eds., *The Institute of Mathematics and its Applications Conference Series, New Series No. 3*, Clarenden Press, Oxford, UK, 1985, pp. 169–212.

[71] RUGE, J., AND STÜBEN, K. Algebraic Multigrid (AMG). In *Multigrid Methods*, S. McCormick, Ed., vol. 3 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia (PA), USA, 1987, pp. 73–130.

[72] SAAD, Y. ILUT: A dual threshold incomplete ILU factorization. *Numer. Lin. Alg. Appl. 1* (1994), 387–402.

[73] SAAD, Y. Multilevel ILU with reorderings for diagonal dominance. *SIAM J. Sci. Comp.* (2005). To appear.

[74] SAAD, Y. *Iterative Methods for Sparse Linear Systems*. 2nd ed., 2000. `http://www-users.cs.umn.edu/~saad/books.html`.

[75] SAAD, Y., AND SCHULTZ, M. GMRes: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp. 7* (1986), 856–869.

[76] SAAD, Y., AND VAN DER VORST, H. Iterative solution of linear systems in the 20th century. *J. Comp. Appl. Math. 123* (2000), 1–33.

[77] SCHARFETTER, D., AND GUMMEL, H. Large-signal analysis of a silicon read diode oscillator. *IEEE Trans. Electr. Dev. ED-16* (1969), 64–77.

[78] SCHENK, O., RÖLLIN, S., AND GUPTA, A. *The Effects of Nonsymmetric Matrix Permutations and Scalings in Semiconductor Device and Circuit Simulation*. Tech. Rep. 2003/9, Integrated Systems Laboratory, Swiss Fed. Inst. of Technology (ETH), Zürich, Switzerland, 2003.

[79] SCHMIDTHÜSEN, B. *Grid Adaptation for the Stationary Two-Dimensional Drift-Diffusion Model in Semiconductor Device Simulation*. PhD thesis, Swiss Fed. Inst. of Technology (ETH), Zürich, Switzerland, 2001. Also available as Series in Microelectronics, Vol. 126, Hartung-Gorre, Konstanz, Germany, 2002.

[80] SCHRÖDER, J., AND TROTTENBERG, U. Reduktionsverfahren für Differenzengleichungen bei Randwertaufgaben I. *Numer. Math. 22* (1973), 37–68.

[81] SCHRÖDER, J., TROTTENBERG, U., AND REUTERSBERG, H. Reduktionsverfahren für Differenzengleichungen bei Randwertaufgaben I. *Numer. Math. 26* (1976), 429–459.

[82] SELBERHERR, S. *Analysis and Simulation of Semiconductor Devices*. Springer, Wien, 1984.

[83] SENEZ, V., COLLARD, D., FERREIRA, P., AND BACCUS, B. Two-dimensional simulation of local oxidation of silicon: calibrated viscoelastic flow analysis. *IEEE Trans. Electr. Dev. 43* (1996), 720–731.

[84] STOER, J. *Numerische Mathematik*. Springer, Berlin, 7th ed., 1994.

[85] STOER, J., AND BULIRSCH, R. *Numerische Mathematik 2*. Springer, Berlin, 3rd ed., 1990.

[86] STÜBEN, K. Algebraic multigrid (AMG): Experiences and comparisons. *Appl. Math. Comp. 13* (1983), 419–452.

[87] STÜBEN, K. An introduction to algebraic multigrid. In *Multigrid* [94], pp. 413–532. Also available as *GMD Report 70*, November 1999. Fraunhofer SCAI, Sankt Augustin, Germany.

[88] STÜBEN, K. A review of algebraic multigrid. *J. Comp. Appl. Math. 128* (2001), 281–309. Also available as *GMD Report 69*, November 1999. Fraunhofer SCAI, Sankt Augustin, Germany.

[89] STÜBEN, K., AND CLEES, T. *SAMG User's Manual, Release 21c*. Fraunhofer SCAI, Sankt Augustin, Germany, July 2003. `http://www.scai.fraunhofer.de/samg.html`.

[90] STÜBEN, K., DELANEY, P., AND CHMAKOV, S. Algebraic Multigrid (AMG) for Ground Water Flow and Oil Reservoir Simulation. In *Proceedings of the Conference MODFLOW and More 2003: Understanding through Modeling*, International Ground Water Modeling Center (IGWMC), Colorado School of Mines, Golden, Colorado (CO), USA, Sept 17 - 19, 2003.

[91] STÜBEN, K., AND TROTTENBERG, U. Multigrid methods: Fundamental algorithms, model problem analysis and applications. In *Multigrid Methods*, W. Hackbusch and U. Trottenberg, Eds., vol. 960 of *Lecture Notes in Mathematics*, Springer, Berlin, 1982, pp. 1–176. Also available as *GMD-Studien 96*, December 1984. Fraunhofer SCAI, Sankt Augustin, Germany.

[92] SYNOPSYS, MONTAINVIEW (CA), USA (former Avant! Corp., Fremont (CA), USA). *TAURUS, Release 2001.4*, 2001.

[93] SZE, S., Ed. *Modern Semiconductor Device Physics*. Wiley, New York, 1998.

[94] TROTTENBERG, U., OOSTERLEE, C., AND SCHÜLLER, A. *Multigrid*. Academic Press, London, 2001.

[95] TROTTENBERG, U., AND WITSCH, K. *Zur Kondition diskreter elliptischer Randwertaufgaben*. GMD-Studien 60, Fraunhofer SCAI, Sankt Augustin, Germany, 1981.

[96] VAN DER VORST, H. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comp. 13* (1992), 631–644.

[97] VANĚK, P., BREZINA, M., AND MANDEL, J. Convergence of algebraic multigrid based on smoothed aggregation. *Numer. Math. 88* (2001), 559–579.

[98] VANĚK, P., MANDEL, J., AND BREZINA, M. *Algebraic multigrid on unstructured meshes*. UCD/CCM Report 34. Center for Computational Mathematics, University of Colorado at Denver, Denver (CO), USA, 1994.

[99] VANĚK, P., MANDEL, J., AND BREZINA, M. Algebraic multigrid by smoothed aggregation for second and fourth order problems. *Computing 56* (1996), 179–196.

[100] VARGA, R. *Matrix Iterative Analysis*. Prentice Hall, Englewood Cliffs (NJ), USA, 1962.

[101] WABRO, M. Coupled algebraic multigrid methods for the Oseen problem. *Universität Linz, Austria.* (2003), SFB–Report No. 03–2.

[102] WAGNER, C. On the algebraic construction of multilevel transfer operators (for convection-diffusion-reaction equations). In *Multigrid Methods VI, Proceedings of the Sixth European Multigrid Conference, Gent, Belgium, September 27-30, 1999*, E. Dick, K. Riemslagh, and J. Vierendeels, Eds., vol. 14 of *Lecture Notes in Computational Science and Engineering*, Springer, Berlin, 2000.

[103] WAGNER, C. *Introduction to algebraic multigrid. Course notes of an algebraic multigrid course at the University of Heidelberg in the Wintersemester 1998/99*. `http://www.iwr.uni-heidelberg.de/~christian.wagner`, 1999.

[104] WAGNER, C. On the algebraic construction of multilevel transfer operators. *Computing 65* (2000), 73–95.

[105] WAGNER, C., AND WITTUM, G. Adaptive filtering. *Numer. Math. 78* (1997), 305–328.

[106] WALTER, R. *Lineare Algebra und analytische Geometrie*. Vieweg, Braunschweig, Germany, 2nd ed., 1993.

[107] WAN, W., CHAN, T., AND SMITH, B. An energy-minimization interpolation for robust multigrid methods. *SIAM J. Sci. Comp. 21* (2000), 1632–1649.

[108] WATERLOO MAPLE INC. *Maple V Release 4, Version 4.00b*. Waterloo (ON), Canada, 1996.

[109] WEISS, J., MARUSZEWSKI, J., AND SMITH, W. Implicit solution of the Navier-Stokes equations on unstructured meshes. In *13th AIAA CFD Conference, June 1997*, AIAA Paper 97-2103, 1997.

[110] WIENANDS, R. *Extended Local Fourier Analysis for Multigrid: Optimal Smoothing, Coarse Grid Correction, and Preconditioning*. PhD thesis, Universität zu Köln, Cologne, Germany, 2001. Also available as *GMD Research Series 20*, 2001. Fraunhofer SCAI, Sankt Augustin, Germany.

[111] WINTER, G. *Fourieranalyse zur Konstruktion schneller MGR-Verfahren*. PhD thesis, Institut für Angewandte Mathematik, Rheinische Friedrich-Wilhelms-Universität Bonn, Germany, 1983.

List of Figures

List of Tables

Erklärung

Ich versichere, daß ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; daß diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; daß sie - abgesehen von unten angegebenen Teilpublikationen - noch nicht veröffentlicht worden ist sowie, daß ich eine solche Veröffentlichung vor Abschluß des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Ulrich Trottenberg betreut worden.

Teilpublikationen

- Füllenbach, T. und Stüben, K. und Mijalković, S., Application of an algebraic multigrid solver to process simulation problems, Seiten 225–228, *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), Seattle (WA), USA, 2000*, IEEE, Catalog Number: 00TH8502, ISBN 0-7803-6279-9,
- Füllenbach, T., Algebraische Mehrgitterverfahren für Systeme partieller Differentialgleichungen, *Forschungsbericht, Graduiertenkolleg Scientific Computing* 4/2000 - 3/2001, Seiten 39–50, Universität zu Köln, 2001.
- Füllenbach, T., Algebraic multigrid for selected systems of partial differential equations, *Forschungsbericht, Graduiertenkolleg Scientific Computing* 4/2001 - 3/2002, Seiten 27–45, Universität zu Köln, 2002.
- Füllenbach, T. und Stüben, K., Algebraic multigrid for selected PDE systems, Seiten 399–410, *Elliptic and Parabolic Problems, Rolduc and Gaeta 2001. Proceedings of the 4th European Conference*, World Scientific, London, 2002.
- Clees, T. und Stüben, K., Algebraic multigrid for industrial semiconductor device simulation, Seiten 110–130, *Challenges in Scientific Computing - CISC 2002. Proceedings of the Conference Challenges in Scientific Computing", Berlin, October 2-5, 2002*, Bänsch, E. (Editor), Lecture Notes in Computational Science and Engineering 35, Springer, Berlin, 2003.
- Clees, T., Algebraic multigrid for selected systems of partial differential equations, *Abschlußbericht, Graduiertenkolleg Scientific Computing*, Seiten 20–25, Universität zu Köln, 2003.

Datum

Unterschrift