

Multigrid Methods: From Geometrical to Algebraic Versions

Gundolf HAASE

*Institute of Computational Mathematics
Johannes Kepler University of Linz
Altenbergerstr. 69, A-4040 Linz, Austria*

Ulrich LANGER

*Institute of Computational Mathematics
Johannes Kepler University of Linz
Altenbergerstr. 69, A-4040 Linz, Austria*

Notes taken by Stefan REITZINGER

Abstract

Nowadays the multigrid technique is one of the most efficient methods for solving a large class of problems including elliptic boundary value problems for partial differential equations (PDEs) or systems of PDEs. Our lecture notes start with a motivation for the multigrid idea and a brief review of the multigrid history, its situation and the perspectives. The next part of our paper is devoted to the algorithmical aspects of the construction of multigrid methods including also algebraic approaches and parallelization techniques. Algebraic multigrid methods are now quite popular in practical applications because they can be included in conventional finite element packages without changing the data structure of the package. In the theoretical part of our lecture, we present some general approaches to the convergence and efficiency analysis of multigrid methods. Finally, we discuss implementation issues and present some numerical results obtained from the application of our algebraic multigrid package PEBBLES to large-scale problems in medicine and engineering.

1 Motivation of the Multigrid Idea

Let us consider linear systems

$$K_h \underline{u}_h = \underline{f}_h \tag{1}$$

of algebraic equations arising from a Finite Element (FE), Finite Difference (FD), or a Finite Volume (FV) discretization of some formally self-adjoint, second order, scalar, elliptic partial differential equation in a bounded computational domain $\Omega \subset \mathbb{R}^d$ ($d = 1, 2, 3$), where $K_h \in \mathbb{R}^{N_h \times N_h}$, $\underline{f}_h \in \mathbb{R}^{N_h}$, and $\underline{u}_h \in \mathbb{R}^{N_h}$ denote the system matrix (stiffness matrix), the right-hand-side vector (load vector), and the solution vector of the unknown nodal parameters, respectively. The number N_h of unknowns is related to the usual discretization parameter h by the relation $N_h = O(h^{-d})$. The stiffness matrix K_h is supposed to be sparse, symmetric

and positive definite. The spectral condition number $\kappa(K_h)$ of the stiffness matrix behaves typically like $O(h^{-2})$ as the discretization parameter tends to zero. This practically means that the stiffness matrix has a large condition number on a fine grid.

The structure and the bad conditioning of the stiffness matrix K_h considerably affects the efficiency of classical direct and iterative solvers. In the case of direct solvers (e.g. Gaussian elimination, Cholesky factorization), the memory demand M and number Q of arithmetical operations asymptotically grow as $O(h^{-2d+1})$ and $O(h^{-3d+1})$, respectively. The bad conditioning of K_h results in the loss of about $\lg \kappa(K_h)$ valid digits independently of d . In the one-dimensional case ($d=1$), the direct methods are asymptotically optimal with respect to the grow of M and Q . The superior proportional grow of M and Q in 2D ($d=2$) and 3D ($d=3$) is due to the so-called “fill-in” during the elimination, or factorization step. That’s the reason why the direct methods are in general only efficient for small systems related to coarser grids. In contrast to the direct methods, the memory demand of the classical iterative methods such as the Jacobi- and the Gauss-Seidel methods is proportional to the number of unknowns N_h . This fact is due to the sparsity structure of the stiffness matrix K_h . The number of nonzero element per row, or column of K_h is basically constant for a reasonable mesh refinement procedure. Hence, the total Number of Nonzero Elements (NNE) in the stiffness matrix K_h is proportional to the number of unknowns $N_h = O(h^{-d})$. Therefore, the matrix-by-vector multiplication of the type $K_h * \underline{u}_h$ that is the basic operation in every iterative process can be carried out with $O(h^{-d})$ arithmetical operations. So, the memory demand and the costs per iteration step are asymptotically optimal. However, the classical iterative methods suffer from the bad conditioning of the stiffness matrix that leads to a dramatical grow in the number of iterations as h tends to zero. More precisely, the number of iterations $I(\varepsilon)$ that is necessary to reduce the initial iteration error by the factor $\varepsilon \in (0, 1)$ behaves like $O(\kappa(K_h) \ln(\varepsilon^{-1})) = O(h^{-2} \ln(\varepsilon^{-1}))$. Summing up, we observe that the classical iterative methods require asymptotically $Q(\varepsilon) = O(h^{-d-2} \ln(\varepsilon^{-1}))$ arithmetical operations in order to reduce the initial iteration error by the factor ε [3, 59].

Let us have a closer look at the convergence properties of the classical iterative methods for solving finite element equations. For the simplicity of the analysis, we consider the simplest finite (piecewise linear) element discretization of the simplest second order boundary value problem: Find some function $u(\cdot)$ defined on the interval $[0, 1]$ such that the differential equation

$$-u''(x) = f(x), \quad x \in (0, 1) \quad (2)$$

and the homogeneous Dirichlet boundary condition

$$u(0) = u(1) = 0 \quad (3)$$

are fulfilled, where the right-hand side f in (2) is a given function on $(0, 1)$. The boundary value problem (2)-(3) is obviously the one-dimensional analogy of the well-known Dirichlet problem for the Poisson equation that plays an important role in a lot of practical applications. The starting point for the finite element discretization of the boundary value problem (2)-(3) is its variational formulation. Multiplying the differential equation (2) by some test function v that vanishes at the Dirichlet boundary $x = 0$ and $x = 1$, integrating over our computational domain $\Omega = (0, 1)$, and finally integrating by parts, we arrive at the variational formulation:

Find some function $u \in V_0 = H_0^1(0, 1)$ such that

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x)dx \quad \forall v \in V_0, \quad (4)$$

where the Sobolev space $H_0^1(0, 1)$ contains all quadratically integrable function (L_2 - functions) possessing quadratically integrable weak derivatives and vanishing at $x = 0$ and $x = 1$. Now the finite element approximation u_h to our solution u is searched in the form of the piecewise linear, continuous function

$$u_h(x) = \sum_{j=1}^{n-1} u_j p_j(x) \quad (5)$$

with unknown coefficients (nodal values) u_j and given ansatz functions $p_j(x)$ corresponding to the nodes $x_j = jh$, where $j = 1, 2, \dots, N_h$, with $N_h = n-1$ and $h = 1/n$ (see Fig. 1). Inserting

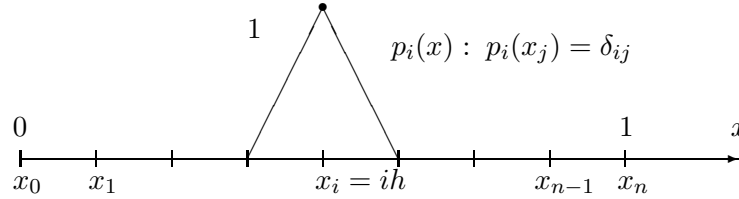


Figure 1: Piecewise linear ansatz functions (linear elements).

the ansatz (5) into the variational formulation (4) and testing with all basis functions $v = p_i$ ($i = \overline{1, n-1}$), we obtain the finite element equations

$$\sum_{j=1}^{n-1} u_j \int_0^1 p_j'(x)p_i'(x)dx = \int_0^1 f(x)p_i(x)dx, \quad \forall i = \overline{1, n-1} \quad (6)$$

that are equivalent to a system of the form (1) with the tridiagonal stiffness matrix

$$K_h = K_h^{FE} = \frac{1}{h} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}_{N_h \times N_h}, \quad (7)$$

the vector $\underline{u}_h = [u_j]_{j=\overline{1, n-1}} \in \mathbb{R}^{N_h}$ of the unknown nodal values and the load vector $\underline{f}_h = [f_i]_{i=\overline{1, n-1}} \in \mathbb{R}^{N_h}$, where

$$f_i = \int_0^1 f(x)p_i(x)dx, \quad i = \overline{1, N_h}, \quad N_h = n-1. \quad (8)$$

We note that the standard finite difference discretization where the second-order derivative is replaced by the second difference quotient results in a similar system with the system matrix $K_h^{FD} = h^{-1}K_h^{FE}$ and the coefficients $f_i = f(x_i)$ of the right-hand side \underline{f}_h .

Now we solve system (1) with the system matrix (7) and the right-hand side (8) by means of the ω -Jacobi method

$$D \frac{\underline{u}^{k+1} - \underline{u}^k}{\omega} + K \underline{u}^k = \underline{f}, \quad k = 0, 1, \dots, \quad (9)$$

with $D = \text{diag}(K) = [2/h \delta_{ij}]_{i,j=1,n-1}$ and the given initial guess $\underline{u}^0 \in \mathbb{R}^{N=n-1}$. Here and in the following we omit the discretization subscribed h in the matrix and vector notations. For $\omega = 1$, the ω -Jacobi method coincides with the classical Jacobi iteration. Later on we will use the relaxation parameter ω for controlling the smoothing properties of the ω -Jacobi iteration. The iteration error

$$\underline{z}^k = \underline{u} - \underline{u}^k \quad (10)$$

obviously satisfies the error iteration scheme

$$\underline{z}^{k+1} = (I - \omega \frac{h}{2} K) \underline{z}^k = S(\omega) \underline{z}^k = S^{k+1}(\omega) \underline{z}^0 \quad (11)$$

with the iteration operator (error transition operator) $S(\omega) = I - \omega D^{-1} K$. In order to make a Fourier analysis of the propagation of the iteration error, we need the eigenvalues and the eigenvectors of the stiffness matrix K . It is easy to verify that the eigenvalues λ_l and the eigenvectors $\underline{\varphi}_l$ of the stiffness matrix K are given by the formulas

$$\lambda_l = \frac{4}{h} \sin^2 \left(\frac{l\pi}{2n} \right) \text{ and } \underline{\varphi}_l = \left[\sqrt{2} \sin \left(l\pi \frac{i}{n} \right) \right]_{i=1, \dots, n-1}, \quad (12)$$

respectively, with $l = 1, 2, \dots, n-1$. The scaling of the eigenvectors $\{\underline{\varphi}_l\}$ is chosen in such a way that they are orthonormal with respect to the discrete L_2 -scalar product, i.e.

$$(\underline{\varphi}_l, \underline{\varphi}_j)_h = \sum_{i=1}^{n-1} h \sqrt{2} \sin \left(l\pi \frac{i}{n} \right) \sqrt{2} \sin \left(j\pi \frac{i}{n} \right) = \delta_{lj}. \quad (13)$$

The eigenvalues λ_l obviously satisfy the inequalities

$$8h < \lambda_1 = \frac{4}{h} \sin^2 \frac{\pi}{2n} < \lambda_2 < \dots < \lambda_{n-1} = \frac{4}{h} \sin^2 \frac{n-1}{n} \frac{\pi}{2} < \frac{4}{h}, \quad (14)$$

from which we obtain the spectral condition number estimate

$$\kappa(K) := \frac{\lambda_{n-1}}{\lambda_1} = \frac{\sin^2 \frac{n-1}{n} \frac{\pi}{2}}{\sin^2 \frac{\pi}{2n}} \leq \frac{1}{2} h^{-2}, \quad (15)$$

i.e. the spectral condition number behaves like $O(h^{-2})$ as mentioned above. Inserting the Fourier decomposition of the initial iteration error

$$\underline{z}^0 = \sum_{l=1}^{n-1} \alpha_l \underline{\varphi}_l \quad (16)$$

into the error iteration scheme (11), we obtain the Fourier decomposition

$$\underline{z}^{k+1} = (I - \omega \frac{h}{2} K)^{k+1} \underline{z}^0 = \sum_{l=1}^{n-1} \alpha_l (1 - \omega \frac{h}{2} \lambda_l)^{k+1} \underline{\varphi}_l \quad (17)$$

of the iteration error after $k + 1$ iterations, where the coefficients

$$\alpha_l = (\underline{z}^0, \underline{\varphi}_l)_h = \sum_{i=1}^{n-1} h z_i^0 \underline{\varphi}_l(i) = \sqrt{2} \sum_{i=1}^{n-1} h z_i^0 \sin l\pi \frac{i}{h} \quad (18)$$

denote the Fourier coefficients of the initial iteration error \underline{z}^0 . Using Parseval's equality

$$\|\underline{z}^0\|_h^2 = (\underline{z}^0, \underline{z}^0)_h = \sum_{l=1}^{n-1} \alpha_l^2, \quad (19)$$

we finally arrive at the iteration error estimate

$$\begin{aligned} \|\underline{z}^{k+1}\|_h^2 &= (\underline{z}^{k+1}, \underline{z}^{k+1})_h = \sum_{l=1}^{n-1} \alpha_l^2 (1 - \omega \frac{h}{2} \lambda_l)^{2(k+1)} \\ &\leq \left[\max_{l=1, n-1} |1 - \omega \frac{h}{2} \lambda_l| \right]^{2(k+1)} \sum_{l=1}^{n-1} \alpha_l^2 \\ &= \left[\max_{l=1, n-1} |1 - \omega \frac{h}{2} \lambda_l| \right]^{2(k+1)} \|\underline{z}^0\|_h^2. \end{aligned} \quad (20)$$

Therefore, the spectral radius

$$\rho(h, \omega) = \rho(S) = \max_{l=1, n-1} \left| 1 - \omega \frac{h}{2} \lambda_l \right| \quad (21)$$

of the iteration operator S describes the error damping per iteration. The error damping factor $\rho(h, \omega)$ is often called *convergence factor*, or *convergence rate*. The analysis of the convergence factor $\rho(h, \omega)$, which is illustrated in Fig. 2, shows us that the optimal (minimal) rate

$$\rho_{opt} = \rho(h, 1) = 1 - 2 \sin^2 \frac{\pi}{2n} = \cos \pi h = 1 - O(h^2) \approx 1 - \frac{\pi^2}{2} h^2 \quad (22)$$

is attained at $\omega = \omega_{opt} := 1$, i.e. for the classical Jacobi method. The ω -Jacobi method (9) converges if and only if $\omega \in (0, 4/(h\lambda_{n-1}))$, where the upper bound $4/(h\lambda_{n-1})$ tends to 1. From Fig. 2 we can already observe an important property of the ω -Jacobi method, namely a fast reduction of the high frequencies can be obtained via underrelaxation, e.g. for $\omega \approx 1/2$. The analysis of the reduction factors

$$\left(1 - \omega \frac{h}{2} \lambda_l \right) = 1 - 2\omega \sin^2 \frac{l\pi}{2n} = 1 - \omega(1 - \cos(l\pi h)) \quad (23)$$

for different values of ω in dependence of $l = \overline{1, n-1}$ (frequencies) shows that the best smoothing is attained at $\omega \approx 2/3$ (see Fig. 3). Indeed, let us define the *smoothing factor*

$$\begin{aligned} \mu(h, \omega) = \mu(S) &= \max_{\frac{n}{2} \leq l \leq n-1} \left| 1 - \omega \frac{h}{2} \lambda_l \right| \\ &= \max\{|1 - \omega|, |1 - \omega(1 + \cos \pi h)|\} \end{aligned} \quad (24)$$

and the *asymptotic smoothing factor*

$$\mu^*(\omega) = \sup_{h \leq 1/4} \mu(h, \omega) = \max\{|1 - \omega|, |1 - 2\omega|\} \quad (25)$$

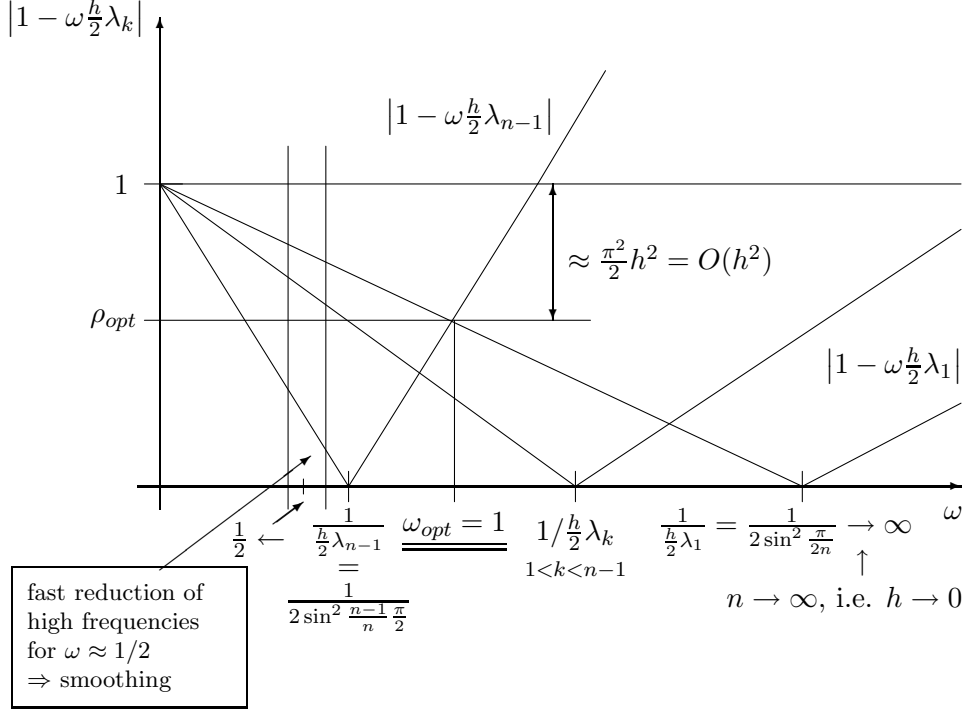


Figure 2: Reduction factors.

as measures for the reduction of the high frequencies in the error belonging to the indices $l = n/2, \dots, n-1$. Here and below we assume for simplicity that n is even and greater or equal than 4. The optimal (minimal) values

$$\mu(h, \omega_h^*) = \min_{0 \leq \omega \leq 1} \mu(h, \omega) = \frac{\cos \pi h}{2 + \cos \pi h} \quad (26)$$

and

$$\mu^*(\omega^*) = \min_{0 \leq \omega \leq 1} \mu^*(\omega) = \frac{1}{3} \quad (27)$$

are attained at $\omega_h^* = 2/(2 + \cos \pi h)$ and $\omega^* = 2/3$, respectively. Therefore, the underrelaxed ω -Jacobi method is able to reduce the high frequency part in the iteration error dramatically after a few iterations only whereas there is almost no reduction of the very low frequencies in the iteration error. For instance, in the case of $h = 10^{-3}$ and $\omega = 2/3$, the high frequency part is already reduced by the factor $0.7 \cdot 10^{-4}$ after 10 iterations whereas the overall error (i.e. the low frequency part) is only reduced by the factor 0.999967 ! However, as illustrated in the left-hand part of Fig. 4, the low frequencies ($l = 1, \dots, (n/2) - 1$) in the iteration error can be well approximated on the coarser grid with the grid size $2h$, whereas the high frequencies ($l = n/2, \dots, n-1$) in the iteration error cannot be represented on the coarser grid at all (see the right-hand part of Fig. 4).

The combination of the *smoothing property* of the underrelaxed ω -Jacobi method with the *approximation property* of the coarser grid allows us to kill the high frequency part as well as the low frequency part very efficiently. The successive use of *smoothing* and *coarse grid correction* results in a twogrid iteration process called *twogrid method* (TGM) that is

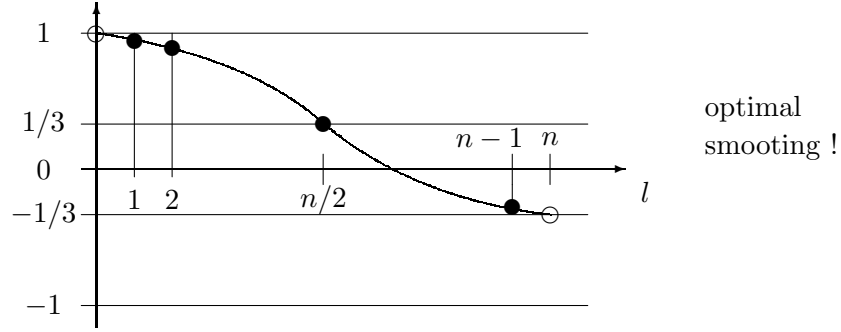
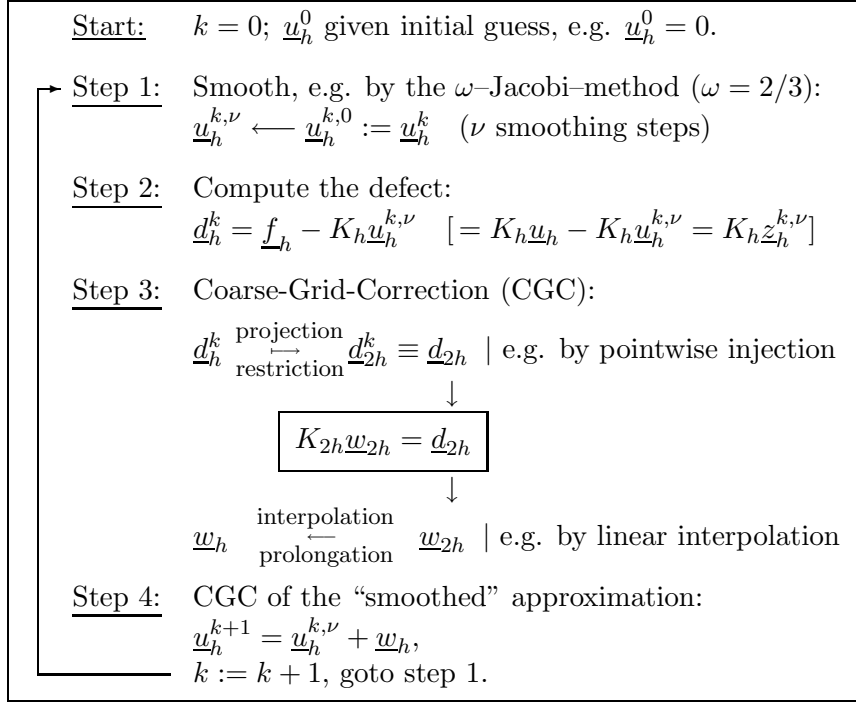


Figure 3: $\omega = \frac{2}{3}$: $1 - \frac{2}{3} \frac{h}{2} \lambda_l = 1 - \frac{4}{3} \sin^2 \frac{l\pi}{2n} = \frac{1}{3} + \frac{2}{3} \cos \frac{l\pi}{n}$.

sketched below and that is algorithmically described in §3.1 more precisely:



In the twogrid method, the coarse grid defect system

$$\boxed{K_{2h} \underline{w}_{2h} = \underline{d}_{2h}} \quad (28)$$

is solved by some direct method. However, in many practical applications, the direct solution of the coarse grid systems (28) is too expensive because the coarse grid systems are simply too large. In this case, one can repeat steps 1 - 4 for the coarse grid systems on the grid “2h” γ times with the initial guess $\underline{w}_{2h}^0 = 0$, where γ is usually equal to 1 (V-cycle) or to 2 (W-cycle). Now we have to solve coarse grid systems on the grid “4h”. Again, repeating the steps 1 - 4 γ times, and taking into account coarser and coarser grids up to some coarsest grid where the direct solver can be used efficiently, we end up with the *multigrid method* (MGM) that is described in §3.2 in detail.

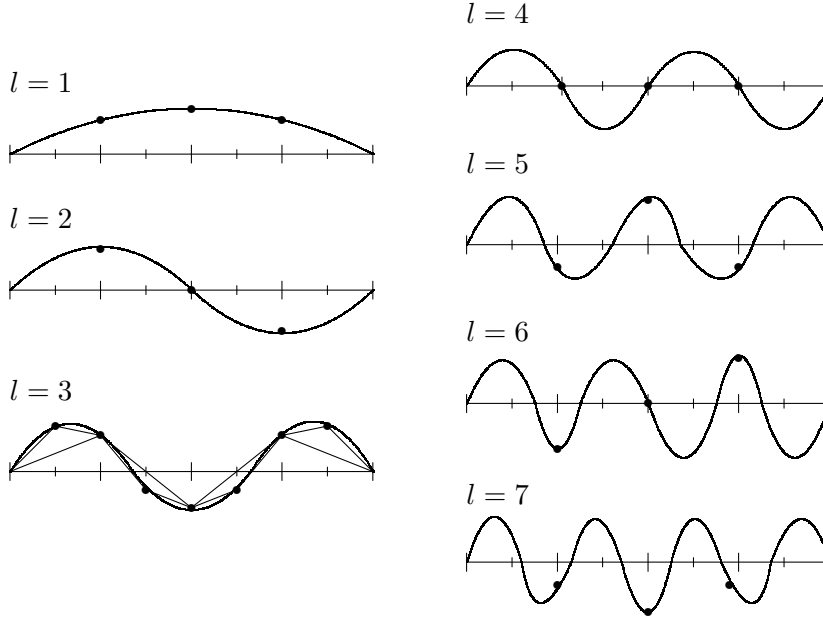


Figure 4: Error reduction via coarse grid correction ($n=8$).

2 History - Situation - Perspectives

We observed in §1 that the combination of the following algorithmic ingredients should result in efficient multigrid algorithms:

1. smoothing of rough error parts via classical iterative methods,
2. approximation of smooth errors on a coarser grid,
3. recurrent call of 1 and 2 on a sequence of coarser and coarser grids,
4. nested iteration for producing good initial guesses.

In his pioneering paper [28] from 1961, R.P. Fedorenko proposed the first twogrid method for solving the finite difference equations arising from the five-point approximation of the Poisson equation in a rectangular domain. Already the first numerical results presented in this paper showed the great potential of this methods. In his second paper [29] from 1964, he gave a first convergence analysis and proposed to combine 1 and 2 with 3 that results in a complete multigrid cycle. In 1966, N.S. Bachwalov provided the first rigorous convergence proof on the basis of the sum splitting technique (see also §3.1) and showed that the nested iteration technique with the multigrid method as nested iteration process allows us to produce approximate solutions that differ from the exact solution of the boundary value problem in the order of the discretization error with optimal complexity [4]. In the 70ies, these techniques were generalized to variational finite difference equations and general finite element equations by G.P. Astrachanzenov [2] and by V.G. Korneev [53], respectively.

Also in the 70ies, A. Brandt in Israel [16, 17] and W. Hackbusch in Germany [37] reinvented the multigrid method independently of the Russian school, generalized it to new classes of problems, and developed the theory. In particular, W. Hackbusch proposed the so-called product splitting that reduces the twogrid rate estimate to the proof of the approximation property and the smoothing property [39]. At the end of the 70ies and at the beginning of the 80ies there was a real boom in research and in publication of research results on multigrid methods. The proceedings [39] of the first European Conference on Multigrid Methods, which was held at Cologne-Porz in 1981, reflect the state-of-the-art in the research on multigrid methods, give the first interesting applications and present a multigrid bibliography with more than 200 entries. This conference at Cologne-Porz created the series of the European Conferences on Multigrid Methods the proceedings of which give a good reflection of the research results during the last 20 years [39, 40, 41, 43, 42, 26]. The first monograph on multigrid methods was published by W. Hackbusch in 1985 [38]. In this monograph, the reader finds a complete overview of the most important results on multigrid and related methods obtained till 1985.

The theoretical convergence analysis is one thing. An efficient implementation of the multigrid method for certain classes of problems is another game. Let us first have a look at the practical convergence behavior of the basic multigrid cycles (V-cycle, generalized V-cycle, W-cycle) for solving the discrete Poisson equation that was already studied by R.P. Fedorenko in his very first paper [28]. Here we consider the Poisson equation $-\Delta u = 1$ in the unit square $\Omega := (0, 1) \times (0, 1)$ under homogeneous Dirichlet conditions $u = 0$ on the boundary $\partial\Omega$ of the unit square, and use the simplest finite element discretization with linear triangular elements on a uniform $n \times n$ triangular grid. This finite element discretization results in a linear system of the form (1) with $N_h = (n - 1)^2$ finite element equations and the discretization size $h = 1/(n - 1)$. In this case, the finite element stiffness matrix coincides with the well-known five-point finite difference approximation of the Laplacian up to some scaling factor h^{-1} . Table 1 shows the number $I(\varepsilon)$ of iteration that are needed in order to reduce the initial defect $\underline{d}_h^0 := \underline{f}_h - K_h \underline{u}_h^0 = \underline{f}_h$ (zero initial guess) by the factor $\varepsilon := 10^{-4}$ in the Euclidean norm, the convergence rate η and the CPU time in seconds for the basic multigrid cycles (V-cycle, generalized V-cycle¹, W-cycle). The cycles are built by the following components: ν_1 lexicographically forward Gauss-Seidel pre-smoothing sweeps, ν_2 lexicographically backward Gauss-Seidel post-smoothing sweeps, linear finite element interpolation as prolongation, and the transposed operation as restriction. In the upper part of Table 1 we study the dependency of the multigrid convergence on the levels for the case $(\nu_1, \nu_2) = (1, 0)$ (only one pre-smoothing step!). The last two rows show the results for the cases $(\nu_1, \nu_2) = (1, 1)$ and $(\nu_1, \nu_2) = (2, 1)$ on the finest grid ($l = 12$, corresponding to 16.744.464 unknowns) that we considered. The calculations were carried out by the multigrid C code **MGTEST** on a PENTIUM IV PC with 800 MHz clock rate. The multigrid test code **MGTEST** can be downloaded.²

The classical convergence analysis techniques mentioned above do not cover the practical important case of the V-cycle. Another drawback of the classical techniques consists in the assumption of “sufficiently many” smoothing steps. This contradicts the practical experience which is clearly reflected in Table 1. The V-cycle converges almost uniformly even if only 1 smoothing sweep is used per cycle !! Therefore, new convergence theories that cover these interesting cases at least for some classes of problems under additional assumptions have

¹See §3.2 for the definition of the generalized V-cycle.

²<http://www.numa.uni-linz.ac.at/Staff/haase/Lectures/Montreal01/index.html>

l	n	V-cycle			gener. V-cycle			W-cycle		
		I	η	CPU [sec]	I	η	CPU [sec]	I	η	CPU [sec]
6	64	15	0.521	0.01	11	0.429	0.01	10	0.365	0.01
7	128	16	0.538	0.08	12	0.431	0.07	10	0.366	0.06
8	256	16	0.559	0.39	12	0.438	0.40	10	0.366	0.37
9	512	17	0.571	1.73	12	0.445	1.75	10	0.366	1.64
10	1024	18	0.582	7.44	12	0.454	5.73	10	0.367	5.52
11	2048	18	0.599	29.76	12	0.463	23.68	10	0.367	23.22
12	4096	19	0.607	130.89	13	0.465	107.66	10	0.367	105.36
12	4096	12	0.285	73.74	5	0.135	54.73	5	0.135	65.74
12	4096	6	0.210	65.08	4	0.062	54.17	4	0.062	62.98

Table 1: Convergence behavior of different multigrid cycles.

been developed by D. Braess and W. Hackbusch [12] and others [14, 15, 74]. A successful commercial use of multigrid methods for solving hard practical problems depends not only on the level-independent convergence behavior that is typical for multigrid methods, but also on their robustness against bad parameters (e.g. geometrical parameters, large coefficient jumps, singularities, anisotropies etc.) arising often in many practical applications. The construction of efficient and at the same time robust multigrid algorithms for problems with bad parameters is a hot topic in the current research activities (see, e.g., [65]).

The multigrid boom at the beginning of the 80ies initiated many interesting research directions and many new applications of multigrid and related methods. Algebraic multigrid (AMG) methods are one of the most important issues at least from a practical point of view. In contrast to the geometrical multigrid method, its algebraic version recovers algebraically the ingredients for a complete multigrid algorithm from the fine grid data only (see §4). The first serious approach to AMG was made in 1982 by A. Brandt, S. McCormick and J.W. Ruge [21]. This approach was afterwards improved in several papers [20, 18, 63, 64]. There was a revival of the research activities in AMG methods and their analysis in the mid-nineties [19, 71, 9, 72, 30, 52]. Since then a lot of new ideas have been published (see, e.g., [23, 47, 44] and the review paper [68] by K. Stüben). There is an urgent need in efficient implementation of robust AMG methods that can be used in commercial codes. It is much easier to use an AMG code in an existing software package as solver than redesign this package in such a way that hierarchical data structures for implementing geometrical multigrid solvers are available. In §4, we give an introduction to AMG and discuss the issues connected with an efficient implementation of some AMG methods.

Multigrid methods can exploit new computer architectures, especially parallel computers with distributed memory very well (see, e.g. [27]). In principle, there are two approaches to the parallelization of multigrid methods. Firstly, multigrid methods can be used as local solvers in domain decomposition preconditioners (see e.g. [35]). Secondly, multigrid methods can be directly parallelized with slight modifications of the multigrid algorithm on the basis of data partitioning techniques [6, 31, 34, 54, 45]. In §5, we will discuss the direct approach to the parallelization of multigrid methods via some data partitioning technique

that is borrowed from the nonoverlapping domain decomposition method. Parallel multigrid methods implemented on professional parallel computers will allow us to solve really large-scale problems. Teraflop parallel computers such as the ASCII-white can handle 10^9 complexity problems in seconds. On the other side, parallel multigrid application software will also be superior to conventional software on low cost PC clusters due to the asymptotical optimality and efficiency of parallel multigrid algorithms.

During the last 20 years the development of multigrid software has been mainly pushed by research groups working at universities or at similar research institutions. R. Bank was certainly one of the pioneers in developing general purpose finite element multigrid codes. His code PLTMG was used by many researchers [5]. Similar research codes have been developed for different classes of applications by other groups too. Here we mention only the parallel multigrid package UG that is now widely used in research as well as in commercial applications [7] and our parallel geometrical and algebraic multigrid packages FEPP [55] and PEBBELS [60].

Since the pioneered monograph [38] by W. Hackbusch, many further books have been published wherein the reader can find interesting additional information on the theory and application of multigrid methods in survey form, e.g. the books by W.L. Briggs, V.E. Henson and S.F. McCormick [24], D. Braess [10], J.H. Bramble [13], S.C. Brenner and L.R. Scott [22], S.F. McCormick [58], S. Reitzinger [60], U. Rüde [62], V.V. Shaidurov [66], S. Vandewalle [70], P. Wesseling [73], U. Trottenberg, C. Oosterlee and A. Schüller [69] with a contribution by K. Stüben [67]. Most recent information about publications on multigrid method, related conferences, multigrid software etc. can be found on the multigrid home page <http://www.mgnet.org> in the internet.

3 Multigrid Algorithms and their Analysis

3.1 The Twogrid Method

At the end of §1, we sketched our first twogrid algorithm for solving system (1). Algorithm 1 now gives a complete algorithmic description of one step of a twogrid method for solving (1), where the subscripts $h = h_q$ and $H = h_{q-1}$ indicate the quantities belonging to the fine grid and the coarse grid, respectively (c.f. also §3.2). Let us here emphasize that the use of different fonts for the vectors ($\underline{u}_h, \underline{v}_h, \dots$ and $\underline{f}_h, \underline{d}_h, \dots$ for the so-called accumulated types and distributed types, respectively) and different arrows in the assignments (the arrows \Leftarrow and \leftarrow indicate assignments with and without communication, respectively) is due to the parallelization of Algorithm 1 and is explained in §5. The same is true for the remaining algorithms in this section. For the serial version of algorithms discussed in this section, this differentiation is not relevant! We note that the standard coarsening is connected with the doubling of the discretization parameter, i.e. $h_{q-1} = 2h_q$. We assume that the smoothing procedures $G_h^{pre/post}(\cdot)$ are some kind of regular, affine-linear fixed point iteration that can be rewritten in the form

$$G_h(\underline{u}_h) := S_h \underline{u}_h + T_h \underline{f}_h, \quad (29)$$

where S_h and T_h denote the iteration operator and the affine-linear part, respectively. For consistency reasons, the affine-linear part must have the form $T_h = (I_h - S_h)K_h^{-1}$, where I_h again denotes the identity. Primarily, we think of the underrelaxed ω -Jacobi method,

Algorithm 1 Twogrid Method: $\underline{u}_h \leftarrow \text{TGM}(\mathbf{K}_h, \underline{u}_h, \underline{f}_h, h)$

```

for  $i = 1$  step 1 until  $\nu_1$  do
     $\underline{u}_h \leftarrow \mathbf{G}_h^{pre}(\underline{u}_h)$  {presmoothing}
end for
 $\underline{d}_h \leftarrow \underline{f}_h - \mathbf{K}_h \cdot \underline{u}_h$  {defect calculation}
 $\underline{d}_H \leftarrow I_h^H \cdot \underline{d}_h$  {restriction}
 $\underline{w}_H \leftarrow \mathbf{K}_H^{-1} \cdot \underline{d}_H$  {direct coarse grid solver}
 $\underline{w}_h \leftarrow I_H^h \cdot \underline{w}_H$  {prolongation}
 $\underline{u}_h \leftarrow \underline{u}_h + \underline{w}_h$  {coarse grid correction}
for  $i = 1$  step 1 until  $\nu_2$  do
     $\underline{u}_h \leftarrow \mathbf{G}_h^{post}(\underline{u}_h)$  {postsmoothing}
end for

```

the Gauss-Seidel method, or block-versions of these relaxation methods as smoothers. The numbers ν_1 and ν_2 of pre- and post-smoothing sweeps are usually small, e.g., for standard problems (Poisson-like problems), $(\nu_1, \nu_2) = (1, 0)$, $(1, 1)$, or $(2, 1)$ are quite sufficient (c.f. Table 1 in §2). The restriction operator $I_h^H : R^{N_h} \rightarrow R^{N_H}$ maps a fine grid vector onto some coarse grid vector, whereas the prolongation operator $I_H^h : R^{N_H} \rightarrow R^{N_h}$ maps a coarse grid vector onto some fine grid vector. In the standard FE case where the prolongation operator is naturally defined by the FE interpolation and the restriction operator is simply the transposed prolongation operator, i.e. $I_h^H = (I_H^h)^T$, the coarse grid stiffness matrix \mathbf{K}_H fulfills the so-called Galerkin relation

$$\mathbf{K}_H = I_h^H \mathbf{K}_h I_H^h. \quad (30)$$

On the other hand, the Galerkin relation (30) can be used to derive the coarse grid matrix \mathbf{K}_H from the fine grid stiffness matrix \mathbf{K}_h . By the way, the Galerkin projection (30) is the standard technique for deriving the coarse grid matrices in the Algebraic Multigrid Method discussed in §4.

It is easy to see from Algorithm 1 that the twogrid method itself can be rewritten in the form of an affine-linear fixed point iteration

$$\underline{u}_h^{k+1} = M_h^H \underline{u}_h^k + A_h^H \underline{f}_h, \quad k = 0, 1, \dots, \quad (31)$$

with the twogrid iteration operator M_h^H and its affine-linear part

$$A_h^H = (I_h - M_h^H) \mathbf{K}_h^{-1}, \quad (32)$$

where the superscript k again denotes the iteration index. Setting \underline{f}_h formally to 0, then we can directly derive the representation

$$M_h^H = (S_h^{post})^{\nu_2} C_h^H (S_h^{pre})^{\nu_1}, \quad (33)$$

of the twogrid iteration operator M_h^H from Algorithm 1, the representation (29) of the smoothing procedure and the twogrid iteration scheme (31), with the coarse-grid-correction operator

$$C_h^H = I_h - I_H^h \mathbf{K}_H^{-1} I_h^H \mathbf{K}_h. \quad (34)$$

The twogrid iteration operator M_h^H transforms the old iteration error $\underline{z}_h^k := \underline{u}_h - \underline{u}_h^k$ into the new iteration error \underline{z}_h^{k+1} , i.e.

$$\underline{z}_h^{k+1} := \underline{u}_h - \underline{u}_h^{k+1} = M_h^H \underline{z}_h^k. \quad (35)$$

It is well known from the theory of iterative methods that the twogrid iteration converges if and only if the spectral radius $\rho(M_h^H)$ of the twogrid iteration operator M_h^H is strictly smaller than 1. In §1, we observed that the spectral radius of the ω -Jacobi method tends to 1 if h tends to 0. This behavior is typical for all classical iterative methods. In contrast to this, the spectral radius $\rho(M_h^H)$ of the twogrid iteration operator M_h^H remains well separated from 1 if h tends to 0. In practice, the people are interested in convergence rate estimates with respect to some norm $\|\cdot\|$ rather than in spectral radius estimates. From (35), we immediately derive the norm estimate

$$\|\underline{z}_h^{k+1}\| \leq \|M_h^H\| \|\underline{z}_h^k\|, \quad (36)$$

where we assume that the operator (matrix) norm corresponds to the vector norm in which we are interested. The straightforward estimation

$$\|M_h^H\| \leq \|S_h^{post}\|^{\nu_2} \|C_h^H\| \|S_h^{pre}\|^{\nu_1} \quad (37)$$

of the twogrid iteration operator M_h^H fails because $\|S_h^{pre/post}\| = 1 - O(h^2)$ (c.f. ω -Jacobi smoother in §1) and $\|C_h^H\| \geq 1$ (exercise !). In order to obtain an h -independent bound less than 1, one has to take into account our observation from §1, namely that the smoothers $S_h^{pre/post}$ kill the high frequency part in the iteration error, whereas the coarse-grid-correction C_h^H reduces only the low frequency part.

Let us briefly discuss two classical proof techniques that make heavily use of this observation. It is enough to consider the presmoothing case $\nu_2 = 0$ only. Indeed, if K_H satisfies the Galerkin relation (30), then the coarse-grid-correction C_h^H is a projection, i.e. $C_h^H = C_h^H \cdot C_h^H$. Therefore, the estimation of $\|M_h^H\|$ can obviously be reduced to separate estimations of the presmoothing and the postsmoothing cases.

The *first technique* (sum splitting) was mainly developed by the Russian school [4, 2, 53, 56]. Inserting the identity I_h in form of the sum $I_h = P_h^{low} \oplus P_h^{high}$ of two projections into M_h^H between C_h^H and S_h^{pre} , we obtain the estimate

$$\|M_h^H\| \leq \|C_h^H P_h^{low}\| \|S_h^{pre}\|^{\nu_1} + \|C_h^H\| \|P_h^{high} (S_h^{pre})^{\nu_1}\| \quad (38)$$

where P_h^{low} and P_h^{high} are the projections onto the subspaces spanned by the low frequency eigenvectors and high frequency eigenvectors, respectively. For sufficiently many presmoothing steps ν_1 and under some additional regularity assumptions, the right-hand side of estimate (38) can be bounded by some $\sigma_* < 1$ that is independent on h . This is due to the *approximation property* that allows us to make the term $\|C_h^H P_h^{low}\|$ small and to the *smoothing property* that allows us to make the term $\|P_h^{high} (S_h^{pre})^{\nu_1}\|$ arbitrary small for sufficiently many presmoothing steps ν_1 .

The *second technique* (product splitting) was proposed by W. Hackbusch [39, 38]. Now the identity I_h is inserted into M_h^H between C_h^H and S_h^{pre} in form of the product $I_h = K_h^{-1} K_h$. If we are able to show the so-called *approximation property*

$$\|C_h^H K_h^{-1}\| \leq c h^\delta \quad (39)$$

and the so-called *smoothing property*

$$\|K_h(S_h^{pre})^{\nu_1}\| \leq \eta(\nu_1) h^{-\delta}, \quad (40)$$

then we immediately arrive at the rate estimate

$$\|M_h^H\| \leq \|C_h^H K_h^{-1}\| \|K_h(S_h^{pre})^{\nu_1}\| \leq c \eta(\nu_1) =: \sigma_*, \quad (41)$$

with positive, h -independent constants c and δ and with some non-negative function $\eta(\nu_1)$ that tends to 0 for $\nu_1 \rightarrow \infty$. Therefore, the bound $\sigma_* = c \eta(\nu_1)$ in (41) can again be made less than 1 for sufficiently many presmoothing steps ν_1 . In order to show the approximation property (39), we usually need some regularity results for the boundary value problem under consideration. In [11], this analysis is extended to nonconforming FE discretization. The drawback of these classical proof techniques consists in the fact that sufficiently many smoothing steps are required for convergence. This strong requirement contradicts the practical experience where often one smoothing step gives satisfactory results (c.f. also §2). Therefore, new proof techniques are needed. Such techniques were developed in the 80-ies and 90-ies [12, 38, 14, 15, 13, 74].

3.2 The Multigrid Method

The direct solution of the coarse grid systems in the twogrid Algorithm 1 is too expensive in many practical applications. So, it is reasonable to apply the same technique to the coarse grid systems taking into account coarser and coarser grids (c.f. §1). The right-hand side of the coarse grid systems is some residual (defect). Thus, the initial guess for the iterative solution of the coarse grid systems should be 0. One step of a complete *multigrid method* (MGM), or more precisely, of an l -grid method, is described by Algorithm 2, where the subscript q indicates all quantities belonging to the grid level characterized by the discretization parameter h_q and the level index q is running from the coarsest grid $q = 1$ to the finest grid $q = l$. As in the twogrid case, the K_q , $G_q^{pre/post}(\cdot)$, I_q^{q-1} and I_{q-1}^q denote the stiffness matrices, the smoothing operators, the restrictions and the prolongations, respectively. We refer to §3.1 and §5 for explanations of the meaning of different fonts for the vectors and of the use of different arrows. We recall that this differentiation is only aligned with the parallelization of the algorithms.

Of course, the procedure $\text{MGM}(K_q, \underline{u}_q, \underline{f}_q, q)$ is applied to the fine grid system (c.f. (1) with $h = h_l$)

$$K_l \underline{u}_l = \underline{f}_l, \quad (42)$$

in the solution of which we are interested. The iteration with procedure $\text{MGM}(\cdot)$ will be terminated if some accuracy test is attained. The algorithmic description of the multigrid solution of (42) is given in Algorithm 3.

The integers $\gamma(q)$ ($q = 2, \dots, l-1$) control the multigrid cycling regime. If $\gamma(q) = 1$ for all $q = 2, \dots, l-1$, then the cycling regime is called *V-cycle*. In the case that $\gamma(q) = 2$ for all $q = 2, \dots, l-1$, we call the cycling regime *W-cycle*. In Figure 5 we illustrate these cycles by so-called pictograms that explain also the names V-cycle and W-cycle. We speak about an *alternating cycle* if $\gamma(q)$ changes between 1 and 2 from one level to next level. If $\gamma(q) = 2$ on coarser grid and 1 on finer grids, than such a cycling regime is called *composed cycle*. Usually the numbers $\nu_1(q)$ and $\nu_2(q)$ of pre- and postsmoothing steps are small and constant for all

Algorithm 2 Multigrid Method: $\underline{u}_q \Leftarrow \text{MGM}(\mathbf{K}_q, \underline{u}_q, \underline{f}_q, q)$

```

if  $q == 1 := \text{COARSESTLEVEL}$  then
     $\underline{u}_q \Leftarrow \text{SOLVE}(\mathbf{K}_q \cdot \underline{u}_q = \underline{f}_q)$                                 {solution on the coarsest grid}
else
    for  $i = 1$  step 1 until  $\nu_1(q)$  do
         $\underline{u}_q \Leftarrow G_q^{\text{pre}}(\underline{u}_q)$                                 {pre-smoothing}
    end for
     $\underline{d}_q \Leftarrow \underline{f}_q - \mathbf{K}_q \cdot \underline{u}_q$                                 {defect calculation}
     $\underline{d}_{q-1} \Leftarrow I_q^{q-1} \cdot \underline{d}_q$                                 {restriction}
     $\underline{w}_{q-1} \Leftarrow 0$                                 {coarse grid initial guess}
    for  $j = 1$  step 1 until  $\gamma(q-1)$  do
         $\underline{w}_{q-1} \Leftarrow \text{MGM}(\mathbf{K}_{q-1}, \underline{w}_{q-1}, \underline{d}_{q-1}, q-1)$     {recurrent call}
    end for
     $\underline{w}_q \Leftarrow I_{q-1}^q \cdot \underline{w}_{q-1}$                                 {prolongation}
     $\underline{u}_q \Leftarrow \underline{u}_q + \underline{w}_q$                                 {coarse grid correction}
    for  $i = 1$  step 1 until  $\nu_2(q)$  do
         $\underline{u}_q \Leftarrow G_q^{\text{post}}(\underline{u}_q)$                                 {post-smoothing}
    end for
end if

```

Algorithm 3 Multigrid solver for the fine grid system $\mathbf{K}_l \underline{u}_l = \underline{f}_l$.

```

 $\underline{u}_l \Leftarrow \underline{u}_l^0$                                 {define the initial guess}
while  $\|\underline{f}_l - \mathbf{K}_l \underline{u}_l\| > \varepsilon \|\underline{f}_l - \mathbf{K}_l \underline{u}_l^0\|$  do
     $\underline{u}_l \Leftarrow \text{MGM}(\mathbf{K}_l, \underline{u}_l, \underline{f}_l, l)$                                 {multigrid iteration}
end while

```

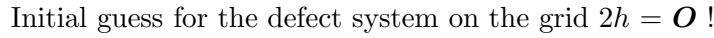
levels $q = 2, \dots, l$ (see §3.1). However, it may be reasonable to enlarge the numbers of pre- and postsmoothing steps on the coarser grids. If the numbers of pre- and postsmoothing steps are enlarged from one grid to the next coarser grid, e.g. $\nu_1(q) = 2\nu_1(q+1)$ and $\nu_2(q) = 2\nu_2(q+1)$ for all $q = 2, \dots, l-1$, in the V-cycle, then such a cycle is called *generalized V-cycle*. The system matrices \mathbf{K}_q on the coarser grids $q = 1, \dots, l-1$ can be also defined by Galerkin projection (30).

At the first glance, it seems that the multigrid algorithm requires much more memory M than the corresponding single grid method. But this is not true if the right coarsening strategy is used. Let us assume that the standard coarsening is used where the discretization parameter is doubled from one grid to the next coarser grid, i.e. $h_q = 2h_{q+1}$ for all $q = 2, \dots, l-1$. In this case, we can assume without loss of generality that the numbers of unknowns N_q fulfill the relations

$$N_q \leq N_{q+1}/\tau \quad \forall q = 1, \dots, l-1. \quad (43)$$

with the enlargement factor $\tau = 2^d$. Taking into account the sparsity of the stiffness matrices, we observe from Algorithm 2 that the memory M_q needed on level q is only proportional to the number N_q of unknowns on level q , i.e. there is a q -independent positive constant c such that

$$M_q \leq c N_q \quad \forall q = 1, \dots, l. \quad (44)$$



Inequalities (43) and (44) immediately yields the estimate

of the total memory M required by the multigrid method for solving (42) that gives the final estimates

for the 2D and 3D cases, respectively. Therefore, the multigrid method does not need much more memory than the corresponding single grid method.

In analogy to the twogrid method (31), the multigrid method described by Algorithm 3 can also be rewritten in the form of an affine-linear fixed point iteration

with the *multigrid iteration operator* M_l and the iteration index k provided that the smoothing procedures $G_q^{pre/post}(\cdot)$ itself are affine-linear fixed point iterations of the form

where S_q denotes the iteration operator of the smoothing procedure and q runs from 2 until l . Replacing the exact solution $\underline{w}_{l-1} = K_{l-1}^{-1} \underline{d}_{l-1}$ of the coarse grid system by the approximate solution

obtained by $\gamma(l-1)$ (l-1)-grid cycles with the zero initial guess $\underline{w}_{l-1}^0 = 0$, we arrive at the recurrent definition of the multigrid iteration operator M_l given in the following lemma.

Lemma 3.1 *The multigrid iteration operator M_l is defined by the recurrent relations*

$$M_q = (S_q^{post})^{\nu_2(q)} (I_q - I_{q-1}^q (I_{q-1} - M_{q-1}^{\gamma(q-1)}) K_{q-1}^{-1} I_q^{q-1} K_q) (S_q^{pre})^{\nu_1(q)} \quad (50)$$

where $q = 3, 4, \dots, l$. If the systems on the coarsest grid with the level index $q = 1$ are solved by a direct method (that is assumed throughout this paper), then M_2 is nothing else than the twogrid iteration operator M_2^1 defined by relations (33) - (34) with $h = h_2$ and $H = h_1$.

A closer look at (50) shows that M_q can be represented in the form

$$M_q = M_q^{q-1} + A_{q-1}^q M_{q-1}^{\gamma(q-1)} B_q^{q-1} \quad (51)$$

of a perturbed twogrid iteration operator M_q^{q-1} , where

$$A_{q-1}^q = (S_q^{post})^{\nu_2(q)} I_{q-1}^q \text{ and } B_q^{q-1} = K_{q-1}^{-1} I_q^{q-1} K_q (S_q^{pre})^{\nu_1(q)}. \quad (52)$$

The following lemma provides recurrent relations for the rates that are the basis for uniform rate estimates under appropriate cycling regimes.

Lemma 3.2 *Let us assume that*

$$\|M_q^{q-1}\| \leq \sigma_* \text{ and } \|A_{q-1}^q\| \cdot \|B_q^{q-1}\| \leq c \quad (53)$$

for all $q = 2, 3, \dots, l$ ($l = 3, 4, \dots$), with level-independent positive constants $\sigma_* < 1$ (twogrid rate, see §3.1) and c . Then

$$\|M_l\| \leq \eta_l, \quad (54)$$

where η_l is recurrently defined by the relation

$$\eta_q = \sigma_* + c(\eta_{q-1})^{\gamma(q-1)}, \quad q = 2, 3, \dots, l \quad (l = 3, 4, \dots), \quad (55)$$

with $\eta_2 = \sigma_*$.

The proof of Lemma 3.2 follows immediately from the assumptions (53) and simple norm estimates applied to the representation (51) of the q -grid iteration operators. \square

If the (h_q, h_{q-1}) -TGM converges with a sufficiently small rate σ_* for all q and if the cycling regime is appropriately chosen, then the recurrence (55) of Lemma 3.2 will yield to level-independent convergence rate estimates for the MGM (47). Indeed, in the case of the W-cycle, i.e. $\gamma(q) = \gamma = 2$ for all q , the recurrence (55) gives the uniform rate estimate

$$\|M_l\| \leq \eta_* := \frac{1}{2c} (1 - \sqrt{1 - 4c\sigma_*}) \leq 2\sigma_* \quad (56)$$

provided that $\sigma_* \leq 1/(4c)$. Similar rate estimates can be derived for the alternating cycle, the composed cycle and for the generalized V-cycle (exercise). However, in the case of the pure V-cycle, i.e. $\gamma(q) = \gamma = 1$ for all q , the fixed point equation $\eta = \sigma_* + c\eta$ corresponding to the recurrence (55) has a positive solution $\eta_* = \sigma_*/(1 - c)$ if and only if $c < 1$. Unfortunately, in many practical applications, $c \geq 1$. Therefore, this proof technique does not provide a rate estimate for the V-cycle even if the twogrid rate is small. This contradicts all practical experience where reasonable convergence rates have been observed even for the V-cycle with

only one smoothing step (see also Table 1 in §2). For some special classes of problems, this case was covered by new convergence theories [12, 38, 14, 15, 13, 74].

Let us finally count the arithmetical operations that are needed for one step of the multigrid iteration (47), i.e. for one call of the multigrid procedure $\text{MGM}(\mathbf{K}_l, \underline{u}_l, \underline{f}_l, l)$. For simplicity, we assume that $\gamma(q) = \gamma$ for all $q = 2, \dots, l-1$ and that the number of pre- and post-smoothing steps are equal $\nu = \nu_1 + \nu_2$ at all levels. Furthermore, we suppose that the number of unknowns is growing by the factor τ from level to level. More precisely, inequalities (43) are valid with some enlargement factor $\tau > 1$ (standard enlargement factor is $\tau = 2^d$). Now it is clear that at each level $q = 2, \dots, l$ the presmoothing, the defect calculation, the restriction, the prolongation, the correction and the postsmoothing need not more than $c_2 N_q$ arithmetical operations, where c_2 is some level-independent positive constant. Summing up over all levels, we obtain the following estimate for the number of arithmetical operations $Q(M_l)$ required by one call of the multigrid procedure at the finest level:

$$\begin{aligned} Q(M_l) &= c_2 N_l + \gamma \left\{ c_2 \frac{N_l}{\tau} + \gamma \left\{ c_2 \frac{N_l}{\tau^2} + \dots + \gamma \left\{ c_2 \frac{N_l}{\tau^{l-2}} + c_1 \frac{N_l}{\tau^{l-1}} \right\} \dots \right\} \right\} \\ &\leq \max \left\{ c_2, \frac{c_3}{\gamma} \right\} \sum_{j=0}^{l-1} \left(\frac{\gamma}{\tau} \right)^j N_l, \end{aligned} \quad (57)$$

where $c_1 N_1$ is the bound on the number of arithmetical operations needed on the coarsest grid. Calculating the sum for the possible choices of γ and τ , we arrive at the estimates

$$Q(M_l) \leq \begin{cases} O(N_l) & \text{for } 1 \leq \gamma < \tau, \\ O(N_l \ln(N_l)) & \text{for } \gamma = \tau, \\ O(N_l^{1+\ln(\beta)}) & \text{for } \gamma > \tau. \end{cases} \quad (58)$$

Summarizing the results on the convergence and on the operation count for multigrid methods, we can finally prove the following theorem:

Theorem 3.3 *Let us assume that*

$$1) \ 1 \leq \gamma < \tau \text{ and}$$

$$2) \ ||M_l| \leq \eta_* < 1,$$

where η_* is supposed to be independent of h_l . Then the multigrid method (47) algorithmically described by Algorithms 3 and 2 needs

$$Q(\varepsilon) = O(\ln(\varepsilon^{-1}) N_l) \quad (59)$$

arithmetical operations in order to reduce the initial error $\|\underline{u}_l - \underline{u}_l^0\|$ by the factor $\varepsilon := 10^{-s} \in (0, 1)$.

The proof follows from the fact that $Q(\varepsilon) = I(\varepsilon)Q(M_l)$ is nothing else than the product of the number of iterations $I(\varepsilon)$ and the costs of arithmetical operations per iteration step $Q(M_l)$. Relation (58) provides the estimate $Q(M_l)$ for $1 \leq \gamma < \tau$. The number of iterations $I(\varepsilon)$ results from the condition $\eta_*^{I(\varepsilon)} \leq \varepsilon$ that gives $I(\varepsilon) \geq \ln(\varepsilon^{-1}) / \ln(\eta_*^{-1})$. \square

Therefore, multigrid methods are asymptotically optimal with respect to the operation count (Theorem 3.3) and memory demand (estimates (46)). These properties make the multigrid methods so attractive for the solution of large-scale problems.

The multigrid iteration scheme (47) can be rewritten in the canonical form of a preconditioned Richardson iteration

$$C_l \frac{\underline{u}_l^{k+1} - \underline{u}_l^k}{\omega} + K_l \underline{u}_l^k = \underline{f}_l, \quad k = 0, 1, \dots, \quad (60)$$

with the relaxation parameter $\omega = 1$ and the so-called multigrid preconditioner

$$C_l = K_l(I_l - M_l)^{-1}. \quad (61)$$

On one hand, this observation, together with well-known convergence results for the preconditioned Richardson method, can be used for the convergence analysis of multigrid methods itself [14, 15]. On the other hand, we can accelerate the convergence of the basic multigrid algorithm by an appropriate choice of the relaxation parameter ω . In particular, if the stiffness matrix K_l and the multigrid preconditioner C_l are symmetric and positive definite (spd), then other acceleration techniques such as the gradient technique, the Chebyshev acceleration, or the Conjugate Gradient (CG) technique can be used. Since the CG method is the most popular acceleration technique for spd systems in practice, we will present a detailed description of the CG method with spd multigrid preconditioners of the form (61) in the next subsection.

3.3 The Multigrid Preconditioned CG-Method

In hard practical applications, geometrical and, especially, algebraic multigrid methods are commonly used to construct very efficient preconditioners C_l that are then used as preconditioner in another iteration process such as a Krylov subspace iteration. In this subsection we assume that the stiffness matrix K_l is spd and in that case, the Preconditioned Conjugate Gradient (PCG) method is certainly the most widely used basic iteration scheme [3, 59]. Algorithm 4 gives a detailed algorithmic description of the PCG method. We recall again that different notations for the vectors and different arrows are only relevant for the parallel version of Algorithm 4 (see also §5). Before discussing the construction of the preconditioner C_l via multigrid techniques in detail, we shall recall the well-known rigorous convergence estimate for the PCG method [3, 59]. In order to reduce the initial iteration error in the K_l -energy norm $\|\cdot\|_{K_l}^2 := (K_l \cdot, \cdot)$ by some factor $\varepsilon := 10^{-s} \in (0, 1)$, i.e.

$$\|\underline{u}_l - \underline{u}_l^{I(\varepsilon)}\|_{K_l} \leq \varepsilon \|\underline{u}_l - \underline{u}_l^0\|_{K_l}, \quad (62)$$

we need at most

$$I(\varepsilon) = \text{entire} \left(\frac{\ln(\varepsilon^{-1} + \sqrt{\varepsilon^{-2} - 1})}{\ln(\varrho^{-1})} + 1 \right) \quad (63)$$

PCG iterations provided that the preconditioner C_l is spd and spectrally equivalent to K_l , i.e. there exist positive constants γ_1 and γ_2 such that

$$\gamma_1 (C_l \underline{v}_l, \underline{v}_l) \leq (K_l \underline{v}_l, \underline{v}_l) \leq \gamma_2 (C_l \underline{v}_l, \underline{v}_l) \quad \forall \underline{v}_l \in R^{N_l}, \quad (64)$$

Algorithm 4 Multigrid preconditioned CG: $\underline{u}_l \leftarrow \text{MGPCG}(K_l, \underline{u}_l, \underline{f}_l)$

```

{Initialization step:}
 $\underline{u}_l \leftarrow \underline{u}_l^0$                                      {define the initial guess}
 $\underline{d}_l \leftarrow \underline{f}_l - K_l \cdot \underline{u}_l$                {defect calculation}
 $\underline{w}_l \leftarrow C_l^{-1} \cdot \underline{d}_l$                      {multigrid preconditioning}
 $\delta^0 \leftarrow (\underline{w}_l, \underline{d}_l)$ 
 $\delta \leftarrow \delta^0$ 
 $\underline{s}_l \leftarrow \underline{w}_l$                                      {search direction}
{Iteration:}
while  $\delta > \varepsilon^2 \delta^0$  do
   $\alpha \leftarrow \delta / (K_l \underline{s}_l, \underline{s}_l)$ 
   $\underline{u}_l \leftarrow \underline{u}_l + \alpha \underline{s}_l$                              {update of the iterate}
   $\underline{d}_l \leftarrow \underline{d}_l + \alpha K_l \underline{s}_l$                      {update of the defect}
   $\underline{w}_l \leftarrow C_l^{-1} \cdot \underline{d}_l$                      {multigrid preconditioning}
   $\hat{\delta} \leftarrow (\underline{w}_l, \underline{d}_l)$ 
   $\beta \leftarrow \hat{\delta} / \delta$ 
   $\delta \leftarrow \hat{\delta}$ 
   $\underline{s}_l \leftarrow \underline{w}_l + \beta \underline{s}_l$                              {update of the search direction}
end while

```

where $\varrho = (1 - \sqrt{\gamma_1/\gamma_2})/(1 + \sqrt{\gamma_1/\gamma_2})$. Of course, the accuracy test in the K_l -energy norm is not practicable because the solution \underline{u}_l of system (42) is unknown. We prefer to use the $K_l C_l^{-1} K_l$ -energy norm test

$$(\underline{w}_l^{I(\varepsilon)}, \underline{d}_l^{I(\varepsilon)}) = \|\underline{u}_l - \underline{u}_l^{I(\varepsilon)}\|_{K_l C_l^{-1} K_l}^2 \leq \varepsilon^2 \|\underline{u}_l - \underline{u}_l^0\|_{K_l C_l^{-1} K_l}^2 \quad (65)$$

rather than the defect norm test. Indeed, if C_l is close to K_l as it is the case for a multigrid preconditioner, then $K_l C_l^{-1} K_l$ is close to K_l as well.

The multigrid preconditioning step

$$C_l^{-1} \cdot \underline{d}_l := (I_l - (M_l)^j) K_l^{-1} \cdot \underline{d}_l \quad (66)$$

in Algorithm 4 means nothing else but the application of j (in practice: $j = 1$, or 2) multigrid cycles to the defect system

$$K_l \underline{w}_l = \underline{d}_l \quad (67)$$

with the zero initial guess $\underline{w}_l^0 = 0$.

The parameters and the components of the multigrid algorithm defining the multigrid preconditioning step (66) must be chosen in such a way that the multigrid preconditioner

$$C_l = K_l (I_l - (M_l)^j)^{-1} \quad (68)$$

is spd and satisfies the spectral equivalence inequalities (64) with spectral equivalence constants γ_1 and γ_2 such that the relative spectral condition number $\kappa(C_l^{-1} K_l) \leq \gamma_2/\gamma_1$ is as small as possible.

The next two lemmas provide precise conditions ensuring the symmetry of the multigrid preconditioner (68).

Lemma 3.4 *Let us assume that the multigrid iteration operator M_l defined by the recursion (50) is self-adjoint in the K_l -energy scalar product, i.e.*

$$(M_l \underline{u}_l, \underline{v}_l)_{K_l} = (\underline{u}_l, M_l \underline{v}_l)_{K_l} \quad \forall \underline{u}_l, \underline{v}_l \in R^{N_l}, \quad (69)$$

briefly, $M_l = M_l^*$. Then the multigrid preconditioner (68) is symmetric.

Proof: $M_l = M_l^*$ is obviously equivalent to the relation $K_l^{-1} M_l^T = M_l K_l^{-1}$, where M_l^T denotes the matrix that is transposed to M_l . Using this relation, we immediately see that C_l^{-1} is symmetric. Indeed,

$$\begin{aligned} (C_l^{-1})^T &= ((I_l - (M_l)^j) K_l^{-1})^T = K_l^{-1} - K_l^{-1} M_l^T \cdot \dots \cdot M_l^T \\ &= K_l^{-1} - M_l \cdot \dots \cdot M_l K_l^{-1} = C_l^{-1} \end{aligned}$$

Therefore, C_l is symmetric too. \square

Lemma 3.5 *Let us assume that*

$$I_q^{q-1} = (I_{q-1}^q)^T \quad \text{and} \quad \bar{S}_q^{post} = (\bar{S}_q^{pre})^* \quad \forall q = 2, \dots, l, \quad (70)$$

where $\bar{S}_q^{pre} = (S_q^{pre})^{\nu_1(q)}$ and $\bar{S}_q^{post} = (S_q^{post})^{\nu_2(q)}$ for all $q = 2, \dots, l$. Then the multigrid iteration operator M_l is self-adjoint in the K_l -energy scalar product.

Proof: A simple calculation shows that under the assumptions (70) all twogrid iteration operators are self-adjoint in the K_q -energy scalar product, i.e. $M_q^{q-1} = (M_q^{q-1})^*$ for all $q = 2, \dots, l$. Using representation (51) of the multigrid iteration operator M_q , we can easily proof its self-adjointness by induction (see [50, 49] for details). \square

Now we can proof our main theorem on multigrid preconditioners.

Theorem 3.6 *Let us suppose that the symmetry conditions (70) of Lemma 3.5 are fulfilled. Furthermore, we assume that all restrictions I_q^{q-1} ($q = 2, \dots, l$) have full rank and that all stiffness matrices fulfil the so-called Galerkin condition*

$$K_{q-1} = I_q^{q-1} K_q I_q^q \quad \forall q = 1, \dots, l. \quad (71)$$

Finally, we suppose that there is a rate estimate for the multigrid method involved, i.e. there exists some positive constant $\eta = \eta_* \in (0, 1)$ such that

$$\varrho(M_l) = \|M_l\|_{K_l} := \sup_{\underline{v}_l \in R^{N_l}} \frac{\|M_l \underline{v}_l\|_{K_l}}{\|\underline{v}_l\|_{K_l}} (=) \leq \eta < 1, \quad (72)$$

where $\varrho(M_l)$ denotes the spectral radius of M_l . Then the multigrid preconditioner (68) is spd and satisfies the spectral equivalence inequalities (64) with the spectral equivalence $\gamma_1 = 1 - \eta^j$ and $\gamma_2 = 1$.

Proof: The symmetry of the multigrid preconditioner C_l directly results from Lemmas 3.4 and 3.5. The positive definiteness would follow from the spectral equivalence inequalities (64). Thus, it remains to prove the spectral equivalence inequalities (64) with the spectral

constants given in Theorem 3.6. Let us suppose for a moment that the multigrid iteration operator M_l is non-negative in the K_l -scalar product, i.e.

$$(M_l \underline{v}_l, \underline{v}_l)_{K_l} = (K_l M_l \underline{v}_l, \underline{v}_l) \geq 0 \quad \forall \underline{v}_l \in R^{N_l}. \quad (73)$$

Representation (68) and inequality (73) yield the estimates

$$\begin{aligned} (C_l^{-1} \underline{u}_l, \underline{u}_l) &= (K_l^{-1} \underline{u}_l, \underline{u}_l) - ((M_l)^j K_l^{-1} \underline{u}_l, \underline{u}_l) \\ &= (K_l^{-1} \underline{u}_l, \underline{u}_l) - ((M_l)^j \underline{v}_l, \underline{v}_l)_{K_l} \\ &\leq 1 \cdot (K_l^{-1} \underline{u}_l, \underline{u}_l) \quad \forall \underline{u}_l \in R^{N_l}, \end{aligned} \quad (74)$$

with the substitution $\underline{u}_l = K_l \underline{v}_l$. On the other side, we get the inequalities

$$\begin{aligned} (C_l^{-1} \underline{u}_l, \underline{u}_l) &\leq (K_l^{-1} \underline{u}_l, \underline{u}_l) - \|M_l\|_{K_l}^j (\underline{v}_l, \underline{v}_l)_{K_l} \\ &\leq (1 - \eta^j) \cdot (K_l^{-1} \underline{u}_l, \underline{u}_l) \quad \forall \underline{u}_l \in R^{N_l}. \end{aligned} \quad (75)$$

Inequalities (74) and (75) imply $\gamma_2 = 1$ and $\gamma_1 = 1 - \eta^j$, respectively. Of course, it remains to show inequality (73). Again, this inequality is proven in two steps. In a first step, inequality (73) is shown for all twogrid iteration operators M_q^{q-1} instead of M_l . Then, using the representation (51) of the multigrid iteration operator M_q , we can easily proof their non-negativeness by induction (see [50, 49] for details). \square

The second part of the assumptions (70) ensuring the symmetry of the multigrid preconditioner imposes a strong condition on the choice and the arrangement of the pre- and post-smoothing procedures. The following lemma represents a practical proposal for the right choice of the pre- and post-smoothing iteration operators S_q^{pre} and S_q^{post} in the pre- and post-smoothing iteration processes (48).

Lemma 3.7 *Let us assume that pre- and post-smoothing iteration operators have the form*

$$S_q^{pre} := I_q - \omega_q B_q^{-1} K_q \quad \text{and} \quad S_q^{post} := I_q - \omega_q (B_q^{-1})^T K_q \quad (76)$$

with regular $N_q \times N_q$ matrices B_q and appropriately chosen parameters ω_q . If the number $\nu_1(q)$ of pre-smoothing sweeps is equal to the number $\nu_2(q)$ of post-smoothing sweeps, then $\bar{S}_q^{post} = (\bar{S}_q^{pre})^$ for all $q = 2, \dots, l$.*

The *Proof* is left to the reader (see also [50, 49] for the proof). \square

The ω -Jacobi as well as the Gauss-Seidel method fit into the framework of Lemma 3.7. Indeed, the ω -Jacobi method corresponds to $B_q = D_q = \text{diag}(K_q)$. In the Gauss-Seidel case, the pre-smoothing and post-smoothing sweeps have to be arranged in a symmetrical way, e.g. lexicographically forward Gauss-Seidel sweeps in the pre-smoothing steps correspond to lexicographically backward Gauss-Seidel sweeps in the post-smoothing steps.

3.4 The Full Multigrid Method

The choice of the initial guess in Algorithm 3 is certainly important for the efficiency of multigrid methods in practical applications. If the boundary value problem under consideration allows a coarse discretization, then we can solve the corresponding coarse grid system

$$K_1 \underline{u}_1 = \underline{f}_1 \quad (77)$$

by means of some direct method and we can use the obtained coarse grid solution \underline{u}_1 for the adaptive mesh refinement and for constructing some initial guess \underline{u}_2^0 for the next finer mesh by interpolation. Then we improve this interpolated coarse grid solution by calling the multigrid (twogrid) procedure $\text{MGM}(\mathbf{K}_2, \underline{u}_2, \mathbf{f}_2, 2)$ several times. The continuation of this *nested iteration* process leads to the so-called *Full Multigrid Method* (FMGM) that is formally described by Algorithm 5. We mentioned in §2 that N.S. Bachvalow already proposed this nested iteration

Algorithm 5 Full multigrid method: $\underline{v}_q \Leftarrow \text{FMGM}(\mathbf{K}_q, \underline{f}_q, q = 1, l)$

```

 $\underline{v}_1 \Leftarrow \text{SOLVE}(\mathbf{K}_1 \cdot \underline{u}_1 = \underline{f}_1)$  {solution on the coarsest grid}
for  $q = 2$  step 1 until  $l$  do
     $\underline{v}_q \Leftarrow I_{q-1}^q \cdot \underline{v}_{q-1}$  {prolongation}
    for  $k = 1$  step 1 until  $k_q$  do
         $\underline{v}_q \Leftarrow \text{MGM}(\mathbf{K}_q, \underline{v}_q, \underline{f}_q, q)$  {nested multigrid iteration}
    end for
end for

```

approach with the multigrid iteration as nested iteration and showed for his problem that under some assumptions the number of arithmetical operations for computing some FMG approximate $\underline{v}_l^{k_l}$ that differ from the exact solution in the order of the discretization error is direct proportional to the number N_l of unknowns on the finest grid. In this sense, the FMGM is asymptotically optimal. Indeed, this result is stated in Theorem 3.8 under quite general assumptions.

Theorem 3.8 *Let us assume that there are positive and level-independent constants $\eta_* \in (0, 1)$ (multigrid rate), $c_A(u)$ (approximation constant), c_I (interpolation constant) and $\alpha = \sqrt[d]{\tau}$ (enlargement constant) such that the assumptions*

$$(A1) \quad \|M_q\| \leq \eta_* < 1 \quad (\text{see } \S 3.2),$$

$$(A2) \quad \|I_{q-1}^q \underline{u}_{q-1} - \underline{u}_q\| \leq c_A(u) h_q^p,$$

$$(A3) \quad \|I_{q-1}^q\| := \sup_{\underline{v}_{q-1} \in \mathbb{R}^{N_{q-1}}} \frac{\|I_{q-1}^q \underline{v}_{q-1}\|_{\mathbb{R}^{N_q}}}{\|\underline{v}_{q-1}\|_{\mathbb{R}^{N_{q-1}}}} \leq c_I,$$

$$(A4) \quad h_{q-1}/h_q \leq \alpha \quad \text{and} \quad 1 \leq \gamma < \tau$$

hold for $q = 2, 3, \dots, l$ and for all $l = 2, 3, \dots$, where \underline{u}_{q-1} and \underline{u}_q are the exact solution of our finite element equations (1) for $h = h_{q-1}$ and $h = h_q$, respectively. The notation $c_A(u)$ indicates nothing else but the dependence of $c_A(u)$ on the solution u of the underlying boundary value problem. If the number $i := k_q$ of nested iteration satisfies the condition

$$\eta_*^i < 1/c_1 \tag{78}$$

for all $q = 2, 3, \dots, l$, then the estimate

$$\|\underline{u}_q - \underline{v}_q^i\| \leq c_2(\eta_*) c_A(u) h_q^p \tag{79}$$

holds for $q = l$, where $c_1 = c_I \alpha^p$, $c_2(\eta_*) = \eta_*^i / (1 - c_1 \eta_*^i)$, \underline{u}_l again denotes the exact solution of (1) at the finest level l , and \underline{v}_l^i is the FMG approximate produced by Algorithm 5. The number

of arithmetical operations that are needed to produce the FMG approximate \underline{v}_l^i is proportional to the number N_l of unknowns on the finest grid corresponding to the discretization parameter $h = h_l$.

Proof: The crucial estimate (79) is proved by induction. Indeed, for $q = 1$, estimate (79) is trivially fulfilled since we assumed that the coarse grid system (77) is solved by some direct method. Let us suppose that estimate (79) is valid for $q - 1$. We now show that (79) also holds for q . Indeed, using assumptions (A1) - (A4), we arrive at the estimates

$$\begin{aligned}
\|\underline{u}_q - \underline{v}_q^i\| &\leq \eta_*^i \|\underline{u}_q - \underline{v}_q^0\| = \eta_*^i \|\underline{u}_q - I_{q-1}^q \underline{v}_{q-1}^i\| \\
&= \eta_*^i \|\underline{u}_q - I_{q-1}^q \underline{u}_{q-1} + I_{q-1}^q \underline{u}_{q-1} - I_{q-1}^q \underline{v}_{q-1}^i\| \\
&\leq \eta_*^i \left\{ \|\underline{u}_q - I_{q-1}^q \underline{u}_{q-1}\| + \|I_{q-1}^q\| \|\underline{u}_{q-1} - \underline{v}_{q-1}^i\| \right\} \\
&\leq \eta_*^i \left\{ c_A(u) h_q^p + c_I c_2(\eta_*) c_A(u) h_{q-1}^p \right\} \\
&= \eta_*^i (1 + c_I c_2(\eta_*) (h_{q-1}/h_q)^p) c_A(u) h_q^p \\
&\leq \eta_*^i (1 + c_I c_2(\eta_*) (\alpha)^p) c_A(u) h_q^p = c_2(\eta_*) c_A(u) h_q^p
\end{aligned}$$

those finally prove estimate (79). The costs estimate that is stated in the theorem follows from Theorem 3.3 and a simple counting of the arithmetical operations needed by Algorithm 5. \square

It is even sufficient to solve the coarse grid system (77) only approximately, e.g. via some algebraic multigrid method (see §4). The only thing that we have to ensure is the accuracy prescribed by (79) for $q = 1$. Such a *hybrid multigrid method* can certainly be very efficient in such practical applications where the coarse grid system (77) is already quite large.

In order to illustrate the practical relevance of estimate (79), we suppose that $c_I = 1$, $\alpha = 2$, $p = 1$ and $\eta_* = 0.1$. We notice that these constants are typical for linear finite elements and standard refinement. For this setting of the involved constant, we observe that only one ($i = 1$) nested multigrid iteration is sufficient for producing FMGM iterates that differ from the exact solution in the order of the discretization error.

Of course, instead of the multigrid method $\text{MGM}(\mathbf{K}_q, \underline{\mathbf{v}}_q, \underline{\mathbf{f}}_q, q)$, we can also use the multigrid preconditioned CG method $\text{MGPCG}(\mathbf{K}_q, \underline{\mathbf{u}}_q, \underline{\mathbf{f}}_q)$ described by Algorithm 4 as nested iteration in Algorithm 5. In contrast to this MGPCG nested iteration process, P. Deuffhard proposed some cascadic nested iteration process that uses a decreasing number of unpreconditioned (!) CG iterations on the finer grids [25] (see [8] for the analysis). Nowadays, this nested iteration process is called *CASCADE algorithm*.

4 Algebraic Multigrid Methods

4.1 Basic Idea of Algebraic Multigrid

Let us assume that only the fine grid information is available, i.e,

- the matrix $K_h = K_\ell$ and the right hand side $f_h = f_\ell$,
- the mesh $\tau_h = \tau_\ell$ and the node set ω_ℓ .

Then we want to construct the following coarse grid components from that given information:

- matrices K_q and right hand sides f_q , $q = \overline{1, \ell - 1}$,
- coarse mesh τ_q and coarse nodes set ω_q , $q = \overline{1, \ell - 1}$,
- intergrid transfer operators I_{q-1}^q, I_q^{q-1} , $q = \overline{2, \ell}$.

If we would have all of the above components then we could define appropriate smoothing operators S_q and we could apply the standard multigrid algorithm MGM on the sequence of matrices $\{K_q\}_{q=1}^\ell$. The pure (= classical) algebraic multigrid method (AMG) uses only the matrix information stored in K_h .

In order to derive an AMG algorithm, we start with an ideal twogrid approach. This idea cannot be realized in 2D or 3D but its approximation will lead us to good interpolation operators I_{q-1}^q . The coarse operators are derived immediately afterwards and an introduction into coarsening strategies finishes this section. The fine grid index h will be skipped for matrices in the remaining §4.

4.2 An exact Twogrid Method

Assumption: There exists already a classification of nodes into N_C coarse grid nodes (“C”) and N_F fine grid nodes (“F”) with index sets ω_C and ω_F such that $\omega_\ell = \omega_C \cup \omega_F$ and $\omega_C \cap \omega_F = \emptyset$.

This induces the block structure of the fine grid problem (with $\underline{u}_C \in \mathbb{R}^{N_C}$ and $\underline{u}_F \in \mathbb{R}^{N_F}$)

$$K \cdot \underline{u} = \begin{bmatrix} K_{CC} & K_{CF} \\ K_{FC} & K_{FF} \end{bmatrix} \cdot \begin{bmatrix} \underline{u}_C \\ \underline{u}_F \end{bmatrix} = \begin{bmatrix} \underline{f}_C \\ \underline{f}_F \end{bmatrix} = \underline{f}. \quad (80)$$

By introducing the transformation matrix $T = \begin{bmatrix} I_C & \mathbf{O} \\ K_{FF}^{-1} K_{FC} & I_F \end{bmatrix}$ we can express K from (80) as triple product

$$K = T^T \cdot \begin{bmatrix} \hat{S}_C & \mathbf{O} \\ \mathbf{O} & K_{FF} \end{bmatrix} \cdot T \quad (81)$$

with the Schur complement matrix $\hat{S}_C = K_{CC} - K_{CF} K_{FF}^{-1} K_{FC}$. Rearranging (81) leads to

$$\begin{aligned} \hat{K} &:= \begin{bmatrix} \hat{S}_C & \mathbf{O} \\ \mathbf{O} & K_{FF} \end{bmatrix} = T^{-T} \cdot K \cdot T^{-1} \\ &= \begin{bmatrix} I_C & -K_{CF} K_{FF}^{-1} \\ \mathbf{O} & I_F \end{bmatrix} \cdot \begin{bmatrix} K_{CC} & K_{CF} \\ K_{FC} & K_{FF} \end{bmatrix} \cdot \begin{bmatrix} I_C & \mathbf{O} \\ -K_{FF}^{-1} K_{FC} & I_F \end{bmatrix} \end{aligned} \quad (82)$$

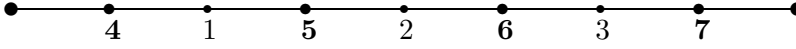
and we extract $\hat{R} = \begin{bmatrix} I_C & -K_{CF} K_{FF}^{-1} \end{bmatrix}$ and $\hat{P} = \begin{bmatrix} I_C \\ -K_{FF}^{-1} K_{FC} \end{bmatrix}$ such that we get the obvious relation

$$\hat{S}_C = \hat{R} \cdot K \cdot \hat{P}. \quad (83)$$

If we interpret \hat{S}_C as coarse grid matrix K_H and \hat{P} as (exact) interpolation I_H^h , respective \hat{R} as restriction I_h^H then (83) represents the Galerkin approach (30) and we can define the twogrid Algorithm 1 with these components. This twogrid method with exact interpolation and restriction acts as a direct solver.

Example 1D: (see (2) and (3) in §1: $-u''(x) = f(x)$, $x \in (0, 1)$; $u(0) = u(1) = 0$)

An FEM discretization with linear elements and $N = 7$, $h = \frac{1}{8}$ leads to components:



$$K = \frac{1}{h} \left[\begin{array}{ccc|ccc} 2 & & & -1 & -1 & & \\ & 2 & & & -1 & -1 & \\ & & 2 & & & -1 & -1 \\ \hline -1 & & & 2 & & & \\ -1 & -1 & & & 2 & & \\ & -1 & -1 & & & 2 & \\ & & -1 & & & & 2 \end{array} \right] \begin{array}{l} \mathbf{C} \\ \mathbf{F} \end{array} = \begin{bmatrix} K_{CC} & K_{CF} \\ K_{FC} & K_{FF} \end{bmatrix},$$

$$\hat{P} = \frac{1}{2} \begin{bmatrix} 2 & & & & \\ & 2 & & & \\ & & 2 & & \\ 1 & & & & \\ 1 & 1 & & & \\ & & 1 & 1 & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} = -K_{FF}^{-1} K_{FC} \quad (\hat{R} \equiv \hat{P}^T), \text{ i.e., linear prolongation (!),}$$

$$\hat{S}_C = \hat{R} \cdot K \cdot \hat{P} = \frac{1}{(2h)} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}.$$

The coarse grid matrix \hat{S}_C is identical to a matrix K_H resulting from the FEM discretization of the coarse mesh. The exact twogrid method with the components above works perfectly in 1D since K_{FF} is a diagonal matrix and therefore easy to invert.

Example 2D: (see also §2: $-\Delta u = f$ in $\Omega = (0, 1) \times (0, 1)$, $u = 0$ on $\partial\Omega$)

An FEM discretization with linear triangular elements and $N = 49$, $h = \frac{1}{8}$ leads to a system

matrix K with the stencil $K = \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}.$

Now, K_{FF} is no longer a diagonal matrix:

\Rightarrow Inverting K_{FF} for calculation of interpolation operator is as expensive as solving the original problem

\Rightarrow Exact twogrid is not applicable in 2D (and 3D) !

Observation: The structure of the entries in $\hat{P}(\hat{R})$ and \hat{S}_C calculated via (82)-(83) coincides rather good with the structure of a bilinear prolongation/restriction and a coarse matrix resulting from a 5-point stencil discretization, respectively a 9-point stencil discretization. Therefore, there is a chance to approximate \hat{P} with something cheaper.

4.3 Prolongation

Idea: Approximation of $-K_{FF}^{-1}K_{FC}$ by an appropriate operator B_{FC} . This approximation has to be good enough, otherwise the coarse operator derived via Galerkin approach does no longer reflect the characteristics of the fine grid operator, e.g., null space and changing coefficients.

4.3.1 Matrix Weighted Prolongation in 1D

Let us have a look at the simple 1D example

$$\begin{aligned} & - (a(x)u'(x))' = 0 \quad \forall x \in (0, \tfrac{1}{2}) \cup (\tfrac{1}{2}, 1) \quad (84) \\ \text{with interface conditions} \quad & u(\tfrac{1}{2} - 0) = u(\tfrac{1}{2} + 0) \\ & a(x)u'(x)|_{x=\frac{1}{2}-0} = a(x)u'(x)|_{x=\frac{1}{2}+0} \\ \text{with boundary conditions} \quad & u(0) = 1, \quad u(1) = 0 \end{aligned}$$

where $a(x) = \begin{cases} a_1 = 10 & x \in [0, \frac{1}{2}] \\ a_2 = 1 & x \in (\frac{1}{2}, 1] \end{cases}$. The exact solution $u(x) = \begin{cases} -\frac{2}{11}x + 1 & x \in [0, \frac{1}{2}] \\ -\frac{20}{11}(x - 1) & x \in [\frac{1}{2}, 1] \end{cases}$ is nothing but the harmonic extension of the boundary data into the domain $(0, 1)$ with respect to the differential operator (84).

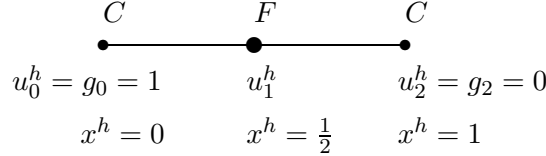


Figure 6: A simple 1D grid

The FEM discretization in Fig. 6 with linear elements and the grid nodes $\tau_h = \{x_0^h = 0, x_1^h = \frac{1}{2}, x_2^h = 1\}$ defines a FE basis $\Phi^h = \{\varphi_0^h(x), \varphi_1^h(x), \varphi_2^h(x)\}$ and results, after a partial modification with respect to the Dirichlet boundary conditions, in the system

$$K \cdot \underline{u}^h = \frac{1}{2} \begin{bmatrix} 2 & 0 & 0 \\ -a_1 & a_1 + a_2 & a_2 \\ 0 & 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} u_0^h \\ u_1^h \\ u_2^h \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \underline{f}^h \quad (85)$$

with the solution $u_0^h = 1$, $u_2^h = 0$ and $u_1^h = \frac{a_1 u_0 + a_2 u_2}{a_1 + a_2} = \frac{-1}{K_{11}}(K_{10}u_0 + K_{12}u_2) = \frac{10}{11}$.

The FE solution $u^h(x) = \Phi^h \cdot \underline{u}$ is identical to $u(x)$ in this example.

The classical FE basis $\Phi^H = \{\varphi_0^H(x), \varphi_1^H(x)\}$ of the coarse grid $\tau_H = \{x_0^H = 0, x_1^H = 1\}$ can

be derived from fine basis Φ^h by using the linear interpolation $I_h^H = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{bmatrix}$ and the relation

$\Phi^H = \Phi^h \cdot I_h^H$, i.e., $\varphi_0^H = \varphi_0^h + \frac{1}{2}\varphi_1^h$ and $\varphi_1^H = \frac{1}{2}\varphi_1^h + \varphi_2^h$. The related coarse grid solution

$u^H(x) = \Phi^H \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is a really poor approximation of the fine grid solution $u^h(x)$. This

approximation can be improved by choosing a non-standard coarse basis $\hat{\Phi}^H$ derived from

$\hat{\Phi}^h$ with the exact interpolation $\hat{P} = \begin{bmatrix} I_C \\ -K_{FF}^{-1}K_{FC} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{a_1}{a_1+a_2} & \frac{a_2}{a_1+a_2} \\ 0 & 1 \end{bmatrix}$ (matrix K from (85)).

Therefore, the new coarse basis is $\hat{\Phi}^H = \Phi^h \cdot \hat{P}$, i.e., $\varphi_0^H = \varphi_0^h + \frac{10}{11}\varphi_1^h$ and $\varphi_1^H = \frac{1}{11}\varphi_1^h + \varphi_2^h$, and the coarse FE solution $\hat{u}^H(x) = \hat{\Phi}^H \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is again identical to the fine grid solution u^h .

The coarse function $\hat{u}^H(x)$ is the solution of (84), i.e., $\hat{u}^H(x)$ is the harmonic extension, and matrix \hat{P} is the discrete extension operator from coarse grid nodes to fine grid nodes. If we

write $\hat{P} = \begin{bmatrix} 1 & 0 \\ \frac{K_{10}}{K_{10}+K_{12}} & \frac{K_{12}}{K_{10}+K_{12}} \\ 0 & 1 \end{bmatrix}$ then \hat{P} is the matrix weighted linear interpolation, i.e.,

$$u_1^h = \frac{1}{K_{10} + K_{12}}(K_{10}u_0 + K_{12}u_2) . \quad (86)$$

4.3.2 Matrix Weighted Prolongation for a 5-point stencil

The 2D- or 3D-approximation of \hat{P} has to be a local approximation of the discrete extension operator from coarse space \mathbb{R}^{N_C} to fine space $\mathbb{R}^{N_C+N_F}$.

Let us consider a prolongation for a 5-point stencil on an equidistant tensor product mesh.

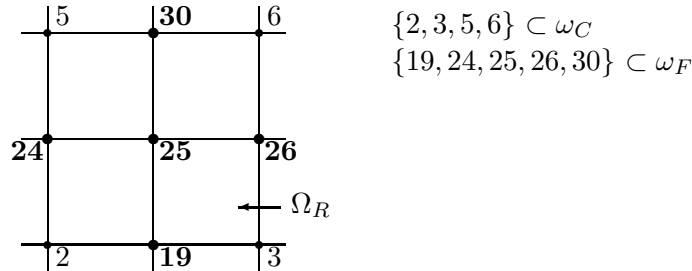


Figure 7: A simple 2D grid

1. $u_{19}, u_{24}, u_{26}, u_{30}$ are calculated with matrix weighted interpolation (86).
2. u_{25} : We determine the discrete harmonic extension in Ω_R , i.e., we multiply the 25th row of matrix K with the vector of unknowns \underline{u} and solve $0 = \sum_{i=1}^N K_{25,i} \cdot u_i$, and get
$$u_{25} = \frac{-1}{K_{25,25}} \left\{ K_{25,19} u_{19} + K_{25,24} u_{24} + K_{25,26} u_{26} + K_{25,30} u_{30} \right\}$$

$$\vdots$$
substitute u_{19}, \dots, u_{30} with result from item 1.

Further algebraic transformations lead to the matrix weighted bilinear interpolation

$$u_{25} = \frac{-1}{K_{25,25}} \left\{ \left(\frac{K_{25,19}K_{19,2}}{K_{19,2} + K_{s,3}} + \frac{K_{25,24}K_{24,2}}{K_{24,2} + K_{24,5}} \right) u_2 + \left(\frac{K_{25,19}K_{19,3}}{K_{19,2} + K_{19,3}} + \frac{K_{25,26}K_{26,3}}{K_{26,3} + K_{26,6}} \right) u_3 \right. \\ \left. + \left(\frac{K_{25,24}K_{24,5}}{K_{24,2} + K_{24,5}} + \frac{K_{25,30}K_{30,5}}{K_{30,5} + K_{30,6}} \right) u_5 + \left(\frac{K_{25,26}K_{26,6}}{K_{26,3} + K_{26,6}} + \frac{K_{25,30}K_{30,6}}{K_{30,5} + K_{30,6}} \right) u_6 \right\} \quad (87)$$

4.3.3 Prolongation for general meshes

Let us illustrate the graph of an arbitrary FE matrix as a mesh and mark the fine and coarse nodes (ω_C , ω_F) as in Fig. 8. Therein we have:

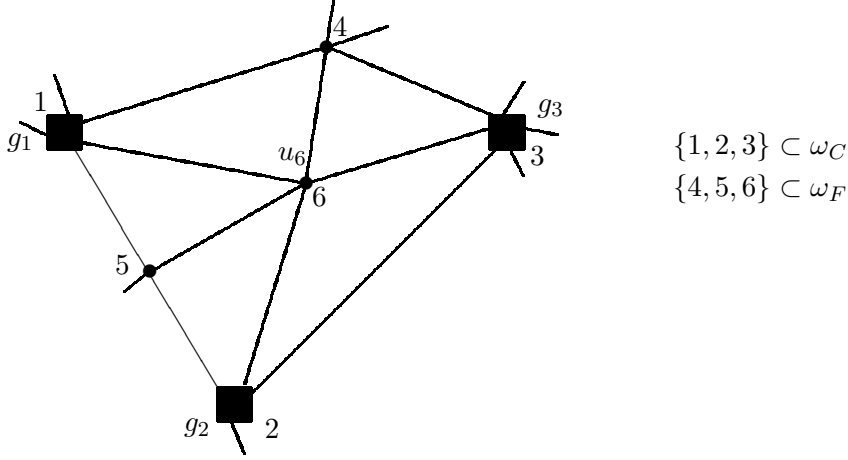


Figure 8: Neighborhood of node 6.

- Set of coarse neighbors: $C^i = \{ \text{nodes} \in \omega_C, \text{ connected with } \underbrace{\text{node } i}_{\in \omega_F} \}$,
e.g., $C^6 = \{1, 2, 3\}$, $C^5 = \{1, 2, \dots\}$, $C^4 = \{1, 3, \dots\}$.
Set of fine neighbors: $F^i = \{ \text{nodes} \in \omega_F, \text{ connected with node } i \in \omega_F \}$
e.g., $F^6 = \{4, 5\}$
- We know that interpolation weights $\alpha_{ij} = 1 \quad \forall j \in \omega_C$ and we have to determine:

$$\alpha_{ij} = \begin{cases} 1 & \forall i \in \omega_F \quad \forall j \in C^i \\ 0 & \text{else} \end{cases}$$

Now, we will determine the integration weights α_{ij} :

Variant 0:

Linear interpolation (86) between (coarse) father and (fine) son nodes.

Unstructured grid has no unique father son relations anymore.

Therefore, node 6 may use all 3 coarse nodes for interpolation, i.e. $j \in C^6$.

Variant 1 (improved):

Interpolation in $i = 6 \iff$ harmonic extension inside the polygon

$$1 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$$

$$\iff \sum_{j \in F^i \cup C^i \cup \{i\}} K_{ij} u_j = 0$$

$$+ \text{B.C. } u_j = g_j \quad \forall j \in F^i \cup C^i$$

$$\iff K_{66} u_6 = - \underbrace{\sum_{j=1}^3 K_{6j} g_j}_{\substack{j \in C^i \\ g_j \text{ known}}} - \underbrace{\sum_{j=4}^5 K_{6j} g_j}_{\substack{j \in F^i \\ g_j \text{ unknown!}}} \quad (*)$$

Now, linear interpolation (86) for calculation of unknowns $g_j \ j \in F^i$

$$g_5 = \frac{K_{51} g_1 + K_{52} g_2}{K_{51} + K_{52} [\underbrace{+ K_{53}}_{=0}]} \left[= \hat{c}_{51} g_1 + \hat{c}_{52} g_2 \right] \text{ and } g_4 = \frac{K_{41} g_1 + K_{43} g_3}{K_{41} + K_{43}}.$$

$\xRightarrow{(*)}$

$$u_6 = \frac{-1}{K_{66}} \left\{ \left(K_{61} + \frac{K_{65} K_{51}}{K_{51} + K_{52}} + \frac{K_{64} K_{41}}{K_{41} + K_{43}} \right) g_1 + \right. \\ \left. + \left(K_{62} + \frac{K_{65} K_{52}}{K_{51} + K_{52}} \right) g_2 + \left(K_{63} + \frac{K_{64} K_{43}}{K_{41} + K_{43}} \right) g_3 \right\}$$

Introducing

$$\hat{c}_{ij} := \sum_{k \in F^i} \frac{K_{ik} \cdot K_{kj}}{\sum_{l \in C^i} K_{kl}} \quad (88)$$

leads to

$$u_i^h = u_i = \frac{-1}{K_{ii}} \left\{ \sum_{j \in C^i} (K_{ij} + \hat{c}_{ij}) \underbrace{g_j}_{=u_j^H} \right\}$$

and finally to the interpolation weights

$$\hat{\alpha}_{ij} = \begin{cases} 1 & i = j \in \omega_C \\ \frac{-(K_{ij} + \hat{c}_{ij})}{K_{ii}} & i \in \omega_F, j \in C^i \\ 0 & \text{else} \end{cases} \quad (89)$$

Formulae (89) can be further improved without changing the pattern of the prolongation matrix P .

Variant 2: J.W. Ruge and K. Stüben proposed in [63, 64] an improved calculation of g_5, g_4 .

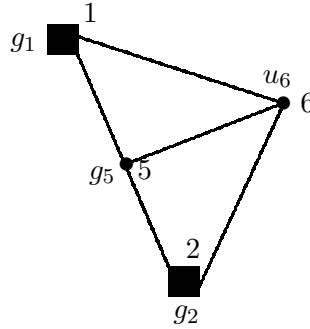


Figure 9: Restricted neighborhood of node 5

We investigate g_5 in the local system (Fig. 9) $K_{51} g_1 + K_{52} g_2 + \tilde{K}_{55} g_5 + K_{56} u_6 = 0$ with

known values g_1, g_2, u_6 and $\tilde{K}_{55} = -(K_{51} + K_{52} + K_{56})$ (row sum has to be zero)

$$\Rightarrow g_5 = \frac{K_{51}}{K_{56} + K_{51} + K_{52}} g_1 + \frac{K_{52}}{K_{56} + K_{51} + K_{52}} g_2 + \frac{K_{56}}{K_{56} + K_{51} + K_{52}} u_6 \quad (\text{also } g_4).$$

The same methodology as in variant 1 leads to formulas

$$c_{ij} := \sum_{k \in F^i} \frac{K_{ik} \cdot K_{kj}}{\sum_{l \in C^i} K_{kl} + K_{ki}} \quad (90)$$

$$\alpha_{ij} := \begin{cases} 1 & i = j \in \omega_C \\ \frac{-(K_{ij} + c_{ij})}{K_{ii} + c_{ii}} & i \in \omega_F, j \in C^i \\ 0 & \text{else} \end{cases} \quad (91)$$

4.4 Coarsening

4.4.1 Motivation

The pure algebraic multigrid uses only matrix information to derive the coarse system, i.e., known: matrix $K = K_\ell$, right-hand side $f = f_\ell$, find grid indices ω_ℓ

unknown: coarse grid indices $\omega_k = \omega_{k+1,C}$ $k = \overline{1, \ell-1}$
transfer operators P_k, R_k $k = \overline{1, \ell-1}$
matrices K_k $k = \overline{1, \ell-1}$
right-hand side f_k $k = \overline{1, \ell-1}$

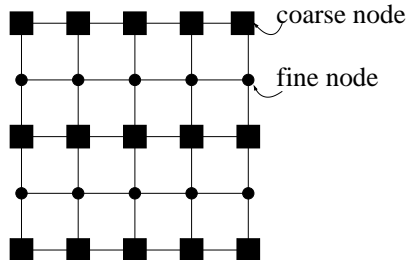
known: $[\omega,] K$

unknown: $[\omega_C, \omega_F,] P, R, K_H = R K P$

The main problem consists in finding the coarse nodes ω_C such that the main properties of the operator K are reflected in the coarse operator K_H . The following determination of the prolongation operator will be done analogously to §4.3.2.

Example anisotropic operator: $\varepsilon \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f$ in $(0,1)^2$ with boundary conditions.

The unit square is discretized by an equidistant tensor product grid and the standard finite difference discretization results in the 5-point stencil

$$\frac{1}{h^2} \begin{bmatrix} & & -1 & & \\ -\varepsilon & 2\varepsilon + 2 & & -\varepsilon & \\ & & -1 & & \end{bmatrix},$$


i.e., if ε tends to zero then the problems in y -direction become completely independent. Therefore, a coarsening in y -direction can be applied but no coarsening in x -direction, see the figure above for illustration. This semi-coarsening cannot be reproduced by a simple general coarsening strategies without taking into account the operator characteristics. (Special line smoothers could solve the problem in the given example but this is not a general approach.)

A closer look at the matrix entries resulting from the stencil gives us the idea that coars-

ening is allowed into the direction of relatively large entries. This is indeed the basic idea of the following coarsening strategy.

4.4.2 Coarsening by Ruge/Stüben

Let us define the strength of coupling to a node i with respect to a set of nodes $I \subset V = \omega$ (mostly a single node) by

$$d(i, I) := \frac{1}{\max_k \{|K_{ik}|\}} \sum_{j \in I} -K_{ij} \ .$$

This leads to the set of nodes whose coupling to node i is stronger than a threshold parameter α

$$S^i := \{j \in V \mid d(i, \{j\}) \geq \alpha\} \quad \begin{array}{c} \bullet \\ i \longleftarrow j \\ \bullet \end{array}$$

and to the set of nodes to which node i possesses a strong coupling.

$$S^{i,T} := \{j \in V \mid i \in S^j\} \quad \begin{array}{c} \bullet \\ i \longrightarrow j \\ \bullet \end{array}$$

The relation $S^i = S^{i,T}$ holds for matrices K resulting from a discretization of the Laplace operator but this is no longer true for anisotropic operators.

The coarsening algorithm by J.W. Ruge and K. Stüben [30, 63, 64] tries in phase I to find a set of coarse nodes ω_C with the property that no two points are strongly connected to each other.

Algorithm 6 Coarsening Phase I: Split $V = \omega$ into C, F

```

 $C = \emptyset, F = \emptyset$ 
while  $C \cup F \neq V$  do
  pick  $i \in V \setminus \{C \cup F\}$  with maximal  $|S^{i,T}| + |S^{i,T} \cap F|$  {heuristic}
  if  $|S^{i,T}| + |S^{i,T} \cap F| = 0$  then
     $F := V \setminus C$  { $\hat{=}$  STOP}
  else
     $C := C \cup \{i\}, F := F \cup \{S^{i,T} \setminus C\}$ 
  end if
end while

```

Phase II of the algorithm has to guarantee that each node in F is either strongly connected to a node from C , or it is coupled to a node from F with a strong coupling to a node from C .

- All parts marked **sans serif** in the algorithms above are heuristic, i.e., other strategies can be used.
- Parameters α, β are also heuristically. In [30], the authors chose $\alpha = 0.25$ (obvious for the 5-point stencil) and $\beta = 0.35$.
- The Ruge–Stüben algorithm produces again a regular coarse grid from a regular fine grid if the node numbering supports the heuristic part in the algorithms properly.

Algorithm 7 Coarsening Phase II: Split $V = \omega$ into C, F

```

 $T := \emptyset$  {Set of already checked nodes}
while  $T \subset F$  do
  pick  $i \in F \setminus T$ ;  $T := T \cup \{i\}$ ;  $\tilde{C} := \emptyset$ 
   $C^i = S^i \cap C$  {strong neighbor in  $C$ }
   $F^i = S^i \cap F$  {strong neighbor in  $F$ }
  while  $F^i \neq \emptyset$  do
    pick  $j \in F^i$ ;  $F^i := F^i \setminus \{j\}$ 
    if  $d(j, C^i) \leq \beta \cdot d(i, \{j\})$  then
      if  $|\tilde{C}| = 0$  then
         $\tilde{C} := \{j\}$ ;  $C^i = C^i \cup \{j\}$ 
      else
         $C := C \cup \{i\}$ ;  $F = F \setminus \{i\}$ 
      end if
    end if
  end while
   $C := C \cup \tilde{C}$ ,  $F = F \setminus \tilde{C}$ 
end while
 $\omega_C = C$ ,  $\omega_F := F$ 

```

- If the geometry is known then we can improve the rough approximation of the geometry by initializing C with these geometric nodes.
 → start Ruge–Stüben algorithm with $C = \{\text{geometric nodes}\}$

The coarsening algorithm on page 32 requires a criteria for strong connections. We presented the criteria $|K_{ij}| \geq \alpha \max_k \{|K_{ik}|\}$ for a node j strongly coupled to node i [63]. Another choice is $|K_{ij}| \geq \Theta \sqrt{K_{ii} \cdot K_{jj}}$ by P. Vanek, J. Mandel, M. Brzina Br [72] with a level dependent parameter Θ .

4.4.3 Other Coarsening Strategies

A simple but useful coarsening technique is the agglomeration technique by D. Braess [9] which determines aggregates by finding patterns in the matrix graph. Each aggregate will be handled as one (virtual) coarse node. P. Vanek et al. [72] used their coarsening criteria for considering operator properties in the construction of agglomerates. This technique is fast, well suited for block systems resulting from systems of PDEs and produce a sparse coarse matrix K_H . The agglomeration technique uses piecewise constant interpolation for intergrid transfer so that robustness with respect to varying material parameters cannot be achieved. If this interpolation is followed by (at least) one smoothing step then robustness can be achieved [71, 72]. But each smoothing sweep in the prolongation and restriction extends the interpolation stencil by one layer of fine nodes so that the coarse matrix is getting denser and the whole algorithm will be more expensive.

Another very interesting approach is called AMGe and was introduced by the Livermore group [23]. The main idea consists in preserving non-overlapping elements and their matrices on all grids. Neighboring elements will be combined to patches allowing to determine the coarse nodes with respect to the quality of the interpolation. The quality of interpolation is

described by the functional

$$\frac{\langle (I - PR)\underline{e}, (I - PR)\underline{e} \rangle}{\langle K\underline{e}, \underline{e} \rangle}$$

which has to be minimized for all test vectors $\underline{e} \in \mathbb{R}^N \setminus \ker(K)$ under the constraint of small arithmetic work in the multigrid algorithm. The minimization of the global functional is too expensive and therefore the appropriate functionals on the patches are minimized. This approach has been refined for element agglomeration techniques [47] and the strategy can be applied if no element information is available [44].

4.5 Calculation of coarse matrix

The approximation of the exact prolongation $\hat{P} = \begin{bmatrix} I_C \\ -K_{FF}^{-1}K_{FC} \end{bmatrix}$ by an approximation $P = \begin{bmatrix} I_C \\ B_{FC} \end{bmatrix}$ in §4.3 changes formulae (82) into

$$\begin{bmatrix} K_H & \neq \mathbf{O} \\ \neq \mathbf{O} & K_{FF} \end{bmatrix} = \begin{bmatrix} I_C & B_{FC}^T \\ \mathbf{O} & I_F \end{bmatrix} \cdot \begin{bmatrix} K_{CC} & K_{CF} \\ K_{FC} & K_{FF} \end{bmatrix} \cdot \begin{bmatrix} I_C & \mathbf{O} \\ B_{FC} & I_F \end{bmatrix} \quad (92)$$

with $K_H = \hat{S}_C + T_C$ and the distortion $T_C = (B_{FC}^T + K_{CF}K_F^{-1})K_F(B_{FC} + K_F^{-1}K_{FC})$. The non-zero off-diagonal blocks of the left term in (92) cannot be considered in the solution process and therefore we use $C := \begin{bmatrix} K_H & \mathbf{O} \\ \mathbf{O} & K_{FF} \end{bmatrix}$ to approximate the exact operator in (82). This approximation can be quantified via the condition number κ

$$\kappa(C^{-1}\hat{K}) = \left(\sqrt{\mu} + \sqrt{\mu+1} \right)^2$$

where $\mu = \varrho(\hat{S}_C^{-1}T_C)$ denotes the spectral radius of $\hat{S}_C^{-1}T_C$ [35]. The coarse matrix K_H is generated by the Galerkin approach (30)

$$K_H = R \cdot K \cdot P = K_C - K_{CF}B_{FC} - B_{FC}^TK_{FC} + B_{FC}^TK_FB_{FC} \quad (93)$$

with transfer operators $P, R = P^T$ calculated via (91). Note, that (93) does not contain K_F^{-1} and that the sparsity pattern of K_H is determined by the term $B_{FC}^TK_FB_{FC}$. We can calculate the coarse matrix entries with the formulae

$$[K_H]_{ij} = \left[\sum_{k \in \mathcal{N}_i} \sum_{l \in \mathcal{N}_j} \alpha_{ki} K_{kl} \alpha_{lj} \right]_{ij}, \quad (94)$$

where $\mathcal{N}_i = C^i \cup F^i \cup \{i\}$ denotes the (strong) neighborhood of a node i . A straight forward implementation of (94) for generating the coarse system matrix results in very large computational times since many search procedures with complexity $\mathcal{O}(N_C^2)$ are necessary. An interchange of the inner and outer loops lead to an optimal algorithm of complexity $\mathcal{O}(N_F) = \mathcal{O}(N_C)$, see Algorithm 8. F^i, C^i are defined as in §4.2.

Algorithm 8 Algorithm for generating the coarse matrix according to (94)

```

 $K_H := 0$ 
for all  $k \in \omega_F \cup \omega_C$  do
  Determine  $F^k, C^k$ 
  for all  $l \in F^k \cup C^k \cup \{k\}$  do
     $T := C^k \cup C^l \cup (\{l\} \cap \omega_C)$ 
    for all  $i \in T$  do
      for all  $j \in T$  do
         $K_{H,i,j} := K_{H,i,j} + \alpha_{ki} K_{kl} \alpha_{lj}$ 
      end for
    end for
  end for
end for

```

4.6 Problem discussion

The classical AMG by J.W. Ruge and K. Stüben was originally designed for M-matrices as system matrices. An M-matrix is, roughly spoken, a weakly diagonally dominant matrix ($|K_{ii}| \geq \sum_{j \neq i} |K_{ij}|$) with positive diagonal entries and non-positive off-diagonal entries. This M-matrix property represents the discrete maximum principle.

If we use quadrilateral finite elements, or FE basis functions of higher polynomial degree, or the matrix results from a system of PDEs, or we have a poor triangulation then the M-matrix property is easily lost in the system matrix. We list a few approaches how to handle this case:

- a) Ignore small positive off-diagonal entries or add (lump) them to the diagonal.
The lumping changes the discrete operator, i.e., the matrix does no longer represent the original differential operator and therefore this method gets unstable easily.
- b) Use agglomeration, see §4.4.3.
- c) Use AMGe techniques, see §4.4.3.
- d) If we have access to the element stiffness matrices $K^{(r)}$, which accumulate to the stiffness matrix K , then we can use the element preconditioning technique [36]. This technique consists in finding element matrices $C^{(r)}$ which are M-matrices that approximate best the original matrices $K^{(r)}$ in the spectral sense. The matrix C accumulated from the $C^{(r)}$ matrices will be used in the AMG. This technique results in very good results for certain classes of finite elements.
- e) Another technique [34] constructs an auxiliary matrix B by using knowledge of fine mesh coordinates and operator properties. This matrix B is an M-matrix by construction and it will be used for finding the coarse nodes. All remaining components in AMG are standard.

5 Parallelization

5.1 Data Partitioning and Basic Parallel Routines

Our data partitioning bases on the non-overlapping distribution of finite elements that form a subdomain but all the statements and techniques in this section can be easily adapted to other data decompositions. This induces in 2D three classes of nodes: inner nodes, edge nodes and vertex nodes. Face nodes are collected in an additional class in 3D. All nodes excluding the inner nodes are often named as interface nodes.

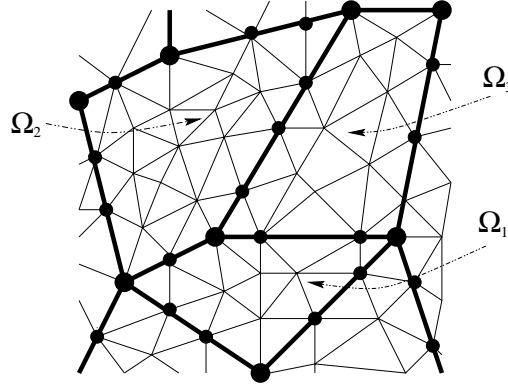


Figure 10: Nonoverlapping domain decomposition

The mapping of a vector $\underline{w} = (w^{[i]})_i \in \mathbb{R}^{N_h}$ in global numbering onto a local vector $\underline{w}_s \in \mathbb{R}^{N_s}$ in subdomain $\overline{\Omega}_s$ ($s = \overline{1, P}$) is represented symbolically by $\mathcal{A}_s \cdot \underline{w}$ with subdomain connectivity matrices $\mathcal{A}_s : \mathbb{R}^{N_h} \mapsto \mathbb{R}^{N_s}$. The transpose \mathcal{A}_s^T of these binary matrices \mathcal{A}_s maps a local vector back to the global one. The index set of all subdomains, a component $w^{[j]}$, $j \in \omega_h$ belongs to, is denoted by

$$\sigma^{[j]} := \{s \mid \exists i \in \omega_s : \mathcal{A}_s^{[i,j]} \neq 0\} \quad \forall j \in \omega_h . \quad (95)$$

There are (at least) two possibilities to store those components and consequently that vector (see also Fig. 11). A vector $\underline{\mathbf{w}}$ is called an accumulated vector if each vector component $\mathbf{w}^{[i]}$ is stored in all subdomains Ω_s , $s \in \sigma^{[i]}$ with its full value. The local vectors $\underline{\mathbf{w}}_s$ can be represented as

$$\underline{\mathbf{w}}_s := \mathcal{A}_s \cdot \underline{\mathbf{w}} . \quad (96)$$

A vector $\underline{\mathbf{r}}$ is called a distributed vector if it is decomposed into local vectors $\underline{\mathbf{r}}_s$ such that

$$\underline{\mathbf{r}} = \sum_{s=1}^P \mathcal{A}_s^T \cdot \underline{\mathbf{r}}_s \quad (97)$$

holds, i.e., all subdomains Ω_s , $s \in \sigma^{[i]}$ store only $\underline{\mathbf{r}}_s$ and possess a portion of the full vector value $\mathbf{r}^{[i]}$ which can be determined only by taking the sum in (97). The conversion of a distributed vector $\underline{\mathbf{v}}$ into an accumulated vector $\underline{\mathbf{w}}$ can be done by evaluating the sum in (97) followed by the restriction (96),

$$\underline{\mathbf{w}} \Leftarrow \underline{\mathbf{v}} \quad : \quad \underline{\mathbf{w}}_s := \mathcal{A}_s \cdot \underline{\mathbf{w}} = \mathcal{A}_s \cdot \sum_{s=1}^P \mathcal{A}_s^T \cdot \underline{\mathbf{v}}_s . \quad (98)$$

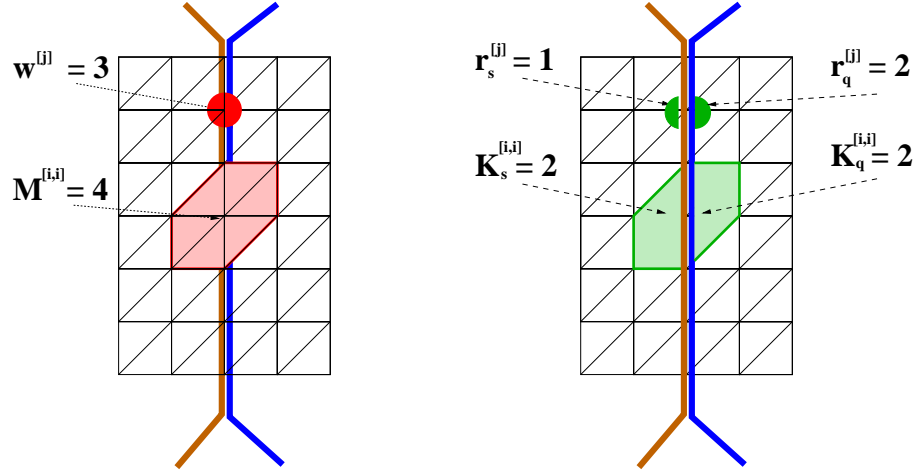


Figure 11: Illustration of accumulated and distributed vectors and matrices.

A matrix can also be stored in two ways. A matrix \mathfrak{P} is called accumulated if its local restrictions \mathfrak{P}_s possess the full entries of it, and we can write

$$\mathfrak{P}_s := \mathcal{A}_s \cdot \mathfrak{P} \cdot \mathcal{A}_s^T. \quad (99)$$

We call a matrix \mathbf{K} distributed if it is the sum of locally stored matrices \mathbf{K}_s

$$\mathbf{K} := \sum_{s=1}^P \mathcal{A}_s^T \cdot \mathbf{K}_s \cdot \mathcal{A}_s \quad (100)$$

holds, i.e., each subdomain $\overline{\Omega}_s$ stores only a part of its full entries. Distributed matrices \mathbf{K}_s are automatically obtained after the FE-accumulation in the subdomains Ω_s due to our non-overlapping construction principle. The sum in (100) is equivalent to a global matrix accumulation which will not be carried out in the parallel case.

It is trivial that additive combinations of vectors from the same type do not require any communication. The inner product of different type vectors requires one global reduce operation of the local inner products:

$$\langle \underline{\mathbf{w}}, \underline{\mathbf{r}} \rangle = \underline{\mathbf{w}}^T \underline{\mathbf{r}} = \underline{\mathbf{w}}^T \sum_{s=1}^P \mathcal{A}_s^T \underline{\mathbf{r}}_s = \sum_{s=1}^P (\mathcal{A}_s \underline{\mathbf{w}})^T \underline{\mathbf{r}}_s = \sum_{s=1}^P \langle \underline{\mathbf{w}}_s, \underline{\mathbf{r}}_s \rangle. \quad (101)$$

Any other combination of vector types requires more effort. The multiplication of a distributed matrix \mathbf{K} with an accumulated vector $\underline{\mathbf{w}}$

$$\mathbf{K} \cdot \underline{\mathbf{w}} = \sum_{s=1}^P \mathcal{A}_s^T \mathbf{K}_s \mathcal{A}_s \cdot \underline{\mathbf{w}} = \sum_{s=1}^P \mathcal{A}_s^T (\mathbf{K}_s \cdot \underline{\mathbf{w}}_s) = \sum_{s=1}^P \mathcal{A}_s^T \underline{\mathbf{v}}_s = \underline{\mathbf{v}}, \quad (102)$$

results in a distributed vector. The realization requires no communication at all because we only have to compute locally $\underline{\mathbf{v}}_s = \mathbf{K}_s \cdot \underline{\mathbf{w}}_s$.

The situation changes if we use an accumulated matrix \mathfrak{P} . Here we have to ensure that the pattern of \mathfrak{P} fulfills the condition

$$\forall i, j \in \omega_h : \quad \sigma^{[i]} \not\subseteq \sigma^{[j]} \implies \mathfrak{P}^{[i,j]} = 0 . \quad (103)$$

If the pattern condition (103) holds, then the operations

$$\underline{\mathbf{w}} = \mathfrak{P} \cdot \underline{\mathbf{w}}_H \quad \text{and} \quad \underline{\mathbf{r}}_H = \mathfrak{P}^T \cdot \underline{\mathbf{r}} \quad (104)$$

can be performed locally without any communication, i.e., $\forall s = \overline{1, P}$

$$\underline{\mathbf{w}}_s = \mathfrak{P}_s \cdot \underline{\mathbf{w}}_{H,s} \quad \text{and} \quad \underline{\mathbf{r}}_{H,s} = \mathfrak{P}_s^T \cdot \underline{\mathbf{r}}_s . \quad (105)$$

Under the assumption (103), the triple product $\mathbf{K}_H := \mathfrak{P}^T \cdot \mathbf{K} \cdot \mathfrak{P}$ can be performed locally without communication

$$\mathbf{K}_{H,s} = \mathfrak{P}_s^T \cdot \mathbf{K}_s \cdot \mathfrak{P}_s . \quad (106)$$

5.2 Parallel Multigrid

The operations (101) and (102) allow already to formulate a parallel preconditioned conjugate gradient (PCG) algorithm for solving the matrix equation $\mathbf{K}\underline{\mathbf{u}} = \underline{\mathbf{f}}$ with a preconditioner C_K , see [57] and Algorithm 4 on page 20.

One possible choice for the preconditioner is $C_K^{-1} = (I - M_K)\mathbf{K}^{-1}$, with M_K being the multigrid iteration operator for \mathbf{K} , see §3. The parallel multigrid iteration is presented in Algorithm 9, where now $\ell = 1$ stands for the finest grid since the number of coarser grids is unknown in the beginning!! Therefore, the preconditioner is called as $\text{PMG}(\{\mathbf{K}^1, \mathfrak{P}^1\}, \underline{\mathbf{u}}, \underline{\mathbf{f}}, 1)$. The algorithm applies the parallel version of the smoother $\text{SMOOTH}(\mathbf{K}, \underline{\mathbf{u}}, \underline{\mathbf{f}})$ and its transpose, e.g., a block Jacobi smoother with Gauss-Seidel smoothing in blocks containing interior unknowns of the subdomains, see also [48]. Furthermore, the interpolation \mathfrak{P}_h has to fulfill the pattern condition (103) and we take \mathfrak{P}_h^T as restriction. The coarse grid system can be solved directly (either in parallel or sequentially on one processor) or again by some iterative method similar to the parallel CG. We change the notation of level index from subscript K_j to superscript K^j to emphasize that now $j = 1$ denotes the fine grid.

Algorithm 9 Parallel multigrid $\underline{\mathbf{u}} \leftarrow \text{PMG}(\{\mathbf{K}^j, \mathfrak{P}^j\}, \underline{\mathbf{u}}, \underline{\mathbf{f}}, j)$

```

if  $j == \text{COARSELEVEL}$  then
   $\underline{\mathbf{u}} \leftarrow \text{SOLVE}(\mathbf{K}^j \cdot \underline{\mathbf{u}} = \underline{\mathbf{f}})$ 
else
   $\tilde{\underline{\mathbf{u}}} \leftarrow \text{SMOOTH}(\mathbf{K}^j, \underline{\mathbf{u}}, \underline{\mathbf{f}})$ 
   $\underline{\mathbf{d}} \leftarrow \underline{\mathbf{f}} - \mathbf{K}^j \cdot \tilde{\underline{\mathbf{u}}}$ 
   $\underline{\mathbf{d}}_H \leftarrow (\mathfrak{P}^j)^T \cdot \underline{\mathbf{d}}$ 
   $\underline{\mathbf{w}}_H \leftarrow 0$ 
   $\underline{\mathbf{w}}_H \leftarrow \text{PMG}(\{\mathbf{K}^j, \mathfrak{P}^j\}, \underline{\mathbf{w}}_H, \underline{\mathbf{d}}_H, j + 1)$ 
   $\underline{\mathbf{w}} \leftarrow \mathfrak{P}^j \cdot \underline{\mathbf{w}}_H$ 
   $\tilde{\underline{\mathbf{u}}} \leftarrow \tilde{\underline{\mathbf{u}}} + \underline{\mathbf{w}}$ 
   $\underline{\mathbf{u}} \leftarrow \text{SMOOTH}^T(\mathbf{K}^j, \tilde{\underline{\mathbf{u}}}, \underline{\mathbf{f}})$ 
end if

```

All steps in Algorithm 9 with single-lined arrows \leftarrow are local on each processor using local data only. On the other hand the use of a double-lined arrow \Leftarrow indicates that communication is involved. Despite the coarse grid solver, only the smoothing sweeps require communication.

5.3 Parallel Setup for AMG

In the design of our parallel AMG code we look for components which minimize the communication overhead without increasing the overall effort perceptibly. If the prolongation matrices \mathfrak{P}^ℓ satisfy the pattern condition (103), we conclude from (105) and (106) that all the operations in Algorithm 12 and Algorithm 9 involving \mathfrak{P}^ℓ are local. The pattern of \mathfrak{P}^ℓ is controlled by the strong connections $\{S^{i,T}\}^\ell$ as input parameters in the function WEIGHTS calculating the prolongation weights according to §4.3. Therefore, we have to control the sets of strong connections in the coarsening step to ensure the required interpolation pattern, see also [31].

5.3.1 Communication groups and node sets

Condition (103) is a good criteria for checking an already given matrix. But for AMG we need to derive a constructive algorithm from that criteria for the construction of the prolongation operators \mathfrak{P}^ℓ . Obviously, each node belongs to a communication group σ_k which is characterized by the identical set of subdomains for all group members. Now, subset relations between these communication groups as in condition (103) are used to derive the admissible pattern for prolongation. These subset relations will group the nodes into 4 disjoint sets of nodes with respect to that communication group σ_k . We define the set of all communication groups by

$$\Sigma = \{\sigma_k\}_k := \bigcup_{i \in \omega_h} \sigma^{[i]}. \quad (107)$$

The elements of Σ are uniquely ordered by the rules

1. $|\sigma_k| > |\sigma_m| \Rightarrow k < m$,
2. $|\sigma_k| = |\sigma_m|$ and σ_k lexicographically larger than $\sigma_m \Rightarrow k < m$.

This ordering prevents communication deadlocks in the algorithm. Furthermore we introduce the node sets for a certain communication group $\sigma_k \in \Sigma$:

- (a) active nodes: $\omega_a := \{i \in \omega_h : \sigma^{[i]} \equiv \sigma_k\}$,
- (b) visible nodes: $\omega_v := \{i \in \omega_h : \sigma^{[i]} \subset \sigma_k\}$,
- (c) grouped nodes: $\omega_g := \{i \in \omega_h : \sigma^{[i]} \supset \sigma_k\}$,
- (d) invisible nodes : $\{i \in \omega_h : \sigma^{[i]} \not\subset \sigma_k \wedge \sigma^{[i]} \not\supset \sigma_k\} \equiv \omega_h \setminus (\omega_a \cup \omega_v \cup \omega_g)$.

The model example in Fig. 12 using a linear FE discretization possesses by definition (95) the communication groups $\{1, 2, 3, 4\}$, $\{1, 2\}$, $\{1, 3\}$, $\{2, 4\}$, $\{3, 4\}$ and trivially $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$. Further, we get for subdomain 1, i.e., Ω_1 , four groups where $\sigma_1 = \{1, 2, 3, 4\}$, $\sigma_2 = \{1, 2\}$, $\sigma_3 = \{1, 3\}$ and $\sigma_4 = \{1\}$. The last one does not require any communication but simplifies the algorithms. It is obvious from Fig. 12, that $\sigma_1 = \sigma^{[25]}$, $\sigma_2 = \sigma^{[18]}$, $\sigma_3 = \sigma^{[24]}$ and $\sigma_4 = \sigma^{[1]}$.

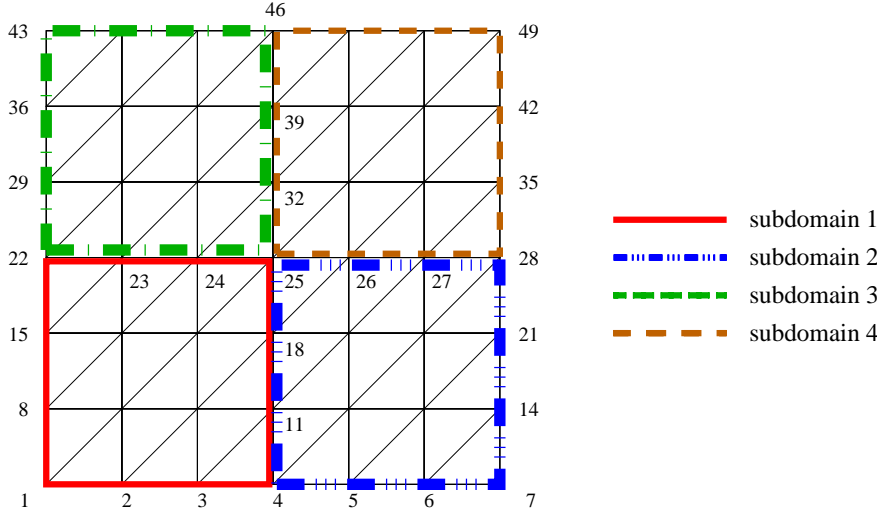


Figure 12: Non-overlapping subdomains with triangulation and row-wise numbering.

In addition we obtain for subdomain Ω_1 with respect to $\sigma_3 = \{1, 3\}$ the following derived node sets $\omega_a = \{22, 23, 24\}$, $\omega_g = \{25\}$ and $\omega_v = \{1, 2, 3, 8, 9, 10, 15, 16, 17\}$, and the invisible nodes $\{4, 11, 18\}$.

5.3.2 Parallel coarsening

The determination of strong connections as well as the calculation of interpolation weights require the accumulated matrix entries which cannot be provided by the distributed matrix \mathbf{K} . Therefore, we have to store the stiffness matrix temporarily twice, i.e., as distributed matrix and as a temporary accumulated matrix. The latter one is locally stored as $\tilde{\mathbf{K}}_s = \mathcal{A}_s \tilde{\mathbf{K}} \mathcal{A}_s^T$.

A first step to control the admissible interpolation pattern consists in changing the accumulated matrix such that (103) is fulfilled

$$\tilde{\mathbf{K}} = \sum_{s=1}^P \mathcal{A}_s^T \tilde{\mathbf{K}}_s \mathcal{A}_s$$

$$\text{with} \quad \tilde{\mathbf{K}}_s^{[i,j]} = \begin{cases} \mathbf{K}_s^{[i,j]} & \text{iff } \sigma^{[i]} \subseteq \sigma^{[j]} \vee \sigma^{[i]} \supseteq \sigma^{[j]} \\ 0 & \text{else} \end{cases}, \quad (108)$$

where $\tilde{\mathbf{K}}_s^{[i,j]}$ is introduced only for notation purposes. This means for Fig. 12, that the resulting matrix $\tilde{\mathbf{K}}$ has zero entries $\tilde{\mathbf{K}}^{[18,26]} = \tilde{\mathbf{K}}^{[24,32]} = 0$ although the original entries may be $\mathbf{K}^{[18,26]} \neq 0$, $\mathbf{K}^{[24,32]} \neq 0$. The same holds for the transposed entries. The sets of strong connections S^i , $S^{i,T}$ can be determined by the sequential function $coarse(i, j, \tilde{\mathbf{K}}_s)$ (see §4.4) and automatically, these sets do not contain forbidden node connections.

The second step takes advantage of the fact that our ordering of the communication groups from §5.3.1 provides already the right relation between the nodes due to ω_a , ω_v and ω_g . The local coarsening will be done subsequently on the active sets of all communication groups σ_k . We have to guarantee coherence of the resulting coarse and fine nodes for active sets shared by more than one subdomain. The easiest way consists in a coarsening only

Algorithm 10 Coarsening $(\omega_C, \omega_F) \leftarrow \text{COARSEP}(\{S_h^{i,T}\}, \omega_m, \omega_a, \omega_v, \omega_C, \omega_F)$

```

while  $\omega_C \cup \omega_F \not\supseteq \omega_a$  do
   $k \leftarrow \text{PICK}(\{S_h^{i,T}\}, \omega_m \setminus (\omega_C \cup \omega_F), \omega_F)$ 
  if  $|S_h^{k,T}| + |S_h^{k,T} \cap \omega_F| = 0$  then
     $\omega_F \leftarrow \omega_F \cup (\omega_a \setminus \omega_C)$ 
  else
     $\omega_C \leftarrow \omega_C \cup \{k\}$ 
     $\omega_F \leftarrow \omega_F \cup ((S_h^{k,T} \setminus \omega_C) \cap \omega_v)$ 
  end if
end while

```

on the root process of communication group σ_k , broadcast the result within σ_k and do the coarsening with the marked coarse nodes ω_m on the remaining processes. The two coarsening parts before and after the communication can be handled by the same routine

$$(\omega_C, \omega_F) \leftarrow \text{COARSEP}(\{S_h^{i,T}\}, \omega_m, \omega_a, \omega_v, \omega_C, \omega_F),$$

see Algorithm 10, which can be derived from the sequential version. The sequential routine is contained in the parallel one via the call

$$(\omega_C, \omega_F) \leftarrow \text{COARSEP}(\{S_h^{i,T}\}, \omega_h, \omega_h, \omega_h, \omega_C, \omega_F).$$

A third step has to restrict the strong connections such that no fine node from the active set ω_a has connections to the visible nodes from ω_v . These three steps guarantee the admissible pattern of the prolongation matrix for operation (105). Algorithm 11 presents the routine for parallel coarsening on each subdomain Ω_s .

5.3.3 Coarse grid matrix and parallel setup

The computation of the interpolation matrix \mathfrak{P}_h is based on the set of strong connections $\{S_h^{i,T}\}$. That is why \mathfrak{P}_h fulfills the pattern condition (103) by construction. Therefore, we call

$$\mathfrak{P}_h \leftarrow \text{WEIGHTS}(\{S_h^{i,T}\}, \tilde{\mathfrak{R}}, \omega_C, \omega_F)$$

on each processor without any communication. Moreover, we can apply (106) with the distributed fine grid matrix \mathbf{K}_h so that the coarse matrix \mathbf{K}_H is again distributed stored and can be calculated fully in parallel without communication, i.e., we call

$$\mathbf{K}_H \leftarrow \mathfrak{P}_h^T \mathbf{K}_h \mathfrak{P}_h$$

locally on each processor.

Collecting the subroutines of previous sections we obtain Algorithm 12 which realizes the parallel setup with the call $\text{PARSETUP}(\mathbf{K}^1, \omega^1, 1)$ on the finest level. It is to be executed on all processors. The coarsening part in Algorithm 12 terminates if the global number of coarse nodes $|\omega|$ becomes smaller than COARSEGRID , cf. the sequential part. The operators \mathcal{A}_s^ℓ denote the subdomain-connectivity matrices with respect to level ℓ .

Algorithm 11 Parallel coarsening $(\{S_h^{i,T}\}, \omega_C, \omega_F) \leftarrow \text{PARCOARSE}(\{S_h^{i,T}\}, \omega_h)$

Determine the set of communication groups Σ
 $\omega_C \leftarrow \emptyset, \quad \omega_F \leftarrow \emptyset$
for all $k = 1, \dots, |\Sigma|$ **do**
 $\omega_a \leftarrow \{i \in \omega_h : \sigma^{[i]} \equiv \sigma_k\}$
 $\omega_v \leftarrow \{i \in \omega_h : \sigma^{[i]} \subset \sigma_k\}$
 $\omega_g \leftarrow \{i \in \omega_h : \sigma^{[i]} \supset \sigma_k\}$
 if $\text{ROOT}(\sigma_k)$ **then**
 $(\omega_C, \omega_F) \leftarrow \text{COARSEP}(\{S_h^{i,T}\}, \omega_a, \omega_a, \omega_v, \omega_C, \omega_F)$
 $\omega_m \leftarrow \omega_C \cap \omega_a$
 end if
 $\omega_m \Leftarrow \text{BROADCAST}(\sigma_k, \omega_m)$
 if not($\text{ROOT}(\sigma_k)$) **then**
 $(\omega_C, \omega_F) \leftarrow \text{COARSEP}(\{S_h^{i,T}\}, \omega_m, \omega_a, \omega_v, \omega_C, \omega_F)$
 end if
 for all $i \in \omega_a$ **do**
 if $i \in \omega_F$ **then**
 $S_h^{i,T} \leftarrow S_h^{i,T} \cap (\omega_a \cup \omega_g)$
 else
 $S_h^{i,T} \leftarrow S_h^{i,T} \cap (\omega_a \cup \omega_v)$
 end if
 end for
end for

Algorithm 12 Parallel Setup $\{\mathcal{K}^j, \mathfrak{P}^j\}_{j=1}^{\text{COARSELEVEL}} \leftarrow \text{PARSETUP}(\mathcal{K}_h, \omega_h, \ell)$

if $|\omega| > \text{COARSEGRID}$ **then**
 $\mathcal{K}^\ell \leftarrow \mathcal{K}_h$
 $\tilde{\mathfrak{K}} \Leftarrow \left(\sum_{s=1}^P (\mathcal{A}_s^\ell)^T \tilde{\mathcal{K}}_s \mathcal{A}_s^\ell \right)_{\text{pattern}}$
 $\{S^{i,T}\}^\ell \leftarrow \text{GETSTRONG}(\tilde{\mathfrak{K}})$
 $(\{S^{i,T}\}^\ell, \omega_C, \omega_F) \Leftarrow \text{PARCOARSE}(\{S^{i,T}\}^\ell, \omega^\ell)$
 $\mathfrak{P}^\ell \leftarrow \text{WEIGHTS}(\{S^{i,T}\}^\ell, \tilde{\mathfrak{K}}, \omega_C, \omega_F)$
 $\mathcal{K}_H \leftarrow (\mathfrak{P}^\ell)^T \mathcal{K}^\ell \mathfrak{P}^\ell$
 $\omega_H \leftarrow \omega_C$
 $\text{PARSETUP}(\mathcal{K}_H, \omega_H, \ell + 1)$
else
 $\text{COARSELEVEL} \leftarrow \ell$
end if

6 Applications

6.1 Potential Equation in the Human Head

People with epileptic fits can only be cured by removing those parts of their brain causing the fit. The main difficulty consists in identifying location and size of the malfunctioning

brain-matter, called medical source reconstruction. Therein, one has to solve about 10.000 times the potential equation

$$-\operatorname{div}(k(x, y, z)\nabla u(x, y, z)) = f(x, y, z) \quad (109)$$

in the human head Ω with Neumann boundary conditions on $\partial\Omega$ and the source distribution $f(x, y, z)$. Several hundred thousand tetrahedra elements are necessary to reflect the main characteristics of the human head in the discrete model in Fig. 13. The conductivity $k(x, y, z)$ varies from 0.33 $[1/\Omega m]$ in all skin- and brain-elements, over 0.0042 $[1/\Omega m]$ in the skull elements to 1.0 $[1/\Omega m]$ in the CSF, i.e., within the layer between brain and skull and within the ventricular system [1].

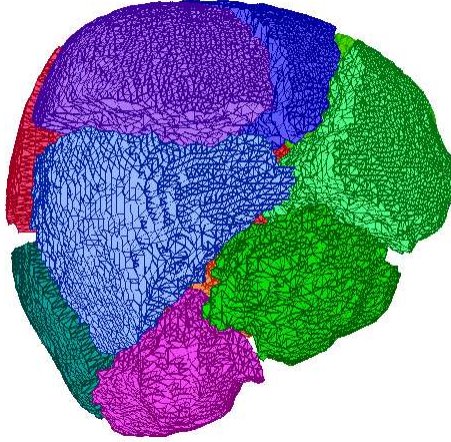


Figure 13: FE-meshes of human head, partitioned for 12 processors with METIS and visualized with PMVIS: 713733 tetrahedras and 118299 nodes

We run the experiment on an SGI ORIGIN 200 with R10000, 195 MHz processors and overall 6GB of main memory and solved the finite element equations derived from (109) with an AMG-preconditioned CG using one $V(1,1)$ -cycle in the AMG-preconditioner. The local accumulation of the matrix on one processor took 173.4 seconds, parallelized on 12 processors a setup time of 14.89 seconds was achieved.

Figure 14 shows the wall-clock time (WCPU) of our parallel AMG-preconditioned CG solver PEBBLES [60] compared to the standard parallel Jacobi-preconditioned CG. The number of iterations for both solvers, necessary for the required accuracy, is indicated in the graphs. The time for the setup of the preconditioner is not included, since it has to be carried out only once per head model and is thus negligible with regard to the solution of the inverse problem. To give an impression, the setup of the AMG on 1 processor took 29.9 seconds and parallelized on 12 processors 7.4 seconds. The 3D potential distribution was calculated on one processor within 195.8 seconds with the Jacobi-CG method, whereas the parallel AMG-CG method on 12 processors needed 2.6 seconds. This is a factor of about 75 (7.5 through multi-level preconditioning and 10 through parallelization).

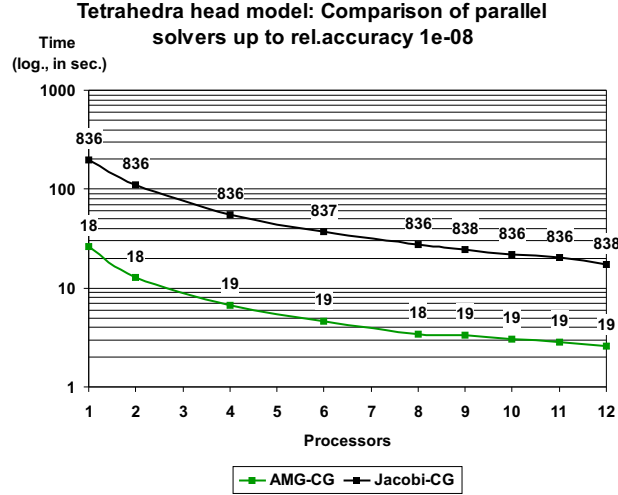


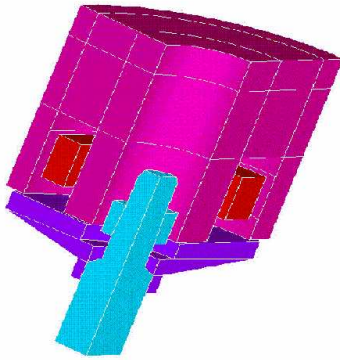
Figure 14: *WCPU* with respect to number of processors for the solver part. The numbers of iterations are shown with the graphs.

6.2 Magnetostatic Valve

One cutting edge in combustion engine design is the development of very fast valves for more fuel injections into the combustion chamber than nowadays per cycle. Magnetic valves are one candidate to achieve super fast valves. One major part in the simulation of these components consists in solving the 3D magnetostatic Maxwell equations

$$\mathbf{curl}(\nu \mathbf{curl} \mathbf{u}) + \sigma \mathbf{u} = \mathbf{f} \quad (110)$$

in the computational domain Ω with appropriate boundary conditions on $\partial\Omega$, where $\sigma = 10^{-4}$, $\nu(x, y, z)$ and $\mathbf{f}(x, y, z)$ are given [51]. Figure 15 presents the mesh of the FEM discretization



N_h	it	setup [sec.]	solver [sec.]
8714	14	0.53	2.49
65219	28	4.17	44.01
504246	63	34.49	792.49

Figure 15: Material distribution in one quarter of the valve, Numerical results.

with Nedelec-elements containing half a million unknowns and the iteration counts for an

AMG preconditioned CG with the stopping accuracy $\varepsilon = 10^{-8}$. The code PEBBLES runs on an SGI Octane, R12000 300 MHz. We mention that the discrete Maxwell equations require special AMG components [61].

6.3 C-magnet: Geometric versus Algebraic Multigrid

Let us consider the C-magnet depicted in Fig 16. The magnet field is generated by a current in the coil and can be described with (110). The coarse mesh contains 2907 unknowns and the the fine mesh 180426 unknowns after 2 refinement steps in each tetrahedral element. We



Figure 16: C-Magnet: geometry and mesh with 153408 elements and 206549 nodes.

want to compare algebraic multigrid applied to the fine grid problem with geometric multigrid (GMG) using the hierarchy of the three meshes. The hybrid smoother proposed in [46] is used, see also [32]. The required decomposition into subdomains for solving the problem on a parallel computer (SGI ORIGIN 2000, 300 MHz) has been obtained by recursive spectral bisection. In order to analyze the parallel performance we consider the following components of the algorithm: generation of the system matrix, setup and solver, see Table 2.

Table 2: Number of processors (P), wall clock time (T) in seconds for generating the system matrix, the setup phase and the solver. Corresponding speedup (S), number of iterations (It.) and speedup with respect to 1 iteration (S(1)).

P			Geometric MG						Algebraic MG					
	SysMat		Setup		Solver				Setup		Solver			
	T	S	T	S	It.	T	S		T	S	It.	T	S	S(1)
1	40.3	1.0	13.1	1.0	13	99.4	1.0		44.4	1.0	60	247.3	1.0	1.0
2	19.6	2.0	10.8	1.2	13	69.2	1.4		24.4	1.8	52	127.1	1.9	1.7
4	9.7	4.1	9.1	1.4	13	31.8	3.1		12.6	3.5	52	48.9	5.0	4.3
8	4.7	8.5	8.8	1.5	13	16.2	6.1		7.6	5.8	58	28.2	8.8	8.4

As expected, the generation of the system matrix shows optimal speedup since this component is free of communication. In the case of GMG, the setup phase involves the setup of the smoother and the LU-decomposition of the coarse-grid system matrix with 2907 unknowns. Since the setup is dominated by the sequential LU-decomposition, only low speedups

can be observed. In the case of AMG, the setup involves the coarsening itself and the LU-decomposition of the coarse-grid system which has less than 500 unknowns. Hence, the setup shows reasonable speedups since it is dominated by the coarsening of the inner nodes. In both cases, AMG and GMG, the solver shows a similar speedup behavior. Although most of the gain is due to the additional CPU capacity, the additional cache which comes with each processor also contributes to the acceleration. We refer to [33] for more detailed information.

Acknowledgements

The authors would like to thank the NATO Advanced Study Institute and the Austrian Science Found (FWF) for supporting this contribution to the summer workshop on “Modern Methods in Scientific Computing and Applications”. Special thanks are given to Johanna Kienesberger and Stefan Reitzinger who helped us in preparing the lectures and these lecture notes. We are also indebted to Carsten Wolters (Max Planck Institute of Cognitive Neuroscience Leipzig, Germany) and to Manfred Kaltenbacher (Department of Sensor Technology at the University of Erlangen-Nürnberg, Germany) for providing the examples used in the numerical experiments with our AMG code PEBBELS in §6.1 and §6.2, respectively.

References

- [1] A. Anwander, M. Kuhn, S. Reitzinger, and C. Wolters. A parallel algebraic multigrid solver for the finite element method source localization in the human brain. *Computing and Visualization in Science*, 2002. (to appear).
- [2] G.P. Astrachancev. An iterative method for solving elliptic net problems. *USSR Comput. Math. math. Phys.*, 11(2):171–182, 1971.
- [3] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, 1994.
- [4] N.S. Bachvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Comput. Math. Math. Phys.*, 6(5):101–135, 1966.
- [5] R. Bank. *PLTMG: A software package for solving elliptic partial differential equations*. SIAM, Philadelphia, Philadelphia, 1994.
- [6] B. Bastian. *Parallele Adaptive Mehrgitterverfahren*. Teubner Skripten zur Numerik, B.G. Teubner Stuttgart, Stuttgart, 1996.
- [7] P. Bastian. UG version 2.0 – short manual. Preprint 92–14, IWR Heidelberg, 1992.
- [8] F.A. Bornemann and P. Deuffhard. The cascadic multigrid method for elliptic problems. *Numer. Math.*, 75:135–152, 1996.
- [9] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.
- [10] D. Braess. *Finite Elements: Theory, Fast Solvers and Applications in Solid Mechanics*. Cambridge University Press, Cambridge, 1997.

- [11] D. Braess, M. Dryja, and W. Hackbusch. A multigrid method for nonconforming fe-discretisations with application to non-matching grids. *Computing*, 63:1–25, 1999.
- [12] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *SIAM J. Numer. Anal.*, 20:967–975, 1983.
- [13] J. H. Bramble. *Multigrid methods*. Pitman Research Notes in Mathematics, Longman Scientific and Technical, London, 1993.
- [14] J. H. Bramble and J. E. Pasciak. New convergence estimates for multigrid algorithms. *Math. Comput.*, 49(180):311–329, 1987.
- [15] J. H. Bramble and J. E. Pasciak. New estimates for multilevel algorithms including the V-cycle. *Math. Comput.*, 60:447–471, 1993.
- [16] A. Brandt. Multi-level adaptive techniques (mlat) for fast numerical solution to boundary value problems. In *Lecture Notes in Physics*, 1973. Proc. 3rd Internat. Conf. on Numerical Methods in Fluid Mechanics, Paris, 1972.
- [17] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comput.*, 31:333–390, 1977.
- [18] A. Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.*, 19:23–56, 1986.
- [19] A. Brandt. Multiscale scientific computation: Review 2001. In T. Barth, R. Haimes, and T. Chan, editors, *Multiscale and Multiresolution Methods*. Springer-Verlag, Berlin-Heidelberg-New York, 2001. Proceeding of the Yosemite Educational Symposium, October 2000.
- [20] A. Brandt, S.F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D.J. Evans, editor, *Sparcity and its Application*, pages 257–284, 1984.
- [21] A. Brandt, S.F. McCormick, and J.W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application in geodetic computations. Technical Report CO POB 1852, Inst. Comp. Studies State Univ., 1982.
- [22] S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer Verlag, New York, 1994.
- [23] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge. Algebraic multigrid based on element interpolation (AMGe). *SIAM J. on Scientific Computing*, 22(5):1570–1592, 2000.
- [24] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2000. Second Edition.
- [25] P. Deuffhard. Cascadic conjugate gradient methods for elliptic partial differential equations I: Algorithm and numerical results. Preprint SC 93–23, Konrad-Zuse-Zentrum fuer Informationstechnik Berlin, 1993.

- [26] E. Dick, K. Rienslagh, and J. Vierendeels, editors. *Multigrid Methods VI. Proceedings of the Sixth European Multigrid Conference*, Berlin, 2000. Springer. Gent, September 27-30, 1999.
- [27] C. Douglas, G. Haase, and U. Langer. *A Tutorial on Elliptic PDEs and Parallel Solution Methods*. SIAM, 2002. in preparation.
- [28] R.P. Fedorenko. A relaxation method for elliptic difference equations. *USSR Comput. Math. math. Phys.*, 1(5):1092–1096, 1961.
- [29] R.P. Fedorenko. The speed of convergence of one iterative process. *USSR Comput. Math. math. Phys.*, 4(3):227–235, 1964.
- [30] T. Grauschopf, M. Griebel, and H. Regler. Additive Multilevel–Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic–Multigrid Coarsening for Second Order Elliptic PDEs. SFB–Bericht Nr. 342/02/96, TU München, 1996.
- [31] G. Haase. A parallel amg for overlapping and non-overlapping domain decomposition. *Electronic Transactions on Numerical Analysis (ETNA)*, 10:41–55, 2000.
- [32] G. Haase, M. Kuhn, and U. Langer. Parallel multigrid 3d Maxwell solvers. *Parallel Computing*, 6(27):761–775, 2001.
- [33] G. Haase, M. Kuhn, U. Langer, S. Reitzinger, and J. Schöberl. Parallel Maxwell solvers. In U. van Rienen, M. Günther, and D. Hecht, editors, *Scientific Computing in Electrical Engineering*, pages 71–78, Berlin-Heidelberg-New York, 2000. Springer-Verlag.
- [34] G. Haase, M. Kuhn, and S. Reitzinger. Parallel AMG on distributed memory computers. *SIAM SISC*, 2002. accepted for publication.
- [35] G. Haase, U. Langer, and A. Meyer. The approximate Dirichlet decomposition method. part I,II. *Computing*, 47:137–167, 1991.
- [36] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl. Algebraic multigrid methods based on element preconditioning. *Int. J. of Computer Mathematics*, 80(3-4), 2001.
- [37] W. Hackbusch. Implementation of the multi-grid method for solving partial differential equations. Technical Report RA 82, IBM T.J. Watson Research Centre, 1976.
- [38] W. Hackbusch. *Multigrid Methods and Applications*. Springer–Verlag, Berlin, 1985.
- [39] W. Hackbusch and U. Trottenberg, editors. *First European Conference on Multigrid Methods*, Berlin, Heidelberg, New York, 1982. Lecture Notes in Mathematics, v. 960, Springer-Verlag. Köln-Porz, November 23-27, 1981.
- [40] W. Hackbusch and U. Trottenberg, editors. *Second European Conference on Multigrid Methods*, Berlin, Heidelberg, New York, 1985. Lecture Notes in Mathematics, v. 1228, Springer-Verlag. Köln, Oktober 1-4, 1985.
- [41] W. Hackbusch and U. Trottenberg, editors. *Third European Conference on Multigrid Methods*, Basel, Boston, Berlin, 1991. ISNM 98, Birkhäuser. Bonn, Oktober 1-4, 1990.

- [42] W. Hackbusch and G. Wittum, editors. *Multigrid Methods V. Proceedings of the Fifth European Multigrid Conference*, Berlin, 1998. Springer. Stuttgart, 1996.
- [43] P.W. Hemker and P. Wesseling, editors. *Multigrid Methods IV. Proceedings of the Fourth European Multigrid Conference*, Basel, 1994. Birkhäuser. Amsterdam, July 6-9, 1993.
- [44] V. Henson and P. Vassilevski. Element-free AMGe: General algorithms for computing interpolation weights in AMG. *SIAM J. on Scientific Computing*, 23(2):629–650, 2001.
- [45] V.E. Henson and U.M. Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. Technical Report UCRL-JC-141495, Lawrence Livermore National Laboratory, 2000.
- [46] R. Hiptmair. Multigrid methods for Maxwell’s equations. *SIAM J. Numer. Anal.*, 36:204–225, 1999.
- [47] J. Jones and P. Vassilevski. AMGe based on element agglomeration. *SIAM J. on Scientific Computing*, 23(1):109–133, 2001.
- [48] M. Jung. On the parallelization of multi-grid methods using a non-overlapping domain decomposition data structure. *Applied Numerical Mathematics*, 23(1):119–137, 1997.
- [49] M. Jung and U. Langer. Applications of multilevel methods to practical problems. *Surveys on Mathematics for Industry*, 1:217–257, 1991.
- [50] M. Jung, U. Langer, A. Meyer, W. Queck, and M. Schneider. Multigrid preconditioners and their applications. In G. Telschow, editor, *Proceedings of the "3rd Multigrid Seminar" held at Biesenthal, GDR, May 1989*, pages 11–52, Berlin, 1989. Karl-Weierstrass-Institute of the Academy of Science of the GDR. Report-Nr. R-MATH-03/89.
- [51] M. Kaltenbacher, S. Reitzinger, and J. Schöberl. Algebraic multigrid for solving 3D nonlinear electrostatic and magnetostatic field problems. *IEEE Trans.on Magnetics*, 36(4):1561–1564, 2000.
- [52] F. Kicking. Algebraic multigrid for discrete elliptic second-order problems. In W. Hackbusch, editor, *Multigrid Methods V. Proceedings of the 5th European Multigrid conference, Stuttgart 1996*, pages 157–172. Berlin: Springer Lecture Notes Comput. Sci. Eng. 3, 1998.
- [53] V.G. Korneev. *Finite element schemes of higher order of accuracy (in Russian)*. Leningrad University Press, Leningrad, 1977.
- [54] A. Krechel and K. Stüben. Parallel algebraic multigrid based on subdomain blocking. *Parallel Computing*, 8(27):1009–1031, 2001.
- [55] M. Kuhn, U. Langer, and J. Schöberl. Scientific computing tools for 3d magnetic field problems. In J.R. Witheman, editor, *The Mathematics of Finite Elements and Applications*, pages 239–258. Elsevier, Amsterdam, 2000. Proceeding of the MAFELAP X, Brunel University, 22–25 June, 1999.
- [56] U. Langer. On the choice of iterative parameters in the relaxation method on a sequence of meshes. *USSR Comput. Math. Math. Phys.*, 22(5):98–114, 1982.

- [57] K. H. Law. A parallel finite element solution method. *Computer and Structures*, 23(6):845–858, 1989.
- [58] S. McCormick. *Multilevel adaptive methods for partial differential equations*. SIAM, Frontiers in Applied Mathematics, Philadelphia, 1989.
- [59] G. Meurant. *Computer Solution of Large Systems*. North-Holland, Amsterdam, 1999.
- [60] S. Reitzinger. *Algebraic Multigrid Methods for Large Scale Finite Element Equations*. Universitätsverlag Rudolf Trauner, Linz, 2001.
- [61] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretization with edge elements. *Numer. Linear Algebra Appl.*, 2002. (accepted).
- [62] U. Rüde. *Mathematical and Computational Techniques for Multilevel Adaptive methods*. SIAM, Frontiers in Applied Mathematics, Philadelphia, 1993.
- [63] J. W. Ruge and K. Stüben. Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). In D. J. Paddon and H. Holstein, editors, *Multigrid Methods for Integral and Differential Equations*, The Institute of Mathematics and its Applications Conference Series, pages 169–212. Clarendon Press, Oxford, 1985.
- [64] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.
- [65] J. Schöberl. Multigrid methods for a parameter dependent problem in primal variables. *Numer. Math.*, 84:97–119, 1999.
- [66] V.V. Shaidurov. *Multigrid Methods for Finite Elements*. Kluwer, Dordrecht, 1995.
- [67] K. Stüben. Algebraic multigrid: An introduction with applications. In U. Trottenberg, C. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–532. Academic Press, 2000.
- [68] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128:281–309, 2001.
- [69] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2000.
- [70] St. Vandewalle. *Parallel Multigrid Waveform Relaxation for Parabolic Problems*. B.G. Teubner, Stuttgart, 1993.
- [71] P. Vaněk. Acceleration of convergence of a two level algorithm by smoothing transfer operators. *Appl. Math.*, 37:265–274, 1992.
- [72] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid based on smoothed aggregation for second and fourth order problems. *Computing*, 56:179–196, 1996.
- [73] P. Wesseling. *An introduction to Multigrid Methods*. John Wiley, Chichester, 1992.
- [74] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, 0:285–326, 1993.