AMGe — Coarsening Strategies and Application to the Oseen-Equations

Markus Wabro *
Institute of Computational Mathematics
J.K. University Linz, Austria

June 22, 2004

Abstract

We provide some extensions to the AMGe method (algebraic multigrid method based on element interpolation), concerning the agglomeration process, the application to non-conforming elements, and the application to the mixed finite element discretization of the Oseen-linearized Navier-Stokes equations.

This last point, using AMGe for mixed finite elements, gets straight-forward because of the availability of coarse level topologies. We show this exemplarily for the Crouzeix-Raviart element (including a stability result).

The numerical results show that it really pays off to take a closer look at the agglomeration strategy, a 'wrong' choice can lead to insufficient convergence or even divergence of the overall multigrid method.

1 Introduction

The algebraic multigrid method (AMG) was introduced by Ruge and Stüben [RS86] and Brandt et al. [BMR84] mainly for two reasons. First, one obtains a multilevel method without the need of further refinement, which is helpful in situations with large problems and limited amount of computer memory. Second, it is intended to be used as "black-box" solver, e.g. as coarse-grid solver in some geometric multigrid (GMG) methods. An overview of the technique itself and its applications can be found e.g. in Stüben [Stü01].

Unfortunately there are situations (especially when the operators are not represented by symmetric, positive definite M-matrices) where classical AMG might fail. For finite element (FE) discretized problems one way out was the development of the *algebraic multigrid method based on element interpolation* (AMGe).

*Supported by the Austrian Science Foundation "Fonds zur Förderung der wissenschaftlichen Forschung (FWF)"

Up to now this method has undergone three important stages of development. It was introduced by Brezina et al. [BCF$^+$00], where still the classical techniques for the coarse node / fine node (C/F) splitting are used, but a new method of constructing the interpolation operator, based on the local element stiffness matrices. In Jones, Vassilevski [JV01] the coarse grid construction is changed in a way that allows to extract topological information on coarse levels (coarse faces, edges, and elements and their connectivity). The interpolation still depends on the element stiffness matrices, which are assembled to Neumann-type stiffness matrices on local patches (e.g. the coarse-level elements) and which have to be stored during the whole set-up process. This problem is overcome by the approach in Henson, Vassilevski [HV01]. Here, an extension- (extrapolation-) operator (only depending on the assembled stiffness matrix) is used to construct an approximation to the local Neumann-type matrices.

The full availability of the topological information on all levels opens up the possibility of constructing AMGe methods for various types of FE discretizations, especially for mixed FE-discretizations of saddle point problems.

We will exploit this property when we construct a coupled AMGe (cAMGe) method for the Oseen-linearized Navier-Stokes equations. 'Coupled' here is meant in contrast to methods, where pressure and velocity equations are iteratively decoupled (e.g. using SIMPLE or Uzawa methods) and AMG is used for the resulting scalar problems. We want to have an "all-at-once" approach, although we then have to deal with an indefinite system and with possible (inf-sup) stability problems. Some possibilities of coupled AMG (not AMGe) methods for the Oseen problem can be found in [Wabar], in [Wab03] there are also first steps towards cAMGe.

In this article we will exemplarily construct a cAMGe method for the P_1^{nc}-P_0 discretization of the coupled problem, and show its coarse-level stability.

Besides the development of this coupled AMGe method, the second emphasis of this work is put on the construction of the coarse levels, i.e. how to agglomerate fine-level elements to coarse-level elements. As we will see, this has a crucial influence on the performance of the method.

The article is organized as follows. After describing the main ingredients of an (element agglomerating) AMGe method, we present three new techniques for the agglomeration process, and we show how to build the interpolations for three different FE-types. In the third section we apply the ideas to the Crouzeix-Raviart-discretized Oseen-equations and derive a stability result. The article is concluded with numerical experiments for the presented methods.

1.1 Notation

We want to construct an AMG method for a set of linear equations

$$K_1 x = b_1, \tag{1}$$

where K_1 is a regular $n_1 \times n_1$ matrix and b_1 a given right hand side. The index indicates the level, 1 is the finest level, L will be the coarsest. We create full rank prolongation matrices $P_{l+1}^l : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_l}$, $l = 1, \ldots, L-1$ and $n_1 > n_2 > \ldots > n_L$, and a set of coarse level matrices K_l with

$$K_{l+1} = (P_{l+1}^l)^T K_l P_{l+1}^l. \tag{2}$$

The system in (1) is originating in a FE discretization of some partial differential equation (PDE) on a polygonal resp. polyhedral domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$. We consider finite elements based on triangles resp. tetrahedra, thus we assume that some partitioning of Ω into triangles resp. tetrahedra $\Omega = \bigcup_i \tau_i$ is given, and we denote the set of elements $\mathcal{T}_h := \{\tau_1, \tau_2, \ldots\}$.

2 AMGe Building Blocks

In this section we will present the ingredients of the coarse-level construction of an element agglomerating AMGe method:

1. The C/F splitting, which in most cases consists of
   (a) the element agglomeration process and
   (b) the identification of coarse-level faces, edges, and nodes,
2. and the construction of the interpolation matrix from the coarse degrees of freedom (dofs) to the fine dofs.

2.1 Identification of the Coarse-Level Topology

Assume that on one level (e.g. on the discretization level) we know the element-to-node connectivity, i.e. which nodes are part of a given element. Assume further that a method for the agglomeration of elements is known, satisfying the requirements that each element is part of one unique agglomerate and that each agglomerate is a connected set, meaning that for any two elements part of the the same agglomerate there exists a connected path of elements of this agglomerate connecting the two elements. Then we can apply the following algorithm for the creation of the coarse level topology (as presented in Jones, Vassilevski [JV01]).

Algorithm 1. AMGe coarse level topology

1. Agglomerate the fine elements to coarse elements E_j (with the above properties).
2. Consider all intersections $E_j \cap E_k$ for all pairs of different agglomerated elements E_j and E_k. If such an intersection is maximal, i.e. is not contained in any other intersection, then it is called a face.
3. Consider the faces as sets of nodes. For each node n compute the intersection $\bigcap\{\text{all faces which contain } n\}$. Now the set of minimal, nonempty intersections defines the vertices.

We have formulated the algorithm for the 3D case, but it can be directly applied to 2D problems (then the 'faces' correspond to edges). If (in the 3D case) one additionally wants to identify edges, then this can be done in step 3 using the set of minimal, nonempty intersections which are not already vertices.

2.2 Element Agglomeration Process

For the agglomeration of fine elements to coarse elements we present four different techniques, a classical one and three new possibilities.

2.2.1 Jones-Vassilevski Element Agglomeration

We will start with the 'classical' Jones-Vassilevski algorithm [JV01].

Algorithm 2. Jones-Vassilevski element agglomeration
Assume we have a set of finer level elements $\{e_j\}$ and faces $\{f_j\}$, and introduce an integer weight $w(f_j)$ for each face f_j.

- **initiate.** Set $w(f) \leftarrow 0$ for all faces f;
- **global search.** Find a face f with maximal $w(f)$, if $w(f) = -1$ we are done; set $E \leftarrow \emptyset$;

  1. Set $E \leftarrow E \cup e_1 \cup e_2$, where $e_1 \cap e_2 = f$ and set $w_{\max} \leftarrow w(f)$, $w(f) \leftarrow -1$
  2. Increment $w(f_1) \leftarrow w(f_1) + 1$ for all faces f_1 such that $w(f_1) \neq -1$ and f_1 is a neighbor of f;
  3. Increment $w(f_2) \leftarrow w(f_2) + 1$ for all faces f_2 such that $w(f_2) \neq -1$, f_2 is a neighbor of f, and f_2 and f are faces of a common element;
  4. From the neighbors of f, choose a face g with maximal $w(g)$; if $w(g) \geq w_{\max}$ set $f \leftarrow g$ and go to step 1.;
  5. If all neighbors of f have smaller weight than $w_{\max}$, the agglomerated element E is complete; set $w(g) \leftarrow -1$ for all faces of the elements e contained in E; go to **global search**;

Remark 3. *In [JV01] also modifications to this algorithm are presented which allow some kind of semi-coarsening, i.e. coarsening with the focus in one specific direction (for example determined by convection).*

For the 2D case this algorithm mostly produces nice agglomerated elements, in the 3D case some strange shapes may occur, therefore some adjustments of the algorithm seem to be necessary.

2.2.2 Red-Grey-Black Element Agglomeration

A second method which is fast and produces good looking agglomerates, but often leads to a too strong coarsening (what has a disadvantageous influence on the h-independence) is the following.

Algorithm 4. Red-grey-black element agglomeration

repeat until all elements are colored
  begin
    choose an uncolored element, this is colored black;
    color all uncolored or grey neighboring elements red
      (where 'neighboring' could be induced by faces, edges or nodes);
    color all uncolored elements neighbored to red elements grey;
  end
the black elements plus surrounding red elements build the agglomerated elements;
each grey element is appended to the agglomerate where it "fits best"
  (e.g. to the agglomerate it shares the largest face with);

2.2.3 Optimization Approach

Another class of methods (which originally was aimed at the coarse-level construction of MG methods for finite volume discretizations) was developed by Moulitas and Karypis [MK01b] and implemented in the software package MGridGen [MK01a]. They formulate an optimization problem, where a coarse grid is generated that minimizes some objective function F (i.e. maximizes the quality of the grid in some sense), and the size of the coarse grid elements is subject to some upper and lower bounds. Examples for such objective functions are the sum or the maximum of the aspect ratios of the coarse elements.

The method for solving these optimization problems is based on a multilevel technique described in [MK01b] and the references therein, we sketch it roughly.

Algorithm 5. MgridGen

1. Coarsening phase: A sequence of approximate representations of the optimization problem is generated by constructing coarser and coarser graphs (this sequence of graphs has nothing to do with the levels in our multigrid method!).

2. Initial solution phase: The optimization problem is solved on the coarsest level.

3. Uncoarsening phase: The optimization problem is solved (approximately) on the finer levels, using the solution on the coarser levels.

As the MGridGen library is freely available, the application of the method is particularly easy.

2.2.4 Bottom-Up Graph Partitioning

The three methods above have in common a top-down strategy, i.e. we start on the finest level and agglomerate elements to obtain the next coarser level. The following method will work the other way round, i.e. bottom-up.

Algorithm 6. Bottom-up graph partitioning

1. Build the element-element connectivity graph for the fine level mesh.
2. Utilize any mesh (resp. graph) partitioning algorithm to obtain a (small) number of sub-graphs. This set of sub-graphs represents the coarsest level (each sub-graph is an element).
3. Now apply this strategy recursively on the sub-graphs, until only one-node-graphs remain: the finest level.

Remark 7. *It may happen that the last-but-one level is very close to the finest level. In this case this level should be ignored to avoid a waste of memory-resources.*

For the graph partitioning algorithm mentioned in step 2 of Algorithm 6 we use the METIS-library by Karypis and Kumar [KK98b] (the underlying techniques are explained in [KK98a]).

Lemma 8. *If one of Algorithms 2, 4, or 5 is applied in the construction of an AMGe method, the number of arithmetical operations during the setup phase $\mathcal{Q}_i^S(n_1)$, $i = 2, 4, 5$, is $\mathcal{O}(n_1)$.*

For Algorithm 6 we obtain $\mathcal{Q}_6^S(n_1) = \mathcal{O}\left(n_1 \log(n_1)\right)$.

Proof. Assume for all algorithms a constant coarsening resp. refinement rate r and a minimal number of coarsest grid nodes n_c. Then the number of levels is given by

$$L = \log_r \left(\frac{n_c}{n_1}\right) + 1,$$

the total complexity for Algorithms 2, 4, and 5 (assuming that the complexity on each level is proportional to the number of nodes) calculates as

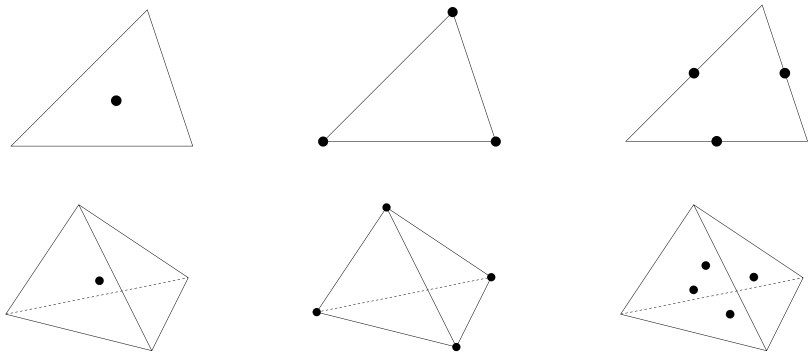
$$\begin{aligned}
\mathcal{Q}_i^S(n_1) &= \mathcal{O}\left(n_1 + rn_1 + \ldots + r^{L-2}n_1\right) \\
&= \mathcal{O}\left(n_1 \cdot \frac{1 - r^{L-1}}{1-r}\right) = \mathcal{O}\left(\frac{n_1 - n_c}{1-r}\right) \\
&= \mathcal{O}\left(n_1\right), \quad \text{for } i = 2, 4, 5.
\end{aligned}$$

For Algorithm 6 the complexity on each level stays constant, thus the overall complexity is calculated as

$$\mathcal{Q}_6^S(n_1) = \mathcal{O}\left((L-1) \cdot n_1\right) = \mathcal{O}\left(n_1 \log n_1\right).$$

□

Figure 1 Some mixed finite elements for triangles and tetrahedra. Dots indicate dofs.



(a) P_0 (b) P_1 (c) P_1^{nc}

2.3 Interpolation

For the construction of the interpolation we can utilize the topological information we have obtained in section 2.1. We will show this exemplarily for three types of finite elements.

2.3.1 P_0-Element

For the finite element with piecewise constant shape functions (as in Figure 1(a)) the construction of the interpolation is particularly easy. Each fine level element is part of exactly one coarse level agglomerate, thus we can use identity-prolongation.

2.3.2 P_1-Element

For piecewise linear, continuous elements (Figure 1(b)) we apply a strategy described in Henson, Vassilevski [HV01], which generalizes the method used in Jones, Vassilevski [JV01].

We first define the neighborhood of a (fine-level) node n by

$$\Omega(n) := \bigcup \{\text{all agglomerated elements that contain } n\}$$

and the minimal set

$$\Lambda(n) := \bigcap \{\text{all agglomerated elements that contain } n\}$$

($\Lambda(n)$ can be a node, edge, face, or element). The set of coarse dofs in $\Omega(n)$ is denoted by $\Omega_c(n)$. Then we define the extended neighborhood

$$\overline{\Omega}(n) := \{j \notin \Omega(n) : k_{ij} \neq 0 \text{ for some } i \in \Omega(n) \setminus \Omega_c(n)\} \cup \Omega(n)$$

(where k_{ij} is the (i, j)-th entry of the fine-level matrix K_l).

As the coarse nodes form a subset of the fine nodes the values there can be identically prolongated. For the edges, faces, and cells we proceed recursively as follows. Assume for a set $\Lambda(n)$ that the interpolation on the unknowns in $\partial\Lambda(n)$ has been fixed,[1] and we want to calculate the interpolation on the nodes in $\Lambda(n) \setminus \partial\Lambda(n)$. For that we build the local matrix of $\Omega(n)$ (extracted from the global matrix K_l) with the underlying partitioning $(\Omega(n) \setminus \partial\Lambda(n)) \cup \partial\Lambda(n) \cup \left(\overline{\Omega}(n) \setminus \Omega(n)\right)$

$$K_{\overline{\Omega}(n)} = \begin{array}{cl} \begin{pmatrix} K_{ii} & K_{ib} & K_{ie} \\ K_{bi} & K_{bb} & K_{be} \\ K_{ei} & K_{eb} & K_{ee} \end{pmatrix} & \begin{array}{l} \} \quad \Omega(n) \setminus \partial\Lambda(n) \\ \} \quad \partial\Lambda(n) \\ \} \quad \overline{\Omega}(n) \setminus \Omega(n) \end{array} \end{array}$$

(i stands for interior, b for boundary, e for extended neighborhood). In [JV01] the small matrix

$$K_{\Omega(n)} = \begin{pmatrix} \tilde{K}_{ii} & \tilde{K}_{ib} \\ \tilde{K}_{bi} & \tilde{K}_{bb} \end{pmatrix} \tag{3}$$

is built using the local stiffness matrices, therefore it is a Neumann-type operator. Extracting it directly from the global stiffness matrix would lead to a Dirichlet-type operator. Thus, in [HV01] an extension mapping $E(n) : \Omega(n) \to \overline{\Omega}(n)$ is built, which with the above partitioning can be written as

$$E(n) = \begin{pmatrix} I & 0 \\ 0 & I \\ E_{ei} & E_{eb} \end{pmatrix},$$

and is used to get something similar to (3)

$$\hat{K}_{\Omega(n)} = \begin{pmatrix} \hat{K}_{ii} & \hat{K}_{ib} \\ \hat{K}_{bi} & \hat{K}_{bb} \end{pmatrix} := \begin{pmatrix} K_{ii} + K_{ie}E_{ei} & K_{ib} + K_{ie}E_{eb} \\ K_{bi} + K_{be}E_{ei} & K_{bb} + K_{be}E_{eb} \end{pmatrix}.$$

Then the prolongation on $\Lambda(n) \setminus \partial\Lambda(n)$ is built by extracting the corresponding entries of

$$-\hat{K}_{ii}^{-1}\hat{K}_{ib}.$$

Remark 9. *For the situation in (3) and for symmetric positive definite K_l the setting*

$$u_i = -\tilde{K}_{ii}^{-1}\tilde{K}_{ib}u_b$$

solves the minimal-energy problem: For given u_b find u_i such that

$$(u_i^T u_b^T) K_{\Omega(n)} \begin{pmatrix} u_i \\ u_b \end{pmatrix}$$

is minimized.

[1] The 'boundary' $\partial\Lambda(n)$ is defined straightforward: if $\Lambda(n)$ is a face/edge then $\partial\Lambda(n)$ are those nodes of $\Lambda(n)$ which belong to more than one face/edge, if $\Lambda(n)$ is an agglomerated element then $\partial\Lambda(n)$ is the union of faces of this element.

What still has to be specified is the extension $E(n)$. In [HV01] various possibilities can be found, we use the there called 'A-extension' (which in our case is in fact a 'K-extension'). If we want to extend a vector v to a dof $i \in \overline{\Omega}(n) \setminus \Omega(n)$ this extension is given by

$$v(i) = \frac{1}{\sum_{j\in S} |k_{ij}|} \sum_{j\in S} \left(|k_{ij}| \cdot v(j)\right),$$

where $S = \{j \in \Omega(i) : k_{ij} \neq 0\}$.

2.3.3 Nonconforming P_1-Element ($P_1^{\mathbf{nc}}$)

For the nonconforming P_1^{nc} element with piecewise linear shape functions, continuous at the midpoints of the edges resp. faces (illustrated in Figure 1(c)) we propose the following strategy.

All dofs on fine edges/faces which are part of the same coarse edge/face are set equal to the dof-value of this coarse edge/face.

For the dofs which lie in the interior of a coarse agglomerate we follow the same strategy as in section 2.3.2 (but without the necessity of either storing the element stiffness matrices or constructing some extension operators). The prolongation matrices for the interior of coarse elements are given by

$$-K_{ii}^{-1} K_{ib},$$

where K_{ii} and K_{ib} can be directly extracted from the global stiffness matrix.

3 Application to the Oseen Equations

The methods above can now be directly applied to the mixed finite element discretization of the Oseen equations (i.e. the fixed-point linearized stationary, incompressible Navier-Stokes equations) in $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$,

$$-\nu\Delta\mathbf{u} + (\mathbf{w}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f}, \tag{4a}$$

$$\operatorname{div}\mathbf{u} = 0, \tag{4b}$$

where $\mathbf{u}(\mathbf{x})$ denotes the velocity of the fluid at point $\mathbf{x}$, $p(\mathbf{x})$ the pressure, $\mathbf{f}(\mathbf{x})$ is some given right hand side, ν the given viscosity (which we assume constant in $\mathbf{x}$), and $\mathbf{w}(\mathbf{x})$ the old approximation of the velocity.

For simplicity we assume Dirichlet boundary conditions on the whole boundary $\Gamma := \partial\Omega$

$$\mathbf{u}|_\Gamma = \bar{\mathbf{u}}.$$

We discretize the system using a mixed finite element method (for details we refer e.g. to [BF91] or [Bra01]) and obtain the problem: Find a couple $(\mathbf{u}_h, p_h)$ in some mixed finite element space $\mathbf{U}_h \times Q_h$ such that

$$\bar{a}(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = \langle F, \mathbf{v}_h\rangle \quad \forall \mathbf{v}_h \in \mathbf{U}_h, \tag{5a}$$

$$b(\mathbf{u}_h, q_h) = 0 \quad \forall q_h \in Q_h, \tag{5b}$$

where

$$\bar{a}(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) = a_D(\mathbf{u}_h, \mathbf{v}_h) + a_C(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) + a_S(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h),$$

and

$$\begin{aligned} a_D(\mathbf{u}_h, \mathbf{v}_h) &= \nu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h), & b(\mathbf{u}_h, q_h) &= -(\operatorname{div} \mathbf{u}_h, q_h), \\ a_C(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) &= ((\mathbf{w}_h \cdot \nabla)\mathbf{u}_h, \mathbf{v}_h), & \langle F, . \rangle &= (\mathbf{f}, .)_0, \end{aligned}$$

and $a_S(.; ., .)$ originates in some convection-stabilization ('D' stands for diffusion, 'C' for convection, and 'S' for stabilization). We use the *Streamline Upwinding Petrov Galerkin Method* (SUPG) and therefore

$$a_S(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) = \beta_h \left((\mathbf{w}_h \cdot \nabla)\mathbf{u}_h, (\mathbf{w}_h \cdot \nabla)\mathbf{v}_h \right),$$

with some stabilization parameter β_h (details can be found e.g. in [Pir89]).

We denote the *FE-isomorphisms* between the discrete spaces and the spaces of coefficient vectors by $\phi_U : (\mathbb{R}^n)^d \to \mathbf{U}_h$ and $\phi_Q : \mathbb{R}^m \to Q_h$. The underline notation is used to indicate their inverses, i.e.

$$\phi_U \underline{\mathbf{v}}_h = \mathbf{v}_h, \qquad \underline{\phi_U \underline{\mathbf{v}}_h} = \underline{\mathbf{v}}_h, \tag{6a}$$

$$\phi_Q \underline{q}_h = q_h, \qquad \underline{\phi_Q \underline{q}_h} = \underline{q}_h. \tag{6b}$$

If it is clear from the context we omit the underlines and ϕ's and identify $\mathbf{v}_h \in \mathbf{U}_h$ and the associated $\underline{\mathbf{v}}_h \in (\mathbb{R}^n)^d$ and analogously q_h and $\underline{q}_h$.

If the mixed finite element is now built of elements for which the construction of an AMGe-interpolation is known, we can easily generate a coupled AMGe method. The coarsening is independent of the concrete element, the interpolations have to be built separately for the pressure and velocity hierarchies.

After having constructed the coarse level hierarchies we need a (coupled) smoother. Possibilities are well known smoothers for geometric multigrid methods, such as transforming smoothers (Wittum [Wit89, Wit90]), the Braess-Sarazin smoother ([BS97], Zulehner [Zul00]), or Vanka smoothers ([Van86], Schöberl, Zulehner [SZ03]).

In principal we now have everything we need for an algebraic multigrid method. But it is not a-priori obvious that the method works, as the (inf-sup) stability of the coarse level systems is not automatically given (and a lack thereof would destroy the convergence properties of the method; c.f. [Wabar, Wab03]).

For the *Crouzeix-Raviart* element P_1^{nc}-P_0 we exemplarily show how a stability result can be obtained. The technique of the proof heavily depends on the fact that on each level a topology is available.

We first introduce the following notations. The intergrid transfer-operators for pressure will be denoted by J_{l+1}^l, for velocity by $\tilde{I}_{l+1}^l$, for velocity-components by I_{l+1}^l, i.e.

$$P_{l+1}^l = \begin{pmatrix} \tilde{I}_{l+1}^l & \\ & J_{l+1}^l \end{pmatrix}, \quad \tilde{I}_{l+1}^l = \begin{pmatrix} I_{l+1}^l & & \\ & I_{l+1}^l & \\ & & I_{l+1}^l \end{pmatrix} \quad \text{(in 3D)}.$$

We denote the spaces for velocity and pressure unknown vectors at level l by $\underline{\mathbf{U}}_l := (\mathbb{R}^{n_l})^d$ and $\underline{Q}_l := \mathbb{R}^{m_l}$ and the coarse function spaces by

$$\mathbf{U}_l := \left\{ \mathbf{v} : \ \exists \underline{\mathbf{w}} \in \underline{\mathbf{U}}_l \text{ such that } \underline{\mathbf{v}} = \tilde{I}_2^1 \tilde{I}_3^2 \dots \tilde{I}_l^{l-1} \underline{\mathbf{w}} \right\},$$
$$Q_l := \left\{ p : \ \exists \underline{q} \in \underline{Q}_l \text{ such that } \underline{p} = J_2^1 J_3^2 \dots J_l^{l-1} \underline{q} \right\}.$$

Analogous to (6) we introduce the *FE-AMG-isomorphisms*

$$\phi_U^l : \underline{\mathbf{U}}_l \to \mathbf{U}_l \ \text{ and } \ \phi_Q^l : \underline{Q}_l \to Q_l, \tag{7}$$

and we will often identify elements of $\underline{\mathbf{U}}_l$ and $\mathbf{U}_l$, and $\underline{Q}_l$ and Q_l

The following lemma is just a translation of the well known result of Fortin [For77].

Lemma 10. *Assume that for given l there exists a linear Operator $\Pi_{l-1}^l : \mathbf{U}_{l-1} \to \mathbf{U}_l$ with*

$$b(\Pi_{l-1}^l \mathbf{v}_{l-1}, q_l) = b(\mathbf{v}_{l-1}, J_l^{l-1} q_l) \quad \textit{for all } q_l \in Q_l \textit{ and } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1} \tag{8}$$

and that

$$\|\Pi_{l-1}^l \mathbf{v}_{l-1}\|_1 \le \delta \|\mathbf{v}_{l-1}\|_1 \quad \textit{for all } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}, \tag{9}$$

with δ independent of h and l.

And assume that on level $l-1$ an inf-sup condition holds, i.e.

$$\inf_{0 \ne q_{l-1} \in Q_{l-1}} \sup_{\mathbf{0} \ne \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}} \frac{b(\mathbf{v}_{l-1}, q_{l-1})}{\|\mathbf{v}_{l-1}\|_1 \|q_{l-1}\|_0} \ge \beta_{l-1}.$$

Then an inf-sup condition holds on level l.

Proof.

$$\begin{aligned}
\inf_{0 \ne q_l \in Q_l} \sup_{\mathbf{0} \ne \mathbf{v}_l \in \mathbf{U}_l} \frac{b(\mathbf{v}_l, q_l)}{\|\mathbf{v}_l\|_1 \|q_l\|_0} &\ge \inf_{0 \ne q_l \in Q_l} \sup_{\mathbf{0} \ne \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}} \frac{b(\Pi_{l-1}^l \mathbf{v}_{l-1}, q_l)}{\|\Pi_{l-1}^l \mathbf{v}_l\|_1 \|q_l\|_0} \\
&= \inf_{0 \ne q_l \in Q_l} \sup_{\mathbf{0} \ne \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}} \frac{b(\mathbf{v}_{l-1}, J_l^{l-1} q_l)}{\|\mathbf{v}_{l-1}\|_1 \|q_l\|_0} \cdot \frac{\|\mathbf{v}_{l-1}\|_1}{\|\Pi_{l-1}^l \mathbf{v}_{l-1}\|_1} \ge \frac{\beta_{l-1}}{\delta}
\end{aligned}$$

□

The proof of the following theorem is rather technical as we often have to switch between two consecutive levels and the finest level.

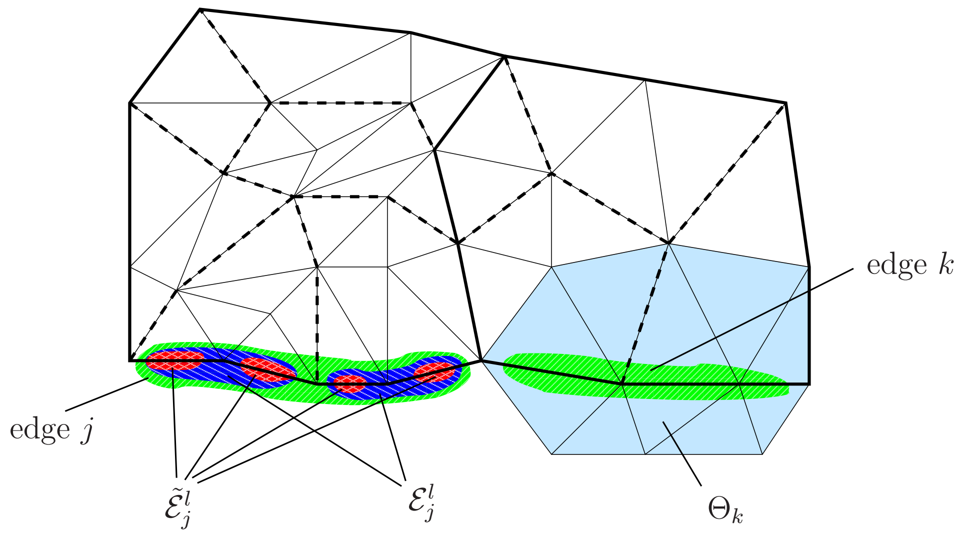
Theorem 11. *Assume that*

$$\frac{h_{max}}{h_{min}} \le \gamma, \tag{10}$$

where h_{max} is the maximal element diameter and h_{min} the minimal diameter (at the finest level), and γ is a positive constant, assume that the (finest level) mesh is shape regular, and assume that the coarse levels are built as described in section 2, the prolongations as in 2.3.1 and 2.3.3 but based on the Stokes-operator (i.e. $\bar{a}(\mathbf{u}_h, \mathbf{v}_h) = \bar{a}(\mathbf{0}, \mathbf{u}_h, \mathbf{v}_h)$).

Then the inf-sup condition holds on all levels.

Figure 2 The solid thick black lines describe two level l elements, the dashed lines level $l-1$ elements and the solid thin lines the finest level elements. For edge j this figure shows the sets $\mathcal{E}_j^l$ and $\tilde{\mathcal{E}}_j^l$, for edge k the tube Θ_k.



As we want to apply Lemma 10 to show this result, we construct an operator Π_{l-1}^l in the following way. Consider the 2D case first (illustrated in Figure 2). Define on level l the index sets $\mathcal{E}_j^l$ of all $(l-1)$-level edges which are part of l-level edge j. We define the length of a l-level edge recursively by

$$e_j^l := \sum_{k \in \mathcal{E}_j^l} e_k^{l-1} \quad (\text{for } l > 1),$$

for $l = 1$ it is determined by the mesh.

We now construct Π_{l-1}^l as follows. For some $(l-1)$-level function v_{l-1} the l-level function $\Pi_{l-1}^l v_{l-1}$ is determined by its values on the (l-level) edges. We set the value on a certain l-level edge to the weighted mean of the values of v_{l-1} on the $(l-1)$-level edges which are part of the edge, i.e.

$$\left(\underline{\Pi_{l-1}^l v_{l-1}}\right)_j = \frac{1}{e_j^l} \sum_{k \in \mathcal{E}_j^l} e_k^{l-1} \left(\underline{v}_{l-1}\right)_k .$$

For a vector valued function $\mathbf{v}_{l-1}$ the term $\Pi_{l-1}^l \mathbf{v}_{l-1}$ will denote the application of Π_{l-1}^l to the components.

Lemma 12. *The operator $\Pi_{l-1}^l : \mathbf{U}_{l-1} \to \mathbf{U}_l$, constructed as above, fulfills (8)*

$$b(\Pi_{l-1}^l \mathbf{v}_{l-1}, q_l) = b(\mathbf{v}_{l-1}, J_l^{l-1} q_l) \quad \textit{for all } q_l \in Q_l \textit{ and } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}.$$

Proof. We want to show that

$$\sum_j \int_{\tau_j} \operatorname{div}(\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot q_l \, \mathrm{d}\mathbf{x} = \sum_j \int_{\tau_j} \operatorname{div} \mathbf{v}_{l-1} \cdot (J_l^{l-1} q_l) \, \mathrm{d}\mathbf{x}.$$

Because $q_l = J_l^{l-1} q_l$ (in functional notation), because q_l is piecewise constant on the l-level agglomerates E_j, and because both $\mathbf{v}_{l-1}$ and $\Pi_{l-1}^l \mathbf{v}_{l-1}$ are piecewise linear on the (finest level) elements τ_j and continuous at the midpoints of their edges, we can use partial integration to derive

$$\sum_j \int_{\tau_j} \operatorname{div}(\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot q_l \,\mathrm{d}\mathbf{x} = \sum_j q_l(E_j) \int_{\partial E_j} (\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot \mathbf{n}$$

and

$$\sum_j \int_{\tau_j} \operatorname{div} \mathbf{v}_{l-1} \cdot (J_l^{l-1} q_l) \,\mathrm{d}\mathbf{x} = \sum_j q_l(E_j) \int_{\partial E_j} \mathbf{v}_{l-1} \cdot \mathbf{n}.$$

By the definition of Π_{l-1}^l we see that

$$\int_{\partial E_j} (\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot \mathbf{n} = \int_{\partial E_j} \mathbf{v}_{l-1} \cdot \mathbf{n} \quad \text{for all agglomerates } E_j,$$

therefore (8) is shown to be true. $\square$

Lemma 13. *The operator Π_{l-1}^l, constructed as above, fulfills also (9)*

$$\|\Pi_{l-1}^l \mathbf{v}_{l-1}\|_1 \le \delta \|\mathbf{v}_{l-1}\|_1 \quad \textit{for all } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}.$$

Proof. The idea of this proof is the introduction of an auxiliary operator $\tilde{\Pi}_{l-1}^l$ on the finer level $\mathbf{U}_{l-1}$, which fulfills (9) and which is identical to Π_{l-1}^l on the coarse edges. Because we use energy minimization for the interpolation in the interior of agglomerates we will then be able to estimate Π_{l-1}^l by $\tilde{\Pi}_{l-1}^l$ which will complete the proof.

We define $\tilde{\Pi}_{l-1}^l : \mathbf{U}_{l-1} \to \mathbf{U}_{l-1}$ by

$$\underline{\tilde{\Pi}_{l-1}^l}(\underline{v}_{l-1})_j := \begin{cases} (\underline{v}_{l-1})_j & \text{if } j \notin \mathcal{E}_k^l \text{ for all } k, \\ \left(\underline{\Pi_{l-1}^l v_{l-1}}\right)_k & \text{if } j \in \mathcal{E}_k^l \text{ for a certain } k. \end{cases}$$

Note that $\tilde{\Pi}_{l-1}^l v_{l-1}$ still 'lives' on level $l-1$, only the values at the l-level faces are averaged.

We try to find an upper bound for $\left| v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1} \right|_1$. Define $\tilde{\mathcal{E}}_j^l$ the index set of all *finest*-level edges which lie on coarse edge j. Set $\tilde{v}_1$ (component of finest-level function $\tilde{\mathbf{v}}_1 \in \mathbf{U}_1$) equal to $v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1}$ on all (finest level) degrees of freedom in $\bigcup_j \tilde{\mathcal{E}}_j^l$ and zero on all other (finest level) degrees of freedom. Then because of the energy minimization in the prolongation

$$\begin{aligned} \int_\Omega \nabla \left(v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1} \right) \nabla \left(v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1} \right) \mathrm{d}\mathbf{x} &\le \int_\Omega \nabla \tilde{v}_1 \nabla \tilde{v}_1 \,\mathrm{d}\mathbf{x} \\ &\le \sum_{\substack{l\text{-level} \\ \text{edges } j}} \int_{\Theta_j} \nabla \tilde{v}_1 \nabla \tilde{v}_1 \,\mathrm{d}\mathbf{x}, \end{aligned} \tag{11}$$

where Θ_j is the tube of (finest level) elements which share a point or edge with l-level edge number j (c.f. Figure 2).

For a (finest level) triangle PQR and the basis function φ_{PQ}, which is equal to 1 at the midpoint of PQ and zero at the midpoints of QR and RP one can easily calculate

$$\int_{PQR} \nabla\varphi_{PQ}\nabla\varphi_{PQ}\,\mathrm{d}\mathbf{x} = \frac{|PQ|^2}{A(PQR)},$$

where $A(PQR)$ denotes the area of the triangle PQR. Now with

$$c_1 := \max_{\tau_j} \frac{(\text{length of longest edge of } \tau_j)^2}{A(\tau_j)}$$

we get

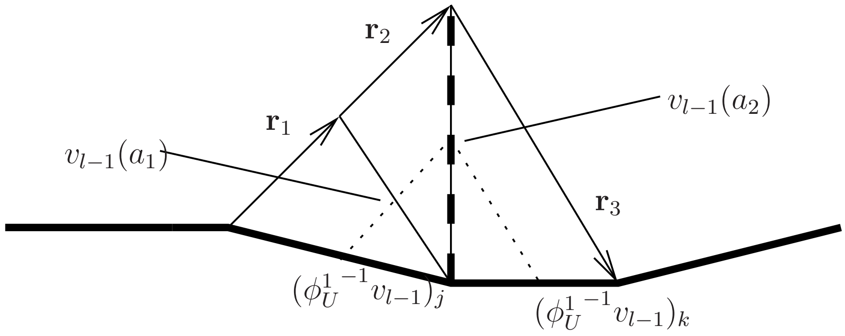
$$\begin{aligned}
\int_{\Theta_j} \nabla\tilde{v}_1\nabla\tilde{v}_1\,\mathrm{d}\mathbf{x} &\le c_1 \sum_{s\in\tilde{\mathcal{E}}_j^l} \left((\underline{\tilde{v}}_1)_s\right)^2 \\
&= c_1 \sum_{s\in\tilde{\mathcal{E}}_j^l} \left[\frac{1}{\sum_{k\in\tilde{\mathcal{E}}_j^l} e_k} \sum_{k\in\tilde{\mathcal{E}}_j^l} e_k \left(\left(\phi_U^{1\;-1} v_{l-1}\right)_s - \left(\phi_U^{1\;-1} v_{l-1}\right)_k\right)\right]^2 \\
&\le c_1 \frac{\sum_{k\in\tilde{\mathcal{E}}_j^l} e_k^2}{\left(\sum_{k\in\tilde{\mathcal{E}}_j^l} e_k\right)^2} \sum_{\substack{s\in\tilde{\mathcal{E}}_j^l,\\ k\in\tilde{\mathcal{E}}_j^l}} \left(\left(\phi_U^{1\;-1} v_{l-1}\right)_s - \left(\phi_U^{1\;-1} v_{l-1}\right)_k\right)^2 \\
&\le c_1 \sum_{\substack{s\in\tilde{\mathcal{E}}_j^l,\\ k\in\tilde{\mathcal{E}}_j^l}} \left(\left(\phi_U^{1\;-1} v_{l-1}\right)_s - \left(\phi_U^{1\;-1} v_{l-1}\right)_k\right)^2 \\
&\le \bar{c}_1 \sum_{\substack{s\in\mathcal{E}_j^l,\\ k\in\mathcal{E}_j^l}} \left((\underline{v}_{l-1})_s - (\underline{v}_{l-1})_k\right)^2,
\end{aligned} \tag{12}$$

where $\phi_U^{1\;-1} v$ is the representation of a coarse function v on $\underline{\mathbf{U}}_1$ as in (7).

We note that ∇v_{l-1} is constant on each finest level element. Therefore we can derive the following estimate (illustrated in Figure 3). Assume that the $(l-1)$-level edges j and k share the node m. We denote the set of all finest level elements which share the node m with $\mathcal{S}_m^{l-1}$, its index set with $\underline{\mathcal{S}}_m^{l-1}$ (where we assume w.l.o.g. $\underline{\mathcal{S}}_m^{l-1} = \{1, 2, \ldots, i+1\}$). For each element τ_e in $\mathcal{S}_m^{l-1}$ we denote the edge-vector of the edge not connected to m ("in direction" $j \to k$) with $\mathbf{r}_e$. Then

$$\begin{aligned}
(\underline{v}_{l-1})_k - (\underline{v}_{l-1})_j &= ((\underline{v}_{l-1})_k - v_{l-1}(a_1)) + (v_{l-1}(a_1) - v_{l-1}(a_2)) + \ldots + (v_{l-1}(a_i) - (\underline{v}_{l-1})_j) \\
&= \frac{1}{2}\left(\nabla v_{l-1}|_{\tau_1} \cdot \mathbf{r}_1 + \ldots + \nabla v_{l-1}|_{\tau_{i+1}} \cdot \mathbf{r}_{i+1}\right) \\
&\le \frac{1}{2}\left(\sqrt{\nabla v_{l-1}|_{\tau_1} \cdot \nabla v_{l-1}|_{\tau_1}}|\mathbf{r}_1| + \ldots + \sqrt{\nabla v_{l-1}|_{\tau_{i+1}} \cdot \nabla v_{l-1}|_{\tau_{i+1}}}|\mathbf{r}_{i+1}|\right),
\end{aligned}$$

Figure 3 Detail of Figure 2, the set $\mathcal{S}_m^{l-1}$.



where $a_1,\ldots,a_i$ are finest level edge midpoints as in Figure 3, thus (using the algebraic-geometric mean inequality)

$$\begin{aligned}\left((\underline{v}_{l-1})_k - (\underline{v}_{l-1})_j\right)^2 &\le \frac{i+1}{4}\left[|\mathbf{r}_1|^2\left(\nabla v_{l-1}|_{\tau_1}\right)^2 + \ldots + |\mathbf{r}_{i+1}|^2\left(\nabla v_{l-1}|_{\tau_{i+1}}\right)^2\right] \\ &\le c_2 \int_{\mathcal{S}_m^{l-1}} \nabla v_{l-1} \nabla v_{l-1}\,\mathrm{d}\mathbf{x}.\end{aligned} \tag{13}$$

We apply this estimate to the last term in (12), which is done directly for those $(l-1)$-level edges s and k which share a node. For all others we have to build a chain of connecting edges.

This leads to

$$\int_{\Theta_l} \nabla \tilde{v}_1 \nabla \tilde{v}_1\,\mathrm{d}\mathbf{x} \le c_3 \int_{\Theta_l} \nabla v_{l-1} \nabla v_{l-1}\,\mathrm{d}\mathbf{x}.$$

Now because of (11) we get

$$\left\|v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1}\right\|_1 \le c_4 \|v_{l-1}\|_1,$$

thus

$$\left\|\tilde{\Pi}_{l-1}^l v_{l-1}\right\|_1 \le (1+c_4)\|v_{l-1}\|_1.$$

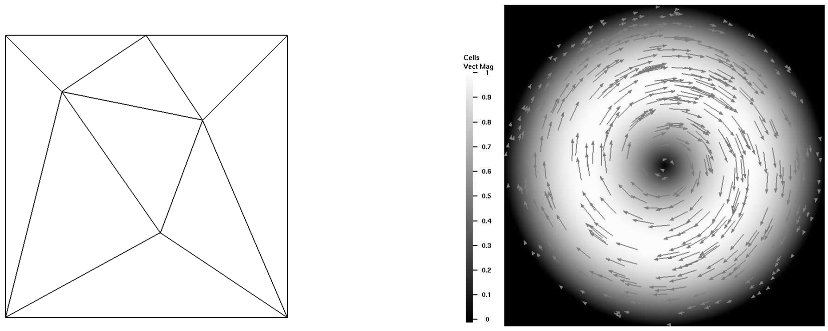
Because we use energy minimization for the interpolation in the interior of coarse agglomerates, $\tilde{I}_l^{l-1}\overline{\Pi_{l-1}^l v_{l-1}}$ has minimal energy amongst all $l-1$-level functions which are identical to it on the l-level edges, therefore

$$\left\|\Pi_{l-1}^l v_{l-1}\right\|_1 \le c_5 \left\|\tilde{\Pi}_{l-1}^l v_{l-1}\right\|_1 \le c_5(1+c_4)\|v_{l-1}\|_1. \tag{14}$$

$\square$

The 3D case. For 3D tetrahedral elements we replace c_1 in (12) by $c_1 h_{j,\max}$, where $h_{j,\max}$ is the maximal element height in tube Θ_j, and c_2 in (13) by $c_2/h_{j,\min}$, where $h_{j,\min}$ is the

Figure 4 Coarsest unit-square mesh and given convection speed $\mathbf{w}$ for first problem.



minimal element height in this tube. Then because of (10) the argumentation remains unchanged, only the scaling argument is based on the (finest level) tetrahedron $PQRS$

$$\int_{PQRS} \nabla\varphi_{PQR}\nabla\varphi_{PQR}\,\mathrm{d}\mathbf{x} = \frac{A(PQR)^2}{V(PQRS)},$$

where $V(PQRS)$ is the volume of the tetrahedron.

Proof of Theorem 11. The proof is completed by combining Lemmata 12 and 13 with Lemma 10. ☐

Remark 14. *The constants $c_1,\ldots,c_5$ that appear in the proofs depend on the finest level triangulation and on the agglomeration strategy (e.g. on how many fine level edges are part of one coarse level edge). In general they may result in $\delta > 1$ (in Lemma 10), thus the inf-sup constant will get worse for each additional level.*

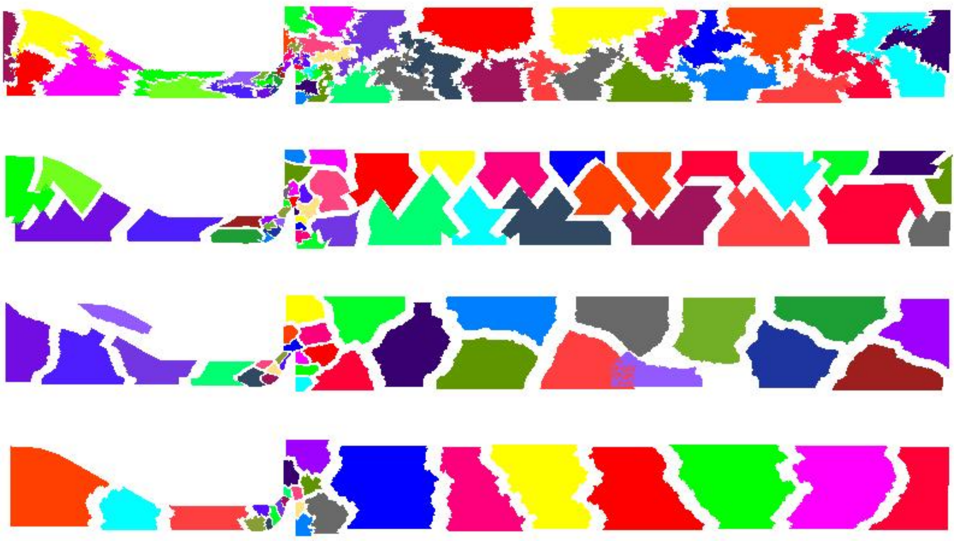
4 Numerical Results

The numerical experiments were carried out using three different geometries resp. meshes,

- the inevitable unit square (with meshes built by the refinement of the coarse mesh in Figure 4),
- the domain around a two-dimensional symmetric valve (provided by AVL List GmbH, Graz, Austria), and
- the domain around two three-dimensional valves (provided by AVL List GmbH).

In Figures 5 and 6 we find the coarsest-level agglomerates of the valve geometries for the different techniques. One can notice that the agglomerates produced with the classical Jones-Vassilevski algorithm have frayed edges, in 3D even holes. A comparison to the bottom-up approach would not be fair (as the coarsest agglomerates are the first ones, so the algorithm has full freedom to produce nice ones), but also the topologies originating in the other two approaches look much better than the classical ones.

Figure 5 Coarsest level agglomerates for the 2D valve. ('classical'... Jones-Vassilevski agglomeration, 'rgb'... red-grey-black, 'metis'... bottom-up partitioning using metis, 'mgridgen'... optimization approach)



For the following tests we introduce a measure for (time-) efficiency

$$T_{0.1} := \frac{\begin{pmatrix}\text{average CPU time in minutes for the reduction}\\ \text{of the norm of the residual by a factor of 0.1}\end{pmatrix}}{\text{number of unknowns}}.$$

This number would be constant for different levels of refinement if we had an optimal method, i.e. if the work for a given reduction of the residual is $\mathcal{O}(n)$, where n is the number of unknowns.

4.1 Convection-Diffusion Equations

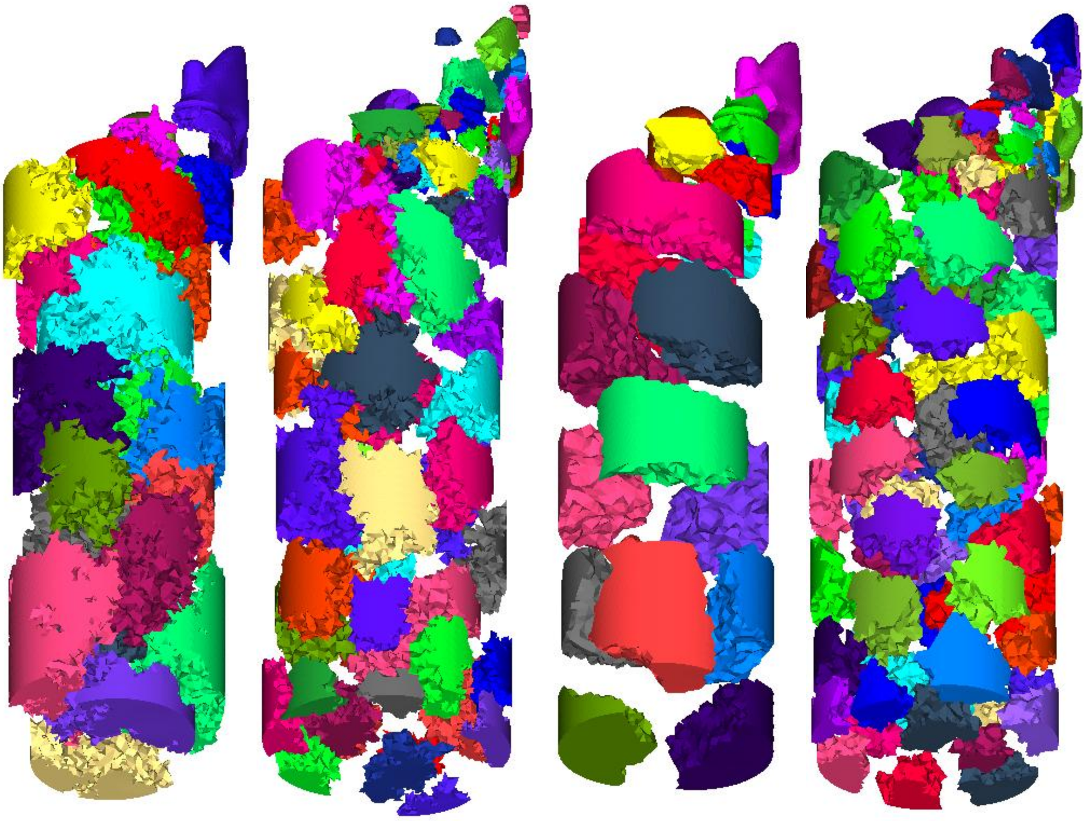
For all examples in this section we use a P_1-discretization of the convection diffusion equation in some domain $\Omega \subset \mathbb{R}^d$

$$-\nu\Delta u + (\mathbf{w} \cdot \nabla)u = f, \tag{15}$$

with given (divergence free) convective speed $\mathbf{w}$, right hand side f, viscosity ν, and Dirichlet boundary conditions

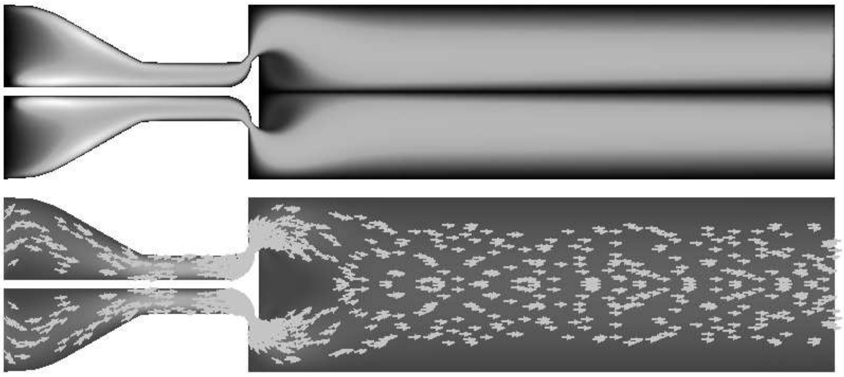
$$u|_\Gamma = \bar{u}.$$

Figure 6 Coarsest level agglomerates for the 3D valves. ('classical'. . . Jones-Vassilevski agglomeration, 'rgb'. . . red-grey-black, 'metis'. . . bottom-up partitioning using metis, 'mgridgen'. . . optimization approach)



classical rgb metis mgridgen

Figure 7 A solution of the 2D valve problem (upper part of the figure) with $\mathbf{w}$ given as in the lower part of the figure. We imposed homogeneous Dirichlet conditions at the symmetry plane.



The first problem we are considering lives on the unit square, with

$$\mathbf{w}(\mathbf{x}) = \begin{pmatrix} m_2 - x_2 \\ x_1 - m_1 \end{pmatrix} \cdot \begin{cases} -16\|\mathbf{m} - \mathbf{x}\| + 8 & \text{if } \|\mathbf{m} - \mathbf{x}\| < 0.5 \\ 0 & \text{otherwise} \end{cases}, \quad \mathbf{m} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix},$$

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{0} - \mathbf{x}\| < 1 \\ 0 & \text{otherwise} \end{cases}$$

(c.f. Figure 4), and $\bar{u} = 0$.

The second problem was the 2D valve, with $\mathbf{w}$ originating in the solution of the full Navier-Stokes equations, with a source ($f \neq 0$) near the inlet (see Figure 7 for a visualization of $\mathbf{w}$ and a solution of the problem).

For both problems we performed h-dependence tests (c.f. Table 1 and Figure 8; for both we used a W-8-8 cycle with Gauß-Seidel smoothers). The values there only concern the iteration-phase, not the set-up. A comparison of the CPU-time needed for set-up (construction of the topology and of the prolongation) and the actual solution phase can be found in Figure 9(a).

The next test deals with the ν dependence (on a fixed grid). The results can be found in Table 2 and Figure 10 (cycles used are the same as before, i.e. W-8-8, Gauß-Seidel smoother).

Some h- and ν-dependence-test for the 3D valves (again with $\mathbf{w}$ coming from the solution of a Navier-Stokes problem) can be found in Table 3 (W-8-8 cycles, Gauß-Seidel smoothers). The timings for the preprocessing phases and the multigrid iterations are illustrated in Figure 9(b).

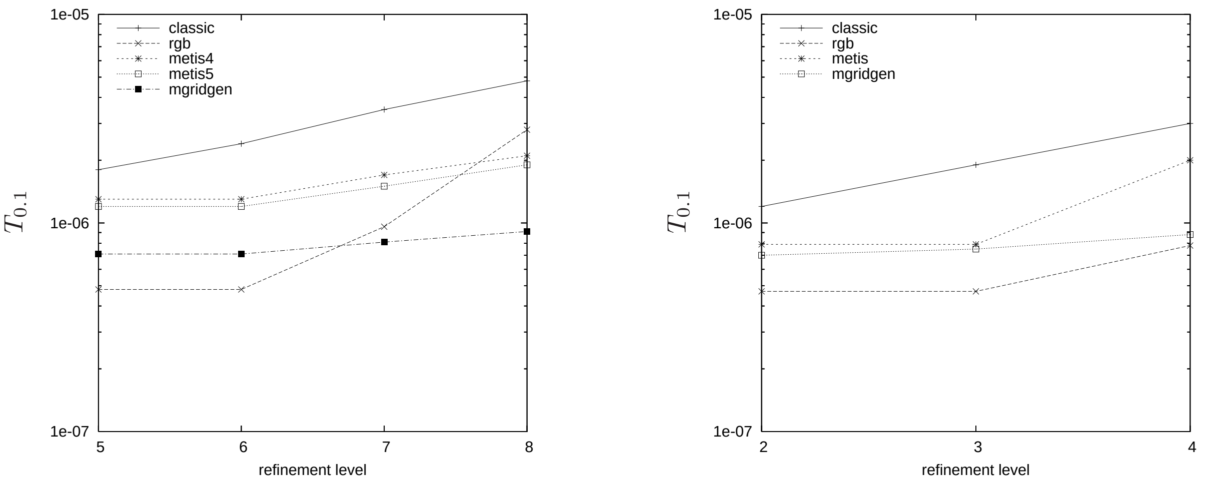
The tests show, that the choice of the agglomeration strategy has a strong influence on the efficiency of the method, sometimes even on the question, if the method is converging. The general tendency which can be observed, is that the optimization approach of MGridGen leads to the best results, although the costs for the topology-construction are pretty high (but for 3D problems low in comparison to the prolongation-construction).

Table 1 Dependence of the reduction per multigrid step (red/step) and $T_{0.1}$ on grid refinements for the unit-square problem ($\nu = 0.01$) and the 2D valve ($\nu = 0.001$).

unit square				
ref. level	5	6	7	8
dofs	4689	18593	74049	295553
classical				
red/step	0.10	0.18	0.22	0.27
$T_{0.1}$	1.8e-6	2.4e-6	3.5e-6	4.8e-6
red-grey-black				
red/step	0.13	0.16	0.36	0.69
$T_{0.1}$	4.8e-7	4.8e-7	9.6e-7	2.8e-6
MGridGen				
red/step	0.16	0.05	0.06	0.06
$T_{0.1}$	7.1e-7	7.1e-7	8.1e-7	9.1e-7
bottom-up, (metis, 4-fold refinement)				
red/step	0.13	0.06	0.11	0.16
$T_{0.1}$	1.3e-6	1.3e-6	1.7e-6	2.1e-6
bottom-up, (metis, 5-fold refinement)				
red/step	0.14	0.13	0.15	0.19
$T_{0.1}$	1.2e-6	1.2e-6	1.5e-6	1.9e-6

2D valve			
ref. level	2	3	4
dofs	11621	45865	182225
classical			
red/step	0.06	0.08	0.16
$T_{0.1}$	1.2e-6	1.9e-6	3e-6
red-grey-black			
red/step	0.07	0.13	0.27
$T_{0.1}$	4.7e-7	4.7e-7	7.8e-7
MGidGen			
red/step	0.05	0.05	0.07
$T_{0.1}$	7e-7	7.5e-7	8.8e-7
bottom-up, (metis, 5-fold refinement)			
red/step	0.02	0.02	0.4
$T_{0.1}$	7.9e-7	7.9e-7	2e-6

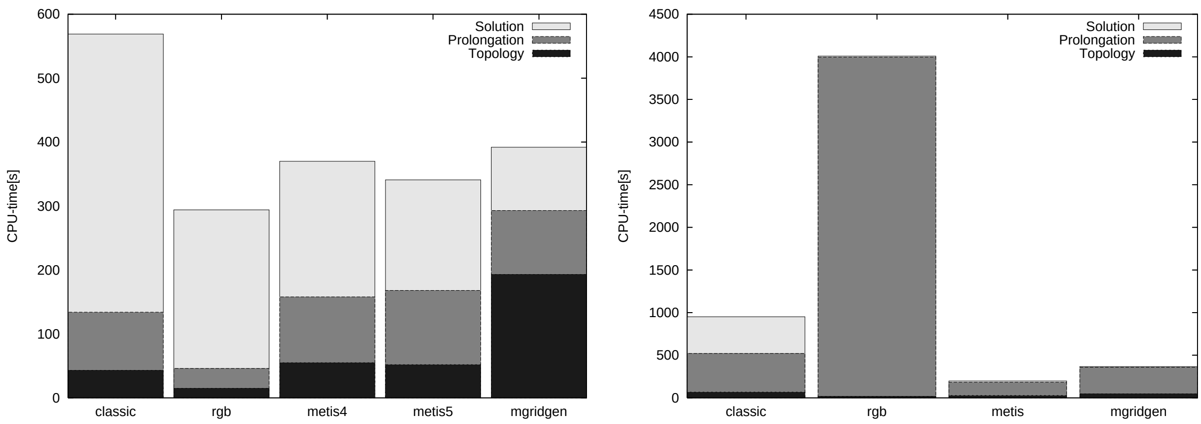
Figure 8 Dependence of the efficiency on the refinement level.



(a) unit-square

(b) 2D valve

Figure 9 CPU-time for the important steps of the AMGe solution process for the convection-diffusion problem ('metis4' resp. 'metis5' denote the bottom-up strategy using metis with 4- resp. 5-fold refinement).



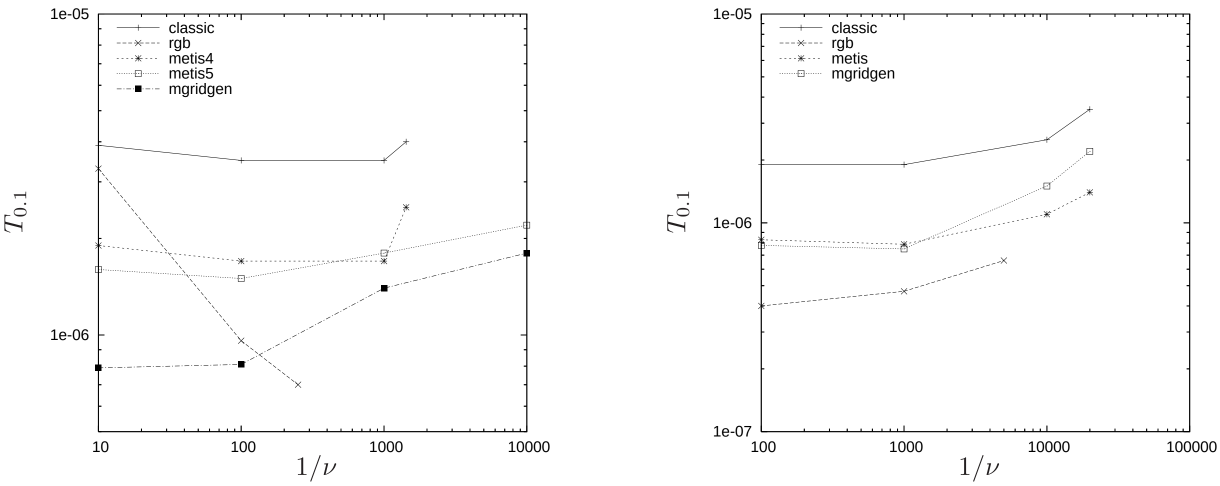
(a) unit-square, 295553 dofs, $\nu = 0.01$

(b) 3D valves, 32463 dofs, $\nu = 0.001$

Table 2 Dependence of the reduction per multigrid step (red/step) and $T_{0.1}$ on ν for the unit-square problem (74049 dofs) and the 2D valve (45865 dofs). For $\nu < 7 \cdot 10^{-4}$ resp. $\nu < 5 \cdot 10^{-5}$ we had no convergence (for the red-grey-black method this occured already for $\nu < 4 \cdot 10^{-3}$ resp. $\nu < 2 \cdot 10^{-4}$).

unit square					**2D valve**				
ν	1e-1	1e-2	1e-3	7e-4	ν	1e-2	1e-3	1e-4	5e-5
classical					**classical**				
red/step	0.26	0.22	0.22	0.28	red/step	0.08	0.09	0.16	0.27
$T_{0.1}$	3.9e-6	3.5e-6	3.5e-6	4e-6	$T_{0.1}$	1.9e-6	1.9e-6	2.5e-6	3.5e-6
red-grey-black					**red-grey-black**				
red/step	0.74	0.36	0.22 (ν=4e-3)		red/step	0.08	0.13	0.21 (ν=2e-4)	
$T_{0.1}$	3.3e-6	9.6e-7	7e-7 (ν=4e-3)		$T_{0.1}$	4e-7	4.7e-7	6.6e-7 (ν=2e-4)	
MGridGen					**MGridGen**				
red/step	0.06	0.06	0.19	0.27	red/step	0.06	0.05	0.26	0.39
$T_{0.1}$	7.9e-7	8.1e-7	1.4e-6	1.8e-6	$T_{0.1}$	7.8e-7	7.5e-7	1.5e-6	2.2e-6
bottom-up, (metis, 5-fold ref.)					**bottom-up**				
red/step	0.19	0.15	0.22	0.28	red/step	0.02	0.02	0.05	0.09
$T_{0.1}$	1.6e-6	1.5e-6	1.8e-6	2.2e-6	$T_{0.1}$	8.3e-7	7.9e-7	1.1e-6	1.4e-6

Figure 10 Dependence of the efficiency on decreasing ν.



(a) unit-square (b) 2D valve

Table 3 h- and ν-dependences for the 3D valves. 'tl' stands for "too long", i.e. the results were not computable within reasonable time. A too long computing time was also the reason for not performing ν-tests for the classical and the red-grey-black topologies.

h**-dependence**			ν**-dependence**			
dofs	32 463	229215	1e-2	1e-3	5e-4	ν
classical						
red/step	4.7e-7	tl	—	—	—	
$T_{0.1}$	3.5e-5	tl	—	—	—	
red-grey-black						
red/step	6.9e-4	tl	—	—	—	
$T_{0.1}$	1.1e-6	tl	—	—	—	
MGridGen						
red/step	0.006	0.037	3e-4	0.006	0.01	red/step
$T_{0.1}$	6.2e-7	1.6e-6	4.4e-7	6.2e-7	7.6e-7	$T_{0.1}$
bottom up						
red/step	4e-5	0.031	3e-6	4e-5	1e-4	red/step
$T_{0.1}$	8.7e-7	1.7e-6	6.5e-7	8.7e-7	1e-6	$T_{0.1}$

Table 4 Results for the solution of the Oseen equations for the driven cavity problem ($\nu = 0.1$).

ref. level	5	6	7	8
dofs	37,024	147,776	590,464	2,360,576
classical				
red/step	0.48	0.52	0.54	0.57
$T_{0.1}$	2.7e-6	3.6e-6	4.3e-6	5e-6
red-grey-black				
red/step	0.64	0.67	0.69	0.70
$T_{0.1}$	2.3e-6	1.9e-6	2.3e-6	2.4e-6
MGridGen				
red/step	0.23	0.26	0.32	0.30
$T_{0.1}$	7e-7	7.5e-7	1.1e-6	1.1e-6
bottom-up (metis)				
red/step	0.52	0.53	0.58	0.60
$T_{0.1}$	1.6e-6	2.3e-6	2.7e-6	2.9e-6

The red-grey-black method seems rather unpredictable, sometimes quite good, sometimes (especially for stronger convection) insufficient. The bottom-up and the classical method lie in between, the agglomerates which are produced by the classical methods sometimes look disadvantageous (bad aspect ratio) and result in a slow convergence.

4.2 Oseen Equations

To check the h-dependence of the cAMGe method for the P_1^{nc}-P_0 element we solve the Oseen equations for the driven cavity problem (unit square, $\bar{\mathbf{u}} = (1\ 0)^T$ at the top edge, $\bar{\mathbf{u}} = \mathbf{0}$ at the other edges), with $\mathbf{w}$ near the solution of the full Navier-Stokes problem and mild convection ($\nu = 0.1$). The results of the tests (with Vanka smoother and W-2-2 cycle) can be found in Table 4 and Figure 11.

What we finally want to compare is the usability of the methods for a full nonlinear Navier-Stokes problem. Here the construction of the topology has to be performed only once (except some kind of semi-coarsening depending on intermediate solutions is wanted), the interpolations can also be kept for more than one nonlinear-iteration. The solver itself should be fast (but it has to be kept in mind, that there is no need to solve the linear problems with very high precision). We have applied the methods for the Navier-Stokes problem on the 3D valves with $\nu = 0.001$ and illustrated the timings in Figure 12. The method with the classical topology-construction did not reach a solution with a reasonable number of smoothing steps and within reasonable time.

Figure 11 Results for the solution of the Oseen equations for the driven cavity problem ($\nu = 0.1$).

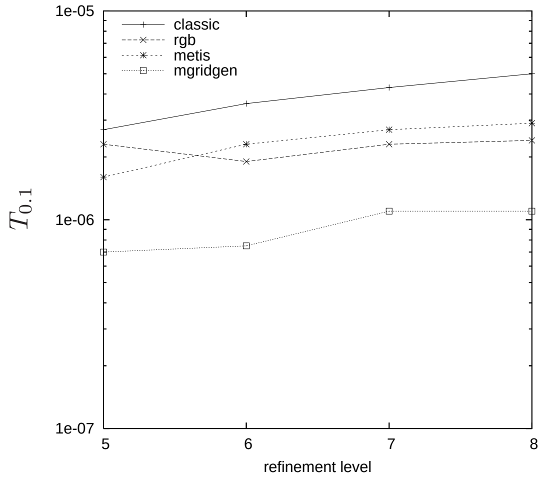
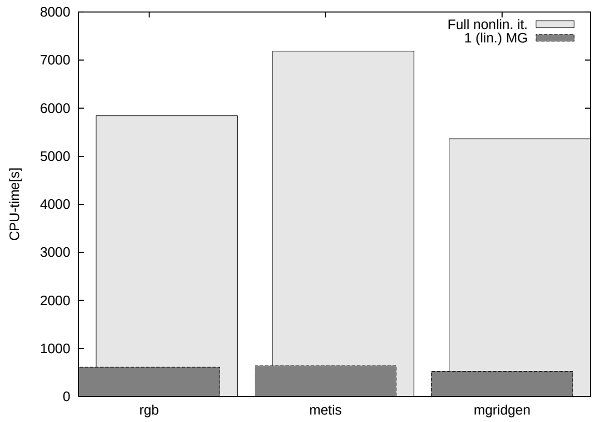


Figure 12 The timings for the full nonlinear problem on the 3D valves (light grey). The dark grey boxes in the foreground illustrate the time needed for the solution of one (the last) linear problem.



5 Conclusions

We have shown that the strategy for the construction of the agglomerates has a crucial influence on the efficiency of the method. The optimization approach, which is the most sophisticated upon the methods tested (and also needs the most CPU-time), in general leads to the best results.

As already mentioned in Vassilevski et al. [JV01, HV01] the AMGe method should only be used if classical AMG methods fail. The AMGe setup-phase is costly, and the memory consumption can get very high, as many connectivity matrices containing the topological information are stored.

One situation where it is sensible to use AMGe is the coupled approach for a mixed FE discretization of the Oseen equations. First, it appears to be a flexible possibility for generalizations (we have demonstrated this for the P_1^{nc}-P_0 element), and secondly — because of the underlying topological information — it gives rise to analytic techniques which are not available for other methods (which we have exploitet for a stability result).

References

[BCF$^+$00] M. Brezina, A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteufel, S.F. McCormick, and J.W. Ruge. Algebraic multigrid based on element interpolation (AMGe). *SIAM J. Sci. Comput.*, 22(5):1570–1592, 2000.

[BF91] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods.* Springer series in computational mathematics. Springer, New York, 1991.

[BMR84] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D.J. Evans, editor, *Sparsity and its applications*, pages 257–284. Cambridge University Press, Cambridge, 1984.

[Bra01] D. Braess. *Finite elements.* Cambridge University Press, 2nd edition, 2001.

[BS97] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Appl. Numer. Math.*, 23:3–20, 1997.

[For77] M. Fortin. An analysis of the convergence of mixed finite element methods. *RAIRO Anal. Numér.*, 11:341–354, 1977.

[HV01] V.E. Henson and P.S. Vassilevski. Element-free AMGe: General algorithms for computing interpolation weights in AMG. *SIAM J. Sci. Comput.*, 23(2):629–650, 2001.

[JV01] J.E. Jones and P.S. Vassilevski. AMGe based on element agglomeration. *SIAM J. Sci. Comput.*, 23(1):109–133, 2001.

[KK98a] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[KK98b] G. Karypis and V. Kumar. *METIS — A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*. University of Minnesota, 1998. available at: `www-users.cs.umn.edu/~karypis/metis/`.

[MK01a] I. Moulitsas and G. Karypis. *MGridGen/ParMGridGen — Serial/Parallel Library for Generating Coarse Grids for Multigrid Methods*. University of Minnesota, 2001. available at: `www-users.cs.umn.edu/~moulitsa/software.html`.

[MK01b] I. Moulitsas and G. Karypis. Multilevel algorithms for generating coarse grids for multigrid methods. In *Conference Proceedings of "Supercomputing 2001"*, 2001.

[Pir89] O. Pironneau. *Finite Element Methods for Fluids*. John Wiley & Sons, Chichester, 1989.

[RS86] J.W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. McCormick, editor, *Multigrid Methods*, volume 5 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, 1986.

[Stü01] K. Stüben. A review of algebraic multigrid. *Computational and Applied Mathematics*, 128:281–309, 2001.

[SZ03] J. Schöberl and W. Zulehner. On Schwarz-type smoothers for saddle point problems. *Numer. Math.*, 95(2):377–399, 2003.

[Van86] S. Vanka. Block-implicit multigrid calculation of two-dimensional recirculating flows. *Comp. Meth. Appl. Mech. Eng.*, 59(1):29–48, 1986.

[Wab03] M. Wabro. *Algebraic Multigrid Methods for the Numerical Solution of the Incompressible Navier-Stokes Equations*. PhD thesis, University of Linz, 2003. published by Universitätsverlag Rudolf Trauner.

[Wabar] M. Wabro. Coupled algebraic multigrid methods for the Oseen problem. *Computing and Visualization in Science*, to appear.

[Wit89] G. Wittum. Multi-grid methods for Stokes and Navier-Stokes equations. *Numer. Math.*, 54:543–563, 1989.

[Wit90] G. Wittum. On the convergence of multi-grid methods with transforming smoothers. *Numer. Math.*, 57:15–38, 1990.

[Zul00] W. Zulehner. A class of smoothers for saddle point problems. *Computing*, 65(3):227–246, 2000.