Algebraic multigrid based on computational molecules, 1: Scalar elliptic problems

J.K. Kraus, J. Schicho

RICAM-Report 2005-05

Algebraic multigrid based on computational molecules, 1: Scalar elliptic problems

J.K. Kraus and J. Schicho

March 21, 2005

Abstract

We consider the problem of splitting a symmetric positive definite (SPD) stiffness matrix A arising from finite element discretization into the sum of edge matrices thereby assuming that A is given as the sum of symmetric positive semidefinite (SPSD) element matrices. We give necessary and sufficient conditions for the existence of a splitting into SPSD edge matrices and provide a feasible algorithm for their computation.

Based on this disassembling process we present a new concept of "strong" and "weak" connections (edges), which provides a basis for selecting the coarse-grid nodes in algebraic multigrid methods. Furthermore, we examine the utilization of computational molecules (small collections of edge matrices) for deriving interpolation rules. The reproduction of edge matrices on coarse levels offers the opportunity to combine classical coarsening algorithms with effective (energy minimizing) interpolation principles yielding a flexible and robust new variant of AMG.

AMS Subject Classification: 65F10, 65N20, 65N30
Key words: edge matrices, algebraic multigrid, interpolation weights, coarse-grid selection

1 Introduction

We are concerned with the solution of large-scale systems of linear equations

$$A\mathbf{u} = \mathbf{f} \tag{1}$$

arising from finite element (FE) dicretization of self-adjoint elliptic boundary-value problems. In this situation, the matrix A in (1) is typically sparse and symmetric positive definite (SPD).

In many instances (of this huge class of problems) Algebraic MultiGrid (AMG) methods [2, 3, 4, 5] can be used to build highly efficient and robust linear solvers [10, 12, 15, 17, 18, 20]. AMG using element interpolation (AMGe) [6, 13, 14], so-called spectral AMGe [9], and AMG based on smoothed aggregation [16, 21, 23, 24, 25] have even broadened the range of applicability of the classical AMG algorithm [17]. These more recent developments are based on techniques of energy-minimizing interpolation (or prolongation), which can be attained by different means. The basic functioning of algebraic multigrid, however, is the same in all cases.

A two-grid method for solving (1) is given by:

$$\text{Relax } \nu_1 \text{ times on } A\mathbf{u} = \mathbf{f}. \tag{2}$$

$$\text{Correct } \mathbf{u} \leftarrow \mathbf{u} + P(P^T AP)^{-1} P^T (\mathbf{f} - A\mathbf{u}). \tag{3}$$

$$\text{Relax } \nu_2 \text{ times on } A\mathbf{u} = \mathbf{f}. \tag{4}$$

Here P is an $n \times n_c$ interpolation (or prolongation) matrix that transfers coarse-grid corrections onto the fine grid whereas P^T gives the corresponding restriction of fine-grid vectors onto the coarse grid. This requires the selection of a proper coarse grid $\mathcal{D}_c$, i.e., a set of $n_c < n$ coarse-grid nodes, which is a critical point in the design of two-grid (and multigrid) methods.

An (algebraic) multigrid method recursively uses the two-grid method (2)–(4) to solve the linear systems involving the so-called Galerkin coarse-grid operator $A_c := P^T AP$. The coarse-grid correction step (3) applies the matrix $C := I - PA_c^{-1} P^T A$, which shows that $\text{Range}(P) = \text{Range}(I - C)$. Further, if A is SPD (A induces an inner product) the matrix C provides the A-orthogonal projection onto $\text{Range}(P)$. The coarse-grid correction (3) then minimizes the energy norm of the fine-grid error over all possible corrections in the range of P [7]. Moreover, a basic requirement for any multigrid method is that relaxation and coarse-grid correction complement each other, i.e., error not reduced by one has to be reduced by the other [6].

The computation of edge matrices, we are suggesting in the present paper, is motivated by the fact that they provide a good starting point for building efficient AMG components, while keeping their set-up costs low. The main emphasis of this paper is on the algebraic construction of edge matrices and their utilization in the framework of algebraic multigrid: We discuss how to alter the concept of “strong” and “weak” connections, as it is used in the process of coarse-grid selection (and interpolation) with classical AMG. The interpolation component in our approach is very similar to the element interpolation used in so-called AMGe methods. However, the *computational molecules* involved in the arising local min-max problem are assembled from edge matrices in our case.

The general idea is to achieve higher flexibility and adaptivity by using the presented edge-based methodology. The resulting method lies in-between classical AMG (the concept of “strong” and “weak” edges affects the coarsening as well as the formation of interpolation molecules) and AMGe based on element agglomeration (small-sized neighborhood matrices form the basis for the computation of the actual interpolation coefficients).

Numerical tests indicate the robustness of the considered method to which we refer as AMGm (Algebraic MultiGrid based on computational molecules) with respect to operator anisotropy as well as perturbations of the M-matrix property.

2 Edge Matrices

Let A_T be an $(nd) \times (nd)$ symmetric and semipositive element matrix. Here, n denotes the number of nodes in the element T, and d denotes the number of degrees of freedom in each node. For i, j, $1 \le i < j \le n$, let E_{ij} be an $(nd) \times (nd)$ symmetric matrix whose entries are zero except for the $(2d) \times (2d)$ entries corresponding to the nodes i, j. We will say that E_{ij} is an *edge matrix* of A_T iff $E_{ij}\mathbf{v} = \mathbf{0}$ for all $\mathbf{v} \in \ker(A_T)$. We say that A_T has a positive splitting (decomposition) iff we can write it as a sum of positive semidefinite edge matrices.

The main goal in this section is to give a necessary and sufficient criterion for the existence of a positive splitting in the case $d = 1$. We also specify the construction for the case when this criterion is fulfilled.

Using the terminology from [22], we say that a matrix is *irreducible* iff the graph with nodes $1, \dots, n$ and edges representing nonzero matrix entries is connected. If A_T is reducible, then there is a numbering of nodes for which A_T has the shape of a block diagonal matrix. One can show that A_T has a positive splitting iff every diagonal block has a positive splitting. Therefore, we will focus our attention to irreducible matrices.

We say that $A_T = (a_{ij})_{i,j}$ is an L-matrix iff $a_{ii} > 0$ and $a_{ij} \le 0$ for $1 \le i \ne j \le n$. If A_T is not an L-matrix, then there is a unique L-matrix $B := (b_{ij})_{i,j}$ such that $|a_{ij}| = |b_{ij}|$ for all i, j. We say that B is the L-ation of A_T.

Lemma 2.1 *A symmetric matrix A_T has a positive splitting iff its L-ation has a positive splitting.*

Proof. Let $A_T = \sum_{i,j} E_{ij}$ be a splitting into edge matrices. Then the L-ation of A_T is the sum of the L-ations of the edge matrices E_{ij}. Because positivity of a symmetric 2×2-matrix does not depend on the sign of the off-diagonal entry, the original splitting is positive iff the L-ated splitting is positive. $\square$

Lemma 2.2 *If A_T is an irreducible singular L-matrix, then its kernel has dimension 1, and it is generated by a positive vector.*

Proof. This is well-known [1]. We include a proof for the sake of self-containedness.

Let $\mathbf{v} = (v_1, \dots, v_n)^t$ be an element of the kernel. By simultanuous permutation of rows and columns, and maybe replacing $\mathbf{v}$ by $-\mathbf{v}$, we may assume that $v_1, \dots, v_l > 0$, $v_{l+1}, \dots, v_m < 0$, and $v_{m+1} = \cdots = v_n = 0$ for indices l, m such that $1 \le l \le m \le n$. Then A_T can be written as block matrix

$$A_T = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix},$$

with positive semidefinite A_{ii}, and $A_{ij} \le 0$ elementwise for $i \ne j$.

For $i = l+1, \dots, m$, we have

$$a_{i1}v_1 + \cdots + a_{il}v_l + \sum_{j=l+1}^{m} a_{ij}v_j = 0,$$

hence

$$\sum_{j=l+1}^{m} a_{ij}v_iv_j = v_i \left(\sum_{j=1}^{l} a_{ij}v_j \right) \le 0.$$

Summing up, we get $\sum_{i,j=l+1}^{m} a_{ij}v_iv_j \le 0$. But A_{22} is positive semidefinite, which shows that we have equality everywhere. Hence $A_{21} = 0$ and the vectors $\mathbf{w} := (v_1, \dots, v_l, 0, \dots, 0)^t$ and $\mathbf{v} - \mathbf{w}$ are both in the kernel of A. But $A\mathbf{w} = 0$ implies $A_{31} = 0$, and $A(\mathbf{v} - \mathbf{w}) = 0$ imples $A_{32} = 0$. Since A is assumed to be connected, we have $l = m = n$. □

Lemma 2.3 *If A_T is an irreducible singular L-matrix, then it has a unique splitting into edge matrices. Moreover, this splitting is positive.*

Proof. By Lemma 2.2, there is a positive vector $\mathbf{v} = (v_1, \dots, v_n)^t$ generating the kernel. For any i, j, $1 \le i < j \le n$, there is a unique edge matrix E_{ij} annihilating $\mathbf{v}$ and with off-diagonal entry a_{ij}, namely the matrix with nonzero entries

$$\begin{pmatrix} -a_{ij}v_j/v_i & a_{ij} \\ a_{ij} & -a_{ij}v_i/v_j \end{pmatrix}$$

(except when $a_{ij} = 0$, which yields a zero edge matrix). Direct computation shows that the sum of these E_{ij} equals A_T. This shows that we have a unique splitting. The positivity is a consequence of $a_{ij} \le 0$ and $v_iv_j > 0$. □

Lemma 2.4 *Any positive semidefinite matrix can be written as a sum of a singular positive semidefinite matrix and a positive semidefinite diagonal matrix.*

Proof. Let B be a positive semidefinite matrix. Let D be a nonzero positive diagonal matrix. The set of all real numbers λ such that $B - \lambda D$ is positive semidefinite is closed and it contains zero. Moreover, it is bounded because $-D$ is certainly not positive semidefinite. If λ_0 is the maximum of this set, then $B - \lambda_0 D$ is singular and positive semidefinite and $\lambda_0 D$ is a positive semidefinite diagonal matrix. □

Theorem 2.1 *A symmetric matrix A_T has a positive splitting iff its L-ation is positive semidefinite.*

Proof. Without loss of generality, we may assume that A_T is irreducible. By Lemma 2.1, we may also assume that A_T is an L-matrix. Then one direction is obvious: if A_T has a positive splitting, then it is positive semidefinite.

Assume that A_T is positive semidefinite. If A_T is singular, then it has a positive descomposition by Lemma 2.3. Otherwise, we use Lemma 2.4 and write $A_T = A' + D$, where A' is a singular L-matrix and D is a positive diagonal matrix. By Lemma 2.3, A' has a positive splitting. Clearly, D has a positive splitting. By summing the edge matrices for A' and for D, we get a positive splitting for A_T. $\square$

We turn to the question how to compute a positive splitting, in case we know there exists one? In the case $n = 3$, we use the computer algebra system Maple to compute an explicit formula

$$\begin{aligned}
A_T &= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} \\
&= \begin{pmatrix} \frac{a_{11}a_{22}a_{33}+a_{12}^2a_{33}-a_{11}a_{23}^2-a_{22}a_{13}^2}{2(a_{22}a_{33}-a_{23}^2)} & a_{12} & 0 \\ a_{12} & \frac{a_{11}a_{22}a_{33}+a_{12}^2a_{33}-a_{11}a_{23}^2-a_{22}a_{13}^2}{2(a_{11}a_{33}-a_{13}^2)} & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
&+ \begin{pmatrix} \frac{a_{11}a_{22}a_{33}+a_{13}^2a_{22}-a_{11}a_{23}^2-a_{33}a_{12}^2}{2(a_{22}a_{33}-a_{23}^2)} & 0 & a_{13} \\ 0 & 0 & 0 \\ a_{13} & 0 & \frac{a_{11}a_{22}a_{33}+a_{13}^2a_{22}-a_{11}a_{23}^2-a_{33}a_{12}^2}{2(a_{11}a_{22}-a_{12}^2)} \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{a_{11}a_{22}a_{33}+a_{23}^2a_{11}-a_{11}a_{23}^2-a_{22}a_{13}^2}{2(a_{11}a_{33}-a_{13}^2)} & a_{23} \\ 0 & a_{23} & \frac{a_{11}a_{22}a_{33}+a_{23}^2a_{11}-a_{11}a_{23}^2-a_{22}a_{13}^2}{2(a_{11}a_{22}-a_{12}^2)} \end{pmatrix}.
\end{aligned}$$

Proposition 2.1 *If the L-ation of A_T is positive semidefinite, then the splitting above is positive.*

Proof. A matrix is positive semidefinite iff the determinants of all main minors are greater than or equal to zero. We compute all determinants of main minors of the summands above, and factorize them; it turns out that the only factors that appear are either determinants of main minors of A_T, or determinants of main minors of A_T', which is constructed from A_T by multiplying each off diagonal element with -1. By a combinatorial case by case analysis, the proposition can then be checked easily. $\square$

If $n > 3$, we proceed as follows. If A_T is a singular positive semidefinite L-matrix, then we can use the explicit construction in the proof of Lemma 2.3. If A_T is a nonsingular positive semidefinite L-matrix, then we compute the smallest eigenvalue $\lambda_{\min}$, construct a positive splitting of $A_T - \lambda_{\min} I$, and add suitable diagonal edge matrices. Finally, if A_T is not an L-matrix but its L-ation is positive semidefinite, then we L-ate A_T, compute a positive splitting of the L-ation, and de-L-ate the edge matrices again by multiplying the off diagonal elements with ± 1.

3 Coarse grid selection

3.1 "Weak" and "strong" edges

In contrast to Geometric MultiGrid (GMG) the relaxation in AMG is fixed [17]. Thus the coarsening process and the interpolation rule have to be chosen in a way such that the range of interpolation approximates those errors not efficiently reduced by relaxation.[1] These algebraically smooth error components $\mathbf{e}$ are characterized by

$$\|S\mathbf{e}\|_A \approx \|\mathbf{e}\|_A, \tag{5}$$

wherein S denotes the smoother. For (most of) the common smoothers, e.g., Gauß-Seidel or Jacobi, error that is slow to converge in energy norm equivalently fulfills the condition

$$a_{ii} e_i \approx - \sum_{j \neq i} a_{ij} e_j. \tag{6}$$

In particular, for M-matrices this means that for each node i the error component e_i is essentially determined by those e_j for which $-a_{ij}$ is large. This leads to the following definition of strong connections used in classical AMG [17]:

Definition 3.1 *(Strong connections in Classical AMG) Node i is strongly connected to node j (strongly depends on j) if $-a_{ij} \geq \theta \cdot \max_{k \neq i} \{-a_{ik}\}$ with some $0 < \theta \leq 1$ (e.g., $\theta = 0.25$).*

Here we want to base the concept of strong connectivity on edge matrices. In [8] a reliable evaluation of strong connections based on element stiffness matrices has been presented. It uses the simple formula

$$s_{ij} := \frac{|a_{ij}|}{\sqrt{a_{ii} a_{jj}}} \tag{7}$$

[1]In GMG the hierarchy of grid equations is given, i.e., a fixed coarsening is used, and the smoother is adjusted in order to obtain efficient multigrid cycles.

where $A_T = (a_{ij})_{i,j}$ is a local stiffness matrix corresponding to some element $T \in \mathcal{T}$. Note that (7) defines the energy cosine of the abstract angle between the i-th and j-th (nodal) basis function. However, the reproduction of local element stiffness matrices on coarser levels increases the set-up costs of an AMG method significantly and–what is even more serious–is subject to strong geometrical restrictions. That is why we suggest to construct edge matrices (from element matrices) and reproduce those on coarser levels. The concept of "strong" edges can be established as follows:

Definition 3.2 *(Direct connections) Any two nodes i and j are said to be directly connected iff there is an edge $\{i,j\}$ connecting i and j; let E_{ij} denote the corresponding edge matrix.*

Now for every loop of length 3 (triangle) in the *algebraic grid* with direct connections (edges) $\{i,j\}$, $\{j,k\}$, and $\{k,i\}$ we consider the molecule

$$M^{(i,j,k)} := E_{ij} + E_{jk} + E_{ki}. \tag{8}$$

Furthermore, let

$$\mathcal{M}^{\triangle} := \{M^{(i,j,k)} = \{c_{pq}\} : c_{pp} \neq 0 \quad \forall p = 1,2,3\} \tag{9}$$

be the set of all such local matrices given as the sum of three edge contributions (for edges that form a triangle) having non vanishing diagonal entries. Then the following definition provides an altered concept of strong connections ("strong" edges).

Definition 3.3 *(Strong connections via edge matrices) The strength of a (direct) connection $\{i,j\}$ is defined by*

$$s_{ij} := \min\{1, \min_{M^{(i,j,k)} \in \mathcal{M}^{\triangle}} \{|c_{p_i p_j}|/(\sqrt{c_{p_i p_i} c_{p_j p_j}})\}\} \tag{10}$$

where connections with $s_{ij} \geq \theta$ are said to be strong, $0 < \theta < 1$ (e.g., $\theta = 1/3$). Here p_i and p_j denote the local indices associated with nodes i and j, respectively, i.e., $1 \leq p_i \equiv p(i), p_j \equiv p(j) \leq 3$.

Remark 3.1 *Note that the strength of a connection $\{i,j\}$ computed via either of the formulas (7) or (10) will be non-negative and bounded above by 1 for general symmetric positive semidefinite stiffness matrices. A value close to one (an abstract angle close to zero) indicates a strong connection. Extending the definition (10) by $\min_{\emptyset}\{\ldots\} := \infty$ one gets $s_{ij} = 1$ whenever nodes i and j are directly connected but there is no path of length two connecting i and j via a third node k. In consequence of the special construction of coarse edges presented in Section 5.1 this maintains the connectivity of coarse grids for connected fine grids, in practice.*

3.2 Coarsening algorithm

There are several reasonable ways of selecting the coarse grid nodes in AMG. Our approach is similar to the one used in classical AMG [17] in that it is based on a concept of strong connections (here "strong" edges). However, since the precise coarsening algorithm we use in detail differs from the one proposed in [17] it will be presented in this section. Following [17] a *good* coarse grid $\mathcal{D}_c$ should satisfy the following two criterions:

C1: *$\mathcal{D}_c$ should be a maximum independent set, which means that no strong connections within $\mathcal{D}_c$ are allowed.*

C2: *Each node j being strongly connected to an f-node i is either contained in $\mathcal{D}_c$ or it strongly depends on at least one c-node k that itself is strongly connected to node i.*

Similar to the procedure proposed in [17] we select the coarse grid in a two-stage process: First, a quick c-node choice attempts to enforce criterion (C1). Then, at a second stage, all f-nodes resulting from the first stage are tested to ensure that criterion (C2) holds, adding new c-nodes if necessary.

The main difference to the methodology in [17] is that our relation of strong connectivity, as defined in Definition 3.3, is symmetric whereas this is not the common practice in classical AMG, even not in the SPD case. That means, whenever a node i is strongly connected to a node j the reverse (j being strongly connected to i) is also true. However, this even simplifies the selection of an initial coarse grid that takes into account criterion (C1). A greedy algorithm serving this purpose is given by Algorithm 3.1. The testing of the initial coarse grid, and its adjustment with respect to criterion (C2), can be performed according to Algorithm 3.2. Note that we slightly simplified the corresponding procedure from [17] by avoiding multiple testing of f-nodes in the course of adding c-nodes (doing without an intermediate choice of "tentative" c-nodes). Instead, we prefer to decide immediately which f-nodes are going to be changed into c-nodes: if two f-nodes are strongly connected to each other with no c-node that strongly depends on both of them the necessity arises to change either of them to a c-node; we take the one having fewer strongly dependent c-nodes. For a formal description of this two stage process we define the node sets:

$\mathcal{D}_f$	…	fine nodes (f-nodes)
$\mathcal{D}_c$	…	coarse nodes (c-nodes)
$\mathcal{D} := \mathcal{D}_f \cup \mathcal{D}_c$	…	all nodes
$\mathcal{N}_i$	…	direct neighbors of node i
$\mathcal{N}_i^f := \mathcal{N}_i \cap \mathcal{D}_f$	…	fine direct neighbors
$\mathcal{S}_i$	…	strongly connected direct neighbors of node i
$\mathcal{S}_i^c := \mathcal{S}_i \cap \mathcal{D}_c$	…	strongly connected coarse direct neighbors
$\lvert\mathcal{D}\rvert$ or $\lvert\mathcal{S}_m\rvert$	…	cardinality of $\mathcal{D}$ respectively $\mathcal{S}_m$

Algorithm 3.1 *(Selection of initial coarse grid)*

```
$\mathcal{D}_c := \emptyset$;    $\mathcal{D}_f := \emptyset$;    $U := \mathcal{D}$;
$\lambda_m = |\mathcal{S}_m| \quad \forall m = 1, 2, \ldots, |\mathcal{D}|$;    $n = 0$;
while $(n < |\mathcal{D}|)$
    find $i$ such that $\lambda_i = \max_{m \in U} \lambda_m$
    $\mathcal{D}_c := \mathcal{D}_c \cup \{i\}$
    $U := U \setminus \{i\}$
    $n := n + 1$
    for all $j \in \mathcal{S}_i \cap U$
        $\mathcal{D}_f := \mathcal{D}_f \cup \{j\}$
        $U := U \setminus \{j\}$
        $n := n + 1$
        for all $k \in \mathcal{S}_j \cap U$
            $\lambda_k := \lambda_k + 1$
```

Algorithm 3.2 *(Adjustment of initial coarse grid)*

```
$\lambda_m = 0 \quad \forall m = 1, 2, \ldots, |\mathcal{D}|$;    $i = 0$;
while $(i < |\mathcal{D}|)$
    $i := i + 1$
    if $(i \in \mathcal{D}_f)$
        $n_1 = 0$
        for all $k \in \mathcal{S}_i \cap \mathcal{D}_c$
            $n_1 := n_1 + 1$
            $\lambda_k = 1$;
        for all $j \in \mathcal{S}_i \cap \mathcal{D}_f$
            $n_2 = 0$;    $n_3 = 0$;
            for all $k \in \mathcal{S}_j \cap \mathcal{D}_c$
                $n_2 := n_2 + 1$
                if $(\lambda_k = 1)$
                    $n_3 := n_3 + 1$
            if $(n_3 < 1)$
                if $(n_1 < n_2)$
                    $\mathcal{D}_c := \mathcal{D}_c \cup \{i\}$
                    $\mathcal{D}_f := \mathcal{D}_f \setminus \{i\}$
                else
                    $\mathcal{D}_c := \mathcal{D}_c \cup \{j\}$
                    $\mathcal{D}_f := \mathcal{D}_f \setminus \{j\}$
                    $n_1 := n_1 + 1$
                    $\lambda_j = 1$
        for all $k \in \mathcal{N}_i \quad \lambda_k = 0$
```

4 Interpolation

In this section we want to figure out how to benefit from edge matrices when constructing the interpolation component of our AMGm method. The basic idea is to construct suitable small-sized *computational molecules* from edge matrices and to choose the interpolation coefficients in such a way that they provide a local minimum energy extension with respect to the considered *interpolation molecule.*

4.1 Interpolation molecules

We will say that M is a computational molecule if M is a small-sized irreducible matrix that can be assembled from edge matrices. Let $\mathcal{E}_M$ be a small subset of the set of all edges $\mathcal{E}$, $\mathcal{E}_M \subset \mathcal{E}$. Then, for notational convenience we represent the molecule associated with the edge set $\mathcal{E}_M$ by

$$M := \sum_{\{i,j\} \in \mathcal{E}_M} E_{ij}. \tag{11}$$

Note that M is a small-sized $n_M \times n_M$ matrix where n_M denotes the number of distinct nodes k belonging to any of the edges $\{i,j\} \in \mathcal{E}_M$. To be precise, this matrix is obtained from the full-sized $N \times N$ matrix

$$C := \sum_{\{i,j\} \in \mathcal{E}_M} R_{ij}^T E_{ij} R_{ij} \tag{12}$$

by deleting all its zero rows and columns; the $2 \times N$ permutation matrices R_{ij} in (12) provide the mapping to the global ordering of nodes.

Consider now the set $\{A_T\}$ of individual element matrices all of which are split (disassembled) into edge matrices, i.e.,

$$A_T = \sum_{\{i,j\} \subset T} E_{ij} \quad \forall T. \tag{13}$$

We note that if the splittings (13) are positive throughout, i.e., all edge matrices E_{ij} are SPSD, then every computational molecule locally preserves the kernel of the global stiffness matrix:

Lemma 4.1 *Let $B = \sum_{T \in \mathcal{T}_B} A_T$ and $\mathbf{v}_B \in \ker(B)$, i.e., $B\mathbf{v}_B = \mathbf{0}$. Further, let $M = \sum_{\{i,j\} \in \mathcal{E}_M} E_{ij}$ be any computational molecule such that every edge $\{i,j\} \in \mathcal{E}_M$ belongs to some element $T \in \mathcal{T}_B$. Moreover, let $\mathbf{v}_M$ denote the restriction of $\mathbf{v}_B$ to the edges in $\mathcal{E}_M$, i.e., $\mathbf{v}_M := \mathbf{v}_B|_{\mathcal{E}_M}$. If the splitting (13) is positive for all elements $T \in \mathcal{T}_B$, it follows that*

$$M\mathbf{v}_M = \mathbf{0}. \tag{14}$$

Proof. For an SPSD matrix B the condition $B\mathbf{v}_B = \mathbf{0}$ is equivalent to $\mathbf{v}_B^T B \mathbf{v}_B = 0$. Thus,

$$\begin{aligned} 0 &= \mathbf{v}_B^T B \mathbf{v}_B = \mathbf{v}_B^T \left(\sum_{\{i,j\}\in\mathcal{E}_M} E_{ij} + \sum_{\{i,j\}\in\mathcal{E}_B\setminus\mathcal{E}_M} E_{ij} \right) \mathbf{v}_B \\ &= \mathbf{v}_M^T \left(\sum_{\{i,j\}\in\mathcal{E}_M} E_{ij} \right) \mathbf{v}_M + \underline{\mathbf{v}}^T F \underline{\mathbf{v}} \end{aligned}$$

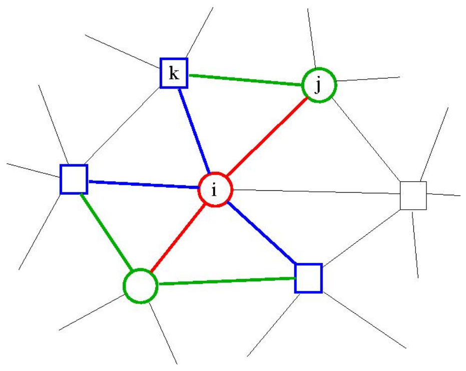
for an SPSD matrix F and adequate restriction $\underline{\mathbf{v}}$ of the vector $\mathbf{v}_B$. This proves $\mathbf{v}_M^T M \mathbf{v}_M = 0$ and thus (14). □

The task is now to define suitable computational molecules for building interpolation. Assume that "weak" and "strong" edges have been identified, the coarse grid has been selected, and a set of edge matrices is available. Then for any f-node i (to which interpolation is desired) we define a so-called interpolation molecule

$$M(i) := \sum_{k\in\mathcal{S}_i^c} E_{ik} + \sum_{j\in\mathcal{N}_i^f:\,\exists k\in\mathcal{S}_i^c\cap\mathcal{N}_j} E_{ij} + \sum_{k\in\mathcal{S}_i^c\cap\mathcal{N}_j:\,j\in\mathcal{N}_i^f} E_{jk}. \tag{15}$$

This molecule arises from assembling all edge matrices associated with three types of edges: The first sum corresponds to the strong edges connecting node i to some coarse direct neighbor k (*interpolatory edges*). The second sum represents edges connecting the considered f-node i to any of its fine direct neighbors j being directly connected to at least one c-node k that is strongly connected to node i. Finally, the last sum in (15) corresponds to these latter mentioned connections (edges) between fine direct neighbors j and strongly connected coarse direct neighbors k of node i. The formation of interpolation molecules is illustrated in Figure 1.

Figure 1: Formation of interpolation molecule



4.2 Interpolation rule

Element interpolation has been established in connection with so-called AMGe methods [6, 14]. This technique is based on a heuristic for SPD matrices that takes into account the nature of algebraically smooth error, cf. (5): Provided that a standard smoother is used, error that is slow to converge in energy norm corresponds to the lower part of the spectrum. Hence, one tries to fit interpolation to these low-energy modes, in particular, to the (near) null space components. The key idea is to construct local neighborhood matrices that represent the correct coupling between any given fine node and its interpolatory coarse (neighbor) nodes. In AMGe these neighborhood matrices are local versions of the stiffness matrix, i.e., small collections of element matrices. We propose the usage of the interpolation molecule (15), instead.

For a given f-node i let

$$M(i) = M = \begin{pmatrix} M_{ff} & M_{fc} \\ M_{cf} & M_{cc} \end{pmatrix} \tag{16}$$

be the interpolation molecule where the 2×2 block structure in (16) corresponds to the n_M^f f-nodes and the n_M^c c-nodes the molecule is based on. Then there is a bijection between the local and the global ordering of these nodes, which maps the global number i to some local number i', $1 \le i' \le n_M^f$. Consider now the small-sized (local) interpolation matrix

$$P_M = P = \begin{pmatrix} P_{fc} \\ I_{cc} \end{pmatrix} \tag{17}$$

associated with (16). The $n_M^f \times n_M^c$ submatrix P_{fc} produces interpolation in the f-nodes; for the c-nodes P equals the identity. Under the assumption that M is SPSD the AMGe interpolation concept can be applied directly [6, 11]:

For any vector $\mathbf{e}^T = (\mathbf{e}_f^T, \mathbf{e}_c^T) \perp \ker(M)$ we denote by

$$\mathbf{d}_f := \mathbf{e}_f - P_{fc}\mathbf{e}_c \tag{18}$$

the defect of (local) interpolation. With the objective of realizing the AMGe heuristic we choose P_{fc} to be the argument that minimizes

$$\max_{\mathbf{e}\perp\ker(M)} \frac{(\mathbf{e}_f - P_{fc}\mathbf{e}_c)^T(\mathbf{e}_f - P_{fc}\mathbf{e}_c)}{\mathbf{e}^T M \mathbf{e}}. \tag{19}$$

Using the substitutions (18) and

$$G := P_{fc}^T M_{ff} P_{fc} + P_{fc}^T M_{fc} + M_{cf} P_{fc} + M_{cc} \tag{20}$$

we have the follwing equivalence for (19):

$$\max_{\mathbf{d}_f,\mathbf{e}_c} \frac{\mathbf{d}_f^T\mathbf{d}_f}{\begin{pmatrix} \mathbf{d}_f + P_{fc}\mathbf{e}_c \\ \mathbf{e}_c \end{pmatrix}^T \begin{pmatrix} M_{ff} & M_{fc} \\ M_{cf} & M_{cc} \end{pmatrix} \begin{pmatrix} \mathbf{d}_f + P_{fc}\mathbf{e}_c \\ \mathbf{e}_c \end{pmatrix}}$$

$$= \max_{\mathbf{d}_f,\mathbf{e}_c} \frac{\mathbf{d}_f^T\mathbf{d}_f}{\langle M_{ff}(\mathbf{d}_f + P_{fc}\mathbf{e}_c), \mathbf{d}_f + P_{fc}\mathbf{e}_c\rangle + 2\langle M_{fc}\mathbf{e}_c, \mathbf{d}_f + P_{fc}\mathbf{e}_c\rangle + \langle M_{cc}\mathbf{e}_c, \mathbf{e}_c\rangle}$$

$$= \max_{\mathbf{d}_f,\mathbf{e}_c} \frac{\mathbf{d}_f^T\mathbf{d}_f}{\begin{pmatrix} \mathbf{d}_f \\ \mathbf{e}_c \end{pmatrix}^T B \begin{pmatrix} \mathbf{d}_f \\ \mathbf{e}_c \end{pmatrix}} \tag{21}$$

where

$$B = \begin{pmatrix} M_{ff} & M_{ff}P_{fc} + M_{fc} \\ P_{fc}^T M_{ff} + M_{cf} & G \end{pmatrix} \tag{22}$$

is SPSD. Hence

$$\min_{P_{fc}} \max_{\mathbf{d}_f,\mathbf{e}_c} \frac{\mathbf{d}_f^T\mathbf{d}_f}{\begin{pmatrix} \mathbf{d}_f \\ \mathbf{e}_c \end{pmatrix}^T B \begin{pmatrix} \mathbf{d}_f \\ \mathbf{e}_c \end{pmatrix}} = \min_{P_{fc}} \max_{\mathbf{d}_f} \frac{\mathbf{d}_f^T\mathbf{d}_f}{\min_{\mathbf{e}_c} \begin{pmatrix} \mathbf{d}_f \\ \mathbf{e}_c \end{pmatrix}^T B \begin{pmatrix} \mathbf{d}_f \\ \mathbf{e}_c \end{pmatrix}}$$

$$= \min_{P_{fc}} \max_{\mathbf{d}_f} \frac{\mathbf{d}_f^T\mathbf{d}_f}{\mathbf{d}_f^T \left[M_{ff} - (M_{ff}P_{fc} + M_{fc})G^{-1}(P_{fc}^T M_{ff} + M_{cf}) \right] \mathbf{d}_f}. \tag{23}$$

Assuming that M_{ff} and G both are SPD we observe that the denominator of (23) for an arbitrary but fixed vector $\mathbf{d}_f$ is maximized and thus the minimum is attained for

$$P_{fc} := -M_{ff}^{-1} M_{fc}, \tag{24}$$

which results in the value $1/(\lambda_{\min}(M_{ff}))$. This motivates to choose the interpolation coefficients for node i to equal the i'-th row of (24).

Remark 4.1 *In case the interpolation molecule M defined via (15) is not SPSD the interpolation coefficients can be computed from $M' = M^2$ which then simply plays the role of M in (24). Actually, this results in the interpolation rule*

$$P_{fc} := -(M_{ff}^2 + M_{fc}M_{cf})^{-1}(M_{ff}M_{fc} + M_{fc}M_{cc}), \tag{25}$$

which provides the minimizer of

$$\max_{\mathbf{e} \perp \ker(M^2)} \frac{(\mathbf{e}_f - P_{fc}\mathbf{e}_c)^T(\mathbf{e}_f - P_{fc}\mathbf{e}_c)}{\mathbf{e}^T M^2 \mathbf{e}}.$$

For a more general framework of AMG, including convergence analysis, we refer to [11].

5 Multilevel method

In this section we will outline the multilevel procedure for AMGm (Algebraic MultiGrid based on computational molecules). So far we discussed how to disassemble element matrices into edge matrices that can be utilized in the coarse-grid selection process as well as in the interpolation set-up, resulting in a new two-level method. However, assuming that the individual element matrices are given for the initial (fine) grid only, we need some technique for generating course-edge matrices in order to enable recursion and finally define a multilevel algorithm on this basis.

5.1 Generation of coarse-edge matrices

The construction of edge matrices as described in Section 2 is such that their total contributions exactly add up to the fine-grid stiffness matrix (without any essential boundary conditions imposed), i.e., the global stiffness matrix A in (1) is alternatively composed of the (level-zero) edge-matrices, instead of the element matrices. However, it is not intended to assemble the coarse-grid operators from coarse-edge matrices, in the present approach. Hence, we give up this property (at coarse levels) in favour of the following practicable procedure for an inexpensive computation of coarse-edge matrices.

- Firstly, one generates a coarse edge connecting any pair of c-nodes $\{i, j\}$ if either or both of the statements below are true:

    1. i and j are directly connected on the fine grid $\mathcal{D}$
    2. $\exists k \in \mathcal{D}_f$: $\{i, j\} \subset \mathcal{S}_k^c$

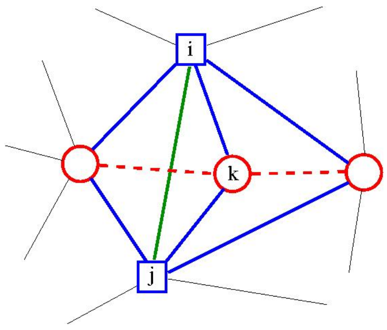
- Secondly, for any coarse edge $\{i, j\}$ one forms a specific computational molecule $M^{(i,j)}$, which is a *pre-stage* for the computation of the corresponding edge matrix; here, the molecule $M^{(i,j)}$ accumulates the contributions (edge-matrices) of all edges that yield paths of length two, starting in node i, passing some f-node k, and ending in node j, including also the contribution of a *direct* edge $\{i, j\}$ if there is one, see Figure 2.

- Finally, one generates the coarse-edge matrix by computing the Schur complement of $M^{(i,j)}$ with respect to its two c-nodes (eliminating all dofs associated with all of its f-nodes), i.e.,

$$E_{ij}^c := M_{cc}^{(i,j)} - M_{cf}^{(i,j)} \left(M_{ff}^{(i,j)}\right)^{-1} M_{fc}^{(i,j)}. \qquad (26)$$

Thus, another particular class of molecules serves for computing the coarse-edge matrices in our method. As can be seen from Figure 2 the important pairs of edges in these molecules are those connecting f-nodes k to both of the c-nodes i and j, which are going to be connected via a coarse edge. Additionally, one could take into account the (fine) edges connecting the corresponding f-nodes among each other, as indicated by the dotted lines in Figure 2. However, according to our experience this modification does not improve the results significantly (at least not for the test problems considered in the next section) but it slightly increases the set-up costs. That is why we prefer to assemble $M^{(i,j)}$ without any f-f connections, resulting in a cheaper evaluation of (26).

Figure 2: Basic molecule for coarse-edge matrix



5.2 AMGm

Regarding the multilevel algorithm, we notice that the AMGm method agrees with classical AMG, except for the coarse-grid selection and the interpolation component, which are controlled by edge matrices in case of AMGm. One can also view this as involving an auxiliary problem–the one determined by the edge matrices–in the coarsening process. The coarse-grid matrices, however, are still computed via the usual Galerkin tripple matrix product, i.e., $A_{k+1} = P_k^T A_k P_k$ for all levels $k = 0, 1, \dots, l-1$.

6 Numerical experiments

For the numerical experiments presented in this section we considered the boundary-value problem

$$-\nabla \cdot [C\nabla \mathbf{u}] = f \quad \text{in} \quad \Omega \subset \mathrm{R}^2 \tag{27}$$

$$u = g \quad \text{on} \quad \Gamma_D \subset \partial\Omega \tag{28}$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on} \quad \Gamma_N \subset \partial\Omega \setminus \Gamma_1. \tag{29}$$

Two different specifications of the matrix C, the right-hand side f, the domain Ω, and of the boundary conditions yield the considered test problems:

Problem 1

$$C = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}, \quad f = 0, \quad \Omega = (-3,3)\times(-3,3) \setminus (\Omega_1 \cup \Omega_2),$$

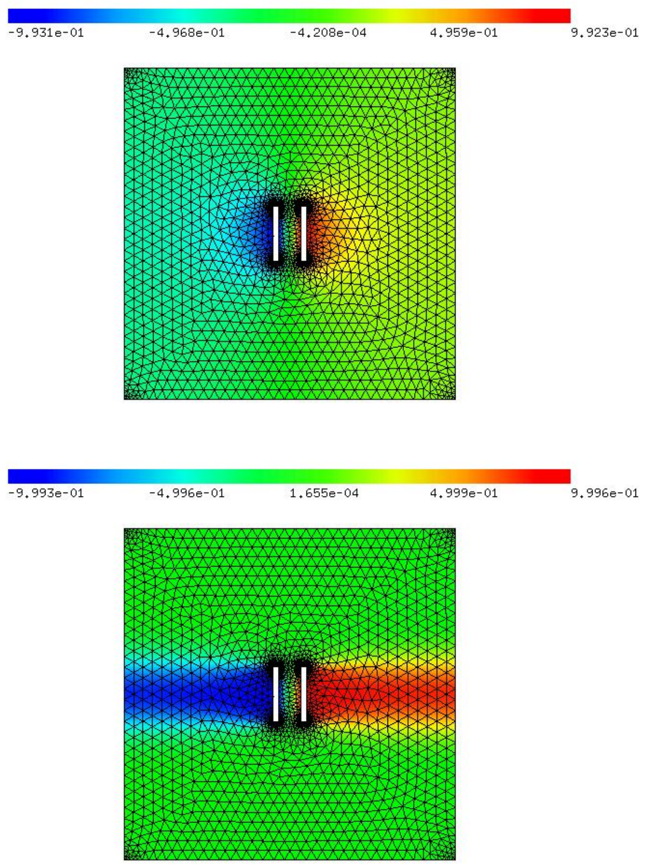
where $\Omega_1 = (0.2, 0.3)\times(-0.5, 0.5)$, $\Omega_2 = (-0.3, -0.2)\times(-0.5, 0.5)$,

$\Gamma_N = \partial((-3,3)\times(-3,3))$, $\Gamma_D = \partial\Omega_1 \cup \partial\Omega_2$, *and*

$$g = \begin{cases} 1 & on \quad \partial\Omega_1 \\ -1 & on \quad \partial\Omega_2 \end{cases}$$

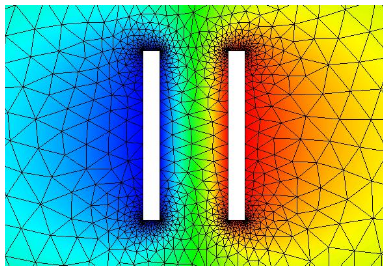
The solution of this problem–describing the potential of the electric field of a capacitor embedded in an (an)isotropic material–is plotted in Figure 3.

Figure 3: Problem 1: solution for $\epsilon = 1$ and $\epsilon = 0.01$



For discretization we used a finite element space of piecewise linear functions with Lagrangian basis, where the underlying locally refined triangular mesh was generated using the NETGEN[2] mesh generator [19] used in NGSolve[3], see Figure 4.

Figure 4: Problem 1: locally refined unstructured mesh



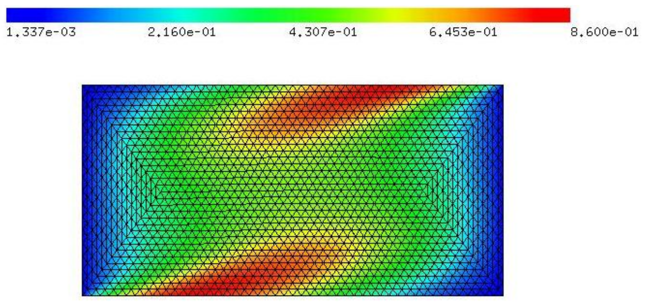
Problem 2

$$C = \begin{pmatrix} \epsilon + (\cos\Phi)^2 & \sin\Phi\cos\Phi \\ \sin\Phi\cos\Phi & \epsilon + (\sin\Phi)^2 \end{pmatrix}, \quad f = 1, \quad \Omega = (0,2)\times(0,1),$$

$\Gamma_N = \{(x,y) : 0 \le x \le 2 \text{ } and \text{ } y \in \{0,1\}\},\ \ \Gamma_D = \partial\Omega \setminus \Gamma_N,\ and\ \ g = 0.$

Again we used linear shape-functions, but this time for a quasi-uniform (structured) triangular mesh, in order to compute a numerical soultion by the finite element method, see Figure 5. A value of $\pi/12$ was chosen for the angle Φ causing the direction of anisotropy to be not aligned with the mesh.

Figure 5: Problem 2: solution for $\epsilon = 0.01$



Note that both problems result in stiffness matrices that are not contained in the class of M-matrices. Moreover, the variation from an M-matrix

[2]http://www.hpfem.jku.at/netgen/index.html
[3]http://www.hpfem.jku.at/ngsolve/index.html

increases when ϵ tends to zero, i.e., the positive off-diagonal entries gain weight in this case. This usually makes the problem harder to solve for (algebraic) multigrid methods.

We focused on a conjugate gradient solver with a single AMGm iteration acting as a preconditioner (PCG). In the first experiment we studied the convergence of this type of PCG method for Problem 1, using either a V(1,1), or a V(2,2), or a W(1,1) cycle, performing one or two symmetric Gauß-Seidel pre- and post-smoothing step(s). Table 1 contains the number of PCG iterations that was required to reduce the residual norm by a factor 10^{-8}, the average convergence factor, as well as the *grid complexity* σ^{Ω} and the *operator complexity* σ^{A}.[4]

Table 1: AMGm convergence results for Problem 1

#elements		4062		16248		64992		259968	
#levels		3		4		6		8	
$\epsilon = 1$:	V(1,1)	7	0.06	9	0.11	10	0.15	11	0.18
	V(2,2)	5	0.03	7	0.05	7	0.07	9	0.11
	W(1,1)	6	0.04	8	0.08	8	0.08	8	0.09
	σ^{Ω}	1.62		1.72		1.77		1.76	
	σ^{A}	2.24		2.61		2.88		2.90	
$\epsilon = 0.5$:	V(1,1)	6	0.05	8	0.10	10	0.15	11	0.18
	V(2,2)	5	0.03	6	0.05	8	0.08	9	0.11
	W(1,1)	6	0.04	7	0.07	8	0.08	8	0.09
	σ^{Ω}	1.72		1.82		1.88		1.86	
	σ^{A}	2.46		2.87		3.21		3.24	
$\epsilon = 0.1$:	V(1,1)	8	0.10	10	0.15	11	0.17	11	0.17
	V(2,2)	5	0.03	8	0.08	8	0.09	9	0.11
	W(1,1)	7	0.07	9	0.12	8	0.09	7	0.06
	σ^{Ω}	1.79		1.84		1.89		1.88	
	σ^{A}	2.60		2.87		3.14		3.17	
$\epsilon = 0.05$:	V(1,1)	8	0.10	12	0.20	11	0.18	12	0.21
	V(2,2)	6	0.03	8	0.10	9	0.11	10	0.14
	W(1,1)	7	0.07	10	0.15	8	0.08	7	0.07
	σ^{Ω}	1.80		1.85		1.91		1.90	
	σ^{A}	2.62		2.90		3.23		3.26	
$\epsilon = 0.01$:	V(1,1)	12	0.20	15	0.28	16	0.30	18	0.35
	V(2,2)	7	0.07	11	0.17	12	0.21	14	0.25
	W(1,1)	10	0.15	12	0.21	11	0.18	11	0.17
	σ^{Ω}	1.79		1.88		1.93		1.93	
	σ^{A}	2.60		3.02		3.33		3.39	

[4] σ^{Ω} is the ratio of the total number of points on all grids to that on the fine grid, whereas σ^{A} is the ratio of the total number of nonzero entries in all matrices to that in the fine-grid matrix.

For the second experiment we decided to compare the performance of AMGm to that of classical AMG (both used as a preconditioner for conjugate gradients) when solving Problem 2 on a quasiuniform mesh with decreasing mesh size, i.e., 49152, 196608, and 786432 elements, initializing the iteration with the zero start vector again. In Tables 2–4 we list the number of PCG iterations that reduced the norm of the initial residual by a factor 10^{-6}. In order to get a comparison in terms of computing time, we used the PCG method with classical AMG† preconditioning as implemented in the commercial software package FEMLAB version 3.1.[5] We used the same setting (except for the threshold parameter θ, which was 1/3 for AMGm and 1/4 for AMG†) for both methods. The solution time provided in the respective right column includes the set-up time for both preconditioners. All computations were performed on a 2.0 GHz Unix-machine with 1.5 GB RAM.

Table 2: Performance comparison for Problem 2: 49152 elements (5 levels)

	AMG†						AMGm					
ϵ	V(1,1)		V(2,2)		W(1,1)		V(1,1)		V(2,2)		W(1,1)	
1.0	15	1.65	13	1.78	10	1.79	11	1.48	9	1.46	9	1.70
0.5	15	1.88	13	2.00	11	2.19	9	1.23	8	1.34	7	1.31
0.1	20	2.04	18	2.53	14	2.30	11	1.34	9	1.37	9	1.42
0.05	23	2.17	21	2.30	16	2.44	12	1.38	10	1.43	9	1.40
0.01	34	2.52	32	2.78	24	2.91	15	1.50	12	1.53	11	1.53

Table 3: Performance comparison for Problem 2: 196608 elements (7 levels)

	AMG†						AMGm					
ϵ	V(1,1)		V(2,2)		W(1,1)		V(1,1)		V(2,2)		W(1,1)	
1.0	36	15.3	34	17.4	19	15.7	12	7.6	10	8.4	9	8.6
0.5	20	15.3	19	18.0	12	16.8	10	6.5	9	7.4	8	7.0
0.1	28	16.3	26	18.0	17	18.8	12	7.0	10	7.8	9	7.5
0.05	31	16.6	29	18.2	20	19.0	13	7.3	11	8.1	10	7.8
0.01	44	18.5	41	19.9	30	24.9	17	8.4	14	9.3	12	8.8

7 Concluding remarks

The application of any AMG method splits into a set-up phase and a solution phase. Hence, solving a linear system (1) for a single right-hand side, the computational costs of these two phases have to be well balanced in order to achieve a low total effort. It is one of the main concerns of the

[5]http://www.comsol.com

Table 4: Performance comparison for Problem 2: 786432 elements (9 levels)

	AMG[†]						AMGm					
ϵ	V(1,1)		V(2,2)		W(1,1)		V(1,1)		V(2,2)		W(1,1)	
1.0	79	227	72	240	35	236	12	35	10	40	9	40
0.5	56	300	51	304	20	294	12	33	10	36	8	34
0.1	52	294	50	302	21	295	12	33	10	37	9	35
0.05	60	293	56	306	27	300	14	35	12	41	10	37
0.01	60	293	56	307	36	317	21	44	17	49	13	43

present paper to introduce a robust method with a fast and low-complexity set-up. In fact, the set-up of AMGm took approximately half the (total) time reported in Tables 2–4. The improved robustness is achieved by controlling the coarse-grid selection and the interpolation component via edge-matrices. Regarding Problem 2, Tables 2–4 report much faster convergence (and thence shorter solution time) when using AMGm instead of classical AMG. Regarding Problem 1, the convergence rates reported in Table 1, satisfy for all tested cycles.[6] The grid- and operator complexity thereby incurred is in the range of that for classical AMG [17].

Future investigations will deal with the generalization of the presented AMGm methodology to cover also systems of PDEs, e.g., arising in structural mechanics.

Acknowledgements. The authors would like to thank Josef Synka who carried out the computations with FEMLAB, which served for the presented performance comparison.

This work was supported by the Austrian Academy of Sciences and by the Austrian Science Fund (FWF) in the frame of special research area F013.

References

[1] O. Axelsson: Iterative Solution Methods. Cambridge University Press, 1994.

[2] A. Brandt: Algebraic multigrid theory: the symmetric case. Preliminary Proceedings for the International Multigrid Conference, Copper Mountain, Colorado, 1983.

[3] A. Brandt: Algebraic multigrid theory: the symmetric case. Appl. Math. Comput. 19 (1986), 23-56.

[6] In particular, as one would expect, the iteration count for the W(1,1) cycle is (almost) independent of the mesh size for both problems.

[4] A. Brandt, S.F. McCormick, and J.W. Ruge: Algebraic multigrid (AMG) for automatic multigrid solutions with applications to geodetic computations. Report, Inst. for Computational Studies, Fort Collins, Colorado, 1982.

[5] A. Brandt, S.F. McCormick, and J.W. Ruge: Algebraic multigrid (AMG) for sparse matrix equations. In: Sparsity and Its Applications, D.J. Evans, ed., Cambridge Univ. Press, Cambridge, 1985, 257-284.

[6] M. Brezina, A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, and J.W. Ruge: Algebraic multigrid based on element interpolation (AMGe). SIAM J. Sci. Comput. 22 (2000), 1570-1592.

[7] W.L. Briggs, V.E. Henson, and S.F. McCormick: A Multigrid Tutorial. 2nd Edition. SIAM Books, Philadelphia, 2001.

[8] T.F. Chan and P. Vanek: Detection of strong coupling in algebraic multigrid solvers. In Multigrid Methods VI, vol. 14, Springer–Verlag, Berlin, 2000, 11-23.

[9] T. Chartier, R.D. Falgout, V.E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P.S. Vassilevski: Spectral AMGe (ρAMGe). To appear in SIAM J. Sci. Comput.

[10] A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, G.N. Miranda, and J.W. Ruge: Robustness and scalability of algebraic multigrid. SIAM J. Sci. Stat. Comput. 21 (2000), 1886-1908.

[11] R.D. Falgout and P.S. Vassilevski: On generalizing the algebraic multigrid framework. SIAM Journal on Numerical Analysis 42 (2004), 1669-1693.

[12] G. Haase, U. Langer, S. Reitzinger, J. Schöberl: Algebraic multigrid methods based on element preconditioning. International Journal of Computer Mathematics 78 (2004), 575-598.

[13] V.E. Henson and P. Vassilevski: Element-free AMGe: general algorithms for computing the interpolation weights in AMG. SIAM J. Sci. Comput. 23 (2001), 629-650.

[14] J.E. Jones and P. Vassilevski: AMGe based on element agglomeration. SIAM J. Sci. Comput. 23 (2001), 109-133.

[15] U. Langer, S. Reitzinger, J. Schicho: Symbolic methods for the element preconditioning technique. In: Proc. SNSC Hagenberg, U. Langer and F. Winkler, eds., Springer, 2002.

[16] J. Mandel, M. Brezina, P. Vaněk: Energy optimization of algebraic multigrid bases. Computing 62 (1999), 205-228.

[17] J.W. Ruge and K. Stüben: Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). In: Multigrid Methods for Integral and Differential Equations, D.J. Paddon and H. Holstein, eds., The Institute of Mathematics and Its Applications Conference Series, Clarendon Press, Oxford, 1985, 169-212.

[18] J.W. Ruge and K. Stüben: Algebraic multigrid (AMG). In: Multigrid Methods, vol. 3 of Frontiers in Applied Mathematics, S.F. McCormick, ed., SIAM, Philadelphia, 1987, 73-130.

[19] J. Schöberl: NETGEN–An advancing front 2D/3D-mesh generator based on abstract rules. Comput. Visual. Sci. (1997), 41-52.

[20] K. Stüben: Algebraic multigrid (AMG): experiences and comparisons. Appl. Math. Comput. 13 (1983), 419-452.

[21] R.S. Tuminaro and C. Tong: Parallel smoothed aggregation multigrid: aggregation strategies on massively parallel machines. Report, Sandia National Laboratories, 2000.

[22] R.S. Varga: Matrix Iterative Analysis. Prentice-Hall, 1962.

[23] P. Vaněk, M. Brezina, and J. Mandel: Convergence of algebraic multigrid based on smoothed aggregation. Numer. Math. 88 (2001), 559-579.

[24] P. Vaněk, J. Mandel, and M. Brezina: Algebraic multigrid based on smoothed aggregation for second and fourth order problems. Computing 56 (1996), 179-196.

[25] W.L. Wan, T.F. Chan, and B. Smith: An energy-minimizing interpolation for robust multigrid methods. Siam J. Sci. Comput. 21 (2000), 1632-1649.

Johannes Kraus and Josef Schicho
Johann Radon Institute for Computational and Applied Mathematics
Altenbergerstraße 69
A-4040 Linz
Austria
johannes.kraus@oeaw.ac.at, josef.schicho@oeaw.ac.at