Algebraic Multigrid for Discontinuous Galerkin Discretizations

Markus Blatt
Dr. Markus Blatt - HPC-Simulation Software & Services
16 PUBLICATIONS **809** CITATIONS
SEE PROFILE

Robert Scheichl
University of Bath
103 PUBLICATIONS **1,917** CITATIONS
SEE PROFILE

NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 0000; **00**:1–22
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla

Algebraic Multigrid for Discontinuous Galerkin Discretizations

Peter Bastian[1], Markus Blatt[1]*and Robert Scheichl[2]

[1] *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Ruprechts–Karls–Universität Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany*
[2] *Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, United Kingdom*

SUMMARY

We present a new algebraic multigrid (AMG) algorithm for the solution of linear systems arising from discontinuous Galerkin discretizations of heterogeneous elliptic problems. The algorithm is based on the idea of subspace corrections and the first coarse level space is the subspace spanned by continuous linear basis functions. The linear system associated with this space is constructed algebraically using a Galerkin approach with the natural embedding as the prolongation operator. For the construction of the linear systems on the subsequent coarser levels non-smoothed aggregation AMG techniques are used. In a series of numerical experiments we establish the efficiency and robustness of the proposed method for various symmetric and non-symmetric interior penalty discontinuous Galerkin methods, including several model problems with complicated, high-contrast jumps in the coefficients. The solver is robust with respect to an increase in the polynomial degree of the discontinuous Galerkin approximation space (at least up to degree 6), computationally efficient, and it is affected only mildly by the coefficient jumps and by the mesh size h (i.e. $O(\log h^{-1})$ number of iterations). Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: interior penalty discontinuous Galerkin methods; elliptic PDEs; robustness for heterogeneous coefficients; subspace correction method; aggregation-type algebraic multigrid

1. INTRODUCTION

In the seventies the development of the discontinuous Galerkin (DG) finite element method for discretizing partial differential equations started (cf. [1, 2, 3, 4, 5, 6]). It is based on a totally discontinuous finite element space and has many advantageous properties. Due to the missing continuity constraint of the basis functions its use with non-conforming unstructured grids is straight forward and allows for easy mesh adaptation techniques. The choice of the basis functions used is flexible and allows for easy adaptation in the polynomial order. For flow problems an important property of the DG discretizations is its element-wise mass conservation. A disadvantage is that

*Correspondence to: Markus Blatt, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Ruprechts–Karls–Universität Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany, E-mail: Markus.Blatt@iwr.uni-heidelberg.de the resulting linear systems have a higher number of degrees of freedom when compared with the continuous methods. This leads to even larger and more ill-conditioned systems and makes the use of optimal solvers even more mandatory. The quest for such solvers is still ongoing as recent publications such as [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21] show. The most promising types of solvers in this area are multigrid and domain decomposition methods.

Our solver is an extension of the algebraic multigrid method based on non-smoothed aggregation presented in [22]. It is used as an inexact solver to compute a correction in the subspace of continuous linear basis functions on the original mesh. Therefore the method can be seen as an efficient implementation of a subspace correction method (cf. [23]) for discontinuous Galerkin. Especially for higher order trial functions this leads to a tremendous reduction of the number of degrees of freedom used on the finer levels. This approach works for both symmetric and non-symmetric interior penalty discontinuous Galerkin discretizations provided the penalty parameter is chosen sufficiently large. For the method of Oden and Baumann, that is lacking the penalty parameter, additional measures have to be taken. On the finest level we employ overlapping as well as non-overlapping Schwarz methods as smoothers. To construct the local subdomains needed in these smoothers we use the greedy aggregation algorithm presented in [22]. The same aggregation algorithm is then used to set up the coarser levels for the conforming subspace problem. We consider only conforming grids here, but an extension to nonconforming grids is easily conceivable, e.g. in the auxiliary space framework proposed in [24].

In a series of numerical experiments on two and three-dimensional test problems with constant, checkerboard and random coefficients, we establish the efficiency and robustness of the proposed approach.

We start in the next section by presenting the model problem which we investigate and the discontinuous Galerkin discretization of it. Then we describe the multigrid algorithm in Section 3. After presenting numerical convergence results for our solver in Section 4, we conclude the paper with a summary of the achievements and an outlook onto future perspectives.

2. MODEL PROBLEM AND ITS DISCRETIZATION

Let Ω be a polygonal domain in $\mathbb{R}^d$, $d = 1, 2, 3$, with boundary $\partial\Omega = \bar{\Gamma}_D \cup \bar{\Gamma}_N$, such that Γ_D has nonzero measure and $\Gamma_D \cap \Gamma_N = \emptyset$. We consider the heterogeneous elliptic model problem

$$-\nabla \cdot (K\nabla u) = f \qquad \text{in } \Omega, \tag{1a}$$

$$u = g \qquad \text{on } \Gamma_D, \tag{1b}$$

$$-(K\nabla u) \cdot n = j \qquad \text{on } \Gamma_N, \tag{1c}$$

where K is a uniformly symmetric positive definite permeability tensor which may be highly heterogeneous. Given an extension u_g of the Dirichlet data to the complete domain Ω, as well as the space $V = H^1_D(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \Gamma_D\}$, the weak formulation of (1) consists of finding $u \in u_g + V$ such that

$$(K\nabla u, \nabla v)_{0,\Omega} = (f, v)_{0,\Omega}\,, \qquad \forall v \in V.$$

Here $(.,.)_{0,\Omega}$ denotes the L^2-scalar product on Ω.

2.1. Discontinuous Galerkin Method

Discontinuous Galerkin (DG) methods are a class of numerical schemes that has been studied extensively in the last two decades, see [25] for an introduction and below for some key references. In particular, we use the family of weighted interior penalty discontinuous Galerkin (WIPG) schemes introduced in [26].

Let $\{\mathcal{T}_h\}_{h>0}$ denote a family of conforming triangulations of the domain Ω. An element of the triangulation is denoted by T, h_T is its diameter, $|T|$ its volume, and n_T its unit outer normal vector. F is called an "interior face" independent of the dimension if there are two elements $T^-(F), T^+(F) \in \mathcal{T}_h$ with $T^-(F) \cap T^+(F) = F$ and F has nonzero measure. All interior faces are collected in the set $\mathcal{F}_h^i$. The intersection of $T \in \mathcal{T}_h$ with the boundary $\partial\Omega$ of non-zero measure is called a boundary face. All boundary faces on the Dirichlet boundary are collected in the set $\mathcal{F}_h^{\Gamma_D}$, and those on the Neumann boundary in $\mathcal{F}_h^{\Gamma_N}$. The set of all faces is thus $\mathcal{F} = \mathcal{F}_h^i \cup \mathcal{F}_h^{\Gamma_D} \cup \mathcal{F}_h^{\Gamma_N}$. The diameter of a face is denoted by h_F and its volume by $|F|$. With each $F \in \mathcal{F}$ we associate a unit normal vector n_F (depending on position if F is curved) oriented from $T^-(F)$ to $T^+(F)$ for an interior face. For a boundary face F, n_F coincides with the exterior unit normal on the face.

The DG approximation space of degree k associated with $\mathcal{T}_h$ is now defined as

$$V_h^k = \{v \in L^2(\Omega) \,:\, \forall T \in \mathcal{T}_h, v|_T \in \mathbb{P}_k\},$$

where $\mathbb{P}_k = \{p : p = \sum_{\|\alpha\|_1 \le k} c_\alpha x^\alpha\}$ are the polynomials of at most degree k. On an interior face F a function $v \in V_h$ is two-valued and its values v^- and v^+ are the restrictions from $T^-(F)$ and $T^+(F)$, respectively. For $F \in \mathcal{F}_h^i$ and $v \in V_h$ we introduce the jump and the weighted average

$$[\![v]\!]_F = v^- - v^+, \qquad \{v\}_\omega = \omega^- v^- + \omega^+ v^+,$$

with the weights satisfying $\omega^- + \omega^+ = 1, \omega^-, \omega^+ \ge 0$.

In the WIPG schemes the discrete solution $u_h \in V_h^k$ is now chosen such that it satisfies the variational equation

$$a_h(u_h, v) = l_h(v), \qquad \forall v \in V_h^k, \tag{2}$$

with the bilinear form

$$\begin{aligned} a_h(u,v) = &\sum_{T\in\mathcal{T}_h} (K\nabla u, \nabla v)_{0,T} \\ &+ \sum_{F\in\mathcal{F}_h^i} \Big[\theta\big([\![u]\!], \{n_F^{\mathsf{T}} K\nabla v\}_\omega\big)_{0,F} - \big(\{n_F^{\mathsf{T}} K\nabla u\}_\omega, [\![v]\!]\big)_{0,F} + \gamma_F\big([\![u]\!], [\![v]\!]\big)_{0,F}\Big] \\ &+ \sum_{F\in\mathcal{F}_h^{\Gamma_D}} \Big[\theta\big(u, n_F^{\mathsf{T}} K\nabla v\big)_{0,F} - \big(n_F^{\mathsf{T}} K\nabla u, v\big)_{0,F} + \gamma_F(u,v)_{0,F}\Big] \end{aligned} \tag{3}$$

and the linear form

$$l_h(v) = \sum_{T\in\mathcal{T}_h} (f,v)_{0,T} + \sum_{F\in\mathcal{F}_h^{\Gamma_D}} \Big[\theta\big(g, n_F^{\mathsf{T}} K\nabla v\big)_{0,F} + \gamma_F(g,v)_{0,F}\Big] - \sum_{F\in\mathcal{F}_h^{\Gamma_N}} (j,v)_{0,F}. \tag{4}$$

Different choices of θ, of the penalty parameter γ_F, and of the weights $\omega^{\pm}$ lead to the following well known types of DG methods: A choice of

- $\theta = -1$, $\omega^{\pm} = 1/2$ and $\gamma_F \geq \gamma_0$ sufficiently large, leads to the symmetric interior penalty (IP or SIPG) method, [1, 2, 6];
- $\theta = +1$, $\omega^{\pm} = 1/2$ and $\gamma_F > 0$ leads to the non-symmetric interior penalty (NIPG) method, [5, 6];
- $\sigma = +1$, $\omega^{\pm} = 1/2$ and $\gamma_F = 0$ leads to the method of Baumann and Oden (OBB), see [3, 4, 5].

However, in the case of highly varying coefficients $K(x)$, as discussed in detail in [26], the weights $\omega^{\pm}$ need to be chosen differently and as a function of the permeability, i.e.

$$\omega^- = \frac{\delta^+_{Kn}}{\delta^-_{Kn} + \delta^+_{Kn}}, \qquad \omega^+ = \frac{\delta^-_{Kn}}{\delta^-_{Kn} + \delta^+_{Kn}},$$

with $\delta^{\pm}_{Kn} = n_F^{\mathrm{T}} K^{\pm} n_F$ for $F \in \mathcal{F}_h^i$ and $\delta_{Kn} = n_F^{\mathrm{T}} K n_F$ for $F \in \mathcal{F}_h^{\partial\Omega}$, leading to the WIPG family of [26] which we use in this publication.

The choice of the interior penalty parameter γ_F is crucial to ensure as much independence from the problem and mesh parameters as possible. We define the penalty parameter as

$$\gamma_F = \begin{cases} \alpha \, \dfrac{2\delta^-_{Kn}\delta^+_{Kn}}{\delta^-_{Kn} + \delta^+_{Kn}} \, k(k+d-1) \, \dfrac{|F|}{\min(|T^-(F)|, |T^+(F)|)}, & \forall F \in \mathcal{F}_h^i, \\ \alpha \, \delta_{Kn} \, k(k+d-1) \, \dfrac{|F|}{|T^-(F)|}, & \forall F \in \mathcal{F}_h^{\partial\Omega}. \end{cases} \tag{5}$$

with a user-defined parameter α. This choice is a combination of suggestions made in three different papers. The harmonic average of "normal" permeabilities was introduced and analyzed in [26], the dependence on the polynomial degree was analyzed in [27], and the mesh-dependence is taken from [28].

DG methods are particularly suited for heterogeneous elliptic problems due to their cell-wise conservation properties including the ability to handle full tensors. Approximation quality is comparable to mixed finite elements, see [29]. For proofs of the approximation properties of the above-mentioned methods see the well known publications [3, 5, 30].

2.2. *Block Notation of Algebraic Systems*

Choosing a basis $\Phi_h = \{\phi_1, \ldots, \phi_n\}$ for the DG approximation space V_h^k and expanding the solution in this basis as $u = \sum_{j=1}^n \mathbf{u}_j \phi_j$, the discrete variational problem (2) is equivalent to a system of linear equations

$$\mathbf{A}\mathbf{u} = \mathbf{f} \tag{6}$$

for the coefficients $\mathbf{u}$ with $a_{i,j} = a_h(\phi_j, \phi_i)$ and $f_i = l_h(\phi_i)$.

In DG methods a natural block structure is imposed on the stiffness matrix $\mathbf{A}$ by grouping all basis functions of an element together. Let us assume that (as usual) the basis Φ_h is chosen in such away that the support of any basis function $\phi_i \in \Phi_h$ is restricted to a single element $T \in \mathcal{T}_h$. Then, in order to efficiently deal with block-structured matrices we introduce the following notation. For any finite index set $I \subset \mathbb{N}$ we define the vector space $\mathbb{R}^I$ to be isomorphic to $\mathbb{R}^{|I|}$ with components indexed by $i \in I$. Thus $\mathbf{x} \in \mathbb{R}^I$ can be interpreted as a mapping $\mathbf{x} : I \to \mathbb{R}$ and $x_i = \mathbf{x}(i)$. In the same way, for any two finite index sets $I, J \subset \mathbb{N}$ we write $\mathbf{A} \in \mathbb{R}^{I \times J}$ with the interpretation $\mathbf{A} : I \times J \to \mathbb{R}$ and $a_{i,j} = \mathbf{A}(i,j)$. Finally, for any subset $I' \subseteq I$ we define the restriction matrix $\mathbf{R}_{I,I'} : \mathbb{R}^I \to \mathbb{R}^{I'}$ as $(\mathbf{R}_{I,I'}\mathbf{x})_i = x_i \, \forall i \in I'$.

Now, let $I_T = \{i \in \mathbb{N} : \text{supp } \phi_i \subseteq \overline{T}\}$. Then

$$\bigcup_{T \in \mathcal{T}_h} I_T = I = \{1, \ldots, n\} \quad \text{and} \quad I_T \cap I_{T'} = \emptyset, \, \forall T \neq T', \tag{7}$$

and so $\{I_T\}$ forms a partitioning of the index set I. Using this partitioning and imposing an ordering $\mathcal{T}_h = \{T_1, \ldots, T_m\}$ on the mesh elements, the linear system (6) can be written in block form

$$\begin{pmatrix} \mathbf{A}_{T_1,T_1} & \ldots & \mathbf{A}_{T_1,T_m} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{T_m,T_1} & \ldots & \mathbf{A}_{T_m,T_m} \end{pmatrix} \begin{pmatrix} \mathbf{u}_{T_1} \\ \vdots \\ \mathbf{u}_{T_m} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_{T_1} \\ \vdots \\ \mathbf{f}_{T_m} \end{pmatrix} \tag{8}$$

where $\mathbf{A}_{T_i,T_j} = \mathbf{R}_{I,I_{T_i}} \mathbf{A} \mathbf{R}^T_{I,I_{T_j}}$, $\mathbf{u}_{T_i} = \mathbf{R}_{I,I_{T_i}} \mathbf{u}$ and $\mathbf{f}_{T_j} = \mathbf{R}_{I,I_{T_j}} \mathbf{f}$.

3. ALGEBRAIC MULTIGRID METHOD BASED ON SUBSPACE CORRECTION

Our objective is to solve the linear system (6) efficiently using a multigrid algorithm. Geometric multigrid algorithms for the system (6) have been presented in [7, 9, 10, 11, 12, 13]. In this paper we follow the idea of implementing an efficient inexact subspace correction method (cf. [23]). Related theoretical work can be found in [18, 21].

In each iteration the following two steps are performed in a sequential (multiplicative) way:

1. Apply a small number of steps with an overlapping or non-overlapping Schwarz smoother on the system (6) exploiting the block structure of the matrix.
2. Compute an approximate coarse space correction in the subspace of conforming, piecewise linear finite element functions on the same grid using an aggregation-type algebraic multigrid method.

Let us recall the basic ideas of subspace correction methods (see [23] for details). Let V be a finite element space, $a_h(\cdot, \cdot) : V \times V \to \mathbb{R}$ a bilinear form, and $l_h(\cdot) : V \to V$ a linear form defining the finite element system. Furthermore, let us assume that $V_i \subset V$, $i = 0, \ldots, m$, are a finite number of subspaces of V with associated restriction operators $R_i : V \to V_i$. The prolongation or extension operator from V_i to V can be chosen to be R_i^{T}. The subspace correction method consists now in finding corrections $R_i^{\mathrm{T}} w_i$, with $w_i \in V_i$, to the current iterate $u \in V$ that satisfy

$$a(u + w_i, v) = l(v) \quad \forall v \in V_i \, .$$

In its additive version all corrections w_i, $i = 0, \ldots, m$, are first computed with u^k as the current iterate u. The new iterate is then defined as $u^{k+1} = u^k + \sum_{i=0}^m R_i^{\mathrm{T}} w_i$. In the multiplicative form, we need to choose an ordering and then loop over all subspaces V_i. In each step of the loop we use the previously computed iterate $u^{k+\frac{i}{m+1}}$ as the current iterate u when computing the correction $w_i \in V_i$. At the end of each step we set $u^{k+\frac{i+1}{m+1}} = u^{k+\frac{i}{m+1}} + R_i^{\mathrm{T}} w_i$.

We now make particular choices for the subspaces for our model problem in the case of the DG space $V = V_h^k$ on a conforming mesh $\mathcal{T}_h$. For all k, the space V_0 is chosen to be the subspace of continuous, piecewise linear finite elements for simplices or piecewise multi-linear finite elements for cubes associated with the same mesh $\mathcal{T}_h$. The subspaces V_i, $i = 1, \ldots, m$, on the other hand, consist of functions in V_h^k whose support is restricted to some (overlapping or non-overlapping) subdomains $\Omega_i \subset \Omega$, such that $\{\Omega_i\}_{i=1}^m$ forms a partitioning of Ω. To algebraically construct the overlapping subdomains we use a greedy aggregation algorithm which we describe in the next subsection.

3.1. Aggregation Algorithm

Several components in our solver rely heavily on a greedy and heuristic aggregation algorithm. It is a further development and extension of the version published by Raw (cf. [31]) for algebraic multigrid methods (see also [32]). We use it to construct overlapping subdomains for our Schwarz smoothers on the fine level, as well as to define the coarse levels in our algebraic multigrid algorithm for continuous piecewise linear elements.

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with a set of vertices $\mathcal{V}$ and edges $\mathcal{E}$ and let $w_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}$ and $w_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}$ be positive weight functions, to be defined later. These functions are used to classify the edges and vertices of our graph. Let

$$N(i) := \{j \in \mathcal{V} \mid \exists (j,i) \in \mathcal{E}\}$$

be the set of adjacent vertices of vertex i and let

$$\eta_{\max}(i) := \max_{k \in N(i)} \frac{w_{\mathcal{E}}((k,i)\, w_{\mathcal{E}}((i,k))}{w_{\mathcal{V}}(i)\, w_{\mathcal{V}}(k)}. \tag{9}$$

Definition 3.1. (a) An edge (j,i) is called *strong*, if and only if

$$\frac{w_{\mathcal{E}}((i,j))\, w_{\mathcal{E}}((j,i))}{w_{\mathcal{V}}(i)\, w_{\mathcal{V}}(j)} > \delta \, \min(\eta_{\max}(i), \eta_{\max}(j)), \tag{10}$$

for a given threshold $0 < \delta < 1$. We denote by $N_\delta(i) \subset N(i)$ the set of all vertices adjacent to i that are connected to it via a strong edge.

(b) A vertex i is called *isolated* if and only if $\eta_{\max}(i) < \beta$, for a prescribed threshold $0 < \beta \ll 1$. We denote by $I(V) \subset V$ the set of all isolated vertices of the graph.

Our greedy aggregation algorithm is described in Algorithm 1. Until all non-isolated vertices are aggregated, we start a new aggregate with a non-isolated vertex. The first aggregate is seeded with a vertex that has the least connections to other vertices. The other aggregates are seeded with vertices that are non-aggregated neighbor's of the most recently created aggregates. If no such vertex exists, we use a non-isolated, non-aggregated vertex that has the least connections to non-aggregated vertices. At the same time we associate the index of the seed vertex with this new aggregate and add

Algorithm 1 Build Aggregates

```
procedure AGGREGATION($\mathcal{V}$, $\mathcal{E}$, $s_{\text{min}}$, $s_{\text{max}}$, $d_{\text{max}}$)
    $U \leftarrow \mathcal{V} \setminus I(\mathcal{V})$                ▷ First candidates are non-isolated vertices
    $I \leftarrow \emptyset$                                    ▷ Coarse index set
    Select arbitrary seed $v \in \{u \in U : \#N_\delta(u) \leq \#N_\delta(w)\ \forall w \in \mathcal{V}\}$
    while $U \neq \emptyset$ do
        $\mathcal{A}_v \leftarrow \{v\}$
        $U \leftarrow U \setminus \mathcal{A}_v$
        $I \leftarrow I \cup \{v\}$
        GROWAGGREGATE($\mathcal{A}$, $\mathcal{V}$, $\mathcal{E}$, $s_{\text{min}}$, $d_{\text{max}}$, $U$)
        ROUNDAGGREGATE($\mathcal{A}$, $\mathcal{V}$, $\mathcal{E}$, $s_{\text{max}}$, $U$)
        if $\#\mathcal{A}_i = 1$ then                  ▷ Merge one vertex aggregate with neighbors
            $C \leftarrow \{\mathcal{A}_j : j \in I \setminus \{i\}$ and $\exists w \in \mathcal{A}_j$ with $w \in N_\delta(v)\}$
            if $C \neq \emptyset$ then
                Choose $\mathcal{A}_k \in C$
                $I \leftarrow I \setminus \{s\}$
                $\mathcal{A}_k \leftarrow \mathcal{A}_k \cup \mathcal{A}_s$
            end if
        end if
        if $U \neq \emptyset$ then
            Select arbitrary seed $v \in U \cap \{w : N(w) \cap \cup_{k \in I} \mathcal{A}_k \neq \emptyset\}$
        end if
    end while
    $U \leftarrow I(\mathcal{V})$                               ▷ Aggregate isolated vertices
    while $U \neq \emptyset$ do
        Select arbitrary seed $v \in U$
        $\mathcal{A}_v \leftarrow \{v\}$
        $U \leftarrow U \setminus \mathcal{A}_v$
        $I \leftarrow I \cup \{v\}$
        GROWISOAGGREGATE($\mathcal{A}$, $\mathcal{V}$, $\mathcal{E}$, $s_{\text{min}}$, $d_{\text{max}}$, $U$)
    end while
    $\mathcal{A} \leftarrow \{\mathcal{A}_i : i \in I\}$
    return $(\mathcal{A}, I)$
end procedure
```

it to the index set I. The algorithm returns both the index set I for the set of aggregates as well as the set $\mathcal{A} = \{\mathcal{A}_i : i \in I\}$ of all aggregates it has built.

The first step in the construction of an aggregate in Algorithm 1 is to add new vertices to our aggregate until we reach the minimal prescribed aggregate size s_{min}. This is outlined in Algorithm 2. When adding new vertices, we always choose those with the most strong connections to the vertices already in the aggregate. Here we give preference to vertices where both edge (i, j) and edge (j, i) are strong. The functions $\text{cons}_1(v, \mathcal{A})$ and $\text{cons}_2(v, \mathcal{A})$ return the number of one-way and two-way connections between the vertex v and all vertices of the aggregate $\mathcal{A}$, respectively. If there is more than one candidate, we choose the vertex that adds the least new connections between aggregates. As a measure for this we use the function $\text{connect}(v, \mathcal{A})$. It counts neighbors of v that are not yet aggregated or belong to an aggregate that is not yet connected to aggregate $\mathcal{A}$. Neighbors of v that belong to aggregates that are already connected to aggregate $\mathcal{A}$ are counted twice. In this way, we are able to choose among all candidates the vertex with the lowest number of connections to other not yet aggregated vertices that are neighbors of the aggregate. The function $\text{neighbors}(v, \mathcal{A})$

Algorithm 2 Grow Aggregate Step

```
function GrowAggregate($\mathcal{A}$, $\mathcal{V}$, $\mathcal{E}$, $s_{\min}$, $d_{\max}$, $U$)
    while $\#\mathcal{A} \leq s_{\min}$ do                    ▷ Makes aggregate $\mathcal{A}$ bigger until its size is $s_{\min}$
        $C_0 \leftarrow \{v \in N(\mathcal{A}) :\ \mathrm{diam}(\mathcal{A}, v) \leq d_{\max}\}$      ▷ Limit the diameter of the aggregate
        $C_1 \leftarrow \{v \in C_0 :\ \mathrm{cons}_2(v, \mathcal{A}) \geq\ \mathrm{cons}_2(w, \mathcal{A})\ \forall w \in N(\mathcal{A})\}$
        if $C_1 = \emptyset$ then                              ▷ No candidate with two-way connections
            $C_1 \leftarrow \{v \in C_0 :\ \mathrm{cons}_1(v, \mathcal{A}) \geq\ \mathrm{cons}_1(w, \mathcal{A})\ \forall w \in N(\mathcal{A})\}$
        end if
        if $\#C_1 > 1$ then                                    ▷ More than one candidate
            $C_1 \leftarrow \{v \in C_1 :\ \frac{\mathrm{connect}(v,\mathcal{A})}{N(v)} \geq \frac{\mathrm{connect}(w,\mathcal{A})}{N(w)}\ \forall w \in C_1\}$
        end if
        if $\#C_1 > 1$ then                                    ▷ More than one candidate
            $C_1 \leftarrow \{v \in C_1 :\ \mathrm{neighbors}(v, \mathcal{A}) \geq\ \mathrm{neighbors}(w, \mathcal{A})\ \forall w \in C_1\}$
        end if
        if $C_1 = \emptyset$ then break
        end if
        Select one candidate $c \in C_1$
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{c\}$                                  ▷ Add candidate to aggregate
        $U \leftarrow U \setminus \{c\}$
    end while
end function
```

counts the number of neighbors of vertex v that are not yet aggregated neighbors of the aggregate $\mathcal{A}$. This criterion tries to maximize the number of candidates for choosing the next vertex. Finally, we ensure that the aggregate does not have a bigger diameter than the prescribed maximum value $d_{\max}$ after the new vertex is added.

Algorithm 3 Round Aggregate Step

```
function RoundAggregate($\mathcal{A}$, $\mathcal{V}$, $\mathcal{E}$, $s_{\max}$, $U$)        ▷ Rounds aggregate $\mathcal{A}$ while size $< s_{\max}$
    while $\#\mathcal{A} \leq s_{\max}$ do
        $C \leftarrow \{v \in N_\alpha :\ \mathrm{cons}(v, \mathcal{A}) >\ \mathrm{cons}(v, U)\}$
        Select arbitrary candidate $c \in C$
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{c\}$                                  ▷ Add candidate to aggregate
        $U \leftarrow U \setminus \{c\}$
    end while
end function
```

In a second step we aim to make the aggregates “rounder”. This is sketched in Algorithm 3. We add all non-aggregated adjacent vertices that have more connections to the current aggregate than to other non-aggregated vertices or to other aggregates until we reach the maximum allowed size $s_{\max}$ of our aggregate.

If after these two steps an aggregate still consists of only one vertex, we try to find another aggregate that the vertex is strongly connected to. If such an aggregate exists, we add the vertex to that aggregate and choose a new seed vertex.

Finally, once all the non-isolated vertices are aggregated, we try to build aggregates for the isolated vertices. Where possible, we build these by aggregating adjacent isolated vertices that have at least one common neighboring aggregate consisting of non-isolated vertices. This is done in the function GrowIsoAggregate which we do not present here.

3.2. Schwarz–Type Smoothers for the DG System

The linear system (6) is coercive for both the NIPG and the SIPG method provided the penalty parameter γ_F is sufficiently large. This depends on the choice of the parameter α in (5). If this is the case, block versions of traditional relaxation methods, such as Jacobi, Gauss-Seidel or SOR, can be used as smoothers on the DG space V_h^k. In the context of subspace correction (or Schwarz) methods described above, this corresponds to choosing a set of non-overlapping subdomains $\{\Omega_i\}$. The simplest choice that ensures that the grid resolves the partition is $\Omega_i := T_i \in \mathcal{T}_h$, $i = 1, \ldots, m$, i.e. each subdomain consists of a single fine grid element and the associated degrees of freedom. In this case, the subspace V_i, for each $i = 1, \ldots, m$, is then simply chosen to be $\operatorname{span}\{\phi_k \mid k \in I_{T_i}\}$. For both NIPG and SIPG with sufficiently large γ_F we settle for these simple subspaces and use exact subspace solvers. This is equivalent to using block Gauss-Seidel with the algebraic blocks described in (8).

Unfortunately, non-overlapping subspace correction methods loose their smoothing properties for low penalty parameters γ_F. This behavior was already observed in [13], where geometric multigrid methods are applied to NIPG discretizations. However, in the same paper it is shown numerically that the corresponding subspace correction methods using overlapping subspaces are robust for low and high penalty parameters. Therefore, we use subspace correction methods with overlapping subspaces for non-symmetric interior penalty methods with small or zero $gamma_F$.

To find good (non-overlapping or overlapping) partitionings $\{\Omega_i\}$ of Ω, we neglect geometric properties of the grid and use only algebraic information encoded in the stiffness matrix A by resorting to the aggregation algorithm described in the previous section. As input for our algorithm, we use the connectivity graph of the block matrix (8), i.e. $V = \{1, \ldots m\}$ and $E = \{(i,j) \mid A_{T_iT_j} \neq 0\}$. The weight functions are defined as $w_V(i) = \|A_{T_iT_i}\|_\infty$ and $w_E((i,j) = \|A_{T_iT_j}\|_\infty$. Therefore, each of the resulting aggregates $\mathcal{A}_i$, $i \in I$, can be associated with a set of mesh elements $\{T_k : k \in \mathcal{A}_i\}$. For our non-overlapping smoothers we now simply choose

$$\Omega_i := \operatorname{interior}\left(\bigcup_{k \in \mathcal{A}_i} T_k \right).$$

To get an overlapping partitioning we start with the same aggregates $\{\mathcal{A}_i : i \in I\}$ and augment each aggregate $\mathcal{A}_i$ by the set of all elements that share a face with any of the elements in $\mathcal{A}_i$, i.e. we choose

$$\Omega_i := \operatorname{interior}\left(\bigcup_{k \in \widetilde{\mathcal{A}}_i} T_k \right), \quad \text{where} \quad \widetilde{\mathcal{A}}_i := \mathcal{A}_i \cup \left\{ l : T_l \cap \left(\bigcup_{k \in \mathcal{A}_i} T_k \right) \neq \emptyset \right\}.$$

In Figure 1 a typical overlapping subdomain Ω_i is shown together with the adjacency graph of the system matrix. The aggregate $\mathcal{A}_i$ consists only of one mesh element in this case.

In both cases (overlapping and non-overlapping) we only use very small subdomain sizes. In the non-overlapping case these correspond to one element of the grid and are represented by the blocking used in (8). We use Gaussian elimination as a solver for these subproblems. For the overlapping case the subspaces are larger and correspond to an aggregate of around 37 to 60 mesh elements. It turns out that inexact subspace solvers suffice, leading to substantial savings

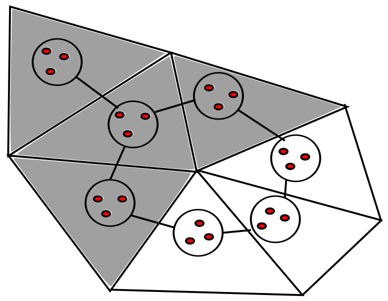
Figure 1. A typical subdomain (shaded) for the smoother.

in computing time in comparison to exact solvers. In the numerical experiments in Section 4 we use a block-ILU(0) preconditioner as the inexact subspace solver, and we always use multiplicative subspace correction. Note that the linear system of each overlapping subspace is a submatrix of (8) and inherits the block form used by the preconditioner from the complete system.

3.3. Coarse Space Correction in the Conforming Subspace

Recall that $\mathcal{T}_h$ is assumed to be a conforming, simplicial mesh and denote by

$$W_h^1 = \{w \in C^0(\Omega) : w|_T \in \mathbb{P}_1 \ \forall T \in \mathcal{T}_h\} \tag{11}$$

the space of conforming linear finite element functions. Furthermore we denote by

$$W_{h,0}^1 = \{w \in W_h^1 : w|_{\Gamma_D} = 0\} \tag{12}$$

the space of conforming linear finite element functions vanishing on the boundary. Other meshes, such as quadrilateral or hexahedral ones, can be handled in an analogous way. Note that $W_h^1 \subset V_h^k$ is a true subspace.

In our subspace correction method we choose now $V_0 = W_h^1$ for the case of SIPG and NIPG with sufficiently large penalty parameter γ_F and $V_0 = W_{h,0}^1$ for the case of NIPG with small γ_F and OBB, i.e. given a current iterate $u^{(s)} \in V_h^k$ we seek a correction $w_0 \in V_0$ such that

$$a_h\Big(u^{(s)} + w_0, v\Big) = l_h(v) \qquad \forall v \in V_0. \tag{13}$$

Due to linearity this is equivalent to

$$b_h(w_0, v) = l_h(v) - a_h(u^{(s)}, v) \qquad \forall v \in V_0,$$

with the reduced bilinear form

$$b_h(u,v) = (K\nabla u, \nabla v)_{0,\Omega} + \sum_{F \in \mathcal{F}_h^{\Gamma_D}} \Big[\theta\big(u, n_F^{\mathrm{T}} K \nabla v\big)_{0,F} - \big(n_F^{\mathrm{T}} K \nabla u, v\big)_{0,F} + \gamma_F (u,v)_{0,F}\Big], \tag{14}$$

for any $u, v \in V_0$. This is due to the fact that all the jump terms in $a_h(\cdot,\cdot)$ are 0 in $V_0 \times V_0$. Note that $b_h(\cdot,\cdot)$ is almost identical to the standard bilinear form for the conforming finite element method.

The only difference is the treatment of Dirichlet boundary conditions, which are enforced in a weak sense in $b_h(\cdot,\cdot)$. Therefore we can employ existing, tested inexact solvers to solve (14).

Let $\Psi_h = \{\psi_1, \ldots, \psi_{n_0}\}$ be the standard Lagrange basis for W_h^1. Due to the nestedness of the spaces W_h^1 and V_h^k any basis function $\psi_i \in \Psi_h$ can be represented in the basis Φ_h, i.e.

$$\psi_i = \sum_{j=1}^{n} (\mathbf{R}_0)_{i,j}\, \phi_j\,, \tag{15}$$

providing a natural restriction operator $\mathbf{R}_0$ from V_h^k to W_h^1. The sparse matrix $\mathbf{R}_0$ has to be provided as an additional input by the user and renders the method not fully algebraic. If $\mathbf{u}^{(s)} \in \mathbb{R}^n$ denotes the coefficient representation of $u^{(s)} \in V_h^k$, then the coarse space correction step (13) is equivalent to solving the linear system

$$\mathbf{A}_0\mathbf{w}_0 = \mathbf{R}_0(\mathbf{f} - \mathbf{A}\mathbf{u}^{(s)}), \tag{16}$$

where $\mathbf{A}_0 = \mathbf{R}_0\mathbf{A}\mathbf{R}_0^T$. The solution vector $\mathbf{w}_0 \in \mathbb{R}^{n_0}$, is the coefficient representation of the correction $w_0 \in W_h^1$. Note that $\mathbf{A}_0$ coincides with the matrix obtained from a standard finite element discretization with the Dirichlet boundary conditions enforced in a weak sense.

The linear system of the conforming subspace is solved inexactly by applying one V-cycle of algebraic multigrid based on aggregation similar to the original method [31] (see [22] for details). The aggregation is performed by Algorithm 1 on the matrix graph of $\mathbf{A}_0$. For the classification of the vertices and edges we use the weight functions $w_{\mathcal{V}}(i) = a_{ii}$ and $w_{\mathcal{E}}(i,j) = \max\{0, -a_{ji}\}$, respectively. Given the aggregates $\{\mathcal{A}_i\}_{i\in I_0}$, we define the piecewise constant prolongation operator as

$$(\mathbf{R}_1^T)_{ij} = \begin{cases} 1 & \text{if } j \in \mathcal{A}_i \\ 0 & \text{otherwise} \end{cases}\,.$$

The matrix on the next level is constructed by the standard Galerkin product: $\mathbf{A}_1 = \mathbf{R}_1\mathbf{A}_0\mathbf{R}_1^T$. The matrices A_l, $l > 1$, on the coarser levels are then constructed in the same way, using Algorithm 1 on the matrix graph of $\mathbf{A}_{l-1}$. As a smoother we use simple point Gauss-Seidel on each level. The system on the coarsest level is solved with a direct solver.

4. NUMERICAL RESULTS

In this section we test our new algebraic multigrid method on various two- and three-dimensional model problems and for various DG approximations. In the tables the acronyms NIPG(k,α), SIPG(k,α) and OBB(k) mean that NIPG, SIPG, and OBB (all using weighted averages to account for discontinuities in K) is used with parameter α and order k for the discretization, respectively. The problems are discretized on a structured cube grid with uniform grid width h. The multigrid method is used as the preconditioner in the conjugate gradient (CG) and BiCGSTAB method for symmetric (SIPG) and unsymmetric linear systems (NIPG and OBB), respectively. We measure the number of iterations (labelled It.) to achieve a relative residual reduction of 10^{-8}, the time needed per iteration (labelled TIt), the time for building the AMG hierarchy (labelled TB), and the time needed for solving the linear system (labelled TS). $V(s_{\text{DG}})$ means that one V-cycle of our AMG method was used as a preconditioner with smoother s_{DG} on the DG level. On all the other levels point-wise Gauss-Seidel is used as the smoother. The smoother s_{DG} is either multiplicative Schwarz with non-overlapping subdomains consisting of single mesh elements (labelled GS) or with overlapping subdomains (labelled OGS). The parameters for the aggregation algorithm in the overlapping case are set to the same values for 2D and 3D. The minimal and maximal prescribed aggregate sizes are $s_{\text{min}} = 25$ and $s_{\text{max}} = 37$. The maximal diameter is $d_{\text{max}} = 15$. This results in overlapping subspaces with, on average, 42 and 73 cells in 2D and 3D, respectively. The block-ILU(0) factorization for each subspace is computed on the fly in each step to save memory. (Note that in the non-overlapping case, since each subdomain consists only of a single element, the block-ILU(0) factorization is in fact an exact factorization.) We always perform one pre- and one post-smoothing step on all levels (in the DG as well as the conforming spaces).

$\gamma_F \frac{\lvert F\rvert}{\min(T^-(F),T^+(F))}$	10^3	10^2	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	0
GS Iter.	33	10	6	19.5	∞	∞	∞	∞	∞
GS CPU Time	5.48	2.00	1.02	3.25	∞	∞	∞	∞	∞
OGS Iter.	6	6	6	9	6	6	6	6	6
OGS CPU Time	3.64	3.65	3.34	5.18	3.46	3.70	3.39	3.42	3.76

Table I. Robustness of smoothers for 2D Poisson problem, $1/h = 128$ elements, NIPG(2, μ)

$\gamma_F \frac{\lvert F\rvert}{\min(T^-(F),T^+(F))}$	10^3	10^2	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	0
GS Iter.	63	30	17	7	8	∞	∞	∞	∞
GS CPU time	9.67	4.50	2.59	1.05	1.31	∞	∞	∞	∞
OGS Iter.	53	22	9	4	4	4	4	4	6
OGS CPU time	38.07	15.87	6.13	3.28	2.56	2.93	2.92	2.54	3.38

Table II. Robustness of smoothers for 3D Poisson problem, $1/h = 16$, NIPG(2, μ)

We start the analysis of our method by solving the Poisson equation, i.e. $K \equiv I$. The overlapping smoothers need far more computing time than the non-overlapping versions. Still, we believe that their usage is already justified by this very simple test case, when the penalty parameter γ_F is not sufficiently big. Therefore we test the behavior of the method for NIPG with varying penalty parameter γ_F. Recall that the limit case $\gamma_F = 0$ for NIPG represents the method of Baumann and Oden (OBB). For sufficiently large $gamma_F$ we use the space W_h^1 to compute the coarse space correction. We switch to $W_{h,0}^1$ as the coarse space for $gamma_F < 0.1$ in the two-dimensional and $gamma_F < 10^{-4}$ in the three-dimensional case. The number of iterations needed to achieve convergence are given in Tables I and II for the two dimensional and three dimensional case, respectively. Where ∞ appears instead of a number, we could not achieve any convergence with the method.

We can see clearly that it is mandatory to use the overlapping Schwarz smoother for vanishing penalty parameter γ_F. We regard this feature as very important as we consider the method of Baumann and Oden a good choice from an approximation point of view for problems with high contrast jumps in the coefficients.

In Table III, we study the robustness with respect to polynomial order k for all three discretization methods. The number of degrees of freedom per mesh is presented in the third column (Dof/E). The solver is very robust for all methods and for all orders that we tested ($k \leq 6$). For both NIPG with non-overlapping Schwarz smoother and for OBB with overlapping Schwarz smoother the behavior

k	Dof	Dof/E	TB	TS	It.
2	98304	6	1.19	3.70	6
3	163840	10	1.27	7.54	5
4	245760	15	1.36	17.84	5
5	344064	21	1.42	58.06	6
6	458752	28	1.79	121.8	6

(a) OBB(p), V(OGS)

k	Dof	Dof/E	TB	TS	It.
2	98304	6	1.20	1.04	6
3	163840	10	1.29	3.27	7
4	245760	15	1.25	7.25	7
5	344064	21	1.45	15.80	6
6	458752	28	1.76	36.37	7

(b) NIPG(p,0.65), V(GS)

k	Dof	Dof/E	TB	TS	It.
2	98304	6	1.17	1.14	13
3	163840	10	1.19	3.82	13
4	245760	15	1.32	8.25	15
5	344064	21	1.43	20.46	16
6	458752	28	1.70	54.04	19

(c) SIPG(p,2.2), V(GS)

k	Dof	Dof/E	TB	TS	It.
2	98304	6	1.19	3.67	12
3	163840	10	1.25	9.10	12
4	245760	15	1.3	21.31	12
5	344064	21	1.68	57.32	11
6	458752	28	1.77	123.86	11

(d) SIPG(p,2.2), V(OGS)

Table III. Robustness with respect to polynomial order k for 2D Poisson, $1/h = 128$

of the solver, in terms of number of iterations, is optimal for polynomial orders $k \leq 6$. Only for SIPG the number of iterations increases slightly with the polynomial order if we use the non-overlapping smoother. If we use the overlapping smoother instead our solver is also fully robust for SIPG. However, at least for smaller polynomial degree this robustness comes at the price of a higher computational cost due to the higher cost of the overlapping smoother.

In Tables IV and V we study the robustness with respect to mesh size h. The results show a

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.072	0.041	4	0.164
64	24576	3	0.316	0.153	4	0.612
128	98304	4	1.108	0.616	6	3.696
256	393216	5	4.988	2.361	7	16.53
512	1572864	6	20.14	10.07	8	80.59
1024	6291456	7	82.41	40.95	10	409.5

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.072	0.0115	8	0.09201
64	24576	3	0.276	0.02178	9	0.196
128	98304	4	1.232	0.09034	12	1.084
256	393216	5	4.724	0.3848	15	5.772
512	1572864	6	19.01	1.559	16	24.95
1024	6291456	7	77.93	6.107	18	109.9

(b) SIPG(2, 1.66), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.072	0.0184	5	0.09201
64	24576	3	0.296	0.0432	5	0.216
128	98304	4	1.184	0.1693	6	1.016
256	393216	5	4.728	0.708	7	4.956
512	1572864	6	19.35	2.725	9	24.53
1024	6291456	7	84.55	11.62	10	116.2

(c) NIPG(2, 0.65), V(GS)

Table IV. Robustness with respect to mesh width h for the 2D Poisson problem

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.144	0.09867	3	0.296
16	40960	3	1.292	0.629	4	2.516
32	327680	4	10.72	5.487	5	27.43
64	2621440	5	87.78	49.59	6	297.6

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.128	0.01511	18	0.272
16	40960	3	1.232	0.07275	27	1.964
32	327680	4	10.64	0.548	38	20.83
64	2621440	5	84.26	4.026	52	209.4

(b) SIPG(2, 1.25), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.168	0.029	8	0.232
16	40960	3	1.296	0.133	12	1.596
32	327680	4	10.32	1.055	14	14.78
64	2621440	5	85.66	8.217	23	189

(c) NIPG(2, 0.48), V(GS)

Table V. Robustness with respect to mesh width h for the 3D Poisson problem

slight increase in the number of iterations needed for convergence. The logarithmic growth with h is fully expected and due to the suboptimal convergence of the aggregation-type algebraic multigrid method with piecewise constant prolongation which is used as the inexact solver on the conforming subspace (see [22, 32]). The slightly better convergence behavior of the solver for OBB in 3D is due to the overlapping Schwarz smoother, since the subspaces are larger and have more overlap than in 2D.

The rest of our examples are model problems with jumping coefficients. The first problem to be investigated is given in the the following example.

Example 4.1. Let our diffusion problem be given by

$$\begin{aligned} -\nabla \cdot \{k(x)\nabla u\} &= 1 && \text{in } \Omega = (0,1)^d,\, d = 2,\, 3, \\ u &= 0 && \text{on } \partial\Omega\,. \end{aligned}$$

The isotropic permeability $k(x) \in \mathbb{R}$ has jumps in a checkerboard manner. The checkerboard has 8^d cells of width $H = 1/8$ in each dimension. Let the function $\lfloor \cdot \rfloor$ return the maximum integer value that is equal to or smaller than the argument. Then the permeability field is described by

$$k(x) = \begin{cases} 20.0 & \lfloor x_0/H \rfloor \text{ even, } \lfloor x_1/H \rfloor \text{ even, and } \lfloor x_2/H \rfloor \text{ even} \\ 0.002 & \lfloor x_0/H \rfloor \text{ odd, } \lfloor x_1/H \rfloor \text{ even, and } \lfloor x_2/H \rfloor \text{ even} \\ 0.2 & \lfloor x_0/H \rfloor \text{ even, } \lfloor x_1/H \rfloor \text{ odd, and } \lfloor x_2/H \rfloor \text{ even} \\ 2000.0 & \lfloor x_0/H \rfloor \text{ odd, } \lfloor x_1/H \rfloor \text{ odd , and } \lfloor x_2/H \rfloor \text{ even} \\ 1000.0 & \lfloor x_0/H \rfloor \text{ even, } \lfloor x_1/H \rfloor \text{ even, and } \lfloor x_2/H \rfloor \text{ odd} \\ 0.001 & \lfloor x_0/H \rfloor \text{ odd, } \lfloor x_1/H \rfloor \text{ even, and } \lfloor x_2/H \rfloor \text{ odd} \\ 0.1 & \lfloor x_0/H \rfloor \text{ even, } \lfloor x_1/H \rfloor \text{ odd, and } \lfloor x_2/H \rfloor \text{ odd} \\ 10.0 & \lfloor x_0/H \rfloor \text{ odd, } \lfloor x_1/H \rfloor \text{ odd, and } \lfloor x_2/H \rfloor \text{ odd} \end{cases},$$

in three dimensions and by

$$k(x) = \begin{cases} 20.0 & \lfloor x_0/H \rfloor \text{ even, and } \lfloor x_1/H \rfloor \text{ even} \\ 0.002 & \lfloor x_0/H \rfloor \text{ odd, and } \lfloor x_1/H \rfloor \text{ even} \\ 0.2 & \lfloor x_0/H \rfloor \text{ even, and } \lfloor x_1/H \rfloor \text{ odd} \\ 2000.0 & \lfloor x_0/H \rfloor \text{ odd, and } \lfloor x_1/H \rfloor \text{ odd} \end{cases},$$

in two dimensions.

We present the results for this problem in two and three dimensions in Tables VI and VII. Clearly the problem is harder than the Poisson problem studied above and on average all the methods require slightly more iterations, but the scalability with respect to h is still very good if not better, staying virtually constant over a large range of values for h.

The final two examples that we investigate are problems with random coefficients.

Example 4.2. Here we study the following heterogeneous model problem with mixed boundary conditions

$$\begin{aligned} -\nabla \cdot \{k(x)\nabla u\} &= 0 && \text{in } \Omega = (0,1)^d, \\ u &= g && \text{on } \Gamma_D, \\ -\nabla u \cdot \nu &= 0 && \text{on } \Gamma_N, \end{aligned}$$

with

$$\Gamma_D = \{x : x_1 = 0 \text{ or } x_1 = 1\}, \quad \Gamma_N = \partial\Omega \setminus \Gamma_D,$$

and

$$g(x) = \begin{cases} 1 & x_1 = 0 \\ 0 & x_1 = 1 \end{cases}.$$

The scalar permeability field $k(x)$ is chosen as a realization of a log-normal random field. More specifically, $\log k(x)$ is chosen to be a realization of a homogeneous, isotropic Gaussian random field with exponential covariance function and mean 0, variance σ^2 and correlation length scale λ. See [33] for more details about this model problem.

Example 4.3. Here, the problem is described by the same equations as in Example 4.2. The only difference lies in the choice of permeability field. It is again based on the log-normal distribution

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.072	0.04267	9	0.384
64	24576	3	0.336	0.1447	12	1.736
128	98304	4	1.268	0.616	16	9.857
256	393216	5	5.084	2.449	15	36.73
512	1572864	6	20.19	10.12	14	141.7
1024	6291456	7	83.74	39.92	14	558.8

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.084	0.01055	11	0.116
64	24576	3	0.252	0.02225	32	0.712
128	98304	4	1.14	0.08971	40	3.588
256	393216	5	4.84	0.3855	36	13.88
512	1572864	6	19.84	1.566	30	46.97
1024	6291456	7	80.04	6.149	39	239.8

(b) SIPG(2, 1.66), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.064	0.01933	6	0.116
64	24576	3	0.272	0.04089	18	0.736
128	98304	4	1.224	0.1725	18	3.104
256	393216	5	4.8	0.7379	13	9.593
512	1572864	6	19.29	2.844	17	48.36
1024	6291456	7	80.06	11.2	18	201.6

(c) NIPG(2, 0.65), V(GS)

Table VI. Chequerboard Problem (Example 4.1) in 2D

1/h	DOF	levels	TB	TIt	It	TS
16	40960	3	1.216	0.7364	25	18.41
32	327680	4	10.07	5.967	27	161.1
64	2621440	5	84.55	48.17	21	1012

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
16	40960	3	1.236	0.07283	68	4.952
32	327680	4	10.1	0.5466	72	39.36
64	2621440	6	82.51	4.015	122	489.8

(b) SIPG(2, 0.48), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
16	40960	3	1.204	0.1344	38	5.108
32	327680	4	10.36	1.07	40	42.79
64	2621440	5	83.28	8.265	41	338.9

(c) NIPG(2, 1.25), V(GS)

Table VII. Checkerboard Problem (Example 4.1) in 3D

described above. However, here we create a binary medium by clustering the values into two sets. Given a realization of the log-normal distribution on $\Omega = (0,1)^d$, sampled at the centers of each mesh element, we compute the arithmetic mean of the permeability values over all of Ω. We cluster together the elements with values above this mean and those with values below. On each of the clusters we replace the actual value of the permeability with the arithmetic mean over the respective cluster. This results in a two-valued permeability field modelling a binary medium. We refer to it as a clipped log-normal field. In contrast to the previous example, the permeabilities now jump across complicated interfaces instead of just changing gradually. This is a very challenging model problem that has been used previously to study robustness of solvers for conforming discretization in [32].

For the log-normally distributed permeability field (Example 4.2), the results for varying problem size are given in Tables VIII and IX. For the clipped version described in Example 4.3, the numbers

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.08001	0.041	4	0.164
64	24576	3	0.308	0.1384	5	0.692
128	98304	4	1.248	0.624	7	4.368
256	393216	5	5.176	2.49	20	49.8
512	1572864	7	20.95	9.849	20	197
1024	6291456	8	86.1	41.25	28	1155

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.056	0.011	8	0.088
64	24576	3	0.272	0.02364	11	0.26
128	98304	4	1.164	0.09271	17	1.576
256	393216	5	4.784	0.3897	24	9.353
512	1572864	7	18.83	1.556	41	63.81
1024	6291456	8	77.54	6.163	51	314.3

(b) SIPG(2, 1.66), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.08001	0.0208	5	0.104
64	24576	3	0.26	0.04467	6	0.268
128	98304	4	1.22	0.1835	8	1.468
256	393216	5	4.892	0.7444	13	9.677
512	1572864	7	19.03	2.954	21	62.02
1024	6291456	8	78.74	11.73	21	246.4

(c) NIPG(2, 0.65), V(GS)

Table VIII. Log–normal Random Problem (Example 4.2) in 2D, $\sigma^2 = 8$, $\lambda = 4h$

can be found in Tables X and XI. In both cases, similarly to [32][Table 3], we left the variance fixed at $\sigma^2 = 8$ and the correlation length is scaled with the grid width such that $\lambda = 4h$. Therefore, the permeability fields become less smooth for larger problems. In 2D this is reflected in a more steeply ascending number of iterations needed for convergence when compared to the Poisson case. However, the number of iterations is reasonable in all cases for these hard problems and comparable to the behavior of aggregation–type AMG for conforming discretization (see [22, 32]). More surprising is the extremely robust behavior in 3D. The numbers of iterations in both examples are virtually identically to the corresponding numbers for the simple Poisson case. It confirms the excellent robustness of our approach. The reason for the better performance in 3D lies probably in the fact that the test problem is actually easier than in 2D for the same model parameters. The problem models percolation through the region Ω from left to right, and it is well known that binary media exhibit a different percolation threshold (with respect to the amount of available percolation paths) in three dimensions than in two dimensions.

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.12	0.09467	3	0.284
16	40960	3	1.212	0.7507	3	2.252
32	327680	4	10.24	6.349	4	25.4
64	2621440	7	87.93	50.77	9	456.9

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.132	0.01653	15	0.248
16	40960	3	1.204	0.07349	27	1.984
32	327680	4	10.4	0.5575	36	20.07
64	2621440	6	83.61	4.281	54	231.2

(b) SIPG(2, 1.25), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.124	0.032	7	0.224
16	40960	3	1.256	0.136	10	1.36
32	327680	4	10.2	0.9998	15	15
64	2621440	6	87.27	8.409	23	193.4

(c) NIPG(2, 0.48), V(GS)

Table IX. Log–normal Random Problem (Example 4.2) in 3D, $\sigma^2 = 8$, $\lambda = 4h$

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.092	0.047	4	0.188
64	24576	3	0.316	0.143	5	0.716
128	98304	4	1.268	0.6274	6	3.768
256	393216	5	4.696	2.589	7	18.13
512	1572864	6	19.66	9.904	12	118.8
1024	6291456	7	88.21	41.6	15	625.4

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.052	0.011	8	0.08801
64	24576	3	0.284	0.02255	11	0.248
128	98304	4	1.228	0.08901	20	1.78
256	393216	5	4.9	0.3816	35	13.36
512	1572864	6	19.79	1.53	47	71.92
1024	6291456	7	79.78	6.236	61	380.4

(b) SIPG(2, 1.66), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
32	6144	2	0.08801	0.02	5	0.1
64	24576	3	0.272	0.04334	6	0.26
128	98304	4	1.208	0.1686	13	2.192
256	393216	5	4.784	0.7174	18	12.91
512	1572864	6	19.46	2.951	23	67.86
1024	6291456	7	81.11	11.47	28	321.3

(c) NIPG(2, 0.65), V(GS)

Table X. Clipped Log–normal Random Problem (Example 4.3) in 2D, $\sigma^2 = 8$, $\lambda = 4h$

We examine the robustness with respect to changes in the variance σ^2 for the 2D clipped log-normal random problem (Example 4.3) in Table XII(a). We use OBB for the discretization on a uniform grid with $h = 1/128$ and the overlapping Schwarz method as the smoother on the DG space.

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.124	0.09601	3	0.288
16	40960	3	1.192	0.635	4	2.54
32	327680	4	10.7	6.648	6	39.89
64	2621440	5	88.42	49.8	7	348.6

(a) OBB(2), V(OGS)

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.112	0.01511	18	0.272
16	40960	3	1.248	0.07245	27	1.956
32	327680	4	10.01	0.5268	38	20.02
64	2621440	6	82.24	4.134	55	227.4

(b) SIPG(2, 1.25), V(GS)

1/h	DOF	levels	TB	TIt	It	TS
8	5120	2	0.116	0.031	8	0.248
16	40960	3	1.192	0.1353	12	1.624
32	327680	4	10.09	1.033	15	15.5
64	2621440	5	80.81	8.095	19	153.8

(c) NIPG(2, 0.48), V(GS)

Table XI. Clipped Log–normal Random Problem (Example 4.3) in 3D, $\sigma^2 = 8$, $\lambda = 4h$

σ^2	1	2	4	8	16
It.	7	8	9	19	53
$\max_{\tau\nu} \frac{k_\tau}{k_\nu}$	5.5	12.2	44.8	366	$1.2 * 10^5$

(a) Fixed correlation length $\lambda = 1/64$, varying σ^2

λ	1/2	1/4	1/8	1/16	1/32	1/64	1/128
It.	6	7	7	8	9	9	12

(b) Fixed variance $\sigma^2 = 4$, varying λ

Table XII. Clipped Log–normal Random Problem (Example 4.3) in 2D

, $h = 1/128$

The first row contains the variance used for the problem, the second row the number of iterations needed for convergence, and the last row the ratio between the highest and lowest permeability values in the chosen realization. In all runs the correlation length is $\lambda = 1/64$. The number of iterations seems to increase only roughly linearly with σ, even though the jump size increases exponentially.

Finally, in Table XII(b) we keep the variance fixed at $\sigma^2 = 4$ and compare the number of iterations needed for convergence with varying correlation length λ for Example 4.3 in 2D. The number of iterations needed by our OBB solver only increases slightly for shorter correlation lengths. Even when the correlation length is equal to the mesh width, i.e. $\lambda = h$, we get a reasonable convergence rate.

5. RELATED WORK AND DISCUSSION

To the best of our knowledge the first publication on AMG as a solver for discontinuous Galerkin discretizations is [15]. It contains a case study of applying (smoothed) element agglomeration AMG to linear systems from piecewise linear SIPG discretizations. The coarse level matrices are created using a Galerkin product $\mathbf{P}^T\mathbf{A}\mathbf{P}$. The prolongation operator $\mathbf{P}$ from the first coarse grid is defined by the natural embedding of the corresponding space of piecewise constant trial functions into that of piecewise linear trial functions both defined on the finest grid. A recursive graph bisection algorithm is used on the adjacency graph of the grid elements to form the remaining coarser meshes. The elements of the coarse meshes are agglomerations of the elements of the fine grid. The prolongation matrices from these meshes represent the natural embeddings of the piece-wise constant trial spaces on the coarse mesh into those on the next fine mesh. No strength of connection criterion is used and therefore this approach cannot be applied to problems with jumping diffusion coefficients. Furthermore the complexity of the hierarchy building is $O(N \log(N))$, where N is the number of unknowns.

More recently, the application of smoothed aggregation algebraic multigrid was investigated by Prill et al. [19]. The authors solve linear systems from NIPG and SIPG discretizations with piecewise constant and bi-linear quadrilateral elements. Higher order elements are discarded with the note that p-multigrid could be used to reduce the polynomial order. No numerical tests for jumping diffusion coefficients were performed. Compared to non-smoothed aggregation multigrid the presented approach produces considerable fill-in on the coarser levels. This effect would even be more amplified for higher order discretizations.

In contrast to the above mentioned related approaches, we have shown here that our preconditioners can be used also for higher order discretizations and for the method of Baumann and Oden. Additionally, we have demonstrated their robustness for problems with highly heterogeneous diffusion coefficients.

In [22] a parallelization approach for multigrid methods is described. This parallelization is already available for the algebraic multigrid method used as an inexact solver for the conforming space. It was shown in the above publication that it scales very well up to 4096 cores for the model problems investigated here. Meanwhile good scalability was achieved for nearly 300 thousand cores, which will be published in a forthcoming paper. The described parallelization approach can easily be applied to the Schwarz smoother used on the discontinuous approximation space and should lead to a very scalable algebraic multigrid solver for discontinuous Galerkin discretizations.

ACKNOWLEDGEMENTS

We would like to thank Christian Engwer for supplying us with a discontinous Galerkin discretization code (not using weighted averages) and the restriction operator to the conforming space. Furthermore we are grateful to the developers of the “Distributed and Unified Numerics Environment” (DUNE), see [34, 35, 36], for providing their PDE and iterative solver code as open source to the public at http://www.dune-project.org. Without them this work would not have been possible.

REFERENCES

1. Wheeler MF. An elliptic collocation-finite element method with interior penalties. *SIAM J. Numer. Anal.* 1978; **15**(1):152–161, doi:10.1137/0715010.
2. Arnold DN. An interior penalty finite element method with discontinuous elements. *SIAM J. Numer. Anal.* 1982; **19**(4):742–760, doi:10.1137/0719052.

3. Oden JT, Babuška I, Baumann CE. A discontinuous hp finite element method for diffusion problems. *J. Comput. Phys.* 1998; **148**(2):491–519, doi:10.1006/jcph.1998.6032.
4. Baumann CE, Oden JT. A discontinuous hp finite element method for convection-diffusion problems. *Comput. Methods Appl. Mech. Engrg.* 1999; **175**(3–4):311–341, doi:10.1016/S0045-7825(98)00359-4.
5. Rivière BM, Wheeler MF, Girault V. Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems I. *Comput. Geosci.* 1999; **3**(3–4):337–360, doi:10.1023/A:1011591328604.
6. Arnold DN, Brezzi F, Cockburn B, Marini LD. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.* 2002; **39**(5), doi:10.1137/S0036142901384162.
7. Bastian P, Reichenberger V. Multigrid for higher order discontinuous Galerkin finite elements applied to groundwater flow. *Technical Report 2000-37*, SFB 359 2000.
8. Feng X, Karakashian OA. Two-level additive schwarz methods for a discontinuous Galerkin approximation of second order elliptic problems. *SIAM J. Numer. Anal.* 2002; **39**(4):1343–1365, doi:10.1137/S0036142900378480.
9. Gopalakrishnan J, Kanschat G. A multilevel discontinuous Galerkin method. *Numer. Math.* 2003; **95**:527–550.
10. Hemker PW, Hoffmann W, van Raalte MH. Two-level Fourier analysis of a multigrid approach for discontinuous Galerkin discretizations. *SIAM J. Sci. Comput.* 2003; **25**(3):1018–1041, doi:10.1137/S1064827502405100.
11. Hemker PW, Hoffmann W, van Raalte MH. Fourier two-level analysis for higher dimensional discontinuous Galerkin discretisation. *Comput. Vis. Sci.* 2004; **7**:159–172, doi:10.1007/s00791-004-0136-1.
12. Brenner SC, Zhao J. Convergence of multigrid algorithms for interior penalty methods. *Appl. Numer. Anal. Comput. Math.* 2005; **2**(1):3–18, doi:10.1002/anac.200410019.
13. Johannsen K. Multigrid methods for nonsymmetric interior penalty discontinuous Galerkin methods. *ICES Report 05-23*, University of Texas at Austin 2005.
14. Dobrev VA, Lazarov RD, Vassilevski PS, Zikatanov LT. Two-level preconditioning of discontinuous Galerkin approximations of second-order elliptic equations. *Numer. Linear Algebra Appl.* 2006; **13**(9):753–770, doi: 10.1002/nla.504.
15. Dobrev VA. Preconditioning of discontinuous Galerkin methods for second order elliptic problems. PhD Thesis, Texas A & M University 2007.
16. Antonietti PF. Domain decomposition, spectral correctness and numerical testing of discontinuous Galerkin methods. PhD Thesis, University of Pavia, Italy 2007.
17. Antonietti PF, Ayuso B. Multiplicative schwarz methods for discontinous Galerkin approximations of elliptic problems. *ESAIM Math. Model. Numer. Anal.* 2008; **42**:443–469, doi:10.1051/m2an:2008012.
18. Brix K, Pinto MC, Dahmen W, Massjung R. Multilevel preconditioners for the interior penalty discontinuous Galerkin method II quantitative studies. *Communications in Computational Physics* 2009; **5**(2–4):296–325.
19. Prill F, Lukáčová-Medvid'ová M, Hartmann R. Smoothed aggregation multigrid for the discontinuous Galerkin method. *SIAM J. Sci. Comput.* 2009; **31**:3503–3528, doi:10.1137/080728457.
20. Ayuso de Dios B, Zikatanov L. Uniformly convergent iterative methods for discontinuous Galerkin discretizations. *J. Sci. Comput.* 2009; **40**:4–36, doi:10.1007/s10915-009-9293-1.
21. Ayuso De Dios B, Holst M, Zhu Y, Zikatanov L. Multilevel preconditioners for discontinuous Galerkin approximations of elliptic problems with jump coefficients. *Technical Report*, arXiv:1012.1287 2011.
22. Blatt M. A parallel algebraic multigrid method for elliptic problems with highly discontinuous coefficients. PhD Thesis, Ruprecht–Karls–Universität Heidelberg 2010.
23. Xu J. Iterative Methods by Space Decomposition and Subspace Correction. *SIAM Review* 1992; **34**(4):581–613, doi:10.1137/1034116.
24. Xu J. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing* 1996; **56**(3):215–235, doi:10.1007/BF02238513.
25. Rivière B. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation*, *Frontiers in Mathematics*, vol. 35. SIAM, 2008.
26. Ern A, F A, Stephansen, Zunino P. A discontinuous Galerkin method with weighted averages for advection-diffussion equations with locally small and anisotropic diffusivity. *IMA J. Numer. Anal.* 2009; **29**:235–256, doi: 10.1093/imanum/drm050.
27. Epshteyn Y, Rivière B. Estimation of penalty parameters for symmetric interior penalty Galerkin methods. *Journal of Computational and Applied Mathematics* 2007; **206**:843–872, doi:10.1016/j.cam.2006.08.029.
28. Houston P, Hartmann R. An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier-Stokes equations. *J. Comput. Phys.* 2008; **227**:9670–9685, doi:10.1016/j.jcp.2008.07.015.
29. Bastian P. Higher order discontinuous Galerkin methods for flow and transport in porous media. *Challenges in Scientific Computing – CISC 2002*, Bänsch E (ed.), no. 35 in LNCSE, 2003; 1–22.

30. Rivière BM, Wheeler MF, Giraul V. A prior error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems. *SIAM J. Numer. Anal.* 2001; **39**(3):902–931, doi: 10.1137/S003614290037174X.
31. Raw M. A coupled algebraic multigrid method for the 3d Navier-Stokes equations. *Fast Solvers for Flow Problems, Proceedings of the 10th GAMM-Seminar*, *Notes on Numerical Fluid Mechanics*, vol. 49. Vieweg-Verlag: Braunschweig, Wiesbaden, 1985.
32. Scheichl R, Vainikko E. Additive Schwarz with aggregation-based coarsening for elliptic problems with highly variable coefficients. *Computing* 2007; **80**(4):319–343, doi:10.1007/s00607-007-0237-z.
33. Cliffe KA, Graham IG, Scheichl R, Stals L. Parallel computation of flow in heterogeneous media modelled by mixed finite elements. *J. Comput. Phys.* 2000; **164**:258–282, doi:10.1006/jcph.2000.6593.
34. Bastian P, Blatt M, Dedner A, Engwer C, Klöfkorn R, Ohlberger M, Sander O. A generic grid interface for parallel and adaptive scientific computing. part I: abstract framework. *Computing* 2008; **82**(2–3):103–119, doi: 10.1007/s00607-008-0003-x.
35. Bastian P, Blatt M, Dedner A, Engwer C, Klöfkorn R, Kornhuber R, Ohlberger M, Sander O. A generic grid interface for parallel and adaptive scientific computing. part II: implementation and test in DUNE. *Computing* 2008; **82**(2–3):121–138, doi:10.1007/s00607-008-0004-9.
36. Blatt M, Bastian P. The iterative solver template library. *Applied Parallel Computing. State of the Art in Scientific Computing*, *Lecture Notes in Computer Science*, vol. 4699, Kågström B, Elmroth E, Dongarra J, Waśniewski J (eds.), Springer, 2007; 666–675, doi:10.1007/978-3-540-75755-9_82.