Comput Geosci (2014) 18:343–356
DOI 10.1007/s10596-014-9400-8

ORIGINAL PAPER

Fast linear solver for diffusion problems with applications to pressure computation in layered domains

P. van Slingerland · C. Vuik

Received: 27 January 2013 / Accepted: 7 January 2014 / Published online: 1 April 2014

Abstract Accurate simulation of fluid pressures in layered reservoirs with strong permeability contrasts is a challenging problem. For this purpose, the Discontinuous Galerkin (DG) method has become increasingly popular. Unfortunately, standard linear solvers are usually too inefficient for the aforementioned application. To increase the efficiency of the conjugate gradient (CG) method for linear systems resulting from symmetric interior penalty (discontinuous) Galerkin (SIPG) discretizations, we cast an existing two-level preconditioner into the deflation framework. The main idea is to use coarse corrections based on the DG solution with polynomial degree $p = 0$. This paper provides a numerical comparison of the performance of the original preconditioner and the resulting deflation variant in terms of scalability and overall efficiency. Furthermore, it studies the influence of the SIPG penalty parameter, weighted averages in the SIPG formulation (SWIP), the smoother, damping of the smoother, and the strategy for solving the coarse systems. We have found that the penalty parameter can best be chosen diffusion-dependent. In that case, both two-level methods yield fast and scalable convergence. Whether preconditioning or deflation is to be favored depends on the choice of the smoother and on the damping of the smoother. Altogether, both two-level methods can contribute to cheaper and more accurate fluid pressure simulations.

P. van Slingerland (✉) · C. Vuik
Delft University of Technology,
Mekelweg 4, 2628 CD Delft,
The Netherlands
e-mail: pvanslingerland@gmail.com

Keywords Deflation · Preconditioning · Two-level · Conjugate gradient · Discontinuous Galerkin

Mathematics Subject Classifications (2010) 65F10 · 65N30 · 65N55

1 Introduction

Layered reservoirs often exhibit very strong permeability contrasts with typical values between 10^{-1} and 10^{-7}. Solving for the pressure in such a system can be numerically challenging. The governing equation is a mildly nonlinear diffusion equation with time-varying coefficients obtained by combining mass conservation and Darcy's law. To study the linear systems resulting from discretizing this equation, this paper considers the stationary linearized equation (although the ideas presented in this paper can be extended to the original problem).

To discretize this equation in space, the Discontinuous Galerkin (DG) method can be particularly suitable [2, 26, 27, 32]. This discretization scheme can be interpreted as a finite volume method that uses piecewise polynomials of degree p rather than piecewise constant functions. As such, it combines the best of both classical finite element methods and finite volume methods, making it particularly suitable for handling non-matching grids and designing hp-refinement strategies.

However, the resulting linear systems are usually larger than those for the aforementioned classical discretization schemes. This is due to the larger number of unknowns per mesh element. At the same time, the condition number typically increases with the number of mesh elements, the polynomial degree, and the stabilization factor [7, 29]. The strong permeability contrasts in the problem sketched above pose an extra challenge. Altogether, standard linear solvers often result in long computational times and/or low accuracy of the approximated fluid pressures.

In search of efficient and scalable algorithms (for which the number of iterations does not increase with, e.g., the number of mesh elements), much attention has been paid to subspace correction methods [39]. Examples include classical geometric (h-)multigrid [5, 16], spectral (p-)multigrid [15, 23], algebraic multigrid [24, 28], and Schwarz domain decomposition [1, 14]. Another interesting strategy can be found in [4], where the authors present a multilevel solver based on a splitting of the DG space into two components that are orthogonal in the energy product.

Usually, these methods can either be used as a standalone solver, or as a preconditioner in an iterative Krylov method. The latter tends to be more robust for problems with a few isolated "bad" eigenvalues, as is the case for the strongly varying problems of our interest.

An alternative for preconditioning is the method of deflation, originally proposed in [21]. This method has been proved effective for layered problems with extreme contrasts in the coefficients in [37]. Deflation is related to multigrid in the sense that it also makes use of a coarse space that is combined with a smoothing operator at the fine level. This relation has been considered from an abstract point of view by Tang et al. [33, 34].

This research seeks to extend this comparison between preconditioning and deflation in the context of DG schemes. In particular, it is focused on the conjugate gradient (CG) method for linear systems resulting from symmetric interior penalty (discontinuous) Galerkin (SIPG) discretizations for stationary diffusion problems with extreme contrasts in the coefficients.

Starting point of this research is one of the two-level methods proposed by [8]. This method uses coarse corrections based on the DG discretization with polynomial degree $p = 0$. Using the analysis in [13], they have shown theoretically (for $p = 1$) that this preconditioner yields scalable convergence of the CG method, independent of the mesh element diameter. Another nice property is that the use of only two levels offers an appealing simplicity. More importantly, the coefficient matrix that is used for the coarse correction is quite similar to a matrix resulting from a central difference discretization, for which very efficient solution techniques are readily available.

To extend the work in [8], we cast the two-level preconditioner into the deflation framework, using the abstract analysis in [34]. Furthermore, we have conducted several numerical experiments to compare the scalability and the overall efficiency of both two-level methods. These results (including $p > 1$) complement the theoretical analysis for the preconditioning variant for $p = 1$ in [8]. Additionally, we have investigated how the efficiency of the CG method is influenced by the SIPG penalty parameter, the use of weighted versus standard averages, the smoother, damping of the smoother, and the strategy for solving the coarse systems.

The outline of this paper is as follows. Section 2 discusses the SIPG method for diffusion problems with large jumps in the coefficients. To solve the resulting systems, Section 3 discusses the two-level preconditioner. Section 4 rewrites the latter as a deflation method. Section 5 compares the performance of both two-level methods through various numerical experiments. Section 6 summarizes the main conclusions.

2 Discretization

We consider the linearized form of the spatially discretized pressure equation, which can be interpreted as a stationary diffusion equation. Section 2.1 discusses the SIPG method for this model following [26]. Section 2.2 describes the resulting linear systems. Section 2.3 motivates the use of a diffusion-dependent penalty parameter following [10].

2.1 SIPG method

We study the following model problem on the spatial domain Ω with boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ and outward normal $\mathbf{n}$:

$$\begin{aligned} -\nabla \cdot (K\nabla u) &= f, && \text{in } \Omega, \\ u &= g_D, && \text{on } \partial\Omega_D, \\ K\nabla u \cdot \mathbf{n} &= g_N, && \text{on } \partial\Omega_N. \end{aligned} \tag{1}$$

The diffusion (or permeability) coefficient K is a symmetric and positive-definite tensor that typically contains large jumps across the domain. The function f is a source term, and g_D and g_N specify Dirichlet and Neumann boundary conditions, respectively.

The SIPG approximation for the model above can be constructed in the following manner. First, choose a mesh with elements $E_1, ..., E_N$. The numerical experiments in this paper are for uniform Cartesian meshes on the domain $\Omega = [0, 1]^2$, although our solver can be applied for a wider range of problems. Next, define the test space V that contains each function that is a polynomial of degree p or lower within each mesh element, and that may be discontinuous at the element boundaries. The SIPG approximation u_h is now defined as the unique element in this test space that satisfies the relation

$$B(u_h, v) = L(v), \qquad \text{for all test functions } v \in V, \tag{2}$$

where B and L are (bi)linear forms that are specified hereafter.

To define these forms for mesh elements of size $h \times h$, we require the following additional notation: the vector $\mathbf{n}_i$ denotes the outward normal of mesh element E_i, the set Γ_h is the collection of all interior edges $e = \partial E_i \cap \partial E_j$, the set Γ_D is the collection of all Dirichlet boundary edges $e = \partial E_i \cap \partial \Omega_D$, and the set Γ_N is the collection of all Neumann boundary edges $e = \partial E_i \cap \partial \Omega_N$. Finally, we introduce the usual trace operators for jumps and averages at the mesh element boundaries: in the interior, we define at $\partial E_i \cap \partial E_j$: $[v] = v_i \mathbf{n}_i + v_j \mathbf{n}_j$ (for scalars v) and $\{\mathbf{v}\} = \frac{1}{2}(\mathbf{v}_i + \mathbf{v}_j)$ (for vectors $\mathbf{v}$), where v_i and $\mathbf{v}_i$ denote the trace of v and $\mathbf{v}$, respectively, along the side of E_i with outward normal $\mathbf{n}_i$. Similarly, at the domain boundary, we define at $\partial E_i \cap \partial \Omega$: $[v] = v_i \mathbf{n}_i$, and $\{\mathbf{v}\} = \mathbf{v}_i$. Using this notation, the forms B and L can be defined as follows:

$$
\begin{aligned}
L(v) &= \int_\Omega f v - \sum_{e \in \Gamma_D} \int_e \left([K \nabla v] + \frac{\sigma}{h} v \right) g_D \\
&\quad + \sum_{e \in \Gamma_N} \int_e v g_N, \\
B(u_h, v) &= \sum_{i=1}^{N} \int_{E_i} K \nabla u_h \cdot \nabla v \\
&\quad + \sum_{e \in \Gamma_h \cup \Gamma_D} \int_e \Big(\frac{\sigma}{h} [u_h] \cdot [v] \\
&\quad - \{K \nabla u_h\} \cdot [v] - [u_h] \cdot \{K \nabla v\} \Big),
\end{aligned}
$$

Where σ is the so-called penalty parameter. This positive parameter penalizes the inter-element jumps to enforce weak continuity and ensure convergence. Although it is presented as a constant here, its value may vary throughout the domain. This is discussed further in Section 2.3 later on.

For a large class of problems with a sufficiently smooth exact solution (as is the case for the numerical examples in this paper), the SIPG method yields convergence of order $p + 1$ [26].

2.2 Linear systems

In order to compute the SIPG approximation defined by (2), it needs to be rewritten as a linear system. To this end, we choose basis functions for the test space V. More specifically, for each mesh element E_i, we define the basis function $\phi_1^{(i)}$, which is zero in the entire domain, except in E_i, where it is equal to one. Similarly, we define higher-order basis functions $\phi_2^{(i)}, ..., \phi_m^{(i)}$, which are higher-order polynomials in E_i and zero elsewhere. In this paper, we use monomial basis functions.

These latter are defined as follows. In the mesh element E_i with center (x_i, y_i) and size $h \times h$, the function $\phi_k^{(i)}$ reads as follows:

$$
\phi_k^{(i)}(x, y) = \left(\frac{x - x_i}{\frac{1}{2}h} \right)^{k_x} \left(\frac{y - y_i}{\frac{1}{2}h} \right)^{k_y},
$$

where k_x and k_y are selected as follows:

k	1	2	3	4	5	6	7	8	9	10	...
k_x	0	1	0	2	1	0	3	2	1	0	...
k_y	0	0	1	0	1	2	0	1	2	3	...
	$p = 0$	$p = 1$		$p = 2$			$p = 3$				...

The dimension of the basis within one mesh element is equal to $m = \frac{(p+1)(p+2)}{2}$.

Next, we express u_h as a linear combination of the basis functions as follows:

$$
u_h = \sum_{i=1}^{N} \sum_{k=1}^{m} u_k^{(i)} \phi_k^{(i)}. \tag{3}
$$

The new unknowns $u_k^{(i)}$ in (3) can be determined by solving a linear system $A\mathbf{u} = \mathbf{b}$ of the form:

$$
\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & & \vdots \\ \vdots & & \ddots & \\ A_{N1} & \dots & & A_{NN} \end{bmatrix}
\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix}
=
\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_N \end{bmatrix},
$$

where the blocks all have dimension m:

$$
A_{ji} = \begin{bmatrix}
B\left(\phi_1^{(i)}, \phi_1^{(j)}\right) & B\left(\phi_2^{(i)}, \phi_1^{(j)}\right) & \dots & B\left(\phi_m^{(i)}, \phi_1^{(j)}\right) \\
B\left(\phi_1^{(i)}, \phi_2^{(j)}\right) & B\left(\phi_2^{(i)}, \phi_2^{(j)}\right) & & \vdots \\
\vdots & & \ddots & \\
B\left(\phi_1^{(i)}, \phi_m^{(j)}\right) & \dots & & B\left(\phi_m^{(i)}, \phi_m^{(j)}\right)
\end{bmatrix},
$$

$$
\mathbf{u}_i = \begin{bmatrix} u_1^{(i)} \\ u_2^{(i)} \\ \vdots \\ u_m^{(i)} \end{bmatrix}, \qquad
\mathbf{b}_j = \begin{bmatrix} L\left(\phi_1^{(j)}\right) \\ L\left(\phi_2^{(j)}\right) \\ \vdots \\ L\left(\phi_m^{(j)}\right) \end{bmatrix},
$$

for all $i, j = 1, ..., N$. This system is obtained by substituting the expression (3) for u_h and the basis functions $\phi_\ell^{(j)}$ for v into (2). Once the unknowns $u_k^{(i)}$ are solved from the system $A\mathbf{u} = \mathbf{b}$, the final SIPG approximation u_h can be obtained from (3).

2.3 Penalty parameter

The SIPG method involves the penalty parameter σ which penalizes the inter-element jumps to enforce weak continuity. This parameter should be selected carefully; on the one hand, it needs to be sufficiently large to ensure that the SIPG method converges and the coefficient matrix A is symmetric and positive definite (SPD) [26]. At the same time, it needs to be chosen as small as possible, since the condition number of A increases rapidly with the penalty parameter [7].

Computable theoretical lower bounds for a large variety of problems have been derived by Epshteyn and Riviere [11]. For one-dimensional diffusion problems, it suffices to choose $\sigma \geq 2\frac{k_1^2}{k_0}p^2$, where k_0 and k_1 are the global lower and upper bound, respectively, for the diffusion coefficient K. However, while this lower bound for σ is sufficient to ensure convergence (assuming the exact solution is sufficiently smooth), it can be unpractical for problems with strong variations in the coefficients. For instance, if the diffusion coefficient K takes values between 1 and 10^{-3}, we obtain $\sigma \geq 2{,}000p^2$, which is inconveniently large. For this reason, it is common practice to choose, e.g., $\sigma = 20$ rather than $\sigma = 20{,}000$ for such problems [8, 25].

An alternative strategy is to choose the penalty parameter based on local values of the diffusion-coefficient K, e.g., choosing $\sigma = 20K$ rather than $\sigma = 20$. It has been demonstrated numerically in [9] that a diffusion-dependent penalty parameter can benefit the efficiency of a linear solver (also cf. Section 5.2). For general tensors K, this strategy can be defined as follows: for an edge with normal $\mathbf{n}$, we set $\sigma = \alpha\lambda$, where $\lambda = \mathbf{n}^T K\mathbf{n}$ and α is a user-defined constant. The latter should be as small as possible in light of the discussion above.

If the diffusion is discontinuous, this definition may not be unique. For instance, in the example above, we could have $\lambda = 1$ on one side and $\lambda = 0.001$ on the other side of an edge. In such cases, it seems a safe choice to use the largest limit value of λ in the definition above (e.g., $\lambda = 1$ in the example). The reason for this is that theoretical stability and convergence analysis are usually based on a penalty parameter that is sufficiently large.

An alternative strategy for dealing with discontinuities is to use the harmonic average of both limit values [4, 6, 10, 12]. In this case, the penalty parameter reads $\sigma = 2\alpha\frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$, where λ_i and λ_j are based on the information in the mesh elements E_i and E_j, respectively, (adjacent to the edge under consideration). This choice is equivalent to using the minimum of both limit values [4, p. 5]. In that sense, it seems less "safe" than the maximum strategy above.

In [4, 6, 10, 12], the "harmonic" penalty parameter is used in combination with the symmetric-weighted interior penalty (SWIP) method. The main difference between the standard SIPG method and the SWIP method is the following: whenever an average of a function at a mesh element boundary is considered (denoted by $\{.\}$ in Section 2.1), the SWIP method uses a weighted average rather than the standard average. For this purpose, the weights typically depend on the diffusion coefficient, i.e., $w_i = \frac{\lambda_j}{\lambda_i+\lambda_j}$ and $w_j = \frac{\lambda_i}{\lambda_i+\lambda_j}$ (note that the harmonic penalty can then be written as $\sigma = \alpha(w_i\lambda_i + w_j\lambda_j)$).

In this paper, we study the effects of both a constant and a diffusion-dependent penalty parameter, using either the maximum or the harmonic strategy above. Furthermore, we consider both the SIPG and the SWIP method. Extension of the aforementioned theory in [11] for the diffusion-dependent penalty parameter is left for future research.

3 Two-level preconditioner

To solve the linear SIPG system obtained in the previous section, we start by considering the two-level preconditioner proposed by Dobrev et al. [8]. Section 3.1 specifies the corresponding coarse correction operator. Section 3.2 defines the resulting two-level preconditioner. Section 3.3 indicates its implementation in a standard preconditioned CG algorithm.

3.1 Coarse correction operator

The two-level preconditioner is defined in terms of a coarse correction operator $Q \approx A^{-1}$ that switches from the original DG test space to a coarse subspace, then performs a correction that is now simple in this coarse space, and finally switches back to the original DG test space. In this case, the coarse subspace is based on the piecewise constant basis functions.

More specifically, the coarse correction operator Q is defined as follows. Let R denote the so-called restriction operator, such that $A_0 := RAR^T$ is the SIPG matrix for polynomial degree $p = 0$. In other words, R is a matrix of size $N \times Nm$, such that $R_{i,(i-1)m+1} = 1$ for all $i = 1, ..., N$, and all other entries are zero (recall the polynomial space dimension $m = \frac{(p+1)(p+2)}{2}$). An example is given below. Using this notation, the coarse correction operator is defined as follows:

$$Q := R^T A_0^{-1} R \tag{4}$$

For example, for a Laplace problem on the domain $[0, 1]^2$ with $p = 1$, a uniform Cartesian mesh with 2×2 elements, and penalty parameter $\sigma = 10$, we obtain the following matrices:

$$A = \left[\begin{array}{ccc|ccc||ccc|ccc}
40 & 1 & 1 & -10 & 9 & 0 & -10 & 0 & 9 & 0 & 0 & 0 \\
1 & 25 & 0 & -9 & 8 & 0 & 0 & -3 & 0 & 0 & 0 & 0 \\
1 & 0 & 25 & 0 & 0 & -3 & -9 & 0 & 8 & 0 & 0 & 0 \\
\hline
-10 & -9 & 0 & 40 & -1 & 1 & 0 & 0 & 0 & -10 & 0 & 9 \\
9 & 8 & 0 & -1 & 25 & 0 & 0 & 0 & 0 & 0 & -3 & 0 \\
0 & 0 & -3 & 1 & 0 & 25 & 0 & 0 & 0 & -9 & 0 & 8 \\
\hline\hline
-10 & 0 & -9 & 0 & 0 & 0 & 40 & 1 & -1 & -10 & 9 & 0 \\
0 & -3 & 0 & 0 & 0 & 0 & 1 & 25 & 0 & -9 & 8 & 0 \\
9 & 0 & 8 & 0 & 0 & 0 & -1 & 0 & 25 & 0 & 0 & -3 \\
\hline
0 & 0 & 0 & -10 & 0 & -9 & -10 & -9 & 0 & 40 & -1 & -1 \\
0 & 0 & 0 & 0 & -3 & 0 & 9 & 8 & 0 & -1 & 25 & 0 \\
0 & 0 & 0 & 9 & 0 & 8 & 0 & 0 & -3 & -1 & 0 & 25
\end{array}\right],$$

$$A_0 = \left[\begin{array}{c|c||c|c}
40 & -10 & -10 & 0 \\
\hline
-10 & 40 & 0 & -10 \\
\hline\hline
-10 & 0 & 40 & -10 \\
\hline
0 & -10 & -10 & 40
\end{array}\right],$$

$$R = \left[\begin{array}{ccc|ccc||ccc|ccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}\right].$$

Observe that A_0 has the same structure as a central difference matrix (aside from a factor $\sigma = 10$). Furthermore, every element in A_0 is also present in the upper left corner of the corresponding block in the matrix A. This is because the piecewise constant basis functions are assumed to be in any polynomial basis. As a consequence, the matrix R contains elements equal to 0 and 1 only, and does not need to be stored explicitly; multiplications with R can be implemented by simply extracting elements or inserting zeros.

3.2 Two-level preconditioner

We can now formulate the two-level preconditioner proposed by Dobrev et al. [8]. To this end, consider the coarse correction operator Q defined in (4), a damping parameter $\omega \leq 1$, and an invertible smoother $M^{-1} \approx A^{-1}$. Then, the result $\mathbf{y} = P_{\text{prec}}\mathbf{r}$ of applying the two-level preconditioner to a vector $\mathbf{r}$ can be computed as follows:

$$\begin{aligned}
\mathbf{y}^{(1)} &= \omega M^{-1}\mathbf{r} && \text{(pre-smoothing)}, \\
\mathbf{y}^{(2)} &= \mathbf{y}^{(1)} + Q(\mathbf{r} - A\mathbf{y}^{(1)}) && \text{(coarse correction)}, \\
\mathbf{y} &= \mathbf{y}^{(2)} + \omega M^{-T}(\mathbf{r} - A\mathbf{y}^{(2)}) && \text{(post-smoothing)}.
\end{aligned} \tag{5}$$

In this paper, we consider block Jacobi and block Gauss-Seidel smoothing. These smoothers have the following property [30]:

$$M + M^T - \omega A \text{ is SPD.} \tag{6}$$

Using the more abstract analysis in [35, p. 66], condition (6) implies that the preconditioning operator P_{prec} is SPD. As a consequence, the two-level preconditioner can be implemented in a standard preconditioned CG algorithm (cf. Section 3.3 hereafter).

Requirement (6) also implies that the two-level preconditioner yields scalable convergence of the CG method (independent of the mesh element diameter) for a large class of problems. This has been shown for polynomial degree $p = 1$ by Dobrev et al. [8], using the analysis in [13].

3.3 Implementation in CG

Assuming (6), the two-level preconditioner is SPD and can be implemented in a standard preconditioned CG algorithm. Below, we summarize the implementation of this scheme for a given preconditioning operator P and start vector $\mathbf{x}_0$:

```
1.  $\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0$
2.  $\mathbf{y}_0 := P\mathbf{r}_0$
3.  $\mathbf{p}_0 := \mathbf{y}_0$
4.  for $j = 0, 1, \ldots$ until convergence do
5.      $\mathbf{w}_j := A\mathbf{p}_j$
6.      $\alpha_j := (\mathbf{r}_j, \mathbf{y}_j)/(\mathbf{p}_j, \mathbf{w}_j)$
7.      $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j\mathbf{p}_j$
8.      $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j\mathbf{w}_j$
9.      $\mathbf{y}_{j+1} = P\mathbf{r}_{j+1}$
10.     $\beta_j := (\mathbf{r}_{j+1}, \mathbf{y}_{j+1})/(\mathbf{r}_j, \mathbf{y}_j)$
11.     $\mathbf{p}_{j+1} := \mathbf{y}_{j+1} + \beta_j\mathbf{p}_j$
12. end
```

4 Deflation variant

Next, we cast the two-level preconditioner into the deflation framework using the abstract analysis in [34]. This results in an alternative two-level scheme. Section 4.1 defines the resulting operator. Section 4.2 compares the two-level preconditioner and the corresponding deflation variant in terms of computational costs. Section 4.3 discusses the coarse systems involved in each iteration. Section 4.4 considers the influence of damping of the smoother.

4.1 Two-level deflation

There are multiple strategies to construct a deflation method based on the components of the two-level preconditioner. An overview of different schemes is given in [34]. Below, we consider the so-called ADEF2 deflation scheme, as this type can be implemented relatively efficiently, and allows for inexact solving of coarse systems (cf. Section 4.3 later on). Lacroix et al. [18] have also studied this method in the alternative context of finite volume discretizations of multiphase flows.

Basically, this deflation variant is obtained by skipping the last smoothing step in (5). In other words, considering the coarse correction operator Q defined in (4), a damping parameter $\omega \leq 1$, and an invertible smoother $M^{-1} \approx A^{-1}$, the result $\mathbf{y} = P_{\text{defl}}\mathbf{r}$ of applying the two-level deflation technique to a vector $\mathbf{r}$ can be computed as follows:

$$\begin{aligned} \mathbf{y}^{(1)} &:= \omega M^{-1}\mathbf{r} && \text{(pre-smoothing)}, \\ \mathbf{y} &:= \mathbf{y}^{(1)} + Q(\mathbf{r} - A\mathbf{y}^{(1)}) && \text{(coarse correction)}. \end{aligned} \tag{7}$$

The operator P_{defl} is not symmetric in general. As such, it seems unsuitable for the standard preconditioned CG method. Interestingly, it can still be implemented successfully in its current asymmetric form, as long as the smoother M^{-1} is SPD (requirement (6) is not needed), and the start vector $\mathbf{x}_0$ is pre-processed according to the following:

$$\mathbf{x}_0 \mapsto Q\mathbf{b} + (I - AQ)^T\mathbf{x}_0. \tag{8}$$

Other than that, the CG implementation remains as discussed in Section 3.3. Indeed, it has been shown by [34, Theorem 3.4] that, under the aforementioned conditions, P_{defl} yields the same CG iterates as an alternative operator (called “BNN” [19], [20]) that actually is SPD. For extensive analysis of (8), we also refer to [17, 36].

4.2 FLOPS

Because the deflation variant skips one of the two smoothing steps, its costs per CG iteration are lower than for the preconditioning variant. In this section, we compare the differences in terms of floating point operations (FLOPS).

Table 1 displays the (approximate) costs for a two-dimensional diffusion problem with polynomial degree p, a Cartesian mesh with $N = n \times n$ elements, and polynomial space dimension $m := \frac{(p+1)(p+2)}{2}$. Using the preconditioning variant, the CG method requires per iteration $(27m^2 + 14m)N$ flops, plus the costs for two smoothing steps and one coarse solve. Using the two-level deflation method, the CG method requires per iteration $(18m^2 + 12m)N$ flops, plus the costs for only one smoothing step and one coarse solve.

A block Jacobi smoothing step with blocks of size m requires $(2m^2 - m)N$ flops, assuming that an LU-decomposition is known. In this case, the smoothing costs are low compared to the costs for a matrix-vector product, and the deflation variant is roughly 30 % cheaper (per iteration). For more expensive smoothers, this factor becomes larger. A block Gauss-Seidel sweep (either forward or backward) requires the costs for one block Jacobi step, plus the costs for the updates based on the off-diagonal elements, which are approximately $4m^2N$ flops.

Table 1 Comparing the computational costs per CG iteration for A-DEF2 deflation and the two-level preconditioner for our applications

Operation	Flops	# defl.	# prec.
mat-vec (Au)	$9\,m^2N$	2	3
Inner product (u^Tv)	$2\,mN$	2	2
Scalar multiplication (αu)	mN	3	3
Vector update ($u \pm v$)	mN	5	7
Smoothing ($M^{-1}u$)	Variable	1	2
Coarse solve ($A_0^{-1}u_0$)	Variable	1	1

4.3 Coarse systems

Both two-level methods require the solution of a coarse system in each iteration, involving the coefficient matrix A_0. In Section 3.1, we have seen that A_0 has the same structure and size ($N \times N$) as a central difference matrix. As a consequence, a direct solver is not feasible for most practical applications. At the same time, many effective inexact solvers are readily available for this type of system.

For some deflation methods, including DEF1, DEF2, R-BNN1, R-BNN2, such an inexact coarse solver is not suitable (an overview of alternative preconditioning strategies is given in [34], and is summarized in Table 2). This is because those methods contain eigenvalue clusters at 0, so that small perturbations in those schemes (e.g., due to inexact coarse solves) can result in an “unfavorable spectrum, resulting in slow convergence of the method”—[34, p. 353]. ADEF2 does not have this limitation, as it clusters these eigenvalues at 1 rather than 0. This is one of the reasons why we focus on this particular deflation variant. However, we stress that (fast) convergence of the overall method is only established if the accuracy of the approximate solution of the coarse systems is sufficiently high, as is also supported by our numerical results in Section 5.3 later on.

In Section 5.3, we will investigate the use of an inexact coarse solver that applies the CG method in an inner loop with a scalable algebraic multigrid preconditioner. This

Table 2 Summary of alternative preconditioning strategies as given in [34]

Name	Operator
PREC	M^{-1}
DEF1	$M^{-1}R^T$
DEF2	RM^{-1}
A-DEF2	$RM^{-1} + Q$
BNN	$RM^{-1}R^T + Q$
R-BNN1	$RM^{-1}R^T$
R-BNN2	RM^{-1}

strategy will be studied for both the two-level preconditioner and the ADEF2 deflation variant.

An alternative strategy is the flexible CG (FCG) method [3, 22]. The main difference with standard CG lies in the explicit orthogonalization and truncation of the search direction vectors, possibly combined with a restart strategy. We do not study the FCG method in this paper, as we will see in Section 5.3 that the simpler standard preconditioned CG method is sufficient for our application. Alternative problems might benefit from the more advanced FCG strategy though. Investigation of the latter is left for future research.

4.4 Damping

Damping often benefits the convergence of multigrid methods [40]. For multigrid methods with smoother $M = I$, a "typical choice of [ω] is close to $\frac{1}{||A||_2}$," although a "better choice of [ω] is possible if we make further assumptions on how the eigenvectors of A associated with small eigenvalues are treated by coarse-grid correction"—[33, p. 1727]. In that reference, the latter is established for a coarse space that is based on a set of orthonormal eigenvectors of A. However, such a result does not seem available yet for the coarse space (and smoothers) currently under consideration.

At the same time, deflation may not be influenced by damping at all. The latter has been observed theoretically in [33, p. 1727] for the DEF(1) variant. For the ADEF2 variant under consideration, such a result is not yet available.

Altogether, it is an open question how the damping parameter can best be selected in practice. For this reason, we use an emprical approach in this paper, and study the effects on both two-level methods for several values of $\omega \leq 1$.

5 Numerical experiments

Next, we compare the two-level preconditioner and the corresponding deflation variant through numerical experiments. Section 5.1 specifies the experimental setup. Section 5.2 studies the influence of SIPG penalty parameter. Section 5.3 investigates the effectiveness of an inexact solver for the coarse systems. Section 5.4 studies the impact of (damping of) the smoother on the overall computational efficiency. Section 5.5 considers similar experiments for more challenging test cases.

5.1 Experimental setup

We consider multiple diffusion problems of the form (1) with strong contrasts in the coefficients on the domain $[0, 1]^2$. At first, we primarily focus on the problem illustrated in Fig. 1. This test case, inspired by [38], has five

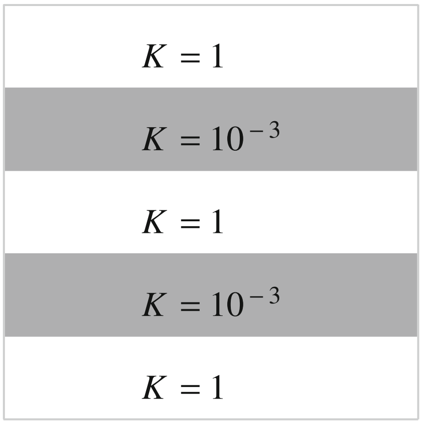


Fig. 1 Permeability for the problem with five layers

layers, and the diffusion is either 1 or 10^{-3} in each layer, mimicking the presence of sandstone and shale. In Section 5.5, we also study problems that mimic the occurrence of sand inclusions within a layer of shale, and ground water flow. Furthermore, we consider an anisotropic problem.

The Dirichlet boundary conditions and the source term f are chosen such that the exact solution reads $u(x, y) = \cos(10\pi x)\cos(10\pi y)$ (unless indicated otherwise). We stress that this choice does not impact the matrix or the performance of the linear solver, as we use random start vectors (see below). Furthermore, subdividing the domain into 10×10 equally sized squares, the diffusion coefficient is constant within each square.

All model problems are discretized by means of the SIPG method as discussed in Section 2, although the SWIP variant with weighted averages is also discussed. We use a uniform Cartesian mesh with $N = n \times n$ elements, where $n = 20, 40, 80, 160, 320$. Furthermore, we use monomial basis functions with polynomial degree $p = 3$ (results for $p = 1, 2$ are similar though). For $N = 320^2$ mesh elements, this means that the number of degrees of freedom is a little over 10^6. In most cases, the penalty parameter is chosen diffusion-dependent, $\sigma = 20\mathbf{n}^T K\mathbf{n}$, using the largest limit value at discontinuities (cf. Section 2.3). However, we also study a constant penalty parameter, and a parameter based on harmonic means.

The linear systems resulting from the SIPG discretizations are solved by means of the CG method, combined with either the two-level preconditioner (5) or the corresponding deflation variant (7). Unless specified otherwise, damping is not used. For the smoother M^{-1}, we use block Jacobi with small blocks of size $m \times m$ (recall that $m = \frac{(p+1)(p+2)}{2}$). For the preconditioning variant, we also consider block Gauss-Seidel with the same block size (deflation requires a symmetric smoother).

Diagonal scaling is applied as a pre-processing step in all cases, and the same random start vector $\mathbf{x}_0$ is used for all problems of the same size. Pre-processing of the start vector according to (8) is applied for deflation only, as it makes no difference for the preconditioning variant. For the stopping criterion, we use the following:

$$\frac{\|r_k\|_2}{\|b\|_2} \leq \text{TOL}, \tag{9}$$

Where TOL $= 10^{-6}$, and r_k is the residual after the kth iteration.

Coarse systems, involving the SIPG matrix A_0 with polynomial degree $p = 0$, are solved directly in most cases. However, a more efficient alternative is provided in Section 5.3. In any case, the coarse matrix A_0 is quite similar to a central difference matrix, for which very efficient solvers are readily available.

Finally, we remark that all computations are carried out using a Xeon E3-1240 V2 system and the GFortran (4.7.1) compiler.

5.2 The influence of the penalty parameter

This section studies the influence of the SIPG penalty parameter on the convergence of the CG and the SIPG methods. We compare the differences between using a constant penalty parameter, and a diffusion-dependent value. Similar experiments have been considered in [9] for the two-level preconditioner for $p = 1$, a single mesh, and symmetric Gauss-Seidel smoothing (solving the coarse systems using geometric multigrid). They found that "proper weighting," i.e., a diffusion-dependent penalty parameter, "is essential for the performance." In this section, we consider $p = 3$, and both preconditioning and deflation, both with block Jacobi smoothing. Furthermore, we analyze the scalability of the methods by considering multiple meshes. Our results are consistent with those in [9].

Table 3 displays the number of CG iterations required for convergence for a Poisson problem (i.e., $K = 1$ everywhere) with $\sigma = 20$. Because the diffusion coefficient is constant, a diffusion-dependent value ($\sigma = 20K$) would yield the same results. We observe that both the two-level preconditioner (TL prec.) and the deflation variant (TL defl.) yield fast and scalable convergence (independent of the mesh element diameter). For comparison, the results for standard Jacobi and block Jacobi preconditioning are also displayed (not scalable). Interestingly, the two-level deflation method requires fewer iterations than the preconditioning variant, even though its costs per iteration are about 30 % lower (cf. Section 4.2).

Table 4 considers the same test (using a constant $\sigma = 20$), but now for the problem with five layers (cf. Fig. 1). It can be seen that the convergence is no longer fast and scalable for this problem with large jumps in the coefficients. The deflation method is significantly faster than the preconditioning variant, but neither produce satisfactory results.

5.2.1 Diffusion-dependent penalty parameter

In Table 5, we revisit the experiment in Table 4, but this time for a diffusion-dependent penalty parameter ($\sigma = 20K$, using the largest limit value of K at discontinuities). Due to this alternative discretization, the results are now similar to those for the Poisson problem (cf. Table 3): both two-level methods yield fast and scalable convergence.

These results motivate the use of a diffusion-dependent penalty parameter, provided that this strategy does not worsen the accuracy of the SIPG discretization compared to a constant penalty parameter. In Fig. 2, it is verified that a diffusion-dependent penalty parameter actually improves the accuracy of the SIPG approximation. The higher accuracy can be explained by the fact that the discretization contains more information of the underlying physics for a diffusion-dependent penalty parameter. Altogether, the

Table 3 Both two-level methods yield fast scalable convergence for a problem with constant coefficients (Poisson, # CG iterations, $\sigma = 20$)

Mesh	$N = 20^2$	$N = 40^2$	$N = 80^2$	$N = 160^2$
Jacobi	325	576	1,114	1,903
Block Jacobi (BJ)	206	357	696	1,183
TL Prec., 2x BJ	49	52	53	54
TL Defl., 1x BJ	36	37	37	38

Table 4 For a problem with extreme contrasts in the permeability, a constant penalty yields poor convergence (five layers, # CG iterations, $\sigma = 20$)

Mesh	$N = 20^2$	$N = 40^2$	$N = 80^2$	$N = 160^2$
Jacobi	2,675	5,064	9,104	15,657
Block Jacobi (BJ)	1,357	2,960	5,660	9,783
TL Prec., 2x BJ	1,089	2,352	4,709	8,781
TL Defl., 1x BJ	453	591	667	698

Table 5 For a diffusion-dependent penalty parameter, both two-level methods yield fast scalable convergence for a problem with large permeability contrasts (five layers, # CG iterations, $\sigma = 20K$)

Mesh	$N = 20^2$	$N = 40^2$	$N = 80^2$	$N = 160^2$
Jacobi	1,303	1,490	1,919	3,109
Block Jacobi (BJ)	244	425	697	1,485
TL Prec., 2x BJ	55	56	56	57
TL Defl., 1x BJ	47	48	48	48

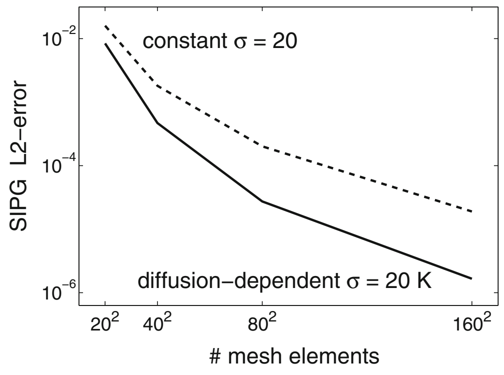


Fig. 2 A diffusion-dependent penalty parameter yields better SIPG accuracy (five layers, $\sigma = 20K$)

penalty parameter can best be chosen diffusion-dependent, and we will do so in the remaining of this paper.

5.2.2 Weighted averages

The results for the diffusion-dependent penalty parameter in Fig. 2 and Table 5 were established using the largest limit value of K in the definition of σ at the discontinuities. In this section, we consider the influence of using weighted averages, resulting in the SWIP method and a diffusion-dependent penalty parameter based on harmonic means (cf. Section 2.3). For this purpose, we study the problem with five layers again.

We have found that using $\sigma = 20K$ with this approach results in negative eigenvalues, implying that the scheme is not coercive, and resulting in poor CG convergence. The same is true for $\sigma = \alpha K$ with $\alpha = 100, 200, 500, 1{,}000$. When using $\alpha = 20{,}000$, the matrix is positive-definite (tested for $N = 10, 20$ and $p = 1, 2, 3$). Similar outcomes were found using the SIPG scheme rather than the SWIP scheme (for the same "harmonic" penalty parameter).

At the mesh element edges where the diffusion coefficient K is discontinuous, using $\alpha = 20{,}000$ and a harmonic penalty yields $\sigma = \frac{20}{1.001}$. At the same time, using $\alpha = 20$ and a "maximum" penalty (i.e., using the largest limit value at discontinuities) yields $\sigma = 20$. These values are nearly the same. However, at all other edges, where K is continuous, σ is 1,000 times larger for the harmonic penalty (with $\alpha = 20{,}000$) than for the maximum penalty (with $\alpha = 20$). Because the penalty parameter should be chosen as small as possible (cf. Section 2.3), we conclude that it can best be based on the largest limit value at discontinuities. This is in line with our earlier speculation that using the maximum is a "safe" choice.

We have also combined the "maximum" penalty with the SWIP method and compared the outcomes to the earlier results for the SIPG method (both for $\sigma = 20K$). We have found that the discretization accuracy and the CG convergence are practically the same; the relative absolute difference in the discretization error is less than 2 % (for $p = 1, 2, 3$ and $N = 20^2, 40^2, 80^2, 160^2$). Comparing Table 6 (SWIP) to Table 5 (SIPG), it can be seen that the number of CG iterations required for convergence is nearly identical.

Table 6 The difference between the SWIP method (this table) and the SIPG method (cf. Table 5) is small (five layers, # CG iterations)

Mesh	$N = 20^2$	$N = 40^2$	$N = 80^2$	$N = 160^2$
Jacobi	1,309	1,500	1,935	3,129
Block Jacobi (BJ)	244	424	697	1485
TL Prec., 2x BJ	55	56	56	57
TL Defl., 1x BJ	47	48	49	49

Altogether, we conclude that both the SIPG and the SWIP method are suitable for our application, as long as the penalty parameter is chosen diffusion-dependent, using the largest limit value at discontinuities. We will apply this strategy using the standard SIPG method in the remaining of this paper.

5.3 Coarse systems

To solve the coarse systems with coefficient matrix A_0, a direct solver is usually not feasible in practice. This is because A_0 has the same structure and size ($N \times N$) as a central difference matrix. To improve on the efficiency of the coarse solver, we have investigated the cheaper alternative of applying the CG method again in an inner loop (cf. Section 4.3). This section discusses the results using the algebraic multigrid preconditioner ML20 in the HSL software package,[1] which is based on a classical scheme described in [31]. The inner loop uses a stopping criterion of the form (9).

Table 7 displays the number of outer CG iterations required for convergence using the two-level preconditioner and deflation variant, respectively, (for the problem with five layers). Different values of the inner tolerance TOL are considered in these tables. For comparison, the results for the direct solver are also displayed. We observe that low accuracy in the inner loop is sufficient to reproduce the latter. For both two-level methods, the inner tolerance can be 10^4 times as large as the outer tolerance. For the highest acceptable inner tolerance TOL $= 10^{-2}$, the number of iterations in the inner loop is between 2 and 5 in all cases (not displayed in the tables).

In terms of computational time, the difference between the direct solver and the inexact AMG-CG solver is

[1]HSL, a collection of Fortran codes for large-scale scientific computation. See http://www.hsl.rl.ac.uk/

Table 7 Coarse systems can be solved efficiently by using an inexact solver with a relatively low accuracy (TOL) in the inner loop (five layers, # outer CG iterations)

Mesh	$N = 40^2$	$N = 80^2$	$N = 160^2$	$N = 320^2$
Preconditioning				
Direct	56	56	57	58
$\mathrm{TOL} = 10^{-3}$	56	56	57	58
$\mathrm{TOL} = 10^{-2}$	56	57	58	58
$\mathrm{TOL} = 10^{-1}$	59	65	70	78
Deflation				
Direct	48	48	48	49
$\mathrm{TOL} = 10^{-3}$	48	48	48	49
$\mathrm{TOL} = 10^{-2}$	48	48	48	49
$\mathrm{TOL} = 10^{-1}$	48	54	79	67

negligible for the problems under consideration. However, for large three-dimensional problems, it can be expected that the inexact coarse solver is much faster, and thus crucial for the overall efficiency and scalability of the linear solver.

5.4 Smoothers and damping

This section discusses the influence of the smoother and damping on both two-level methods. In particular, we consider multiple damping values $\omega \in [0.5, 1]$, and both Jacobi and (block) Gauss-Seidel smoothing. The latter is applied for the preconditioning variant only, as deflation requires a symmetric smoother.

Table 8 displays the number of CG iterations required for convergence for the problem with five layers. For the deflation variant (Defl.), we have found that damping makes no difference for the CG convergence, so the outcomes for $\omega < 1$ are not displayed. Such a result has also been observed theoretically in [33] for an alternative deflation variant (known as DEF(1)).

Table 8 Damping can improve the convergence for the preconditioner, but has no influence for deflation (five layers, # CG Iterations)

Mesh	$N = 40^2$	$N = 80^2$	$N = 160^2$	$N = 320^2$
Prec., 2x BJ ($\omega = 1$)	56	56	57	58
($\omega = 0.9$)	40	40	42	43
($\omega = 0.8$)	36	37	39	39
($\omega = 0.7$)	35	36	36	37
($\omega = 0.6$)	35	36	36	37
($\omega = 0.5$)	36	37	38	39
Prec., 2x BGS ($\omega = 1$)	34	35	35	37
($\omega = 0.9$)	34	34	35	36
($\omega = 0.8$)	35	34	36	37
($\omega = 0.7$)	35	36	37	38
($\omega = 0.6$)	36	37	38	39
($\omega = 0.5$)	37	39	39	40
Defl., 1x BJ ($\omega = 1$)	48	48	48	49

Table 9 The deflation method and the block Jacobi smoother tend to be cheaper due to lower costs per iteration (five layers, CPU time in seconds)

Mesh	$N = 40^2$	$N = 80^2$	$N = 160^2$	$N = 320^2$
Prec., 2x BJ ($\omega = 1$)	0.19	0.82	3.86	17.61
($\omega = 0.9$)	0.14	0.60	2.80	13.14
($\omega = 0.8$)	0.12	0.56	2.60	11.96
($\omega = 0.7$)	0.12	0.54	2.41	11.37
($\omega = 0.6$)	0.12	0.54	2.41	11.36
($\omega = 0.5$)	0.12	0.55	2.54	11.95
Prec., 2x BGS ($\omega = 1$)	0.21	0.93	3.93	18.07
($\omega = 0.9$)	0.21	0.91	3.93	17.62
($\omega = 0.8$)	0.22	0.91	4.04	18.08
($\omega = 0.7$)	0.21	0.96	4.15	18.55
($\omega = 0.6$)	0.22	0.98	4.26	19.03
($\omega = 0.5$)	0.23	1.03	4.37	19.50
Defl., 1x BJ ($\omega = 1$)	0.11	0.50	2.39	11.66

For the preconditioning variant (Prec.), damping can both improve and worsen the efficiency; for block Jacobi smoothing, choosing, e.g., $\omega = 0.7$, can reduce the number of iterations by 37 %; for block Gauss-Seidel smoothing, choosing $\omega < 1$ has either no influence or a small negative impact in most cases.

We have also performed the experiment for standard point Jacobi and point Gauss-Seidel (not displayed in the table). However, this did not lead to satisfactory results; over 250 iterations in all cases, even when the relaxation parameter was sufficiently low to ensure that (6) is satisfied (only restricting for point Jacobi). Altogether, the block (Jacobi/Gauss-Seidel) smoothers yield significantly better results than the point (Jacobi/Gauss-Seidel) smoothers.

We speculate that this is due to the following: the coarse correction operator Q simplifies the matrix A to A_0, which

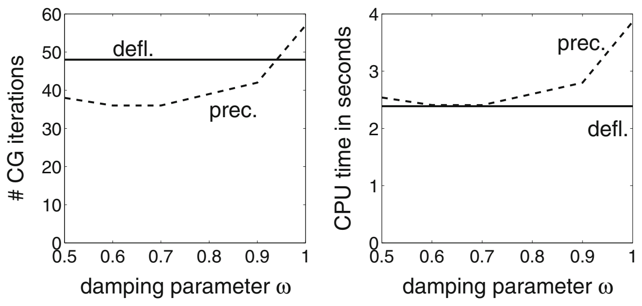
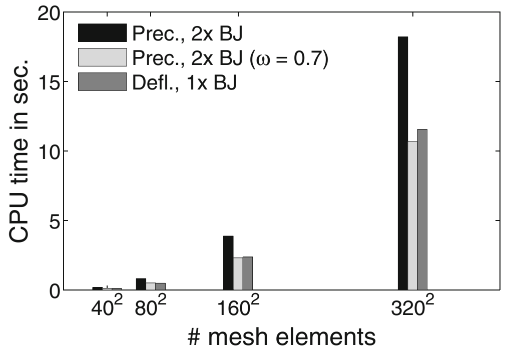
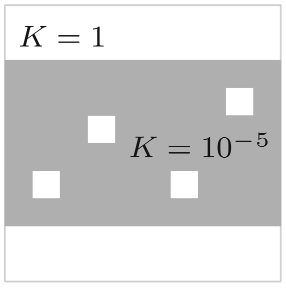


Fig. 3 Unless an optimal damping parameter is known, deflation is cheaper due to lower costs per iteration (five layers, $N = 160^2$, block Jacobi)

Table 10 Sand inclusions





# CG iterations				
Mesh	$N = 40^2$	$N = 80^2$	$N = 160^2$	$N = 320^2$
Degrees of freedom	16,000	64,000	256,000	1,024,000
Prec., 2x BJ	53	56	58	59
Prec., 2x BJ ($\omega = 0.7$)	32	34	34	34
Defl., 1x BJ	46	47	48	48

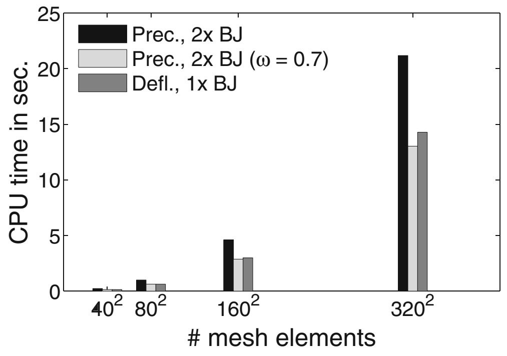
eliminates the “higher-order” information in each element (regarding the higher-order basis functions), but preserves the “mesh” information (i.e., which elements are neighbors and which are not). Intuitively, a suitable smoother would reintroduce this higher-order information, originally contained in dense blocks of size $m \times m$. The block (Jacobi/Gauss-Seidel) smoothers are better suited for this task, which could explain why they are more effective.

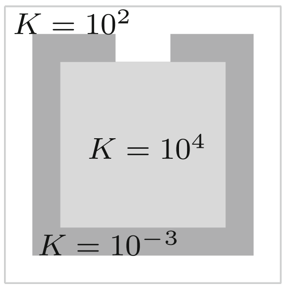
5.4.1 Computational time

Based on the results in Table 8, it appears that the preconditioning variant with either block Jacobi (with optimal damping) or block Gauss-Seidel is the most efficient choice. However, the costs per iteration also need to be taken into account.

Table 9 reconsiders the results in Table 8 but now in terms of the computational time in seconds (using a direct coarse

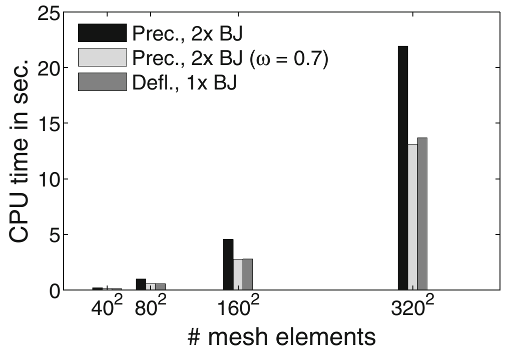
Table 11 Ground water





# CG iterations				
Mesh	$N = 40^2$	$N = 80^2$	$N = 160^2$	$N = 320^2$
Degrees of freedom	16,000	64,000	256,000	1,024,000
Prec., 2x BJ	67	68	68	69
Prec., 2x BJ ($\omega = 0.7$)	41	42	42	42
Defl., 1x BJ	59	59	60	60

Table 12 Anisotropy



$$K = 10^{-3}\begin{bmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

$$K = \begin{bmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

# CG iterations				
Mesh	$N = 40^2$	$N = 80^2$	$N = 160^2$	$N = 320^2$
Degrees of freedom	16,000	64,000	256,000	1,024,000
Prec., 2x BJ	61	67	68	71
Prec., 2x BJ ($\omega = 0.7$)	39	39	41	42
Defl., 1x BJ	54	55	57	57

solver). It can be seen that block Gauss-Seidel smoothing is relatively expensive. The deflation variant (with block Jacobi smoothing) is the fastest in nearly all cases. This is due to the fact that it requires only one smoothing step per iteration, instead of two. When an optimal damping parameter is known, the preconditioning variant reaches a comparable efficiency. This is also illustrated in Fig. 3. However, it is an open question how the damping parameter can best be selected in practice.

5.5 Other test cases

In this section, we repeat the experiments in Table 9 for more challenging test cases. For the preconditioning variant, we only display the results for block Jacobi smoothing without damping ($\omega = 1$) and with optimal damping ($\omega = 0.7$).

Tables 10 and 11 consider problems that mimic the occurrence of sand inclusions within a layer of shale and groundwater flow, respectively. Similar problems have been studied in [38]. Table 12 considers an anisotropic problem with two layers (with exact solution $u(x, y) = cos(2\pi y)$). Because the diffusion is a full tensor, this test case mimics the effect of using a non-Cartesian mesh.

It can be seen from these tables that, as before, both two-level methods yield fast and scalable convergence. Without damping, deflation is the most efficient. When an optimal damping value is known, the preconditioning variant performs comparable to deflation.

6 Conclusion

This paper compares the two-level preconditioner proposed in [8] and the alternative ADEF2-deflation variant for linear systems resulting from SIPG discretizations. We have found that both two-level methods yield fast and scalable convergence for diffusion problems with large jumps in the coefficients. This result is obtained provided that the SIPG penalty parameter is chosen dependent on local values of the permeability (using the largest limit value at discontinuities). The latter also benefits the accuracy of the SIPG discretization. Furthermore, the impact of using weighted averages (SWIP) is then small. Coarse systems can be solved efficiently by applying the CG method again in an inner loop with low accuracy.

The main difference between both methods is that the deflation method can be implemented by skipping one of the two smoothing steps in the algorithm for the preconditioning variant. This may be particularly advantageous for expensive smoothers, although the basic block Jacobi smoother was found to be highly effective for the problems under consideration. Without damping, deflation can be up to 35 % cheaper than the original preconditioner. If an optimal damping parameter is used, both two-level strategies yield similar efficiency (deflation appears unaffected by damping). However, it remains an open question how the damping parameter can best be selected in practice.

Altogether, both two-level methods can contribute to cheaper and more accurate pressure simulations for layered systems with strong permeability contrasts. Future research could focus on theoretical support for these findings and more advanced, larger-scale test cases, including three-dimensional problems on more realistic geometries.

References

1. Antonietti, P.F., Ayuso, B.: Schwarz domain decomposition preconditioners for discontinuous Galerkin approximations of elliptic problems: non-overlapping case. M2AN Math. Model. Numer. Anal. **41**(1), 21–54 (2007)
2. Arnold, D.N., Brezzi, F., Cockburn, B., Marini, L.D.: Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J. Numer. Anal. **39**(5), 1749–1779 (electronic) (2002)
3. Axelsson, O., Vassilevski, P.S.: Variable-step multilevel preconditioning methods. I. Selfadjoint and positive definite elliptic problems. Numer. Linear Algebra Appl. **1**(1), 75–101 (1994)
4. Ayuso de Dios, B., Holst, M., Zhu, Y., Zikatanov, L.: Multilevel preconditioners for discontinuous Galerkin approximations of elliptic problems with jump coeffients. arXiv:1012.1287v2 (2012)
5. Brenner, S.C., Zhao, J.: Convergence of multigrid algorithms for interior penalty methods. Appl. Numer. Anal. Comput. Math. **2**(1), 3–18 (2005)
6. Burman, E., Zunino, P.: A domain decomposition method based on weighted interior penalties for advection-diffusion-reaction problems. SIAM J. Numer. Anal. **44**(4), 1612–1638 (electronic) (2006)
7. Castillo, P.: Performance of discontinuous Galerkin methods for elliptic PDEs. SIAM J. Sci. Comput. **24**(2), 524–547 (2002)
8. Dobrev, V.A., Lazarov, R.D., Vassilevski, P.S., Zikatanov, L.T.: Two-level preconditioning of discontinuous Galerkin approximations of second-order elliptic equations. Numer. Linear Algebra Appl. **13**(9), 753–770 (2006)
9. Dobrev, V.A., Lazarov, R.D., Zikatanov, L.T.: Preconditioning of symmetric interior penalty discontinuous Galerkin FEM for elliptic problems. In: Domain Decomposition Methods in Science and Engineering XVII, Lecture Notes in Computer Science and Engineering, vol. 60, pp. 33–44. Springer, Berlin (2008)
10. Dryja, M.: On discontinuous Galerkin methods for elliptic problems with discontinuous coefficients. Comput. Methods Appl. Math. **3**(1), 76–85 (electronic) (2003). Dedicated to Raytcho Lazarov
11. Epshteyn, Y., Rivière, B.: Estimation of penalty parameters for symmetric interior penalty Galerkin methods. J. Comput. Appl. Math. **206**(2), 843–872 (2007)
12. Ern, A., Stephansen, A., Zunino, P.: A discontinuous Galerkin method with weighted averages for advection-diffusion equations with locally small and anisotropic diffusivity. IMA J. Numer. Anal. **29**(2), 235–256 (2009). doi:10.1093/imanum/drm050
13. Falgout, R.D., Vassilevski, P.S., Zikatanov, L.T.: On two-grid convergence estimates. Numer. Linear Algebra Appl. **12**(5–6), 471–494 (2005)
14. Feng, X., Karakashian, O.A.: Two-level additive Schwarz methods for a discontinuous Galerkin approximation of second order elliptic problems. SIAM J. Numer. Anal. **39**(4), 1343–1365 (electronic) (2001)
15. Fidkowski, K.J., Oliver, T.A., Lu, J., Darmofal, D.L.: p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. J. Comput. Phys. **207**(1), 92–113 (2005). doi:10.1016/j.jcp.2005.01.005
16. Gopalakrishnan, J., Kanschat, G.: A multilevel discontinuous Galerkin method. Numer. Math. **95**(3), 527–550 (2003)
17. Kuznetsov, Y.: Matrix computational processes in subspaces. In: Glowinski, R., Lions, J. (eds) Computing Methods in Applied Sciences and Engineering, vol. VI, pp. 15–31. North-Holland. (1984). Proceedings of the 6th International Symposium on Computing Methods in Applied Sciences and Engineering, Versailles, France, December 12–16, (1983)
18. Lacroix, S., Vassilevski, Y., Wheeler, J., Wheeler, M.: Iterative solution methods for modeling multiphase flow in porous media fully implicitly. SIAM J. Sci. Comput. **25**(3), 905–926 (2003)
19. Mandel, J.: Balancing domain decomposition. Commun. Numer. Methods Eng. **9**, 233–241 (1993)
20. Mandel, J., Brezina, M.: Balancing domain decomposition for problems with large jumps in coefficients. Math. Comput. **65**(216), 1387–1401 (1996)
21. Nicolaides, R.A.: Deflation of conjugate gradients with applications to boundary value problems. SIAM J. Numer. Anal. **24**(2), 355–365 (1987)
22. Notay, Y.: Flexible conjugate gradients. SIAM J. Sci. Comput. **22**(4), 1444–1460 (electronic) (2000)
23. Persson, P.O., Peraire, J.: Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations. SIAM J. Sci. Comput. **30**(6), 2709–2733 (2008)
24. Prill, F., Lukáčová-Medviďová, M., Hartmann, R.: Smoothed aggregation multigrid for the discontinuous Galerkin method. SIAM J. Sci. Comput. **31**(5), 3503–3528 (2009)
25. Proft, J., Rivière, B.: Discontinuous Galerkin methods for convection-diffusion equations for varying and vanishing diffusivity. Int. J. Numer. Anal. Model. **6**(4), 533–561 (2009)
26. Rivière, B.: Discontinuous Galerkin methods for solving elliptic and parabolic equations. In: Frontiers in Applied Mathematics, vol. 35. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2008). Theory and implementation
27. Rivière, B., Wheeler, M., Banaš, K.: Part ii. discontinuous galerkin method applied to a single-phase flow in porous media. Comput. Geosci. **4**, 337–349 (2000)
28. Saad, Y., Suchomel, B.: ARMS: an algebraic recursive multilevel solver for general sparse linear systems. Numer. Linear Algebra Appl. **9**(5), 359–378 (2002). doi:10.1002/nla.279
29. Sherwin, S.J., Kirby, R.M., Peiró, J., Taylor, R.L., Zienkiewicz, O.C.: On 2D elliptic discontinuous Galerkin methods. Int. J. Numer. Methods Eng. **65**(5), 752–784 (2006)
30. van Slingerland, P., Vuik, C.: Scalable two-level preconditioning and deflation base on a piecewise constant subspace for (SIP)DG systems. Tech. Rep. 12–11, Delft University of Technology (2012)
31. Stüben, K.: An introduction to algebraic multigrid. In: Trottenberg, U., Oosterlee, C.W., Schüller, A. (eds.) Multigrid, pp. 413–532. Academic Press, New York (2001)
32. Sun, S., Wheeler, M.: Local problem-based a posteriori error estimators for discontinuous galerkin approximations of reactive transport. Comput. Geosci. **11**(2), 87–101 (2007)
33. Tang, J.M., MacLachlan, S.P., Nabben, R., Vuik, C.: A comparison of two-level preconditioners based on multigrid and deflation. SIAM J. Matrix Anal. Appl. **31**(4), 1715–1739 (2010)
34. Tang, J.M., Nabben, R., Vuik, C., Erlangga, Y.A.: Comparison of two-level preconditioners derived from deflation, domain

decomposition, and multigrid methods. J. Sci. Comput. **39**(3), 340–370 (2009)
35. Vassilevski, P.S.: Multilevel block factorization preconditioners. Matrix-based analysis and algorithms for solving finite element equations. Springer, New York (2008)
36. Vassilevski, Y.: A hybrid domain decomposition method based on aggregation. Numer Linear Algebra Appl. **11**(4), 327–341 (2004)
37. Vuik, C., Segal, A., Meijerink, J.: An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients. J. Comput. Phys. **152**, 385–403 (1999)
38. Vuik, C., Segal, A., Meijerink, J., Wijma, G.: The construction of projection vectors for a Deflated ICCG method applied to problems with extreme contrasts in the coefficients. J. Comput. Phys. **172**, 426–450 (2001)
39. Xu, J.: Iterative methods by space decomposition and subspace correction. SIAM Rev **34**(4), 581–613 (1992)
40. Yavneh, I.: Why multigrid methods are so efficient. Comput. Sci. Eng. **8**(6), 12–22 (2006)