A preconditioner for CG that does not need symmetry

P. van Slingerland*†and C. Vuik†

December 8, 2011

Extended summary

Although it is common to use a *symmetric* preconditioner for the conjugate gradient method, in this paper, we demonstrate that a *non-symmetric* preconditioning strategy can be significantly more efficient. The focus, motivation and conclusions of this research can be summarized as follows.

Focus This work is focused on an effective preconditioning strategy to increase the efficiency of the Conjugate Gradient (CG) method for Symmetric and Positive-Definite (SPD) linear systems resulting from Symmetric Interior Penalty (discontinuous) Galerkin (SIPG) discretizations for diffusion problems with extreme contrasts in the coefficients, such as those encountered in oil reservoir simulations.

Motivation A discontinuous Galerkin discretization can be thought of as a finite volume method that uses (discontinuous) piecewise polynomials of degree p rather than piecewise constants. As such, it combines the best of both classical finite element methods and finite volume methods, and it is particularly suitable for handling non-matching grids and designing hp-refinement strategies. However, a relevant drawback is that its resulting linear system is often ill-conditioned and relatively large due to the large number of unknowns per mesh element. In search of suitable iterative solution techniques, much attention has been paid to subspace correction methods, such as Schwarz domain decomposition [1]; geometric (h-)multigrid [2]; spectral (p-)multigrid [6]; and algebraic multigrid [7]. In particular, Dobrev et al. [4] have proposed a spectral two-level preconditioner that makes use of coarse corrections based on the solution approximation with polynomial degree $p = 0$. It has been shown theoretically that this preconditioner yields uniform convergence of the CG method (independent of the mesh element diameter) for a large class of problems. 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.

However, two main issues remain when using this preconditioner for a SIPG matrix A: First, *two* smoothing steps must be applied during each iteration, and the smoother needs to satisfy an inconvenient criterion to ensure that the preconditioning operator is SPD. The second issue is that the SIPG method involves a stabilizing penalty parameter, whose influence on both A and the preconditioner is not well understood for problems with strongly varying coefficients. On the one hand, this parameter needs to be chosen suffciently large to ensure that the SIPG method is stable and convergent, and that A is SPD. At the same time, it needs to be chosen as small as possible to avoid an unnecessarily large condition number. Known computable theoretical lower bounds [5] are based on the ratio between the *global* maximum and minimum of the diffusion coefficient, and are therefore impractical for our application.

*Email: p.vanslingerland@tudelft.nl
†Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands

To eliminate one of the two smoothing steps and the inconvenient restriction on the smoother at the same time, we have cast the spectral two-level preconditioner into the deflation framework [11]. Additionally, we have studied the potential of a penalty parameter that is based on *local* values of the diffusion coefficient, instead of the usual strategy to use one global constant for the entire domain.

This paper discusses how and why the proposed spectral two-level deflation method can be incorporated in a CG algorithm in the form of an *asymmetric* preconditioning operator. Furthermore, it demonstrates numerically how the resulting iterative scheme performs for diffusion problems with strongly varying coefficients, for both a constant and a diffusion-dependent penalty parameter. More background information is provided in technical report [10].

Conclusions Our main findings are illustrated in Figure 1: by reformulating the spectral two-level preconditioner as a deflation method and using a diffusion-dependent penalty parameter, the SIPG convergence is significantly better, and the CG method can become over 100 times faster, while retaining uniform convergence (independent of the mesh element diameter).

Altogether, the efficiency of a symmetric two-level preconditioner can be significantly improved by switching to an asymmetric deflation variant. Furthermore, the SIPG penalty parameter can best be chosen diffusion-dependent for problems with extreme contrasts in the coefficients.

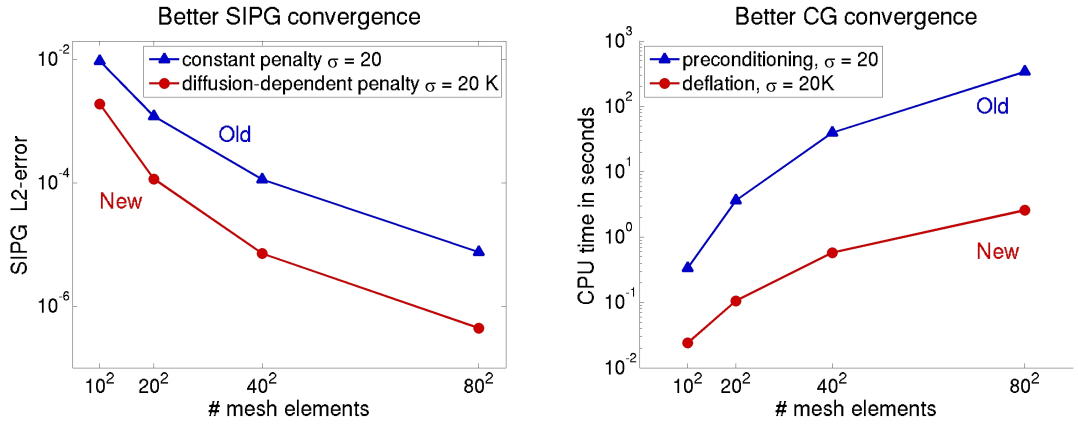


Figure 1: Illustration for a diffusion problem on a square domain with five layers, in which the diffusion coefficient K is either 1 or 10^{-3}. By reformulating the spectral two-level preconditioner as a deflation method and using a diffusion-dependent penalty parameter ($\sigma = 20K$ rather than $\sigma = 20$), the SIPG convergence is significantly better, and the CG method can become over 100 times faster. Also cf. Section 3.1 later on for more details ($p = 3$, uniform Cartesian mesh).

1 Discretization

1.1 Introducing the SIPG method and the resulting linear systems

SIPG method The SIPG approximation for a diffusion problem $-\nabla\cdot(K\nabla u) = f$ can be defined in the following manner. For a given mesh, define the test space V, which 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

$$B(u_h, v) = L(v), \qquad \text{for all test functions } v \in V, \tag{1}$$

where B and L are certain (bi)linear forms that characterize the SIPG method [9].

Monomial basis functions In order to compute the SIPG approximation defined by (1), it needs to be rewritten as a linear system. To this end, we choose monomial basis functions $\phi_k^{(i)}$ for the test space V. For instance, for a one-dimensional uniform mesh with element size h and polynomial degree $p = 2$, the basis functions are zero in the entire domain, except in mesh element i with center x_i, where they read:

$$\phi_1^{(i)}(x) = 1, \qquad \phi_2^{(i)}(x) = \frac{x - x_i}{\frac{1}{2}h}, \qquad \phi_3^{(i)}(x) = \left(\frac{x - x_i}{\frac{1}{2}h}\right)^2.$$

Two-dimensional monomial basis functions are defined similarly [9]. Next, we express u_h as a linear combination of the basis functions:

$$u_h = \sum_{i=1}^{N} \sum_{k=1}^{m} u_k^{(i)} \phi_k^{(i)}, \tag{2}$$

where $m := p+1$ for one-dimensional problems, and $m := \frac{(p+1)(p+2)}{2}$ for two-dimensional problems.

Linear system The new unknowns $u_k^{(i)}$ in (2) can be determined by solving a linear system $A\boldsymbol{u} = \boldsymbol{b}$ of the form:

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & & \vdots \\ \vdots & & \ddots & \\ A_{N1} & \dots & & A_{NN} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_N \end{bmatrix}, \tag{3}$$

where the blocks all have dimension m, and where, for all $i, j = 1, ..., N$:

$$A_{ji} = \begin{bmatrix} B(\phi_1^{(i)}, \phi_1^{(j)}) & B(\phi_2^{(i)}, \phi_1^{(j)}) & \dots & B(\phi_m^{(i)}, \phi_1^{(j)}) \\ B(\phi_1^{(i)}, \phi_2^{(j)}) & B(\phi_2^{(i)}, \phi_2^{(j)}) & & \vdots \\ \vdots & & \ddots & \\ B(\phi_1^{(i)}, \phi_m^{(j)}) & \dots & & B(\phi_m^{(i)}, \phi_m^{(j)}) \end{bmatrix}, \quad \boldsymbol{u}_i = \begin{bmatrix} u_1^{(i)} \\ u_2^{(i)} \\ \vdots \\ u_m^{(i)} \end{bmatrix}, \quad \boldsymbol{b}_j = \begin{bmatrix} L(\phi_1^{(j)}) \\ L(\phi_2^{(j)}) \\ \vdots \\ L(\phi_m^{(j)}) \end{bmatrix}. \tag{4}$$

This system is obtained by substituting the expression (2) for u_h and the basis functions $\phi_\ell^{(j)}$ for v into (1). A concrete matrix example is provided in Section 2.1 later on. Once the unknowns $u_k^{(i)}$ are solved from the system $A\boldsymbol{u} = \boldsymbol{b}$, the final SIPG approximation u_h can be obtained from (2).

1.2 Switching to a diffusion-dependent penalty parameter

Penalty parameter The SIPG method discussed in the previous section involves a penalty parameter σ, which enforces stability by penalizing inter-element discontinuities. On the one hand, this parameter needs to be chosen suffciently large to ensure that the SIPG method converges and the coefficient matrix A is SPD. 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 [3].

Theoretical bounds Computable theoretical lower bounds have been derived for a large variety of problems by Epshteyn and Riviere [5]. For one-dimensional diffusion problems, they propose:

$$\begin{aligned} \sigma &\geq \tfrac{k_1^2}{k_0} p^2, && \text{for interior edges,} \\ \sigma &\geq 2\tfrac{k_1^2}{k_0} p^2, && \text{for boundary edges,} \end{aligned} \tag{5}$$

where k_0 and k_1 are the *global* lower and upper bounds for the diffusion coefficient K (and p is the polynomial degree). However, while these lower bounds are sufficient to ensure stability and convergence, lower values of σ are usually applied in practice for diffusion problems with strongly varying coefficients [4, 8]. A common choice is $\sigma = 10$ or $\sigma = 20$.

Illustration To illustrate why the lower bounds (5) are unpractical for problems with strongly varying coefficients, consider the one-dimensional diffusion problem $-(Ku')' = 0$ on the domain $[0, 1]$ with a large jump in the diffusion coefficient:

$$K(x) = \begin{cases} 1, & \text{for } x \leq \frac{1}{2}, \\ 0.001, & \text{else.} \end{cases} \tag{6}$$

For this problem, the penalty parameter σ needs to be chosen close to 10 000 according to (5), which would lead to an inconveniently large condition number of the coefficient matrix. The question is whether this is really necessary: when the domains $[0, \frac{1}{2})$ and $[\frac{1}{2}, 1]$ are considered separately, a value close to $\sigma = 10$ is reasonable in the first domain, and a value close to $\sigma = 0.01$ is suitable in the second. This reasoning advocates to apply (5) using *local* values of the diffusion coefficient (e.g. $\sigma = 10K$) instead of global ones (e.g. $\sigma = 10\,000$). Indeed, we demonstrate in Section 3 later on that this local adaptation to the underlying physics leads to smaller condition numbers and faster convergence of both the SIPG and CG method.

2 Preconditioning

2.1 Introducing the original spectral two-level preconditioner

Coarse correction operator To solve the SPD linear systems discussed in the previous section, we focus on the CG method in combination with the uniform spectral two-level preconditioner introduced by Dobrev et al. [4]. This preconditioner is defined in terms of a coarse correction operator $Q \approx A^{-1}$ that switches from the test space V to a coarse subspace, then performs a correction that is now simple in this coarse space, and finally switches back to the original test space. In this case, the coarse subspace consists of all piecewise constants. More specifically, the coarse correction operator Q reads:

$$Q := \underbrace{R^T}_{\text{prolongation}} \quad \underbrace{A_0^{-1}}_{\text{coarse correction}} \quad \underbrace{R}_{\text{restriction}},$$

where the so-called *restriction* operator R is defined such that $A_0 := RAR^T$ is the SIPG matrix with polynomial degree zero. The matrix A_0 is also referred to as the *Galerkin matrix*, or *coarse matrix*.

Matrix example For example, for a two-dimensional Laplace problem with $p = 1$, a uniform Cartesian mesh with 2×2 elements, and penalty parameter $\sigma = 10$ we have:

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

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

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

Observe that 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 $\phi_1^{(i)}$ are in any monomial 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.

Spectral two-level preconditioner Now that the coarse correction operator Q has been defined, we can formulate the spectral two-level preconditioner. The result $\boldsymbol{y} = \mathcal{M}_{\text{prec}}\boldsymbol{r}$ of applying this preconditioner to a vector $\boldsymbol{r}$ can be computed in three steps:

$$\begin{aligned} \boldsymbol{y}^{(1)} &:= M^{-1}\boldsymbol{r} && \text{(pre-smoothing)}, \\ \boldsymbol{y}^{(2)} &:= \boldsymbol{y}^{(1)} + Q(\boldsymbol{r} - A\boldsymbol{y}^{(1)}) && \text{(coarse correction)}, \\ \boldsymbol{y} &:= \boldsymbol{y}^{(2)} + M^{-T}(\boldsymbol{r} - A\boldsymbol{y}^{(2)}) && \text{(post-smoothing)}, \end{aligned} \tag{7}$$

where $M^{-1} \approx A^{-1}$ is an invertible smoother, for which we typically use block Jacobi. The preconditioning operator $\mathcal{M}_{\text{prec}}$ can also be written explicitly as (cf. e.g. [11]):

$$\mathcal{M}_{\text{prec}} = M(M + M^T - A)^{-1}M^T + (I - M^{-T}A)Q(I - AM^{-1}). \tag{8}$$

For a large class of problems, it can be shown that this preconditioner is uniform [4], assuming that $M + M^T - A$ is SPD. The same requirement implies that the operator $\mathcal{M}_{\text{prec}}$ is SPD [12].

2.2 Switching to deflation

Spectral two-level deflation There are two relevant drawbacks of the spectral two-level preconditioner: the first is that *two* smoothing steps are required. The second is that the smoother must be chosen such that $M + M^T - A$ is SPD. Unfortunately, for large problems, it is usually not easy to verify this requirement. Furthermore, it is typically *not* satisfied for standard Jacobi smoothing. To eliminate one of the two smoothing steps and the inconvenient restriction on the smoother at the same time, we propose to cast the spectral two-level preconditioner into the deflation framework. This is achieved by skipping the last smoothing step in (7). In other words, the result $\boldsymbol{y} = \mathcal{M}_{\text{defl}}\boldsymbol{r}$ of applying the spectral two-level deflation technique to a vector $\boldsymbol{r}$ can be computed as:

$$\begin{aligned} \boldsymbol{y}^{(1)} &:= M^{-1}\boldsymbol{r} && \text{(pre-smoothing)}, \\ \boldsymbol{y} &:= \boldsymbol{y}^{(1)} + Q(\boldsymbol{r} - A\boldsymbol{y}^{(1)}) && \text{(coarse correction)}. \end{aligned} \tag{9}$$

The operator $\mathcal{M}_{\text{defl}}$ can also be written explicitly as (cf. [11]):

$$\mathcal{M}_{\text{defl}} = P^T M^{-1} + Q, \qquad P := I - AQ.$$

Asymmetric implementation The operator $\mathcal{M}_{\text{defl}}$ is not symmetric in general. Interestingly, it can still be implemented successfully in a PCG algorithm in its current asymmetric form, for *any* SPD smoothing operator M^{-1}, as long as the starting vector $\boldsymbol{x}_0$ is preprocessed in one cheap step:

$$\boldsymbol{x}_0 \mapsto Q\boldsymbol{b} + P^T\boldsymbol{x}_0. \tag{10}$$

Indeed, it was shown in [11] that the standard PCG algorithm with (10) produces the same iterates when using either one of the following two preconditioning operators:

1. $\mathcal{M}_{\text{defl}}$,
2. $\mathcal{M}_{\text{BNN}} := P^T M^{-1} P + Q$, which is SPD as long as the coarse matrix A_0 and the smoother M^{-1} are SPD. The latter is well-known and can be shown using e.g. the more abstract analysis in [12].

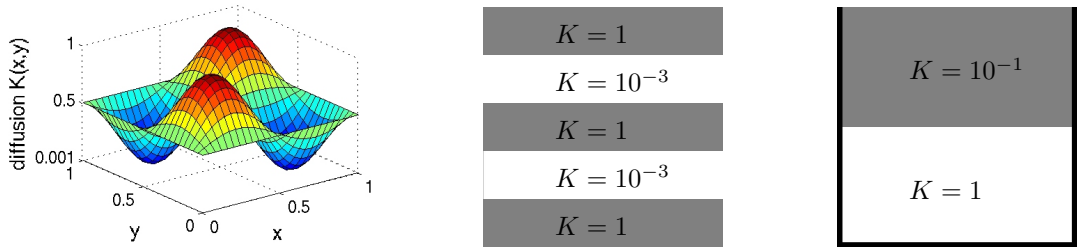
This explains why the deflation operator $\mathcal{M}_{\text{defl}}$ does not need symmetry.

Flops Now that both spectral two-level methods have been defined, it is interesting to compare them in terms of computational costs. To this end, we consider a two-dimensional diffusion problem on a uniform Cartesian mesh with $N = n^2$ mesh elements. Furthermore, we define $m := \frac{(p+1)(p+2)}{2}$, where p is the polynomial degree. Using the spectral two-level preconditioner, the CG method then requires per iteration $(30m^2+14m)N$ flops, plus the costs for two smoothing steps and one coarse solve. Using the spectral two-level deflation method, the CG method requires per iteration $(20m^2 + 12m)N$ flops, plus the costs for one smoothing step and one coarse solve. This is significantly cheaper, especially when the smoothing costs are high.

3 Numerical experiments

3.1 Experimental setup

Model setup To validate the performance of the proposed deflation technique and diffusion-dependent penalty parameter, we consider four test cases, which are defined and illustrated in Figure 2. Three of these problems have strong variations in the coefficients, resulting in ill-conditioned systems that are challenging to solve. Similar problems were studied in [4, 8, 13].



(a) Smooth problem (b) Five layers (c) Bowl

Figure 2: Test cases: we consider four diffusion problems $-\nabla \cdot (K\nabla u) = f$ on the domain $[0,1]^2$: a smooth problem ($K(x,y) = 0.5005 + 0.4995\sin(2\pi x)\sin(2\pi y)$), a problem with five layers (of the same thickness), and a problem with two layers (of the same thickness) in a 'bowl'. For comparison, we also consider a standard Poisson problem ($K = 1$). Homogeneous Neumann boundary conditions are applied at the black edges for the bowl problem. Dirichlet boundary conditions are applied everywhere else, and these and the source term f are chosen such that the exact solution reads $u(x,y) = \cos(2\pi x)\cos(a\pi y)$, where $a = 5$ for problem with five layers, and $a = 2$ for the other cases.

SIPG Setup All model problems are discretized by means of the SIPG method as discussed in Section 1. We use a uniform, Cartesian, lexicographically ordered mesh with $n \times n$ elements, where $n = 10, 20, 40, 80$. Furthermore, we use monomial basis functions with polynomial degree up to $p = 1, 2, 3$. The penalty parameter is chosen diffusion-dependent, $\sigma = 20K$ (using the largest limit value of K at the location of the discontinuities), as motivated in Section 1.2. Although it is common to use only one value for σ per edge in the mesh, we choose to let σ follow the diffusion coefficient naturally along the edge. For comparison, we also consider a common constant value, $\sigma = 20$.

CG Setup The linear systems resulting from the SIPG discretizations are solved by means of the (deflated) PCG method as discussed in Section 2. Besides the original spectral two-level preconditioner (referred to as *TL prec.*) and the proposed deflation variant (*TL defl.*), we also consider standard diagonal preconditioning (*Diagonal prec.*) and basic Block Jacobi (BJ) preconditioning with blocks of order $m := \frac{(p+1)(p+2)}{2}$. Block Jacobi is also used for the smoothing operator M^{-1} in

both spectral two-level methods: twice for the preconditioning variant, and once for the deflation variant. Coarse systems, involving the SIPG matrix A_0 with polynomial degree $p = 0$, are solved directly. However, we also provide a more efficient coarse solution strategy in Section 3.2 below. Diagonal scaling is applied as a preprocessing step in all cases, and the same random start vector $\boldsymbol{x}_0$ is used for all problems of the same size. Preprocessing of the starting vector (10) is applied for the deflation technique only. For the stopping criterion we use:

$$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{b}\|_2} \leq \text{TOL}, \tag{11}$$

where $\text{TOL} = 10^{-7}$, and $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ is the residual after the k^{th} iteration. Finally, we measure the condition number of the (diagonally-scaled) matrix A as the ratio between the largest and the smallest eigenvalue.

3.2 Numerical results

Constant penalty parameter For a constant penalty parameter $\sigma = 20$, the outcome of the numerical experiments specified in the previous section are displayed in Table 1. For the basic Poisson problem, as expected, the convergence for the two basic preconditioners slows down rapidly for larger meshes. At the same time, both spectral two-level methods yield fast uniform convergence, independent of the mesh element diameter (for sufficiently fine meshes). Interestingly, for the large problems, the deflation variant requires fewer iterations, even though it is less expensive due to lower smoothing costs, and more practical due to the absence of restrictions on the smoother. For the other test cases, for which the coefficients vary strongly, the results are more or less similar, but more extreme: the number of iterations is typically much larger than for the Poisson problem. Additionally, it can be seen that the spectral two-level deflation method can converge up to six times faster than the preconditioning variant.

Diffusion-dependent penalty parameter Table 2 displays the result of repeating the experiments in Table 1 for a diffusion-dependent penalty parameter $\sigma = 20K$. This local adjustment to the underlying phyiscs leads to smaller condition numbers and significantly faster convergence of the CG method. The numbers of iterations are now of the same order as those for the Poisson problem (cf. Table 1). Furthermore, uniform convergence of both spectral two-level methods is clearly established in all cases. As before, the deflation variant tends to converge faster for large problems, and even when the deflation variant requires more iterations, we have found that it is still faster in terms of the total computational time in seconds, due to its lower smoothing costs. Altogether, the combination of the spectral two-level deflation method and a diffusion-dependent penalty parameter can make the CG method over 100 times faster, compared to the original preconditioning variant together with a common constant penalty parameter (cf. Figure 1).

Although the use of a diffusion-dependent penalty parameter has been shown to be an effective strategy to increase the efficiency of the CG method, it still needs to be verified that this approach does not reduce the accuracy the SIPG discretization. In Table 3 (also cf. Figure 1 for an illustration) it can be seen that a diffusion-dependent penalty parameter actually leads to better SIPG convergence, both in order and in absolute value.

Efficient coarse solves To obtain the results in Table 1 and Table 2, a direct solver was used for the coarse systems with coefficient matrix A_0. In practice, this is usually not feasible, since A_0 is a large $N \times N$ matrix, where N is the number of mesh elements. For this reason, we have investigated the cheaper alternative of applying the CG method again in an inner loop with the standard incomplete Cholesky preconditioner without fill-in. The results are displayed in Table 4: this table displays the number of outer iterations required for converge of the CG method with the spectral two-level deflation technique, using a diffusion-dependent penalty parameter. Both the outer and the inner loop use stopping criterion (11). For the inner loop, several values for TOL are considered. For comparison, the results for a direct coarse solver are also displayed. The

maximum number of iterations is set to 300. We observe that low accuracy in the inner loop is sufficient for high accuracy in the outer loop: the inner tolerance can be 10^5 times as large as the outer tolerance.

References

[1] P. F. Antonietti and B. Ayuso. 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] S. C. Brenner and J. Zhao. Convergence of multigrid algorithms for interior penalty methods. *Appl. Numer. Anal. Comput. Math.*, 2(1):3–18, 2005.

[3] P. Castillo. Performance of discontinuous Galerkin methods for elliptic PDEs. *SIAM J. Sci. Comput.*, 24(2):524–547, 2002.

[4] V. A. Dobrev, R. D. Lazarov, P. S. Vassilevski, and L. T. Zikatanov. Two-level preconditioning of discontinuous Galerkin approximations of second-order elliptic equations. *Numer. Linear Algebra Appl.*, 13(9):753–770, 2006.

[5] Y. Epshteyn and B. Rivière. Estimation of penalty parameters for symmetric interior penalty Galerkin methods. *J. Comput. Appl. Math.*, 206(2):843–872, 2007.

[6] K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 207(1):92–113, 2005.

[7] F. Prill, M. Lukáčová-Medviďová, and R. Hartmann. Smoothed aggregation multigrid for the discontinuous Galerkin method. *SIAM J. Sci. Comput.*, 31(5):3503–3528, 2009.

[8] J. Proft and B. Rivière. Discontinuous Galerkin methods for convection-diffusion equations for varying and vanishing diffusivity. *Int. J. Numer. Anal. Model.*, 6(4):533–561, 2009.

[9] B. Rivière. *Discontinuous Galerkin methods for solving elliptic and parabolic equations*, volume 35 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and implementation.

[10] P. van Slingerland and C. Vuik. Spectral two-level deflation for DG: a preconditioner for CG that does not need symmetry. Technical Report 11-12, Delft University of Technology, 2011.

[11] J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga. Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *J. Sci. Comput.*, 39(3):340–370, 2009.

[12] P. S. Vassilevski. *Multilevel block factorization preconditioners*. Springer, New York, 2008. Matrix-based analysis and algorithms for solving finite element equations.

[13] C. Vuik, A. Segal, and J.A. Meijerink. An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients. *Journal of Computational Physics*, 152:385–403, 1999.

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	2.1e+03	8.4e+03	3.2e+04	1.2e+05	5.3e+03	1.8e+04	6.7e+04	2.6e+05	7.0e+03	2.4e+04	8.6e+04	3.3e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	103	221	435	826	190	359	684	1198	215	385	667	1330
Block Jacobi (BJ)	103	221	435	826	120	233	428	779	122	229	432	805
TL prec., 2x BJ	31	37	39	40	39	42	44	45	45	58	61	62
TL defl., 1x BJ	36	41	42	43	36	38	39	39	39	41	42	43

(a) Poisson problem, constant $\sigma = 20$

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	3.9e+03	1.7e+04	7.1e+04	3.0e+05	1.4e+04	6.4e+04	2.6e+05	1.0e+06	2.7e+04	1.0e+05	4.1e+05	1.5e+06
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	157	336	714	1311	337	677	1262	2432	419	766	1427	2514
Block Jacobi (BJ)	157	336	714	1311	224	424	823	1466	232	477	833	1482
TL prec., 2x BJ	41	57	83	116	82	147	275	487	102	214	375	615
TL defl., 1x BJ	51	76	107	147	108	205	350	523	128	240	416	594

(b) Smooth problem, constant $\sigma = 20$

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	4.5e+03	2.8e+04	2.2e+05	2.2e+06	3.4e+05	1.6e+06	5.7e+06	2.1e+07	8.7e+05	2.6e+06	7.9e+06	2.7e+07
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	165	392	1125	2773	600	2218	5916	12249	1000	3353	6684	12630
Block Jacobi (BJ)	165	392	1125	2772	425	1157	3007	6905	692	1785	3999	7710
TL prec., 2x BJ	51	91	188	348	186	490	1471	3022	504	1316	2603	5229
TL defl., 1x BJ	61	127	273	462	152	276	461	598	365	547	769	864

(c) Five layers, constant $\sigma = 20$

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	1.2e+05	4.8e+05	1.9e+06	7.6e+06	2.2e+05	8.6e+05	3.4e+06	1.4e+07	3.1e+05	1.2e+06	4.8e+06	1.9e+07
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	300	763	1640	2967	600	1339	2618	4753	764	1430	2739	5200
Block Jacobi (BJ)	273	683	1447	2635	417	845	1558	2906	423	800	1460	2857
TL prec., 2x BJ	84	116	125	134	146	196	227	238	191	295	408	517
TL defl., 1x BJ	90	111	120	129	106	112	116	119	123	126	129	131

(d) Bowl, constant $\sigma = 20$

Table 1: Comparison of preconditioning strategies in terms of the number of CG iterations required for convergence using a constant penalty parameter $\sigma = 20$.

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	2.5e+03	1.0e+04	3.9e+04	1.5e+05	6.1e+03	2.1e+04	7.6e+04	2.9e+05	8.9e+03	2.9e+04	1.1e+05	4.0e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	122	236	461	889	206	400	721	1362	237	410	729	1393
Block Jacobi (BJ)	116	239	469	885	130	248	438	845	129	244	446	847
TL prec., 2x BJ	32	38	40	41	40	43	44	45	46	56	62	63
TL defl., 1x BJ	36	41	43	44	38	39	39	39	40	41	43	43

(a) Smooth problem, diffusion-dependent $\sigma = 20K$

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	3.1e+04	4.5e+04	7.6e+04	2.5e+05	1.6e+05	1.7e+05	2.3e+05	5.1e+05	3.0e+05	2.8e+05	3.3e+05	6.7e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	300	690	948	1264	600	1247	1469	1876	1000	1665	1913	2317
Block Jacobi (BJ)	123	249	485	883	144	259	490	932	144	255	492	870
TL prec., 2x BJ	35	41	42	42	46	52	49	49	49	62	64	65
TL defl., 1x BJ	43	46	51	52	51	51	54	54	53	56	57	58

(b) Five layers, diffusion-dependent $\sigma = 20K$

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	9.5e+04	3.7e+05	1.5e+06	5.9e+06	1.8e+05	7.2e+05	2.8e+06	1.1e+07	2.5e+05	9.7e+05	3.9e+06	1.5e+07
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	233	454	895	1654	366	633	1222	2423	434	735	1443	2729
Block Jacobi (BJ)	194	394	779	1446	219	415	785	1514	214	423	795	1496
TL prec., 2x BJ	41	44	45	45	52	51	52	52	63	67	67	68
TL defl., 1x BJ	49	50	55	56	50	53	54	54	56	57	57	58

(c) Bowl, diffusion-dependent $\sigma = 20K$

Table 2: Comparison of preconditioning strategies in terms of the number of CG iterations required for convergence using a diffusion-dependent penalty parameter $\sigma = 20K$.

	p=1		p=2		p=3	
mesh	error	order	error	order	error	order
$N = 10^2$	4.12e-01	-	9.36e-02	-	9.47e-03	-
$N = 20^2$	2.48e-01	0.73	2.32e-02	2.01	1.20e-03	2.98
$N = 40^2$	1.54e-01	0.69	4.90e-03	2.25	1.13e-04	3.40
$N = 80^2$	1.10e-01	0.48	6.91e-04	2.82	7.50e-06	3.92

(a) Five layers, fixed $\sigma = 20$

	p=1		p=2		p=3	
mesh	error	order	error	order	error	order
$N = 10^2$	3.02e-01	-	1.93e-02	-	1.90e-03	-
$N = 20^2$	1.15e-01	1.40	1.92e-03	3.33	1.16e-04	4.04
$N = 40^2$	3.43e-02	1.74	2.13e-04	3.17	7.11e-06	4.02
$N = 80^2$	9.12e-03	1.91	2.55e-05	3.06	4.42e-07	4.01

(b) Five layers, diffusion-dependent $\sigma = 20K$

Table 3: SIPG convergence for both a constant and diffusion-dependent penalty parameter: a diffusion-dependent penalty parameter yields better SIPG accuracy, both in order and in absolute value.

degree	p=1				p=2				p=3			
mesh	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=10^2$	$N=20^2$	$N=40^2$	$N=80^2$
condition number	3.1e+04	4.5e+04	7.6e+04	2.5e+05	1.6e+05	1.7e+05	2.3e+05	5.1e+05	3.0e+05	2.8e+05	3.3e+05	6.7e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
exact	43	46	51	52	51	51	54	54	53	56	57	58
TOL $= 10^{-4}$	43	46	51	52	51	51	54	54	53	56	57	58
TOL $= 10^{-3}$	43	47	50	53	51	51	54	54	53	56	57	58
TOL $= 10^{-2}$	44	47	53	55	51	51	53	55	53	56	56	58
TOL $= 10^{-1}$	300	300	300	300	68	81	118	300	67	79	93	141

(a) Five layers, inexact coarse solves, $\sigma = 20K$

Table 4: Solving coarse systems for the spectral two-level deflation technique more efficiently by applying CG again in an inner loop with a standard IC preconditioner: low accuracy in the inner loop is sufficient for high accuracy in the outer loop.