DELFT UNIVERSITY OF TECHNOLOGY

REPORT 11-12

Spectral two-level deflation for DG: a preconditioner for CG that does not need symmetry

P. van Slingerland, and C. Vuik

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2011

Spectral two-level deflation for DG: a preconditioner for CG that does not need symmetry

P. van Slingerland and C. Vuik

September 15, 2011

Abstract

Despite their geometric flexibility and high accuracy, an important drawback of Discontinuous Galerkin (DG) methods is that the resulting linear systems cannot be solved efficiently by standard iterative solution methods. In search of an effective preconditioning strategy for these systems, we cast a uniform spectral two-level preconditioner proposed by Dobrev et al. [6] into the deflation framework [17]. This report discusses how the resulting spectral two-level deflation technique can be implemented in a Conjugate Gradient (CG) algorithm, and studies its performance for Symmetric Interior Penalty Galerkin (SIPG) discretizations for diffusion problems with extreme contrasts in the coefficients. Furthermore, it investigates the influence of the SIPG penalty parameter, which is not well-understood for such applications. Interestingly, we find that the proposed deflation technique essentially follows from the original preconditioning variant by skipping one of the two smoothing steps involved, and dropping the inconvenient restriction on the smoother which the original preconditioner required to be SPD. Despite these simplifications, the spectral two-level deflation method yields uniform convergence (independent of the mesh element diameter), and it is faster and more robust than the original preconditioner for large problems. Regarding the penalty parameter, we find that it can best be based on *local* values of the diffusion coefficient, instead of the usual strategy to use one *global* constant for the entire domain. The former approach allows for local minimization of the penalty parameter, resulting in smaller condition numbers and faster convergence of both the SIPG and the CG method.

1 Introduction

Despite their geometric flexibility and high accuracy, the popularity of Discontinuous Galerkin (DG) discretizations for elliptic problems [3] has been limited for a while. The primary reason for this is that the corresponding coefficient matrix is relatively large compared to e.g. traditional finite element methods. Moreover, this matrix is usually ill-conditioned, as its condition number tends to increase for a smaller mesh element diameter, a higher polynomial degree, or a larger stabilization factor [5, 16]. Altogether, standard iterative solution methods converge rather slowly for linear systems resulting from DG discretizations.

However, the interest in DG methods was increased with the need for handling non-matching grids and designing hp-refinement strategies, for which they are particularly suitable. For this reason, much attention has been paid since to subspace correction methods [21]. For example, Schwarz domain decomposition methods are based on subspaces that arise from subdividing the spatial domain into smaller subdomains [1, 8]; geometric (h-)multigrid methods make use of subspaces resulting from multiple coarser meshes [4, 10]; spectral (p-)multigrid methods apply global corrections by solving problems with a lower polynomial degree [9, 11]; and algebraic multigrid methods use algebraic criteria to separate the unknowns of the original system into two sets, one of which is labelled 'coarse' [12, 15].

Usually, these methods can either be used as a standalone solver, or as a preconditioner in an iterative Krylov method. The second strategy tends to be more robust for problems with a few isolated 'bad' eigenvalues. The latter is common for diffusion problems with extreme contrasts in the coefficients, such as those encountered in oil reservoir simulations [19]. This research focuses on such problems, discretized by means of the Symmetric Interior Penalty Galerkin (SIPG) method [14]. This DG method yields a Symmetric and Positive-Definite (SPD) coefficient matrix, as long as its characterizing penalty parameter is sufficiently large, which is also required for convergence. This makes the Conjugate Gradient (CG) method particularly suitable for solving the corresponding linear systems. *Altogether, this research is focused on an effective preconditioning strategy to increase the efficiency of the CG method for linear systems resulting from SIPG discretizations for diffusion problems with extreme contrasts in the coefficients.*

Starting point of this research is one of the spectral two-level methods proposed by Dobrev et al. [6]. This method is based on the idea of spectral multigrid, except that it uses only two levels: at the coarse level, it applies a correction that is based on the SIPG discretization with polynomial degree $p = 0$. Dobrev et al. showed that this method, and thus the resulting preconditioner, yields uniform convergence (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 applying this spectral two-level preconditioner in a CG algorithm for a SIPG matrix A: First, *two* smoothing steps must be applied during each iteration, and the smoother $M^{-1} \approx A^{-1}$ must be chosen such that $M + M^T - A$ is SPD. The latter is not easily verified for large practical problems. Furthermore, it is often not satisfied if the smoother is the standard Jacobi smoother.

The second issue is that the influence of the penalty parameter 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 suffciently large to ensure that A is SPD and the SIPG method converges. At the same time, it needs to be chosen as small as possible to avoid an unnecessarily large condition number. Computable theoretical lower bounds for the penalty parameter that ensure stability and convergence have been derived for a large variety of problems by Epshteyn and Riviere [7]. However, these bounds are based on the ratio between the *global* maximum and minimum of the diffusion coefficient, and are therefore impractical for our application.

To eliminate one of the two smoothing steps and the inconvenient restriction on the smoother at the same time, we cast the spectral two-level preconditioner into the deflation framework. This is achieved by using the analysis of Tang at al. [17], who considered from an abstract point of view the relations between two-level PCG methods coming from the fields of deflation, domain decomposition and multigrid. Besides the resulting spectral two-level deflation technique, we also study the potential of a penalty parameter that is based on *local* values of the diffusion coefficient.

The outline of this report is as follows. Section 2 discusses the SIPG method and the resulting linear systems for diffusion problems. Furthermore, it studies the influence of the penalty parameter on the properties of the coefficient matrix. Section 3 discusses the spectral two-level preconditioner and transforms it into a deflation technique. Moreover, it considers the influence of the penalty parameter on the coarse level accuracy of these methods. Section 4 compares the numerical performance of the resulting spectral two-level deflation method to that of the original preconditioning variant. Additionally, it examines the influence of the penalty parameter on the convergence speed of both the SIPG and the CG method. Finally, Section 5 summarizes the main conclusions.

2 SIPG method

DG methods are flexible discretization schemes that provide high-order solution approximations for PDEs. This section discusses the SIPG method (Section 2.1) and the resulting linear systems (Section 2.2) for diffusion problems. Moreover, it investigates the influence of the penalty parameter on the coefficient matrix (Section 2.3).

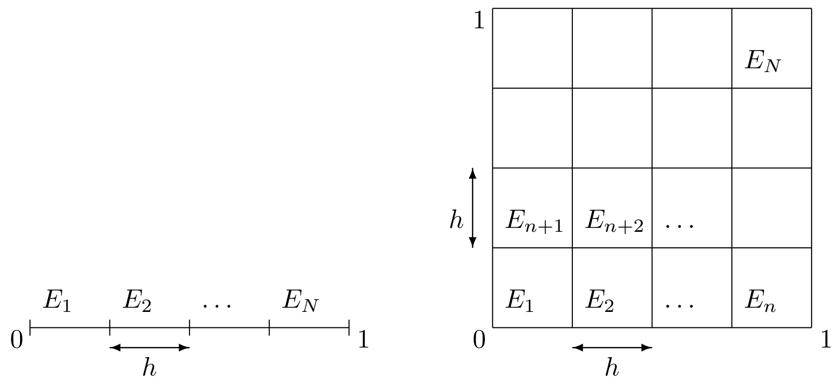


Figure 1: We use a uniform Cartesian mesh for $\Omega = [0,1]^d$ with $N = n^d$ elements and spacing $h = \frac{1}{n}$. However, our solver can be applied to a wider range of problems.

2.1 Formulating the SIPG method for diffusion problems

DG methods usually assume that the solution approximation is a polynomial of degree p or lower within each mesh element, and that it may be discontinuous at the element boundaries. Although these discontinuities provide a convenient flexibility, they also tend to cause instability for second order problems. To enforce weak continuity and ensure stability, the SIPG method, introduced by Wheeler [20] and Arnold [2], penalizes the inter-element discontinuities. This section briefly summarizes known literature regarding this method for diffusion problems. More details can also be found in e.g. [3, 14]. The SIPG method converges at rate $p+1$ for a large class of problems, provided that the penalty parameter is sufficiently large.

Model problem We study the following diffusion problem on a domain Ω with outward normal $\boldsymbol{n}$, boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, source term $f \in L^2(\Omega)$, scalar diffusion coefficient $K \in L^\infty(\Omega)$ (bounded below and above by a positive constant), Dirichlet data $g_D \in H^{\frac{1}{2}}(\partial\Omega_D)$, and Neumann data $g_N \in L^2(\partial\Omega_N)$ (cf. [14] for the definition of the function spaces):

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

Mesh To construct a SIPG approximation u_h for the exact solution u in (1), we need to subdivide the domain Ω into mesh elements $E_1, ..., E_N$. In this report, we consider uniform Cartesian meshes for $\Omega = [0,1]^d$ with $N = n^d$ elements and spacing $h = \frac{1}{n}$ (cf. Figure 1 for the element ordering). Nonetheless, our solver can be applied for other meshes as well. Furthermore, we will use the following notation: let $\boldsymbol{n}_i$ denote the outward normal of the mesh element E_i, let Γ_h denote the collection of all interior edges $e = \partial E_i \cap \partial E_j$ in the mesh shared by two adjacent elements, and let Γ_D denote the collection of all Dirichlet boundary edges $e = \partial E_i \cap \partial\Omega_D$.

Test space and trace operators The SIPG approximation u_h is sought in a 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. To deal with these discontinuities, we need to introduce trace operators for jumps and averages at the mesh element boundaries: at each interior element boundary $\partial E_i \cap \partial E_j \in \Gamma_h$, we define:

$$[v] := v_i\boldsymbol{n}_i + v_j\boldsymbol{n}_j, \qquad \{v\} := \frac{v_i + v_j}{2}, \tag{2}$$

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	...
	$p=0$	$p=1$		$p=2$			$p=3$				...

Table 1: Powers k_x and k_y as a function of k. These values are used to define two-dimensional monomial basis functions

where v_i denotes the trace of the function v along the side of E_i. The trace operators are well-defined if v is in the broken Sobolev space, i.e. $v|_{E_k} \in H^1(E_k)$ for each mesh element E_k. Observe that $[v]$ is a vector, while v is a scalar. Analogously, we define for a vector-valued function $\boldsymbol{\tau}$:

$$[\boldsymbol{\tau}] := \boldsymbol{\tau}_i \cdot \boldsymbol{n}_i + \boldsymbol{\tau}_j \cdot \boldsymbol{n}_j, \qquad \{\boldsymbol{\tau}\} := \frac{1}{2}(\boldsymbol{\tau}_i + \boldsymbol{\tau}_j).$$

Observe that $[\boldsymbol{\tau}]$ is a scalar, while $\boldsymbol{\tau}$ is a vector. Similarly, at the domain boundary, we define at each element boundary $\partial E_i \cap \partial\Omega \in \Gamma_D$:

$$[v] := v_i \boldsymbol{n}_i, \qquad \{v\} := v_i, \qquad [\boldsymbol{\tau}] := \boldsymbol{\tau}_i \cdot \boldsymbol{n}_i, \qquad \{\boldsymbol{\tau}\} := \boldsymbol{\tau}_i. \tag{3}$$

Bilinear form Now that the required notation has been introduced, the SIPG approximation can be defined as the function u_h in the test space V that satisfies:

$$B(u_h, v) = L(v), \qquad \text{for all test functions } v \in V, \tag{4}$$

with (for one-dimensional problems, the boundary integrals below should be interpreted as function evaluations of the integrand):

$$B(u_h, v) = \int_\Omega K\nabla u_h \cdot \nabla v + \sum_{e \in \Gamma_h \cup \Gamma_D} \int_e \Big(-\{K\nabla u_h\} \cdot [v] - [u_h] \cdot \{K\nabla v\} + \frac{\sigma}{h} [u_h] \cdot [v] \Big), \tag{5}$$

$$L(v) = \int_\Omega fv - \sum_{e \in \Gamma_D} \int_e \Big([K\nabla v] + \frac{\sigma}{h} v \Big) g_D + \sum_{e \in \Gamma_N} \int_e v g_N, \tag{6}$$

where σ is the so-called penalty parameter. This parameter penalizes the inter-element jumps to ensure stability and enforce weak continuity. Although it is presented as a constant here, its value may vary throughout the domain. For a large class of problems, the SIPG method is stable and converges with order $p+1$ (where p is the polynomial degree), as long as the penalty parameter σ is large enough [14]. Suitable choices for the penalty parameter are discussed later in Section 2.3.

2.2 Computing the SIPG solution by solving a linear system

In order to compute the SIPG approximation defined by (4) in the previous section, it needs to be rewritten as a linear combination of basis functions for the polynomial test space, in which the coefficients can be determined by solving a linear system. This section briefly summarizes the derivation and properties of this linear system. More details can be found in e.g. [14]. The corresponding coefficient matrix is SPD if the penalty parameter is sufficiently large, which is also required for convergence. As a consequence, the CG method is a suitable candidate for solving the linear systems resulting from SIPG discretizations.

Monomial basis functions To rewrite the SIPG approximation, we start by choosing a basis for the test space V. For the one-dimensional domain $\Omega = [0, 1]$, we use monomial basis functions $\phi_k^{(i)}$, which are zero in the entire domain, except in the element E_i with center x_i, where we define:

$$\phi_k^{(i)}(x) = \left(\frac{x - x_i}{\frac{1}{2}h} \right)^{k-1}, \qquad \text{for all } i = 1, ..., N, \text{ and } k = 1, ..., M := p+1. \tag{7}$$

Similarly, for the two-dimensional domain $\Omega = [0,1]^2$, we define within the element E_i with center (x_i, y_i):

$$\phi_k^{(i)}(x,y) = \left(\frac{x - x_i}{\frac{1}{2}h}\right)^{k_x} \left(\frac{y - y_i}{\frac{1}{2}h}\right)^{k_y}, \quad \text{for all } i = 1, ..., N, \text{ and } k = 1, ..., M := \frac{(p+1)(p+2)}{2}, \tag{8}$$

where k_x and k_y are selected as indicated in Table 1.

Linear system Now that we have a basis for the test space V, we can rewrite (4) as a linear system. To this end, we express the SIPG solution $u_h \in V$ as a linear combination of basis functions $\phi_k^{(i)}$:

$$u_h = \sum_{i=1}^{N} \sum_{k=1}^{M} u_k^{(i)} \phi_k^{(i)}, \tag{9}$$

where the coefficients $u_k^{(i)}$ can be determined by substituting the expression (9) for u_h and the basis functions $\phi_\ell^{(j)}$ for v into (4). After a little rewriting, we then obtain a linear system $A\boldsymbol{u} = \boldsymbol{b}$ of the form:

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & & \vdots \\ \vdots & & \ddots & \\ A_{N1} & \dots & & A_{NN} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_N \end{bmatrix}, \tag{10}$$

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

$$A_{ji} = \begin{bmatrix} B(\phi_1^{(i)}, \phi_1^{(j)}) & B(\phi_2^{(i)}, \phi_1^{(j)}) & \dots & B(\phi_M^{(i)}, \phi_1^{(j)}) \\ B(\phi_1^{(i)}, \phi_2^{(j)}) & B(\phi_2^{(i)}, \phi_2^{(j)}) & & \vdots \\ \vdots & & \ddots & \\ B(\phi_1^{(i)}, \phi_M^{(j)}) & & \dots & B(\phi_M^{(i)}, \phi_M^{(j)}) \end{bmatrix}, \quad \boldsymbol{u}_i = \begin{bmatrix} u_1^{(i)} \\ u_2^{(i)} \\ \vdots \\ u_M^{(i)} \end{bmatrix}, \quad \boldsymbol{b}_j = \begin{bmatrix} L(\phi_1^{(j)}) \\ L(\phi_2^{(j)}) \\ \vdots \\ L(\phi_M^{(j)}) \end{bmatrix}. \tag{11}$$

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 (9).

Matrix properties The coefficient matrix A has the following structure: for polynomial degree $p = 0$, it is an $N \times N$ matrix with the same nonzero pattern as a central difference matrix. As a matter of fact, if the penalty parameter is equal to the diffusion coefficient ($\sigma = K$), then A is *equal* to the central difference matrix. For polynomial degree $p > 0$, it is an $N \times N$ *block* matrix with a similar structure, except that the nonzero elements are now dense blocks, which have size $(p+1) \times (p+1)$ for one-dimensional problems, and size $\frac{(p+1)(p+2)}{2} \times \frac{(p+1)(p+2)}{2}$ for two-dimensional problems (note that $\frac{(p+1)(p+2)}{2}$ is integer for any positive integer p). As a consequence, the size of A increases rapidly with the number of elements N and the polynomial degree p. Finally, the matrix is SPD as long as the penalty parameter σ is sufficiently large, which is also required for convergence. The influence of the penalty parameter is studied in more detail in the next section.

2.3 The effect of the penalty parameter on the coefficient matrix

The penalty parameter has a great influence on the SIPG discretization defined in the previous sections. On the one hand, it 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 [5]. Known theoretical lower bounds are based on the ratio between the *global* maximum and minimum of the diffusion coefficient [7]. As this is unpractical for problems with strongly varying coefficients, we explore the potential of a penalty parameter that is based on *local* values of the diffusion coefficient. This section illustrates the impact on the coefficient matrix for both a common constant penalty parameter and a diffusion-dependent value. We find that the latter strategy yields a better representation of the underlying physics in the matrix coefficients. Furthermore, it allows for local minimization of the penalty parameter. It is verified in Section 4.3 later on that this leads to smaller condition numbers, and faster SIPG and CG convergence.

Computable lower bounds Computable theoretical lower bounds for the penalty parameter that ensure stability and convergence have been derived for a large variety of problems by Epshteyn and Riviere [7]. For one-dimensional diffusion problems, they propose:

$$\begin{aligned} &\sigma \geq \tfrac{k_1^2}{k_0} p^2, && \text{for interior edges,} \\ &\sigma \geq 2\tfrac{k_1^2}{k_0} p^2, && \text{for boundary edges,} \end{aligned} \tag{12}$$

where k_0 and k_1 are the *global* lower and upper bounds for the diffusion coefficient K. 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 [6, 13]. A common choice is $\sigma = 10$ or $\sigma = 20$.

The problem with global values To illustrate why the lower bounds (12) are unpractical for problems with strongly varying coefficients, consider a one-dimensional diffusion problem (1) on the domain $[0, 1]$ with a large jump in the diffusion coefficient:

$$K(x) = \begin{cases} 1, & \text{for } x \leq \frac{1}{2}, \\ 0.001, & \text{else.} \end{cases} \tag{13}$$

For this problem, the penalty parameter σ needs to be chosen close to 10 000 according to (12), 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.

Diffusion-dependent penalty parameter This reasoning advocates to apply (12) using *local* values of the diffusion coefficient (e.g. $\sigma = 10K$) instead of global ones (e.g. $\sigma = 10\,000$). To demonstrate the effect of such a diffusion-dependent penalty parameter on the coefficient matrix, consider the aforementioned one-dimensional diffusion problem (cf. (13)) again with $N = 4$ mesh elements and polynomial degree $p = 1$. If we use the common constant choice, $\sigma = 10$, which is actually still far too small according to (12), the matrix reads:

$$A = 10^{-3} \left[\begin{array}{rr|rr|rr|rr} 80000 & 4000 & -40000 & 36000 & 0 & 0 & 0 & 0 \\ 4000 & 72000 & -36000 & 32000 & 0 & 0 & 0 & 0 \\ \hline -40000 & -36000 & 80000 & 0 & -40000 & 39996 & 0 & 0 \\ 36000 & 32000 & 0 & 80000 & -36000 & 35996 & 0 & 0 \\ \hline 0 & 0 & -40000 & -36000 & 80000 & 0 & -40000 & 39996 \\ 0 & 0 & 39996 & 35996 & 0 & 80000 & -39996 & 39992 \\ \hline 0 & 0 & 0 & 0 & -40000 & -39996 & 80000 & -4 \\ 0 & 0 & 0 & 0 & 39996 & 39992 & -4 & 79992 \end{array} \right].$$

Observe that the jump in the diffusion can hardly be noticed in the matrix coefficients in this case. This is because the penalty parameter is much larger than the smallest value of the diffusion coefficient. On the other hand, if we use a diffusion-dependent value, $\sigma = 10K$ (using the largest value, $K = 1$, at the location of the jump), we obtain:

$$A = 10^{-3} \left[\begin{array}{rr|rr|rr|rr} 80000 & 4000 & -40000 & 36000 & 0 & 0 & 0 & 0 \\ 4000 & 72000 & -36000 & 32000 & 0 & 0 & 0 & 0 \\ \hline -40000 & -36000 & 80000 & 0 & -40000 & 39996 & 0 & 0 \\ 36000 & 32000 & 0 & 80000 & -36000 & 35996 & 0 & 0 \\ \hline 0 & 0 & -40000 & -36000 & 40040 & -39960 & -40 & 36 \\ 0 & 0 & 39996 & 35996 & -39960 & 40040 & -36 & 32 \\ \hline 0 & 0 & 0 & 0 & -40 & -36 & 80 & -4 \\ 0 & 0 & 0 & 0 & 36 & 32 & -4 & 72 \end{array}\right]. \tag{14}$$

The matrix is SPD in both cases, but, this time, the jump in the diffusion is clearly visible. In that respect, the original problem is better represented by the matrix coefficients. For this reason, and for reasons discussed in Section 3.3 and Section 4.3 later on, it is best to use a diffusion-dependent penalty parameter for problems with strongly varying coefficients.

3 Casting the spectral two-level preconditioner into the deflation framework

The linear systems discussed in the previous section can be solved by means of the preconditioned CG method. This section discusses the spectral two-level preconditioner introduced by Dobrev et al. [6] (Section 3.1) and casts it into the deflation framework (Section 3.2). Moreover, it considers the influence of the penalty parameter on the coarse level accuracy of these methods (Section 3.3).

3.1 Formulating the original spectral two-level preconditioner

The condition number of the SIPG coefficient matrix A is typically large for a small mesh element diameter [5]. To avoid that the convergence of the CG method is slower for finer meshes, Dobrev et al. introduced a spectral two-level preconditioner that applies coarse corrections based on the SIPG solution with $p = 0$. This section briefly summarizes the definition and properties of this preconditioner. For a large class of problems, Dobrev et al. showed that the condition number of the preconditioned system is independent of the mesh, resulting in uniform convergence of the PCG method.

Coarse correction operator Spectral two-level methods are 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. Dobrev et al. apply this strategy using the coarse subspace that 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. Observe that Q and A_0 are SPD if A is SPD. 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 mesh with 2×2 elements, and penalty parameter $\sigma = 10$ we have:

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

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

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

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 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. Basically, 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:

$$\begin{aligned}
\boldsymbol{y}^{(1)} &:= M^{-1}\boldsymbol{r} && \text{(pre-smoothing)}, \\
\boldsymbol{y}^{(2)} &:= \boldsymbol{y}^{(1)} + Q(\boldsymbol{r} - A\boldsymbol{y}^{(1)}) && \text{(coarse correction)}, \\
\boldsymbol{y} &:= \boldsymbol{y}^{(2)} + M^{-T}(\boldsymbol{r} - A\boldsymbol{y}^{(2)}) && \text{(post-smoothing)}, \qquad (15)
\end{aligned}$$

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. [17]):

$$\mathcal{M}_{\text{prec}} = M(M + M^T - A)^{-1}M^T + (I - M^{-T}A)Q(I - AM^{-1}). \qquad (16)$$

For a large class of problems, it can be shown that the preconditioner is uniform, assuming that $M + M^T - A$ is SPD [6]. The same requirement implies that the operator $\mathcal{M}_{\text{prec}}$ is SPD [18], which is required for the PCG method.

3.2 Casting the spectral two-level preconditioner into the deflation framework

Despite its uniform convergence, the spectral two-level preconditioner discussed in the previous section has a two relevant drawbacks: 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 cast the spectral two-level preconditioner into the deflation framework. This section discusses how this deflation method can be implemented in a PCG method by applying the analysis by Tang et al. [17]. We find that this can basically be achieved by simply skipping one of the two smoothing steps in the original preconditioning variant. The resulting spectral two-level deflation technique can be applied for any SPD smoothing operator.

Spectral two-level deflation Essentially, we propose to skip the last smoothing step in (15). 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:

$$\begin{aligned} \boldsymbol{y}^{(1)} &:= M^{-1}\boldsymbol{r} && \text{(pre-smoothing)}, \\ \boldsymbol{y} &:= \boldsymbol{y}^{(1)} + Q(\boldsymbol{r} - A\boldsymbol{y}^{(1)}) && \text{(coarse correction)}. \end{aligned} \tag{17}$$

The operator $\mathcal{M}_{\text{defl}}$ can also be written explicitly as (cf. [17]):

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

This operator is not symmetric in general. As such, it seems unsuitable for the *standard* PCG method. Surprisingly, it can still be implemented successfully in a *generalized* PCG algorithm in its current asymmetric form, for *any* SPD smoothing operator M^{-1}. This is explained below. We stress that we have dropped the smoothing criterion that was required for the original preconditioning variant. In other words, unlike the preconditioning variant, the deflation variant allows that we apply no smoothing at all, i.e. that we set $M^{-1} = I$. However, in practice, some smoothing is usually required for fast convergence.

Two-level PCG The spectral two-level deflation technique (17) can be implemented as an asymmetric operator in a PCG algorithm by pre-processing the starting vector $\bar{\boldsymbol{x}}$:

$$\bar{\boldsymbol{x}} \mapsto Q\boldsymbol{b} + P^T\bar{\boldsymbol{x}} =: \mathcal{V}_{\text{start}}. \tag{18}$$

In other words, the generalized PCG algorithm reads, substituting $\mathcal{M}_{\text{defl}}$ for $\mathcal{M}$:

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

Indeed, it was shown in [17] that this algorithm with (18) produces the same iterates when substituting for $\mathcal{M}$ either one of the following three operators (note that $\mathcal{M}_1$ and $\mathcal{M}_2$ are SPD):

- $\mathcal{M}_{\text{defl}}$,

operation	flops (rounded)	# defl.	# prec.
mat-vec (Au)	$10m^2N$	2	3
inner product (u^Tv)	$2mN$	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

Table 2: Comparing the computational costs per CG iteration for A-DEF2 and the spectral preconditioner.

- $\mathcal{M}_1 := P^T M^{-1} P + Q$,
- $\mathcal{M}_2 := P^T M^{-1} P$.

In other words, because of the simple pre-processing step (18), the deflation variant (17) can be applied effectively for arbitrary starting vector $\bar{\boldsymbol{x}}$ and SPD smoothing operator M^{-1}. The costs for the pre-processing are comparable to those for one preconditioning step (17).

Finally, we remark that the original spectral two-level preconditioner (15) can be incorporated in a PCG algorithm in the usual manner: assuming that $M + M^T - A$ is SPD, the algorithm above can be applied for starting vector $\bar{\boldsymbol{x}} =: \mathcal{V}_{\text{start}}$, substituting $\mathcal{M}_{\text{prec}}$ (15) for $\mathcal{M}$. We stress that the resulting iterates are *not* the same as for the deflation variant.

Flops Table 2 compares the computational costs per CG iteration for both spectral two-level methods in terms of FLoating point OPerationS (FLOPS). This table applies to two-dimensional diffusion problems with polynomial degree p, a mesh with N elements, and polynomial space dimension $m := \frac{(p+1)(p+2)}{2}$ (cf. Section 2.1). Using the spectral two-level preconditioner, the CG method requires per iteration $(30m^2 + 14m)N$ flops, plus the costs for two (non-trivial) smoothing steps, and plus the costs for 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 (optional) (pre-)smoothing step, and plus the costs for one coarse solve. This is significantly cheaper, especially when the smoothing costs are high. A block Jacobi smoothing step with blocks of size m requires $(2m^2 - m)N$ flops.

3.3 Studying the influence of the penalty parameter on the coarse level accuracy

It can be expected that the spectral two-level methods defined in the previous section are only effective if the coarse solution, i.e. the SIPG solution with $p = 0$, is sufficiently accurate. To gain insight in the coarse level accuracy, we compute the SIPG solution with $p = 0$ for both a constant and a diffusion-dependent penalty parameter. This section discusses the results and the implications for the performance of the spectral two-level methods. We observed that, although the SIPG method converges at rate $p + 1$ for $p > 0$, the accuracy for $p = 0$ is very limited and depends strongly on the penalty parameter. For a diffusion-dependent penalty parameter, the underlying physics is captured better in the shape of the solution than for a constant penalty parameter. This is valuable coarse level information for the spectral two-level methods. Later on, Section 4.3 demonstrates that the spectral two-level deflation method yields faster CG convergence for a diffusion-dependent penalty parameter than for a constant value. However, fast uniform convergence is obtained in either case.

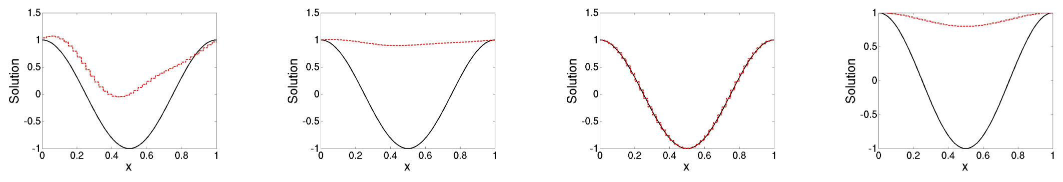
Coarse level accuracy for a constant penalty parameter Figure 2 displays the SIPG approximation with polynomial degree $p = 0$ for a one-dimensional diffusion problem (1) with

Dirichlet boundary conditions, exact solution $u(x) = \cos(2\pi x)$, and a diffusion coefficient that varies smoothly between 0.001 and 1:

$$K(x) = 0.5005 + 0.4995 \sin(2\pi x).$$

If the penalty parameter is constant (cf. Figure 2a and Figure 2b), the accuracy of the approximation is quite poor for this problem. This is because the coefficient matrix is independent of the diffusion coefficient in this case, as can be seen from equations (10), (11), and (5). And since the diffusion process is not captured at all by the discretization, the approximation can not be accurate.

Coarse level accuracy for a diffusion-dependent penalty For a diffusion-dependent penalty parameter $\sigma = K$, the discretization is equivalent to a central difference discretization, and the method converges (cf. Figure 2c). If the penalty parameter $\sigma = 10K$, the method does not converge, but the *shape* of the solution is still captured accurately (cf. Figure 2d). This is because the coefficient matrix is equivalent to a central difference matrix, aside from a constant equal to 10. Altogether, although the approximation for $p = 0$ is not accurate in general, the *shape* of the solution is captured better for a diffusion-dependent σ than for a constant σ. This is valuable information for the spectral two-level methods.



(a) $\sigma = 1$ (b) $\sigma = 10$ (c) $\sigma = K$ (d) $\sigma = 10K$

Figure 2: Accuracy of the SIPG solution (red) with polynomial degree $p = 0$ and $N = 40$ mesh elements compared to the exact solution (black) for different values of the penalty parameter σ. Although the approximation is not accurate in general, the *shape* of the solution is captured better for a diffusion-dependent σ than for a constant σ. This information likely benefits the performance of the spectral two-level methods.

4 Numerical validation: comparing both two-level variants

Now that the spectral two-level methods have been defined, their performance can be tested by means of numerical experiments. After specifying the experimental setup (Section 4.1), this section compares the performance of the proposed spectral two-level deflation method to that of the original preconditioning variant (Section 4.2). Additionally, it examines the influence of the penalty parameter on the convergence speed of both the SIPG and the CG method (Section 4.3).

4.1 Specifying the experimental setup

To validate the spectral two-level deflation method, we study four diffusion problems, discretized by means of the SIPG method, and solved by means of the the generalized (deflated) PCG method. This section discusses the details of the experimental setup. In short, the test cases are smooth or layered problems with extreme contrasts in the coefficients. The SIPG penalty parameter is chosen diffusion-dependent, as motivated by Section 2.3 and Section 3.3, but a distorted version and a common constant value are also considered for comparison. Furthermore, block Jacobi smoothing is used for both the spectral two-level deflation technique and the original preconditioning strategy.

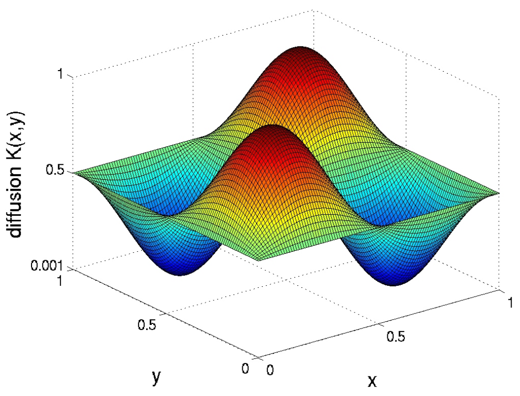


Figure 3: Smooth problem: the diffusion coefficient varies smoothly between 0.001 and 1 (cf. (19)).

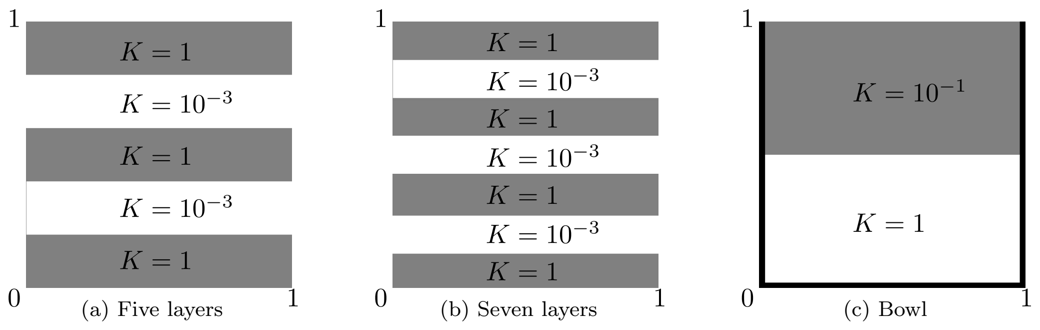


Figure 4: Three layered problems with extreme contrasts in the coefficients: a problem with five layers, a problem with seven layers, and a problem with two layers in a 'bowl'.

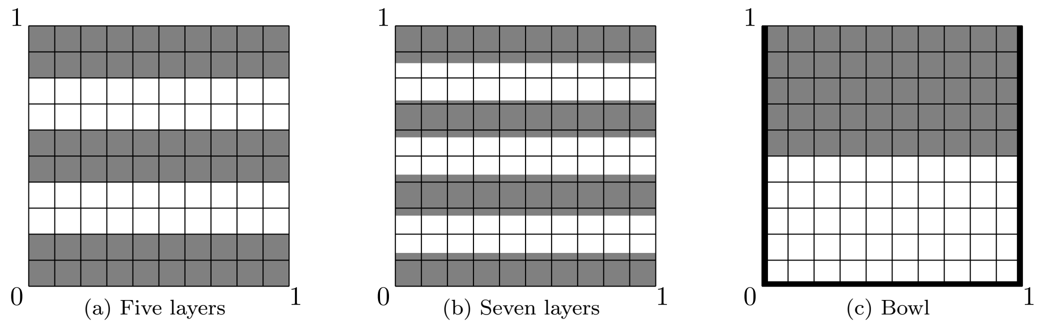


Figure 5: For the problem with five and two layers, the grid lines fall together with the layers. For the problem with seven layers this is not the case.

Test cases We consider the following two-dimensional diffusion problems of the form (1) on the domain $[0,1]^2$:

1. *Smooth problem*: this test case uses a diffusion coefficient that varies smoothly between 0.001 and 1 (cf. Figure 3):
$$K(x,y) = 0.5005 + 0.4995 \sin(2\pi x)\sin(2\pi y). \tag{19}$$
Dirichlet boundary conditions are applied at the entire domain boundary. The strong variations in the diffusion make this an interesting test case.

2. *Five layers*: this test case uses five layers of the same thickness, and the diffusion coefficient is either 1 or 0.001 in each layer (cf. Figure 4a). Dirichlet boundary conditions are applied at the entire domain boundary. This problem is more challenging than the smooth problem due to the jumps in the diffusion. Furthermore, the layered nature is characteristic for oil reservoir simulations. Similar problems were studied in [6, 13, 19].

3. *Seven layers*: this test case is the same as the one with five layers, except for the use of two extra layers (cf. Figure 4b).

4. *Bowl*: this test case uses only two layers of the same thickness, and the diffusion coefficient is either 1 or 0.1 (cf. Figure 4c). The challenging part of this test case is that homogeneous Neumann boundary conditions are applied at the black edges in Figure 4c (resulting in a bowl-shaped problem): these have a strongly negative impact on the conditioning of the SIPG coefficient matrix.

In each case, the Dirichlet boundary conditions and the source term f in (1) are chosen such that the exact solution reads:
$$u(x,y) = \cos(2\pi x)\cos(m\pi y),$$
where $m = 2$ for the smooth problem, and m is the number of layers for the layered problems.

SIPG Setup All model problems are discretized by means of the SIPG method as discussed in Section 2. We use a uniform and Cartesian mesh with $n \times n$ elements, where $n = 10, 20, 40, 80$. Observe that the grid lines fall together with the discontinuities for the problems with five and two layers (cf. Figure 5). For the problem with seven layers, this is not the case, so proper convergence of the SIPG method is not predicted by theory. We include this test case anyway for the sake of completeness. 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 by Section 2.3 and Section 3.3. 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 $\sigma = 20$, and a distorted version of $\sigma = 20K$:

$$\sigma(x, y) = 20K(x, y)(1.25 + 0.25 \sin(2\pi x) \sin(2\pi y)) \geq 20K. \tag{20}$$

This last choice is considered to test the robustness of the methods with respect to the penalty parameter.

CG Setup The linear systems resulting from the SIPG discretizations are solved by means of the generalized (deflated) PCG method as discussed in Section 3.2. Besides the original spectral two-level preconditioner (cf. Section 3.1) and the proposed deflation variant (cf. Section 3.2), we also consider standard diagonal preconditioning and basic block Jacobi preconditioning with blocks of order $\frac{(p+1)(p+2)}{2}$, the dimension of the polynomial test space within one element. Block Jacobi is also used for the smoothing operator M^{-1} in both spectral two-level variants. Coarse systems, involving the SIPG matrix A_0 with polynomial degree $p = 0$, are solved directly unless stated otherwise. However, a more practical strategy is also provided and tested in Section 4.2 below. In any case, the coarse matrix A_0 is quite similar to a central difference matrix, for which very efficient solvers are readily available.

Diagonal scaling is applied as a pre-processing step in all cases, and the same random start vector $\bar{x}$ is used for all problems of the same size. For the stopping criterion we use:

$$\frac{\|r_k\|_2}{\|b\|_2} \leq \text{TOL}, \tag{21}$$

where $\text{TOL} = 10^{-7}$, and $r_k = b - Ax_k$ is the residual after the k^{th} iteration. Additionally, we stop if the number of iterations is equal to the order of the matrix. Finally, we measure the condition number of the (diagonally-scaled) matrix A as the ratio between the largest and the smallest eigenvalue.

4.2 Main results for a diffusion-dependent penalty parameter

To validate the spectral two-level deflation method, we perform the experiments specified in the previous section. This section discusses the outcome of these experiments. We observe that the spectral two-level deflation technique yields uniform convergence, and that it is faster than the original preconditioning variant for larger problems, even though its costs per iteration are lower. Furthermore, we observe that the coarse systems can be solved efficiently by applying the CG method in an inner loop using relatively low accuracy.

Comparison of preconditioning strategies Table 3 compares the four preconditioning strategies in terms of the number of CG iterations required for convergence (also cf. Figure 6 for the convergence per iteration in the A-norm and the L^2-norm). As expected, the convergence for the two basic preconditioners slows down rapidly for larger meshes, whereas both spectral two-level methods yield fast uniform convergence, independent of the mesh element diameter. Interestingly, for 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.

More practical solution strategy for coarse systems To obtain the results in Table 3, a direct solver was used for the coarse systems with coefficient matrix A_0 (cf. Section 3.1). 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, Table 4 considers the cheaper alternative of applying the CG

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	6.5e+03	2.1e+04	7.5e+04	2.9e+05	2.8e+04	4.0e+04	1.4e+05	5.4e+05	7.4e+04	6.0e+04	2.1e+05	8.2e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	188	328	625	1205	538	727	1066	1736	855	926	1402	2152
Block Jacobi (BJ)	138	267	515	982	167	296	524	990	161	298	530	975
TL prec., 2x BJ	34	38	39	40	39	46	46	45	45	56	60	62
TL defl., 1x BJ	39	41	43	44	38	41	42	41	42	43	44	45

(c) Seven 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

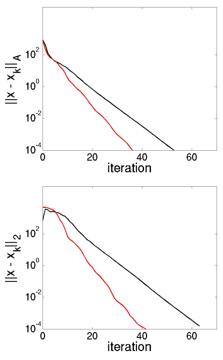
(d) Bowl, diffusion-dependent $\sigma = 20K$

Table 3: Comparing preconditioning techniques in terms of the number of CG iterations required for convergence: both spectral two-level methods yield fast uniform convergence. However, for the large problems, the proposed deflation technique is faster than the original preconditioning variant, even though the latter is more expensive per iteration due to an extra smoothing step.

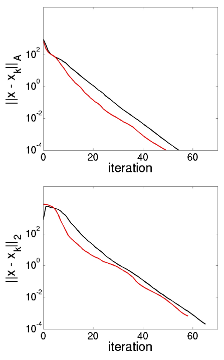
method again in an inner loop with the standard incomplete Cholesky preconditioner without fill-in. This table displays the number of outer iterations required for converge of the CG method with the spectral two-level deflation technique. Both the outer and the inner loop use stopping criterion (21). For the inner loop, we consider several values for TOL. For comparison, the results for a direct coarse solver are also displayed. The maximum number of iterations is set to 300. Observe that low accuracy in the inner loop is sufficient for high accuracy in the outer loop. For example, for the problems with five and seven layers, the inner tolerance can be 10^5 times as large as the outer tolerance.

4.3 Influence of the penalty parameter on the convergence of the SIPG and CG method

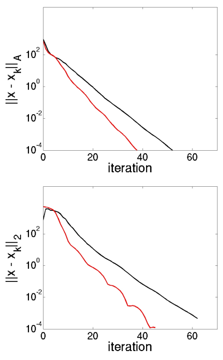
The previous section demonstrated that the spectral two-level method is quite effective for a diffusion-dependent penalty parameter. The question remains how this uncommon choice for the penalty parameter affects the convergence of the SIPG method. Another question is how the convergence of the CG method would change if the coarse matrix A_0 would no longer be equivalent to an accurate central difference matrix, as would be the case for a non-uniform mesh for instance. To answer these two questions, we compute the SIPG solutions for both a common constant and a diffusion-dependent penalty parameter. Furthermore, we repeat the CG experiments of the previous section for both a constant and a *distorted* diffusion-dependent penalty parameter, for



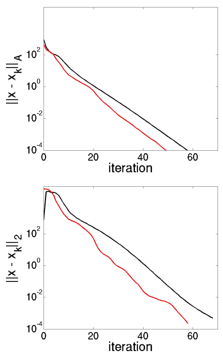
(a) Smooth problem



(b) Five layers



(c) Seven layers



(d) Bowl

Figure 6: CG convergence per iteration in terms of both the A-norm (top row) and the L^2-norm (bottom row) for polynomial degree $p = 3$, $N = 80^2$ mesh elements, and a diffusion-dependent penalty parameter $\sigma = 20K$. The spectral two-level deflation technique (red) is faster than the preconditioning variant (black) in all cases.

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
exact	36	41	43	44	38	39	39	39	40	41	43	43
TOL = 10^{-4}	36	41	43	44	38	39	39	39	40	41	43	43
TOL = 10^{-3}	36	41	43	44	38	39	39	39	40	41	43	44
TOL = 10^{-2}	36	41	43	46	38	39	39	40	40	41	43	44
TOL = 10^{-1}	49	91	160	300	43	64	98	131	45	69	78	300

(a) Smooth problem, inexact coarse solves

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

(b) Five layers, inexact coarse solves

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	6.5e+03	2.1e+04	7.5e+04	2.9e+05	2.8e+04	4.0e+04	1.4e+05	5.4e+05	7.4e+04	6.0e+04	2.1e+05	8.2e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
exact	39	41	43	44	38	41	42	41	42	43	44	45
TOL = 10^{-4}	39	41	43	44	38	41	42	41	42	43	44	45
TOL = 10^{-3}	39	41	43	44	38	41	42	41	42	43	44	45
TOL = 10^{-2}	40	41	44	46	38	41	43	43	42	43	44	45
TOL = 10^{-1}	85	188	300	300	67	87	125	300	63	111	201	300

(c) Seven layers, inexact coarse solves

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
exact	49	50	55	56	50	53	54	54	56	57	57	58
TOL = 10^{-4}	49	50	55	56	50	53	54	54	56	57	57	58
TOL = 10^{-3}	49	50	55	56	50	53	54	54	56	57	57	58
TOL = 10^{-2}	51	56	56	64	50	53	55	55	56	58	59	59
TOL = 10^{-1}	127	300	300	300	89	300	300	300	300	300	300	300

(d) Bowl, inexact coarse solves

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

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 10^2$	3.73e-01	-	4.43e-03	-	2.25e-04	-
$N = 20^2$	1.27e-01	1.56	4.08e-04	3.44	1.25e-05	4.17
$N = 40^2$	3.60e-02	1.81	3.94e-05	3.38	7.33e-07	4.09
$N = 80^2$	9.49e-03	1.93	4.34e-06	3.18	4.45e-08	4.04

(a) Smooth problem, fixed $\sigma = 20$

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 10^2$	2.02e-01	-	3.02e-03	-	1.95e-04	-
$N = 20^2$	6.16e-02	1.71	3.09e-04	3.29	1.20e-05	4.02
$N = 40^2$	1.66e-02	1.90	3.42e-05	3.18	6.97e-07	4.10
$N = 80^2$	4.24e-03	1.97	4.10e-06	3.06	4.24e-08	4.04

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

mesh	p=1		p=2		p=3	
	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

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

mesh	p=1		p=2		p=3	
	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

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

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 10^2$	4.87e-01	-	7.55e-01	-	1.04e+00	-
$N = 20^2$	2.62e-01	0.89	3.97e-01	0.93	4.16e-01	1.32
$N = 40^2$	9.92e-02	1.40	6.04e-02	2.72	5.86e-02	2.83
$N = 80^2$	1.46e-01	-0.56	1.24e-01	-1.04	1.24e-01	-1.08

(e) Seven layers, fixed $\sigma = 20$

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 10^2$	9.35e-01	-	9.93e-01	-	1.08e+00	-
$N = 20^2$	4.32e-01	1.11	4.29e-01	1.21	4.11e-01	1.39
$N = 40^2$	5.55e-02	2.96	5.76e-02	2.90	5.87e-02	2.81
$N = 80^2$	1.27e-01	-1.19	1.24e-01	-1.11	1.24e-01	-1.08

(f) Seven layers, diffusion-dependent $\sigma = 20K$

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 10^2$	2.82e-01	-	7.72e-03	-	2.67e-04	-
$N = 20^2$	1.23e-01	1.20	8.62e-04	3.16	1.47e-05	4.18
$N = 40^2$	4.52e-02	1.44	7.27e-05	3.57	7.89e-07	4.22
$N = 80^2$	1.40e-02	1.69	5.99e-06	3.60	4.54e-08	4.12

(g) Bowl, fixed $\sigma = 20$

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 10^2$	1.77e-01	-	3.39e-03	-	1.93e-04	-
$N = 20^2$	6.33e-02	1.48	3.19e-04	3.41	1.12e-05	4.10
$N = 40^2$	1.78e-02	1.83	3.45e-05	3.21	6.82e-07	4.04
$N = 80^2$	4.60e-03	1.95	4.11e-06	3.07	4.22e-08	4.01

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

Table 5: SIPG convergence for both a constant and diffusion-dependent penalty parameter: the convergence for the problem with seven layers is poor, as its discontinuities do not fall together with the mesh element boundaries. For the other cases, a diffusion-dependent penalty parameter yields higher SIPG accuracy, both in order and in absolute value.

which the coarse matrix A_0 is no longer equivalent to an accurate central difference matrix. This section discusses the results and compares them to those obtained in the previous section. We found that a diffusion-dependent penalty parameter yields smaller condition numbers and faster SIPG and CG convergence than a constant penalty parameter. Additionally, we observe that the spectral two-level deflation method suffers much less from changes in the penalty parameter than the preconditioning variant for large problems. In that sense, it is more robust. Finally, it can be concluded that the proposed combination of a diffusion-dependent penalty parameter and the spectral two-level deflation method can yield over 100 times faster CG convergence than the original combination of a constant penalty parameter and the spectral two-level preconditioner.

SIPG convergence Table 5 displays the SIPG convergence for both a constant and a diffusion-dependent penalty parameter. For all test cases except the problem with seven layers, the SIPG method converges at rate $p + 1$ for both a constant and a diffusion-dependent penalty parameter. Interestingly, for a diffusion-dependent penalty parameter, the accuracy is better, both in order and in absolute value. This is congruent with the observation in Section 2.3 that the underlying physics is better represented by the matrix coefficients for a diffusion-dependent penalty parameter.

As expected, the convergence for the problem with seven layers is poor, as the discontinuities do not fall together with the mesh element boundaries in that case (cf. Section 4.1). This explanation is extra verified in Table 6, which considers the SIPG convergence for the problems with five and seven layers for meshes with $N = 21, 42, 84$ elements: indeed, proper convergence is now obtained for the problem with seven layers, but not for the problem with five layers.

CG convergence for a constant penalty parameter Table 7 repeats the experiments of Section 4.2 for a constant penalty parameter $\sigma = 20$ (also cf. Figure 7 for the convergence per

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 21^2$	2.02e-01	-	3.87e-02	-	1.04e-02	-
$N = 42^2$	1.06e-01	0.93	5.01e-03	2.95	4.81e-03	1.12
$N = 84^2$	9.52e-02	0.16	1.09e-03	2.20	1.51e-03	1.68

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

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 21^2$	9.01e-02	-	1.31e-02	-	8.63e-03	-
$N = 42^2$	2.77e-02	1.70	3.59e-03	1.86	5.97e-03	0.53
$N = 84^2$	8.14e-03	1.76	7.43e-04	2.27	1.66e-03	1.85

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

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 21^2$	2.35e-01	-	4.15e-02	-	2.87e-03	-
$N = 42^2$	1.59e-01	0.57	8.70e-03	2.25	2.73e-04	3.39
$N = 84^2$	1.14e-01	0.48	1.21e-03	2.84	1.79e-05	3.93

(c) Seven layers, fixed $\sigma = 20$

mesh	p=1		p=2		p=3	
	error	order	error	order	error	order
$N = 21^2$	1.20e-01	-	3.74e-03	-	2.76e-04	-
$N = 42^2$	3.78e-02	1.67	4.27e-04	3.13	1.74e-05	3.99
$N = 84^2$	1.02e-02	1.88	5.18e-05	3.04	1.09e-06	4.00

(d) Seven layers, diffusion-dependent $\sigma = 20K$

Table 6: Extra verification using an alternative mesh (cf. Table 5): the SIPG method convergences at rate $p + 1$ for both a constant and a diffusion-dependent penalty parameter, as long as discontinuities fall together with the mesh element bounaries.

iteration in the A-norm and the L^2-norm). Because this value is much larger than the average diffusion-dependent value (cf. Table 3 with $\sigma = 20K$), the condition numbers are much larger. As a result, the convergence of the CG method is typically significantly slower for all preconditioning strategies. Nonetheless, for the large problems, the spectral two-level deflation technique can converge up to six times faster than the preconditioning variant. Finally, for the problem with five layers wit $p = 3$ and $N = 80$, the spectral two-level preconditioner (cf. Table 7 with constant penalty parameter $\sigma = 20$) requires 90 times more iterations than the deflation technique with a diffusion-dependent penalty parameter (cf. Table 3 with $\sigma = 20K$). Taking into account that the deflation technique is less expensive per iteration, it is over 100 times faster in terms of overal computational time.

CG convergence for a distorted diffusion-dependent penalty parameter Table 8 repeats the experiments of Section 4.2 for a distorted version of $\sigma = 20K$ (20) (also cf. Figure 8 for the convergence per iteration in the A-norm and the L^2-norm). The average of this value is a little larger than the average diffusion-dependent value (cf. Table 3 with $\sigma = 20K$), but much smaller than the constant value (cf. Table 7 with $\sigma = 20$). The condition numbers and convergence rates behave accordingly: both are worse compared to the undistorted diffusion-dependent case, but only slightly compared to the constant case. Furthermore, for the large problems, the spectral two-level deflation method can require up to 37% fewer iterations than the preconditioning variant.

5 Conclusions

This research is focused on an effective preconditioning strategy to increase the efficiency of the CG method for linear systems resulting from SIPG discretizations for diffusion problems with extreme contrasts in the coefficients. In short, we propose to apply the spectral two-level preconditioner introduced by Dobrev et al. [6] as a deflation method. Furthermore, we propose to choose the penalty parameter based on *local* values of the diffusion coefficient, instead of the usual strategy to use one *global* parameter for the entire domain. We found that the combination of these two approaches can render the CG method up to 100 times faster.

The spectral two-level preconditioner can be cast into the deflation framework by simply skipping one of the two smoothing steps. Interestingly, the resulting *asymmetric* 'preconditioning' operator can be effectively implemented in a PCG algorithm, as long as the starting vector is pre-processed in one cheap step. Coarse systems can be solved efficiently by applying the CG method again with a relatively large tolerance. Compared to the original preconditioning variant, the spectral two-level deflation strategy is less expensive per iteration due to lower smoothing costs, and more practical due to the absence of restrictions on the smoother. Nevertheless, it

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

(a) 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

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

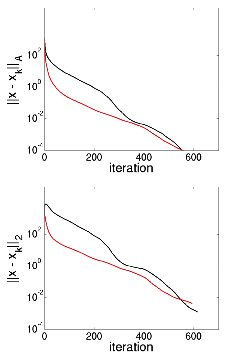
degree	p=1				p=2				p=3			
mesh	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2
condition number	4.0e+03	2.6e+04	1.5e+05	1.2e+06	2.3e+04	4.9e+05	2.6e+06	1.0e+07	1.2e+05	1.4e+06	3.9e+06	1.5e+07
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	155	382	1046	2775	431	1765	4486	9206	895	2680	4868	9691
Block Jacobi (BJ)	155	384	1044	2768	298	1084	2640	5733	552	1465	2876	6036
TL prec., 2x BJ	44	81	170	344	120	494	1158	2339	321	1106	2310	4508
TL defl., 1x BJ	58	115	267	435	167	441	696	760	428	774	852	921

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

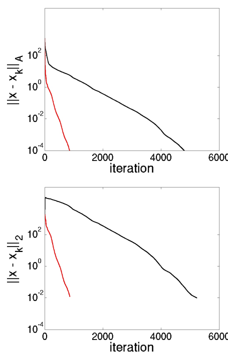
degree	p=1				p=2				p=3			
mesh	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2
condition number	1.2e+05	4.8e+05	1.9e+06	7.6e+06	2.2e+05	8.6e+05	3.4e+06	1.4e+07	3.1e+05	1.2e+06	4.8e+06	1.9e+07
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal 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 $\sigma = 20$

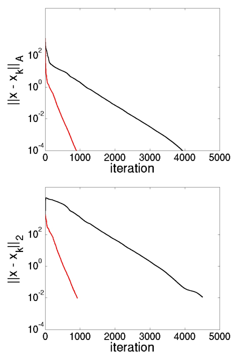
Table 7: Using a constant penalty parameter $\sigma = 20$: this typically yields much slower CG convergence for *all* preconditioning strategies than a diffusion-dependent value (cf. Table 3). Interestingly, for large problems, the spectral two-level deflation technique suffers much less from this than the preconditioning variant, even though the latter is more expensive per iteration due to an extra smoothing step.



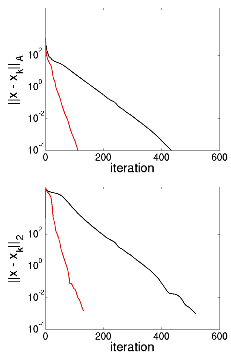
(a) Smooth problem



(b) Five layers



(c) Seven layers



(d) Bowl

Figure 7: CG convergence per iteration in terms of both the A-norm (top row) and the L^2-norm (bottom row) for polynomial degree $p = 3$, $N = 80^2$ mesh elements, and a constant penalty parameter $\sigma = 20$. The spectral two-level deflation technique (red) is faster than the preconditioning variant (black) in all cases (except for the smooth problem, for which the convergence is similar near the end).

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	3.4e+03	1.3e+04	5.2e+04	2.0e+05	8.2e+03	2.8e+04	1.0e+05	3.9e+05	1.2e+04	3.9e+04	1.4e+05	5.4e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	135	251	508	1000	236	447	834	1525	260	469	811	1560
Block Jacobi (BJ)	131	265	522	992	145	282	514	991	147	279	510	964
TL prec., 2x BJ	35	43	46	48	46	51	54	57	56	71	80	85
TL defl., 1x BJ	41	46	50	52	45	47	48	49	48	50	52	53

(a) Smooth problem, distorted version of $\sigma = 20K$

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	3.7e+04	5.7e+04	9.8e+04	3.2e+05	2.2e+05	2.3e+05	3.0e+05	6.5e+05	4.1e+05	3.9e+05	4.5e+05	8.5e+05
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	300	739	1080	1437	600	1434	1705	2168	1000	1902	2251	2750
Block Jacobi (BJ)	132	274	545	1070	161	301	512	1094	161	302	560	1073
TL prec., 2x BJ	39	46	48	50	52	61	60	62	57	74	82	87
TL defl., 1x BJ	47	53	59	62	60	61	62	63	64	64	67	69

(b) Five layers, distorted version of $\sigma = 20K$

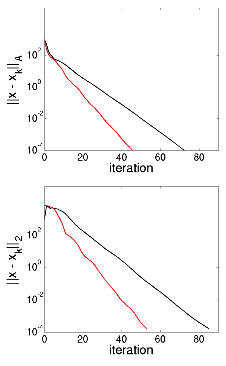
degree	p=1				p=2				p=3			
mesh	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2
condition number	8.3e+03	2.7e+04	9.6e+04	3.7e+05	3.6e+04	5.1e+04	1.8e+05	7.0e+05	9.9e+04	9.3e+04	2.7e+05	1.1e+06
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	199	359	689	1367	531	850	1243	1980	946	1064	1575	2461
Block Jacobi (BJ)	152	309	568	1102	186	343	571	1151	192	316	596	1114
TL prec., 2x BJ	37	43	46	49	45	54	56	56	52	68	80	86
TL defl., 1x BJ	44	48	50	52	45	50	50	51	51	53	53	55

(c) Seven layers, distorted version of $\sigma = 20K$

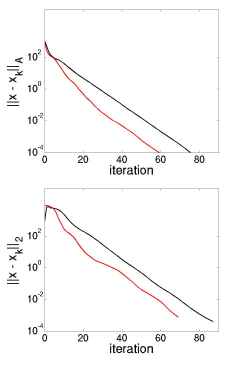
degree	p=1				p=2				p=3			
mesh	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2	N=10^2	N=20^2	N=40^2	N=80^2
condition number	1.3e+05	5.1e+05	2.0e+06	8.1e+06	2.5e+05	9.8e+05	3.9e+06	1.5e+07	3.4e+05	1.3e+06	5.3e+06	2.1e+07
size A	300	1200	4800	19200	600	2400	9600	38400	1000	4000	16000	64000
Diagonal prec.	252	529	1036	2020	408	751	1450	2837	485	879	1644	3227
Block Jacobi (BJ)	212	458	920	1763	240	478	911	1765	234	466	904	1760
TL prec., 2x BJ	47	50	51	52	62	61	60	60	76	84	85	88
TL defl., 1x BJ	55	58	62	63	58	60	61	62	62	64	65	66

(d) Bowl, distorted version of $\sigma = 20K$

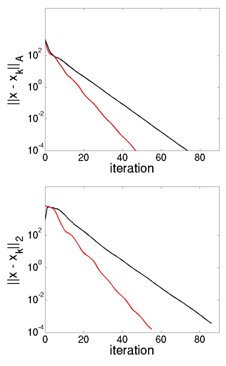
Table 8: Using a distorted version of the diffusion-dependent penalty parameter $\sigma = 20K$ (20): this yields somewhat slower CG convergence for *all* preconditioning strategies than the undistorted case (cf. Table 3). Interestingly, for the larger problems, the spectral two-level deflation technique suffers much less from this than the preconditioning variant, even though the latter is more expensive per iteration due to an extra smoothing step.



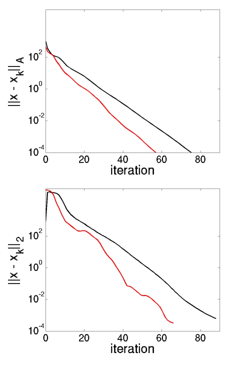
(a) Smooth problem



(b) Five layers



(c) Seven layers



(d) Bowl

Figure 8: CG convergence per iteration in terms of both the A-norm (top row) and the L^2-norm (bottom row) for polynomial degree $p = 3$, $N = 80^2$ mesh elements, and a distorted version of the diffusion-dependent penalty parameter $\sigma = 20K$ (20). The spectral two-level deflation technique (red) is faster than the preconditioning variant (black) in all cases.

yields fast uniform convergence, and it requires significantly fewer iterations for large problems. Altogether, the proposed spectral two-level deflation technique forms an effective preconditioning strategy for our application, especially when smoothing costs need to be minimized.

Compared to a constant penalty parameter, we find that the use of a diffusion-dependent penalty parameter introduces a new flexibility that has two main advantages for problems with strongly varying coefficients. First, the SIPG coefficient matrix resembles the underlying physical model better, which results in more accurate SIPG approximations. Second, the use of local values allows for local minimization of the penalty parameter, resulting in smaller condition numbers, and faster CG convergence. For these reasons, the penalty parameter can best be chosen diffusion-dependent for problems with extreme contrasts in the coefficients.

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] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM J. Numer. Anal.*, 19(4):742–760, 1982.

[3] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779 (electronic), 2002.

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

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

[6] 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.

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

[8] X. Feng and O. A. Karakashian. 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.

[9] 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.

[10] J. Gopalakrishnan and G. Kanschat. A multilevel discontinuous Galerkin method. *Numer. Math.*, 95(3):527–550, 2003.

[11] P.-O. Persson and J. Peraire. Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations. *SIAM J. Sci. Comput.*, 30(6):2709–2733, 2008.

[12] 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.

[13] 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.

[14] 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.

[15] Y. Saad and B. Suchomel. ARMS: an algebraic recursive multilevel solver for general sparse linear systems. *Numer. Linear Algebra Appl.*, 9(5):359–378, 2002.

[16] S. J. Sherwin, R. M. Kirby, J. Peiró, R. L. Taylor, and O. C. Zienkiewicz. On 2D elliptic discontinuous Galerkin methods. *Internat. J. Numer. Methods Engrg.*, 65(5):752–784, 2006.

[17] 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.

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

[19] C. Vuik, A. Segal, and J. 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.

[20] M. F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM J. Numer. Anal.*, 15(1):152–161, 1978.

[21] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Rev.*, 34(4):581–613, 1992.