OPTIMIZATION OF TWO-LEVEL METHODS FOR DG DISCRETIZATIONS OF REACTION-DIFFUSION EQUATIONS

JOSÉ PABLO LUCERO LORCA[1] AND MARTIN JAKOB GANDER[2]

Abstract. In this manuscript, two-level methods applied to a symmetric interior penalty discontinuous Galerkin finite element discretization of a singularly perturbed reaction-diffusion equation are analyzed. Previous analyses of such methods have been performed numerically by Hemker et al. for the Poisson problem. The main innovation in this work is that explicit formulas for the optimal relaxation parameter of the two-level method for the Poisson problem in 1D are obtained, as well as very accurate closed form approximation formulas for the optimal choice in the reaction-diffusion case in all regimes. Using Local Fourier Analysis, performed at the matrix level to make it more accessible to the linear algebra community, it is shown that for DG penalization parameter values used in practice, it is better to use *cell* block-Jacobi smoothers of Schwarz type, in contrast to earlier results suggesting that *point* block-Jacobi smoothers are preferable, based on a smoothing analysis alone. The analysis also reveals how the performance of the iterative solver depends on the DG penalization parameter, and what value should be chosen to get the fastest iterative solver, providing a new, direct link between DG discretization and iterative solver performance. Numerical experiments and comparisons show the applicability of the expressions obtained in higher dimensions and more general geometries. [1] [2] [3]

1. Introduction

Reaction-diffusion equations are differential equations arising from two of the most basic interactions in nature: reaction models the interchange of a substance from one type to another, and diffusion its displacement from a point to its neighborhood. Chemical reactors, radiation transport, and even stock option prices, all have regimes where their mathematical model is a reaction-diffusion equation with applications ranging from engineering to biology and finance [5, 13, 21, 27, 30].

In this paper, we present and analyze two-level methods to solve a symmetric interior penalty discontinuous Galerkin (SIPG) discretization of a singularly perturbed reaction-diffusion equation. Symmetric interior penalty methods [2, 3, 4, 28, 33] are particularly interesting to solve these equations since by imposing boundary conditions weakly they produce less oscillations near the boundaries in singularly perturbed problems [25]. Using this discretization, the reaction operator involves only volume integrals with no coupling between cells. Therefore, all its contributions are included inside the local subspaces when using *cell* block-Jacobi smoothers, which can then be interpreted as non-overlapping Schwarz smoothers

[1] University of Colorado Boulder, https://pablo.world/mathematics
[2] Université de Genève, https://www.unige.ch/~gander
E-mail addresses: pablo.lucero@colorado.com, martin.gander@unige.ch.

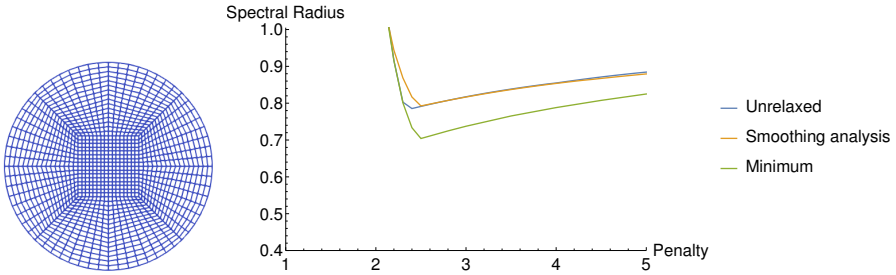


FIGURE 1. Left: circular domain and mesh used for the SIPG discretization of a Poisson problem. Right: spectral radius of the iteration operator as a function of the penalty parameter in SIPG using a *cell* block-Jacobi smoother, without damping (Unrelaxed), with optimized damping from a 1D smoothing optimization alone (Smoothing analysis), and the numerically optimized two level process (Minimum).

(see [11, 12, 26] and references therein). On the other hand, also *point* block-Jacobi smoothers have been considered in the literature, which we study as well.

The SIPG method leaves two parameters to be chosen by the user. One is the penalty parameter, which determines how discontinuous the solution is allowed to be between cells, and the other is the relaxation used for the stationary iteration. For classical finite element or finite difference discretizations of Poisson problems, it is sufficient to optimize the smoother alone by maximizing the damping in the high frequency range to get best performance of the two and multilevel method, which leads for a Jacobi smoother to the damping parameter $\frac{2}{3}$ (see [34]). This is however different for SIPG discretizations, as we show in Figure 1 for a Poisson problem on a disk discretized with SIPG on an irregular mesh. We see that the best damping parameter depends on the penalization parameter in SIPG, and can not be well predicted by a smoothing analysis alone. Our goal here is to optimize the entire two level process for such SIPG discretizations, both for Poisson and singularly perturbed problems.

We apply Local Fourier Analysis (LFA), which has been widely used for optimizing multigrid methods since its introduction in [7]. This tool allows obtaining quantitative estimates of the asymptotic convergence of numerical algorithms, and is particularly useful for multilevel ones. Based on the Fourier transform, the traditional LFA method is accurate for partial differential equations if the influence of boundary conditions is limited. It is well known [8], that the method is exact when periodic boundary conditions are used.

Previous Fourier analyses of such two-level methods for DG discretizations have been performed for the Poisson equation by Hemker et al. (see [18, 19] and references therein), who obtained numerically optimized parameters for *point* block-Jacobi smoothers. Our main results are first, explicit formulas for the relaxation parameters of both *point* and *cell* block-Jacobi smoothers for the Poisson equation and second, the extension to the reaction-diffusion case, where we derive very accurate closed form approximations of the optimal relaxation parameters for the two-level process. Using our analytical results, we can prove that for DG penalization

parameter values used in practice, it is better to use *cell* block-Jacobi smoothers of Schwarz type, in contrast to earlier results that suggested to use *point* block-Jacobi smoothers, based on a smoothing analysis alone. Furthermore, our analysis reveals that there is an optimal choice for the SIPG penalization parameter to get the fastest possible two-level iterative solver. A further important contribution in our opinion is that we present our LFA analysis using linear algebra tools and matrices to make this important technique more accessible in the linear algebra community.

A special point is made on the closed-form nature of our results. The mathematical community is divided between researchers pushing for the numerical optimization of LFA [31, 32] and researchers searching for analytical, closed-form results [24]. We value both approaches in their capacity to spearhead mathematical intutions numerically, that are then addressed formally as it often happens in science. We let go of considering 2D and 3D Fourier symbols, but we do include the complete 2-level method in our optimization instead of separating smoothing from coarse correction, expecting and ultimately confirming that the validity of the optimization is wider than the alternative.

To the best of our knowledge, even though many publications have applied LFA to two-level solvers for DG discretizations of elliptic problems since the work by Hemker et al., closed-form formulas for the relaxation parameter, optimized over the complete two-level process for each SIPG penalty, are missing from the literature since the algebraic expressions involved are quite cumbersome. Our expressions for the Poisson problem are exact in 1D, if periodic boundary conditions are used. Additionally, we provide numerical examples showing their applicability in higher dimensions and non-structured grids.

2. Model problem

We consider the reaction-diffusion model problem

$$-\Delta u + \frac{1}{\varepsilon} u = f \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega, \tag{1}$$

where $\Omega \subset \mathbb{R}^{1,2,3}$ is a convex domain, f is a known source function and $\varepsilon \in (0, \infty)$ is a parameter, defining the relative size of the reaction term.

We introduce the Hilbert spaces $\mathcal{H} = L^2(\Omega)$ and $\mathcal{V} = H_0^1(\Omega)$, where $H_0^1(\Omega)$ is the standard Sobolev space with zero boundary trace. They are provided with inner products $(u, v)_{\mathcal{H}} = \int_\Omega uv dx$ and $(u, v)_{\mathcal{V}} = \int_\Omega \nabla u \cdot \nabla v dx$ respectively. The weak form of problem (1) is: find $u \in \mathcal{V}$ such that

$$a(u, v) = (f, v)_{\mathcal{H}}, \tag{2}$$

where $f \in \mathcal{H}$ and the continuous bilinear form $a(\cdot, \cdot)$ is defined by

$$a(u, v) \coloneqq \int_\Omega \nabla u \cdot \nabla v dx + \frac{1}{\varepsilon} \int_\Omega uv dx = (u, v)_{\mathcal{V}} + \frac{1}{\varepsilon} (u, v)_{\mathcal{H}}. \tag{3}$$

The bilinear form $a(u, v)$ is continuous and $\mathcal{V}$-coercive relatively to $\mathcal{H}$ (see [10, §2.6]), i.e. there exist constants $\gamma_a, C_a > 0$ such that

$$a(u, u) \geq \gamma_a \|u\|_{\mathcal{V}}^2, \quad a(u, v) \leq C_a \|u\|_{\mathcal{V}} \|v\|_{\mathcal{V}}, \quad \forall u, v \in \mathcal{V}. \tag{4}$$

Note that even though γ_a is independent of ε, C_a is not, which motivates our search for robust two-level methods in the next section. From Lax-Milgram's theorem, the variational problem admits a unique solution in $\mathcal{V}$.

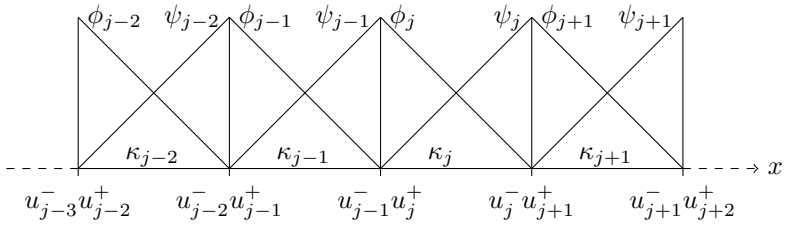


FIGURE 2. Mesh for the discretization and finite element functions.

2.1. **Discretization.** We discretize the domain Ω using segments, quadrilaterals or hexahedra, constituting a mesh $\mathbb{T}$ with cells $\kappa \in \mathbb{T}$ and faces $f \in \mathbb{F}$ using an SIPG finite element discretization. Let $\mathbb{Q}_p(\kappa)$ be the space of tensor product polynomials with degree up to p in each coordinate direction with support in κ. The discontinuous function space V_h is then defined as

$$V_h \coloneqq \left\{ v \in L^2(\Omega) \middle| \forall \kappa, v_{|\kappa} \in \mathbb{Q}_p(\kappa) \right\}. \tag{5}$$

Following [2], we introduce the jump and average operators $[\![u]\!] \coloneqq u^+ - u^-$ and $\{\!\{u\}\!\} \coloneqq \frac{u^- + u^+}{2}$, where the superscript indicates if the nodal value is evaluated from the left of the node ($^-$) or from the right ($^+$), and obtain the SIPG bilinear form

$$\begin{aligned} a_h(u,v) \coloneqq & \int_{\mathbb{T}} \nabla u \cdot \nabla v dx + \frac{1}{\varepsilon} \int_{\mathbb{T}} uv dx \\ & - \int_{\mathbb{F}} \left([\![u]\!] \left\{\!\!\left\{ \frac{\partial v}{\partial n} \right\}\!\!\right\} + \left\{\!\!\left\{ \frac{\partial u}{\partial n} \right\}\!\!\right\} [\![v]\!] \right) ds + \int_{\mathbb{F}} \delta [\![u]\!] [\![v]\!] ds, \end{aligned} \tag{6}$$

where n is the direction normal to the boundary, the boundary conditions have been imposed weakly (i.e. Nitsche boundary conditions [28]) and $\delta \in \mathbb{R}$ is a parameter penalizing the discontinuities at the interfaces between cells. On the boundary there is only a single value, and we set the value that would be on the other side to zero. In order for the discrete bilinear form to be coercive, we must choose $\delta = \delta_0/h$, where h is the largest diameter of the cells and $\delta_0 \in [1, \infty)$ is sufficiently large (see [22]). Coercivity and continuity are proved in [2] for the Laplacian under the assumption that δ_0 is sufficiently large, and these estimates are still valid in the presence of the reaction term, since it is positive definite.

For our analysis, we will focus on a one-dimensional problem[4], with equally spaced nodes and cells with equal size, see Fig. 2 for the mesh and finite element functions. We use the same kind of basis and test functions and we denote them by $\phi_j = \phi_j(x)$ and $\psi_j = \psi_j(x)$ for decreasing and increasing linear functions, respectively, with support in only one cell. The coefficients accompanying each basis function are $u_j^+, u_j^- \in \mathbb{R}$, where the superscript indicates if the nodal value is evaluated from the left of the node ($^-$) or from the right ($^+$).

[4]This is motivated by the seminal work of P. W. Hemker [19] who stated: *"we study the one-dimensional equation, since this can be considered as an essential building block for the higher dimensional case where we use tensor product polynomials"*. We test however our analytical results also in higher dimensions and on meshes which are not tensor products, see Subsection 6.5.

Any $v \in V_h$ can then be written as a linear combination of $\phi_j(x)$ and $\psi_j(x)$,

$$v = \sum_{j=1}^{J} u_j^+ \phi_j(x) + u_j^- \psi_j(x) = \boldsymbol{u} \cdot \boldsymbol{\xi}^\intercal(x), \text{with}$$

$$\boldsymbol{u} \coloneqq \left(\dots, u_{j-1}^+, u_{j-1}^-, u_j^+, u_j^-, u_{j+1}^+, u_{j+1}^-, \dots\right) \in \mathbb{R}^{2J},$$

$$\boldsymbol{\xi}(x) \coloneqq \left(\dots, \phi_{j-1}(x), \psi_{j-1}(x), \phi_j(x), \psi_j(x), \phi_{j+1}(x), \psi_{j+1}(x), \dots\right),$$

and $\phi_j(x), \psi_j(x) \in \mathbb{Q}_1(\kappa_j), \kappa_j \in \mathbb{T}, j \in (1, J)$. With this ordering, the SIPG discretization operator is

(7)

$$A = \begin{pmatrix}
\ddots & \ddots & a_h(\psi_{j-2}, \psi_{j-1}) & & & & \\
\ddots & \ddots & a_h(\phi_{j-1}, \psi_{j-1}) & a_h(\phi_{j-1}, \phi_j) & & & \\
a_h(\psi_{j-1}, \psi_{j-2}) & a_h(\psi_{j-1}, \phi_{j-1}) & a_h(\psi_{j-1}, \psi_{j-1}) & a_h(\psi_{j-1}, \phi_j) & a_h(\psi_{j-1}, \psi_j) & & \\
 & a_h(\phi_j, \phi_{j-1}) & a_h(\phi_j, \psi_{j-1}) & a_h(\phi_j, \phi_j) & a_h(\phi_j, \psi_j) & a_h(\phi_j, \phi_{j+1}) & \\
 & & a_h(\psi_j, \psi_{j-1}) & a_h(\psi_j, \phi_j) & \ddots & \ddots & \\
 & & & a_h(\phi_{j+1}, \phi_j) & \ddots & \ddots &
\end{pmatrix},$$

where the blank elements are zero. Using equation (6), evaluating (7) leads to

$$A\boldsymbol{u} = \frac{1}{h^2} \begin{pmatrix}
\ddots & \ddots & -\frac{1}{2} & & & & \\
\ddots & \ddots & \frac{h^2}{6\varepsilon} & -\frac{1}{2} & & & \\
-\frac{1}{2} & \frac{h^2}{6\varepsilon} & \delta_0 + \frac{h^2}{3\varepsilon} & 1 - \delta_0 & -\frac{1}{2} & & \\
 & -\frac{1}{2} & 1 - \delta_0 & \delta_0 + \frac{h^2}{3\varepsilon} & \frac{h^2}{6\varepsilon} & -\frac{1}{2} & \\
 & & -\frac{1}{2} & \frac{h^2}{6\varepsilon} & \ddots & \ddots & \\
 & & & -\frac{1}{2} & \ddots & \ddots &
\end{pmatrix} \begin{pmatrix} \vdots \\ u_{j-1}^+ \\ u_{j-1}^- \\ u_j^+ \\ u_j^- \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ f_{j-1}^+ \\ f_{j-1}^- \\ f_j^+ \\ f_j^- \\ \vdots \end{pmatrix} =: \boldsymbol{f}, \tag{8}$$

where

$$\boldsymbol{f} = \left(\dots, f_{j-1}^+, f_{j-1}^-, f_j^+, f_j^-, \dots\right) \in \mathbb{R}^{2J}$$

is a vector, analogous to $\boldsymbol{u}$, containing the coefficients of the representation of the right hand side in V_h. In the next section, we describe an iterative two-level solver for the linear system (8).

3. Solver

We solve the linear system (8) with a stationary iteration of the form

$$\boldsymbol{u}^{(i+1)} = \boldsymbol{u}^{(i)} + M^{-1}\left(\boldsymbol{f} - A\boldsymbol{u}^{(i)}\right), \tag{9}$$

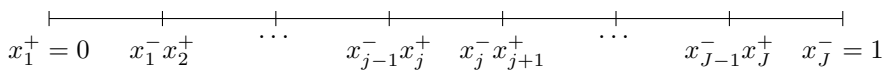


FIGURE 3. Mesh.

where M^{-1} is a two-level preconditioner, using first a cell-wise nonoverlapping Schwarz (*cell* block-Jacobi) smoother D_c^{-1} (see [11, 12]), i.e.

$$D_c^{-1}\boldsymbol{u} \coloneqq h^2 \begin{pmatrix} \ddots & & & & \\ & \delta_0 + \frac{h^2}{3\varepsilon} & \frac{h^2}{6\varepsilon} & & \\ & \frac{h^2}{6\varepsilon} & \delta_0 + \frac{h^2}{3\varepsilon} & & \\ & & & \ddots & \end{pmatrix}^{-1} \begin{pmatrix} \vdots \\ u_j^+ \\ u_j^- \\ \vdots \end{pmatrix}. \tag{10}$$

This smoother takes only into account the relation between degrees of freedom that are contained in each cell (x_j^+ and x_j^- in Fig. 3), i.e. we solve a local discrete reaction-diffusion problem consisting of one cell, like a domain decomposition method with subdomains formed by the cells.

Following [18], we consider as well a *point* block-Jacobi smoother, consisting of a *shifted* block definition, i.e.

$$D_p^{-1}\boldsymbol{u} \coloneqq h^2 \begin{pmatrix} \ddots & & & & \\ & \delta_0 + \frac{h^2}{3\varepsilon} & 1-\delta_0 & & \\ & 1-\delta_0 & \delta_0 + \frac{h^2}{3\varepsilon} & & \\ & & & \ddots & \end{pmatrix}^{-1} \begin{pmatrix} \vdots \\ u_j^- \\ u_{j+1}^+ \\ \vdots \end{pmatrix}. \tag{11}$$

In this case, the smoother takes into account the relation between degrees of freedom associated to a node (x_j^- and x_{j+1}^+ in Fig. 3). The domain decomposition interpretation in this case is less clear than for D_c.

Let the restriction operator be defined as

$$R \coloneqq \frac{1}{2} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} & & & & & \\ & \frac{1}{2} & \frac{1}{2} & 1 & & & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \ddots & \ddots & \ddots \end{pmatrix},$$

and the prolongation operator be $P \coloneqq 2R^\intercal$ (linear interpolation), and set $A_0 \coloneqq RAP$. Then the two-level preconditioner M^{-1}, with one presmoothing step and a relaxation parameter α, acting on a residual g is defined by Algorithm 1, where D^{-1} is the smoother and we'll study the choices D_c^{-1} and D_p^{-1}.

Algorithm 1 Two-level non-overlapping Schwarz preconditioned iteration.

1: compute $\boldsymbol{x} := \alpha D^{-1}\boldsymbol{g}$,
2: compute $\boldsymbol{y} := \boldsymbol{x} + PA_0^{-1}R(\boldsymbol{g} - A\boldsymbol{x})$,
3: obtain $M^{-1}\boldsymbol{g} = \boldsymbol{y}$.

4. Local Fourier Analysis (LFA)

In order to make the important LFA more accessible to the linear algebra community, we work directly with matrices instead of symbols. We consider a mesh as shown in Fig. 3, and assume for simplicity that it contains an even number of elements. Given that we are using nodal finite elements, a function $w \in V_h$ is uniquely determined by its values at the nodes, $\boldsymbol{w} = \left(\ldots, w_{j-1}^+, w_j^-, w_j^+, w_{j+1}^-, \ldots\right)$. For the local Fourier analysis (LFA), we can picture continuous functions that take the nodal values at the nodal points, and since in the DG discretization there are two values at each node, we consider two continuous functions, $w^+(x)$ and $w^-(x)$, which interpolate the nodal values of w to the left and right of the nodes, respectively. We next represent these two continuous functions as combinations of Fourier modes to get an understanding of how they are transformed by the two grid iteration.

4.1. LFA tools.

For a uniform mesh with mesh size h, and assuming periodicity, we can expand $w^-(x)$ and $w^+(x)$ into a finite Fourier series,

$$w^+(x) = \sum_{\widetilde{k}=-(J/2-1)}^{J/2} c_{\widetilde{k}}^+ e^{i2\pi\widetilde{k}x} = \sum_{k=1}^{J/2} c_{k-J/2}^+ e^{i2\pi(k-J/2)x} + c_k^+ e^{i2\pi kx},$$

$$w^-(x) = \sum_{\widetilde{k}=-(J/2-1)}^{J/2} c_{\widetilde{k}}^- e^{i2\pi\widetilde{k}x} = \sum_{k=1}^{J/2} c_{k-J/2}^- e^{i2\pi(k-J/2)x} + c_k^- e^{i2\pi kx}.$$

Enforcing the interpolation condition for these trigonometric polynomials at the nodes, $w_j^+ := w^+(x_j^+)$ and $w_j^- := w^-(x_j^-)$, we obtain

$$w_j^+ = \sum_{k=1}^{J/2} c_{k-J/2}^+ e^{i2\pi(k-J/2)x_j^+} + c_k^+ e^{i2\pi kx_j^+} = \sum_{k=1}^{J/2} c_{k-J/2}^+ e^{i2\pi(k-J/2)(j-1)h} + c_k^+ e^{i2\pi k(j-1)h},$$

$$w_j^- = \sum_{k=1}^{J/2} c_{k-J/2}^- e^{i2\pi(k-J/2)x_j^-} + c_k^- e^{i2\pi kx_j^-} = \sum_{k=1}^{J/2} c_{k-J/2}^- e^{i2\pi(k-J/2)jh} + c_k^- e^{i2\pi kjh}.$$

The representation for $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ as a set of nodal values can therefore be written as

$$\boldsymbol{w}^+ = \begin{pmatrix} w_1^+ \\ \vdots \\ w_j^+ \\ \vdots \\ w_J^+ \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{J/2} c_{k-J/2}^+ + c_k^+ \\ \vdots \\ \sum_{k=1}^{J/2} c_{k-J/2}^+ e^{i2\pi(k-J/2)(j-1)h} + c_k^+ e^{i2\pi k(j-1)h} \\ \vdots \\ \sum_{k=1}^{J/2} c_{k-J/2}^+ e^{i2\pi(k-J/2)(J-1)h} + c_k^+ e^{i2\pi k(J-1)h} \end{pmatrix},$$

$$\boldsymbol{w}^- = \begin{pmatrix} w_1^- \\ \vdots \\ w_j^- \\ \vdots \\ w_J^- \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{J/2} c_{k-J/2}^- e^{i2\pi(k-J/2)h} + c_k^- e^{i2\pi kh} \\ \vdots \\ \sum_{k=1}^{J/2} c_{k-J/2}^- e^{i2\pi(k-J/2)jh} + c_k^- e^{i2\pi kjh} \\ \vdots \\ \sum_{k=1}^{J/2} c_{k-J/2}^- e^{i2\pi(k-J/2)Jh} + c_k^- e^{i2\pi kJh} \end{pmatrix}.$$

We thus write the Fourier representation as a matrix-vector product and define two matrices Q^+ and Q^-, such that $\boldsymbol{w}^+ = Q^+\boldsymbol{c}^+$ and $\boldsymbol{w}^- = Q^-\boldsymbol{c}^-$, where

$$Q^+ := \begin{pmatrix} 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ e^{-i2\pi(1-J/2)(j-1)h} & e^{i2\pi(j-1)h} & \dots & e^{i2\pi(k-J/2)(j-1)h} & e^{i2\pi k(j-1)h} & \dots & 1 & e^{i2\pi(J/2)(j-1)h} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ e^{-i2\pi(1-J/2)(J-1)h} & e^{i2\pi(J-1)h} & \dots & e^{i2\pi(k-J/2)(J-1)h} & e^{i2\pi k(J-1)h} & \dots & 1 & e^{i2\pi(J/2)(J-1)h} \end{pmatrix},$$

$$Q^- := \begin{pmatrix} e^{i2\pi(1-J/2)h} & e^{i2\pi h} & \dots & e^{i2\pi(k-J/2)h} & e^{i2\pi kh} & \dots & 1 & e^{i2\pi(J/2)h} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ e^{i2\pi(1-J/2)jh} & e^{i2\pi jh} & \dots & e^{i2\pi(k-J/2)jh} & e^{i2\pi kjh} & \dots & 1 & e^{i2\pi(J/2)jh} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ e^{i2\pi(1-J/2)Jh} & e^{i2\pi Jh} & \dots & e^{i2\pi(k-J/2)Jh} & e^{i2\pi kJh} & \dots & 1 & e^{i2\pi(J/2)Jh} \end{pmatrix},$$

and

$$\boldsymbol{c}^+ := \begin{pmatrix} c_{1-J/2}^+ & c_1^+ & \dots & c_{k-J/2}^+ & c_k^+ & \dots & c_0^+ & c_{J/2}^+ \end{pmatrix}^\intercal,$$

$$\boldsymbol{c}^- := \begin{pmatrix} c_{1-J/2}^- & c_1^- & \dots & c_{k-J/2}^- & c_k^- & \dots & c_0^- & c_{J/2}^- \end{pmatrix}^\intercal.$$

An element in V_h can then be represented by its nodal elements in a stacked vector

$$\check{\boldsymbol{w}} = \begin{pmatrix} \boldsymbol{w}^+ \\ \boldsymbol{w}^- \end{pmatrix} = \begin{pmatrix} Q^+ & \\ & Q^- \end{pmatrix} \begin{pmatrix} \boldsymbol{c}^+ \\ \boldsymbol{c}^- \end{pmatrix} =: \check{Q}\check{\boldsymbol{c}}.$$

We now reorder the vectors $\check{\boldsymbol{w}}$ and $\check{\boldsymbol{c}}$ to obtain the new vectors $\boldsymbol{w}$ and $\boldsymbol{c}$ such that their elements are ordered from left to right with respect to the mesh. To do so, we define an orthogonal matrix S, such that $\boldsymbol{w} = S^\intercal\check{\boldsymbol{w}}$ and $\check{\boldsymbol{w}} = S\boldsymbol{w}$,

$$S^\intercal := \left(\begin{array}{cc:cc} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \\ & \dots & & \\ & & & \dots \end{array}\right),$$

where the dashed line is drawn between the two columns in the middle of the matrix. Finally, we define the reordered and scaled matrix Q

$$\boldsymbol{w} = S^\intercal\check{Q}S\boldsymbol{c} =: \left(\sqrt{h}\right)^{-1} Q\boldsymbol{c}.$$

The structure of Q is

$$Q = \sqrt{h}\begin{pmatrix}
\cdots & & \cdots & & \cdots & & \cdots & \\
& \cdots & & \cdots & & \cdots & & \cdots \\
\cdots & & e^{i2\pi(k-J/2)(j-2)h} & & e^{i2\pi k(j-2)h} & & \cdots & \\
& \cdots & & e^{i2\pi(k-J/2)(j-1)h} & & e^{i2\pi k(j-1)h} & & \cdots \\
\cdots & & e^{i2\pi(k-J/2)(j-1)h} & & e^{i2\pi k(j-1)h} & & \cdots & \\
& \cdots & & e^{i2\pi(k-J/2)jh} & & e^{i2\pi kjh} & & \cdots \\
\cdots & & e^{i2\pi(k-J/2)jh} & & e^{i2\pi kjh} & & \cdots & \\
& \cdots & & e^{i2\pi(k-J/2)(j+1)h} & & e^{i2\pi k(j+1)h} & & \cdots \\
\cdots & & e^{i2\pi(k-J/2)(j+1)h} & & e^{i2\pi k(j+1)h} & & \cdots & \\
& \cdots & & e^{i2\pi(k-J/2)(j+2)h} & & e^{i2\pi k(j+2)h} & & \cdots \\
\cdots & & \cdots & & \cdots & & \cdots & \\
& \cdots & & \cdots & & \cdots & & \cdots
\end{pmatrix}, \tag{12}$$

where the factor $\sqrt{h}$ is inserted such that Q is unitary (i.e. $Q^* = Q^{-1}$).

If we follow the same procedure for a coarser mesh, created by joining neighboring cells together, the matrix Q_0, analogous to Q, picks up the elements corresponding to the nodes contained in both the coarse and fine meshes,

$$Q_0 = \sqrt{2h}\begin{pmatrix}
\cdots & & \cdots & & \cdots & \\
& \cdots & & \cdots & & \cdots \\
\cdots & & e^{i2\pi(k-J/2)(j-2)h} & & \cdots & \\
& \cdots & & e^{i2\pi kjh} & & \cdots \\
\cdots & & e^{i2\pi(k-J/2)jh} & & \cdots & \\
& \cdots & & e^{i2\pi k(j+2)h} & & \cdots \\
\cdots & & \cdots & & \cdots & \\
& \cdots & & \cdots & & \cdots
\end{pmatrix},$$

where $j \geq 2$ is even and the factor $\sqrt{2h}$ is inserted such that Q_0 is unitary. We next show that Q renders A and D block diagonal and Q_0 and Q do the same for R and P, albeit with rectangular blocks. Therefore the study of the two grid iteration operator is reduced to the study of a generic block. In order to prove this result we need the following lemma.

Lemma 4.1. *Let $C \in \mathbb{R}^{2J\times 2J}$ be a block circulant matrix of the form*

$$C = \begin{pmatrix}
C_0 & C_1 & C_2 & \ldots & 0 & \ldots & C_{-2} & C_{-1} \\
C_{-1} & C_0 & C_1 & C_2 & \ldots & 0 & \ldots & C_{-2} \\
C_{-2} & C_{-1} & C_0 & C_1 & C_2 & \ldots & 0 & \ldots \\
\ldots & C_{-2} & C_{-1} & C_0 & C_1 & C_2 & \ldots & \ldots \\
0 & \ldots & C_{-2} & C_{-1} & C_0 & C_1 & C_2 & \ldots \\
\ldots & 0 & \ldots & C_{-2} & C_{-1} & C_0 & C_1 & \ldots \\
C_2 & \ldots & 0 & \ldots & C_{-2} & C_{-1} & C_0 & \ldots \\
C_1 & C_2 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots
\end{pmatrix},$$

where C_j represents (2×2)-blocks, and let $Q \in \mathbb{R}^{2J\times 2J}$ be the matrix which columns are discrete grid functions *as defined in* (12)*, then the matrix $M = Q^*CQ$ is (2×2)-block diagonal.*

Proof. We compute the block (p, q) of M to be

$$M_{p,q} = \sum_{k=-(J/2-1)}^{J/2-1} \sum_{l=1}^{J} Q^*_{l,p} C_k Q_{((k+l-1)\%J)+1,q},$$

where we denote by $\%J$ equivalency modulo J, and a block (m,n) of Q is

$$Q_{m,n} = \begin{cases} \begin{pmatrix} e^{i2\pi((n+1)/2-J/2)(m-1)h} & 0 \\ 0 & e^{i2\pi((n+1)/2-J/2)mh} \end{pmatrix}, & \text{if } n \text{ is odd,} \\ \begin{pmatrix} e^{i2\pi(n/2)(m-1)h} & 0 \\ 0 & e^{i2\pi(n/2)mh} \end{pmatrix}, & \text{if } n \text{ is even.} \end{cases}$$

As before, we will use for the small blocks the notation $C_k = \left(\begin{smallmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{smallmatrix}\right)$. We consider an off-diagonal block, i.e. $Q_{p,q}$, with $p \neq q$, and take an arbitrary k. Then if p and q are even, we have

$$\begin{aligned} &\sum_{l=1}^{J} Q_{l,p}^* C_k Q_{((k+l-1)\%J)+1,q} \\ &\quad = \sum_{l=1}^{J} \begin{pmatrix} c_{11} e^{\frac{i(((k+l-1)\%J)+1-1)\pi q}{J} - \frac{i(l-1)p\pi}{J}} & c_{12} e^{\frac{i(((k+l-1)\%J)+1)\pi q}{J} - \frac{i(l-1)p\pi}{J}} \\ c_{21} e^{\frac{i(((k+l-1)\%J)+1-1)\pi q}{J} - \frac{ilp\pi}{J}} & c_{22} e^{\frac{i(((k+l-1)\%J)+1)\pi q}{J} - \frac{ilp\pi}{J}} \end{pmatrix} \\ &\quad = \begin{pmatrix} c_{11} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}(p+(k-1)q)\right)\%J\right)} & c_{12} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}(p+kq)\right)\%J\right)} \\ c_{21} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}(k-1)q\right)\%J\right)} & c_{22} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}kq\right)\%J\right)} \end{pmatrix} \sum_{l=1}^{J} e^{\frac{i2\pi}{J}\left(\frac{1}{2}(q-p)l\right)\%J} = 0, \end{aligned}$$

since we identify the sum of the roots of unity. If p and q are odd, we have

$$\begin{aligned} &\sum_{l=1}^{J} Q_{l,p}^* C_k Q_{((k+l-1)\%J)+1,q} \\ &= \sum_{l=1}^{J} \begin{pmatrix} c_{11} e^{\frac{i2((k+l-1)\%J)+1-1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{i2(l-1)\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} & c_{12} e^{\frac{i2((k+l-1)\%J)+1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{i2(l-1)\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} \\ c_{21} e^{\frac{i2((k+l-1)\%J)+1-1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{i2l\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} & c22 e^{\frac{i2((k+l-1)\%J)+1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{i2l\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} \end{pmatrix} \\ &= \begin{pmatrix} c_{11} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}(k+p+(k-1)q)\right)\%J\right)} & c_{12} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}(p+k(q+1)+1)\right)\%J\right)} \\ c_{21} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}(k-1)(q+1)\right)\%J\right)} & c_{22} e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}k(q+1)\right)\%J\right)} \end{pmatrix} \sum_{l=1}^{J} e^{\frac{i2\pi}{J}\left(\frac{1}{2}(q-p)l\right)\%J} = 0, \end{aligned}$$

since again we identify the sum of the roots of unity. If p is odd and q is even, we get

$$\begin{aligned} &\sum_{l=1}^{J} Q_{l,p}^* C_k Q_{((k+l-1)\%J)+1,q} \\ &= \sum_{l=1}^{J} \begin{pmatrix} c_{11} e^{\frac{i((k+l-1)\%J)+1-1)\pi q}{J} - \frac{i2(l-1)\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} & c_{12} e^{\frac{i((k+l-1)\%J)+1)\pi q}{J} - \frac{i2(l-1)\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} \\ c_{21} e^{\frac{i((k+l-1)\%J)+1-1)\pi q}{J} - \frac{i2l\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} & c_{22} e^{\frac{i((k+l-1)\%J)+1)\pi q}{J} - \frac{i2l\left(\frac{p+1}{2}-\frac{J}{2}\right)\pi}{J}} \end{pmatrix} \\ &\qquad \begin{pmatrix} c_{11} e^{-\frac{i\pi(J-p-kq+q-1)}{J}} & c_{12} e^{-\frac{i\pi(J-p-kq-1)}{J}} \\ c_{21} e^{\frac{i(k-1)\pi q}{J}} & c_{22} e^{\frac{ik\pi q}{J}} \end{pmatrix} \sum_{l=1}^{J} e^{\frac{i2\pi}{J}\left(\frac{1}{2}(q-p-1+J)l\right)\%J} = 0. \end{aligned}$$

If p is even and q is odd, we get similarly

$$\begin{aligned} &\sum_{l=1}^{J} Q_{l,p}^* C_k Q_{((k+l-1)\%J)+1,q} = \\ &= \sum_{l=1}^{J} \begin{pmatrix} c_{11} e^{\frac{i2((k+l-1)\%J)+1-1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{i(l-1)p\pi}{J}} & c_{12} e^{\frac{i2((k+l-1)\%J)+1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{i(l-1)p\pi}{J}} \\ c_{21} e^{\frac{i2((k+l-1)\%J)+1-1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{ilp\pi}{J}} & c_{22} e^{\frac{i2((k+l-1)\%J)+1)\pi\left(\frac{q+1}{2}-\frac{J}{2}\right)}{J} - \frac{ilp\pi}{J}} \end{pmatrix} \end{aligned}$$

$$= \begin{pmatrix} c_{11}e^{-\frac{i2\pi}{J}\left(\left(\frac{1}{2}(J(k-1)-p+q-k(q+1)+1)\right)\%J\right)} & c_{12}e^{\frac{i2\pi}{J}\left(\left(\frac{1}{2}(p+k(-J+q+1))\right)\%J\right)} \\ c_{21}e^{-\frac{i2\pi}{J}\left(\left(\frac{1}{2}(k-1)(J-q-1)\right)\%J\right)} & c_{22}e^{-\frac{i2\pi}{J}\left(\left(\frac{1}{2}k(J-q-1)\right)\%J\right)} \end{pmatrix}$$

$$\sum_{l=1}^{J} e^{\frac{i2\pi}{J}\left(\frac{1}{2}(q-p+1-J)l\right)\%J} = 0,$$

and thus M is a (2×2)-block diagonal matrix. □

Given that Lemma 4.1 ensures M is block diagonal, a generic block with block index p, q can be computed as follows:

$$\begin{aligned} M = Q^*CQ &\iff QM = CQ \iff (QM)_{p,q} = (CQ)_{p,q}, \quad \forall p,q \\ &\iff Q_{p,q}M_q = \sum_{k=-(J/2-1)}^{J/2-1} C_k Q_{((k+p-1)\%J)+1,q}, \quad \forall p,q \\ &\iff M_q = (Q^*)_{q,p} \sum_{k=-(J/2-1)}^{J/2-1} C_k Q_{((k+p-1)\%J)+1,q}, \quad \forall p,q \\ &\iff \widetilde{M} = Q_l \widetilde{C} Q_r, \end{aligned}$$

where $\widetilde{C}Q_r = \sum_{k=-(J/2-1)}^{J/2-1} C_k Q_{((k+p-1)\%J)+1,q}$,

$$Q_r := \sqrt{\frac{1}{2}} \begin{pmatrix} e^{i2\pi(k-J/2)(j-2)h} & & e^{i2\pi k(j-2)h} & \\ & e^{i2\pi(k-J/2)(j-1)h} & & e^{i2\pi k(j-1)h} \\ e^{i2\pi(k-J/2)(j-1)h} & & e^{i2\pi k(j-1)h} & \\ & e^{i2\pi(k-J/2)jh} & & e^{i2\pi kjh} \\ e^{i2\pi(k-J/2)jh} & & e^{i2\pi kjh} & \\ & e^{i2\pi(k-J/2)(j+1)h} & & e^{i2\pi k(j+1)h} \\ e^{i2\pi(k-J/2)(j+1)h} & & e^{i2\pi k(j+1)h} & \\ & e^{i2\pi(k-J/2)(j+2)h} & & e^{i2\pi k(j+2)h} \end{pmatrix},$$

$$Q_l := \sqrt{\frac{1}{2}} \begin{pmatrix} e^{-i2\pi(k-J/2)(j-1)h} & & e^{-i2\pi(k-J/2)jh} & \\ & e^{-i2\pi(k-J/2)jh} & & e^{-i2\pi(k-J/2)(j+1)h} \\ e^{-i2\pi k(j-1)h} & & e^{-i2\pi kjh} & \\ & e^{-i2\pi kjh} & & e^{-i2\pi k(j+1)h} \end{pmatrix},$$

and the factor $\sqrt{\frac{1}{2}}$ is chosen such that $Q_l I_{4\times 8} Q_r = I_{4\times 4}$, where $I_{4\times 4}$ is the 4×4 identity matrix and

$$I_{4\times 8} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

We have computed a generic block $\widetilde{M}$ in the block diagonal of M. In the next subsection, we will work with blocks of size 4 by 4, given that we use a coarse correction with coarse cells formed from 2 adjacent fine cells with 2 degrees of freedom each.

4.2. **Analysis of the SIPG operator and associated smoothers.** We extract a submatrix $\widetilde{A}$ containing the degrees of freedom of two adjacent cells from the SIPG operator defined in (8),

$$\widetilde{A} = \begin{pmatrix} -\frac{1}{2} & 1-\delta_0 & \delta_0+\frac{h^2}{3\varepsilon} & \frac{h^2}{6\varepsilon} & -\frac{1}{2} & & \\ & -\frac{1}{2} & \frac{h^2}{6\varepsilon} & \delta_0+\frac{h^2}{3\varepsilon} & 1-\delta_0 & -\frac{1}{2} & \\ & & -\frac{1}{2} & 1-\delta_0 & \delta_0+\frac{h^2}{3\varepsilon} & \frac{h^2}{6\varepsilon} & -\frac{1}{2} \\ & & & -\frac{1}{2} & \frac{h^2}{6\varepsilon} & \delta_0+\frac{h^2}{3\varepsilon} & 1-\delta_0 & -\frac{1}{2} \end{pmatrix}.$$

We can now begin the block-diagonalization,

$$\widehat{A} = Q_l \widetilde{A} Q_r = \frac{1}{h^2} \begin{pmatrix} \delta_0+\frac{h}{3\varepsilon}+\cos(2\pi(k-J/2)h) & 1-\delta_0+\frac{h^2}{6\varepsilon}e^{i2\pi(k-J/2)h} & & \\ 1-\delta_0+\frac{h^2}{6\varepsilon}e^{-i2\pi(k-J/2)h} & \delta_0+\frac{h^2}{3\varepsilon}+\cos(2\pi(k-J/2)h) & & \\ & & \delta_0+\frac{h^2}{3\varepsilon}-\cos(2\pi kh) & 1-\delta_0+\frac{h^2}{6\varepsilon}e^{i2\pi kh} \\ & & 1-\delta_0+\frac{h^2}{6\varepsilon}e^{-i2\pi kh} & \delta_0+\frac{h^2}{3\varepsilon}-\cos(2\pi kh) \end{pmatrix}.$$

The same mechanism can be applied to the smoothers

$$\widetilde{D}_c = \begin{pmatrix} 0 & 0 & \delta_0+\frac{h^2}{3\varepsilon} & \frac{h^2}{6\varepsilon} & 0 & & \\ & 0 & \frac{h^2}{6\varepsilon} & \delta_0+\frac{h^2}{3\varepsilon} & 0 & 0 & \\ & & 0 & 0 & \delta_0+\frac{h^2}{3\varepsilon} & \frac{h^2}{6\varepsilon} & 0 \\ & & & 0 & \frac{h^2}{6\varepsilon} & \delta_0+\frac{h^2}{3\varepsilon} & 0 \end{pmatrix}, \tag{14}$$

$$\widehat{D}_c = Q_l \widetilde{D}_c Q_r = \frac{1}{h^2} \begin{pmatrix} \delta_0+\frac{h}{3\varepsilon} & \frac{h^2}{6\varepsilon}e^{i2\pi(k-J/2)h} & & \\ \frac{h^2}{6\varepsilon}e^{-i2\pi(k-J/2)h} & \delta_0+\frac{h^2}{3\varepsilon} & & \\ & & \delta_0+\frac{h^2}{3\varepsilon} & \frac{h^2}{6\varepsilon}e^{i2\pi kh} \\ & & \frac{h^2}{6\varepsilon}e^{-i2\pi kh} & \delta_0+\frac{h^2}{3\varepsilon} \end{pmatrix}, \tag{15}$$

and

$$\widetilde{D}_p = \begin{pmatrix} 0 & 1-\delta_0 & \delta_0+\frac{h^2}{3\varepsilon} & 0 & 0 & & \\ & 0 & 0 & \delta_0+\frac{h^2}{3\varepsilon} & 1-\delta_0 & 0 & \\ & & 0 & 1-\delta_0 & \delta_0+\frac{h^2}{3\varepsilon} & 0 & 0 \\ & & & 0 & 0 & \delta_0+\frac{h^2}{3\varepsilon} & 1-\delta_0 & 0 \end{pmatrix}, \tag{16}$$

$$\widehat{D}_p = Q_l \widetilde{D}_p Q_r = \frac{1}{h^2} \begin{pmatrix} \delta_0+\frac{h}{3\varepsilon} & 1-\delta_0 & & \\ 1-\delta_0 & \delta_0+\frac{h^2}{3\varepsilon} & & \\ & & \delta_0+\frac{h^2}{3\varepsilon} & 1-\delta_0 \\ & & 1-\delta_0 & \delta_0+\frac{h^2}{3\varepsilon} \end{pmatrix}. \tag{17}$$

We continue with the analysis of the restriction, prolongation and coarse operators.

4.3. **Analysis of the restriction, prolongation and coarse operators.** The same block-diagonalization is possible for the restriction and prolongation operators. The calculation for the restriction gives

(18)
$$\widetilde{R} = \frac{1}{2}\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} & & & & & \\ & \frac{1}{2} & \frac{1}{2} & 1 & & & & \\ & & & & 1 & \frac{1}{2} & \frac{1}{2} & \\ & & & & & \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix},$$

$$\widehat{R} = \frac{1}{2} Q_{l0} \widetilde{R} Q_r$$
$$= \frac{1}{2\sqrt{2}} \begin{pmatrix} 2 + e^{i2\pi(k-J/2)h} & e^{i2\pi(k-J/2)h} & (-1)^j \left(2 + e^{i2\pi kh}\right) & (-1)^j \left(e^{i2\pi kh}\right) \\ (-1)^j \left(e^{-i2\pi(k-J/2)h}\right) & (-1)^j \left(2 + e^{-i2\pi(k-J/2)h}\right) & e^{-i2\pi kh} & 2 + e^{-i2\pi kh} \end{pmatrix},$$

and for the prolongation operator we obtain

$$P = 2R^{\intercal}, \qquad \widehat{P} = Q_l \widetilde{P} Q_{r0} = 2\widehat{R}^*, \tag{19}$$

and finally for the coarse operator

$$Q_0^* A_0 Q_0 = Q_0^* R A P Q_0 = Q_0^* R Q Q^* A Q Q^* P Q_0$$
$$\implies \widehat{A}_0 = \widehat{R}\widehat{A}\widehat{P} = \frac{1}{H^2} \begin{pmatrix} 2\delta_0 + \frac{H^2}{3\varepsilon} - \cos(2\pi kH) & (-1)^j \left(1 - 2\delta_0 + \frac{H^2}{6\varepsilon} e^{i2\pi kH}\right) \\ (-1)^j \left(1 - 2\delta_0 + \frac{H^2}{6\varepsilon} e^{-i2\pi kH}\right) & 2\delta_0 + \frac{H^2}{3\varepsilon} - \cos(2\pi kH) \end{pmatrix}, \tag{20}$$

where $H = 2h$. We notice that the coarse operator is different for j even and j odd; however, the matrices obtained for both cases are similar, with similarity matrix $(-1)^j I$ where I is the identity matrix, and therefore have the same spectrum. In the rest of the paper we assume j is even, without loss of generality. This means that we will be studying a node that is present in both the coarse and fine meshes. We can now completely analyze the two grid iteration operator.

4.4. **Analysis of the two grid iteration operator.** The error reduction capabilities of Algorithm 1 are given by the spectrum of the iteration operator

$$E = (I - P A_0^{-1} R A)(I - \alpha D^{-1} A),$$

and we have shown that the 4-by-4 block Fourier-transformed operator

$$\widehat{E}(k) = (I - \widehat{P}(k)\widehat{A}_0^{-1}(k)\widehat{R}(k)\widehat{A}(k))(I - \alpha \widehat{D}^{-1}(k)\widehat{A}(k))$$

has the same spectrum. Then, we will focus on studying the spectral radius $\rho\left(\widehat{E}(k)\right)$ in the next section, in order to find the optimal relaxation parameter α_{opt}.

5. Study of optimal relaxation parameters

We begin by recalling the study performed by Hemker et al. [19] for the Poisson equation.

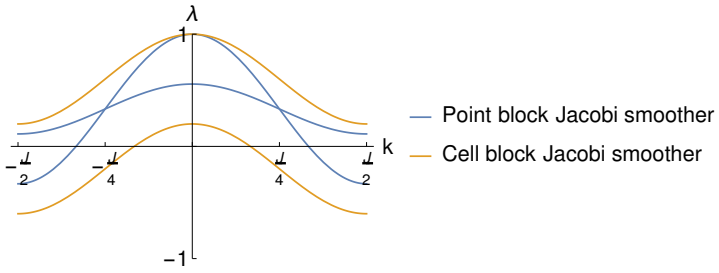


FIGURE 4. Spectrum of the *point* block-Jacobi and *cell* block-Jacobi smoothers for $\delta_0 = 2$, with optimized relaxation parameter without taking into account the coarse solver, following Hemker et al. in [19].

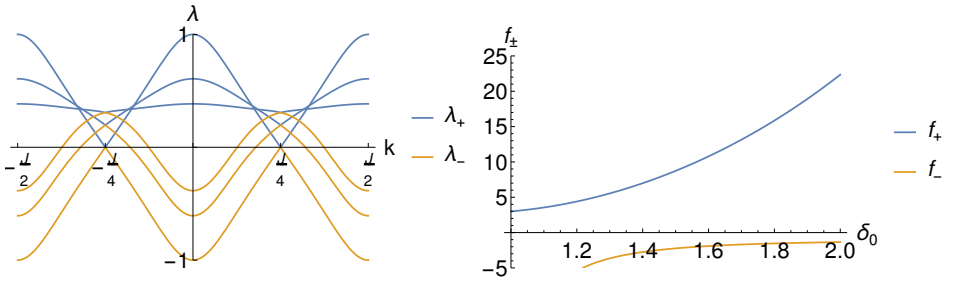
5.1. **Hemker et al. results.** In §4.1 of [19], a smoothing analysis is performed, which is an important first step in LFA studies. A comparison of the spectrum of the *point* block-Jacobi and *cell* block-Jacobi smoother with a relaxation parameter optimized only via a smoothing analysis is shown in Figure 4. The smoothing analysis predicts an optimal relaxation parameter 4/5 for the *point* block-Jacobi smoother, and 2/3 for the *cell* block-Jacobi smoother. We see that the smoothing capabilities of the *point* block-Jacobi smoother are better than the *cell* block-Jacobi smoother, since the upper half of the spectrum corresponding to the higher frequencies is better damped (equioscillation between $J/4$ and $J/2$).

In our study, we take into account the interaction of smoothing and coarse correction when optimizing the relaxation parameter, in order to get the best possible two level method, and we deduce explicit formulas for the relaxation parameter. We will show that, for DG penalization parameter values δ_0 lower than a certain threshold δ_c, which we determine explicitly, the *cell* block-Jacobi smoother of Schwarz type leads to a more efficient two-level method than the *point* block-Jacobi smoother. This threshold is higher than the frequently used DG penalization parameter value $\delta_0 = p(p+1) = 2$, where $p = 1$ here is the polynomial degree[5]. This shows that, for these penalization regimes, it is of interest in practice to use the *cell* block-Jacobi smoother instead of the *point* block-Jacobi smoother which looks preferable based on the smoothing analysis alone.

5.2. **Poisson equation.** We begin with the study of the Poisson equation, for which we can completely quantify the optimal choice of the relaxation parameter in the smoothing procedure to get the best error reduction in the two level algorithm. The best choice is characterized by equioscillation of the spectrum, in the sense that the absolute values of the maximum and minimum eigenvalues of the error reduction operator are equal, and is given in the following two Theorems.

Theorem 5.1 (Optimal *point* block-Jacobi two-level method)**.** *Let A be the first order, nodal, SIPG discretization matrix of a 1D Laplacian with periodic boundary conditions. The optimal relaxation parameter α_{opt}, in order to maximize the error*

[5]The value $\delta_0 = p(p+1)$ is used for example in the `deal.II` Finite Element Library [1] we will use in Subsection 6.5.



(A) λ_+ and λ_- for $\delta_0 = 1, 1.2, 2$ (in decreasing absolute value at $k = 0$) using α_{opt}.

(B) f_+ and f_-.

FIGURE 5

reduction of Algorithm 1, using a point *block-Jacobi smoother is given by*

$$\alpha_{opt} = \frac{(2\delta_0 - 1)^2}{6\delta_0^2 - 6\delta_0 + 1}. \tag{21}$$

Proof. We compute the spectrum of $\widehat{E}(k)$ and find its extrema for $-J/2 \le k \le J/2$. $\widehat{E}(k)$ has 4 eigenvalues, two of which are zero since the coarse operator is of rank 2. We focus on the non-zero eigenvalues λ_+ and λ_-, with $\lambda_+ \ge \lambda_-$, shown as function of k for several values of δ_0 in Figure 5a,

$$\lambda_\pm = 1 + \alpha \frac{-1 + 8\delta_0 - 10\delta_0^2 - \left(2\delta_0^2 - 4\delta_0 + 1\right) c_k \pm \sqrt{(c_k+1)(1-\delta_0)\,(c_k - f_-)\,(c_k - f_+)}}{(2\delta_0 - 1)(4\delta_0 - c_k - 1)}, \tag{22}$$

where $c_k = \cos\left(\frac{4\pi k}{J}\right)$ contains the dependence on k, and

$$f_\pm(\delta_0) = \frac{1 - 6\delta_0 + 8\delta_0^2 - 8\delta_0^3 + 4\delta_0^4 \pm \sqrt{1 - 8\delta_0 + 16\delta_0^2 - 48\delta_0^3 + 120\delta_0^4 - 160\delta_0^5 + 128\delta_0^6 - 64\delta_0^7 + 16\delta_0^8}}{2(\delta_0 - 1)}.$$

The function $f_\pm(\delta_0)$ satisfies the following properties for $\delta_0 \ge 1$, as one can see from a direct computation (see Figure 5b):

(1) $f_+(\delta_0)$ is monotonically increasing, $\lim_{\delta_0 \to 1} f_+(\delta_0) = 3$ and $\lim_{\delta_0 \to \infty} f_+(\delta_0) \to \infty$, therefore $(c_k - f_+(\delta_0)) < 0$;
(2) $f_-(\delta_0)$ is monotonically increasing, $\lim_{\delta_0 \to 1} f_-(\delta_0) \to -\infty$ and $\lim_{\delta_0 \to \infty} f_-(\delta_0) = -1$, therefore $(c_k - f_+(\delta_0)) > 0$;
(3) $1 - \delta_0 \le 0$ and $c_k + 1 \ge 0$, and thus with (1) and (2) we have $(c_k + 1)(1 - \delta_0)\,(c_k - f_-(\delta_0))\,(c_k - f_+(\delta_0)) \ge 0$, and therefore $\lambda_\pm(\delta_0) \in \mathbb{R}$;
(4) $\lim_{\delta_0 \to 1}(c_k+1)(1-\delta_0)\,(c_k - f_-(\delta_0))\,(c_k - f_+(\delta_0)) = (c_k+1)\,(3 - c_k)$, therefore $\lambda_+(\delta_0) = \lambda_-(\delta_0) \iff c_k = -1$, i.e. $k = J/4$.

In order to obtain the extrema of $\lambda_\pm$ in k, we need to study $\frac{\partial \lambda_\pm}{\partial k}$, and since $\frac{\partial \lambda_\pm}{\partial k} = \frac{\partial \lambda_\pm}{\partial c_k}\frac{\partial c_k}{\partial k}$, we first compute

$$\frac{\partial \lambda_\pm}{\partial c_k} =$$
$$\alpha\Big[-1+9\delta_0-28\delta_0^2+64\delta_0^3-64\delta_0^4+32\delta_0^5+\left(-3+23\delta_0+64\delta_0^3-64\delta_0^4-56\delta_0^2+32\delta_0^5\right)c_k$$
$$+\left(-3+15\delta_0-12\delta_0^2\right)c_k^2+(\delta_0-1)c_k^3\pm 16(1-\delta_0)\delta_0^2\sqrt{(c_k+1)(1-\delta_0)\left(c_k-f_-\right)\left(c_k-f_+\right)}\Big]\Big/$$
$$\left(\pm 2(2\delta_0-1)(-4\delta_0+c_k+1)^2\sqrt{(c_k+1)(1-\delta_0)\left(c_k-f_-\right)\left(c_k-f_+\right)}\right).$$

We begin by looking for zeros of the numerator; separating the term with the square root and squaring both sides of the equation leads to

$$(-4\delta_0+c_k+1)^2$$
$$\Big[1-10\delta_0+41\delta_0^2-144\delta_0^3+256\delta_0^4-192\delta_0^5+64\delta_0^6$$
$$+\left(128\delta_0^6-384\delta_0^5+512\delta_0^4-368\delta_0^3+148\delta_0^2-40\delta_0+4\right)c_k$$
$$+\left(64\delta_0^6-192\delta_0^5+256\delta_0^4-240\delta_0^3+158\delta_0^2-52\delta_0+6\right)c_k^2$$
$$+\left(-16\delta_0^3+36\delta_0^2-24\delta_0+4\right)c_k^3+\left(\delta_0^2-2\delta_0+1\right)c_k^4\Big]=0.$$

This operation might add spurious roots to the original expression, so we analyze them individually. The left hand side is a product of two factors, the second of which is a 4th degree polynomial in c_k. The application of the Cardano-Tartaglia formula leads to complex roots for $\delta_0 \geq 1$, leaving only two real roots coming from the first factor, both at $c_k = -1+4\delta_0$, but $\delta_0 \geq 1$ and $|c_k| \leq 1$, so there is no real root of $\frac{\partial \lambda_\pm}{\partial c_k}$. We deduce that $\frac{\partial \lambda_\pm}{\partial k}$ is zero only where $\frac{\partial c_k}{\partial k} = 0$, i.e., $k = J/4, J/2$.

We remark at this point that because the dependency on k is contained in c_k, the eigenvalues at $k = 0$ will be the same than at $k = J/2$, so it suffices to consider only the case $k = J/2$.

We realize as well that the denominator vanishes for $c_k = -1$ (i.e. $k = J/4$), and for the derivative when approaching this value, we get $\lim_{k\to J/4}\frac{\partial \lambda_\pm}{\partial k} = \lim_{k\to J/4}\frac{\partial \lambda_\pm}{\partial c_k}\frac{\partial c_k}{\partial k}$; multiplying and dividing by the factor $\sqrt{(c_k+1)(1-\delta_0)\left(c_k-f_-\right)\left(c_k-f_+\right)}$ we obtain

$$\lim_{k\to J/4}\frac{\partial \lambda_\pm}{\partial k} = \lim_{k\to J/4}\frac{\frac{\partial c_k}{\partial k}}{\sqrt{(c_k+1)(1-\delta_0)\left(c_k-f_-\right)\left(c_k-f_+\right)}}\lim_{k\to J/4}\frac{\partial \lambda_\pm}{\partial c_k}\sqrt{(c_k+1)(1-\delta_0)\left(c_k-f_-\right)\left(c_k-f_+\right)}$$
$$=\begin{cases}\dfrac{2\sqrt{2}\pi}{\sqrt{\delta_0}J}\lim\limits_{k\to J/4}\dfrac{\partial \lambda_\pm}{\partial c_k}\sqrt{(c_k+1)(1-\delta_0)\left(c_k-f_-\right)\left(c_k-f_+\right)}, & k\to (J/4)^+,\\ -\dfrac{2\sqrt{2}\pi}{\sqrt{\delta_0}J}\lim\limits_{k\to J/4}\dfrac{\partial \lambda_\pm}{\partial c_k}\sqrt{(c_k+1)(1-\delta_0)\left(c_k-f_-\right)\left(c_k-f_+\right)}, & k\to (J/4)^-,\end{cases}$$
$$=\begin{cases}\pm\dfrac{\sqrt{2}\alpha\pi}{(2\delta_0-1)\sqrt{\delta_0}J}, & k\to (J/4)^+,\\ \mp\dfrac{\sqrt{2}\alpha\pi}{(2\delta_0-1)\sqrt{\delta_0}J}, & k\to (J/4)^-,\end{cases}$$

therefore at $k = J/4$, λ_+ has a minimum and λ_- has a maximum as observed in Fig. 5a.

In order to determine if the extremum at $k = J/2$ is a minimum or a maximum, we compute the second derivative,

$$\left.\frac{\partial^2 \lambda_+}{\partial k^2}\right|_{k=J/2} = \frac{8\pi^2\alpha(1-2\delta_0(2(\delta_0-2)\delta_0+3))}{(2\delta_0-1)^3(2(\delta_0-1)\delta_0+1)J^2} < 0 \iff 1-6\delta_0+8\delta_0^2-4\delta_0^3 < 0,$$

which always holds for $\delta_0 \geq 1$, and thus at $k = J/2$, λ_+ has a maximum. Similarly, for λ_-, we find

$$\left.\frac{\partial^2 \lambda_-}{\partial k^2}\right|_{k=J/2} = \frac{8\pi^2\alpha(2\delta_0(2(\delta_0-1)\delta_0+1)-1)}{(2\delta_0(\delta_0(2\delta_0-3)+2)-1)J^2} < 0 \iff -1+2\delta_0-4\delta_0^2+4\delta_0^3 < 0,$$

which never holds for $\delta_0 \geq 1$, and thus at $k = J/2$, λ_- has a minimum, as we can see in Fig. 5a.

To minimize the spectral radius, due to the monotonicity of the eigenvalues in the parameter α, we can minimize the absolute value of $\lambda_\pm$ by just centering the eigenvalue distribution around zero. Using the explicit formulas for the extrema, this is achieved by equioscillation when the relaxation parameter α_{opt} satisfies $\lambda_+\big|_{k=J/2} = -\lambda_-\big|_{k=J/2}$, which gives (21). □

Theorem 5.2 (Optimal *cell* block-Jacobi two-level method)**.** *Let A be the first order, nodal, SIPG discretization matrix of a 1D Laplacian with periodic boundary conditions. The optimal relaxation parameter α_{opt}, in order to maximize the error reduction of Algorithm 1 using a* cell *block-Jacobi smoother is given by*

$$\alpha_{opt} = \begin{cases} \frac{\delta_0(2\delta_0-1)}{2\delta_0^2-1}, & \text{for } 1 \leq \delta_0 \leq \widetilde{\delta}_{0+}, \\ \frac{2\delta_0^2(2\delta_0-1)}{\delta_0\left|2\delta_0^2-4\delta_0+1\right|+2\delta_0^3+4\delta_0^2-5\delta_0+1}, & \text{for } \widetilde{\delta}_{0+} \leq \delta_0 \leq \widetilde{\delta}_{0-}, \\ \frac{2\delta_0^2}{2\delta_0^2+\delta_0-1}, & \text{for } \widetilde{\delta}_{0-} \leq \delta_0, \end{cases}$$

where $\widetilde{\delta}_{0+} = \frac{1}{12}\left(8 + \sqrt[3]{152-24\sqrt{33}} + 2\sqrt[3]{19+3\sqrt{33}}\right) = 1.41964\ldots$ *and* $\widetilde{\delta}_{0-} = 3/2$.

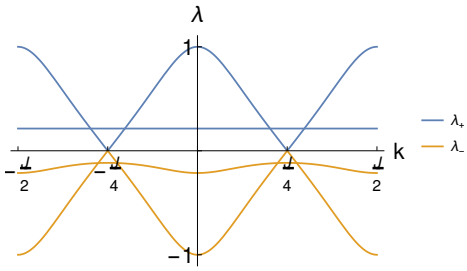
Proof. As in the proof of Theorem 5.1, we compute the spectrum of $\widehat{E}(k)$ and find its extrema for $-J/2 \leq k \leq J/2$. Again $\widehat{E}(k)$ has 4 eigenvalues, two of which are zero.

The non-zero eigenvalues λ_+ and λ_- are real, with $\lambda_+ \geq \lambda_-$, and are given by

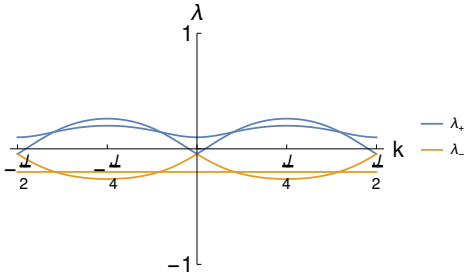
$$\lambda_\pm = 1 + \alpha\left(\frac{2+\delta_0\left(c_k-4\delta_0-1\right) \pm \sqrt{\left(\delta_0^2-2\right)\left(c_k-f_-\right)\left(c_k-f_+\right)}}{\delta_0\left(4\delta_0-c_k-1\right)}\right), \tag{23}$$

where $c_k = \cos\left(\frac{4\pi k}{J}\right)$ and $f_\pm(\delta_0) = \frac{\delta_0\left(4\delta_0^2-7\delta_0+2\right)\pm 2\sqrt{(2\delta_0-3)(4\delta_0^3-8\delta_0^2+4\delta_0-1)}}{\delta_0^2-2}$, (see Figs. 6a, 6b and 6c). A direct computation shows for $\delta_0 \geq 1$ that (see Fig. 6d)

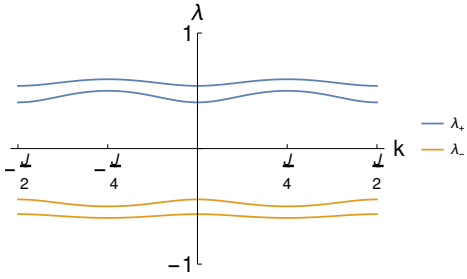
(1) $f_+ = -1 \iff \delta_0 = 1$,
(2) $f_- = 1 \iff \delta_0 = \frac{2+\sqrt{2}}{2}$,
(3) $f_\pm \notin \mathbb{R} \iff \delta_0 \in (\sqrt{2}, \frac{2+\sqrt{2}}{2})$,
(4) elsewhere $|f_\pm| > 1$.



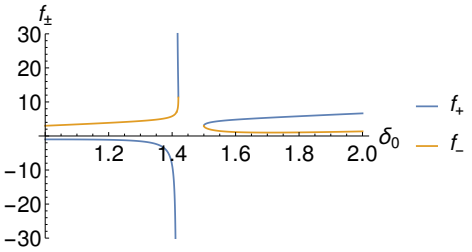
(A) λ_+ and λ_- for $\delta_0 = 1, \widetilde{\delta}_{0+}$ (in decreasing absolute value at $k = 0$) using α_{opt}.



(B) λ_+ and λ_- for $\delta_0 = \widetilde{\delta}_{0-}, \frac{2+\sqrt{2}}{2}$ (in decreasing absolute value at $k = 0$) using α_{opt}.



(C) λ_+ and λ_- for $\delta_0 = 3, 4$ (in increasing absolute value at $k = 0$) using α_{opt}.



(D) f_+ and f_-.

FIGURE 6

To find the extrema of $\lambda_\pm$ in k, we compute again the derivative $\frac{\partial \lambda_\pm}{\partial k} = \frac{\partial \lambda_\pm}{\partial c_k}\frac{\partial c_k}{\partial k}$ and obtain

$$\frac{\partial \lambda_\pm}{\partial c_k} = \alpha \frac{6 - 26\delta_0 + 50\delta_0^2 - 24\delta_0^3 + \left(6\delta_0^2 - 10\delta_0 + 2\right) c_k \mp \sqrt{4(\delta_0 - 1)^2 \left(\delta_0^2 - 2\right)\left(c_k - f_-\right)\left(c_k - f_+\right)}}{\pm\delta_0(-4\delta_0 + c_k + 1)^2 \sqrt{\left(\delta_0^2 - 2\right)\left(c_k - f_-\right)\left(c_k - f_+\right)}}. \tag{24}$$

We now look for roots of the numerator

$$6 - 26\delta_0 + 50\delta_0^2 - 24\delta_0^3 + \left(6\delta_0^2 - 10\delta_0 + 2\right) c_k \mp \sqrt{4(\delta_0 - 1)^2 \left(\delta_0^2 - 2\right)\left(c_k - f_-\right)\left(c_k - f_+\right)} = 0. \tag{25}$$

We first note that if $f_- = f_+ = f$, i.e. $(2\delta_0 - 3)(4\delta_0^3 - 8\delta_0^2 + 4\delta_0 - 1) = 0$, we have

$$6 - 26\delta_0 + 50\delta_0^2 - 24\delta_0^3 \pm f\sqrt{4(\delta_0 - 1)^2 \left(\delta_0^2 - 2\right)} + \left(6\delta_0^2 - 10\delta_0 + 2 \mp \sqrt{4(\delta_0 - 1)^2 \left(\delta_0^2 - 2\right)}\right) c_k = 0.$$

The factor multiplying the c_k has roots,

$$6\delta_0^2 - 10\delta_0 + 2 \mp \sqrt{4(\delta_0 - 1)^2 \left(\delta_0^2 - 2\right)} = 0 \tag{26}$$

$$\begin{aligned}
&\implies \left(-6\delta_0^2+10\delta_0-2\right)^2 = 4(\delta_0-1)^2\left(\delta_0^2-2\right)\\
&\iff 8\delta_0^4-28\delta_0^3+32\delta_0^2-14\delta_0+3=0\\
&\iff (2\delta_0-3)(4\delta_0^3-8\delta_0^2+4\delta_0-1)=0,
\end{aligned}$$

where we might have added spurious roots to the original expression by squaring both sides, so we analyze them individually. We see that this is the same condition for $f_- = f_+ = f$. There are, therefore, $\widetilde{\delta}_{0\pm}$ such that $\frac{\partial\lambda_\pm}{\partial k} = 0$ independently of k. Such $\widetilde{\delta}_{0\pm}$ are found by obtaining the real roots of the polynomial from equation (26),

$$\begin{aligned}
\widetilde{\delta}_{0+} &= \frac{1}{12}\left(8+\sqrt[3]{152-24\sqrt{33}}+2\sqrt[3]{19+3\sqrt{33}}\right) = 1.41964\ldots,\\
\widetilde{\delta}_{0-} &= \frac{3}{2}.
\end{aligned}$$

We now take equation (25) and compute the roots with respect to c_k,

$$\begin{aligned}
&\left(6-26\delta_0+50\delta_0^2-24\delta_0^3+\left(6\delta_0^2-10\delta_0+2\right)c_k\right)^2 =\\
&\qquad 4(\delta_0-1)^2\left(\delta_0^2-2\right)(c_k-f_-)(c_k-f_+);
\end{aligned}$$

a simplification gives

$$c_k^2+(2-8\delta_0)c_k+\left(16\delta_0^2-8\delta_0+1\right)=0,$$

which has two roots that are equal to $c_k = -1+4\delta_0$, but $\delta_0 \geq 1$, so there is no real root of $\frac{\partial\lambda_\pm}{\partial c_k}$. We deduce from this and the chain rule, that $\frac{\partial\lambda_\pm}{\partial k}$ is zero only where $\frac{\partial c_k}{\partial k} = 0$, hence the roots are located at $k = J/4, J/2$ (i.e. $c_k = 1, -1$), except when λ_+ or λ_- do not depend on k.

We remark at this point that because the dependency on k is contained in c_k, the eigenvalues at $k = 0$ will be the same than at $k = J/2$. In what follows, we will only analyze the case $k = J/2$.

We see that the denominator of (24) has roots at

(1) $\delta_0 = \sqrt{2}$, but given that $|c_k| \leq 1$ we have

$$\lim_{\delta_0\to\sqrt{2}}\left(\delta_0^2-2\right)(c_k-f_-)(c_k-f_+) = -4(-50+35\sqrt{2}+(-7+5\sqrt{2})c_k) \neq 0;$$

since $f_\pm$ contains the term (δ_0^2-2) in the denominator.

(2) $\delta_0 = 1, c_k = -1$ i.e. $k = J/4$,

$$\begin{aligned}
\lim_{\substack{\delta_0\to 1\\ k\to J/4}}\frac{\partial\lambda_\pm}{\partial k} &= \lim_{\substack{\delta_0\to 1\\ k\to J/4}}\frac{\partial\lambda_\pm}{\partial c_k}\frac{\partial c_k}{\partial k} = \pm\frac{2\alpha}{(3-c_k)^{\frac{3}{2}}\sqrt{1+c_k}}\left(-\frac{4\pi s_k}{J}\right)\\
&= \begin{cases}\mp\frac{\sqrt{2}\alpha\pi}{J}, & k\to(J/4)^+\\ \pm\frac{\sqrt{2}\alpha\pi}{J}, & k\to(J/4)^-\end{cases},
\end{aligned}$$

where $s_k = \sin\left(\frac{4\pi k}{J}\right)$, hence there is a minimum for λ_+ and a maximum for λ_-;

(3) $\delta_0 = \frac{2+\sqrt{2}}{2}, c_k = 1$, where

$$\lim_{\substack{\delta_0\to 1\\ k\to J/2}}\frac{\partial\lambda_\pm}{\partial k} = \lim_{\substack{\delta_0\to 1\\ k\to J/2}}\frac{\partial\lambda_\pm}{\partial c_k}\frac{\partial c_k}{\partial k}$$

$$= \lim_{k\to J/2}\left(\frac{2\alpha}{c_k-3-2\sqrt{2}}\left(1-\sqrt{2}\pm\frac{c_k-5}{\sqrt{(c_k-1)\left(\left(2\sqrt{2}-1\right)c_k-7-2\sqrt{2}\right)}}\right)\right)\left(-\frac{4\pi s_k}{J}\right)$$

$$= \begin{cases} \pm\frac{4\alpha\pi(2-\sqrt{2})}{(2+\sqrt{2})J}, & k\to(J/2)^+ \\ \mp\frac{4\alpha\pi(2-\sqrt{2})}{(2+\sqrt{2})J}, & k\to(J/2)^- \end{cases},$$

therefore it is a minimum for λ_+ and a maximum for λ_-.

Thus, in the following we will assume that $\delta_0 \neq 1$ and $\delta_0 \neq \frac{2+\sqrt{2}}{2}$.

In order to determine if the extremum at $k = J/4$ is a minimum or a maximum we compute the second derivative,

$$\left.\frac{\partial^2\lambda_+}{\partial k^2}\right|_{k=J/4} < 0 \iff \frac{4\pi^2\alpha\left(1-4\delta_0+8\delta_0^2-4\delta_0^3\right)}{\delta_0^3J^2(\delta_0-1)\left(2\delta_0-1\right)} < 0 \iff 1-4\delta_0+8\delta_0^2-4\delta_0^3 < 0.$$

The only real root of this polynomial is $\widetilde{\delta_{0+}}$, and we conclude that at $k = J/4$, for $\delta_0 < \widetilde{\delta_{0+}}$, λ_+ has a minimum, and conversely, for $\delta_0 > \widetilde{\delta_{0+}}$ it has a maximum. For the second eigenvalue, we get

$$\left.\frac{\partial^2\lambda_-}{\partial k^2}\right|_{k=J/4} < 0 \iff \frac{4\pi^2\alpha\left(2\delta_0-3\right)}{\delta_0J^2(\delta_0-1)\left(2\delta_0-1\right)} < 0 \iff 2\delta_0-3 < 0,$$

and we conclude that at $k = J/4$, for $\delta_0 < \widetilde{\delta_{0-}}$, λ_- has a maximum, and conversely, for $\delta_0 > \widetilde{\delta_{0-}}$ it has a minimum.

Similarly, at $k = J/2$, we find

$$\left.\frac{\partial^2\lambda_+}{\partial k^2}\right|_{k=J/2} < 0$$

$$\iff \frac{8\pi^2\alpha\left(\frac{(2\delta_0-1)(d(2\delta_0(3\delta_0-7)+9)-2)}{|1-2(\delta_0-1)\delta_0(2\delta_0-3)|}+\delta_0-1\right)}{\delta_0(J-2dJ)^2} < 0$$

$$\iff 2-13\delta_0+32\delta_0^2-34\delta_0^3+12\delta_0^4+(\delta_0-1)\left|-4\delta_0^3+10\delta_0^2-6\delta_0+1\right| < 0$$

$$\iff \begin{cases} -1+4\delta_0-8\delta_0^2+4\delta_0^3 < 0 & \text{if } \delta_0 < \frac{2+\sqrt{2}}{2}, \\ -2+9\delta_0-14\delta_0^2+6\delta_0^3 < 0 & \text{if } \delta_0 = \frac{2+\sqrt{2}}{2}, \\ 2\delta_0-3 < 0 & \text{if } \delta_0 > \frac{2+\sqrt{2}}{2}, \end{cases}$$

$$\iff -1+4\delta_0-8\delta_0^2+4\delta_0^3 < 0,$$

and we conclude that at $k = J/2$, for $\delta_0 < \widetilde{\delta_{0+}}$, λ_+ has a maximum, and conversely, for $\delta_0 > \widetilde{\delta_{0+}}$ it has a minimum. And finally,

$$\left.\frac{\partial^2\lambda_-}{\partial k^2}\right|_{k=J/2} < 0$$

$$\iff -2+13\delta_0-32\delta_0^2+34\delta_0^3-12\delta_0^4+(\delta_0-1)\left|-4\delta_0^3+10\delta_0^2-6\delta_0+1\right| < 0$$

$$\iff \begin{cases} 3-2\delta_0 < 0 & \text{if } \delta_0 < \frac{2+\sqrt{2}}{2}, \\ 2-9\delta_0+14\delta_0^2-6\delta_0^3 < 0 & \text{if } \delta_0 = \frac{2+\sqrt{2}}{2}, \\ 1-4\delta_0+8\delta_0^2-4\delta_0^3 < 0 & \text{if } \delta_0 > \frac{2+\sqrt{2}}{2}, \end{cases}$$

$$\iff 3-2\delta_0 < 0,$$

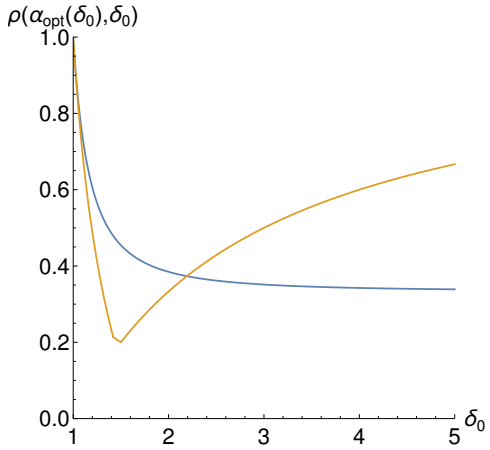


FIGURE 7. Spectral radius $\rho\left(\alpha_{\text{opt}}(\delta_0), \delta_0\right)$ of the iteration operator of Algorithm 1 using an optimal relaxation parameter, for a *point* block-Jacobi smoother (blue) and a *cell* block-Jacobi smoother (orange) as function of the penalization parameter δ_0.

and we conclude that at $k = J/2$, for $\delta_0 > \widetilde{\delta}_{0-}$, λ_- has a maximum, and conversely, for $\delta_0 < \widetilde{\delta}_{0-}$ it has a minimum.

In order to minimize the spectral radius we have to center again the eigenvalue distribution around zero, using the explicit formulas developed above. The result thus follows from the solution of

$$\begin{cases} \lambda_+\big|_{k=J/2} = -\lambda_-\big|_{k=J/2}, & \text{for } 1 \le \delta_0 \le \widetilde{\delta}_{0+}, \\ \lambda_+\big|_{k=J/4} = -\lambda_-\big|_{k=J/2}, & \text{for } \widetilde{\delta}_{0+} \le \delta_0 \le \widetilde{\delta}_{0-}, \\ \lambda_+\big|_{k=J/4} = -\lambda_-\big|_{k=J/4}, & \text{for } \widetilde{\delta}_{0-} \le \delta_0. \end{cases}$$

□

Figure 7 shows the contraction factor as function of the penalization parameter δ_0 for the *point* block-Jacobi and *cell* block-Jacobi two-level methods using the best relaxation parameter α_{opt} from Theorem 5.1 and 5.2. We see that the *cell* block-Jacobi smoother outperforms the *point* block-Jacobi smoother for values of $\delta_0 \le \delta_c = 1 + \frac{1}{6}\sqrt[3]{54 - 6\sqrt{33}} + \sqrt[3]{\frac{1}{4} + \frac{\sqrt{33}}{36}} \approx 2.19149$. For larger penalization parameters δ_0 the *point* block-Jacobi two-level method converges faster. This can be understood intuitively as follows: the more we penalize the jumps, the more important the face terms in the bilinear form become and, after a threshold, a preconditioner that takes into account all the terms containing this penalization begins performing better than a preconditioner which does not.

Note that we put explicitly parameter dependencies of the spectral radius throughout our manuscript to emphasize the variables we are interested in, and not all the dependencies; for instance $\rho(\alpha)$ does not imply that ρ depends only on alpha, but that we are interested in the α dependency in the specific figure/context.

It should be noted that even though large values of δ_0 are a better choice when using the *point* block-Jacobi smoother, this also means that the discretization of the

coarse space will be harder to invert, since according to equation (20) the penalty is doubled.

We can also observe that we obtain the best performance for $\delta_0 = \delta_{0-} = \frac{3}{2}$, shown in Figure 7 as the minimum of the orange curve. This shows that the penalization parameter in SIPG has a direct influence on the two-level solver, and there is an optimal choice $\delta_0 = \delta_{0-}$ for best performance. Choosing other values for δ_0 can make the solver slower by an order of magnitude, even if the best relaxation parameter is chosen! It would therefore be of interest to lower this value in software packages, see also footnote 5.

5.3. **Reaction-diffusion equation.** We now use LFA to study the more general reaction-diffusion case. The computations become substantially more involved, but we will still be able to *center* the spectrum to derive relaxation parameter values that lead to very effective two-level methods, even though we can not formally prove optimality as in the simpler case of the Poisson equation in the previous subsection. We will however provide numerical evidence for the optimality in Section 6. For the reaction-diffusion case, we see from the elements in the matrices shown in §4.1 that the key physical parameter is

$$\gamma := \frac{\varepsilon}{h^2} = \varepsilon J^2. \tag{27}$$

When ε becomes small, i.e. the reaction dominated case, the mesh size needs to resolve boundary layers, and we then need $h \sim \sqrt{\varepsilon}$ [15, §1.3.2] (see also [23] and references therein), which implies that γ is of order 1. When ε is not small however, the mesh size does not depend on ε, and thus γ can become large. We therefore need a two-level method which is robust for a large range of physical values γ.

5.3.1. Point *block-Jacobi smoother.* By direct calculation, the eigenvalues of the iteration operator of Algorithm 1 for the reaction-diffusion equation case using a *point* block-Jacobi smoother are of the form

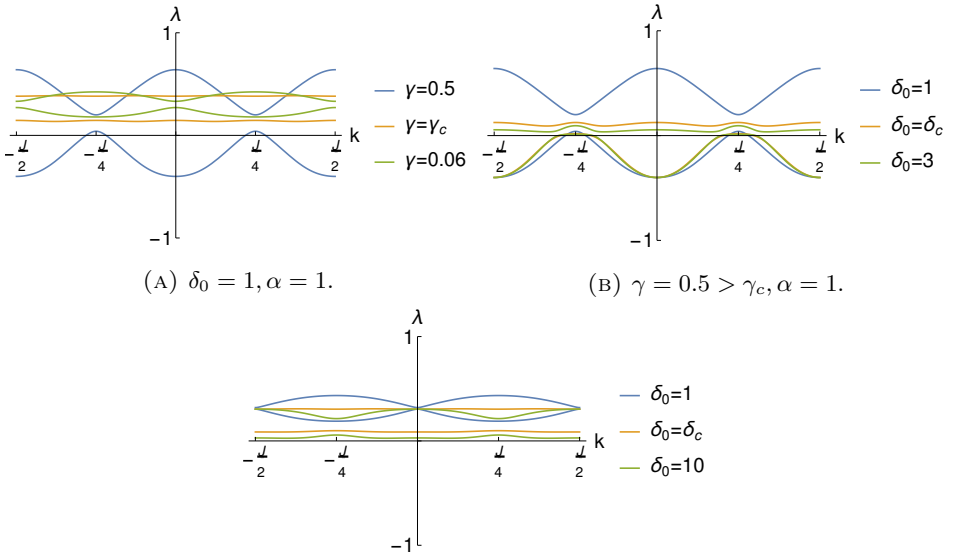
$$\lambda_\pm = \frac{c_1 + c_2 x + c_3 x^2 \pm \sqrt{c_4 + c_5 x + c_6 x^2 + c_7 x^3 + c_8 x^4 + c_9 x^5}}{c_{10} + c_{11} x + c_{12} x^2}, \tag{28}$$

where $x = \cos\left(\frac{4\pi k}{J}\right)$, and the $c_1, \ldots, c_{12}$, depending on δ_0, are defined in Appendix A. Figure 8a shows the spectrum for penalization parameter $\delta_0 = 1$. We see that there is a threshold on the physical parameter γ where the frequency k, at which the maximum absolute value of the eigenvalues determining the spectral radius occurs, changes from $J/2$ to $J/4$. The critical γ can be computed by solving $\lambda_+(\gamma)\big|_{k=J/2} = \lambda_+(\gamma)\big|_{k=J/4}$, and it is given by

$$\gamma_c(\delta_0) = \frac{1}{3\left(\sqrt{4(\delta_0 - 1)\delta_0 + 5} + (3 - 2\delta_0)\right)}. \tag{29}$$

Similarly, Figures 8b and 8c show the spectrum for $\gamma = 0.5$ and $\gamma = 0.05$. We see that there is a threshold on δ_0 where the frequency k, at which the maximum absolute value of λ_+ occurs, changes from $J/2$ to $J/4$. The critical δ_0 can be computed as well by solving $\lambda_+(\delta_0)\big|_{k=J/2} = \lambda_+(\delta_0)\big|_{k=J/4}$, and it is given by

$$\delta_c^+ = \frac{-5 + 9\gamma\left(6\gamma^2 + 8\gamma + 1\right) + \sqrt{(3\gamma+1)\left(3\gamma\left(12\gamma\left(3\gamma\left(3\gamma(3\gamma+7)+20\right)+25\right)+53\right)+10\right)}}{6\gamma(12\gamma+5)} \tag{30}$$



(A) $\delta_0 = 1, \alpha = 1$.

(B) $\gamma = 0.5 > \gamma_c, \alpha = 1$.

(C) $\gamma = 0.05 < \gamma_c, \alpha = 1$.

FIGURE 8. Spectrum of the iteration operator of algorithm (1) using a *point* block-Jacobi smoother for a varying stabilization parameter δ_0 of the SIPG method and reaction scaling γ.

for $\gamma > \gamma_c$, and

$$\delta_c^- = \frac{1 + 2\gamma\,(6\gamma - 11) - \sqrt{4\gamma\,(2\gamma+1)\,(3\,(6\gamma+7)+1)+1}}{8\gamma\,(6\gamma-1)} \tag{31}$$

for $\gamma \leq \gamma_c$. This allows us to obtain α_{opt} for different regimes, again using the equioscillation principle, which we rigorously proved for the Laplace case to obtain Theorem 5.1 and Theorem 5.2, but which we can only observe numerically in the more complex singularly perturbed reaction diffusion case here to minimize the spectral radius: the equations to be solved for equioscillation are

$$\begin{cases} \lambda_+\big|_{k=\frac{J}{4}} + \lambda_-\big|_{k=\frac{J}{4}} = 0 & \text{for } \gamma \leq \gamma_c, \delta_0 \leq \delta_c^-, \\ \lambda_+\big|_{k=\frac{J}{2}} + \lambda_-\big|_{k=\frac{J}{2}} = 0 & \text{for } \gamma \leq \gamma_c, \delta_0 > \delta_c^- \text{ or } \gamma > \gamma_c, \delta_0 \leq \delta_c^+, \\ \lambda_+\big|_{k=\frac{J}{4}} + \lambda_-\big|_{k=\frac{J}{2}} = 0 & \text{for } \gamma > \gamma_c, \delta_0 > \delta_c^+, \end{cases} \tag{32}$$

which leads to the corresponding relaxation parameters that equioscillate,

$$\alpha_{\text{opt}} = \begin{cases} \frac{8(3\gamma+1)(2\delta_0\gamma+1)(3(2\delta_0-1)\gamma+1)}{(12\delta_0\gamma+5)(12(2\delta_0-1)\gamma^2+8\delta_0\gamma+1)}, & \text{for } \gamma \leq \gamma_c, \delta_0 \leq \delta_c^-, \\ \frac{8(3\gamma+1)(3(2\delta_0-1)\gamma+1)^2}{(6\gamma+1)(9\gamma(4(6(\delta_0-1)\delta_0+1)\gamma+8\delta_0-5)+5)}, & \text{for } \gamma \leq \gamma_c, \delta_0 > \delta_c^- \\ & \text{or } \gamma > \gamma_c, \delta_0 \leq \delta_c^+, \\ \frac{4(3\gamma+1)(2\delta_0\gamma+1)(3(2\delta_0-1)\gamma+1)}{\gamma(108\delta_0(2\delta_0-1)\gamma^2+6(\delta_0(6\delta_0+19)-8)\gamma+19\delta_0+9)+2}, & \text{for } \gamma > \gamma_c, \delta_0 > \delta_c^+. \end{cases} \tag{33}$$

Figure 9 shows the behavior of α_{opt} and the corresponding convergence factor of the two-level method as a function of δ_0 for several values of the reaction scaling $\gamma = \frac{\varepsilon}{h^2}$. Note that $\lim_{\gamma\to\infty} \delta_c^+ \to \infty$ and $\lim_{\gamma\to\infty} \alpha_{\text{opt}} \to \frac{(2\delta_0-1)^2}{6\delta_0^2-6\delta_0+1}$ (from the second expression), which is consistent with Theorem 5.1. We see from the right

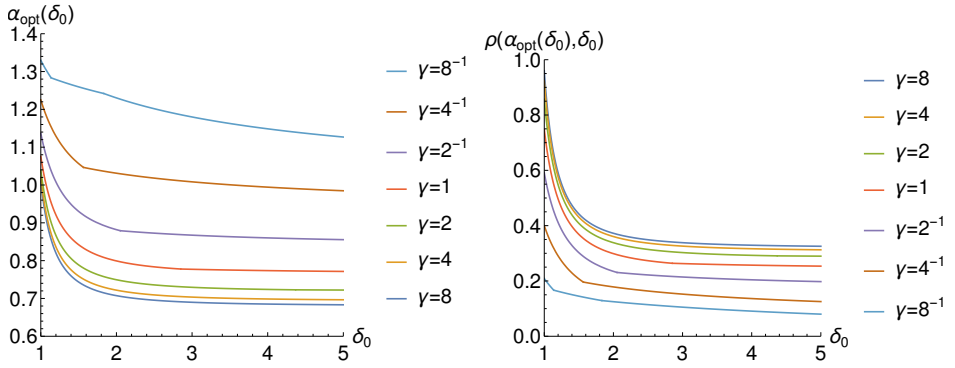


FIGURE 9. Optimized relaxation parameter $\alpha_{\text{opt}}(\delta_0)$ and corresponding convergence factor of Algorithm 1 using a *point* block-Jacobi smoother as function of the stabilization parameter δ_0 of the SIPG method for different reaction scalings $\gamma = \frac{\varepsilon}{h^2}$.

plot in Figure 9 that the *point* block-Jacobi two-level method is convergent for all $\delta_0 > 1$ with the optimized choice α_{opt}, and the convergence factor remains below about 0.4 for penalization δ_0 above 2, even when the reaction scaling γ becomes large, so the method is robust for large γ. We also see from the left plot in Figure 9 that overrelaxation is needed (i.e. $\alpha_{\text{opt}} > 1$), for typical values of δ_0 around 2, when γ becomes small, but for γ large we need underrelaxation (i.e. $\alpha_{\text{opt}} < 1$).

5.3.2. Cell *block-Jacobi smoother*. By direct calculation, the eigenvalues of the iteration operator of Algorithm 1 for the reaction-diffusion equation case using a *cell* block-Jacobi smoother are of the form

$$\lambda_{\pm} = \frac{c_1 + c_2 x + c_3 x^2 \pm \sqrt{c_4 + c_5 x + c_6 x^2 + c_7 x^3 + c_8 x^4}}{c_9 + c_{10} x + c_{11} x^2}, \tag{34}$$

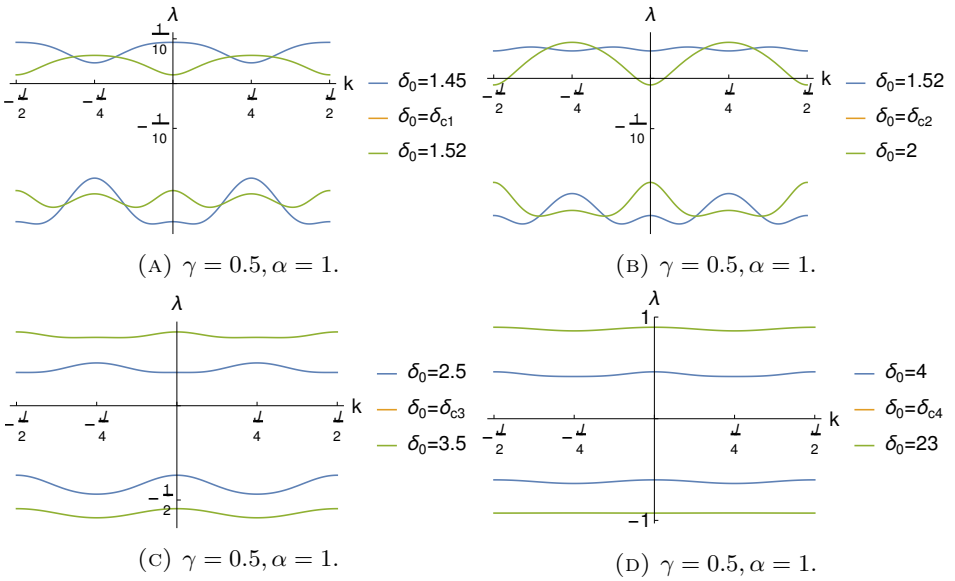
where $x = \cos\left(\frac{4\pi k}{J}\right)$, and the $c_1, \ldots, c_{11}$, depending on δ_0, are defined in Appendix B. Figures 10a, 10b, 10c and 10d show the spectrum of the iteration operator of Algorithm 1 for $\gamma = \frac{1}{2}$. We can see that, in contrast to the case of the Poisson equation, the maxima and minima are not located only at $0, J/4, J/2$, however we approximate the behavior optimizing by considering only the values at $0, J/4, J/2$. Therefore, in order to equioscillate the spectrum we see that the following equations need to hold:

$$\begin{cases} \lambda_+\big|_{k=\frac{J}{2}} + \lambda_-\big|_{k=\frac{J}{2}} = 0, & \text{for } \delta_0 \leq \delta_{c1} \text{ or } \delta_0 \geq \delta_{c4}, \\ \lambda_+\big|_{k=\frac{J}{4}} + \lambda_-\big|_{k=\frac{J}{2}} = 0, & \text{for } \delta_0 \leq \delta_{c2}, \\ \lambda_+\big|_{k=\frac{J}{4}} + \lambda_-\big|_{k=\frac{J}{4}} = 0, & \text{for } \delta_0 \leq \delta_{c3}, \\ \lambda_+\big|_{k=\frac{J}{2}} + \lambda_-\big|_{k=\frac{J}{4}} = 0, & \text{for } \delta_0 \leq \delta_{c4}, \end{cases} \tag{35}$$

where

$$\delta_{c1} = -\frac{1}{36\gamma^2}\left(4\gamma(1-6\gamma) + \xi(\gamma) + \frac{\gamma^2\left(12\gamma(12\gamma+5)+1\right)}{\xi(\gamma)}\right),$$

$$\delta_{c2} = \frac{-3 + 36\gamma^2 + 2\gamma + \sqrt{4\gamma\left(3\gamma\left(4\gamma(27\gamma+35)+65\right)+37\right)+9}}{16\gamma(3\gamma+1)},$$



(A) $\gamma = 0.5, \alpha = 1.$ (B) $\gamma = 0.5, \alpha = 1.$

(C) $\gamma = 0.5, \alpha = 1.$ (D) $\gamma = 0.5, \alpha = 1.$

FIGURE 10. Spectrum of the iteration operator of algorithm (1) using a *cell* block-Jacobi smoother for a varying stabilization parameter δ_0 of the SIPG method and reaction scaling $\gamma \geq \gamma_c$.

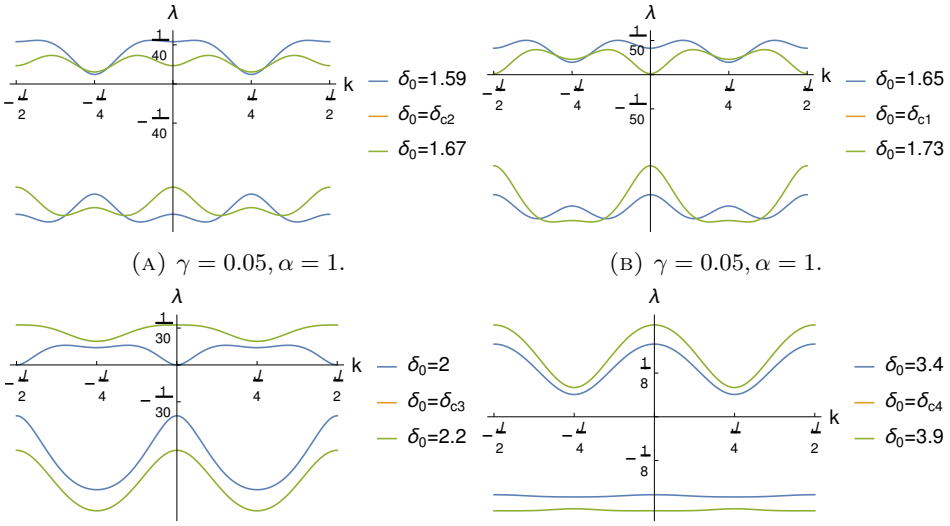
$$\delta_{c3} = 2\gamma + 2,$$
$$\delta_{c4} = 3\left(6\gamma^2 + 4\gamma + 1\right).$$

with $\xi(\gamma) = \gamma\sqrt[3]{3\sqrt{3\left(12\gamma\left(27\gamma\left(8\gamma\left(\gamma\left(6\gamma\left(33\gamma+46\right)+155\right)+44\right)+51\right)+89\right)+25\right)} - 2\left(3\gamma+1\right)\left(12\gamma\left(57\gamma+20\right)+13\right)}$.

We observe that at $\gamma = \gamma_c = 0.16607\ldots$ we have $\delta_{c1}(\gamma) = \delta_{c2}(\gamma)$. For $\gamma \leq \gamma_c$, we have $\delta_{c2} \leq \delta_{c1} \leq \delta_{c3} \leq \delta_{c4}$, which means that the distribution of critical values of δ_0 changes and we have to perform again the same equioscillation analysis as we did previously.

Figures 11a, 11b, 11c and 11d show the spectrum of the iteration operator of algorithm (1) for $\gamma = \frac{1}{20}$. In order to center the spectrum we see that the following equations need to hold:

$$\begin{cases} \lambda_+\big|_{k=\frac{J}{2}} + \lambda_-\big|_{k=\frac{J}{2}} = 0, & \text{for } \delta_0 \leq \delta_{c2} \text{ or } \delta_0 \geq \delta_{c4}, \\ \lambda_+\big|_{k=\frac{J}{2}} + \lambda_-\big|_{k=\frac{J}{4}} = 0, & \text{for } \delta_0 \leq \delta_{c1}, \\ \lambda_+\big|_{k=\frac{J}{4}} + \lambda_-\big|_{k=\frac{J}{4}} = 0, & \text{for } \delta_0 \leq \delta_{c3}, \\ \lambda_+\big|_{k=\frac{J}{2}} + \lambda_-\big|_{k=\frac{J}{4}} = 0, & \text{for } \delta_0 \leq \delta_{c4}. \end{cases} \tag{36}$$



(A) $\gamma = 0.05, \alpha = 1.$ (B) $\gamma = 0.05, \alpha = 1.$

(C) $\gamma = 0.05, \alpha = 1.$ (D) $\gamma = 0.05, \alpha = 1.$

FIGURE 11. Spectrum of the iteration operator of algorithm (1) using a *cell* block-Jacobi smoother for a varying stabilization parameter δ_0 of the SIPG method and reaction scaling $\gamma \leq \gamma_c$.

Following equations (35) and (36), the optimal relaxation parameter is

(37)

$$\alpha_{\text{opt}} = \begin{cases} \dfrac{2(2\delta_0\gamma+1)(6\delta_0\gamma+1)(3(2\delta_0-1)\gamma+1)}{3\gamma\left(24\delta_0\left(2\delta_0^2-1\right)\gamma^2+2\left(18\delta_0^2+\delta_0-6\right)\gamma+9\delta_0-1\right)+2}, \begin{cases} \text{for } \gamma \geq \gamma_c, 1 \leq \delta_0 \leq \delta_{c1}, \\ \text{or } \gamma \geq \gamma_c, \delta_0 \geq \delta_{c4}, \\ \text{or } \gamma \leq \gamma_c, 1 \leq \delta_0 \leq \delta_{c2}, \\ \text{or } \gamma \leq \gamma_c, \delta_0 \geq \delta_{c4}, \end{cases} \\ \dfrac{(2\delta_0\gamma+1)(6\delta_0\gamma+1)}{\gamma(6(4\delta_0-1)\gamma+5\delta_0+6)+1}, \text{ for } \gamma \geq \gamma_c, \delta_{c1} \leq \delta_0 \leq \delta_{c2}. \\ \dfrac{(3\gamma+1)(2\delta_0\gamma+1)(6\delta_0\gamma+1)(3(2\delta_0-1)\gamma+1)}{3\gamma(18\delta_0(8(\delta_0-1)\delta_0+1)\gamma^3+6(4\delta_0(2\delta_0(\delta_0+1)-3)+1)\gamma^2+(\delta_0(31\delta_0-6)-8)\gamma+6\delta_0-2)+1}, \\ \text{for } \gamma \leq \gamma_c, \delta_{c2} \leq \delta_0 \leq \delta_{c1}, \\ \dfrac{2(3\gamma+1)(2\delta_0\gamma+1)(6\delta_0\gamma+1)}{(3(\delta_0+1)\gamma+2)(12(2\delta_0-1)\gamma^2+8\delta_0\gamma+1)}, \begin{cases} \text{for } \gamma \geq \gamma_c, \delta_{c2} \leq \delta_0 \leq \delta_{c3}, \\ \text{or } \gamma \leq \gamma_c, \delta_{c1} \leq \delta_0 \leq \delta_{c3}, \end{cases} \\ \dfrac{2(3\gamma+1)(2\delta_0\gamma+1)(6\delta_0\gamma+1)}{\gamma(36\delta_0(2\delta_0+1)\gamma^2+6(\delta_0(4\delta_0+9)+4)\gamma+13\delta_0+15)+2}, \begin{cases} \text{for } \gamma \geq \gamma_c, \delta_{c3} \leq \delta_0 \leq \delta_{c4}, \\ \text{or } \gamma \leq \gamma_c, \delta_{c3} \leq \delta_0 \leq \delta_{c4}. \end{cases} \end{cases}.$$

Figure 12 shows the behavior of α_{opt} and the corresponding convergence factor of the two-level method as a function of δ_0 for several values of the reaction scaling $\gamma = \frac{\varepsilon}{h^2}$. From the left plot in Figure 12, we see that it would be quite difficult to guess a good choice of the relaxation parameter α without analysis. From the right plot in Figure 12, we see that the *cell* block-Jacobi two level method is also convergent for all values of the penalization parameter $\delta_0 > 1$ and reaction scaling γ when using the optimized relaxation parameter α_{opt}, and it has much better convergence properties for moderate sizes of the penalization parameter δ_0 around 2 than the *point* block-Jacobi two-level method from Figure 9. However convergence is worse

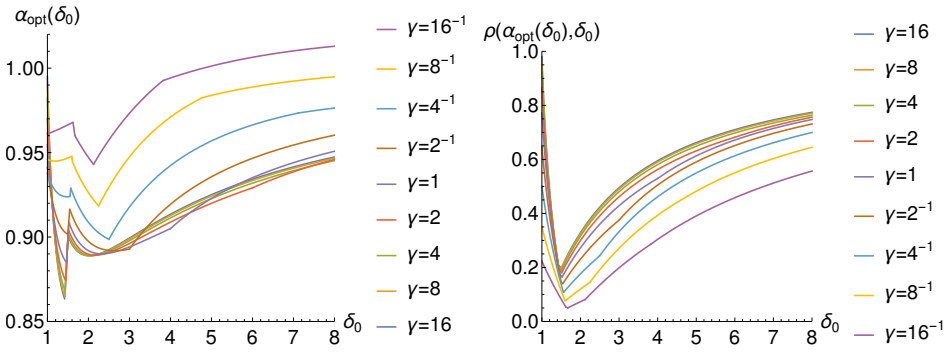


FIGURE 12. Optimized relaxation parameter $\alpha_{\text{opt}}(\delta_0)$ and corresponding convergence factor of Algorithm 1 using a *cell* block-Jacobi smoother as function of the stabilization parameter δ_0 of the SIPG method for different reaction scalings $\gamma = \frac{\varepsilon}{h^2}$.

for larger sizes of the penalization parameter δ_0 than for the *point* block-Jacobi two-level method. We also see from the left plot in Figure 12 that overrelaxation can become necessary when the penalization parameter δ_0 becomes large, especially when γ is small.

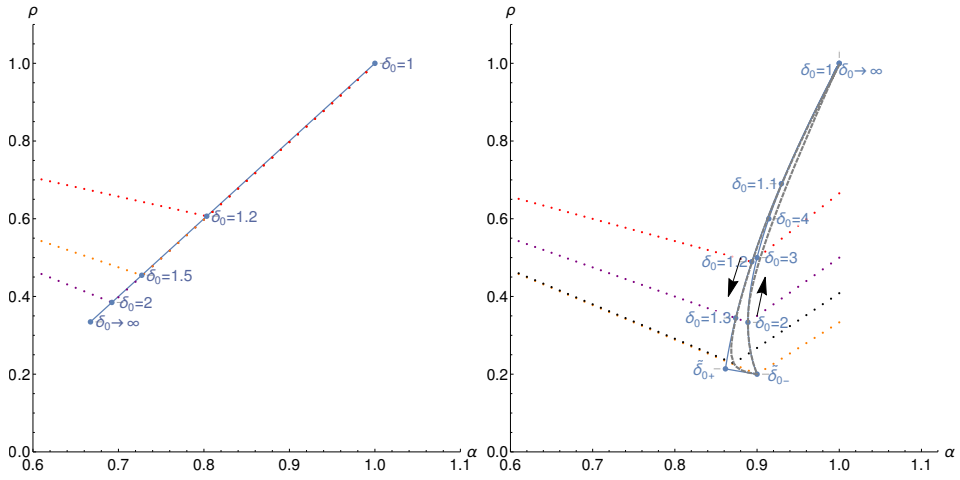
As in the case of Laplace's equation, we see that we obtain the best performance for δ_0 around $\frac{3}{2}$, shown in Figure 12 as the minimum of the curves on the right, and this depends only little on the reaction scaling γ. This shows that also in the reaction-diffusion case, choosing the penalization parameter in SIPG wisely can make the associated iterative solver much faster than just choosing it large enough, even with optimized relaxation parameter α!

6. Numerical experiments

We now show by numerical experiments that the expressions we obtained, though quite lengthy in the reaction-diffusion case, are indeed very good approximations of the optimal relaxation parameters, as a function of the penalization parameter δ_0 and in the reaction case the reaction scaling $\gamma = \frac{\varepsilon}{h^2}$. To do so, we assemble the system matrix on a uniform 64-element mesh, with Dirichlet boundary conditions, and compute numerically the spectral radii of the two-level operators using the QR method, as implemented in LAPACK 3.6.0, accessed with Python 3.5.2.

6.1. *Point* block-Jacobi smoother for the Poisson equation.

The dotted lines in Figure 13a are numerically computed spectral radii ρ vs. relaxation parameter α for $\delta_0 = 1.2$ (red), for $\delta_0 = 1.5$ (orange) and for $\delta_0 = 2$ (purple) for the two-level method with the *point* block-Jacobi smoother. We see that they all attain a minimum value giving fastest convergence, which coincides with the theoretical prediction of Theorem 5.1 marked with blue dots and a label indicating the value of δ_0 used. We also added a theoretical blue dot for $\delta_0 = 1$ (top right) and $\delta_0 \to \infty$ (bottom left), and the entire theoretically predicted parametric line $\rho(\alpha_{\text{opt}}(\delta_0), \delta_0)$, also in blue with $\alpha_{\text{opt}}(\delta_0)$ from Theorem 5.1. We see that our theoretical result based on the typical LFA assumption of periodic boundary conditions predicts the performance with Dirichlet boundary conditions very well. One might be tempted



(A) Numerically computed spectral radius using a *point* block-Jacobi smoother to solve the Poisson equation. Red points: $\delta_0 = 1.2$, orange points: $\delta_0 = 1.5$, purple points: $\delta_0 = 2$. Blue points and blue line: predicted theoretically optimized spectral radius $\rho(\alpha_{\text{opt}})$.

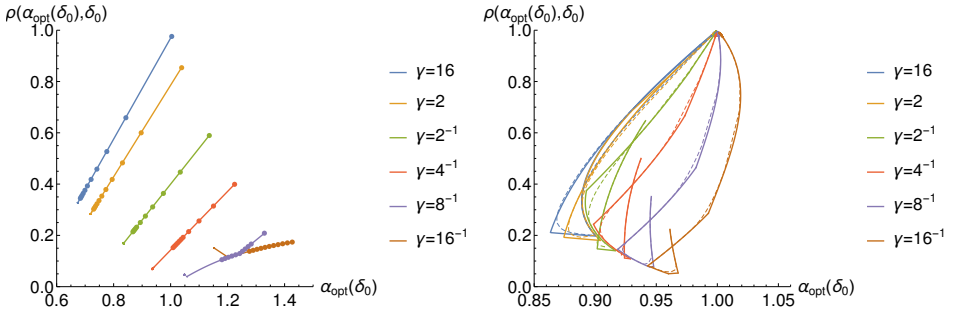
(B) Numerically computed spectral radius using a *cell* block-Jacobi smoother to solve the Poisson equation. Red points: $\delta_0 = 1.2$, orange points: $\delta_0 = \widetilde{\delta}_{0-}$, purple points: $\delta_0 = 2$, black points: $\delta_0 = \widetilde{\delta}_{0+}$. Dashed blue: entire curve of numerically computed optimized spectral radii. Solid blue: predicted theoretically optimized spectral radii $\rho(\alpha_{\text{opt}})$.

FIGURE 13

to use large values of δ_0 in order to have as small a spectral radius as possible, but for large δ_0, the coarse problem is more difficult to solve because the δ_0 is doubled as we showed in §4.3 and the condition number of the unpreconditioned coarse operator grows. It would be interesting to investigate if the capacity of this smoother to deal with large values of δ_0 can be used to our advantage in a multigrid setting.

6.2. ***Cell* block-Jacobi smoother for the Poisson equation.** The dotted lines in Figure 13b are numerically computed spectral radii ρ vs. relaxation parameter α for $\delta_0 = 1.2$ (red), $\delta_0 = \widetilde{\delta}_{0+} \approx 1.41964$ (black), $\delta_0 = \widetilde{\delta}_{0-} = 1.5$ (orange) and $\delta_0 = 2$ (purple) for the two level method with the *cell* block-Jacobi smoother. Like for the *point* block-Jacobi smoother they all attain a minimum value which gives fastest convergence. With blue dots, we mark the theoretical predictions of Theorem 5.2, also for a few more values of $\delta_0 \in \{1, 1.1, 1.3, 4, \infty\}$. In contrast to the *point* block-Jacobi smoother case, the two values $\delta_0 = 1$ and $\delta_0 = \infty$ lead to the same point on the curve at the top right, which shows that this method also deteriorates when δ_0 becomes large. We also plot the entire theoretically predicted parametric line $\rho(\alpha_{\text{opt}}(\delta_0), \delta_0)$ in solid blue with $\alpha_{\text{opt}}(\delta_0)$ from Theorem 5.2 and the corresponding numerically determined one in dashed blue [6]. This shows that the theoretical prediction is very accurate, except for values around $\delta_0 \approx \widetilde{\delta}_{0+}$ where there is a

[6]We did not plot this dashed line for the *point* block-Jacobi smoother case in Figure 13a, since it would not have been visible under the predicted line.



(A) Measured optimal spectral radius using a *point* block-Jacobi smoother to solve a reaction-diffusion equation (points) overlayed on theoretically predicted optimal values (solid line).

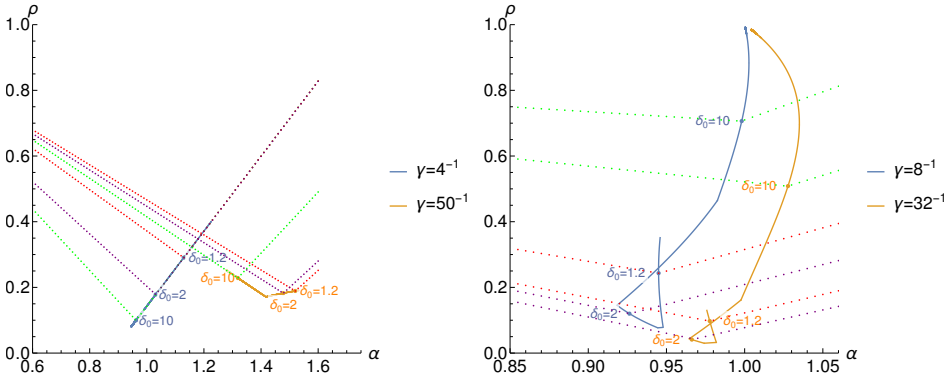
(B) Measured optimal spectral radius using a *cell* block-Jacobi smoother to solve a reaction-diffusion equation (dashed line) overlayed on theoretically predicted optimal values (solid line).

FIGURE 14

small difference. We checked that this is due to the Dirichlet boundary conditions, by performing numerical experiments using periodic boundary conditions which made the results match the predicted line. We also observed that the dashed line approaches the predicted line when decreasing the mesh size. Therefore, even though Theorem 5.2 was obtained with the typical LFA assumption of periodic boundary conditions, the predictions are again very good also for the Dirichlet case. Note that in contrast to the *point* block-Jacobi case, where best performance is achieved for large δ_0, for *cell* block-Jacobi the best performance is achieved for $\delta_0 = \widetilde{\delta_0}_-$, and convergence is almost twice as fast as for *point* block-Jacobi with a similar value for δ_0. Clearly, also in practice, the DG penalization parameter influences very much the performance of the two-level solver, even when using the best possible relaxation parameter.

6.3. ***Point* block-Jacobi smoother for the reaction-diffusion equation.** Results for the solution of a reaction-diffusion equation using a two-level method with the *point* block-Jacobi smoother are shown in Figure 14a.

Theoretically predicted parametric curves are shown for $\delta_0 \in [1, \infty)$, while numerically computed values are shown as points for $\delta_0 \in [1, 50]$. The top right end of the curves corresponds to $\delta_0 = 1$, while the bottom left end corresponds to $\delta_0 \to \infty$. In blue, we can see the measured ρ_{opt}, α_{opt} as dots plotted on top of the predicted parametric curve of the same color, for $\gamma = 16$. As expected, we see that a large value of γ almost reproduces the predicted curve that we observed for the Poisson equation (c.f. Figure 13a). As we modify γ and make it smaller (in orange, green, red, violet and brown, for $\gamma = 2, 2^{-1}, 4^{-1}, 8^{-1}, 16^{-1}$ respectively), the parametric curve moves towards the bottom right of the figure, while keeping its shape until $\gamma \approx 7^{-1}$ where it features a point with discontinuous derivative. Keeping in mind that the rightmost end of each curve corresponds to $\delta_0 = 1$ and the leftmost end corresponds to $\delta_0 \to \infty$, we observe that for any finite value of γ the method is robust for any value of δ_0, i.e. the convergence factor remains bounded away from 1. Large values of γ require underrelaxation, and small values overrelaxation, and



(A) Measured spectral radius using a *point* block-Jacobi smoother to solve a reaction-diffusion equation (points) overlayed on theoretically predicted optimal values (solid line). Note that the minimum spectral radius always falls on the theoretically predicted line.

(B) Measured spectral radius using a *cell* block-Jacobi smoother to solve a reaction-diffusion equation (dashed line) overlayed on theoretically predicted optimal values (solid line). Note that the minimum spectral radius always falls on the theoretically predicted line.

FIGURE 15

in between there are γ values that require both overrelaxation for small δ_0 and underrelaxation for large δ_0 to be optimal. When γ is very small, the regime becomes insensitive to the values of δ_0, which is expected since all the terms in the bilinear form that describe derivatives are negligible in comparison to the reaction term and even at very large values of δ_0, the *point* block-Jacobi smoother can neutralize the operator's dependency on δ_0; see also the bottom curve in Figure 9 on the right.

6.4. ***Cell* block-Jacobi smoother for the reaction-diffusion equation.** Results for the solution of a reaction-diffusion equation using a two-level method with the *cell* block-Jacobi smoother are shown in Figure 14b. Theoretically predicted parametric curves are shown for $\delta_0 \in [1, \infty)$, while numerically computed values are shown as dashed lines for $\delta_0 \in [1, 50]$. All the curves end at $\rho_{\text{opt}} = 1$, $\alpha_{\text{opt}} = 1$, while they begin at smaller values of ρ_{opt} for smaller values of γ. Once again in blue, we show the measured ρ_{opt}, α_{opt} with a dashed line, and the predicted value as a solid line, for $\gamma = 16$. Such a large value of γ is almost equivalent to the Poisson equation and the shapes of the curves of Figure 13b are reproduced. When we set γ to smaller values (in orange, green, red, violet and brown, for $\gamma = 2, 2^{-1}, 4^{-1}, 8^{-1}, 16^{-1}$ respectively), we see that convergence rapidly improves for values of δ_0 that are order one, including $\delta_0 = 1$, represented as the beginning of the curve that moves down and to the right of the figure. For moderate values of δ_0, very small values of γ will even result in an exact solver with the smoother alone. Convergence however still deteriorates as $\delta_0 \to \infty$, since, unlike the point block-Jacobi smoother, the cell block-Jacobi smoother cannot neutralize the operator's dependency on δ_0 for δ_0 large. The measured results (dashed) and theoretically predicted ones (solid) show very good agreement. Also, we see that small values of γ can require overrelaxation when δ_0 becomes large.

Figure 15b shows experiments for a range of relaxation parameters, in order to illustrate that when using the optimal relaxation, the spectral radius falls on the line of predicted values. Each dot on the v-shaped dotted line is an experiment performed for a different α. The predicted optimal point on the solid line is indicated with a label.

6.5. **Higher dimensions, different geometries and further research.** We now test our closed form optimized relaxation parameters from the 1D analysis in higher dimensions and on geometries and meshes that go far beyond a simple tensor product generalization. To that end, we use the `deal.II` finite element library [1]. We show in Figure 16 a set of comparisons of the optimality of our closed form optimized relaxation parameters for the Poisson problem, using *cell* block-Jacobi smoothers. In each case, we show the mesh used and a comparison between the unrelaxed method, the relaxation of $2/3$ coming from the smoothing analysis alone, the one predicted by Theorem 5.2, and the numerically best performing one, which we obtained by running the code for many parameters and then taking the best performing one. The closeness between our closed form optimized parameters from the 1D analysis and the numerically best working one in higher dimensions is clear evidence that the seminal quote from P. W. Hemker in footnote 4 is more than justified.

In principle, it would be possible to extend our analysis to the case of tensor product meshes in 2D (and 3D), but this would pose important technical difficulties: in Section 4, we have seen that considering the complete 2-level error operator necessitates analyzing a 4×4 matrix instead of a 2×2 matrix needed for the smoothing analysis alone. For a tensor product grid in 2D, the error operator of the complete 2-level analysis would be 16×16, and a direct analysis like the one we performed in 1D would require finding exact expressions, depending on the coefficients, of polynomials of degree 16. Such difficulties have been faced by D. Le Roux et al., for specific wave propagation applications [24], and they require, when possible at all, a very careful algebraic analysis and general understanding of the tensor interactions. To the best of our knowledge, for higher dimensions, the community has turned to the numerical study of the resulting matrices, see e.g. [6, 9, 16, 17, 20] and references therein, which can not give the same depth understanding as an analytical study. Some generalizations that tackle higher dimensions and different boundary conditions can be found in [29].

The advantage of our approach is that we can see the interactions between different components in a very clear way in 1D, and thus achieve deeper insight into the functioning of the numerical method in 1D. Furthermore, our numerical experiments in higher dimensions show that the 1D results are still giving close to optimal relaxation parameters, even on non-tensor and irregular meshes, which indicates that our 1D analysis captures fundamental diffusion and singularly perturbed reaction diffusion behavior of the underlying operator, not just in 1D and for tensor product meshes. A further illustration of the interest of our detailed 1D analysis is our publication [14] showing that the optimization can be carried as far as to obtain an exact solver from an iterative one, with exact analytical expressions for the relaxation parameters involved.

The complexity of the analytical expressions found in our 2-level analysis not withstanding, we managed, based on the results in the present manuscript, to obtain analytical expressions for finite difference stencils in 2D and 3D by using

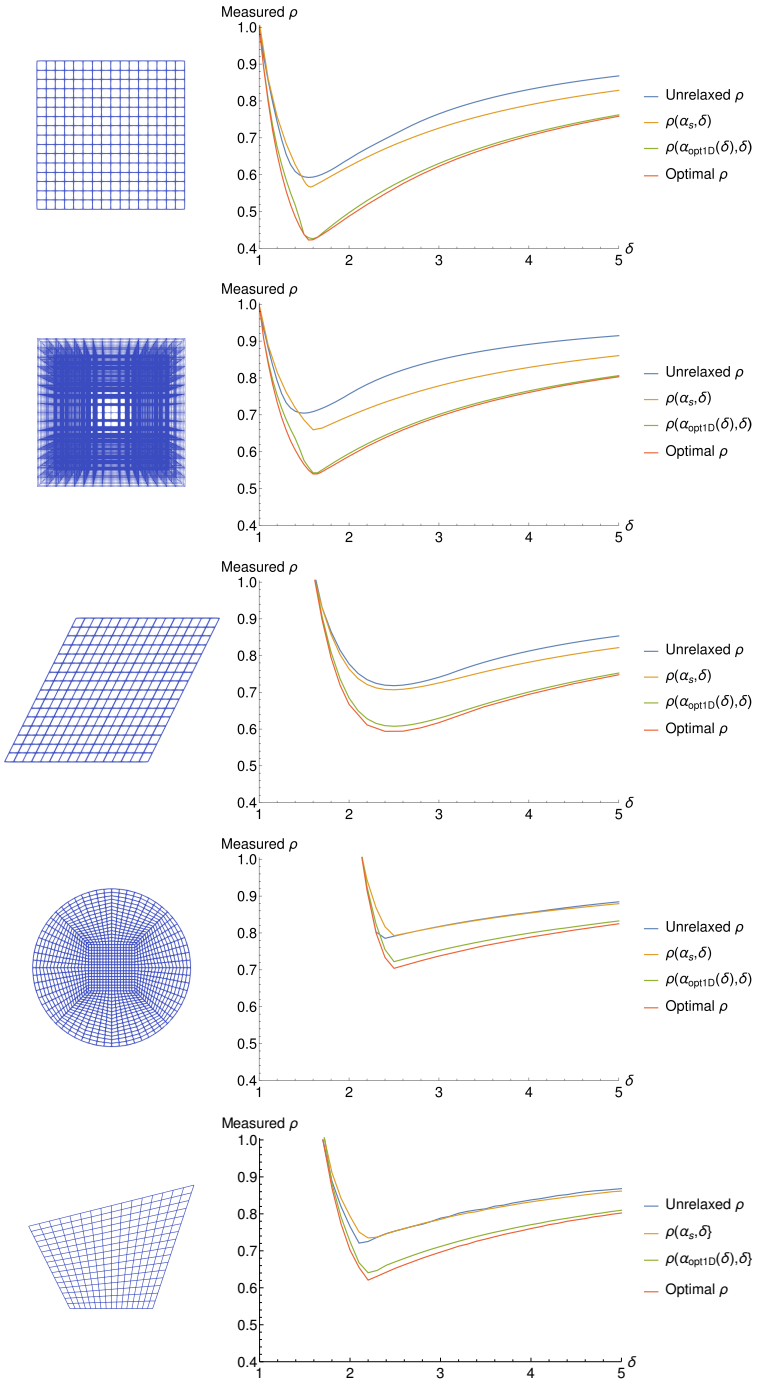


FIGURE 16. Comparison of the spectral radius of the two level operator for the Poisson problem on different geometries and meshes in higher dimensions. We compare the unrelaxed method, the relaxation $\alpha_s = 2/3$ coming from the smoothing analysis alone, the optimized α_{opt1D} from Theorem 5.2, and the numerically best working one.

different, red/black decompositions, establishing a link with cyclic reduction. The work is however extensive and will appear elsewhere.

7. Conclusion

We optimized the relaxation parameter in two-level iterative methods for solving symmetric interior penalty discontinuous Galerkin discretized Poisson and reaction-diffusion equations using a *cell* block-Jacobi and a *point* block-Jacobi smoother. Our optimization for the complete two-level process shows that the *cell* block-Jacobi smoother leads to a more effective two-level method for moderate sizes of the penalization parameter, while the *point* block-Jacobi smoother is superior for large penalization parameters. Our analysis also reveals that the penalization parameter in SIPG should not only be chosen large enough such that the DG method converges, but it can be chosen to optimize the performance of the associated iterative two-level solver. A good choice can lead to an iterative solver that converges an order of magnitude faster than other choices, and this even using the best possible relaxation parameter in the smoother. While we performed our analysis in 1D, our numerical experiments in higher dimensions on irregular domains with irregular meshes clearly show that our closed form optimized relaxation parameters work very well also in these situations, with very close to best possible performance of the SIPG two level method.

References

[1] Arndt, D., Bangerth, W., Blais, B., Fehling, M., Gassmöller, R., Heister, T., Heltai, L., Köcher, U., Kronbichler, M., Maier, M., Munch, P., Pelteret, J.P., Proell, S., Simon, K., Turcksin, B., Wells, D., Zhang, J.: The `deal.II` library, version 9.3. Journal of Numerical Mathematics (2021, accepted for publication). URL `https://dealii.org/deal93-preprint.pdf`

[2] Arnold, D.N.: An interior penalty finite element method with discontinuous elements. SIAM J. Numer. Anal. **19**(4), 742–760 (1982). DOI 10.1137/0719052

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

[4] Baker, G.A.: Finite element methods for elliptic equations using nonconforming elements. Math. Comp. **137**(31), 45–59 (1977)

[5] Becherer, D., Schweizer, M.: Classical solutions to reaction-diffusion systems for hedging problems with interacting itô and point processes. The Annals of Applied Probability **2**(15), 1111–1144 (2005)

[6] Bolten, M., Rittich, H.: Fourier analysis of periodic stencils in multigrid methods. SIAM Journal on Scientific Computing **40**(3), A1642–A1668 (2018). DOI 10.1137/16M1073959. URL `https://doi.org/10.1137/16M1073959`

[7] Brandt, A.: Multi-level adaptive solutions to boundary-value problems. Mathematics of Computation **31**(138), 333–390 (1977). URL `http://www.jstor.org/stable/2006422`

[8] Brandt, A.: Rigorous quantitative analysis of multigrid. i. constant coefficients two-level cycle with L^2-norm. SIAM J. Numer. Anal. **6**(31), 1695–1730 (1994). DOI 10.1137/0731087

[9] Brown, J., He, Y., MacLachlan, S.: Local fourier analysis of balancing domain decomposition by constraints algorithms. SIAM Journal on Scientific Computing **41**(5), S346–S369 (2019). DOI 10.1137/18M1191373. URL `https://doi.org/10.1137/18M1191373`

[10] Dautray, R., Lions, J.L.: Mathematical analysis and numerical methods for science and technology. Volume 2. , Functional and Variational Methods. Springer-Verlag, Berlin Heidelberg New York London Paris Tokyo (1985)

[11] Dryja, M., Krzyżanowski, P.: A massively parallel nonoverlapping additive Schwarz method for discontinuous Galerkin discretization of elliptic problems. Numerische Mathematik **132**(2), 347–367 (2016). DOI 10.1007/s00211-015-0718-5

[12] Feng, X., Karakashian, O.: Two-level non-overlapping Schwarz methods for a discontinuous Galerkin method. SIAM J. Numer. Anal. **39**(4), 1343–1365 (2001)
[13] Fife, P.C.: Mathematical Aspects of Reacting and Diffusing Systems. Springer Verlag Berlin Heidelberg New York (1979). DOI 10.1007/978-3-642.93111-6
[14] Gander, M.J., Lucero Lorca, J.P.: Should multilevel methods for discontinuous Galerkin discretizations use discontinuous interpolation operators? In: S.C. Brenner, E. Chung, A. Klawonn, F. Kwok, J. Xu, J. Zou (eds.) Domain Decomposition Methods in Science and Engineering XXVI, pp. 273–280. Springer International Publishing, Cham (2022)
[15] Gie, G., Hamouda, M., Jung, C., Temam, R.: Singular Perturbations and Boundary Layers. Applied Mathematical Sciences. Springer International Publishing (2018). URL `https://books.google.ch/books?id=d2V7DwAAQBAJ`
[16] He, Y., MacLachlan, S.: Two-level fourier analysis of multigrid for higher-order finite-element discretizations of the laplacian. Numerical Linear Algebra with Applications **27**(3), e2285 (2020). DOI https://doi.org/10.1002/nla.2285. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nla.2285`
[17] He, Y., Rhebergen, S., Sterck, H.D.: Local fourier analysis of multigrid for hybridized and embedded discontinuous Galerkin methods. SIAM Journal on Scientific Computing **43**(5), S612–S636 (2021). DOI 10.1137/20M1346985. URL `https://doi.org/10.1137/20M1346985`
[18] Hemker, P., Hoffmann, W., van Raalte, M.: Two-level fourier analysis of a multigrid approach for discontinuous Galerkin discretization. SIAM Journal on Scientific Computing **3**(25), 1018–1041 (2003)
[19] Hemker, P.W., Hoffmann, W., van Raalte, M.H.: Fourier two-level analysis for discontinuous Galerkin discretization with linear elements. Numerical Linear Algebra with Applications **5** – **6**(11), 473–491 (2004)
[20] Kahl, K., Kintscher, N.: Automated local Fourier analysis (aLFA). BIT Numerical Mathematics **60**, 1572–9125 (2020). DOI 10.1007/s10543-019-00797-w
[21] Kanschat, G., Lucero Lorca, J.P.: A weakly penalized discontinuous Galerkin method for radiation in dense, scattering media. CMAM **16**(4), 563–577 (2016). DOI 10.1515/cmam-2016-0023
[22] Karakashian, O., Collins, C.: Two-level additive Schwarz methods for discontinuous Galerkin approximations of second-order elliptic problems. IMA Journal of Numerical Analysis **37**, 1800–1830 (2017). DOI 10.1093/imanum/drw061
[23] Kopteva, N., O'Riordan, E.: Shishkin meshes in the numerical solution of singularly perturbed differential equations. International Journal of Numerical Analysis and Modeling **7** (2010)
[24] Le Roux, D.Y., Eldred, C., Taylor, M.A.: Fourier analyses of high-order continuous and discontinuous Galerkin methods. SIAM Journal on Numerical Analysis **58**(3), 1845–1866 (2020). DOI 10.1137/19M1289595. URL `https://doi.org/10.1137/19M1289595`
[25] Lew, A.J., Buscaglia, G.C.: A discontinuous-Galerkin-based immersed boundary method. International Journal for Numerical Methods in Engineering **76**(4), 427–454 (2008). DOI 10.1002/nme.2312. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2312`
[26] Lucero Lorca, J.P., Kanschat, G.: Multilevel Schwarz preconditioners for singularly perturbed symmetric reaction-diffusion systems. Electron. Trans. Numer. Anal. **54**, 89–107 (2021). DOI 10.1553/etna_vol54s89
[27] Manteuffel, T.A., Ressel, K.J.: Least-squares finite-element solution of the neutron transport equation in diffusive regimes. SIAM J. Numer. Anal. **35**(2), 806–835 (1998)
[28] Nitsche, J.: Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei der Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. Abh. Math. Sem. Univ. Hamburg **36**, 9–15 (1971)
[29] Rodrigo CarmenGaspar, F.J., Zikatanov, L.T.: On the validity of the local fourier analysis. Journal of Computational Mathematics **37**(3), 340–348 (2018). DOI https://doi.org/10.4208/jcm.1803-m2017-0294. URL `http://global-sci.org/intro/article_detail/jcm/12725.html`
[30] Smoller, J.: Shock Waves and Reaction–Diffusion Equations. No. 258 in XXI, 581 S., 162 Abb., DM 128. Berlin, Heidelberg, New York, Springer – Verlag (1983)

[31] van der Vegt, J., Rhebergen, S.: hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows: Part i. multilevel analysis. Journal of Computational Physics **231**(22), 7537–7563 (2012). DOI https://doi.org/10.1016/j.jcp.2012.05.038. URL https://www.sciencedirect.com/science/article/pii/S0021999112003129

[32] van der Vegt, J., Rhebergen, S.: Hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. part ii: Optimization of the runge–kutta smoother. Journal of Computational Physics **231**(22), 7564–7583 (2012). DOI https://doi.org/10.1016/j.jcp.2012.05.037. URL https://www.sciencedirect.com/science/article/pii/S0021999112003117

[33] Wheeler, M.F.: An elliptic collocation finite element method with interior penalties. SIAM J. Numer. Anal. **39**(15(1)), 152–161 (1978)

[34] Zhou, Y.: Fourier Analysis and Local Fourier Analysis for Multigrid Methods. Master's thesis, Johannes Kepler Universität Linz (2009)

Appendix A. Reaction-diffusion iteration operator eigenvalue coefficients using a *point* block-Jacobi smoother

$$
\begin{aligned}
c_1 =& -8640\alpha\delta_0^2\gamma^4 - 14400\alpha\delta_0^2\gamma^3 - 2544\alpha\delta_0^2\gamma^2 + 6912\alpha\delta_0\gamma^4 \\
&+ 7776\alpha\delta_0\gamma^3 - 3744\alpha\delta_0\gamma^2 - 992\alpha\delta_0\gamma - 864\alpha\gamma^4 + 288\alpha\gamma^3 + 2208\alpha\gamma^2 \\
&- 80\alpha + 6912\delta_0^2\gamma^4 + 11520\delta_0^2\gamma^3 + 3072\delta_0^2\gamma^2 - 5184\delta_0\gamma^4 - 5184\delta_0\gamma^3 \\
&+ 2688\delta_0\gamma^2 + 1280\delta_0\gamma + 864\gamma^4 - 1248\gamma^2 + 128 \\
c_2 =& -384\gamma + 240\alpha\gamma + 256\delta_0\gamma - 160\alpha\delta_0\gamma + 1392\alpha\gamma^2 - 2688\delta_0\gamma^2 \\
&+ 480\alpha\delta_0\gamma^2 + 1536\delta_0^2\gamma^2 - 960\alpha\delta_0^2\gamma^2 + 3456\gamma^3 - 3168\alpha\gamma^3 - 9216\delta_0\gamma^3 \\
&+ 12096\alpha\delta_0\gamma^3 + 2304\delta_0^2\gamma^3 - 5760\alpha\delta_0^2\gamma^3 + 3456\delta_0\gamma^4 - 3456\alpha\delta_0\gamma^4 \\
&- 6912\delta_0^2\gamma^4 + 6912\alpha\delta_0^2\gamma^4 \\
c_3 =& 96\gamma^2 + 144\alpha\gamma^2 - 192\alpha\delta_0\gamma^2 + 48\alpha\delta_0^2\gamma^2 - 576\alpha\gamma^3 + 576\delta_0\gamma^3 \\
&+ 864\alpha\delta_0\gamma^3 - 576\alpha\delta_0^2\gamma^3 - 864\gamma^4 + 864\alpha\gamma^4 + 1728\delta_0\gamma^4 - 3456\alpha\delta_0\gamma^4 \\
&+ 1728\alpha\delta_0^2\gamma^4 \\
c_4 =& 2985984\alpha^2\delta_0^4\gamma^8 + 9953280\alpha^2\delta_0^4\gamma^7 + 6469632\alpha^2\delta_0^4\gamma^6 - 3041280\alpha^2\delta_0^4\gamma^5 \\
&+ 278784\alpha^2\delta_0^4\gamma^4 - 5971968\alpha^2\delta_0^3\gamma^8 - 18911232\alpha^2\delta_0^3\gamma^7 - 9123840\alpha^2\delta_0^3\gamma^6 \\
&+ 8487936\alpha^2\delta_0^3\gamma^5 - 2442240\alpha^2\delta_0^3\gamma^4 + 353280\alpha^2\delta_0^3\gamma^3 + 5971968\alpha^2\delta_0^2\gamma^8 \\
&+ 18911232\alpha^2\delta_0^2\gamma^7 + 8957952\alpha^2\delta_0^2\gamma^6 - 8543232\alpha^2\delta_0^2\gamma^5 + 1833984\alpha^2\delta_0^2\gamma^4 \\
&- 1373184\alpha^2\delta_0^2\gamma^3 + 100864\alpha^2\delta_0^2\gamma^2 - 746496\alpha^2\delta_0\gamma^8 + 248832\alpha^2\delta_0\gamma^7 \\
&+ 10368000\alpha^2\delta_0\gamma^6 + 13906944\alpha^2\delta_0\gamma^5 + 2062080\alpha^2\delta_0\gamma^4 + 856320\alpha^2\delta_0\gamma^3 \\
&- 276480\alpha^2\delta_0\gamma^2 + 8192\alpha^2\delta_0\gamma - 248832\alpha^2\gamma^7 - 829440\alpha^2\gamma^6 + 359424\alpha^2\gamma^5 \\
&+ 2062080\alpha^2\gamma^4 + 734976\alpha^2\gamma^3 + 195072\alpha^2\gamma^2 - 9216\alpha^2\gamma + 256\alpha^2 \\
c_5 =& 11943936\alpha^2\delta_0^4\gamma^7 + 17915904\alpha^2\delta_0^4\gamma^6 - 6967296\alpha^2\delta_0^4\gamma^5 + 608256\alpha^2\delta_0^4\gamma^4 \\
&- 21897216\alpha^2\delta_0^3\gamma^7 - 25214976\alpha^2\delta_0^3\gamma^6 + 25712640\alpha^2\delta_0^3\gamma^5 - 5981184\alpha^2\delta_0^3\gamma^4 \\
&+ 457728\alpha^2\delta_0^3\gamma^3 + 20901888\alpha^2\delta_0^2\gamma^7 + 25049088\alpha^2\delta_0^2\gamma^6 - 20542464\alpha^2\delta_0^2\gamma^5 \\
&+ 10243584\alpha^2\delta_0^2\gamma^4 - 2339328\alpha^2\delta_0^2\gamma^3 + 83968\alpha^2\delta_0^2\gamma^2 - 3732480\alpha^2\delta_0\gamma^8
\end{aligned}
$$

$$
\begin{aligned}
&- 17169408\alpha^2\delta_0\gamma^7 - 12773376\alpha^2\delta_0\gamma^6 + 12690432\alpha^2\delta_0\gamma^5 - 2449152\alpha^2\delta_0\gamma^4 \\
&+ 2492160\alpha^2\delta_0\gamma^3 - 313344\alpha^2\delta_0\gamma^2 + 4096\alpha^2\delta_0\gamma + 746496\alpha^2\gamma^8 \\
&+ 1741824\alpha^2\gamma^7 - 1575936\alpha^2\gamma^6 - 4810752\alpha^2\gamma^5 + 126720\alpha^2\gamma^4 - 292608\alpha^2\gamma^3 \\
&+ 201216\alpha^2\gamma^2 - 10752\alpha^2\gamma \\
c_6 = &- 5971968\alpha^2\delta_0^4\gamma^8 - 7962624\alpha^2\delta_0^4\gamma^7 + 16920576\alpha^2\delta_0^4\gamma^6 - 4866048\alpha^2\delta_0^4\gamma^5 \\
&+ 382464\alpha^2\delta_0^4\gamma^4 + 11943936\alpha^2\delta_0^3\gamma^8 + 11943936\alpha^2\delta_0^3\gamma^7 - 36163584\alpha^2\delta_0^3\gamma^6 \\
&+ 23003136\alpha^2\delta_0^3\gamma^5 - 4174848\alpha^2\delta_0^3\gamma^4 + 89088\alpha^2\delta_0^3\gamma^3 - 11943936\alpha^2\delta_0^2\gamma^8 \\
&- 10948608\alpha^2\delta_0^2\gamma^7 + 35997696\alpha^2\delta_0^2\gamma^6 - 19491840\alpha^2\delta_0^2\gamma^5 + 11828736\alpha^2\delta_0^2\gamma^4 \\
&- 685056\alpha^2\delta_0^2\gamma^3 - 512\alpha^2\delta_0^2\gamma^2 + 4478976\alpha^2\delta_0\gamma^8 - 7464960\alpha^2\delta_0\gamma^7 \\
&- 35168256\alpha^2\delta_0\gamma^6 - 1852416\alpha^2\delta_0\gamma^5 - 8808192\alpha^2\delta_0\gamma^4 + 1552128\alpha^2\delta_0\gamma^3 \\
&+ 4976640\alpha^2\gamma^7 + 8792064\alpha^2\gamma^6 - 663552\alpha^2\gamma^5 + 105984\alpha^2\gamma^4 - 988416\alpha^2\gamma^3 \\
&+ 2304\alpha^2\gamma^2 \\
c_7 = &- 11943936\alpha^2\delta_0^4\gamma^7 + 5971968\alpha^2\delta_0^4\gamma^6 - 995328\alpha^2\delta_0^4\gamma^5 + 55296\alpha^2\delta_0^4\gamma^4 \\
&+ 21897216\alpha^2\delta_0^3\gamma^7 - 22560768\alpha^2\delta_0^3\gamma^6 + 6137856\alpha^2\delta_0^3\gamma^5 - 654336\alpha^2\delta_0^3\gamma^4 \\
&- 15360\alpha^2\delta_0^3\gamma^3 - 20901888\alpha^2\delta_0^2\gamma^7 + 22063104\alpha^2\delta_0^2\gamma^6 - 10202112\alpha^2\delta_0^2\gamma^5 \\
&+ 2585088\alpha^2\delta_0^2\gamma^4 + 84480\alpha^2\delta_0^2\gamma^3 + 4478976\alpha^2\delta_0\gamma^8 + 17418240\alpha^2\delta_0\gamma^7 \\
&- 10450944\alpha^2\delta_0\gamma^6 + 1907712\alpha^2\delta_0\gamma^5 - 4020480\alpha^2\delta_0\gamma^4 - 145152\alpha^2\delta_0\gamma^3 \\
&- 1492992\alpha^2\gamma^8 - 1990656\alpha^2\gamma^7 + 7382016\alpha^2\gamma^6 + 3704832\alpha^2\gamma^5 \\
&+ 2080512\alpha^2\gamma^4 + 76032\alpha^2\gamma^3 \\
c_8 = &2985984\alpha^2\delta_0^4\gamma^8 - 1990656\alpha^2\delta_0^4\gamma^7 + 497664\alpha^2\delta_0^4\gamma^6 - 55296\alpha^2\delta_0^4\gamma^5 \\
&+ 2304\alpha^2\delta_0^4\gamma^4 - 5971968\alpha^2\delta_0^3\gamma^8 + 6967296\alpha^2\delta_0^3\gamma^7 - 2488320\alpha^2\delta_0^3\gamma^6 \\
&+ 359424\alpha^2\delta_0^3\gamma^5 - 18432\alpha^2\delta_0^3\gamma^4 + 5971968\alpha^2\delta_0^2\gamma^8 - 7962624\alpha^2\delta_0^2\gamma^7 \\
&+ 3483648\alpha^2\delta_0^2\gamma^6 - 940032\alpha^2\delta_0^2\gamma^5 + 50688\alpha^2\delta_0^2\gamma^4 - 3732480\alpha^2\delta_0\gamma^8 \\
&+ 7216128\alpha^2\delta_0\gamma^7 + 248832\alpha^2\delta_0\gamma^6 + 1216512\alpha^2\delta_0\gamma^5 - 55296\alpha^2\delta_0\gamma^4 \\
&- 4727808\alpha^2\gamma^7 - 1824768\alpha^2\gamma^6 - 580608\alpha^2\gamma^5 + 20736\alpha^2\gamma^4 \\
c_9 = &- 746496\alpha^2\delta_0\gamma^8 - 248832\alpha^2\delta_0\gamma^7 + 746496\alpha^2\gamma^8 + 248832\alpha^2\gamma^7 \\
c_{10} = &6912\delta_0^2\gamma^4 + 11520\delta_0^2\gamma^3 + 3072\delta_0^2\gamma^2 - 5184\delta_0\gamma^4 - 5184\delta_0\gamma^3 \\
&+ 2688\delta_0\gamma^2 + 1280\delta_0\gamma + 864\gamma^4 - 1248\gamma^2 + 128 \\
c_{11} = &- 6912\delta_0^2\gamma^4 + 2304\delta_0^2\gamma^3 + 1536\delta_0^2\gamma^2 + 3456\delta_0\gamma^4 - 9216\delta_0\gamma^3 - 2688\delta_0\gamma^2 \\
&+ 256\delta_0\gamma + 3456\gamma^3 - 384\gamma \\
c_{12} = &1728\delta_0\gamma^4 + 576\delta_0\gamma^3 - 864\gamma^4 + 96\gamma^2
\end{aligned}
$$

Appendix B. Reaction-diffusion iteration operator eigenvalue coefficients using a *cell* block-Jacobi smoother

$$
\begin{aligned}
c_1 = &16(-144\alpha\delta_0^3\gamma^4 - 192\alpha\delta_0^3\gamma^3 - 36\alpha\delta_0^2\gamma^4 - 216\alpha\delta_0^2\gamma^3
\end{aligned}
$$

$$
\begin{aligned}
&\quad - 170\alpha\delta_0^2\gamma^2 + 72\alpha\delta_0\gamma^4 + 96\alpha\delta_0\gamma^3 - 84\alpha\delta_0\gamma^2 - 50\alpha\delta_0\gamma + 36\alpha\gamma^3 \\
&\quad + 60\alpha\gamma^2 - 4\alpha + 144\delta_0^3\gamma^4 + 192\delta_0^3\gamma^3 - 36\delta_0^2\gamma^4 + 96\delta_0^2\gamma^3 + 176\delta_0^2\gamma^2 - 24\delta_0\gamma^3 \\
&\quad + 12\delta_0\gamma^2 + 48\delta_0\gamma - 3\gamma^2 + 4) \\
c_2 =&16(144\alpha\delta_0^3\gamma^4 - 96\alpha\delta_0^3\gamma^3 + 72\alpha\delta_0^2\gamma^4 + 216\alpha\delta_0^2\gamma^3 - 46\alpha\delta_0^2\gamma^2 \\
&\quad - 72\alpha\delta_0\gamma^4 + 60\alpha\delta_0\gamma^3 + 72\alpha\delta_0\gamma^2 - 4\alpha\delta_0\gamma - 36\alpha\gamma^3 + 12\alpha\gamma^2 \\
&\quad + 6\alpha\gamma - 144\delta_0^3\gamma^4 + 96\delta_0^3\gamma^3 - 240\delta_0^2\gamma^3 + 64\delta_0^2\gamma^2 - 108\delta_0\gamma^2 + 8\delta_0\gamma - 12\gamma) \\
c_3 =&16(-36\alpha\delta_0^2\gamma^4 - 12\alpha\delta_0\gamma^3 + 36\delta_0^2\gamma^4 + 24\delta_0\gamma^3 + 3\gamma^2) \\
c_4 =&1024\alpha^2\gamma^2(5184\delta_0^6\gamma^6 + 13824\delta_0^6\gamma^5 + 9216\delta_0^6\gamma^4 - 18144\delta_0^5\gamma^6 - 43200\delta_0^5\gamma^5 \\
&\quad - 17136\delta_0^5\gamma^4 + 10944\delta_0^5\gamma^3 + 21060\delta_0^4\gamma^6 + 36720\delta_0^4\gamma^5 - 16236\delta_0^4\gamma^4 \\
&\quad - 33624\delta_0^4\gamma^3 + 4665\delta_0^4\gamma^2 - 9072\delta_0^3\gamma^6 - 864\delta_0^3\gamma^5 + 41760\delta_0^3\gamma^4 + 23292\delta_0^3\gamma^3 \\
&\quad - 16140\delta_0^3\gamma^2 + 858\delta_0^3\gamma + 1944\delta_0^2\gamma^6 - 3672\delta_0^2\gamma^5 - 12384\delta_0^2\gamma^4 + 7524\delta_0^2\gamma^3 \\
&\quad + 15018\delta_0^2\gamma^2 - 3096\delta_0^2\gamma + 61\delta_0^2 + 1836\delta_0\gamma^5 + 2592\delta_0\gamma^4 - 2340\delta_0\gamma^3 \\
&\quad - 1116\delta_0\gamma^2 + 2931\delta_0\gamma - 228\delta_0 + 432\gamma^4 + 1080\gamma^3 + 504\gamma^2 - 72\gamma + 219) \\
c_5 =&1024\alpha^2\gamma^2(-10368\delta_0^6\gamma^6 - 6912\delta_0^6\gamma^5 + 9216\delta_0^6\gamma^4 + 33696\delta_0^5\gamma^6 + 5184\delta_0^5\gamma^5 \\
&\quad - 46944\delta_0^5\gamma^4 + 10656\delta_0^5\gamma^3 - 37584\delta_0^4\gamma^6 + 23760\delta_0^4\gamma^5 + 65988\delta_0^4\gamma^4 \\
&\quad - 48960\delta_0^4\gamma^3 + 4518\delta_0^4\gamma^2 + 16848\delta_0^3\gamma^6 - 34776\delta_0^3\gamma^5 - 23616\delta_0^3\gamma^4 + 65916\delta_0^3\gamma^3 \\
&\quad - 19836\delta_0^3\gamma^2 + 834\delta_0^3\gamma - 3888\delta_0^2\gamma^6 + 14040\delta_0^2\gamma^5 - 4752\delta_0^2\gamma^4 - 26532\delta_0^2\gamma^3 \\
&\quad + 24900\delta_0^2\gamma^2 - 3498\delta_0^2\gamma + 56\delta_0^2 - 3672\delta_0\gamma^5 + 1944\delta_0\gamma^4 + 2772\delta_0\gamma^3 \\
&\quad - 8028\delta_0\gamma^2 + 3960\delta_0\gamma - 222\delta_0 - 864\gamma^4 - 432\gamma^3 + 576\gamma^2 - 756\gamma + 216) \\
c_6 =&1024\alpha^2\gamma^2(5184\delta_0^6\gamma^6 - 6912\delta_0^6\gamma^5 + 2304\delta_0^6\gamma^4 - 12960\delta_0^5\gamma^6 + 36288\delta_0^5\gamma^5 \\
&\quad - 18864\delta_0^5\gamma^4 + 2592\delta_0^5\gamma^3 + 12312\delta_0^4\gamma^6 - 54000\delta_0^4\gamma^5 + 52380\delta_0^4\gamma^4 - 15912\delta_0^4\gamma^3 \\
&\quad + 1041\delta_0^4\gamma^2 - 6480\delta_0^3\gamma^6 + 30888\delta_0^3\gamma^5 - 54720\delta_0^3\gamma^4 + 33012\delta_0^3\gamma^3 - 5640\delta_0^3\gamma^2 \\
&\quad + 156\delta_0^3\gamma + 1296\delta_0^2\gamma^6 - 11880\delta_0^2\gamma^5 + 19476\delta_0^2\gamma^4 - 25236\delta_0^2\gamma^3 + 10038\delta_0^2\gamma^2 \\
&\quad - 762\delta_0^2\gamma + 4\delta_0^2 + 1296\delta_0\gamma^5 - 6480\delta_0\gamma^4 + 3636\delta_0\gamma^3 - 6228\delta_0\gamma^2 + 1209\delta_0\gamma \\
&\quad - 12\delta_0 + 324\gamma^4 - 1080\gamma^3 + 36\gamma^2 - 684\gamma + 6) \\
c_7 =&1024\alpha^2\gamma^2(-2592\delta_0^5\gamma^6 + 1728\delta_0^5\gamma^5 + 3888\delta_0^4\gamma^6 - 6480\delta_0^4\gamma^5 + 1548\delta_0^4\gamma^4 \\
&\quad - 1296\delta_0^3\gamma^6 + 4536\delta_0^3\gamma^5 - 4896\delta_0^3\gamma^4 + 468\delta_0^3\gamma^3 + 1296\delta_0^2\gamma^6 + 1512\delta_0^2\gamma^5 \\
&\quad + 2808\delta_0^2\gamma^4 - 1548\delta_0^2\gamma^3 + 48\delta_0^2\gamma^2 + 1080\delta_0\gamma^5 + 1944\delta_0\gamma^4 + 1116\delta_0\gamma^3 \\
&\quad - 180\delta_0\gamma^2 + 216\gamma^4 + 432\gamma^3 + 180\gamma^2) \\
c_8 =&1024\alpha^2\gamma^2(324\delta_0^4\gamma^6 + 216\delta_0^3\gamma^5 - 648\delta_0^2\gamma^6 + 36\delta_0^2\gamma^4 - 540\delta_0\gamma^5 - 108\gamma^4) \\
c_9 =&8(2\delta_0\gamma + 1)(6\delta_0\gamma + 1)(3(8\delta_0 - 1)\gamma^2 + 32\delta_0\gamma - 3\gamma^2 + 8) \\
c_{10} =&-64\gamma(2\delta_0\gamma + 1)(6\delta_0\gamma + 1)(\delta_0(3\gamma - 2) + 3) \\
c_{11} =&48\gamma^2(2\delta_0\gamma + 1)(6\delta_0\gamma + 1)
\end{aligned}
$$