Discontinuous Galerkin Methods: Linear Systems and Hidden Accuracy

Discontinuous Galerkin Methods: Linear Systems and Hidden Accuracy

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
vrijdag 14 juni 2013 om 15:00 uur

door

Paulien van Slingerland

wiskundig ingenieur
geboren te Leiderdorp

Dit proefschrift is goedgekeurd door de promotor:

Prof.dr.ir. C. Vuik

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. C. Vuik	Technische Universiteit Delft, promotor
Prof.dr. Y. Notay	Université Libre de Bruxelles
Prof.dr. J. Qiu	Xiamen University
Prof.dr.ir. B. Koren	Technische Universiteit Eindhoven
Dr. H.Q. van der Ven	Nationaal Lucht- en Ruimtevaartlaboratorium
Prof.dr.ir. J.D. Jansen	Technische Universiteit Delft
Prof.dr.ir. C.W. Oosterlee	Technische Universiteit Delft

The research in this dissertation was carried out at and supported by Delft Institute of Applied Mathematics, Delft University of Technology. Part of the work was also sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-09-1-3055.

Discontinuous Galerkin Methods: Linear Systems and Hidden Accuracy.
Dissertation at Delft University of Technology.

ISBN: 978-94-6186-157-3

Summary

Discontinuous Galerkin Methods:
Linear Systems and Hidden Accuracy

Just like it is possible to predict tomorrow's weather, it is possible to predict e.g. the presence of oil in a reservoir, or the air flow around a newly designed air foil. These predictions are often based on computer simulations, for which the Discontinuous Galerkin (DG) method can be particularly suitable. This discretization scheme uses discontinuous piecewise polynomials of degree p to combine the best of both classical finite element and finite volume methods. This thesis focuses on its linear systems and 'hidden' accuracy.

Linear DG systems are relatively large and ill-conditioned. In search of efficient linear solvers, much attention has been paid to subspace correction methods. A particular example is a two-level preconditioner with a coarse space that is based on the DG scheme with polynomial degree $p = 0$. This more or less reduces the DG matrix to a (smaller) central difference matrix, for which many efficient linear solvers are readily available. An alternative for preconditioning is deflation. To contribute to the ongoing comparison between multigrid and deflation, and to extend the available analysis for the aforementioned two-level preconditioner, we have cast it into the deflation framework, and studied the impact of both variants on the convergence of the Conjugate Gradient (CG) method. This thesis discusses the results for Symmetric Interior Penalty (discontinuous) Galerkin (SIPG) discretizations for diffusion problems with strong contrasts in the coefficients. In addition, it considers the influence of the SIPG penalty parameter, weighted averages in the SIPG formulation (SWIP), the smoother, damping of the smoother, and the strategy for solving the coarse systems. We have found that both two-level methods yield fast and

scalable CG convergence (independent of the mesh element diameter), provided that the penalty parameter is chosen dependent on local values of the diffusion coefficient. The latter choice also benefits the discretization accuracy. Without damping, deflation can be up to 35% faster. If an optimal damping parameter is used, both two-level strategies yield similar efficiency. However, it remains an open question how the damping parameter can best be selected in practice.

At the same time, DG approximations can contain 'hidden' accuracy. For a large class of sufficiently smooth periodic problems, there exists a post-processor that enhances the DG convergence from order $p+1$ to order $2p+1$. Interestingly, this technique needs to be applied only once, at the final simulation time, and does not contain any information of the underlying physics or numerics. To be able to post-process near non-periodic boundaries and shocks as well, we have developed the so-called position-dependent post-processor, and analyzed its impact on the discretization accuracy in both the L^2- and the L^∞-norm. This thesis presents the results for DG (upwind) discretizations for hyperbolic problems, and demonstrates the benefits of post-processing for streamline visualization. Our numerical and theoretical results show that the position-dependent post-processor can enhance the DG convergence from order $p+1$ to order $2p+1$. Furthermore, unlike before, this technique can be applied in the entire domain to enhance the smoothness and accuracy of the DG approximation, even near non-periodic boundaries and shocks.

Altogether, this work contributes to shorter computational times and more accurate visualization of DG approximations. This strengthens the increasing consensus that DG methods can be an effective alternative for classical discretizations schemes, and sustains the idea that numerical approximations can contain more information than we originally thought.

Samenvatting

Discontinuous Galerkin Methoden: Lineare Stelsels en Verborgen Nauwkeurigheid

Net zoals het mogelijk is om het weer van morgen te voorspellen, is het mogelijk om bijvoorbeeld de aanwezigheid van olie in een reservoir te voorspellen, of de luchtstroming rond een nieuw ontwerp van een vliegtuigvleugel. Doorgaans zijn deze voorspellingen gebaseerd op computersimulaties, waar de *Discontinuous Galerkin* (DG) methode bij uitstek geschikt voor kan zijn. Dit discretisatieschema gebruikt stuksgewijze polynomen van graad p om de voordelen van klassieke eindige elementen en eindige volume methoden te combineren. Dit proefschrift focust op de bijbehorende lineaire stelsels en 'verborgen' nauwkeurigheid.

Lineaire DG stelsels zijn relatief groot en slecht geconditioneerd. Op zoek naar effciënte lineaire solvers is er veel aandacht besteed aan *subspace correction* methoden. Een specifiek voorbeeld is een *two-level* preconditionering waarbij de ruimte op het grove niveau gebaseerd is op het DG schema met polynomiale graad $p = 0$. Dit reduceert de DG matrix min of meer tot een (kleinere) centrale differentie matrix, waarvoor reeds veel efficiënte lineaire solvers beschikbaar zijn. Een alternatief voor preconditioneren is deflatie. Om bij te dragen aan de voortdurende vergelijking tussen multigrid en deflatie, en om de beschikbare analyse voor de eerder genoemde *two-level* preconditionering uit te breiden, hebben we deze in een deflatietechniek omgezet, en het effect van beide methoden op de convergentie van de *Conjugate Gradient* (CG) methode bestudeerd. Dit proefschrift beschrijft de resultaten voor *Symmetric Interior Penalty (discontinuous) Galerkin* (SIPG) discretizaties voor diffusieproblemen met sterke contrasten in de coëfficiënten. Daarnaast beschouwt het de invloed

van de SIPG *penalty* parameter, gewogen gemiddelden in de SIPG formulering (SWIP), de *smoother*, demping van de *smoother*, en de strategie om de lineaire stelsels op het grove niveau op de lossen. We hebben gevonden dat beide *two-level* methoden snelle en schaalbare CG convergentie leveren (onafhankelijk van de diameter van de grid elementen), mits de *penalty* parameter afhankelijk van lokale waarden van de diffussiecoëfficiënt gekozen wordt. Dit laatste verbetert ook de nauwkeurigheid van de discretisatie. Zonder demping kan deflatie tot 35% sneller zijn. Als een optimale demping parameter gebruikt wordt, zijn beide *two-level* strategieën ongeveer even efficiënt. Desalniettemin is het een open vraag hoe de penalty parameter het beste gekozen kan worden in de praktijk.

Tegelijkertijd kunnen DG benaderingen 'verborgen' nauwkeurigheid bevatten. Voor een grote klasse van voldoende gladde periodieke problemen bestaat er een filter, die de DG convergentie verbetert van orde $p+1$ naar orde $2p+1$. Interssant genoeg hoeft deze techniek slechts één keer toegepast te worden, terwijl deze geen enkele informatie bevat over de onderliggende fysische en numerieke vergelijkingen. Om ook nabij niet-periodieke randen en schokken te kunnen filteren, hebben we het zogeheten positie-afhankelijke filter ontwikkeld, en zijn effect geanalyseerd op de nauwkeurigheid van de discretisatie in de L^2- en de L^∞-norm. Dit proefschrift presenteert de resultaten voor DG (upwind) discretisaties voor hyperbolische problemen, en demonstreert de voordelen van filteren voor visualisaties in de vorm van stroomlijnen. Daarnaast, in tegenstelling tot hiervoor, kan deze techniek toegepast worden in het gehele domein om de gladheid en nauwkeurigheid van de DG approximatie te verbeteren, zelfs nabij niet-periodieke randen en schokken.

Al met al draagt dit werk bij aan kortere rekentijden en nauwkeuriger visualisatie van DG approximaties. Dit versterkt de toenemende consensus dat DG methoden een effectief alternatief kunnen zijn voor klassieke discretisatieschema's, en onderschrijft het idee dat numerieke benaderingen meer informatie kunnen bevatten dan we oorspronkelijk dachten.

Contents

1 Introduction **1**
- 1.1 Introduction . . . 1
- 1.2 Discontinuous Galerkin (DG) methods . . . 2
- 1.3 Linear DG systems . . . 4
- 1.4 Hidden DG accuracy . . . 5
- 1.5 Outline . . . 6

2 Linear DG systems **7**
- 2.1 Introduction . . . 8
- 2.2 Discretization . . . 8
  - 2.2.1 SIPG method for diffusion problems . . . 9
  - 2.2.2 Linear system . . . 10
  - 2.2.3 Penalty parameter . . . 11
- 2.3 Two-level preconditioner . . . 12
  - 2.3.1 Coarse correction operator . . . 12
  - 2.3.2 Two-level preconditioner . . . 14
  - 2.3.3 Implementation in a CG algorithm . . . 14
- 2.4 Deflation variant . . . 15
  - 2.4.1 Two-level deflation . . . 15
  - 2.4.2 FLOPS . . . 16
  - 2.4.3 Coarse systems . . . 17
  - 2.4.4 Damping . . . 17
- 2.5 Numerical experiments . . . 18
  - 2.5.1 Experimental setup . . . 18
  - 2.5.2 The influence of the penalty parameter . . . 20
  - 2.5.3 Coarse systems . . . 23
  - 2.5.4 Smoothers and damping . . . 24

2.5.5 Other test cases . . . 26
2.6 Conclusion . . . 26

3 Theoretical scalability **33**
3.1 Introduction . . . 34
3.2 Methods and assumptions . . . 34
3.2.1 SIPG discretization for diffusion problems . . . 35
3.2.2 Linear system . . . 36
3.2.3 Two-level preconditioning and deflation . . . 37
3.3 Abstract relations for any SPD matrix A . . . 39
3.3.1 Using the error iteration matrix . . . 39
3.3.2 Implications for the two-level methods . . . 41
3.3.3 Comparing deflation and preconditioning . . . 42
3.4 Intermezzo: regularity on the block diagonal of A . . . 43
3.4.1 Using regularity of the mesh . . . 44
3.4.2 The desired result in terms of 'small' bilinear forms . . . 45
3.4.3 Regularity on the block diagonal of A . . . 48
3.5 Main result: scalability for SIPG systems . . . 50
3.5.1 Main result: scalability for SIPG systems . . . 50
3.5.2 Special case: block Jacobi smoothing . . . 52
3.5.3 Influence of damping and the penalty parameter for block Jacobi smoothing . . . 54
3.6 Conclusion . . . 56

4 Hidden DG accuracy **57**
4.1 Introduction . . . 58
4.2 Discretization . . . 59
4.2.1 DG for one-dimensional hyperbolic problems . . . 59
4.2.2 DG for two-dimensional hyperbolic systems . . . 59
4.3 Original post-processing strategies . . . 60
4.3.1 B-splines . . . 61
4.3.2 Symmetric Post-processor . . . 61
4.3.3 One-sided post-processor . . . 63
4.4 Position-dependent post-processor . . . 65
4.4.1 Generalized post-processor . . . 65
4.4.2 Position-dependent post-processor . . . 67
4.4.3 Post-processing in two dimensions . . . 69
4.5 Numerical Results . . . 70
4.5.1 L^2-Projection . . . 71
4.5.2 Constant coefficients . . . 73
4.5.3 Dirichlet BCs . . . 75
4.5.4 Variable coefficients . . . 76
4.5.5 Discontinuous coefficients . . . 77
4.5.6 Two-dimensional system . . . 78
4.5.7 Two-dimensional streamlines . . . 78

4.6 Conclusion 80

5 Theoretical Superconvergence **85**
5.1 Introduction 86
5.2 Methods and notation 86
5.2.1 Post-processor 86
5.2.2 DG discretization for hyperbolic problems 88
5.2.3 Additional notation 88
5.3 Auxiliary results 90
5.3.1 Estimating $\|u - u^\star\|$ 90
5.3.2 Derivatives of B-splines 92
5.4 The main result in abstract form 94
5.4.1 Reducing the post-processor to its building blocks . . . 94
5.4.2 Treating the remaining building blocks 96
5.4.3 The main error estimate in abstract form 98
5.5 The main result for DG approximations 99
5.5.1 Unfiltered DG convergence 99
5.5.2 Main result: extracting DG superconvergence 102
5.5.3 Implications for the position-dependent post-processor . 103
5.6 Conclusion 104

6 Conclusion **105**
6.1 Introduction 105
6.2 Linear DG systems 105
6.3 Hidden DG accuracy 106
6.4 Future research 107

1 Introduction

1.1 Introduction

Just like it is possible to predict tomorrow's weather, it is possible to predict the presence of oil in a reservoir, or the air flow around a newly designed air foil, for instance. Such predictions can be based on field or model experiments, but also on computer simulations. The latter can be significantly cheaper (in terms of both time and money), and allow us to study potentially dangerous situations in a safe and virtual setting.

Unfortunately, for most real-life applications, the underlying physical model is too complex to calculate the corresponding solution exactly. Hence, we must settle for an approximation instead. The mathematical field of numerical analysis is concerned with advanced numerical algorithms for this purpose.

In principle, the numerical algorithms available can provide solution approximations with an arbitrarily small error. However, the computational time (and computer memory) must also been taken into account: tomorrow's weather forecast should not take a week to compute, for instance. This trade-off between accuracy and efficiency can be compared to using pixels in a digital camera: more pixels mean sharper photos, but also more data to process and store. It remains a challenge to develop numerical algorithms with 'smarter pixels' to improve this balance for many applications.

In this context, a particular challenge is formed by problems with shocks or strong contrasts in the model coefficients. The latter is usually the case for oil reservoir simulations, for example, where layers of different material (sand, shale, etc.) may result in permeability contrasts with typical values between 10^{-1} and 10^{-7}. In this thesis, we seek to improve on existing numerical algorithms for such applications. In particular, we focus on so-called Discontinuous Galerkin (DG) discretizations schemes, and their linear systems and 'hidden accuracy'.

The remaining of this chapter introduces this research in the following man-

ner: Section 1.2 provides a short introduction to Discontinuous Galerkin (DG) discretizations and their advantages in the context of finite volume schemes. Section 1.3 motivates the importance of an efficient solver for the corresponding linear systems. Section 1.4 discusses the hidden accuracy that DG approximations can contain and its extraction through post-processing. Finally, Section 1.5 provides an outline of this thesis.

1.2 Discontinuous Galerkin (DG) methods

Discontinuous Galerkin (DG) methods [63, 4, 21, 39] are flexible discretization schemes for approximating the solution to partial differential equations. A DG method can be thought of as a Finite Volume Method (FVM) that uses piecewise polynomials rather than piecewise constants. More specifically, for a given problem and mesh, a DG approximation is a polynomial of degree p or lower within each mesh element, while it may be discontinuous at the element boundaries. As such, it combines the best of both classical finite element and finite volume methods. This makes DG methods particularly suitable for handling non-matching grids and local *hp*-refinement strategies.

A DG method typically yields a higher convergence rate than a FVM (assuming similar numerical fluxes), as it makes use of a higher polynomial degree. Figure 1.1 illustrates this for a one-dimensional problem with five mesh elements.[1] This higher order accuracy comes at a higher price though. While the FVM requires only one unknown per mesh element, a DG method requires multiple unknowns per mesh element: one for each polynomial basis function, e.g. $p+1$ for one-dimensional problems, and $\frac{(p+1)(p+2)}{2}$ for two-dimensional problems. As a consequence, DG matrices are relatively large (both in size and in terms of the total number of nonzeros), resulting in more computational costs. Figure 1.2 illustrates this for a two-dimensional Laplace problem on a 3×3 uniform Cartesian mesh.

In this light, it is interesting to compare the accuracy of the DG method and the FVM for a fixed number of unknowns, rather than a fixed number of mesh elements. Figure 1.3 demonstrates that a DG method can yield much higher accuracy for the same number of unknowns.[2] Together with the advantages mentioned earlier, this motivates the current study of DG methods in this thesis.

[1] In the illustrations in this section, the DG scheme under consideration is the SIPG method specified in Section 2.2 and 3.2. The FVM method is based on central fluxes.

[2] In this figure, we study a two-dimensional diffusion problem with five layers, where the diffusion K is either 1 or 0.001 in each layer. The meshes are Cartesian and uniform. See Section 2.5.1 for further details.

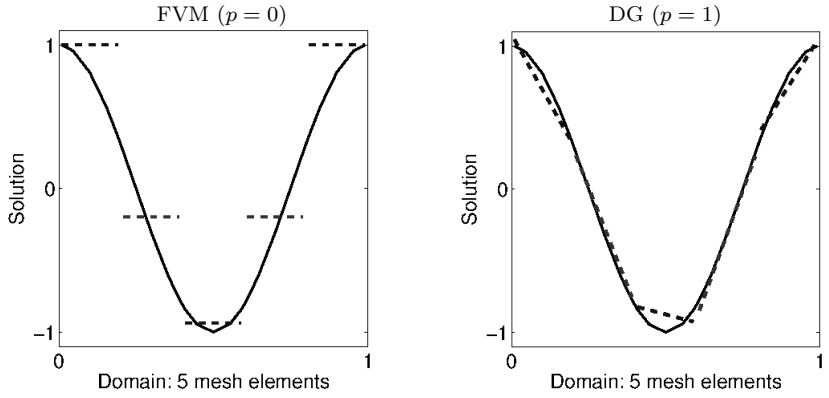


FIGURE 1.1. *A DG method typically yields a higher convergence rate than a FVM, as it makes use of a higher polynomial degree p.*

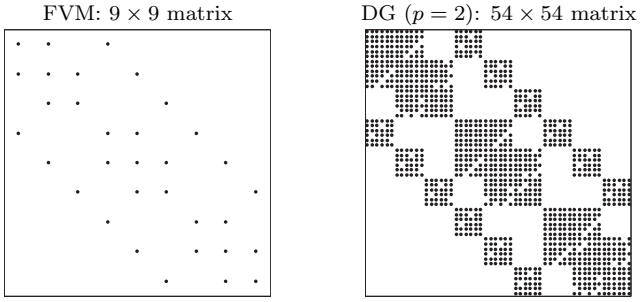


FIGURE 1.2. *DG matrices are relatively large (both for a 3 × 3 mesh).*

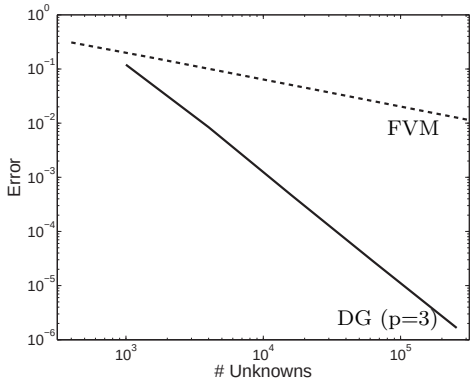


FIGURE 1.3. *DG discretizations can yield significantly better accuracy (in the L^2-norm) for the same number of unknowns.*

1.3 Linear DG systems

In the previous section, we have seen that linear DG systems are relatively large. At the same time, their condition number typically increases with the number of mesh elements, the polynomial degree, and the stabilization factor [17, 72]. Figure 1.4 illustrates this.[3] Problems with extreme contrasts in the coefficients (cf. Section 1.1) usually pose an extra challenge.

In search of efficient and scalable algorithms (for which the number of iterations does not increase with e.g. the number of mesh elements), much attention has been paid to subspace correction methods [83]. For example, Schwarz domain decomposition methods are based on subspaces that arise from subdividing the spatial domain into smaller subdomains [3, 32]; geometric (h-)multigrid methods make use of subspaces resulting from multiple coarser meshes [34, 13]; spectral (p-)multigrid methods apply global corrections by solving problems with a lower polynomial degree [33, 58]; and algebraic multigrid methods use algebraic criteria to separate the unknowns of the original system into two sets, one of which is labeled 'coarse' [59, 68]. A particular example makes use of a single coarse space that is based on the DG scheme with polynomial degree $p = 0$. This two-level method, proposed by Dobrev et al. [24], more or less reduces the DG matrix to a (smaller) central difference matrix, for which many efficient linear solvers are readily available.

Usually, the subspace correction methods above can either be used as a standalone solver, or as a preconditioner in an iterative Krylov method. The second strategy tends to be more robust for problems with a few isolated 'bad' eigenvalues, which is common for problems with strongly varying coefficients.

An alternative for preconditioning is the method of deflation. This technique has been developed in the late eighties by Nicolaides [55], Dostal [26] and Mansfield [46], and further studied in [69, 80, 81], among others. Deflation is strongly related to the subspace correction methods mentioned earlier, as found in [52, 53, 54, 76, 75].

To contribute to this ongoing comparison between multigrid and deflation, and to extend the available analysis for the aforementioned two-level preconditioner [24], we have cast it into the deflation framework, and studied the impact of both variants on the convergence of the Conjugate Gradient (CG) method. This thesis discusses the results for Symmetric Interior Penalty (discontinuous) Galerkin (SIPG) discretizations for diffusion problems with strong contrasts in the coefficients. In addition, it considers the influence of the SIPG penalty parameter, weighted averages in the SIPG formulation (SWIP), the smoother, damping of the smoother, and the strategy for solving the coarse systems.

[3] In this figure, we study the two-dimensional Laplace problem on a uniform Cartesian mesh with 10×10 mesh elements. We use the SIPG discretization with (stabilizing) penalty parameter $\sigma = 2p^2$. See Section 2.5.1 for further details.

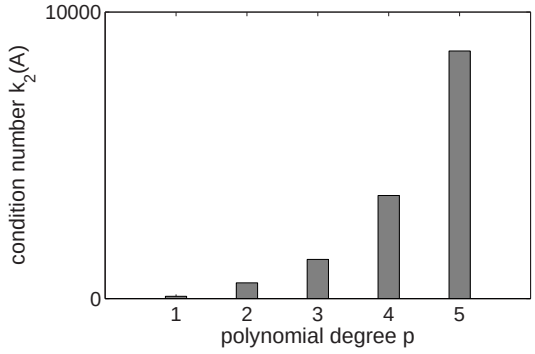


FIGURE 1.4
The condition number of a DG matrix can increase rapidly with the polynomial degree p.

1.4 Hidden DG accuracy

DG approximations can contain 'hidden accuracy': although the convergence rate of a DG scheme is typically of order $p+1$ (where p is the polynomial degree), the Fourier analysis of Lowrie [45] reveals an accurate mode that evolves with an accuracy of order $2p + 1$. Furthermore, Adjerid et al. [1] proved super-convergence of order $p + 2$ at the roots of a Radau polynomial of degree $p + 1$ on each element, and of order $2p + 1$ at the downwind point of each element. See also [2] and references therein.

This hidden accuracy can be extracted by applying a local averaging operator, which has been introduced by Bramble and Schatz [11] in the context of Ritz-Galerkin approximations for elliptic problems. They showed that this so-called symmetric post-processor yields super-convergence of order $2p + 1$. Interestingly, this technique needs to be applied only once, at the final simulation time, and does not contain any information of the underlying physics or numerics.

Thomee [77] provided alternative proofs for the results in [11] using Fourier transforms. This point of view reveals that the symmetric post-processor is related to the Lanczos filter (see e.g. [38, p. 163]), which is a classical filter in the context of spectral methods for reducing the Gibbs phenomenon. Thomee also demonstrated that a modified version of the post-processor can be used to obtain accurate derivative approximations. This work was extended in [78] for semidiscrete Galerkin finite element methods for parabolic problems.

Cockburn, Luskin, Shu, and Süli [22] combined the ideas in [11] and [51] to apply the post-processor to DG approximations for linear periodic hyperbolic PDEs. They showed that super-convergence of order $2p + 1$ can be extracted by the symmetric post-processor for this alternative application type. Since then, the post-processor has been modified to be able to post-process near non-periodic boundaries [64], applied to extract derivatives (following [77]) [65, 66], improved computationally [49, 50], studied for non-uniform rectangular [23], structured triangular [47] and unstructured triangular [48] meshes, analyzed for

linear convection-diffusion problems [41], and nonlinear hyperbolic conservation laws [42], and used to enhance streamline visualizations [73, 82].

To improve on the accuracy and smoothness of the one-sided post-processor [64] near non-periodic boundaries and shocks, we have developed the so-called position-dependent post-processor, and analyzed its impact on the discretization accuracy in both the L^2- and the L^∞-norm. This thesis presents the results for DG (upwind) discretizations for hyperbolic problems, and demonstrates the benefits of the proposed post-processor for streamline visualization.

1.5 Outline

The outline of this thesis is as follows.

Chapter 2 discusses the two-level preconditioner proposed by Dobrev et al. [24], and casts it into the deflation framework. The difference between both methods is studied through various numerical experiments.

Chapter 3 provides theoretical support for the convergence of both the two-level preconditioner and the deflation variant. In particular, it derives upper bounds for the condition number of the preconditioned/deflated system.

Chapter 4 discusses the original (symmetric and) one-sided post-processor [64], and proposes the position-dependent post-processor. Both techniques are compared numerically in terms of smoothness and accuracy.

Chapter 5 presents theoretical error estimates in both the L^2- and the L^∞-norm for the position-dependent post-processor.

Chapter 6 summarizes the main conclusions of this thesis and indicates possibilities for future research.

2

Linear DG systems

This chapter is based on:

P. van Slingerland, C. Vuik, *Fast linear solver for pressure computation in layered domains*. Submitted to Comput. Geosci.

2.1 Introduction

Linear DG systems are relatively large and ill-conditioned (cf. Section 1.3). In search of efficient linear solvers, much attention has been paid to subspace correction methods. A particular example is a two-level preconditioner proposed by Dobrev et al. [24]. This method uses coarse corrections based on the DG discretization with polynomial degree $p = 0$. Using the analysis of Falgout et al. [31], Dobrev et al. have shown theoretically (for polynomial degree $p = 1$) that this preconditioner yields scalable convergence of the CG method, independent of the mesh element diameter. Another nice property is that the use of only two levels offers an appealing simplicity. More importantly, the coefficient matrix that is used for the coarse correction is quite similar to a matrix resulting from a central difference discretization, for which very efficient solution techniques are readily available.

An alternative for preconditioning is the method of deflation. This method has been proved effective for layered problems with extreme contrasts in the coefficients in [80, 81]. Deflation is related to multigrid in the sense that it also makes use of a coarse space that is combined with a smoothing operator at the fine level. This relation has been considered from an abstract point of view by Tang et al. in [76, 75].

To continue this comparison between preconditioning and deflation in the context of DG schemes, and to extend the work in [24], we have cast the two-level preconditioner into the deflation framework, and studied the impact of both variants on the convergence of the Conjugate Gradient (CG) method. This chapter discusses the numerical results for Symmetric Interior Penalty (discontinuous) Galerkin (SIPG) discretizations for diffusion problems with strong contrasts in the coefficients. In addition, it considers the influence of the SIPG penalty parameter, weighted averages in the SIPG formulation, the smoother, damping of the smoother, and the strategy for solving the coarse systems. Theoretical analysis of both two-level methods will be considered in Chapter 3.

The outline of this chapter is as follows. Section 2.2 discusses the SIPG method for diffusion problems with large jumps in the coefficients. To solve the resulting systems, Section 2.3 discusses the two-level preconditioner. Section 2.4 rewrites the latter as a deflation method. Section 2.5 compares the performance of both two-level methods through various numerical experiments. Section 2.6 summarizes the main conclusions.

2.2 Discretization

This section specifies the DG discretization under consideration. Section 2.2.1 discusses the SIPG method for stationary diffusion problems following [63]. Section 2.2.2 describes the resulting linear systems. Section 2.2.3 motivates the use of a diffusion-dependent penalty parameter following [27].

2.2.1 SIPG method for diffusion problems

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

$$\begin{aligned} -\nabla\cdot(K\nabla u) &= f, && \text{in } \Omega, \\ u &= g_D, && \text{on } \partial\Omega_D, \\ K\nabla u\cdot\mathbf{n} &= g_N, && \text{on } \partial\Omega_N. \end{aligned} \tag{2.1}$$

We assume that the diffusion K is a symmetric and positive-definite tensor whose eigenvalues are bounded below and above by positive constants, and that the other model parameters are chosen such that a weak solution of (2.1) exists[1].

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

$$B(u_h, v) = L(v), \qquad \text{for all test functions } v \in V, \tag{2.2}$$

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

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

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

[1] That is, $f, g_N \in L^2(\Omega)$, and $g_D \in H^{\frac{1}{2}}(\Omega)$ [63, p. 25, 26].

where σ is the so-called penalty parameter. This positive parameter penalizes the inter-element jumps to enforce weak continuity and ensure convergence. Although it is presented as a constant here, its value may vary throughout the domain. This is discussed further in Section 2.2.3 later on.

For a large class of sufficiently smooth problems, the SIPG method yields convergence of order $p+1$ [63].

2.2.2 Linear system

In order to compute the SIPG approximation defined by (2.2), it needs to be rewritten as a linear system. To this end, we choose basis functions for the test space V. More specifically, for each mesh element E_i, we define the basis function $\phi_1^{(i)}$ which is zero in the entire domain, except in E_i, where it is equal to one. Similarly, we define higher-order basis functions $\phi_2^{(i)}, ..., \phi_m^{(i)}$, which are higher-order polynomials in E_i and zero elsewhere. In this chapter, we use monomial basis functions.

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

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

where k_x and k_y are selected as follows:

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

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

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

$$u_h = \sum_{i=1}^{N} \sum_{k=1}^{m} u_k^{(i)} \phi_k^{(i)}. \tag{2.3}$$

The new unknowns $u_k^{(i)}$ in (2.3) can be determined by solving a linear system $A\mathbf{u} = \mathbf{b}$ of the form:

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

where the blocks all have dimension m:

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

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

for all $i, j = 1, ..., N$. This system is obtained by substituting the expression (2.3) for u_h and the basis functions $\phi_\ell^{(j)}$ for v into (2.2). Once the unknowns $u_k^{(i)}$ are solved from the system $A\mathbf{u} = \mathbf{b}$, the final SIPG approximation u_h can be obtained from (2.3).

2.2.3 Penalty parameter

The SIPG method involves the penalty parameter σ which penalizes the inter-element jumps to enforce weak continuity. This parameter should be selected carefully: on the one hand, it needs to be sufficiently large to ensure that the SIPG method converges and the coefficient matrix A is Symmetric and Positive Definite (SPD) [63]. At the same time, it needs to be chosen as small as possible, since the condition number of A increases rapidly with the penalty parameter [17].

Computable theoretical lower bounds for a large variety of problems have been derived by Epshteyn and Riviere [28]. For one-dimensional diffusion problems, it suffices to choose $\sigma \geq 2\frac{k_1^2}{k_0}p^2$, where k_0 and k_1 are the *global* lower and upper bound respectively for the diffusion coefficient K. However, while this lower bound for σ is sufficient to ensure convergence (assuming the exact solution is sufficiently smooth), it can be unpractical for problems with strong variations in the coefficients. For instance, if the diffusion coefficient K takes values between 1 and 10^{-3}, we obtain $\sigma \geq 2000p^2$, which is inconveniently large. For this reason, it is common practice to choose e.g. $\sigma = 20$ rather than $\sigma = 20\,000$ for such problems [24, 60].

An alternative strategy is to choose the penalty parameter based on *local* values of the diffusion-coefficient K, e.g. choosing $\sigma = 20K$ rather than $\sigma = 20$. It has been demonstrated numerically in [25] that such a diffusion-dependent penalty parameter can benefit the efficiency of a linear solver (also cf. Section 2.5.2). For general tensors K, this strategy can be defined as follows: for an edge with normal $\mathbf{n}$, we set $\sigma = \alpha\lambda$, where $\lambda = \mathbf{n}^T K \mathbf{n}$ and α is a user-defined constant. The latter should be as small as possible in light of the discussion above.

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

An alternative strategy for dealing with discontinuities is to use the harmonic average of both limit values [27, 15, 29, 7]. In this case, the penalty parameter reads $\sigma = 2\alpha\frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$, where λ_i and λ_j are based on the information in the mesh elements E_i and E_j respectively (adjacent to the edge under consideration). This choice is equivalent to using the minimum of both limit values [7, p. 5]. In that sense, it seems less 'safe' than the maximum strategy above.

In [27, 15, 29, 7], the 'harmonic' penalty parameter is used in combination with the so-called Symmetric Weighted Interior Penalty (SWIP) method. The main difference between the standard SIPG method and the SWIP method is the following: whenever an average of a function at a mesh element boundary is considered (denoted by $\{.\}$ in Section 2.2.1), the SWIP method uses a weighted average rather than the standard average. For this purpose, the weights typically depend on the diffusion coefficient, i.e. $w_i = \frac{\lambda_j}{\lambda_i+\lambda_j}$ and $w_j = \frac{\lambda_i}{\lambda_i+\lambda_j}$ (note that the harmonic penalty can then be written as $\sigma = \alpha(w_i\lambda_i + w_j\lambda_j)$).

In this chapter, we study the effects of both a constant and a diffusion-dependent penalty parameter, using either the maximum or the harmonic strategy above. Furthermore, we consider both the SIPG and the SWIP method.

2.3 Two-level preconditioner

To solve the linear SIPG system obtained in the previous section, we start by considering the two-level preconditioner proposed by Dobrev et al [24]. Section 2.3.1 specifies the corresponding coarse correction operator. Section 2.3.2 defines the resulting two-level preconditioner. Section 2.3.3 indicates its implementation in a standard preconditioned CG algorithm.

2.3.1 Coarse correction operator

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

More specifically, the coarse correction operator Q is defined as follows. Let R denote the so-called restriction operator such that $A_0 := RAR^T$ is the SIPG matrix for polynomial degree $p = 0$. More specifically, the matrix R is defined

as the following $N \times N$ block matrix:

$$R = \begin{bmatrix} R_{11} & R_{12} & \dots & R_{1N} \\ R_{21} & R_{22} & & \vdots \\ \vdots & & \ddots & \\ R_{N1} & \dots & & R_{NN} \end{bmatrix},$$

where the blocks all have size $1 \times m$:

$$R_{ii} = \begin{bmatrix} 1 & 0 \dots & 0 \end{bmatrix}, \qquad R_{ij} = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix},$$

for all $i, j = 1, ..., N$ and $i \neq j$. Using this notation, the coarse correction operator is defined as

$$Q := R^T A_0^{-1} R \tag{2.4}$$

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

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

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

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

Observe that A_0 has the same structure as a central difference matrix (aside from a factor $\sigma = 10$). Furthermore, every element in A_0 is also present in the

upper left corner of the corresponding block in the matrix A. This is because the piecewise constant basis functions are assumed to be in any polynomial basis. As a consequence, the matrix R contains elements equal to 0 and 1 only, and does not need to be stored explicitly: multiplications with R can be implemented by simply extracting elements or inserting zeros.

2.3.2 Two-level preconditioner

We can now formulate the two-level preconditioner proposed by Dobrev et al. [24]. This is established by combining the coarse correction operator Q with a (damped) smoother:

Definition 2.3.1 (Two-level preconditioner) Consider the coarse correction operator Q defined in (2.4), a damping parameter $\omega \leq 1$, and an invertible smoother $M^{-1} \approx A^{-1}$. Then, the result $\mathbf{y} = P_{\text{prec}}\mathbf{r}$ of applying the two-level preconditioner to a vector $\mathbf{r}$ can be computed as follows:

$$\begin{aligned} \mathbf{y}^{(1)} &= \omega M^{-1}\mathbf{r} && \text{(pre-smoothing)}, \\ \mathbf{y}^{(2)} &= \mathbf{y}^{(1)} + Q(\mathbf{r} - A\mathbf{y}^{(1)}) && \text{(coarse correction)}, \\ \mathbf{y} &= \mathbf{y}^{(2)} + \omega M^{-T}(\mathbf{r} - A\mathbf{y}^{(2)}) && \text{(post-smoothing)}. \end{aligned} \tag{2.5}$$

In this chapter, we consider block Jacobi and block Gauss-Seidel smoothing. These smoothers have the following property (cf. Chapter 3):

$$M + M^T - \omega A \text{ is SPD.} \tag{2.6}$$

Using the more abstract analysis in [79, p. 66], condition (2.6) implies that the preconditioning operator P_{prec} is SPD. As a consequence, the two-level preconditioner can be implemented in a standard preconditioned CG algorithm (cf. Section 2.3.3 hereafter).

Requirement (2.6) also implies that the two-level preconditioner yields scalable convergence of the CG method (independent of the mesh element diameter) for a large class of problems. This has been shown for polynomial degree $p = 1$ by Dobrev et al. [24], using the analysis in [31]. In Chapter 3, we will extend this theory for $p > 1$.

2.3.3 Implementation in a CG algorithm

Assuming (2.6), the two-level preconditioner is SPD and can be implemented in a standard preconditioned CG algorithm. Below, we summarize the implementation of this scheme for a given preconditioning operator P and start vector $\mathbf{x}_0$:

1. $\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0$

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

2.4 Deflation variant

Next, we cast the two-level preconditioner into the deflation framework using the abstract analysis in [76]. Section 2.4.1 defines the resulting operator. Section 2.4.2 compares the two-level preconditioner and the corresponding deflation variant in terms of computational costs. Section 2.4.3 discusses the coarse systems involved in each iteration. Section 2.4.4 considers the influence of damping of the smoother.

2.4.1 Two-level deflation

There are multiple strategies to construct a deflation method based on the components of the two-level preconditioner. An overview is given in [76]. Below, we consider the so-called 'ADEF2' deflation scheme, as this type can be implemented relatively efficiently, and allows for inexact solving of coarse systems (cf. Section 2.4.3 later on).

Basically, this deflation variant is obtained from Definition 2.3.1 by skipping the last smoothing step in (2.5).

Definition 2.4.1 (Two-level deflation) Consider the coarse correction operator Q defined in (2.4), a damping parameter $\omega \leq 1$, and an invertible smoother $M^{-1} \approx A^{-1}$. Then, the result $\mathbf{y} = P_{\text{defl}}\mathbf{r}$ of applying the two-level deflation technique to a vector $\mathbf{r}$ can be computed as:

$$\begin{aligned} \mathbf{y}^{(1)} &:= \omega M^{-1}\mathbf{r} && \text{(pre-smoothing)}, \\ \mathbf{y} &:= \mathbf{y}^{(1)} + Q(\mathbf{r} - A\mathbf{y}^{(1)}) && \text{(coarse correction)}. \end{aligned} \tag{2.7}$$

The operator P_{defl} is not symmetric in general. As such, it seems unsuitable for the standard preconditioned CG method. Interestingly, it can still be implemented successfully in its current asymmetric form, as long as the smoother

M^{-1} is SPD (requirement (2.6) is not needed), and the start vector $\mathbf{x}_0$ is pre-processed according to:

$$\mathbf{x}_0 \mapsto Q\mathbf{b} + (I - AQ)^T\mathbf{x}_0. \tag{2.8}$$

Other than that, the CG implementation remains as discussed in Section 2.3.3. Indeed, it has been shown by [76, Theorem 3.4] that, under the aforementioned conditions, P_{defl} yields the same CG iterates as an alternative operator (called 'BNN', cf. Section 3.2.3) that actually *is* SPD.

It will be shown in Chapter 3 that, similar to the two-level preconditioner, the deflation variant above yields scalable convergence of the CG method (independent of the mesh element diameter).

2.4.2 FLOPS

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

Table 2.1 displays the costs for a two-dimensional diffusion problem with polynomial degree p, a Cartesian mesh with $N = n \times n$ elements, and polynomial space dimension $m := \frac{(p+1)(p+2)}{2}$. Using the preconditioning variant, the CG method requires per iteration $(27m^2 + 14m)N$ flops, plus the costs for two smoothing steps and one coarse solve. Using the two-level deflation method, the CG method requires per iteration $(18m^2 + 12m)N$ flops, plus the costs for only one smoothing step and one coarse solve.

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

operation	flops (rounded)	# defl.	# prec.
mat-vec (Au)	$9m^2N$	2	3
inner product (u^Tv)	$2mN$	2	2
scalar multiplication (αu)	mN	3	3
vector update ($u \pm v$)	mN	5	7
smoothing ($M^{-1}u$)	variable	1	2
coarse solve ($A_0^{-1}u_0$)	variable	1	1

TABLE 2.1. *Comparing the computational costs per CG iteration for A-DEF2 deflation and the two-level preconditioner for our applications.*

2.4.3 Coarse systems

Both two-level methods require the solution of a coarse system in each iteration, involving the coefficient matrix A_0. In Section 2.3.1, we have seen that A_0 has the same structure and size ($N \times N$) as a central difference matrix. As a consequence, a *direct* solver is not feasible for most practical applications. At the same time, many effective *inexact* solvers are readily available for this type of system.

For some deflation methods, including DEF1, DEF2, R-BNN1, R-BNN2, such an inexact coarse solver is not suitable. This is because those methods contain eigenvalue clusters at 0, so small perturbations in those schemes (e.g. due to inexact coarse solves) can result in an "unfavorable spectrum, resulting in slow convergence of the method" —– [76, p. 353]. ADEF2 does not have this limitation, as it clusters these eigenvalues at 1 rather than 0. This is one of the reasons why we focus on this particular deflation variant.

In Section 2.5.3 we will investigate the use of an inexact coarse solver that applies the CG method in an inner loop with a scalable algebraic multigrid preconditioner. This strategy will be studied for both the two-level preconditioner, and the ADEF2 deflation variant.

An alternative strategy is the Flexible CG (FCG) method [6, 56], which can be used to relax the stopping criterion for inner iterations while remaining as effective as a direct solver. The main difference with standard CG lies in the explicit orthogonalization and truncation of the search direction vectors, possibly combined with a restart strategy. We do not study this topic further in this thesis.

2.4.4 Damping

Damping often benefits the convergence of multigrid methods [84]. For multigrid methods with smoother $M = I$, a "typical choice of $[\omega]$ is close to $\frac{1}{||A||_2}$", although a "better choice of $[\omega]$ is possible if we make further assumptions on how the eigenvectors of A associated with small eigenvalues are treated by coarse-grid correction" —– [75, p. 1727]. In that reference, the latter is established for a coarse space that is based on a set of orthonormal eigenvectors of A. However, such a result does not seem available yet for the coarse space (and smoothers) currently under consideration.

At the same time, deflation may not be influenced by damping at all. The latter has been observed theoretically in [75, p. 1727] for the DEF(1) variant. For the ADEF2 variant under consideration, such a result is not yet available.

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

2.5 Numerical experiments

Next, we compare the two-level preconditioner and the corresponding deflation variant through numerical experiments. Section 2.5.1 specifies the experimental setup. Section 2.5.2 studies the influence of SIPG penalty parameter. Section 2.5.3 investigates the effectiveness of an inexact solver for the coarse systems. Section 2.5.4 studies the impact of (damping of) the smoother on the overall computational efficiency. Section 2.5.5 considers similar experiments for more challenging test cases.

2.5.1 Experimental setup

We consider multiple diffusion problems of the form (2.1) with strong contrasts in the coefficients on the domain $[0,1]^2$. At first, we primarily focus on the problem illustrated in Figure 2.1. This problem has five layers, and the diffusion is either 1 or 10^{-3} in each layer. Such strong contrasts in the coefficients are typically encountered during oil reservoir simulations involving e.g. layers of sand and shale. In Section 2.5.5, we also study problems that mimic the occurrence of sand inclusions within a layer of shale, and ground water flow. Furthermore, we examine a bubbly flow problem, the influence of Neumann boundary conditions, and a full anisotropic problem.

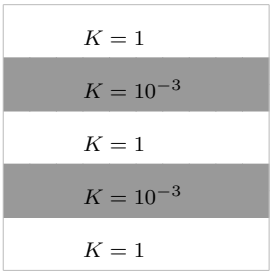


FIGURE 2.1. *Illustration of the problem with five layers*

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

All model problems are discretized by means of the SIPG method as discussed in Section 2.2, although the SWIP variant with weighted averages is also discussed. We use a uniform Cartesian mesh with $N = n \times n$ elements, where $n = 20, 40, 80, 160, 320$. Furthermore, we use monomial basis functions with polynomial degree $p = 2, 3$ (results for $p = 1$ are similar though). For $p = 3$ and $n = 320$, this means that the number of degrees of freedom is a little

over 10^6. In most cases, the penalty parameter is chosen diffusion-dependent, $\sigma = 20\mathbf{n}^T K\mathbf{n}$, using the largest limit value at discontinuities (cf. Section 2.2.3). However, we also study a constant penalty parameter, and a parameter based on harmonic means.

The linear systems resulting from the SIPG discretizations are solved by means of the CG method, combined with either the two-level preconditioner (Definition 2.3.1) or the corresponding deflation variant (Definition 2.4.1). Unless specified otherwise, damping is not used. For the smoother M^{-1}, we use block Jacobi with small blocks of size $m \times m$ (recall that $m = \frac{(p+1)(p+2)}{2}$). For the preconditioning variant, we also consider block Gauss-Seidel with the same block size (deflation requires a symmetric smoother).

Diagonal scaling is applied as a pre-processing step in all cases, and the same random start vector $\mathbf{x}_0$ is used for all problems of the same size. Pre-processing of the start vector according to (2.8) is applied for deflation only, as it makes no difference for the preconditioning variant. For the stopping criterion we use:

$$\frac{\|r_k\|_2}{\|b\|_2} \leq \text{TOL}, \tag{2.9}$$

where $\text{TOL} = 10^{-6}$, and r_k is the residual after the k^{th} iteration.

Coarse systems, involving the SIPG matrix A_0 with polynomial degree $p = 0$, are solved directly in most cases. However, a more efficient alternative is provided in Section 2.5.3. In any case, the coarse matrix A_0 is quite similar to a central difference matrix, for which very efficient solvers are readily available.

Finally, we remark that all computations are carried out using a Intel Core 2 Duo CPU (E8400) at 3 GHz.

2.5.2 The influence of the penalty parameter

This section studies the influence of the SIPG penalty parameter on the convergence of the CG and the SIPG method. We compare the differences between using a constant penalty parameter, and a diffusion-dependent value. Similar experiments have been considered in [25] for the two-level preconditioner for $p = 1$, a single mesh, and symmetric Gauss-Seidel smoothing (solving the coarse systems using geometric multigrid). They found that "proper weighting", i.e. a diffusion-dependent penalty parameter, "is essential for the performance". In this section, we consider $p = 2, 3$, and both preconditioning and deflation (using block Jacobi smoothing). Furthermore, we analyze the scalability of the methods by considering multiple meshes. Our results are consistent with those in [25].

Table 2.2 displays the number of CG iterations required for convergence for a Poisson problem (i.e. $K = 1$ everywhere) with $\sigma = 20$. Because the diffusion coefficient is constant, a diffusion-dependent value ($\sigma = 20K$) would yield the same results. We observe that both the two-level preconditioner (TL prec.) and the deflation variant (TL defl.) yield fast and scalable convergence (independent of the mesh element diameter). For comparison, the results for standard Jacobi and block Jacobi preconditioning are also displayed (not scalable). Interestingly, the two-level deflation method requires fewer iterations than the preconditioning variant, even though its costs per iteration are about 30% lower (cf. Section 2.4.2).

Table 2.3 considers the same test (using a constant $\sigma = 20$), but now for the problem with five layers (cf. Figure 2.1). It can be seen that the convergence is no longer fast and scalable for this problem with large jumps in the coefficients. The deflation method is significantly faster than the preconditioning variant, but neither produce satisfactory results.

degree	p=2				p=3			
mesh	N=20^2	N=40^2	N=80^2	N=160^2	N=20^2	N=40^2	N=80^2	N=160^2
Jacobi	301	581	1049	1644	325	576	1114	1903
Block Jacobi (BJ)	205	356	676	1190	206	357	696	1183
TL Prec., 2x BJ	36	38	39	40	49	52	53	54
TL Defl., 1x BJ	32	33	33	34	36	37	37	38

TABLE 2.2. *Both two-level methods yield fast scalable convergence for a problem with constant coefficients (Poisson, # CG iterations, $\sigma = 20$).*

degree	p=2				p=3			
mesh	N=20^2	N=40^2	N=80^2	N=160^2	N=20^2	N=40^2	N=80^2	N=160^2
Jacobi	1671	4311	9069	15923	2675	5064	9104	15655
Block Jacobi (BJ)	933	2253	4996	9651	1357	2960	5660	9783
TL Prec., 2x BJ	415	1215	2534	3571	1089	2352	4709	8781
TL Defl., 1x BJ	200	414	531	599	453	591	667	698

TABLE 2.3. *For a problem with extreme contrasts in the coefficients, a constant penalty yields poor convergence (five layers, # CG iterations, $\sigma = 20$).*

Switching to a diffusion-dependent penalty parameter

In Table 2.4, we revisit the experiment in Table 2.3, but this time for a diffusion-dependent penalty parameter ($\sigma = 20K$, using the largest limit value of K at discontinuities). Due to this alternative discretization, the results are now similar to those for the Poisson problem (cf. Table 2.2): both two-level methods yield fast and scalable convergence.

These results motivate the use of a diffusion-dependent penalty parameter, provided that that this strategy does not worsen the accuracy of the SIPG discretization compared to a constant penalty parameter. In Figure 2.2, it is verified that a diffusion-dependent penalty parameter actually improves the accuracy of the SIPG approximation (for $p = 3$). Similar results have been observed for $p = 1, 2$ (not displayed). The higher accuracy can be explained by the fact that the discretization contains more information of the underlying physics for a diffusion-dependent penalty parameter. Altogether, the penalty parameter can best be chosen diffusion-dependent, and we will do so in the remaining of this chapter.

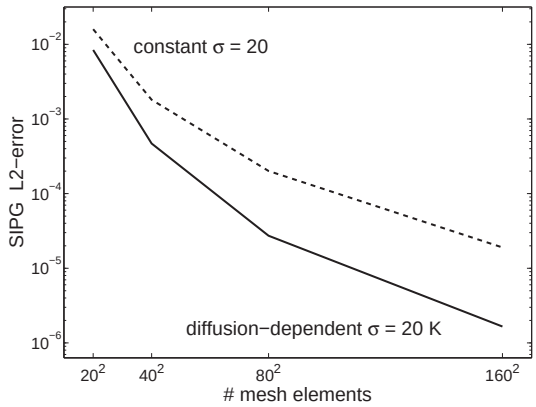


FIGURE 2.2
A diffusion-dependent penalty parameter yields better SIPG accuracy (five layers, $\sigma = 20K$).

degree	p=2				p=3			
mesh	$N=20^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=160^2$
Jacobi	976	1264	1570	2315	1303	1490	1919	3109
Block Jacobi (BJ)	243	424	788	1285	244	425	697	1485
TL Prec., 2x BJ	46	43	43	44	55	56	56	57
TL Defl., 1x BJ	43	45	45	46	47	48	48	48

TABLE 2.4. *For a diffusion-dependent penalty parameter, both two-level methods yield fast scalable convergence for a problem with large permeability contrasts (five layers, # CG iterations, $\sigma = 20K$).*

Weighted averages

The results for the diffusion-dependent penalty parameter in Figure 2.2 and Table 2.4 were established using the largest limit value of K in the definition of σ at the discontinuities. In this section, we consider the influence of using weighted averages, resulting in the SWIP method and a diffusion-dependent penalty parameter based on harmonic means (cf. Section 2.2.3). For this purpose, we study the problem with five layers again.

We have found that using $\sigma = 20K$ with this approach results in negative eigenvalues, implying that the scheme is not coercive, and resulting in poor CG convergence. The same is true for $\sigma = \alpha K$ with $\alpha = 100, 200, 500, 1000$. For $\alpha = 20\,000$, the matrix is positive-definite (tested for $N = 10, 20$ and $p = 1, 2, 3$). Similar outcomes were found using the SIPG scheme rather than the SWIP scheme (for the same 'harmonic' penalty parameter).

At the mesh element edges where the diffusion coefficient K is discontinuous, using $\alpha = 20\,000$ and a harmonic penalty yields $\sigma = \frac{20}{1.001}$. At the same time, using $\alpha = 20$ and a 'maximum' penalty (i.e. using the largest limit value at discontinuities), yields $\sigma = 20$. These values are nearly the same. However, at all other edges, where K is continuous, σ is 1000 times larger for the harmonic penalty (with $\alpha = 20\,000$) than for the maximum penalty (with $\alpha = 20$). Because the penalty parameter should be chosen as small as possible (cf. Section 2.2.3), we conclude that it can best be based on the largest limit value at discontinuities. This is in line with our earlier speculation that using the maximum is a 'safe' choice.

We have also combined the 'maximum' penalty with the SWIP method and compared the outcomes to the earlier results for the SIPG method (both for $\sigma = 20K$). We have found that the discretization accuracy and the CG convergence are practically the same: the relative absolute difference in the discretization error is less then 2% (for $p = 1, 2, 3$ and $N = 20^2, 40^2, 80^2, 160^2$). Comparing Table 2.5 (SWIP) to Table 2.4 (SIPG), it can be seen that the number of CG iterations required for convergence is nearly identical.

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

degree	p=2				p=3			
mesh	N=20^2	N=40^2	N=80^2	N=160^2	N=20^2	N=40^2	N=80^2	N=160^2
Jacobi	980	1270	1575	2325	1309	1500	1935	3129
Block Jacobi (BJ)	244	424	790	1287	244	424	697	1485
TL Prec., 2x BJ	46	43	43	44	55	56	56	57
TL Defl., 1x BJ	43	45	45	46	47	48	49	49

TABLE 2.5. *The difference between the SWIP method (this table) and the SIPG method (cf. Table 2.4) is small (five layers, # CG iterations).*

2.5.3 Coarse systems

To solve the coarse systems with coefficient matrix A_0, a direct solver is usually not feasible in practice. This is because A_0 has the same structure and size ($N \times N$) as a central difference matrix. To improve on the efficiency of the coarse solver, we have investigated the cheaper alternative of applying the CG method again in an inner loop (cf. Section 2.4.3). This section discusses the results using the algebraic multigrid preconditioner *MI_20* in the HSL software package[2], which is based on a classical scheme described in [74]. The inner loop uses a stopping criterion of the form (2.9).

Table 2.6 displays the number of outer CG iterations required for convergence using the two-level preconditioner and deflation variant respectively (for the problem with five layers). Different values of the inner tolerance TOL are considered in these tables. For comparison, the results for the direct solver are also displayed. We observe that low accuracy in the inner loop is sufficient to reproduce the latter. For both two-level methods, the inner tolerance can be 10^4 times as large as the outer tolerance. For the highest acceptable inner tolerance TOL $= 10^{-2}$, the number of iterations in the inner loop is between 2 and 5 in all cases (not displayed in the tables).

In terms of computational time, the difference between the direct solver and the inexact AMG-CG solver is negligible for the problems under consideration. However, for large three-dimensional problems, it can be expected that the inexact coarse solver is much faster, and thus crucial for the overall efficiency and scalability of the linear solver.

Preconditioning:

degree	p=2				p=3			
mesh	N=40^2	N=80^2	N=160^2	N=320^2	N=40^2	N=80^2	N=160^2	N=320^2
direct	43	43	44	44	56	56	57	58
TOL = 10^{-4}	43	43	44	44	56	56	57	58
TOL = 10^{-3}	43	43	44	44	56	56	57	58
TOL = 10^{-2}	43	43	44	44	56	57	58	58
TOL = 10^{-1}	48	62	55	57	59	65	70	78

Deflation:

degree	p=2				p=3			
mesh	N=40^2	N=80^2	N=160^2	N=320^2	N=40^2	N=80^2	N=160^2	N=320^2
direct	45	45	46	46	48	48	48	49
TOL = 10^{-4}	45	45	46	46	48	48	48	49
TOL = 10^{-3}	45	45	46	46	48	48	48	49
TOL = 10^{-2}	45	45	46	46	48	48	48	49
TOL = 10^{-1}	60	81	51	300	48	54	79	67

TABLE 2.6. *Coarse systems can be solved efficiently by using an inexact solver with a relatively low accuracy (TOL) in the inner loop (five layers, # outer CG iterations).*

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

2.5.4 Smoothers and damping

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

Table 2.7 displays the number of CG iterations required for convergence for the problem with five layers. For the deflation variant (Defl.), we have found that damping makes no difference for the CG convergence, so the outcomes for $\omega < 1$ are not displayed. Such a result has also been observed theoretically in [75] for an alternative deflation variant (known as DEF(1)).

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

We have also performed the experiment for standard point Jacobi and point Gauss-Seidel (not displayed in the table). However, this did not lead to satisfactory results: over 250 iterations in all cases, even when the relaxation parameter was sufficiently low to ensure that (2.6) is satisfied (only restricting for point Jacobi). Altogether, the block (Jacobi/Gauss-Seidel) smoothers yield significantly better results than the point (Jacobi/Gauss-Seidel) smoothers.

We speculate that this is due to the following: the coarse correction operator Q simplifies the matrix A to A_0, which eliminates the 'higher-order' information in each element (regarding the higher-order basis functions), but preserves the 'mesh' information (i.e. which elements are neighbors and which are not). Intuitively, a suitable smoother would reintroduce this higher-order information, originally contained in dense blocks of size $m \times m$. The block (Jacobi/Gauss-Seidel) smoothers are better suited for this task, which could explain why they are more effective.

degree	p=2				p=3			
mesh	N=40^2	N=80^2	N=160^2	N=320^2	N=40^2	N=80^2	N=160^2	N=320^2
Prec., 2x BJ ($\omega = 1$)	43	43	44	44	56	56	57	58
($\omega = 0.9$)	34	34	37	37	40	40	42	43
($\omega = 0.8$)	33	34	34	35	36	37	39	39
($\omega = 0.7$)	33	33	33	34	35	36	36	37
($\omega = 0.6$)	32	33	34	35	35	36	36	37
($\omega = 0.5$)	34	34	35	35	36	37	38	39
Prec., 2x BGS ($\omega = 1$)	33	33	34	35	34	35	35	37
($\omega = 0.9$)	33	33	34	34	34	34	35	36
($\omega = 0.8$)	33	33	35	35	35	34	36	37
($\omega = 0.7$)	32	34	36	36	35	36	37	38
($\omega = 0.6$)	33	35	36	37	36	37	38	39
($\omega = 0.5$)	34	35	37	38	37	39	39	40
Defl., 1x BJ ($\omega = 1$)	45	45	46	46	48	48	48	49

TABLE 2.7. *Damping can improve the convergence for the preconditioner, but has no influence for deflation (five layers, # CG Iterations).*

Taking the computational time into account

Based on the results in Table 2.7, it appears that that the preconditioning variant with either block Jacobi (with optimal damping) or block Gauss-Seidel is the most efficient choice. However, the costs per iteration also need to be taken into account.

Table 2.8 reconsiders the results in Table 2.7 but now in terms of the computational time in seconds (using a direct coarse solver). It can be seen that block Gauss-Seidel smoothing is relatively expensive. The deflation variant (with block Jacobi smoothing), is the fastest in nearly all cases. This is due to the fact that it requires only one smoothing step per iteration, instead of two. When an optimal damping parameter is known, the preconditioning variant reaches a comparable efficiency. This is also illustrated in Figure 2.3. However, it is an open question how the damping parameter can best be selected in practice.

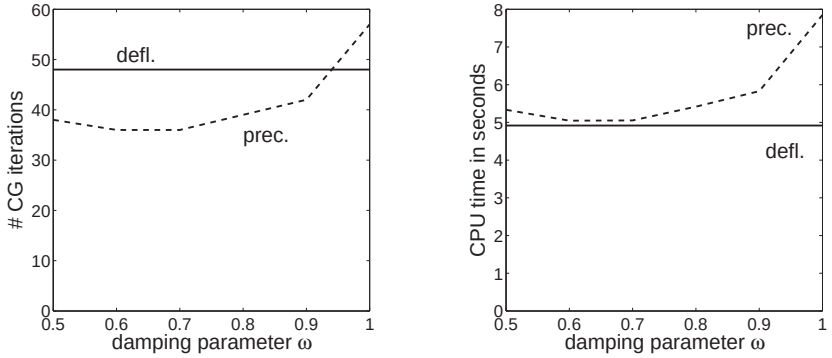


FIGURE 2.3. *Unless an optimal damping parameter is known, deflation is faster due to lower costs per iteration (five layers, $N = 160^2$, block Jacobi).*

degree	p=2				p=3			
mesh	N=40^2	N=80^2	N=160^2	N=320^2	N=40^2	N=80^2	N=160^2	N=320^2
Prec., 2x BJ ($\omega = 1$)	0.09	0.61	3.23	15.43	0.38	1.73	7.85	37.41
($\omega = 0.9$)	0.07	0.48	2.76	13.12	0.27	1.25	5.83	27.91
($\omega = 0.8$)	0.07	0.48	2.53	12.43	0.24	1.16	5.42	25.12
($\omega = 0.7$)	0.07	0.47	2.46	12.11	0.24	1.12	5.05	24.03
($\omega = 0.6$)	0.07	0.47	2.55	12.41	0.24	1.13	5.04	24.11
($\omega = 0.5$)	0.07	0.48	2.61	12.47	0.24	1.16	5.33	25.48
Prec., 2x BGS ($\omega = 1$)	0.15	0.80	3.84	17.50	0.41	1.84	7.75	36.32
($\omega = 0.9$)	0.15	0.80	3.85	17.03	0.41	1.79	7.71	35.36
($\omega = 0.8$)	0.15	0.80	3.96	17.64	0.42	1.79	7.97	36.36
($\omega = 0.7$)	0.14	0.82	4.07	18.10	0.42	1.89	8.22	37.28
($\omega = 0.6$)	0.15	0.84	4.08	18.46	0.43	1.94	8.44	38.23
($\omega = 0.5$)	0.15	0.84	4.19	19.10	0.45	2.06	8.66	39.21
Defl., 1x BJ ($\omega = 1$)	0.07	0.50	2.77	13.61	0.22	1.05	4.92	23.31

TABLE 2.8. *The deflation method and the block Jacobi smoother tend to be faster due to lower costs per iteration (five layers, CPU time in seconds).*

2.5.5 Other test cases

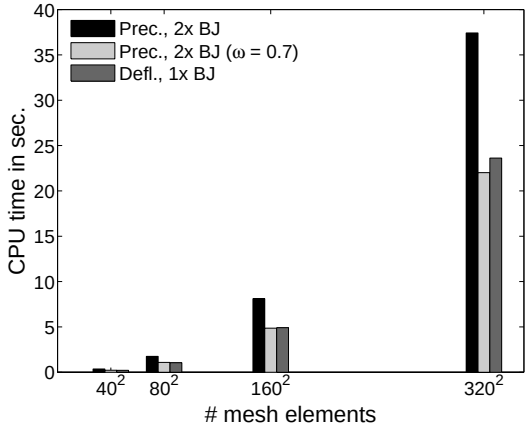
In this section, we repeat the experiments in Table 2.8 for more challenging test cases. For the preconditioning variant, we only display the results for block Jacobi smoothing without damping ($\omega = 1$) and with optimal damping ($\omega = 0.7$).

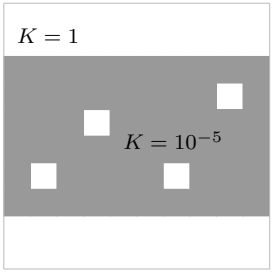
Table 2.9 and Table 2.10 consider problems that mimic the occurrence of sand inclusions within a layer of shale and groundwater flow respectively. Similar problems have been studied in [81]. Table 2.11 displays the results for a bubbly flow problem, inspired by [75]. Table 2.12 studies a bowl-shaped problem: at the black lines in the illustration, homogeneous Neumann boundary conditions are applied. These have a negative impact on the matrix properties, resulting in a challenging problem. Table 2.13 considers an anisotropic problem with two layers (with exact solution $u(x, y) = cos(2\pi y)$). Because the diffusion is a full tensor, this test case mimics the effect of using a non-Cartesian mesh.

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

2.6 Conclusion

This chapter compares the two-level preconditioner proposed in [24] to an alternative (ADEF2) deflation variant for linear systems resulting from SIPG discretizations for diffusion problems. We have found that both two-level methods yield fast and scalable convergence for diffusion-problems with large jumps in the coefficients. This result is obtained provided that the SIPG penalty parameter is chosen dependent on local values of the permeability (using the largest limit value at discontinuities). The latter also benefits the accuracy of the SIPG discretization. Furthermore, the impact of using weighted averages (SWIP) is then small. Coarse systems can be solved efficiently by applying the CG method again in an inner loop with low accuracy. The main difference between both methods is that the deflation method can be implemented by skipping one of the two smoothing steps in the algorithm for the preconditioning variant. This may be particularly advantageous for expensive smoothers, although the basic block Jacobi smoother was found to be highly effective for the problems under consideration. Without damping, deflation can be up to 35% faster than the original preconditioner. If an optimal damping parameter is used, both two-level strategies yield similar efficiency (deflation appears unaffected by damping). However, it remains an open question how the damping parameter can best be selected in practice. Altogether, both two-level strategies can contribute to faster linear solvers for SIPG systems with strong contrasts in the coefficients, such as those encountered in oil reservoir simulations.





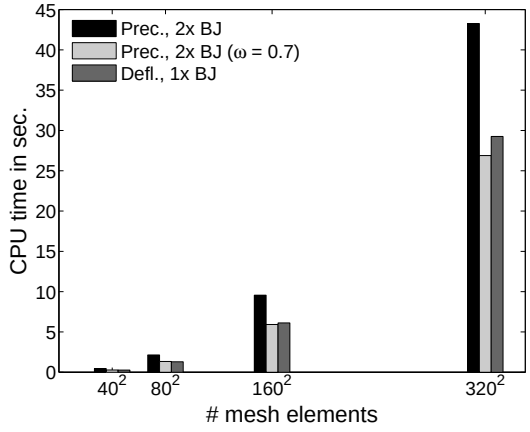
CG Iterations:

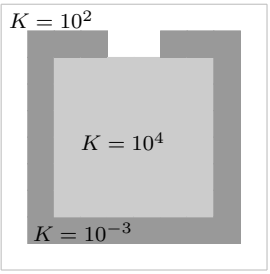
degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	44	48	46	46	53	56	58	59
Prec., 2x BJ ($\omega = 0.7$)	30	30	30	31	32	34	34	34
Defl., 1x BJ	42	42	42	43	46	47	48	48

CPU time in seconds:

degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	0.09	0.68	3.28	16.77	0.36	1.75	8.11	37.42
Prec., 2x BJ ($\omega = 0.7$)	0.06	0.43	2.18	11.38	0.22	1.09	4.86	22.01
Defl., 1x BJ	0.06	0.47	2.44	13.14	0.21	1.05	4.91	23.62

TABLE 2.9. *Sand inclusions*





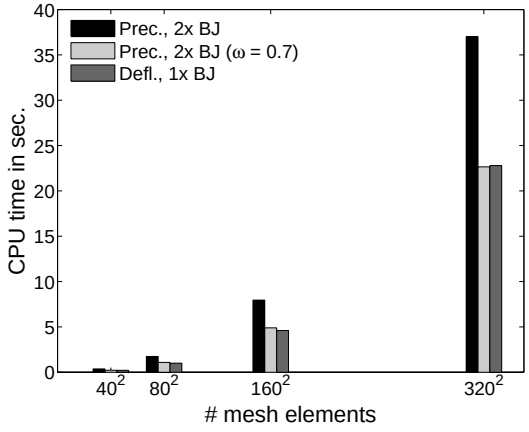
CG Iterations:

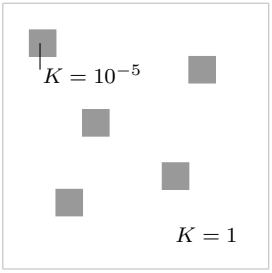
degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	54	52	52	52	67	68	68	69
Prec., 2x BJ ($\omega = 0.7$)	38	38	38	40	41	42	42	42
Defl., 1x BJ	54	54	54	55	59	59	60	60

CPU time in seconds:

degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	0.10	0.74	3.70	19.04	0.45	2.15	9.56	43.27
Prec., 2x BJ ($\omega = 0.7$)	0.07	0.54	2.69	14.74	0.28	1.34	5.93	26.88
Defl., 1x BJ	0.07	0.59	3.12	16.77	0.27	1.30	6.12	29.25

TABLE 2.10. *Ground water*





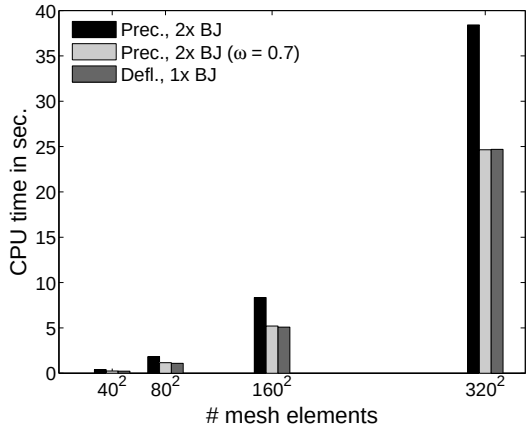
CG Iterations:

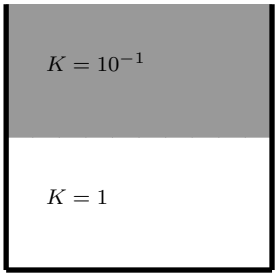
degree	p=2				p=3			
mesh	N=40^2	N=80^2	N=160^2	N=320^2	N=40^2	N=80^2	N=160^2	N=320^2
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	41	42	43	44	55	56	57	58
Prec., 2x BJ ($\omega = 0.7$)	31	31	32	32	33	34	35	35
Defl., 1x BJ	41	39	40	41	45	45	45	46

CPU time in seconds:

degree	p=2				p=3			
mesh	N=40^2	N=80^2	N=160^2	N=320^2	N=40^2	N=80^2	N=160^2	N=320^2
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	0.08	0.59	3.07	16.02	0.37	1.74	7.95	37.00
Prec., 2x BJ ($\omega = 0.7$)	0.06	0.45	2.32	11.77	0.22	1.09	4.89	22.66
Defl., 1x BJ	0.06	0.44	2.35	12.63	0.21	1.00	4.60	22.79

TABLE 2.11. *Bubbly flow*





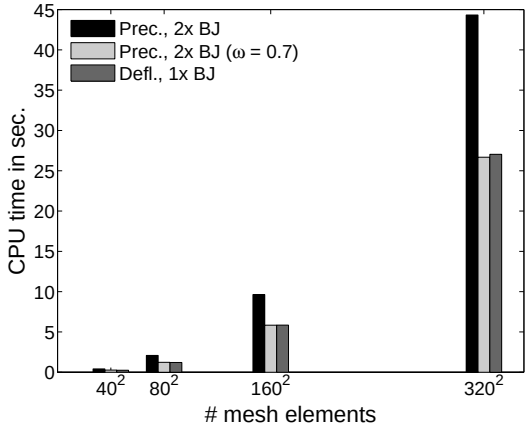
CG Iterations:

degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	45	45	45	45	59	59	60	60
Prec., 2x BJ ($\omega = 0.7$)	34	35	36	36	36	37	37	38
Defl., 1x BJ	47	47	47	47	49	49	50	50

CPU time in seconds:

degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	0.09	0.63	3.26	16.53	0.40	1.83	8.35	38.42
Prec., 2x BJ ($\omega = 0.7$)	0.07	0.49	2.62	13.31	0.24	1.16	5.20	24.65
Defl., 1x BJ	0.07	0.52	2.75	14.41	0.23	1.09	5.09	24.69

TABLE 2.12. *Neumann BCs*



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

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

CG Iterations:

degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	47	48	49	50	61	67	68	71
Prec., 2x BJ ($\omega = 0.7$)	36	37	38	38	39	39	41	42
Defl., 1x BJ	48	49	51	52	54	55	57	57

CPU time in seconds:

degree	p=2				p=3			
mesh	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=320^2$
degrees of freedom	9600	38400	153600	614400	16000	64000	256000	1024000
Prec., 2x BJ	0.09	0.68	3.57	17.65	0.41	2.08	9.63	44.32
Prec., 2x BJ ($\omega = 0.7$)	0.07	0.52	2.79	13.62	0.27	1.23	5.83	26.68
Defl., 1x BJ	0.07	0.54	2.99	15.47	0.25	1.21	5.85	27.04

TABLE 2.13. *Anisotropy*

3

Theoretical scalability

This chapter is based on:

P. van Slingerland, C. Vuik, *Scalable two-level preconditioning and deflation based on a piecewise constant subspace for (SIP)DG systems.* Submitted to JCAM.

3.1 Introduction

Chapter 2 reformulated the two-level preconditioner proposed in [24] as a deflation method, and demonstrated numerically that both two-level variants can yield fast and scalable CG convergence using (damped) block Jacobi smoothing. In this chapter, we focus on theoretical support for these findings.

For symmetric problems, convergence theory for a large class of two-level methods has been established by Falgout et al. [31]. They have derived spectral bounds for the error iteration matrix corresponding to such two-level methods. These results are abstract in the sense that they apply for a large family of coarse spaces and smoothers, and for any symmetric and positive-definite coefficient matrix A. The extension to the nonsymmetric case has been presented by Notay [57].

By applying the work in [31] SIPG matrices, Dobrev et al [24] have shown theoretically that their two-level preconditioner (with coarse corrections based on the DG discretization with polynomial degree $p = 0$) yields scalable convergence of the CG method (independent of the mesh element diameter). This result was established for SIPG schemes with polynomial degree $p = 1$.

To extend the work in [24], in this chapter, we derive bounds for the condition number of the preconditioned system for both two-level methods studied in Chapter 2. In particular, we show that these bounds are independent of the mesh element diameter. Unlike before, our results also apply for $p > 1$ (besides $p = 1$). Another difference is that we include BNN/ADEF2 deflation in the analysis, and compare it to the original preconditioning variant. Additionally, we demonstrate that the required restrictions on the smoother are satisfied for (damped) block Jacobi smoothing.

The outline of this chapter is as follows. Section 3.2 specifies both two-level methods for the linear SIPG systems under consideration. Section 3.3 discusses abstract relations for the condition number of the preconditioned/deflated system, valid for any SPD matrix A. Section 3.4 derives an auxiliary property of SIPG matrices that is a consequence of regularity of the mesh. Section 3.5 uses this property to obtain the main scalability result. Finally, Section 3.6 summarizes the main conclusions.

3.2 Methods and assumptions

Basically, this chapter considers the same linear SIPG systems and two-level preconditioning strategies as in Chapter 2, but now for a larger class of meshes and basis functions. This section specifies the slightly alternative formulations and additional assumptions used in the theoretical analysis to come. Section 3.2.1 specifies the diffusion model and discretizes it by means of the SIPG method. Section 3.2.2 discusses the resulting linear systems. Section 3.2.3 considers two two-level preconditioning strategies for solving the resulting linear system by means of the preconditioned CG method.

3.2.1 SIPG discretization for diffusion problems

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

$$\begin{aligned} -\nabla \cdot (K\nabla u) &= f, && \text{in } \Omega, \\ u &= g_D, && \text{on } \partial\Omega_D, \\ K\nabla u \cdot \mathbf{n} &= g_N, && \text{on } \partial\Omega_N. \end{aligned} \tag{3.1}$$

We assume that the diffusion K is a scalar that is bounded below and above by positive constants, and that the other model parameters are chosen such that a weak solution of (3.1) exists[1]. Additionally, we assume and that Ω is either an interval ($d = 1$), polygon ($d = 2$) or polyhedra ($d = 3$).

To discretize the model problem (3.1), we subdivide Ω into mesh elements $E_1, ..., E_N$ with maximum element diameter h.

We assume that each mesh element E_i is affine-equivalent with a certain reference element E_0 that is an interval/polygon/polyhedra (independent of h) with mutually affine-equivalent edges. (3.2)

Note that all meshes consisting entirely of either intervals, triangles, tetrahedrons, parallelograms, or parallelepipeds satisfy this property.

Furthermore, we assume that the mesh is regular in the sense of [18, p. 124]. To specify this property, for all $i = 0, ..., N$, let h_i and ρ_i denote the diameter of E_i, and the diameter of the largest ball contained in E_i respectively. We can now define regularity as[2]:

$$\frac{h_i}{\rho_i} \lesssim 1, \qquad \forall i = 1, ..., N. \tag{3.3}$$

Now that we have specified the mesh, we can construct an SIPG approximation for our model problem (3.1). To this end, define the test space V that contains each function that is a polynomial of degree p or lower within each mesh element, and that may be discontinuous at the element boundaries. The SIPG approximation u_h is now defined as the unique element in this test space that satisfies the relation

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

where B and L are (bi)linear forms that are similar to those defined earlier in Section 2.2.1, but now for a larger class of meshes. The only difference is that the quantity h is replaced by an alternative value that depends on the edge under consideration (and that K is scalar).

[1] That is, $f, g_N \in L^2(\Omega)$ and $g_D \in H^{\frac{1}{2}}(\Omega)$ [63, p. 25, 26].

[2] Throughout this paper, we use the symbol $\lesssim$ in expressions of the form "$F(x) \lesssim G(x)$ for all $x \in X$" to indicate that there exists a constant $C > 0$, independent of the variable x and the maximum mesh element diameter h (or the number of mesh elements), such that $F(x) \leq CG(x)$ for all $x \in X$. The symbol $\gtrsim$ is defined similarly.

To specify this, we use the same notation as before: the vector $\mathbf{n}_i$ denotes the outward normal of mesh element E_i; the set Γ_h is the collection of all interior edges $e = \partial E_i \cap \partial E_j$; the set Γ_D is the collection of all Dirichlet boundary edges $e = \partial E_i \cap \partial\Omega_D$; the set Γ_N is the collection of all Neumann boundary edges $e = \partial E_i \cap \partial\Omega_N$; and $[.]$ and $\{.\}$ denote the usual trace operators for jumps and averages at the mesh element boundaries, as defined in Section 2.2.1.

Additionally, for all edges e, we write h_e to denote the length of the largest mesh element adjacent to e for one-dimensional problems, the length of e for two-dimensional problems, and the square root of the surface area of e for three-dimensional problems. Using this notation, the forms B and L are defined as follows (for one-dimensional problems, the boundary integrals below should be interpreted as function evaluations of the integrand):

$$\begin{aligned}
B_\Omega(u_h, v) &= \sum_{i=1}^{N} \int_{E_i} K \nabla u_h \cdot \nabla v, \\
B_\sigma(u_h, v) &= \sum_{e \in \Gamma_h \cup \Gamma_D} \int_e \frac{\sigma}{h_e} [u_h] \cdot [v], \\
B_r(u_h, v) &= - \sum_{e \in \Gamma_h \cup \Gamma_D} \int_e \Big(\{K \nabla u_h\} \cdot [v] + [u_h] \cdot \{K \nabla v\} \Big), \\
B(u_h, v) &= B_\Omega(u_h, v) + B_\sigma(u_h, v) + B_r(u_h, v), \\
L(v) &= \int_\Omega f v - \sum_{e \in \Gamma_D} \int_e \left([K \nabla v] + \frac{\sigma}{h_e} v \right) g_D + \sum_{e \in \Gamma_N} \int_e v g_N,
\end{aligned} \tag{3.4}$$

where $\sigma \geq 0$ is the penalty parameter (cf. Section 2.2.3). Its value may vary throughout the domain, and we assume that it is bounded below and above by positive constants (independent of the maximum element diameter h). Furthermore, we assume that it is sufficiently large so that the scheme is coercive[3] [63, p. 38–40], i.e.:

$$B_\Omega(v, v) + B_\sigma(v, v) \lesssim B(v, v), \qquad \forall v \in V. \tag{3.5}$$

3.2.2 Linear system

In order to compute the SIPG approximation, we need to choose a basis for the test space V. In Section 2.2.2, we have discussed the monomial basis functions for uniform Cartesian meshes. We now consider a more general class: for all $i = 1, ..., N$, let $F_i : E_i \to E_0$ denote an invertible affine mapping (which exists by assumption (3.2)). Furthermore, let the functions $\phi_k^{(0)} : E_0 \to \mathbb{R}$ (with

[3] For coercivity, it suffices that $\sigma \geq 2Cn_0 \frac{k_1^2}{k_0}$, where k_0 and k_1 are the global lower and upper bounds for the diffusion coefficient K respectively, n_0 is the maximum number of neighbors an element can have (e.g. $n_0 = 4$ for a two-dimensional quadrilateral mesh), and C is a constant occurring in a trace inequality that does not depend on h (but may depend on p). See [63, p. 23, 38-39] for more details.

$k = 1, ..., m$) form a basis for the space of all polynomials of degree p and lower on the reference element (setting $\phi_1^{(0)} = 1$). Using this basis on the reference element, the basis function $\phi_k^{(i)}$ (with $k = 1, ..., m$ and $i = 1, ..., N$) is zero in the entire domain, except in the mesh element E_i, where it reads $\phi_k^{(i)} = \phi_k^{(0)} \circ F_i$.

Now that we have defined the basis functions, we can express u_h as a linear combination of these functions and construct a linear system. Although we are now considering a larger class of meshes and basis functions, we arrive at the same forms we have seen earlier in Section 2.2.2:

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

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

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & & \vdots \\ \vdots & & \ddots & \\ A_{N1} & \dots & & A_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}, \tag{3.6}$$

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

$$\begin{aligned} A_{ji} &= \begin{bmatrix} B(\phi_1^{(i)}, \phi_1^{(j)}) & B(\phi_2^{(i)}, \phi_1^{(j)}) & \dots & B(\phi_m^{(i)}, \phi_1^{(j)}) \\ B(\phi_1^{(i)}, \phi_2^{(j)}) & B(\phi_2^{(i)}, \phi_2^{(j)}) & & \vdots \\ \vdots & & \ddots & \\ B(\phi_1^{(i)}, \phi_m^{(j)}) & \dots & & B(\phi_m^{(i)}, \phi_m^{(j)}) \end{bmatrix} \\ \mathbf{u}_i &= \begin{bmatrix} u_1^{(i)} \\ u_2^{(i)} \\ \vdots \\ u_m^{(i)} \end{bmatrix}, \quad \mathbf{b}_j = \begin{bmatrix} L(\phi_1^{(j)}) \\ L(\phi_2^{(j)}) \\ \vdots \\ L(\phi_m^{(j)}) \end{bmatrix}. \end{aligned} \tag{3.7}$$

Note that A is Symmetric and Positive-Definite (SPD), as the bilinear form B is SPD (cf. (3.4) and (3.5)).

3.2.3 Two-level preconditioning and deflation

To solve the linear SIPG system by means of the preconditioned CG method, we consider the two-level preconditioner and the corresponding deflation variant discussed in Chapter 2. For the deflation method, we consider the alternative (yet equivalent) BNN formulation. We specify both methods below, as well as some additional assumptions on the smoother.

Recall the restriction operator R, which is defined such that $A_0 := RAR^T$ is the SIPG matrix for polynomial degree $p = 0$, and the coarse correction operator $Q := R^T A_0^{-1} R$. The two-level preconditioner (Definition 2.3.1) combines this operator with a nonsingular smoother $M_{\text{prec}}^{-1} \approx A^{-1}$. The result $\mathbf{y} = P_{\text{prec}}^{-1}\mathbf{r}$ of applying the two-level preconditioner to a vector $\mathbf{r}$ can be computed in three steps:

$$\begin{aligned} \mathbf{y}^{(1)} &= M_{\text{prec}}^{-1}\mathbf{r} && \text{(pre-smoothing)}, \\ \mathbf{y}^{(2)} &= \mathbf{y}^{(1)} + Q(\mathbf{r} - A\mathbf{y}^{(1)}) && \text{(coarse correction)}, \\ \mathbf{y} &= \mathbf{y}^{(2)} + M_{\text{prec}}^{-T}(\mathbf{r} - A\mathbf{y}^{(2)}) && \text{(post-smoothing)}. \end{aligned} \tag{3.8}$$

Basically, the BNN deflation variant is obtained by turning (3.8) inside out, and using an SPD smoother $M_{\text{defl}}^{-1} \approx A^{-1}$. The result $\mathbf{y} = P_{\text{defl}}^{-1}\mathbf{r}$ of applying the BNN deflation technique to a vector $\mathbf{r}$ can now be computed as:

$$\begin{aligned} \mathbf{y}^{(1)} &:= Q\mathbf{r} && \text{(pre-coarse correction)}. \\ \mathbf{y}^{(2)} &:= \mathbf{y}^{(1)} + M_{\text{defl}}^{-1}(\mathbf{r} - A\mathbf{y}^{(1)}) && \text{(smoothing)}, \\ \mathbf{y} &:= \mathbf{y}^{(2)} + Q(\mathbf{r} - A\mathbf{y}^{(2)}) && \text{(post-coarse correction)}. \end{aligned} \tag{3.9}$$

Both two-level strategies can be implemented in a standard preconditioned CG algorithm (cf. Section 2.3.3). We stress that the BNN deflation variant can be implemented more efficiently in a CG algorithm by using the so-called ADEF2 deflation variant (cf. Section 2.4). However, for the theoretical purposes in this paper, it is more convenient to study BNN rather than ADEF2. Furthermore, we require some additional assumptions on the smoothers, indicated hereafter.

To specify these, for any SPD matrix M, let

$$\pi_M := R^T(RMR^T)^{-1}RM \tag{3.10}$$

denote the projection onto the coarse space $\text{Range}(R^T)$ that yields the best approximation in the M-norm [31]. Additionally, for any nonsingular matrix M such that $M + M^T - A$ is SPD, define the symmetrization

$$\widetilde{M} := M^T(M + M^T - A)^{-1}M.$$

Using this notation, we can now specify the additional smoother requirements. For the preconditioner, we assume:

$$M_{\text{prec}} + M_{\text{prec}}^T - A \text{ is SPD}. \tag{3.11}$$

$$h^{2-d}\mathbf{v}^T\widetilde{M}_{\text{prec}}\mathbf{v} \lesssim \mathbf{v}^T\mathbf{v}, \qquad \forall \mathbf{v} \in \text{Range}(I - \pi_I). \tag{3.12}$$

For the deflation method, we assume:

$$2M_{\text{defl}} - A \text{ is SPD}, \tag{3.13}$$

$$h^{2-d}\mathbf{v}^T M_{\text{defl}}\mathbf{v} \lesssim \mathbf{v}^T\mathbf{v}, \qquad \forall \mathbf{v} \in \text{Range}(I - \pi_I). \tag{3.14}$$

We have seen in Chapter 2 that the operator P_{prec}^{-1} is SPD assuming that (3.11) is satisfied (for the deflation variant, it suffices that M_{defl} is SPD). The conditions (3.11) and (3.13) imply that "the smoother iteration is a contraction in the A-norm" [31, p. 473]. The main idea behind the conditions (3.12) and (3.14) is that the smoother should scale with h^{2-d} in the same way that A does, and that $\widetilde{M}$ is an efficient preconditioner for A in the space orthogonal to the coarse space Range(R^T) [79, p. 78]. A slightly stronger version of (3.12) is also used in [24] to establish scalable convergence.

It will be shown in Section 3.5.2 that the requirements (3.11)–(3.14) are satisfied for (damped) block Jacobi smoothing (with $m \times m$ blocks). A similar strategy can be used to show (3.11) and (3.12) for (damped) block Gauss-Seidel smoothing (with $m \times m$ blocks)[4].

3.3 Abstract relations for any SPD matrix A

This section discusses abstract relations for the condition number of the preconditioned system for both two-level methods. These results are abstract in the sense that they apply for any SPD matrix A, so they are not restricted to SIPG matrices. Section 3.3.1 discusses the condition number of a preconditioned system for a certain class of operators. Section 3.3.2 considers the specific implications for both two-level methods. Section 3.3.3 compares them under specific assumptions on the smoother.

3.3.1 Using the error iteration matrix

In this section, we consider the condition number of the preconditioned system for arbitrary SPD matrices A and a certain class of SPD preconditioners P^{-1}. Specifically, each preconditioner P^{-1} in this class is such that, for some SPD matrix M, the so-called error iteration matrix $I - P^{-1}A$ has the same eigenvalues as (recall the notation in Section 3.2.3):

$$T_M := (I - \pi_A)(I - M^{-1}A)(I - \pi_A).$$

In Section 3.3.2 hereafter, we will see that both two-level methods are in this class for certain specific choices of M. Defining

$$K_M := \sup_{\mathbf{v}\neq 0} \frac{\|(I - \pi_M)\mathbf{v}\|_M^2}{\|\mathbf{v}\|_A^2}, \tag{3.15}$$

we now have the following result:

[4] To show that (3.11) and (3.12) are valid for block Gauss-Seidel smoothing, note that (3.11) is automatically satisfied as $M_{\text{prec}} + M_{\text{prec}}^T - A$ is the block diagonal of A, which is SPD. To show (3.12), the main idea is to follow a similar strategy as in [79, Proposition 6.12], and then use the result for block Jacobi obtained in Section 3.5.2.

Lemma 3.3.1 *The condition number (in the 2-norm) of the preconditioned system $P^{-1}A$ above can be bounded as follows*[5]*:*

$$\kappa_2(P^{-1}A) \leq \lambda_{\max}(M^{-1}A)K_M. \tag{3.16}$$

Additionally, if[6] $M - A \geq 0$*, then,*

$$\kappa_2(P^{-1}A) = K_M. \tag{3.17}$$

To show this, we use that T_M has real eigenvalues (as $A^{\frac{1}{2}}T_M A^{-\frac{1}{2}}$ is symmetric), and that [57, Theorem 2.1 and Corollary 2.1]:

$$T_M \text{ has } m \text{ times eigenvalue } 0, \tag{3.18}$$

$$\lambda_{\min}(T_M) \geq 1 - \lambda_{\max}(M^{-1}A), \tag{3.19}$$

$$\lambda_{\max}(T_M) = 1 - \frac{1}{\lambda_{\max}(A^{-1}M(I-\pi_M))}. \tag{3.20}$$

Proof (of Lemma 3.3.1) First, note that $P^{-1}A$ and $P^{-\frac{1}{2}}AP^{-\frac{1}{2}}$ have the same positive eigenvalues and singular values. Hence, we may express the condition number as:

$$\begin{aligned}\kappa_2(P^{-1}A) &= \frac{\lambda_{\max}(P^{-1}A)}{\lambda_{\min}(P^{-1}A)} = \frac{1-\lambda_{\min}(T_M)}{1-\lambda_{\max}(T_M)} \\ &\overset{(3.20)}{=} \big(1-\lambda_{\min}(T_M)\big)\lambda_{\max}\big(A^{-1}M(I-\pi_M)\big).\end{aligned} \tag{3.21}$$

Because $I - \pi_M = (I-\pi_M)^2$ is a projection and $M(I-\pi_M)$ is symmetric, it follows that:

$$\begin{aligned}\lambda_{\max}\big(A^{-1}M(I-\pi_M)\big) &= \lambda_{\max}\big(A^{-1}M(I-\pi_M)^2\big) \\ &= \lambda_{\max}\big(A^{-1}(I-\pi_M)^T M(I-\pi_M)\big) \\ &= \lambda_{\max}\big(A^{-\frac{1}{2}}(I-\pi_M)^T M(I-\pi_M)A^{-\frac{1}{2}}\big) \\ &= \sup_{\mathbf{v}\neq 0}\frac{\|(I-\pi_M)\mathbf{v}\|_M^2}{\|\mathbf{v}\|_A^2} \overset{(3.15)}{=} K_M.\end{aligned}$$

Substitution into (3.21) yields:

$$\kappa_2(P^{-1}A) = \big(1-\lambda_{\min}(T_M)\big)K_M. \tag{3.22}$$

Application of (3.19) now completes the proof of (3.16). To show (3.17), assume that $M - A \geq 0$, which implies that $I - A^{\frac{1}{2}}M^{-1}A^{\frac{1}{2}} \geq 0$:

$$\begin{array}{ccc} M & \geq & A \\ M^{-1} & \overset{\text{[40, p. 398, 471]}}{\leq} & A^{-1} \end{array}$$

[5]Throughout this chapter, $\lambda_{\min}$ and $\lambda_{\max}$ denote the smallest and largest eigenvalue of a matrix with real eigenvalues respectively.

[6]Throughout this chapter, for symmetrical matrices $M_1, M_2 \in \mathbb{R}^{n\times n}$, we write $M_1 \leq M_2$ to indicate that $\mathbf{v}^T M_1 \mathbf{v} \leq \mathbf{v}^T M_2 \mathbf{v}$ for all vectors $\mathbf{v} \in \mathbb{R}^n$; the notation $\geq$, $<$, and $>$ is used similarly.

$$\begin{aligned} A^{\frac{1}{2}} M^{-1} A^{\frac{1}{2}} &\leq I \\ I - A^{\frac{1}{2}} M^{-1} A^{\frac{1}{2}} &\geq 0. \end{aligned}$$

Hence, defining the *symmetric* projection $\bar{\pi}_A := A^{\frac{1}{2}} \pi_A A^{-\frac{1}{2}}$, it follows that the eigenvalues of T_M are non-negative:

$$A^{\frac{1}{2}} T_M A^{-\frac{1}{2}} = (I - \bar{\pi}_A)(I - A^{\frac{1}{2}} M^{-1} A^{\frac{1}{2}})(I - \bar{\pi}_A) \geq 0.$$

As a result, (3.18) implies that $\lambda_{\min}(T_M) = 0$. Substitution into (3.22) yields (3.17), which then completes the proof. ■

3.3.2 Implications for the two-level methods

Next, we apply the result in the previous section to analyze the condition number of the preconditioned system for both the two-level preconditioner P_{prec}^{-1} and the corresponding BNN deflation variant P_{defl}^{-1} (as specified in Section 3.2.3). For the two-level preconditioner, it is well-known that

$$\kappa_2(P_{\text{prec}}^{-1} A) \leq K_{\widetilde{M}_{\text{prec}}}. \tag{3.23}$$

This follows as a special case from [31] (also cf. [79, p. 70-73]), and relies on assumption (3.11). Below, we observe that the theory in [57] implies (via Lemma 3.3.1) that (3.23) remains true if we replace the inequality by equality. Furthermore, we obtain similar bounds for the deflation variant. Altogether, we have the following result, which applies for any SPD matrix A:

Lemma 3.3.2 *Suppose that A is SPD and let P_{prec}^{-1} and P_{defl}^{-1} be the two-level operators specified in Section 3.2.3. Then, assuming* (3.11)*, the condition number (in the 2-norm) of the preconditioned system $P_{\text{prec}}^{-1} A$ can be expressed as follows:*

$$\kappa_2(P_{\text{prec}}^{-1} A) = K_{\widetilde{M}_{\text{prec}}}. \tag{3.24}$$

Additionally, assuming (3.13)*, we have for deflation:*

$$\kappa_2(P_{\text{defl}}^{-1} A) \leq \lambda_{\max}(M_{\text{defl}}^{-1} A) K_{M_{\text{defl}}} < 2 K_{M_{\text{defl}}}, \tag{3.25}$$

and, under the stronger assumption $M_{\text{defl}} - A \geq 0$:

$$\kappa_2(P_{\text{defl}}^{-1} A) = K_{M_{\text{defl}}}. \tag{3.26}$$

To show this result, we apply Lemma 3.3.1, using (σ denotes the spectrum):

$$\sigma\big(I - P_{\text{prec}}^{-1} A\big) = \sigma\big(T_{\widetilde{M}_{\text{prec}}}\big), \tag{3.27}$$

$$\sigma\big(I - P_{\text{defl}}^{-1} A\big) = \sigma\big(T_{M_{\text{defl}}}\big). \tag{3.28}$$

These relations follow similar to [75, p. 1730]. Finally, we use that, for any nonsingular M [79, Proposition 3.8]:

$$M + M^T - A > 0 \qquad \Rightarrow \qquad \widetilde{M} - A \geq 0. \tag{3.29}$$

PROOF (OF LEMMA 3.3.2) Combining (3.11) and (3.29) gives $\widetilde{M}_{\text{prec}} - A \geq 0$. Using this with (3.27) in Lemma 3.3.1 yields (3.24). Similarly, (3.26) follows from Lemma 3.3.1 using (3.28) and the assumption $M_{\text{defl}} - A \geq 0$. To show (3.25), note that the first inequality results from Lemma 3.3.1 and (3.28), while the second inequality follows from observing that (3.13) implies that $\lambda_{\max}(M_{\text{defl}}^{-1}A) < 2$. This completes the proof. ■

3.3.3 Comparing deflation and preconditioning

In this section, we compare both two-level methods in terms of the corresponding condition numbers. In [75, Theorem 6.1], it has been shown that the eigenvalues of $P_{\text{prec}}^{-1}A$ and $P_{\text{defl}}^{-1}A$ are equal if $M_{\text{defl}} = \widetilde{M}_{\text{prec}}$. Below, we compare both two-level methods in case they both use the same smoother.

Theorem 3.3.3 *Suppose that A is SPD and let P_{prec}^{-1} and P_{defl}^{-1} be the two-level operators specified in Section 3.2.3. Furthermore, choose $M_{\text{prec}} = M_{\text{defl}} =: M$ SPD with $M - A \geq 0$. Then, both methods are related in the following sense:*

$$\frac{1}{2}\kappa_2(P_{\text{defl}}^{-1}A) \leq \kappa_2(P_{\text{prec}}^{-1}A) \leq \kappa_2(P_{\text{defl}}^{-1}A). \tag{3.30}$$

Before showing this result, we note that it implies that the CG convergence for the preconditioner is asymptotically not worse than for the deflation variant. That is, assuming $M - A \geq 0$. In general, we may have $2M - A > 0$ rather than the stronger assumption $M - A \geq 0$. This is the case for block Jacobi smoothing in the numerical experiments in Section 2.5, where we have seen that deflation can yield fewer iterations (at lower costs per iteration). Nevertheless, if the smoother M is replaced by the damped alternative $\omega^{-1}M$ such that $\omega^{-1}M - A \geq 0$ (note that any $\omega \leq \frac{1}{2}$ suffices if $2M - A > 0$), then the result above applies (although a larger ω might give better results). Altogether, Theorem 3.3.3 provides insight in the way both two-level methods are related, but does not imply that preconditioning is always better.

To show Theorem 3.3.3, we use, for any SPD matrices M, N:

$$K_M \leq \left\| N^{-\frac{1}{2}} M (I - \pi_M) N^{-\frac{1}{2}} \right\| K_N. \tag{3.31}$$

This follows from the more general work in [Not10, Corollary 2.2]. Additionally, we use, for any SPD matrix M [Not05, eq. (45)]:

$$\frac{1}{2}M \leq \widetilde{M} \leq \frac{1}{2 - \lambda_{\max}(M^{-1}A)} M. \tag{3.32}$$

Theorem 3.3.3 can now be shown as follows:

PROOF (OF THEOREM 3.3.3) First, we show that (3.31) implies that, for any SPD matrices M, N and scalar $\alpha > 0$:

$$M \leq \alpha N \qquad \Rightarrow \qquad K_M \leq \alpha K_N. \tag{3.33}$$

To show this, define $\bar{\pi}_M = M^{\frac{1}{2}} \pi_M M^{-\frac{1}{2}}$ and observe that $I - \bar{\pi}_M$ is a *symmetric* projection. Hence:

$$\begin{aligned}
& \left\| N^{-\frac{1}{2}} M (I - \pi_M) N^{-\frac{1}{2}} \right\|_2^2 \\
= \quad & \left\| N^{-\frac{1}{2}} M^{\frac{1}{2}} M^{\frac{1}{2}} (I - \pi_M) M^{-\frac{1}{2}} M^{\frac{1}{2}} N^{-\frac{1}{2}} \right\|_2^2 \\
= \quad & \left\| N^{-\frac{1}{2}} M^{\frac{1}{2}} (I - \bar{\pi}_M) M^{\frac{1}{2}} N^{-\frac{1}{2}} \right\|_2^2 \\
= \quad & \sup_{\mathbf{v} \neq 0} \frac{\left\| (I - \bar{\pi}_M) M^{\frac{1}{2}} N^{-\frac{1}{2}} \mathbf{v} \right\|^2_{M^{\frac{1}{2}} N^{-1} M^{\frac{1}{2}}}}{\|\mathbf{v}\|_2^2} \\
\overset{M \leq \alpha N \Rightarrow M^{\frac{1}{2}} N^{-1} M^{\frac{1}{2}} \leq \alpha}{\leq} \quad & \alpha \sup_{\mathbf{v} \neq 0} \frac{\left\| (I - \bar{\pi}_M) M^{\frac{1}{2}} N^{-\frac{1}{2}} \mathbf{v} \right\|_2^2}{\|\mathbf{v}\|_2^2} \\
\overset{(I - \bar{\pi}_M) \text{ symmetric projection}}{\leq} \quad & \alpha \sup_{\mathbf{v} \neq 0} \frac{\left\| M^{\frac{1}{2}} N^{-\frac{1}{2}} \mathbf{v} \right\|_2^2}{\|\mathbf{v}\|_2^2} \\
\leq \quad & \alpha \sup_{\mathbf{v} \neq 0} \frac{\|\mathbf{v}\|^2_{N^{-\frac{1}{2}} M N^{-\frac{1}{2}}}}{\|\mathbf{v}\|_2^2} \\
\overset{M \leq \alpha N \Rightarrow N^{-\frac{1}{2}} M N^{-\frac{1}{2}} \leq \alpha}{\leq} \quad & \alpha^2 \sup_{\mathbf{v} \neq 0} \frac{\|\mathbf{v}\|_2^2}{\|\mathbf{v}\|_2^2} \\
= \quad & \alpha^2.
\end{aligned}$$

This completes the proof of (3.33). Combining (3.33) and (3.32) now gives:

$$\frac{1}{2} K_M \leq K_{\widetilde{M}} \leq \frac{1}{2 - \lambda_{\max}(M^{-1}A)} K_M.$$

Application of Lemma 3.3.2 (using $M_{\text{prec}} = M_{\text{defl}} =: M$ SPD with $M - A \geq 0$) yields $\kappa_2(P_{\text{defl}}^{-1} A) = K_{\widetilde{M}}$ and $\kappa_2(P_{\text{defl}}^{-1} A) = K_M$. Hence:

$$\frac{1}{2} \kappa_2(P_{\text{defl}}^{-1} A) \leq \kappa_2(P_{\text{prec}}^{-1} A) \leq \frac{1}{2 - \lambda_{\max}(M^{-1}A)} \kappa_2(P_{\text{defl}}^{-1} A).$$

Observing that $\lambda_{\max}(M^{-1}A) \leq 1$ (as $M - A$ SPD) yields (3.30) which then completes the proof. ■

3.4 Intermezzo: regularity on the block diagonal of A

To further refine the abstract relations in the previous section for our SIPG application, we need to derive an auxiliary result: the diagonal blocks of a SIPG matrix A all 'behave' in a similar manner in the space orthogonal to the coarse

space. This section obtains this result in three steps: Section 3.4.1 discusses an auxiliary property based on the regularity of the mesh. Section 3.4.2 uses this property to establish the desired result in terms of 'small bilinear forms'. Section 3.4.3 can then show the main result of this section: regularity on the block diagonal of A.

3.4.1 Using regularity of the mesh

The first step is rather abstract consequence of the regularity of the mesh. To state this result, recall the mapping $F_i : E_i \to E_0$ (cf. Section 3.2.2). Because this mapping is invertible and affine by assumption (3.2), there exists an invertible matrix $G_i \in \mathbb{R}^{d\times d}$ and a vector $\mathbf{g}_i \in \mathbb{R}^d$ such that

$$F_i(\mathbf{x}) = G_i\mathbf{x} + \mathbf{g}_i, \qquad \forall \mathbf{x} \in E_i.$$

Next, let $|G_i^{-1}|$ denote the determinant of G_i^{-1}, and define

$$Z_i := |G_i^{-1}| G_i^T G_i.$$

Using regularity of the mesh, the following spectral properties of Z_i can be shown:

Lemma 3.4.1 *Assuming* (3.2) *and* (3.3)*, the eigenvalues of the matrices Z_i above satisfy the following relation:*

$$1 \lesssim \lambda_{\min}(h^{2-d}Z_i) \leq \lambda_{\max}(h^{2-d}Z_i) \lesssim 1, \qquad \forall i = 1, ..., N. \tag{3.34}$$

To show this result, we use the following relations [18, p. 120–122][7]:

$$\left|G_i^{-1}\right| = \frac{\text{meas}(E_i)}{\text{meas}(E_0)}, \qquad \|G_i\|_2 \leq \frac{h_0}{\rho_i}, \qquad \left\|G_i^{-1}\right\|_2 \leq \frac{h_i}{\rho_0}. \tag{3.35}$$

We can now prove Lemma 3.4.1:

Proof (of Lemma 3.4.1) Because $Z_i := |G_i^{-1}| G_i^T G_i$, and G_i is invertible, we have (cf. [67, p. 26]):

$$\lambda_{\max}(h^{2-d}Z_i) = h^{2-d}\left|G_i^{-1}\right| \lambda_{\max}(G_i^T G_i) = h^{2-d}\left|G_i^{-1}\right| \|G_i\|_2^2,$$
$$\lambda_{\min}(h^{2-d}Z_i) = h^{2-d}\left|G_i^{-1}\right| \frac{1}{\lambda_{\max}\left((G_i^T G_i)^{-1}\right)} = h^{2-d}\left|G_i^{-1}\right| \frac{1}{\left\|G_i^{-1}\right\|_2^2}.$$

Applying the relations in (3.35), using that meas(E_0), h_0 and ρ_0 do not depend on h, and observing that $\rho_i^d \lesssim \text{meas}(E_i) \lesssim h_i^d$ (for all $i = 1, ..., N$), we may write:

$$\lambda_{\max}(h^{2-d}Z_i) \leq h^{2-d}\frac{\text{meas}(E_i)}{\text{meas}(E_0)}\left(\frac{h_0}{\rho_i}\right)^2 \lesssim \frac{\text{meas}(E_i)}{h^d}\left(\frac{h}{\rho_i}\right)^2 \lesssim \left(\frac{h_i}{h}\right)^d\left(\frac{h}{\rho_i}\right)^2,$$

[7] Here, meas(.) denotes the Lesbesque measure.

$$\lambda_{\min}(h^{2-d}Z_i) \geq h^{2-d}\frac{\text{meas}(E_i)}{\text{meas}(E_0)}\left(\frac{\rho_0}{h_i}\right)^2 \gtrsim \frac{\text{meas}(E_i)}{h^d}\left(\frac{h}{h_i}\right)^2 \gtrsim \left(\frac{\rho_i}{h}\right)^d\left(\frac{h}{h_i}\right)^2.$$

Hence, the proof is completed if we can show that

$$1 \leq \frac{h}{h_i} \leq \frac{h}{\rho_i} \lesssim 1, \qquad \forall i = 1, ..., N.$$

The first two inequalities follow from the fact that $\rho_i \leq h_i \leq h$. The last inequality follows as a special case from (3.3). Hence, we obtain (3.34), which completes the proof. ■

3.4.2 The desired result in terms of 'small' bilinear forms

The second step is to use the regularity result in the previous section to obtain the desired result (regularity on the block diagonal of A) in terms of 'small' bilinear forms. To state this result, we require the following notation: let V_0 denote the space of all polynomials of degree p and lower defined on the reference element E_0. Additionally, let Γ_i denote the set of all edges of E_i that are either in the interior or at the Dirichlet boundary. Furthermore, let Γ_0 denote the set of all edges of the reference element E_0. Next, define the following bilinear forms[8]:

$$B_\Omega^{(i)}(v,w) = \int_{E_i} K\nabla(v\circ F_i)\cdot\nabla(w\circ F_i), \qquad B_\Omega^{(0)}(v,w) = \int_{E_0}\nabla v\cdot\nabla w,$$
$$B_\sigma^{(i)}(v,w) = \sum_{e\in\Gamma_i}\int_e \frac{\sigma}{h_e}[v\circ F_i]\cdot[w\circ F_i], \qquad B_\sigma^{(0)}(v,w) = \sum_{e\in\Gamma_0}\int_e [v]\cdot[w],$$

for all $v, w \in V_0$ and $i = 1, ..., N$. Using this notation, we now have the following result:

Lemma 3.4.2 *Suppose that the diffusion coefficient K and the penalty parameter σ are bounded above and below by positive constants (independent of h). Assume that* (3.2) *and* (3.3) *hold. Then, the bilinear forms above satisfy the following relations:*

$$B_\Omega^{(0)}(w,w) \lesssim h^{2-d}B_\Omega^{(i)}(w,w) \lesssim B_\Omega^{(0)}(w,w), \quad \forall w\in V_0, \quad \forall i = 1, ..., N. \tag{3.36}$$

$$0 \leq h^{2-d}B_\sigma^{(i)}(w,w) \lesssim B_\sigma^{(0)}(w,w), \quad \forall w\in V_0, \quad \forall i = 1, ..., N. \tag{3.37}$$

We discuss the proof of both relations individually hereafter.

[8]Here, the trace operators are defined as before by extending the function to be zero outside E_0 and E_i.

Proof (of (3.36)) Because the diffusion coefficient K is bounded below and above by positive constants (independent of h), we may write (all displayed relations below are for all $w \in V_0$ and for all $i = 1, ..., N$):

$$\int_{E_i} \nabla (w \circ F_i) \cdot \nabla (w \circ F_i) \lesssim B_\Omega^{(i)}(w, w) \lesssim \int_{E_i} \nabla (w \circ F_i) \cdot \nabla (w \circ F_i) . \tag{3.38}$$

Next, we apply the chain rule, using that the Jacobian of F_i is equal to G_i:

$$\int_{E_i} \nabla (w \circ F_i) \cdot \nabla (w \circ F_i) = \int_{E_i} \Big(\big(G_i \nabla w\big) \circ F_i \Big) \cdot \Big(\big(G_i \nabla w\big) \circ F_i \Big).$$

A change of variables (from $\mathbf{x} \in E_i$ to $F_i(\mathbf{x}) \in E_0$) introduces a factor $\left|G_i^{-1}\right|$:

$$\begin{aligned} \int_{E_i} \nabla (w \circ F_i) \cdot \nabla (w \circ F_i) &= \int_{E_0} \left|G_i^{-1}\right| \big(G_i \nabla w\big) \cdot \big(G_i \nabla w\big) \\ &= \int_{E_0} (\nabla w)^T \underbrace{\left|G_i^{-1}\right| G_i^T G_i}_{=Z_i} \nabla w. \end{aligned}$$

Substitution of this expression into (3.38) and multiplication with h^{2-d} yields:

$$\int_{E_0} (\nabla w)^T (h^{2-d} Z_i) (\nabla w) \lesssim h^{2-d} B_\Omega^{(i)}(w, w) \lesssim \int_{E_0} (\nabla w)^T (h^{2-d} Z_i) (\nabla w).$$

Application of Lemma 3.4.1 gives:

$$\underbrace{\int_{E_0} \nabla w \cdot \nabla w}_{=B_\Omega^{(0)}(w,w)} \lesssim h^{2-d} B_\Omega^{(i)}(w, w) \lesssim \underbrace{\int_{E_0} \nabla w \cdot \nabla w,}_{=B_\Omega^{(0)}(w,w)}$$

which completes the proof of (3.36). ∎

Proof (of (3.37) for 1D problems) Because the penalty parameter σ is bounded below and above by positive constants (independent of h), and because $B_\sigma^{(i)}$ is SPSD, it follows that (all displayed relations below are for all $w \in V_0$ and for all $i = 1, ..., N$):

$$0 \leq h^{2-d} B_\sigma^{(i)}(w, w) \lesssim \sum_{e \in \Gamma_i} \int_e \frac{h^{2-d}}{h_e} [w \circ F_i] \cdot [w \circ F_i] . \tag{3.39}$$

For one-dimensional problems, the integral over an edge e should be interpreted as the evaluation of the integrand. Furthermore, h_e denotes the size of the largest mesh element adjacent to e, so regularity (3.3) implies that $\frac{h}{h_e} \lesssim 1$ for all e. Hence, we may write (for $d = 1$):

$$0 \leq h B_\sigma^{(i)}(w, w) \lesssim \sum_{e \in \Gamma_i} [w \circ F_i(e)] \cdot [w \circ F_i(e)] .$$

Finally, observe that, for all $e \in \Gamma_i$, the transformed edge value $F_i(e) =: e_0 \in \Gamma_0$. Different $e \in \Gamma_i$ yield different $e_0 \in \Gamma_0$, although not all $e_0 \in \Gamma_0$ are reached in the presence of Neumann boundary conditions. Nevertheless, we may write:

$$0 \leq h B_\sigma^{(i)}(w, w) \lesssim \underbrace{\sum_{e_0 \in \Gamma_0} [w(e_0)] \cdot [w(e_0)] .}_{=B_\sigma^{(0)}(w,w)}$$

This completes the proof of (3.37) for one-dimensional problems. ∎

PROOF (OF (3.37) FOR 2D PROBLEMS) Similar to the one-dimensional case, we can obtain (3.39). For two-dimensional problems, the edges e are line segments and $h_e = \text{meas}(e)$, i.e. the length of e. Hence, for all e there exists an invertible affine mapping $r_e : [0,1] \to e$. By definition of the line integral over e, we may now rewrite (3.39) as (using $d = 2$):

$$0 \leq B_\sigma^{(i)}(w,w) \lesssim \sum_{e \in \Gamma_i} \frac{1}{h_e} \int_0^1 [w \circ F_i \circ r_e(t)] \cdot [w \circ F_i \circ r_e(t)] \left|r_e'\right| \,\mathrm{d}t$$

Because $r_e(t)$ is affine, its derivative r_e' is a constant and:

$$\left|r_e'\right| = \int_0^1 \left|r_e'\right| \,\mathrm{d}t = \int_e 1 \,\mathrm{d}e = \text{meas}(e) = h_e.$$

Hence:

$$0 \leq B_\sigma^{(i)}(w,w) \lesssim \sum_{e \in \Gamma_i} \int_0^1 [w \circ F_i \circ r_e(t)] \cdot [w \circ F_i \circ r_e(t)] \,\mathrm{d}t.$$

Next, consider a single $e \in \Gamma_i$: note that $F_i \circ r_e([0,1]) = F_i(e) =: e_0 \subset \partial E_0$, and define the invertible affine mapping $r_{e_0} := F_i \circ r_e$. As above, we have that $\left|r_{e_0}'\right| = \text{meas}(e_0)$. By definition of the line integral over e_0, we may now write (using that $\text{meas}(e_0)$ does not depend on h):

$$\int_0^1 [w \circ F_i \circ r_e(t)] \cdot [w \circ F_i \circ r_e(t)] \,\mathrm{d}t = \frac{1}{\text{meas}(e_0)} \int_{e_0} [w] \cdot [w] \lesssim \int_{e_0} [w] \cdot [w] \,.$$

Next, we apply this strategy for all $e \in \Gamma_i$, which yield different (disjunct) $e_0 \subset \partial E_0$, although the entire boundary of E_0 is not reached in the presence of Neumann boundary conditions. Combining the results, we may write (after possible repartitioning of the edges e_0):

$$0 \leq B_\sigma^{(i)}(w,w) \lesssim \underbrace{\sum_{e_0 \in \Gamma_0} \int_{e_0} [w] \cdot [w] \,.}_{= B_\sigma^{(0)}(w,w)}$$

This completes the proof of (3.37) for two-dimensional problems ($d = 2$). ■

PROOF (OF (3.37) FOR 3D PROBLEMS) The proof is similar to the two-dimensional case, except that we are now dealing with surface integrals rather than line integrals. Similar the one-dimensional case, we have (3.39). For three-dimensional problems, the faces e are polygons and $h_e = \sqrt{\text{meas}(e)}$, i.e. the square root of the surface area of e. Because all faces are mutually affine-equivalent (3.2), for all e, there exists an invertible affine mapping $r_e : D \to e$ for some polygon $D \subset \mathbb{R}^2$ (independent of h). By definition of the surface integral over e, we may rewrite (3.39) as (using $d = 3$):

$$\begin{aligned} 0 &\leq h^{-1} B_\sigma^{(i)}(w,w) \\ &\lesssim \sum_{e \in \Gamma_i} \frac{1}{h\, h_e} \int_D [w \circ F_i \circ r_e(u,v)] \cdot [w \circ F_i \circ r_e(u,v)] \left| \frac{\partial r_e}{\partial u} \times \frac{\partial r_e}{\partial v} \right| \mathrm{d}u \,\mathrm{d}v. \end{aligned}$$

Because $r_e(u,v)$ is affine, it follows that its derivatives $\frac{\partial r}{\partial u}$ and $\frac{\partial r}{\partial v}$ are constant, and:

$$\begin{aligned} \left| \frac{\partial r}{\partial u} \times \frac{\partial r}{\partial v} \right| &= \frac{1}{\text{meas}(D)} \int_D \left| \frac{\partial r}{\partial u} \times \frac{\partial r}{\partial v} \right| \mathrm{d}u \,\mathrm{d}v \\ &= \frac{1}{\text{meas}(D)} \int_e 1 \,\mathrm{d}e = \frac{\text{meas}(e)}{\text{meas}(D)} = \frac{h_e^2}{\text{meas}(D)}. \end{aligned}$$

Hence:

$$0 \leq h^{-1} B_\sigma^{(i)}(w,w) \lesssim \sum_{e \in \Gamma_i} \frac{h_e}{h \operatorname{meas}(D)} \int_D [w \circ F_i \circ r_e(u,v)] \cdot [w \circ F_i \circ r_e(u,v)] \, \mathrm{d}u \, \mathrm{d}v$$

Because e can be contained in a circle with diameter h, it follows that $\operatorname{meas}(e) \lesssim h^2$, hence $\frac{h_e}{h} \lesssim 1$:

$$0 \leq h^{-1} B_\sigma^{(i)}(w,w) \lesssim \sum_{e \in \Gamma_i} \frac{1}{\operatorname{meas}(D)} \int_D [w \circ F_i \circ r_e(u,v)] \cdot [w \circ F_i \circ r_e(u,v)] \, \mathrm{d}u \, \mathrm{d}v$$

Next, consider a single $e \in \Gamma_i$: note that $F_i \circ r_e(D) = F_i(e) =: e_0 \subset \partial E_0$, and define the invertible affine mapping $r_{e_0} := F_i \circ r_e$. As above, we have that $\left| \frac{\partial r_{e_0}}{\partial u} \times \frac{\partial r_{e_0}}{\partial v} \right| = \frac{\operatorname{meas}(e_0)}{\operatorname{meas}(D)}$. By definition of the surface integral over e_0, we may now write (using that $\operatorname{meas}(e_0)$ does not depend on h):

$$\begin{aligned} \frac{1}{\operatorname{meas}(D)} \int_D [w \circ F_i \circ r_e(u,v)] \cdot [w \circ F_i \circ r_e(u,v)] \, \mathrm{d}u \, \mathrm{d}v &= \frac{1}{\operatorname{meas}(e_0)} \int_{e_0} [w] \cdot [w] \\ &\lesssim \int_{e_0} [w] \cdot [w] \, . \end{aligned}$$

Next, we apply this strategy for all $e \in \Gamma_i$, which yield different (disjunct) $e_0 \subset \partial E_0$, although the entire boundary of E_0 is not reached in the presence of Neumann boundary conditions. Combining the results, we may write (after possible repartitioning of the edges e_0):

$$0 \leq h^{-1} B_\sigma^{(i)}(w,w) \lesssim \underbrace{\sum_{e_0 \in \Gamma_0} \int_{e_0} [w] \cdot [w] \, .}_{= B_\sigma^{(0)}(w,w)}$$

This completes the proof of (3.37) for three-dimensional problems ($d = 3$). ■

3.4.3 Regularity on the block diagonal of A

As a final step, we now demonstrate that the diagonal blocks of a SIPG matrix A all behave in a similar manner in the space orthogonal to the coarse space. To state this result, we require the following notation: suppose that A_Ω results from the bilinear form $h^{2-d} B_\Omega$ in the same way that A results from the bilinear form B: this is established by substituting A_Ω for A and $h^{2-d} B_\Omega$ for B in (3.6) and (3.7). Similarly, suppose that the matrices A_σ and A_r result from the bilinear forms $h^{2-d} B_\sigma$ and $h^{2-d} B_r$ respectively. Altogether, we may write

$$A = h^{d-2}(A_\Omega + A_\sigma + A_r). \tag{3.40}$$

Finally, let D_σ be the result of extracting the diagonal blocks of size $m \times m$ from A_σ. Using this notation, we now have regularity on the block diagonal of A in the following sense:

Theorem 3.4.3 *Suppose that the diffusion coefficient K and the penalty parameter σ are bounded above and below by positive constants (independent of h). Assume that* (3.2) *and* (3.3) *hold. Then, the matrices A_Ω and D_σ above satisfy the following relations:*

$$\mathbf{v}^T\mathbf{v} \lesssim \mathbf{v}^T A_\Omega \mathbf{v}, \qquad \forall \mathbf{v} \in \mathrm{Range}(I - \pi_I), \tag{3.41}$$

$$\mathbf{v}^T A_\Omega \mathbf{v} \lesssim \mathbf{v}^T\mathbf{v}, \qquad \forall \mathbf{v} \in \mathbb{R}^{mN}, \tag{3.42}$$

$$0 \leq \mathbf{v}^T D_\sigma \mathbf{v} \lesssim \mathbf{v}^T\mathbf{v}, \qquad \forall \mathbf{v} \in \mathbb{R}^{mN}. \tag{3.43}$$

To show this result, the main idea is to observe that A_Ω is an $N \times N$ block diagonal matrix with blocks of size $m \times m$, where the first row and column in every diagonal block is zero: this follows from the fact that $B_\Omega(\phi_k^i, \phi_\ell^{(j)}) = 0$ for $i \neq j$, and that the gradient of the piecewise constant basis function $\phi_1^{(j)}$ is (piecewise) zero. Altogether, A_Ω is of the following form:

$$A_\Omega = \left[\begin{array}{cc|c|cc|c|cc} 0 & 0 & & & & & & \\ 0 & A_\Omega^{(1)} & & & & & & \\ \hline & & \ddots & & & & & \\ \hline & & & 0 & 0 & & & \\ & & & 0 & A_\Omega^{(i)} & & & \\ \hline & & & & & \ddots & & \\ \hline & & & & & & 0 & 0 \\ & & & & & & 0 & A_\Omega^{(N)} \end{array}\right]. \tag{3.44}$$

As a consequence, we can treat the diagonal blocks individually by applying Lemma 3.4.2, and then combine the results (a similar strategy is used for the block diagonal D_σ).

To show (3.41), we also use the nature of $\pi_I = R^T R$, which is the projection operator onto the coarse space $\mathrm{Range}(R^T)$ that yields the best approximation in the 2-norm. As a result, the space $\mathrm{Range}(I - \pi_I)$ is orthogonal to $\mathrm{Range}(R^T)$, where the latter corresponds to the piecewise constant basis functions. In particular, any $\mathbf{v} \in \mathrm{Range}(I - \pi_I) \subset \mathbb{R}^{Nm}$ is of the form:

$$\mathbf{v} = \left[\begin{array}{c} 0 \\ \hline \mathbf{v}_1 \\ \hline \vdots \\ \hline 0 \\ \hline \mathbf{v}_i \\ \hline \vdots \\ \hline 0 \\ \hline \mathbf{v}_N \end{array}\right]. \tag{3.45}$$

Using these ideas, we can now show Theorem 3.4.3:

PROOF (OF THEOREM 3.4.3) Let $A_\Omega^{(i)}$ denote the result of deleting the first row and column in the i^{th} diagonal block in A_Ω, as indicated in (3.44). In other words:

$$\left(A_\Omega^{(i)}\right)_{\ell-1,k-1} = h^{2-d} B_\Omega^{(i)}(\phi_k^{(0)}, \phi_\ell^{(0)}),$$

for all $k, \ell = 2, ..., m$. Next, observe that $B_\Omega^{(0)}$ is independent of h and symmetric. Furthermore, for all higher-order polynomials $v \in \text{span}\{\phi_2^{(0)}, ..., \phi_m^{(0)}\} \setminus \{0\}$, the gradient of v is nonzero, which implies that $B_\Omega^{(0)}(v, v) > 0$. In other words, $B_\Omega^{(0)}$ is even positive-definite for the subspace under consideration. As a consequence, applying Lemma 3.4.2, we obtain a result similar to (3.41), but then for the individual diagonal blocks:

$$\mathbf{w}^T\mathbf{w} \lesssim \mathbf{w}^T A_\Omega^{(i)} \mathbf{w}, \qquad \forall \mathbf{w} \in \mathbb{R}^{m-1}, \qquad \forall i = 1, ..., N.$$

Using the notation in (3.45), this relations hold in particular for $\mathbf{w} = \mathbf{v}_i$, for all $i = 1, ..., N$. Summing over all i then yields:

$$\sum_{i=1}^{N} \mathbf{v}_i^T \mathbf{v}_i \lesssim \sum_{i=1}^{N} \mathbf{v}_i^T A_\Omega^{(i)} \mathbf{v}_i, \qquad \forall \mathbf{v} \in \text{Range}(I - \pi_I),$$

Using the notation in (3.45), this can be rewritten as (3.41), which then completes its proof. The relations (3.42) and (3.43) follow in a similar manner from Lemma 3.4.2 (without deleting the first row and column in each diagonal block). ■

3.5 Main result: scalability for SIPG systems

Combining the results in the previous section, we can now show the main result of this chapter: both two-level methods yield scalable convergence (independent of the mesh element diameter) of the preconditioned CG method for SIPG systems. This result has been shown by Dobrev et al. [24] for the preconditioning variant for $p = 1$. In this section, we extend these results for $p \geq 1$ and for the deflation variant. Section 3.5.1 obtains the aforementioned scalability for a general class of smoothers. Section 3.5.2 shows that the required smoother criteria are valid for block Jacobi smoothing. Section 3.5.3 studies the influence of damping and the penalty parameter on the upper bound of the condition numbers for block Jacobi smoothing.

3.5.1 Main result: scalability for SIPG systems

To state the main scalability result of this chapter, let A be the discretization matrix resulting from an SIPG scheme with $p \geq 1$, as defined in Section 3.2.1. Furthermore, let P_{prec}^{-1} and P_{defl}^{-1} denote the two-level preconditioner and BNN deflation variant respectively, as specified in Section 3.2.3. The main result can now be stated as follows:

Theorem 3.5.1 (Main result) *Suppose that the diffusion coefficient K and the penalty parameter σ are bounded above and below by positive constants (independent of h). Assume that* (3.2), (3.3), *and* (3.5) *hold. Furthermore, assume that the smoother conditions* (3.11), (3.12), (3.13), *and* (3.14) *are satisfied. Then, both two-level methods yield scalable CG convergence in the sense that the condition number κ_2 (in the 2-norm) of the preconditioned system can be bounded independently of the maximum mesh element diameter h:*

$$\kappa_2(P_{\text{prec}}^{-1}A) \lesssim 1, \qquad \kappa_2(P_{\text{defl}}^{-1}A) \lesssim 1. \tag{3.46}$$

To show Theorem 3.5.1, the main idea is to consider Lemma 3.3.2:

$$\kappa_2(P_{\text{prec}}^{-1}A) = K_{\widetilde{M}_{\text{prec}}}, \qquad \kappa_2(P_{\text{defl}}^{-1}A) < 2K_{M_{\text{defl}}}. \tag{3.47}$$

The proof is then completed by showing that $K_{\widetilde{M}_{\text{prec}}}, K_{M_{\text{defl}}} \lesssim 1$, for any smoother that satisfies the criteria above. This is established using the auxiliary result Theorem 3.4.3, and coercivity (3.5) in matrix form:

$$h^{d-2}\mathbf{v}^T\left(A_\Omega + A_\sigma\right)\mathbf{v} \lesssim \mathbf{v}^T A\mathbf{v}, \qquad \forall \mathbf{v} \in \mathbb{R}^{Nm}. \tag{3.48}$$

Altogether, Theorem 3.5.1 can now be shown as follows:

Proof (of Theorem 3.5.1) First, we will show that $K_{\widetilde{M}_{\text{prec}}} \lesssim 1$ (a similar strategy yields $K_{M_{\text{defl}}} \lesssim 1$). For ease of notation, we will write $\widetilde{M}$ for $\widetilde{M}_{\text{prec}}$. The main idea is to show that $\left\|(I-\pi_{\widetilde{M}})\mathbf{v}\right\|_{\widetilde{M}} \lesssim \|\mathbf{v}\|_A$ for all $\mathbf{v}$: because $\pi_{\widetilde{M}}$ is a projection onto the coarse space Range(R^T) that yields the best approximation in the $\widetilde{M}$-norm, we can replace $\pi_{\widetilde{M}}$ by the suboptimal projection π_I, and then combine the properties established so far:

$$\begin{aligned}
\left\|(I-\pi_{\widetilde{M}})\mathbf{v}\right\|_{\widetilde{M}}^2 &\leq \|(I-\pi_I)\mathbf{v}\|_{\widetilde{M}}^2 \\
&\overset{(3.12)}{\lesssim} h^{d-2}\|(I-\pi_I)\mathbf{v}\|_2^2 \\
&\overset{(3.41)}{\lesssim} h^{d-2}\|(I-\pi_I)\mathbf{v}\|_{A_\Omega}^2 \\
&\overset{(3.44),\,(3.45)}{=} h^{d-2}\|\mathbf{v}\|_{A_\Omega}^2 \\
&\overset{B_\sigma \text{ SPSD} \Rightarrow A_\sigma \text{ SPSD}}{\lesssim} h^{d-2}\|\mathbf{v}\|_{A_\Omega + A_\sigma}^2 \\
&\overset{(3.48)}{\lesssim} \|\mathbf{v}\|_A^2 \qquad \forall \mathbf{v} \in \mathbb{R}^{Nm}.
\end{aligned}$$

Substitution of this relation into the definition of $K_{\widetilde{M}}$ yields:

$$K_{\widetilde{M}} := \sup_{\mathbf{v}\neq 0} \frac{\left\|(I-\pi_{\widetilde{M}})\mathbf{v}\right\|_{\widetilde{M}}^2}{\|\mathbf{v}\|_A^2} \lesssim 1,$$

A similar strategy, using (3.14) instead of (3.12), yields $K_{M_{\text{defl}}} \lesssim 1$. Substitution of $K_{\widetilde{M}_{\text{prec}}}, K_{M_{\text{defl}}} \lesssim 1$ into (3.47) now yields (3.46), which completes the proof of Theorem 3.5.1. ■

3.5.2 Special case: block Jacobi smoothing

This section demonstrates that Theorem 3.5.1 is valid for (damped) block Jacobi smoothing. To specify this result, suppose that M_{BJ} is the block Jacobi smoother with blocks of size $m \times m$. Next, consider the specific choice

$$M_{\mathrm{prec}} = M_{\mathrm{defl}} = \omega^{-1} M_{\mathrm{BJ}}$$

with damping parameter $\omega > 0$ (independent of h). We assume that $\omega \leq 1$, with $\omega < 1$ strictly for the preconditioning variant.

Additionally, we assume that there exists a permutation matrix P such that A can be permuted as:

$$PAP^T = \Delta - L - L^T, \tag{3.49}$$

with

$$\Delta = \begin{pmatrix} \Delta_1 & & & \\ & \Delta_2 & & \\ & & \ddots & \\ & & & \Delta_q \end{pmatrix}, \qquad -L = \begin{pmatrix} 0 & & & \\ L_1 & 0 & & \\ & \ddots & 0 & \\ & & L_{q-1} & 0 \end{pmatrix},$$

for some block-diagonal matrices $\Delta_1, ..., \Delta_q$ with blocks of size $m \times m$, matrices $L_1, ..., L_{q-1}$, and integer $q \leq N$. Note that this assumption implies that the matrix A has property A^π in the sense of [5, Definition 6.7]. Moreover, we remark that (3.49) is satisfied if the mesh can be colored by two colors[9] (in that case, we can choose $q = 2$, and Δ_1 and Δ_2 each correspond to one of the two colors). In particular, structured rectangular meshes can be colored by two colors and thus satisfy (3.49).

Altogether, assuming[10] (3.49), we can now show that all smoother requirements for Theorem 3.5.1 are satisfied for (damped) block Jacobi smoothing:

Corollary 3.5.2 *If* (3.49) *is satisfied, then Theorem 3.5.1 applies for the damped block Jacobi smoothers* M_{prec} *and* M_{defl} *above, i.e. both two-level methods yield scalable CG convergence in the sense that the condition number* κ_2 *(in the* 2*-norm) of the preconditioned system can be bounded independently of the maximum mesh element diameter* h*:*

$$\kappa_2(P_{\mathrm{prec}}^{-1} A) \lesssim 1, \qquad \kappa_2(P_{\mathrm{defl}}^{-1} A) \lesssim 1.$$

This result follows immediately from Theorem 3.5.1 once we have verified that the conditions (3.11), (3.13), (3.12), and (3.14) are satisfied for the damped

[9] That is, the mesh can be represented by a graph whose vertices can be colored such that connected vertices do not have the same color.

[10] Alternatively, we could assume that the damping parameter ω is sufficiently small. This option is not considered further in this thesis.

block Jacobi smoothers under consideration. In other words, writing $M := \omega^{-1} M_{\mathrm{BJ}}$, we need to show:

$$2M - A > 0, \tag{3.50}$$

$$h^{2-d}\mathbf{v}^T M\mathbf{v} \lesssim \mathbf{v}^T\mathbf{v}, \qquad \forall \mathbf{v} \in \mathrm{Range}(I - \pi_I), \tag{3.51}$$

$$h^{2-d}\mathbf{v}^T \widetilde{M}\mathbf{v} \lesssim \mathbf{v}^T\mathbf{v}, \qquad \forall \mathbf{v} \in \mathrm{Range}(I - \pi_I), \tag{3.52}$$

for all $\omega \leq 1$, with $\omega < 1$ strictly for (3.52). We treat each relation separately.

To show (3.50), we use that (ρ denotes the spectral radius) :

$$\rho(B) < 1, \qquad B := \Delta^{-1}(L + L^T), \tag{3.53}$$

which follows from [5, Theorem 6.38] using (3.49) and the fact that A and M are SPD.

Proof (of (3.50)) Without loss of generality, assume that $\omega = 1$. Next, observe that $PMP^T = \Delta$. Hence,

$$\begin{aligned} P(2M - A)P^T &\overset{PMP^T=\Delta}{=} 2\Delta - PAP^T \\ &\overset{PAP^T:=\Delta-L-L^T}{=} 2\Delta - \Delta + L + L^T \\ &= \Delta\big(I + \underbrace{\Delta^{-1}(L + L^T)}_{=:B}\big) \\ &= \Delta(I + B). \end{aligned}$$

Hence,

$$\lambda_{\min}(2M - A) = \lambda_{\min}(I + \Delta^{\frac{1}{2}} B \Delta^{-\frac{1}{2}}).$$

And because (3.53) implies that $\rho(B) = \rho(\Delta^{\frac{1}{2}} B \Delta^{-\frac{1}{2}}) < 1$, it follows that $2M - A > 0$. This completes the proof. ∎

To show (3.51), the main idea is to use Theorem 3.4.3 and the following property (cf. [24, p. 760] and [43, p. 4]):

$$0 < B(v, v) \lesssim B_\Omega(v, v) + B_\sigma(v, v), \qquad \forall \mathbf{v} \in \mathbb{R}^{Nm}. \tag{3.54}$$

Proof (of (3.51)) Without loss of generality, assume that $\omega = 1$. Next, recall the notation introduced in the beginning of Section 3.4.3. Additionally, similar to D_σ, let D_r be the result of extracting the diagonal blocks of size $m \times m$ from A_r. Using this notation, and the fact that A_Ω is a block diagonal matrix with blocks of size $m \times m$, we may write:

$$h^{2-d}M = A_\Omega + D_\sigma + D_r. \tag{3.55}$$

Next, we write (3.54) in matrix form:

$$0 < \mathbf{v}^T A\mathbf{v} \lesssim \mathbf{v}^T \left(h^{d-2} A_\Omega + h^{d-2} A_\sigma\right)\mathbf{v}, \qquad \forall \mathbf{v} \in \mathbb{R}^{Nm}.$$

Because this relation is also true when considering the diagonal blocks only, we may write:

$$h^{2-d}\mathbf{v}^T M\mathbf{v} \lesssim \mathbf{v}(A_\Omega + D_\sigma)\mathbf{v}, \qquad \forall \mathbf{v} \in \mathbb{R}^{Nm}.$$

Application of Theorem 3.4.3 now yields (3.51), which completes the proof. ∎

To show (3.52), we combine the previous results (3.50) and (3.51):

Proof (of (3.52)) Using (3.50), and the fact that $\omega < 1$ strictly, it can be shown that

$$\widetilde{M} \leq \frac{1}{2(1-\omega)} M, \tag{3.56}$$

which can be seen as follows:

$$\begin{aligned} 2\omega M - A &\geq 0, \\ 2M - A = (2-2\omega)M + \underbrace{2\omega M - A}_{\geq 0} &\geq (2-2\omega)M, \\ (2M-A)^{-1} &\overset{\text{[40, p. 398, 471]}}{\leq} \frac{1}{2(1-\omega)} M^{-1}, \\ \underbrace{M(2M-A)^{-1}M}_{=:\widetilde{M}} &\leq \frac{1}{2(1-\omega)} M. \end{aligned}$$

Combining this relation with (3.51), it now follows that

$$h^{2-d}\mathbf{v}^T \widetilde{M}\mathbf{v} \overset{(3.56)}{\leq} \frac{1}{2(1-\omega)} h^{2-d}\mathbf{v}^T M\mathbf{v} \overset{(3.51)}{\lesssim} \mathbf{v}^T\mathbf{v}, \qquad \forall \mathbf{v} \in \text{Range}(I - \pi_I).$$

This completes the proof of (3.52). ■

3.5.3 Influence of damping and the penalty parameter for block Jacobi smoothing

In Section 2.5.2, we studied the influence of damping and the penalty parameter on the CG convergence. We found that both two-level methods perform significantly better if the penalty parameter is chosen dependent on local values of the diffusion coefficient. Furthermore, a damping parameter around $\omega = 0.7$ was observed to be optimal for the preconditioning variant during the numerical experiments. To gain more insight in these results, in this section, we study the influence of damping and the penalty parameter on the constants in Corollary 3.5.2, where the same damped block Jacobi smoother is used for both two-level methods.

Regarding damping, it can be shown (the proof is given at the end of this section):

$$\kappa_2(P_{\text{defl}}^{-1}A) \leq \frac{2}{\omega} K_{M_{\text{BJ}}}, \qquad \kappa_2(P_{\text{prec}}^{-1}A) < \frac{1}{2\omega(1-\omega)} K_{M_{\text{BJ}}}. \tag{3.57}$$

Although these upper bounds may not be optimal, the fact that the upper bound for the preconditioner blows up as ω tends to 1 is in line with our earlier numerical observation in Section 2.5 that the preconditioning variant performs better for ω safely away from 1.

To study the influence of the penalty parameter, let $\sigma_{\max}^{(i)}$ denote the largest value that the penalty parameter σ attains at the edges of mesh element E_i, and let $K_{\min}^{(i)}$ denote the smallest value that the diffusion coefficient K attains

within E_i (for all $i = 1, ..., N$). We can now bound $K_{M_{\mathrm{BJ}}}$ in terms of the local ratio between the penalty parameter and the diffusion coefficient, assuming[11] $A_\sigma + A_r \geq 0$ (the proof is given at the end of this section):

$$K_{M_{\mathrm{BJ}}} \leq C_1 \max_{i=1,\ldots,N} \frac{\sigma_{\max}^{(i)}}{K_{\min}^{(i)}} + C_2, \tag{3.58}$$

for some positive constants C_1 and C_2 that are independent of the mesh element diameter h *and* the penalty parameter σ (but possibly dependent on the diffusion coefficient K). The result of substituting (3.58) into (3.57) is in line with the observation in Section 2.5 that the penalty parameter can best be chosen dependent on local values of the diffusion coefficient.

We end this section with the proofs of (3.57) and (3.58):

Proof (of (3.57)) The first inequality follows from (3.47), (3.33) and the fact that $M_{\mathrm{defl}} = \omega^{-1} M_{\mathrm{BJ}}$. To show the second inequality, we use that (3.50) implies that $\lambda_{\max}(M_{\mathrm{prec}}^{-1} A) < 2\omega$:

$$\begin{aligned}
\kappa_2(P_{\mathrm{prec}}^{-1} A) &\overset{(3.47)}{=} K_{\widetilde{M}_{\mathrm{prec}}} \\
&\overset{(3.33),\ (3.32)}{\leq} \frac{1}{2 - \lambda_{\max}(M_{\mathrm{prec}}^{-1} A)} K_{M_{\mathrm{prec}}} \\
&\overset{\lambda_{\max}(M_{\mathrm{prec}}^{-1} A) < 2\omega}{<} \frac{1}{2(1-\omega)} K_{M_{\mathrm{prec}}} \\
&\overset{(3.33), M_{\mathrm{prec}} = \omega^{-1} M_{\mathrm{BJ}}}{\leq} \frac{1}{2\omega(1-\omega)} K_{M_{\mathrm{BJ}}}.
\end{aligned}$$

This completes the proof of (3.57). ■

Proof (of (3.58)) By definition,

$$K_{M_{\mathrm{BJ}}} \overset{(3.15)}{=} \sup_{\mathbf{v} \neq 0} \frac{\left\| (I - \pi_{M_{\mathrm{BJ}}}) \mathbf{v} \right\|_{M_{\mathrm{BJ}}}^2}{\|\mathbf{v}\|_A^2}.$$

As in the proof of Theorem 3.5.1, we may replace $\pi_{M_{\mathrm{BJ}}}$ by the suboptimal projection π_I:

$$K_{M_{\mathrm{BJ}}} \leq \sup_{\mathbf{v} \neq 0} \frac{\|(I - \pi_I)\mathbf{v}\|_{M_{\mathrm{BJ}}}^2}{\|\mathbf{v}\|_A^2}.$$

Using the notation in (3.40) and (3.55), we can rewrite this as (note that the factor h^{d-2} cancels):

$$K_{M_{\mathrm{BJ}}} \leq \sup_{\mathbf{v} \neq 0} \frac{\|(I - \pi_I)\mathbf{v}\|_{A_\Omega + D_\sigma + D_r}^2}{\|\mathbf{v}\|_{A_\Omega + A_\sigma + A_r}^2}.$$

Next, we use the assumption $A_\sigma + A_r \geq 0$:

$$K_{M_{\mathrm{BJ}}} \overset{A_\sigma + A_r \geq 0}{\leq} \sup_{\mathbf{v} \neq 0} \frac{\|(I - \pi_I)\mathbf{v}\|_{A_\Omega + D_\sigma + D_r}^2}{\|\mathbf{v}\|_{A_\Omega}^2}$$

[11] This condition seems closely related to coercivity (3.5). How either can be guaranteed in practice (for problems with strong contrasts in the coefficients) is left for future research.

$$\overset{(3.44),\,(3.45)}{=} \sup_{\mathbf{v}\neq 0} \frac{\|(I-\pi_I)\mathbf{v}\|^2_{A_\Omega+D_\sigma+D_r}}{\|(I-\pi_I)\mathbf{v}\|^2_{A_\Omega}}$$

$$= 1 + \sup_{\mathbf{v}\in \mathrm{Range}(I-\pi_I)} \frac{\|\mathbf{v}\|^2_{D_\sigma+D_r}}{\|\mathbf{v}\|^2_{A_\Omega}}.$$

Next, consider the notation for $A_\Omega^{(i)}$ in (3.44) and, similarly, let $D_\sigma^{(i)}$ and $D_r^{(i)}$ denote the result of removing the first row and column from diagonal blok i in D_σ and D_r respectively. Then, we may write, using the notation for the components of $\mathbf{v}$ in (3.45):

$$K_{M_{\mathrm{BJ}}} \leq 1 + \sup_{\mathbf{v}\in \mathrm{Range}(I-\pi_I)} \frac{\sum_{i=1}^N \mathbf{v}_i^T (D_\sigma^{(i)} + D_r^{(i)}) \mathbf{v}_i^T}{\sum_{i=1}^N \mathbf{v}_i^T A_\Omega^{(i)} \mathbf{v}_i}.$$

At the same time, it can be shown (similar to Section 3.4) that there exist positive constants C_Ω, C_σ, C_r independent of h and σ, with C_Ω, C_σ also independent of K, such that, for all $\mathbf{w} \in \mathrm{R}^{m-1}$:

$$\mathbf{w}^T A_\Omega^{(i)} \mathbf{w} \geq C_\Omega K_{\min}^{(i)} \mathbf{w}^T \mathbf{w},$$
$$\mathbf{w}^T D_\sigma^{(i)} \mathbf{w} \leq C_\sigma \sigma_{\max}^{(i)} \mathbf{w}^T \mathbf{w},$$
$$\mathbf{w}^T D_r^{(i)} \mathbf{w} \leq C_r \mathbf{w}^T \mathbf{w}.$$

Combining these relations gives:

$$\mathbf{v}_i^T (D_\sigma^{(i)} + D_r^{(i)}) \mathbf{v}_i^T \leq \frac{C_\sigma \sigma_{\max}^{(i)} + C_r}{C_\Omega K_{\min}^{(i)}} \mathbf{v}_i^T A_\Omega^{(i)} \mathbf{v}_i.$$

Using the latter relation, we may now write:

$$K_{M_{\mathrm{BJ}}} \leq 1 + \sup_{\mathbf{v}\in \mathrm{Range}(I-\pi_I)} \frac{\sum_{i=1}^N \left(\frac{C_\sigma \sigma_{\max}^{(i)} + C_r}{C_\Omega K_{\min}^{(i)}} \mathbf{v}_i^T A_\Omega^{(i)} \mathbf{v}_i \right)}{\sum_{i=1}^N \mathbf{v}_i^T A_\Omega^{(i)} \mathbf{v}_i}$$

$$\leq 1 + \max_{i=1,\ldots,N} \left\{ \frac{C_\sigma \sigma_{\max}^{(i)}}{C_\Omega K_{\min}^{(i)}} \right\} + \max_{i=1,\ldots,N} \left\{ \frac{C_r}{C_\Omega K_{\min}^{(i)}} \right\}.$$

This can be rewritten as (3.58), which then completes the proof. ■

3.6 Conclusion

This chapter is focused on the theoretical analysis of the two-level preconditioner and deflation variant studied numerically in Chapter 2 for linear systems resulting from SIPG discretizations for diffusion problems. For both two-level methods, we have found that the condition number of the preconditioned system can be bounded independently of the mesh element diameter. This result is valid for any polynomial degree $p \geq 1$, which extends the available analysis for the preconditioning variant for $p = 1$ in [24]. We have verified that the restrictions on the smoother are satisfied for block Jacobi smoothing. Altogether, our theory explains the scalable CG convergence observed during the numerical experiments in Chapter 2, and guarantees similar results for a large class of other diffusion problems on a variety of meshes.

4

Hidden DG accuracy

This chapter is based on:

P. van Slingerland, J.K. Ryan, C. Vuik, *Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering for improving Discontinuous Galerkin solutions*, SIAM J. Sci. Comp., **33**(2011), pp 802–825.

4.1 Introduction

DG approximations can contain 'hidden accuracy': although the convergence rate of a DG scheme is typically of order $p + 1$ (where p is the polynomial degree), it can be improved to order $2p + 1$ by applying a post-processor (cf. Section 1.4). Interestingly, this post-processor does not contain any information of the underlying physics or numerics, and needs to be applied only once, at the final time.

The main idea behind the post-processor is to compute a convolution of the DG approximation against a linear combination of $2p+1$ B-splines. A positive side effect of this strategy is that the smoothness of the B-splines is carried over to the DG approximation, which then becomes $p - 1$ times continuously differentiable. This enhanced smoothness can benefit the visualization of the approximation, e.g. in the form of streamlines.

The aforementioned filter, also known as the symmetric post-processor, was introduced by Bramble and Schatz [11] in the context of Ritz-Galerkin approximations for elliptic problems. Cockburn, Luskin, Shu, and Süli [22] demonstrated that the same technique can also be used to enhance the convergence rate of DG approximations from order $p + 1$ to order $2p + 1$. This was shown for linear periodic hyperbolic problems with a sufficiently smooth exact solution. An overview of the development of post-processing techniques can also be found in [22].

To make the post-processor applicable near non-periodic boundaries and shocks, Ryan and Shu [64] introduced the one-sided post-processor. Inspired by the ideas in [16, 35, 51], they shifted the support of the local averaging operator to one side of the evaluation point. Using the resulting one-sided post-processor near boundaries and shocks and the original symmetric post-processor in the interior, it was now possible to post-process the entire domain, even for non-smooth solutions or non-periodic boundary conditions.

To improve the accuracy and smoothness of this one-sided strategy, in this chapter, we propose a position-dependent post-processor. In particular, we investigate the impact of using extra B-splines near the boundary. This way, we seek to reduce the possibility that the post-processor worsens the errors near the boundary (despite superconvergence). Furthermore, we study the effect of smoother transitions in the position of the B-splines. With this strategy, we aim to eliminate the necessity of (re)introducing artificial discontinuities. To compare the performance of the resulting position-dependent post-processor and the original one-sided filter, we discuss seven numerical experiments, including a problem with stationary shocks, a two-dimensional system, and a streamline visualization example (theoretical error estimates are discussed in Chapter 5).

The outline of this chapter is as follows. Section 4.2 specifies the DG schemes under consideration. Section 4.3 provides the basics of the original symmetric and one-sided post-processor. Section 4.4 introduces the position-dependent post-processor. Section 4.5 discusses the numerical experiments.

Section 4.6 summarizes the main conclusions.

4.2 Discretization

This section summarizes the DG method for linear hyperbolic problems. Section 4.2.1 considers the one-dimensional case, following [21]. Section 4.2.2 discusses two-dimensional systems, similar to [22].

4.2.1 DG for one-dimensional hyperbolic problems

Consider the following problem on the interval $[a, b]$:

$$u_t + \big(cu\big)_x = f,$$

with initial condition u_0 and either periodic or Dirichlet boundary conditions at $x = a$. The functions c and f may depend on space and time, but not on u. We assume that the velocity c is positive.

To construct a DG approximation for this problem, consider a uniform mesh with elements $E_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ of length $h > 0$, where $x_{\frac{1}{2}} = a$ and $x_{i+\frac{1}{2}} = x_{i-\frac{1}{2}} + h$ (for all $i = 1, ..., N$). Next, define the test space V that contains each function that is a polynomial of degree p or lower within each mesh element, and that may be discontinuous at the mesh element boundaries. For all $v \in V$, we let $v^{(i)}$ denote the (continuous) restriction of v to E_i.

At the initial time $t = 0$, the DG approximation u_h is the L^2-projection of u_0 onto V. For $t > 0$, it is the function in V such that:

$$\int_a^b (u_h)_t \, v + B(u_h, v) = \int_a^b fv, \qquad \forall v \in V,$$

where B is the following bilinear form (defining $u_h^{(0)}|_{x_{\frac{1}{2}}}$ in terms of the boundary condition for u at $x = a$):

$$B(u_h, v) = -\sum_{i=1}^{N} \int_{E_i} cu_h v_x + \sum_{i=1}^{N} \left((cu_h^{(i)} v^{(i)})|_{x_{i+\frac{1}{2}}} - (cu_h^{(i-1)} v^{(i)})|_{x_{i-\frac{1}{2}}} \right).$$

This form uses an upwind flux approximation for u_h at the element boundaries. For more details, cf. [21].

4.2.2 DG for two-dimensional hyperbolic systems

Similar to the one-dimensional case, we can construct a DG approximation for two-dimensional hyperbolic systems. To specify this, consider the following problem on a square domain Ω:

$$u_t + A_1 u_{x_1} + A_2 u_{x_2} = f, \tag{4.1}$$

with initial condition u_0 and periodic boundary conditions. Here, u and f are vector-valued functions with m entries and the coefficients A_j are constant matrices of size $m \times m$. We assume that the system above is strongly hyperbolic in the sense that, for all $n \in \mathbb{R}^2$, there exists a diagonal matrix Λ and a nonsingular matrix R such that

$$R(A_1 n_1 + A_2 n_2)R^{-1} = \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix}. \tag{4.2}$$

To construct a DG approximation for the system (4.1), we use a uniform Cartesian mesh for the spatial domain Ω with compact mesh elements $E_1, ..., E_N$ of size $h \times h$. Next, define the test space V that contains each vector-valued function with m entries that are polynomials of degree p or lower within each mesh element, and that may be discontinuous at the mesh element boundaries. For all $v \in V$, we let $v^{(i)}$ denote the (continuous) restriction of v to E_i.

At the initial time $t = 0$, the DG approximation u_h is the L^2-projection of u_0 onto V. For $t > 0$, it is the function in V such that ($\langle ., . \rangle$ denotes the standard inner product in ℓ^2):

$$\int_\Omega \langle (u_h)_t, v \rangle + B(u_h, v) = \int_\Omega \langle f, v \rangle, \qquad \forall v \in V,$$

where B is a bilinear form specified hereafter.

To define B, let $\hat{u}_h$ denote the following upwind flux approximation for u_h on the mesh element boundaries [22, p. 587]: consider an edge e of mesh element E_i with outward normal $n^{(i)} = (n_1^{(i)}, n_2^{(i)})$ and neighboring element E_j. Using the notation in (4.2) (substituting $n = n^{(i)}$), define $w := R^{-1} u_h$. We then set $\hat{u}_h = R\hat{w}$, where the k^{th} entry of $\hat{w}$ is defined as:

$$\hat{w}_k = \begin{cases} v_k^{(i)}, & \text{if } \lambda_k > 0, \\ v_k^{(i)}, & \text{else.} \end{cases}$$

Using this numerical flux, the bilinear form B can be specified as follows:

$$\begin{aligned} B(u_h, v) \quad = \quad & -\sum_{i=1}^{N} \int_{E_i} \left\langle u_h, A_1^T v_{x_1} + A_2^T v_{x_2} \right\rangle \\ & + \sum_{i=1}^{N} \sum_{e \in \partial E_i} \int_e \left\langle (A_1 n_1^{(i)} + A_2 n_2^{(i)}) \hat{u}_h, v^{(i)} \right\rangle. \end{aligned}$$

4.3 Original post-processing strategies

The smoothness and accuracy of a DG approximation can be improved by applying a post-processer at the final simulation time. This section provides

the basics of this technique. Section 4.3.1 defines B-splines [70, 71], which are the building blocks of the filter. Section 4.3.2 discusses the original symmetric post-processor studied in [11, 22]. Section 4.3.3 considers the one-sided post-processor introduced in [64] for post-processing near non-periodic boundaries and shocks.

4.3.1 B-splines

A B-spline $\psi^{(p+1)}$ of order $p+1$ can be defined recursively in the following manner[1]:

$$\psi^{(1)} := \mathbb{1}_{[-\frac{1}{2},\frac{1}{2}]}, \qquad \psi^{(p+1)} := \psi^{(p)} \star \psi^{(1)}, \qquad \text{for all } p \geq 1. \tag{4.3}$$

Figure 4.1 provides an illustration of a B-spline. In general, a B-spline of order $p+1$ is a piecewise polynomial of degree p that is $p-1$ times continuously differentiable. Moreover, its support reads $[-\frac{p+1}{2}, \frac{p+1}{2}]$. For more details on B-splines, cf. [70, 71].

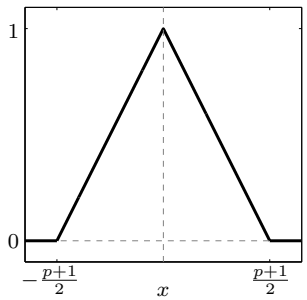


FIGURE 4.1
Illustration of a ***B-spline*** $\psi^{p+1}(x)$ *for* $p = 1$. *Its support is contained in* $[-\frac{p+1}{2}, \frac{p+1}{2}]$.

4.3.2 Symmetric Post-processor

The symmetric post-processor [11, 22] enhances a DG approximation (with polynomial degree p) by convolving it against a linear combination of B-splines.

To specify this technique, we choose $2p+1$ integer nodes that are located symmetrically around the origin:

$$x_j = -p + j, \qquad j = 0, ..., 2p. \tag{4.4}$$

Next, we place a B-spline of order $p+1$ at each kernel node, and define a kernel K that is a linear combination of these B-splines:

$$K(x) = \sum_{j=0}^{2p} c_j \psi^{(p+1)}(x - x_j), \qquad \text{for all } x \in \mathbb{R}, \tag{4.5}$$

[1]Here, the symbol $\star$ denotes the convolution operator. Furthermore, $\mathbb{1}_{[-\frac{1}{2},\frac{1}{2}]}$ is the indicator function that is 1 in $[-\frac{1}{2}, \frac{1}{2}]$ and 0 elsewhere.

where the coefficients c_j are determined by the following linear system:

$$\sum_{j=0}^{2p} c_j \int_{-\infty}^{\infty} \psi^{(p+1)}(x)(x+x_j)^k \,\mathrm{d}x = \begin{cases} 1, & \text{for } k=0, \\ 0, & \text{else.} \end{cases} \tag{4.6}$$

Existence and uniqueness of the solution of the system in (4.6) have been shown in [10, Lemma 8.1]. The coefficients are chosen in this way to ensure that the kernel reproduces polynomials q of degree $2p$ and lower in the sense that $K \star q = q$. The relevance of this property will be discussed further in Section 5.3.1.

The symmetric post-processor can now be defined as follows:

Definition 4.3.1 (Symmetric Post-processor) Consider a *periodic* DG approximation u_h at some final simulation time on the interval $[a, b]$, using mesh elements of size h and polynomials of degree p, as discussed in Section 4.2.1. Let K denote the kernel defined by (4.4), (4.5) and (4.6). Then, the result of post-processing u_h in the evaluation point $\bar{x} \in (a, b)$ is computed by convolving u_h against the scaled kernel K (using a periodic extension of u_h when needed):

$$u_h^*(\bar{x}) = \frac{1}{h} \int_a^b K\left(\frac{\bar{x}-x}{h}\right) u_h(x) \,\mathrm{d}x. \tag{4.7}$$

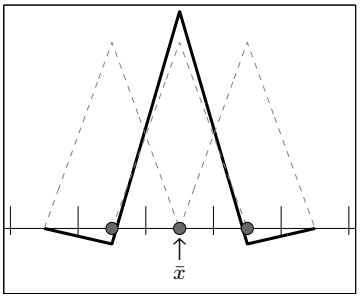


FIGURE 4.2
Application of the scaled ***symmetric kernel*** $K\left(\frac{\bar{x}-x}{h}\right)$ *at the evaluation point* $\bar{x}$ *in the mesh* ($p = 1$). *The kernel nodes are indicated by circles, and the corresponding B-splines by dashed lines.*

Figure 4.2 illustrates the symmetric post-processor. Although this technique does not contain any information of the underlying physics or numerics, it has been shown in [22] that it enhances the DG convergence rate from order $p+1$ to order $2p+1$ for the linear periodic hyperbolic problems under consideration, assuming that the exact solution is sufficiently smooth.

A second feature of the post-processor is that the smoothness of the B-splines is carried over to the approximation, i.e. $u_h^\star$ is $p-1$ times continuously differentiable. This enhanced smoothness can be benefit the visualization of the approximation, e.g. in the form of streamlines (also cf. Section 4.5.7).

Finally, we remark that the application of the post-processor is not restricted to the DG approximations above, but can be applied to any function, also in higher dimensions (cf. Section 4.4.3). However, whether this also yields acceptable accuracy and efficiency does depend on the underlying problem. The post-processor has been effectively applied for DG discretizations on non-uniform rectangular [23] and unstructured triangular [48] meshes. Furthermore, it has been shown successful for certain linear convection-diffusion problems [41] and nonlinear hyperbolic conservation laws [42].

4.3.3 One-sided post-processor

The symmetric post-processor is an effective technique for enhancing the accuracy and smoothness for periodic problems with a sufficiently smooth exact solution. To be able to post-process near non-periodic boundaries and shocks as well, the one-sided post-processor has been proposed in [64].

This technique is similar to the symmetric case, except that the kernel nodes are shifted to one side of the origin, e.g. the right side:

$$x_j = \left\lceil \frac{p+1}{2} \right\rceil + j, \qquad \text{for all } j = 0, ..., 2p. \tag{4.8}$$

The resulting right-sided post-processor is now obtained by replacing (4.4) by (4.8) in Definition 4.3.1 (note that the kernel coefficients c_j now take different values).

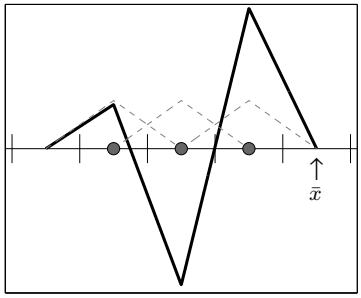


FIGURE 4.3
Application of the scaled ***right-sided kernel*** $K\left(\frac{\bar{x}-x}{h}\right)$ *at the evaluation point* $\bar{x}$ *in the mesh* ($p = 1$). *The kernel nodes are indicated by circles, and the corresponding B-splines by dashed lines. Unlike the symmetric kernel, it can be applied near the right boundary.*

Figure 4.3 illustrates the right-sided post-processor. Unlike the symmetric post-processor (cf. Figure 4.2), it can be applied anywhere close to the right boundary of the domain, without requiring information outside the spatial domain. This is because the support of the right-sided kernel is located entirely on the right side of the origin (although the support of $K\left(\frac{\bar{x}-\cdot}{h}\right)$ is located on the left side of the evaluation point $\bar{x}$, as illustrated). This follows from (4.5), (4.8), and the fact that the support of the B-spline $\psi^{(p+1)}$ is contained in the interval $[-\frac{p+1}{2}, \frac{p+1}{2}]$.

Although the right-sided post-processor is suitable near the right boundary, it is not suitable near the left boundary, as it would require information outside of the spatial domain in this region (this is similar to the symmetric case). To be able to post-process near the left boundary, we can reverse the strategy above and shift the kernel nodes to the other (left) side of the origin. The resulting left-sided post-processor is applicable near the left boundary, but not the right.

Although neither of the post-processors can be applied in the entire domain, we can still cover the entire domain by combining the previous kernel types: in the interior, we can use the symmetric kernel; at the right boundary, we can use the right-sided kernel; at the left boundary, we can use the left-sided kernel; and in the transition regions, we can use kernels that are between a symmetric and a one-sided kernel (specified below). Altogether, we obtain the following post-processor, as proposed in [64]:

Definition 4.3.2 ((Combined) one-sided post-processor) Consider a DG approximation u_h at some final simulation time on the interval $[a, b]$, using mesh elements of size h and polynomials of degree p, as discussed in Section 4.2.1. Let $\bar{x}$ denote an evaluation point in (a, b). Depending on this evaluation point, define the following nodes:

$$x_j = -p + j + \lambda(\bar{x}), \qquad \text{for all } j = 0, ..., 2p, \tag{4.9}$$

$$\lambda(\bar{x}) = \begin{cases} \min\{0, -\lceil \frac{3p+1}{2} \rceil + \lfloor \frac{\bar{x}-a}{h} \rfloor\}, & \text{for } \bar{x} \in [a, \frac{a+b}{2}), \\ \max\{0, \lceil \frac{3p+1}{2} \rceil + \lceil \frac{\bar{x}-b}{h} \rceil\}, & \text{for } \bar{x} \in [\frac{a+b}{2}, b]. \end{cases} \tag{4.10}$$

Let K be the result of substituting these nodes into (4.5) and (4.6)[2]. Then, the result of post-processing u_h in $\bar{x}$ is obtained by computing the convolution with the scaled kernel K as in (4.7).

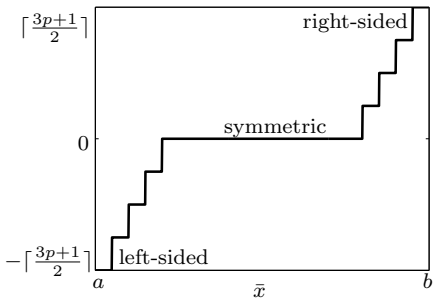


FIGURE 4.4
*Illustration of the **shift function** $\lambda(\bar{x})$ in (4.10). This function selects the appropriate kernel, depending on the evaluation point $\bar{x}$.*

[2]As for the symmetric kernel, existence and uniqueness of the kernel coefficients defined by (4.6) using shifted nodes can be shown following [10, Lemma 8.1].

The shift function λ in (4.10) is designed so that the post-processor above can be applied in the entire domain, even for non-periodic boundary conditions. An illustration of λ is given in Figure 4.4. Different values of λ result in different kernels, so the kernel has become dependent on the evaluation point $\bar{x}$ through λ: the case $\lambda = 0$ corresponds to the symmetric kernel. Similarly, for $\lambda = -\lceil \frac{3p+1}{2} \rceil$ and $\lambda = \lceil \frac{3p+1}{2} \rceil$, the left- and right-sided kernels are obtained.

Observe that the symmetric kernel is applied whenever possible. The reason for this is that the symmetric post-processor yields better accuracy than the non-symmetric variants. The latter has been observed in [64, p. 298] and will also be demonstrated in Section 4.5.

Finally, note that we can now post-process near shocks by treating them as a domain boundary. This is established by subdividing the spatial domain into subdomains such that the shocks occur at the boundaries of these subdomains. After that, each subdomain can be post-processed individually as prescribed by Definition 4.3.2. Unlike the symmetric post-processor, which would render the shock $p-1$ times continuously differentiable, this strategy does not remove the discontinuous nature of the shock. In that sense, the underlying physics is better represented.

4.4 Position-dependent post-processor

The (combined) one-sided post-processor can be applied in the entire domain, including near boundaries and shocks. To enhance the smoothness and accuracy of this technique, in this section, we propose the position-dependent post-processor.

Section 4.4.1 generalizes the original one-sided post-processor by relaxing both the number and the position of the kernel nodes. Section 4.4.2 uses this generalization to introduce the position-dependent post-processor. Section 4.4.3 describes the application of a post-processor for two-dimensional problems.

4.4.1 Generalized post-processor

To improve the one-sided post-processor, the first step is to generalize it by relaxing both the number and the position of the kernel nodes. More specifically, we drop the convention that the kernel must be based on $2p + 1$ kernel nodes, and use $r + 1$ B-splines instead (where r is any positive integer at this point). Furthermore, we allow the kernel nodes to take real values, instead of the original approach to consider integers only. This makes it possible to render the shift function, and thus the overall technique, smooth.

Altogether, the generalized post-processor is obtained by considering the original post-processor in Definition 4.3.2, but then using $r + 1$ kernel nodes and removing the rounding operators in the shift function (4.10):

Definition 4.4.1 (Generalized post-processor) Consider a DG approximation u_h at some final simulation time on the interval $[a, b]$, using mesh elements of size h and polynomials of degree p, as discussed in Section 4.2.1. Let $\bar{x}$ denote an evaluation point in (a, b). Depending on this evaluation point, define the following $r + 1$ kernel nodes:

$$x_j = -\frac{r}{2} + j + \lambda(\bar{x}), \qquad \text{for all } j = 0, ..., r, \tag{4.11}$$

$$\lambda(\bar{x}) = \begin{cases} \min\{0, -\frac{r+p+1}{2} + \frac{\bar{x}-a}{h}\}, & \text{for } \bar{x} \in [a, \frac{a+b}{2}), \\ \max\{0, \frac{r+p+1}{2} + \frac{\bar{x}-b}{h}\}, & \text{for } \bar{x} \in [\frac{a+b}{2}, b]. \end{cases} \tag{4.12}$$

Let K be the result of substituting these nodes into (4.5) and (4.6) (using r rather than $2p$ in the upper bound of the summation). Then, the result of post-processing u_h in $\bar{x}$ is obtained by computing the convolution with the scaled kernel K as in (4.7).

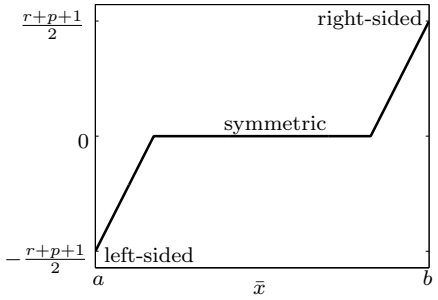


FIGURE 4.5
Illustration of the ***modified shift function*** $\lambda(\bar{x})$ *in* (4.12). *Unlike the original shift function (cf. Figure 4.4), it no longer contains discontinuities.*

The modified shift function in (4.12) is illustrated in Figure 4.5. Compared to the former shift function (cf. Figure 4.4), there are two main advantages: the first is that its values are closer to zero. As a consequence, the resulting kernels are 'more symmetric' and thus more accurate (cf. Section 4.3.3).

The second advantage is that the modified shift function is now continuous everywhere. Because the filtered DG solution is as smooth as the shift function (but typically not smoother than the B-splines), the generalized post-processor yields a better and more realistic smoothness than before.

We remark that the smoothness could be enhanced further by designing a shift function that has the same smoothness of the B-splines throughout the entire domain (note that the present version is not differentiable in two locations). However, such an alternative shift function would be less close to zero, resulting in kernels that are 'less symmetric' and thus less accurate.

It will be shown in Chapter 5 that, similar to the to the symmetric post-processor, the generalized post-processor can extract DG superconvergence of

order $2p + 1$.

4.4.2 Position-dependent post-processor

The generalized post-processor can be seen as a variant of the original one-sided post-processor that yields better smoothness for an arbitrary number of kernel nodes. To enhance the accuracy as well, we now combine two generalized post-processors to incorporate extra kernel nodes in the non-symmetric kernels near the boundary. The resulting technique is called the position-dependent post-processor, and aims to increase the accuracy near the boundary to that established by the symmetric kernel in the interior.

To specify this method, we apply the generalized post-processor twice: once using $2p + 1$ kernel nodes and once using $4p + 1$ kernel nodes (this particular choice is motivated below). After that, we compute a 'smooth' convex combination of these two results to select the extra kernel nodes near the boundary only, while maintaining the enhanced smoothness. Altogether, we arrive at the following definition:

Definition 4.4.2 (Position-dependent post-processor) Consider a DG approximation u_h at some final simulation time on the interval $[a, b]$, using mesh elements of size h, and polynomials of degree p, as discussed in Section 4.2.1. Let $\bar{x}$ denote an evaluation point in (a, b). For this evaluation point, suppose that $u^\star_{h,2p+1}(\bar{x})$ and $u^\star_{h,4p+1}(\bar{x})$ are the result of applying the generalized post-processor (Definition 4.4.1) using $2p + 1$ and $4p + 1$ kernel nodes respectively. Furtermore, let $\theta : [a, b] \to [0, 1]$ be a ('smooth') function that is equal to 1 in the interior and equal to 0 near the boundary. Then, we set

$$u^\star_h(\bar{x}) = \theta(\bar{x})u^\star_{h,2p+1}(\bar{x}) + \big(1 - \theta(\bar{x})\big)u^\star_{h,4p+1}(\bar{x}).$$

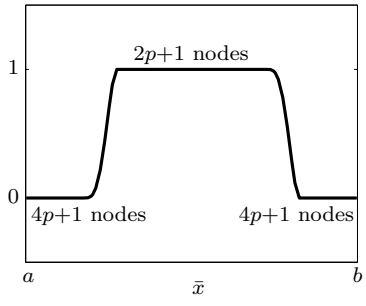


FIGURE 4.6
Illustration of the ***coefficient function*** $\theta(\bar{x})$*, which smoothly selects extra kernel nodes near the boundary for higher accuracy.*

The purpose of the coefficient function θ is to select the appropriate number of kernel nodes, depending on the evaluation point. An illustration of a specific

choice for θ is given in Figure 4.6. Near the boundary, we prefer to use extra nodes to enhance the accuracy of the non-symmetric kernels. Hence, $\theta = 0$ in this region[3], to select $u^\star_{h,4p+1}$ rather than $u^\star_{h,2p+1}$. In the interior, the symmetric kernel already provides the desired accuracy without the additional costs for the use of extra B-splines (costs are discussed further below). Hence, $\theta = 1$ in this region, to select $u^\star_{h,2p+1}$ rather than $u^\star_{h,4p+1}$. Between the boundary regions and the interior, there are two transition regions, where θ varies *smoothly* between 0 and 1 to avoid that artificial discontinuities are introduced[4].

It will be shown in Chapter 5 that, similar to the to the generalized post-processor, the position-dependent post-processor can extract DG superconvergence of order $2p + 1$.

There are several remarks to be made with respect to the position-dependent post-processor:

- The symmetric and position-dependent post-processor are based on the same building blocks. For this reason, we speculate that suitable application types for the symmetric post-processor (away from non-periodic boundaries and shocks) are also suitable for the position-dependent post-processor (but now in the entire domain). This would include the unstructured and non-linear problems mentioned at the end of Section 4.3.2, but verification of the latter is left for future research. One- and two-dimensional linear problems on uniform meshes will be studied in Section 4.5.

- It is not needed to compute both $u^\star_{h,2p+1}$ and $u^\star_{h,4p+1}$ in the entire domain. This is only required in the two small transition regions where $\theta \in (0, 1)$. Near the boundary, we only need to compute $u^\star_{h,4p+1}$. In the interior, we only require $u^\star_{h,2p+1}$.

- Similar to the (combined) one-sided post-processor, we can apply the position-dependent post-processor near shocks, as discussed in Section 4.3.3.

- The use of extra kernel nodes increases the kernel support, and thus the computational costs. Using $r+1$ B-splines of order $p+1$, the convolution with the kernel can be implemented using small matrix-vector multiplications. These matrices have size $(r + 1) \times (p + 1)$ and contain the inner products of the B-splines with the DG basis functions. The vectors (of length $p + 1$) contain the DG modes for a mesh element in the kernel support. Altogether, to compute $u^\star_{h,r+1}$ in an evaluation point, $r + p + 2$ such matrix-vector multiplications are required, plus the summation of all

[3] In this example, the boundary region where $\theta = 0$ is $\frac{3p+1}{2}$ mesh elements wide, i.e. the smallest region where the symmetric kernel with $2p + 1$ nodes cannot be applied.

[4] In this example, the transition regions are two mesh elements wide. In these regions, θ is polynomial of degree $2p + 1$ such that θ is $p + 1$ times differentiable.

elements in the resulting vectors. We stress that the post-processor is applied only once, at the final simulation time. To keep the computational time at a minimum, extra kernel nodes should only be applied where necessary. For a more efficient inexact implementation, see [49, 47, 50].

- Instead of $4p+1$, we could use any number of kernel nodes near the boundary (provided that the convolution remains well-defined). For example, we also ran tests using $3p+1$ and $5p+1$ kernel nodes. As expected, we found that a larger number of kernel nodes leads to more accuracy. However, the difference between using $5p+1$ nodes and $4p+1$ nodes was relatively small. For $3p+1$ nodes, the errors were not improved over the unfiltered errors for the coarser meshes. Based on these experiments, using $4p+1$ kernel nodes is a natural choice for enhancing the errors where necessary, without increasing the kernel support and the computational costs too much.

- For coarse meshes, the use of extra kernel nodes can increase the support of $K\left(\frac{\bar{x}-\cdot}{h}\right)$ such that it is no longer contained in the domain $[a,b]$ (no matter how you shift it). This is the case if $(r+p+1)h$ becomes larger than $b-a$. To be able to apply the post-processor anyway, a possible solution is to scale the kernel by the smaller scaling

$$H = \frac{b-a}{r+p+1} < h \tag{4.13}$$

rather than h in (4.7). This shrinks the kernel support appropriately. It will be demonstrated in Section 4.5 that the use of a smaller scaling for the coarser meshes can reduce the accuracy-enhancing quality of the post-processor. For this reason, the alternative scaling should only be applied when necessary.

4.4.3 Post-processing in two dimensions

All post-processing techniques discussed in this chapter can also be applied for higher-dimensional problems using tensor products. In this section, we illustrate this for a square domain $\Omega = [a_1, b_1] \times [a_2, b_2]$ (for higher-dimensional problems, cf. Section 5.2.1).

To this end, consider an evaluation point $(\bar{x}_1, \bar{x}_2) \in (a_1, b_1) \times (a_2, b_2)$. Let K_1 denote the one-dimensional kernel for the evaluation point $\bar{x}_1 \in (a_1, b_1)$, corresponding to any of the post-processors discussed in the previous sections: symmetric, (combined) one-sided, generalized, or position-dependent[5]. Similarly, let K_2 denote such a kernel for $\bar{x}_2 \in (a_2, b_2)$. Note that K_1 and K_2 are typically not the same kernel, as they are based on different evaluation points.

[5]For the position-dependent post-processor, this means we take the convex combination of the two generalized kernels involved, i.e. $\theta(\bar{x}_1)K_{2p+1} + (1-\theta(\bar{x}_1))K_{4p+1}$.

Using this notation, a DG approximation u_h on the square domain Ω (cf. Section 4.2.2) can now be post-processed by computing the convolution against the tensor product of the two scaled kernels:

$$u_h^\star(\bar{x}_1, \bar{x}_1) = \frac{1}{h^2} \int_{a_1}^{b_1} \int_{a_2}^{b_2} K_1\left(\frac{\bar{x}_1 - x_1}{h}\right) K_2\left(\frac{\bar{x}_2 - x_2}{h}\right) u_h(x_1, x_2)\,\mathrm{d}x_2\,\mathrm{d}x_1.$$

If u_h is vector-valued, then this strategy can be applied to each individual component.

4.5 Numerical Results

The previous section proposed a position-dependent post-processor to improve on the accuracy of the original (combined) one-sided post-processor. In this section, we compare both techniques in terms of numerical experiments (for a theoretical study, cf. Chapter 5).

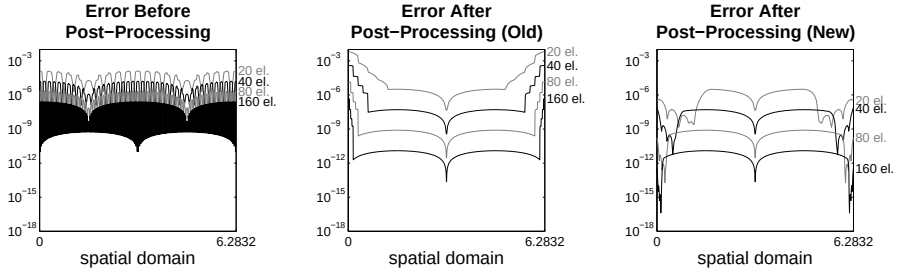
In particular, we consider the L^2-projection of a sine function (Section 4.5.1); four one-dimensional hyperbolic problems, including periodic boundary conditions (Section 4.5.2), Dirichlet boundary conditions (Section 4.5.3), variable coefficients (Section 4.5.4), and two stationary shocks (Section 4.5.5); a two-dimensional system (Section 4.5.6); and the two-dimensional test cases studied in [73], including streamline visualizations (Section 4.5.7).

In our implementation, the DG approximations are based on first order upwind fluxes, monomial basis functions, and uniform meshes, as discussed in Section 4.2. For the time-discretization, we use a third-order SSP-RK scheme [36, 37] with a sufficiently small time step to ensure that the time-stepping errors are not dominating the spatial errors. We apply both the 'old' and the 'new' post-processor at the final time $T = 12.5$, as specified in Definition 4.3.2 and Definition 4.4.2 respectively. Convolutions of the form (4.7) are computed exactly using Gaussian quadrature [64]. Finally, we make use of the ARPREC multi-precision package to reduce round-off errors appropriately [8].

4.5.1 L^2-Projection

The first test case is the L^2-projection of $u(x) = \sin(x)$ onto the space of piecewise polynomials of degree $p = 1, 2, 3$ on the domain $[0, 2\pi]$. This test case can also be interpreted as a DG approximation at the initial time. It is the most elementary case that we can use to test the reliability of our filter.

Table 4.1 demonstrates that both post-processors enhance the convergence rate from $O(h^{p+1})$ to $O(h^{2p+1})$. However, both the orders and the magnitude of the L^2- and L^∞-errors are better for the new post-processor. More importantly, it yields an improvement over the unfiltered errors in both norms, even for coarse meshes. The plots (for $p = 2$) illustrate that both post-processors produce identical results in the interior of the domain, where they apply the same symmetric kernel with $2p + 1$ nodes. The differences occur at the boundary: the new post-processor yields significantly better results without introducing a spurious stair-stepping effect. This can be explained by the use of extra kernel nodes and the continuity of the new shift function (cf. Section 4.4). As a result, unlike before, the new post-processor enhances the accuracy of the DG approximation in the entire domain.



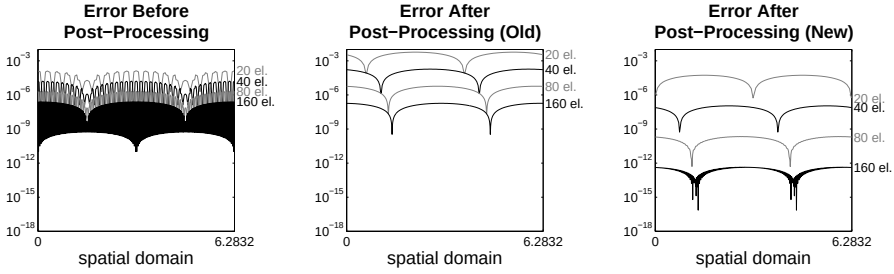
	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
20	6.51e-03	-	5.95e-03	-	1.60e-02	-	2.21e-02	-	4.88e-04	-	1.26e-03	-
40	1.63e-03	2.00	1.50e-03	1.99	1.71e-03	3.22	3.11e-03	2.83	1.90e-05	4.68	5.35e-05	4.56
80	4.07e-04	2.00	3.76e-04	2.00	1.58e-04	3.43	4.00e-04	2.96	9.02e-07	4.40	1.79e-06	4.90
160	1.02e-04	2.00	9.40e-05	2.00	1.42e-05	3.48	5.03e-05	2.99	5.33e-08	4.08	5.69e-08	4.98
Polynomial Degree p = 2												
20	1.73e-04	-	1.28e-04	-	3.95e-03	-	6.68e-03	-	4.19e-06	-	3.14e-06	-
40	2.16e-05	3.00	1.61e-05	2.99	2.11e-04	4.23	3.92e-04	4.09	8.69e-08	5.59	6.71e-08	5.55
80	2.70e-06	3.00	2.02e-06	3.00	5.47e-06	5.27	1.39e-05	4.82	1.38e-09	5.97	7.87e-10	6.41
160	3.38e-07	3.00	2.53e-07	3.00	1.26e-07	5.45	4.46e-07	4.96	2.17e-11	5.99	1.23e-11	6.00
Polynomial Degree p = 3												
20	3.42e-06	-	2.15e-06	-	1.06e-04	-	2.26e-04	-	3.75e-07	-	9.84e-07	-
40	2.14e-07	4.00	1.35e-07	3.99	4.71e-06	4.49	8.96e-06	4.66	6.30e-10	9.22	3.89e-10	11.31
80	1.34e-08	4.00	8.49e-09	4.00	3.41e-08	7.11	8.72e-08	6.68	2.67e-12	7.88	1.53e-12	7.99
160	8.36e-10	4.00	5.31e-10	4.00	2.00e-10	7.41	7.16e-10	6.93	1.06e-14	7.98	5.97e-15	8.00

TABLE 4.1. *L^2-projection*

Isolating the boundary effects To isolate the effect of extra kernel nodes at the boundary, we revisit our previous test case and apply the left-sided post-processor in the entire domain for $2p+1$ ('old') and $4p+1$ ('new') kernel nodes (using a periodic extension of u_h when needed). In other words, the only difference between the two post-processors is now the number of kernel nodes.

Table 4.2 demonstrates that the use of $4p+1$ kernel nodes leads to better accuracy of $O(h^{4p+1})$. The same phenomenon occurs at the boundary in Table 4.1, which explains why the new post-processor improves the errors in that region.

We stress that the accuracy of $O(h^{4p+1})$ cannot be expected in general. Based on the analysis in Section 5.5.3 later on, the theoretical error takes the form $C_1 h^{4p+1} + C_2 h^{2p+1}$, where the order in the first term is equal to the number of kernel nodes (and $C_1, C_2 > 0$ are constant with respect to h). We speculate that the meshes in this test case are sufficiently coarse so that the first error dominates the second. For finer meshes, it is likely that the second error will start to dominate, so that the error then becomes $O(h^{2p+1})$. We will encounter such an example in the next section.



	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
20	6.51e-03	-	5.95e-03	-	4.50e-02	-	2.54e-02	-	3.72e-03	-	2.10e-03	-
40	1.63e-03	2.00	1.50e-03	1.99	5.70e-03	2.98	3.22e-03	2.98	1.18e-04	4.97	6.71e-05	4.97
80	4.07e-04	2.00	3.76e-04	2.00	7.15e-04	3.00	4.03e-04	3.00	3.72e-06	4.99	2.11e-06	4.99
160	1.02e-04	2.00	9.40e-05	2.00	8.94e-05	3.00	5.05e-05	3.00	1.17e-07	5.00	6.62e-08	4.99
Polynomial Degree p = 2												
20	1.73e-04	-	1.28e-04	-	1.03e-02	-	5.79e-03	-	9.52e-05	-	5.37e-05	-
40	2.16e-05	3.00	1.61e-05	2.99	3.29e-04	4.97	1.85e-04	4.96	1.93e-07	8.95	1.09e-07	8.95
80	2.70e-06	3.00	2.02e-06	3.00	1.03e-05	4.99	5.83e-06	4.99	3.81e-10	8.98	2.16e-10	8.97
160	3.38e-07	3.00	2.53e-07	3.00	3.23e-07	5.00	1.82e-07	5.00	7.61e-13	8.97	4.42e-13	8.93
Polynomial Degree p = 3												
20	3.42e-06	-	2.15e-06	-	2.58e-03	-	1.46e-03	-	2.82e-06	-	1.59e-06	-
40	2.14e-07	4.00	1.35e-07	3.99	2.09e-05	6.95	1.18e-05	6.95	3.62e-10	12.93	2.04e-10	12.93
80	1.34e-08	4.00	8.49e-09	4.00	1.65e-07	6.99	9.29e-08	6.99	4.47e-14	12.98	2.53e-14	12.98
160	8.36e-10	4.00	5.31e-10	4.00	1.29e-09	7.00	7.28e-10	7.00	5.51e-18	12.99	3.21e-18	12.94

TABLE 4.2. *L^2-projection: isolating boundary effects*

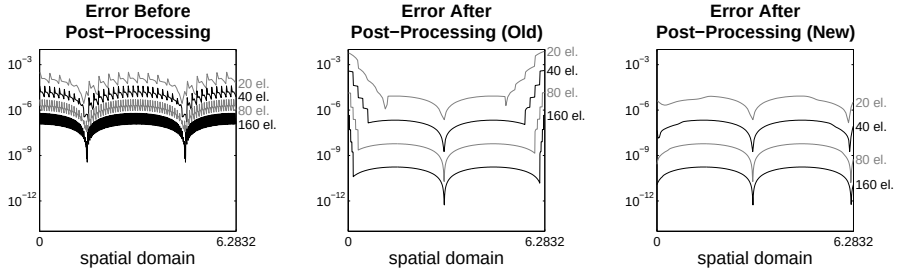
4.5.2 Constant coefficients

We now consider a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions:

$$u_t + u_x = 0, \qquad x \in [0, 2\pi], \qquad t \in [0, 12.5].$$

The initial condition is chosen such that the exact solution reads $u(x,t) = \sin(x-t)$. For $t = 0$, this test case is equivalent to the one discussed in the previous section. Here, we consider the final time $t = 12.5$. This test case is more challenging than the previous one because u_h now contains information of the physics of the PDE and the numerics of the DG method.

Nevertheless, Table 4.3 demonstrates that the results are similar to those for the L^2-projection: we are able to obtain better errors than the DG solution and the old post-processor. In fact, the magnitude of the errors is improved throughout the entire domain when the new position-dependent post-processor is applied to the DG approximation. Furthermore, the convergence rate is improved from $O(h^{p+1})$ to $O(h^{2p+1})$.



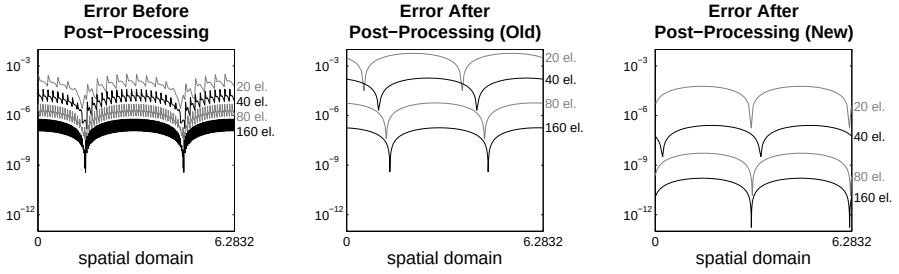
	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
20	1.41e-02	-	1.02e-02	-	1.89e-02	-	2.21e-02	-	9.60e-03	-	5.44e-03	-
40	2.91e-03	2.28	2.69e-03	1.92	2.10e-03	3.17	3.12e-03	2.82	1.20e-03	3.00	6.78e-04	3.00
80	6.81e-04	2.09	7.57e-04	1.83	2.18e-04	3.27	4.03e-04	2.95	1.50e-04	3.00	8.45e-05	3.00
160	1.67e-04	2.03	2.00e-04	1.92	2.34e-05	3.22	5.10e-05	2.98	1.87e-05	3.00	1.05e-05	3.00
Polynomial Degree p = 2												
20	2.68e-04	-	3.18e-04	-	4.00e-03	-	7.50e-03	-	1.30e-05	-	8.41e-06	-
40	3.35e-05	3.00	3.98e-05	3.00	2.11e-04	4.25	4.07e-04	4.20	3.77e-07	5.11	2.16e-07	5.28
80	4.19e-06	3.00	4.97e-06	3.00	5.46e-06	5.27	1.41e-05	4.85	1.06e-08	5.16	5.97e-09	5.18
160	5.24e-07	3.00	6.22e-07	3.00	1.25e-07	5.45	4.49e-07	4.97	3.09e-10	5.10	1.74e-10	5.10
Polynomial Degree p = 3												
20	5.18e-06	-	4.40e-06	-	1.30e-04	-	3.21e-04	-	3.76e-07	-	1.05e-06	-
40	3.24e-07	4.00	2.76e-07	4.00	4.71e-06	4.79	9.45e-06	5.09	6.63e-10	9.15	4.09e-10	11.32
80	2.02e-08	4.00	1.72e-08	4.00	3.41e-08	7.11	8.91e-08	6.73	2.96e-12	7.81	1.69e-12	7.92
160	1.26e-09	4.00	1.08e-09	4.00	2.00e-10	7.41	7.23e-10	6.95	1.29e-14	7.84	7.28e-15	7.86

TABLE 4.3. *Constant coefficients*

Isolating boundary effects Similar to Section 4.5.1, we isolate the effect of extra kernel nodes near the boundary by repeating the experiment while applying the left-sided post-processor throughout the entire domain.

The results are displayed in Table 4.4. As before (cf. Table 4.2), the extra nodes yield faster convergence and smaller errors. This explains the higher accuracy of the position-dependent post-processor near the boundary in Table 4.3.

Unlike before, the convergence rate is typically lower than $O(h^{4p+1})$, and it drops to $O(h^{2p+1})$ for finer meshes. This change in the convergence rate is in line with our earlier speculation that the lower order component of the error, which is $O(h^{2p+1})$, starts to dominate the higher order component for sufficiently fine meshes.



	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
20	1.41e-02	-	1.02e-02	-	4.19e-02	-	2.36e-02	-	1.22e-02	-	6.87e-03	-
40	2.91e-03	2.28	2.69e-03	1.92	5.57e-03	2.91	3.14e-03	2.91	1.24e-03	3.30	6.98e-04	3.30
80	6.81e-04	2.09	7.57e-04	1.83	7.15e-04	2.96	4.03e-04	2.96	1.50e-04	3.05	8.45e-05	3.05
160	1.67e-04	2.03	2.00e-04	1.92	9.04e-05	2.98	5.10e-05	2.98	1.86e-05	3.01	1.05e-05	3.01
Polynomial Degree p = 2												
20	2.68e-04	-	3.18e-04	-	1.03e-02	-	5.79e-03	-	1.05e-04	-	5.90e-05	-
40	3.35e-05	3.00	3.98e-05	3.00	3.29e-04	4.97	1.85e-04	4.97	4.49e-07	7.86	2.53e-07	7.86
80	4.19e-06	3.00	4.97e-06	3.00	1.03e-05	4.99	5.83e-06	4.99	9.34e-09	5.59	5.27e-09	5.59
160	5.24e-07	3.00	6.22e-07	3.00	3.23e-07	5.00	1.82e-07	5.00	2.88e-10	5.02	1.62e-10	5.02
Polynomial Degree p = 3												
20	5.18e-06	-	4.40e-06	-	2.58e-03	-	1.46e-03	-	2.82e-06	-	1.59e-06	-
40	3.24e-07	4.00	2.76e-07	4.00	2.09e-05	6.95	1.18e-05	6.95	3.96e-10	12.80	2.24e-10	12.80
80	2.02e-08	4.00	1.72e-08	4.00	1.65e-07	6.99	9.29e-08	6.99	3.16e-13	10.29	1.78e-13	10.29
160	1.26e-09	4.00	1.08e-09	4.00	1.29e-09	7.00	7.28e-10	7.00	2.32e-15	7.09	1.31e-15	7.09

TABLE 4.4. *Constant coefficients: isolating boundary effects*

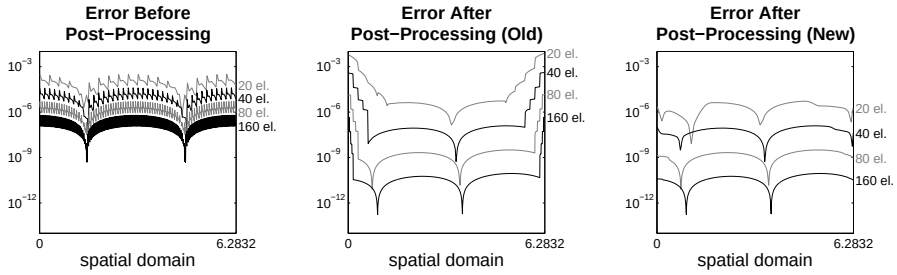
4.5.3 Dirichlet BCs

The previous section discussed a test case with periodic boundary conditions. For those problems, it is actually not necessary to use a one-sided approach near the boundary: the more accurate symmetric post-processor could be applied by using a periodic extension of the DG solution. However, in most real-life applications, the boundary conditions are not periodic. For this reason, we revisit the test case of the previous section, but now using Dirichlet boundary conditions. That is,

$$u_t + u_x = 0, \qquad u(0,t) = \sin(-t), \qquad x \in [0, 2\pi], \qquad t \in [0, 12.5].$$

The initial condition is chosen such that the exact solution reads $u(x,t) = \sin(x-t)$.

The results are displayed in Table 4.5: similar to the periodic case, we observe that the convergence rate is improved from $O(h^{p+1})$ to $O(h^{2p+1})$. Furthermore, the smoothness and the accuracy are improved in the entire domain, including the boundary.



	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
20	1.10e-02	-	1.29e-02	-	1.63e-02	-	2.33e-02	-	3.37e-03	-	2.41e-03	-
40	2.68e-03	2.03	3.29e-03	1.97	1.76e-03	3.22	3.23e-03	2.85	4.14e-04	3.02	3.00e-04	3.01
80	6.67e-04	2.01	8.32e-04	1.98	1.66e-04	3.40	4.13e-04	2.97	5.13e-05	3.01	3.72e-05	3.01
160	1.66e-04	2.00	2.09e-04	1.99	1.55e-05	3.42	5.19e-05	2.99	6.39e-06	3.01	4.63e-06	3.01
Polynomial Degree p = 2												
20	2.68e-04	-	3.17e-04	-	4.00e-03	-	7.50e-03	-	6.98e-06	-	5.23e-06	-
40	3.35e-05	3.00	3.98e-05	2.99	2.11e-04	4.25	4.07e-04	4.20	1.84e-07	5.25	1.24e-07	5.40
80	4.19e-06	3.00	4.97e-06	3.00	5.46e-06	5.27	1.41e-05	4.85	4.63e-09	5.31	3.16e-09	5.29
160	5.24e-07	3.00	6.22e-07	3.00	1.25e-07	5.45	4.49e-07	4.97	1.28e-10	5.18	8.83e-11	5.16
Polynomial Degree p = 3												
20	5.18e-06	-	4.40e-06	-	1.30e-04	-	3.21e-04	-	3.75e-07	-	1.05e-06	-
40	3.24e-07	4.00	2.76e-07	4.00	4.71e-06	4.79	9.45e-06	5.09	6.39e-10	9.20	3.97e-10	11.37
80	2.02e-08	4.00	1.72e-08	4.00	3.41e-08	7.11	8.91e-08	6.73	2.75e-12	7.86	1.59e-12	7.96
160	1.26e-09	4.00	1.08e-09	4.00	2.00e-10	7.41	7.23e-10	6.95	1.12e-14	7.94	6.48e-15	7.94

TABLE 4.5. *Dirichlet BCs*

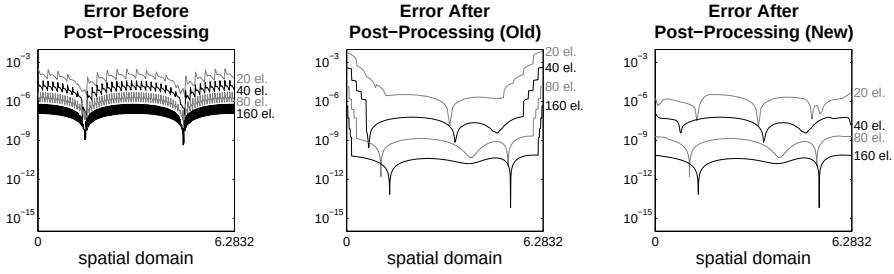
4.5.4 Variable coefficients

As a first step towards nonlinear problems, we now consider a linear problem with smoothly varying coefficients:

$$u_t + (a\,u)_x = f, \qquad a(x,t) = 2 + \sin(x+t), \qquad x \in [0, 2\pi], \qquad t \in [0, 12.5].$$

The boundary conditions are periodic, and the initial condition and the forcing term $f(x,t)$ are chosen such that the exact solution reads $u(x,t) = \sin(x-t)$.

The results are displayed in Table 4.6. Similar to all of the previous test cases, we observe that the convergence rate is improved from from $O(h^{p+1})$ to $O(h^{2p+1})$. Furthermore, the smoothness and the accuracy are improved in the entire domain, including the boundary.



	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
20	1.09e-02	-	1.46e-02	-	1.63e-02	-	2.47e-02	-	2.75e-03	-	2.95e-03	-
40	2.68e-03	2.03	3.53e-03	2.05	1.75e-03	3.22	3.38e-03	2.87	3.48e-04	2.98	2.77e-04	3.41
80	6.66e-04	2.01	8.62e-04	2.03	1.64e-04	3.41	4.31e-04	2.97	4.38e-05	2.99	2.99e-05	3.21
160	1.66e-04	2.00	2.13e-04	2.02	1.52e-05	3.44	5.40e-05	3.00	5.50e-06	3.00	3.63e-06	3.04
Polynomial Degree p = 2												
20	2.68e-04	-	3.31e-04	-	4.00e-03	-	7.50e-03	-	4.58e-06	-	4.67e-06	-
40	3.35e-05	3.00	4.07e-05	3.03	2.11e-04	4.25	4.07e-04	4.21	1.01e-07	5.50	1.18e-07	5.31
80	4.19e-06	3.00	5.03e-06	3.02	5.46e-06	5.27	1.41e-05	4.85	2.77e-09	5.19	2.15e-09	5.78
160	5.24e-07	3.00	6.25e-07	3.01	1.25e-07	5.45	4.49e-07	4.97	9.81e-11	4.82	7.36e-11	4.87
Polynomial Degree p = 3												
20	5.17e-06	-	4.41e-06	-	1.30e-04	-	3.21e-04	-	1.11e-05	-	3.91e-05	-
40	3.23e-07	4.00	2.76e-07	4.00	4.71e-06	4.79	9.45e-06	5.09	6.63e-10	14.03	1.12e-09	15.09
80	2.02e-08	4.00	1.73e-08	4.00	3.41e-08	7.11	8.91e-08	6.73	2.65e-12	7.97	1.53e-12	9.51
160	1.26e-09	4.00	1.08e-09	4.00	2.00e-10	7.41	7.23e-10	6.95	1.06e-14	7.97	7.13e-15	7.75

TABLE 4.6. *Variable coefficients*

4.5.5 Discontinuous coefficients

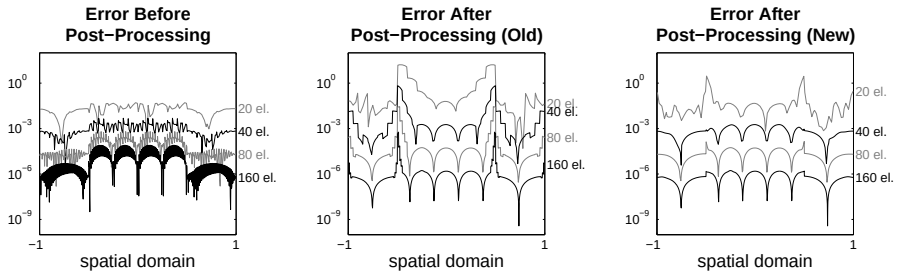
For all of the previous test cases, the exact solution is infinitely smooth. In Section 4.4.2, we emphasized that the position-dependent post-processor can also be applied near a shock, by treating it as a boundary. To test this numerically, we now consider a problem with discontinuous coefficients [64]:

$$u_t + (a\,u)_x = 0, \quad a(x) = \begin{cases} \frac{1}{2}, & x \in [-\frac{1}{2}, \frac{1}{2}], \\ 1, & \text{else}, \end{cases}, \quad x \in [-1, 1], \quad t \in [0, 12.5].$$

The boundary conditions are periodic, and the initial condition is chosen such that the exact solution has two stationary shocks:

$$u(x,t) = \begin{cases} -2\cos\big(4\pi(x - \frac{1}{2}t)\big), & x \in [-\frac{1}{2}, \frac{1}{2}], \\ \cos\big(2\pi(x-t)\big), & \text{else}. \end{cases}$$

The results are displayed in Table 4.7. The accuracy of the new post-processor is better than that of the DG solution as long as the mesh is sufficiently fine. For the coarser meshes, the lower accuracy is because a kernel



	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
20	1.21e+00	-	1.56e+00	-	1.15e+00	-	1.54e+00	-	1.20e+00	-	1.62e+00	-
40	2.72e-01	2.15	3.77e-01	2.05	2.54e-01	2.18	3.49e-01	2.14	2.74e-01	2.13	4.33e-01	1.91
80	3.83e-02	2.83	5.74e-02	2.71	3.63e-02	2.80	4.88e-02	2.84	3.75e-02	2.87	5.02e-02	3.11
160	5.20e-03	2.88	8.62e-03	2.74	4.70e-03	2.95	6.19e-03	2.98	4.75e-03	2.98	6.17e-03	3.03
Polynomial Degree p = 2												
20	3.65e-02	-	5.14e-02	-	6.81e+00	-	1.62e+01	-	5.71e-01	-	2.94e+00	-
40	2.05e-03	4.15	4.84e-03	3.41	1.67e-01	5.35	6.78e-01	4.58	1.25e-03	8.84	1.83e-03	10.66
80	2.17e-04	3.24	6.27e-04	2.95	6.03e-03	4.79	2.79e-02	4.60	4.16e-05	4.91	1.40e-04	3.71
160	2.68e-05	3.02	7.94e-05	2.98	8.41e-05	6.16	5.79e-04	5.59	1.18e-06	5.14	1.69e-06	6.37
Polynomial Degree p = 3												
20	1.08e-03	-	2.45e-03	-	3.58e+00	-	1.25e+01	-	2.27e-01	-	6.61e-01	-
40	6.60e-05	4.04	1.37e-04	4.16	1.87e-02	7.58	9.95e-02	6.97	2.64e-03	6.43	1.85e-02	5.16
80	4.13e-06	4.00	8.74e-06	3.97	6.50e-04	4.84	2.92e-03	5.09	5.20e-06	8.99	6.98e-05	8.05
160	2.58e-07	4.00	5.51e-07	3.99	2.62e-06	7.95	1.77e-05	7.36	4.67e-09	10.12	8.70e-08	9.65

TABLE 4.7. *Discontinuous coefficients*

scale smaller than h is required to ensure that the support of the (scaled) kernel fits between two subsequent boundaries/shocks (cf. Section 4.4.2, (4.13)).

Nevertheless, for $p = 2$ and $p = 3$, the magnitude of the errors is much smaller for the new post-processor than for the old one. For $p = 1$, this is not the case: the errors are slightly worse. We speculate that this is due to the fact that *more* extra kernel nodes are used for $p = 2$ than for $p = 1$. Improvement of this issue is left for future research.

4.5.6 Two-dimensional system

In the previous sections, we considered several one-dimensional problems. However, higher-dimensional fields can also be filtered, as discussed in Section 4.4.3. In this section, we apply such a strategy for the following two-dimensional system:

$$\begin{bmatrix} u \\ v \end{bmatrix}_t + \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_x + \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_y = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad (x, y) \in [0, 2\pi]^2, \quad t \in [0, 12.5].$$

The boundary conditions are periodic, and the initial condition is given by

$$u_0(x, y) = \frac{1}{2\sqrt{2}}(\sin(x + y) - \cos(x + y)),$$
$$v_0(x, y) = \frac{1}{2\sqrt{2}}\big((\sqrt{2} - 1)\sin(x + y) + (1 + \sqrt{2})\cos(x + y)\big).$$

The results for this test case are displayed in Table 4.8 for a final time of $t = 12.5$. Similar to the one-dimensional problems, we observe that the convergence rate is improved from $O(h^{p+1})$ to $O(h^{2p+1})$. Furthermore, unlike the original filter, the position-dependent post-processor improves the DG errors in the entire domain, even for coarse meshes. In other words, the results for this two-dimensional problem are similar to those for the previous smooth one-dimensional cases.

4.5.7 Two-dimensional streamlines

Next, we study the two-dimensional tests of Steffen et al. [73], including streamline visualizations. In each test, a velocity profile (u, v) on the square $[-1, 1]^2$ is given. We consider the L^2-projection of that solution onto the space of piecewise polynomials of degree $p = 1, 2, 3$ (as before, this is similar to a DG approximation at the initial time). The velocity (u, v) is obtained as a function of (x, y) from the real and imaginary parts of a complex number r:

$$u := \mathrm{Re}(r), \qquad v := \mathrm{Im}(r),$$

u-component

	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
$\mathbf{20^2}$	1.22e-01	-	3.94e-02	-	1.16e-01	-	2.77e-02	-	1.18e-01	-	2.73e-02	-
$\mathbf{40^2}$	1.77e-02	2.78	7.12e-03	2.47	1.55e-02	2.90	4.58e-03	2.60	1.54e-02	2.93	3.48e-03	2.97
$\mathbf{80^2}$	2.94e-03	2.59	1.38e-03	2.36	1.96e-03	2.98	6.33e-04	2.85	1.95e-03	2.98	4.39e-04	2.99
Polynomial Degree p = 2												
$\mathbf{20^2}$	1.58e-03	-	1.24e-03	-	1.96e-02	-	1.77e-02	-	3.33e-04	-	1.03e-04	-
$\mathbf{40^2}$	1.95e-04	3.02	1.62e-04	2.94	4.29e-04	5.51	6.00e-04	4.88	1.01e-05	5.05	2.29e-06	5.50
$\mathbf{80^2}$	2.44e-05	3.00	2.05e-05	2.98	9.46e-06	5.50	1.76e-05	5.09	3.12e-07	5.01	7.03e-08	5.03
Polynomial Degree p = 3												
$\mathbf{20^2}$	7.87e-05	-	5.58e-05	-	2.00e-03	-	1.79e-03	-	1.33e-06	-	1.93e-06	-
$\mathbf{40^2}$	4.98e-06	3.98	3.30e-06	4.08	1.10e-05	7.51	1.54e-05	6.86	6.87e-09	7.60	1.69e-09	10.16
$\mathbf{80^2}$	3.11e-07	4.00	1.97e-07	4.06	6.07e-08	7.50	1.17e-07	7.05	5.02e-11	7.10	1.16e-11	7.19

v-component

	Before Post-Processing				After Post-Processing (Old)				After Post-Processing (New)			
mesh	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order	L^2-error	order	L^∞-error	order
Polynomial Degree p = 1												
$\mathbf{20^2}$	1.43e-01	-	3.89e-02	-	1.36e-01	-	3.65e-02	-	1.39e-01	-	3.21e-02	-
$\mathbf{40^2}$	2.04e-02	2.81	6.13e-03	2.67	1.81e-02	2.91	5.42e-03	2.75	1.80e-02	2.95	4.07e-03	2.98
$\mathbf{80^2}$	3.31e-03	2.62	1.63e-03	1.91	2.27e-03	2.99	7.23e-04	2.91	2.26e-03	2.99	5.09e-04	3.00
Polynomial Degree p = 2												
$\mathbf{20^2}$	2.22e-03	-	2.45e-03	-	2.16e-02	-	1.43e-02	-	3.80e-04	-	1.18e-04	-
$\mathbf{40^2}$	2.72e-04	3.03	3.07e-04	3.00	4.89e-04	5.46	6.64e-04	4.43	1.15e-05	5.04	2.63e-06	5.49
$\mathbf{80^2}$	3.39e-05	3.00	3.84e-05	3.00	1.09e-05	5.49	2.18e-05	4.93	3.59e-07	5.01	8.09e-08	5.02
Polynomial Degree p = 3												
$\mathbf{20^2}$	1.14e-04	-	1.26e-04	-	2.21e-03	-	1.41e-03	-	1.60e-06	-	1.86e-06	-
$\mathbf{40^2}$	7.49e-06	3.93	8.21e-06	3.94	1.25e-05	7.46	1.60e-05	6.45	8.32e-09	7.59	2.04e-09	9.84
$\mathbf{80^2}$	4.75e-07	3.98	5.23e-07	3.97	6.98e-08	7.48	1.40e-07	6.84	5.94e-11	7.13	1.37e-11	7.21

TABLE 4.8. *Two-dimensional system*

where, defining the complex number $z := x + iy$, the following three test cases are given:

$$
\begin{aligned}
r =& (z-(0.74+0.35i))(z-(0.68-0.59i)) \\
& (z-(-0.11-0.72i))(\overline{z}-(-0.58+0.64i)) \\
& (\overline{z}-(0.51-0.27i))(\overline{z}-(-0.12+0.84i))^2 \qquad \text{(Case 1)}, \\
r =& (z-(0.94+0.15i))(\overline{z}+(-0.38-0.39i)) \\
& (z-(0.09-0.92i))(\overline{z}-(-0.38+0.84i)) \\
& (\overline{z}-(0.71-0.07i)) \qquad \text{(Case 2)}, \\
r =& -(z-(0.74+0.35i))(z-(0.11-0.11i))^2 \\
& (z-(-0.11+0.72i))(z-(-0.58+0.64i)) \\
& (\overline{z}-(0.51-0.27i)) \qquad \text{(Case 3)}.
\end{aligned}
$$

For each test case, the position-dependent post-processor enhances the convergence rate from $O\left(h^{p+1}\right)$ to $O\left(h^{2p+1}\right)$. This can be seen from Table 4.9, Table 4.10, and Table 4.11. Figure 4.7 illustrates the *local* accuracy improvement for the first case.

An interesting effect can be seen in the three tables: for sufficiently large p, the errors of the post-processed field are of the order of the machine precision, which suggests that the exact solution has been reached. This happens e.g. for the second case with $p \geq 2$. For that problem, the exact solution is a polynomial of degree 5. At the same time, the post-processed solution is a piecewise polynomial of degree no more than $2p+1$ in each variable[6]. Combining these facts (observing $2p+1 \geq 5$ for $p \geq 2$), the high accuracy suggests that the post-processed L^2-projection onto the space of piecewise polynomials of degree p behaves like the L^2-projection onto the space of piecewise polynomials of degree $2p+1$ in each variable. Theoretical support for this phenomenon is left for future research.

A good feature of the post-processor is that it can enhance the accuracy of streamlines, especially near critical points. This was observed by Steffen et al. [73] for the symmetric post-processor, away from the boundary. Figure 4.8 shows that similar improvements are obtained for the position-dependent post-processor in the entire spatial domain (using a standard RK-4 method with time step $\Delta t = 0.01$ to compute the streamlines). We have translated the second field of Steffan et al. so that the critical points are located close to the boundary to emphasize the improved applicability and accuracy of the position-dependent post-processor near the boundary.

4.6 Conclusion

This chapter proposes the position-dependent post-processor as an alternative to the one-sided post-processor [64], and analyzes the impact of both strategies on the accuracy and smoothness of DG (upwind) approximations for hyperbolic problems. Our numerical results demonstrate that the new post-processor can enhance the convergence rate from order $p+1$ to order $2p+1$, in both the L^2- and the L^∞-norm. The differences with the original one-sided method occur at the boundary of the domain: in those regions, the new post-processor uses extra kernel nodes, as well as a smoother transition of the nodes. This results in significantly smaller errors with a more realistic smoothness. Altogether, unlike before, the proposed position-dependent post-processor can be used to obtain better smoothness and accuracy than the unfiltered DG approximation in the entire domain, including near (non-periodic) boundaries and shocks. This can aid to better visualization of the results, e.g. in the form of streamlines.

[6] This is because the post-processed solution is obtained from the convolution of a piecewise polynomial of degree p (the L^2-projection before post-processing) with a piecewise polynomial of degree p in each variable (the kernel, a linear combination of B-splines of degree $p+1$).

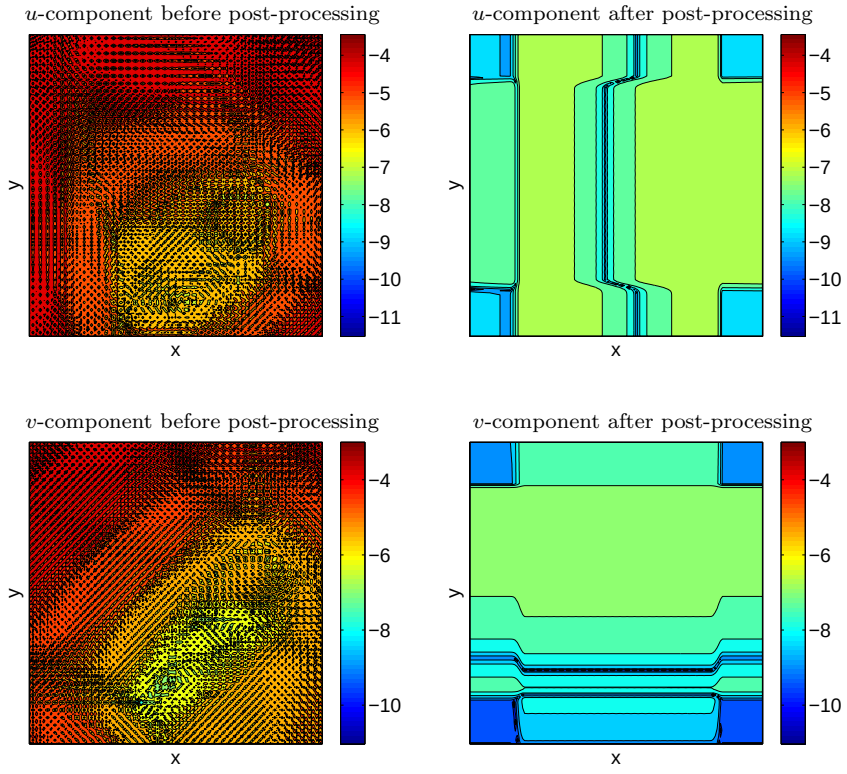


FIGURE 4.7. *Logarithm of the local error before and after post-processing (Case 1, $p = 2$, $N = 40^2$).*

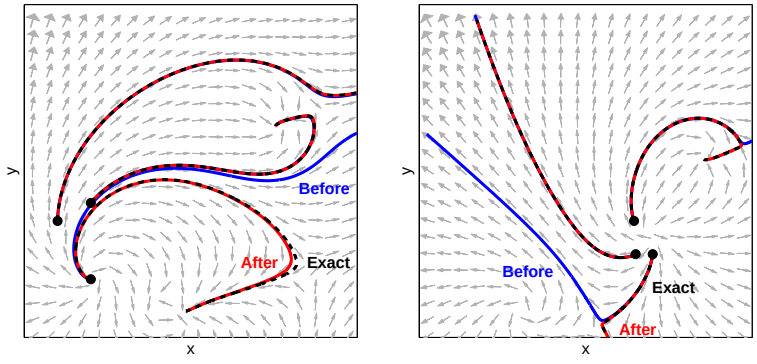


FIGURE 4.8. *Enhanced streamline visualization after post-processing for Case 1 (left) and Case 2 (right) ($p = 1$, $N = 20^2$).*

u-component:

mesh	L^2**-error**				L^∞**-error**			
	Before		**After**		**Before**		**After**	
	Error	Order	Error	Order	Error	Order	Error	Order
	Polynomial Degree p = 1							
$\mathbf{20^2}$	5.36e-02	-	3.58e-03	-	4.63e-01	-	2.26e-02	-
$\mathbf{40^2}$	1.35e-02	1.99	1.20e-04	4.90	1.27e-01	1.87	8.48e-04	4.74
$\mathbf{80^2}$	3.37e-03	2.00	5.98e-06	4.33	3.32e-02	1.93	2.95e-05	4.85
	Polynomial Degree p = 2							
$\mathbf{20^2}$	1.92e-03	-	6.01e-06	-	1.97e-02	-	7.56e-06	-
$\mathbf{40^2}$	2.41e-04	2.99	2.00e-07	4.91	2.67e-03	2.89	2.19e-07	5.11
$\mathbf{80^2}$	3.01e-05	3.00	4.23e-09	5.56	3.47e-04	2.94	4.25e-09	5.69
	Polynomial Degree p = 3							
$\mathbf{20^2}$	4.96e-05	-	1.08e-23	-	4.22e-04	-	1.84e-22	-
$\mathbf{40^2}$	3.11e-06	4.00	7.03e-22	-	2.80e-05	3.91	1.40e-20	-
$\mathbf{80^2}$	1.94e-07	4.00	2.13e-21	-	1.81e-06	3.96	7.16e-20	-

v-component:

mesh	L^2**-error**				L^∞**-error**			
	Before		**After**		**Before**		**After**	
	Error	Order	Error	Order	Error	Order	Error	Order
	Polynomial Degree p = 1							
$\mathbf{20^2}$	1.23e-01	-	4.82e-03	-	1.40e+00	-	3.53e-02	-
$\mathbf{40^2}$	3.10e-02	1.99	1.75e-04	4.78	3.77e-01	1.90	1.31e-03	4.75
$\mathbf{80^2}$	7.75e-03	2.00	9.84e-06	4.16	9.78e-02	1.95	4.57e-05	4.85
	Polynomial Degree p = 2							
$\mathbf{20^2}$	4.20e-03	-	1.14e-05	-	4.78e-02	-	1.56e-05	-
$\mathbf{40^2}$	5.27e-04	3.00	3.07e-07	5.21	6.37e-03	2.91	3.49e-07	5.48
$\mathbf{80^2}$	6.59e-05	3.00	5.99e-09	5.68	8.21e-04	2.95	6.31e-09	5.79
	Polynomial Degree p = 3							
$\mathbf{20^2}$	9.09e-05	-	1.75e-23	-	8.18e-04	-	3.47e-22	-
$\mathbf{40^2}$	5.69e-06	4.00	1.41e-21	-	5.37e-05	3.93	2.87e-20	-
$\mathbf{80^2}$	3.56e-07	4.00	2.44e-21	-	3.44e-06	3.96	6.81e-20	-

TABLE 4.9. *Case 1*

u-component:

mesh	L^2**-error**				L^∞**-error**			
	Before		**After**		**Before**		**After**	
	Error	Order	Error	Order	Error	Order	Error	Order
	Polynomial Degree p = 1							
$\mathbf{20^2}$	2.00e-02	-	2.57e-04	-	1.31e-01	-	7.92e-04	-
$\mathbf{40^2}$	5.00e-03	2.00	1.25e-05	4.36	3.42e-02	1.93	2.65e-05	4.90
$\mathbf{80^2}$	1.25e-03	2.00	8.39e-07	3.90	8.76e-03	1.97	9.38e-07	4.82
	Polynomial Degree p = 2							
$\mathbf{20^2}$	4.65e-04	-	2.68e-26	-	2.80e-03	-	2.95e-25	-
$\mathbf{40^2}$	5.83e-05	3.00	7.75e-25	-	3.62e-04	2.95	1.08e-23	-
$\mathbf{80^2}$	7.28e-06	3.00	2.34e-24	-	4.60e-05	2.98	4.55e-23	-
	Polynomial Degree p = 3							
$\mathbf{20^2}$	6.80e-06	-	4.13e-24	-	2.53e-05	-	5.94e-23	-
$\mathbf{40^2}$	4.25e-07	4.00	3.91e-22	-	1.61e-06	3.97	4.90e-21	-
$\mathbf{80^2}$	2.66e-08	4.00	9.63e-22	-	1.02e-07	3.99	2.31e-20	-

v-component:

mesh	L^2**-error**				L^∞**-error**			
	Before		**After**		**Before**		**After**	
	Error	Order	Error	Order	Error	Order	Error	Order
	Polynomial Degree p = 1							
$\mathbf{20^2}$	2.69e-02	-	2.72e-04	-	1.84e-01	-	8.04e-04	-
$\mathbf{40^2}$	6.74e-03	2.00	1.42e-05	4.26	4.80e-02	1.94	2.73e-05	4.88
$\mathbf{80^2}$	1.69e-03	2.00	9.50e-07	3.90	1.23e-02	1.97	1.05e-06	4.71
	Polynomial Degree p = 2							
$\mathbf{20^2}$	5.73e-04	-	7.39e-26	-	3.49e-03	-	1.04e-24	-
$\mathbf{40^2}$	7.17e-05	3.00	1.45e-24	-	4.50e-04	2.96	2.82e-23	-
$\mathbf{80^2}$	8.97e-06	3.00	3.63e-24	-	5.71e-05	2.98	9.69e-23	-
	Polynomial Degree p = 3							
$\mathbf{20^2}$	7.45e-06	-	1.01e-23	-	2.81e-05	-	1.77e-22	-
$\mathbf{40^2}$	4.66e-07	4.00	7.18e-22	-	1.79e-06	3.98	1.22e-20	-
$\mathbf{80^2}$	2.91e-08	4.00	1.68e-21	-	1.13e-07	3.99	4.11e-20	-

TABLE 4.10. *Case 2*

u-component:

mesh	L^2**-error**				L^∞**-error**			
	Before		**After**		**Before**		**After**	
	Error	Order	Error	Order	Error	Order	Error	Order
	Polynomial Degree p = 1							
$\mathbf{20}^2$	4.92e-02	-	1.34e-03	-	3.98e-01	-	8.15e-03	-
$\mathbf{40}^2$	1.23e-02	1.99	3.73e-05	5.16	1.06e-01	1.90	2.82e-04	4.85
$\mathbf{80}^2$	3.09e-03	2.00	1.56e-06	4.58	2.75e-02	1.95	8.84e-06	5.00
	Polynomial Degree p = 2							
$\mathbf{20}^2$	1.89e-03	-	4.50e-06	-	1.42e-02	-	4.78e-06	-
$\mathbf{40}^2$	2.37e-04	3.00	1.11e-07	5.34	1.88e-03	2.92	7.47e-08	6.00
$\mathbf{80}^2$	2.97e-05	3.00	2.03e-09	5.77	2.41e-04	2.96	1.17e-09	6.00
	Polynomial Degree p = 3							
$\mathbf{20}^2$	4.27e-05	-	3.98e-24	-	2.40e-04	-	5.29e-23	-
$\mathbf{40}^2$	2.67e-06	4.00	5.44e-22	-	1.56e-05	3.94	1.45e-20	-
$\mathbf{80}^2$	1.67e-07	4.00	1.25e-21	-	9.92e-07	3.97	4.04e-20	-

v-component:

mesh	L^2**-error**				L^∞**-error**			
	Before		**After**		**Before**		**After**	
	Error	Order	Error	Order	Error	Order	Error	Order
	Polynomial Degree p = 1							
$\mathbf{20}^2$	4.17e-02	-	7.11e-04	-	2.04e-01	-	3.90e-03	-
$\mathbf{40}^2$	1.05e-02	1.99	2.26e-05	4.98	5.39e-02	1.92	1.28e-04	4.93
$\mathbf{80}^2$	2.62e-03	2.00	1.10e-06	4.35	1.39e-02	1.96	4.17e-06	4.94
	Polynomial Degree p = 2							
$\mathbf{20}^2$	1.66e-03	-	3.65e-26	-	6.80e-03	-	5.33e-25	-
$\mathbf{40}^2$	2.08e-04	3.00	9.01e-25	-	8.95e-04	2.93	2.28e-23	-
$\mathbf{80}^2$	2.60e-05	3.00	2.58e-24	-	1.15e-04	2.96	9.17e-23	-
	Polynomial Degree p = 3							
$\mathbf{20}^2$	3.79e-05	-	4.01e-24	-	1.24e-04	-	5.62e-23	-
$\mathbf{40}^2$	2.37e-06	4.00	4.48e-22	-	8.05e-06	3.95	1.11e-20	-
$\mathbf{80}^2$	1.48e-07	4.00	9.47e-22	-	5.12e-07	3.97	3.13e-20	-

TABLE 4.11. *Case 3*

5

Theoretical Superconvergence

This chapter is based on:

L. Ji, P. van Slingerland, J.K. Ryan, C. Vuik, *Superconvergent error estimates for position-dependent smoothness-increasing accuracy-conserving (SIAC) post-processing of Discontinuous Galerkin Solutions.* Accepted for publication in Math. Comp.

5.1 Introduction

In Chapter 4, we proposed the position-dependent post-processor by generalizing the original symmetric and one-sided post-processor. Various numerical experiments demonstrated that, unlike before, this technique enhances both the smoothness and the unfiltered errors in the entire spatial domain, including the boundary region. Furthermore, an improvement of the convergence rate form order $p+1$ to order $2p+1$ was observed. This chapter focuses on theoretical support for these findings.

For the symmetric filter, such theory is already available: Bramble and Schatz [11] derived superconvergence in the L^2- and L^∞-norm for (continuous) Ritz-Galerkin approximations. Cockburn, Luskin, Shu and Süli [22] extended this work to show an accuracy improvement from order $p+1$ to order $2p+1$ for DG schemes in the L^2-norm for linear periodic hyperbolic problems. Interestingly, these results were established despite the fact that the post-processor does not contain any information of the underlying physics or numerics.

To extend this available theory for the symmetric filter and to explain the numerical results in Chapter 4, in this chapter, we derive error estimates for the generalized and position-dependent post-processor. In particular, we show that it enhances the DG convergence from order $p+1$ to order $2p+1$ in the L^2-norm and to order $\min\{2p+1, 2p+2-d/2\}$ in the L^∞-norm for d-dimensional linear periodic hyperbolic problems. Unlike [11, 22] these estimates are valid in the entire spatial domain, while the post-processor does not require information outside the domain. Furthermore, it is the first time that such results are established in the L^∞-norm for DG approximations.

The outline of this chapter is as follows. Section 5.2 discusses the DG method and post-processor under consideration, as well as some basic notation. Section 5.3 derives two auxiliary properties. Section 5.4 uses these to obtain the main error estimates in abstract form. Section 5.5 considers the implications for DG approximations and links the theory to the numerical observations in Chapter 4. Finally, Section 5.6 summarizes the main conclusions.

5.2 Methods and notation

This section specifies the methods and notation considered in this chapter. Section 5.2.1 recalls the generalized post-processor from Chapter 4. Section 5.2.2 summarizes the DG method for linear periodic hyperbolic problems. Section 5.2.3 introduces some additional notation that we will use frequently.

5.2.1 Post-processor

The position-dependent post-processor is based on a convex combination of two generalized post-processors (cf. Section 4.4). For this reason, in this chapter, we primarily focus on the generalized post-processor. Below, we repeat the def-

inition of the latter, using slightly different notation than before. Implications for the position-dependent post-processor are discussed in Section 5.5.3.

Recall the definition of a B-spline of order $\ell \geq 1$ (cf. (4.3)):

$$\psi^{(1)} := \mathbb{1}_{[-\frac{1}{2},\frac{1}{2}]}, \qquad \psi^{(\ell)} := \psi^{(\ell-1)} \star \psi^{(1)}, \qquad \text{for all } \ell \geq 2. \tag{5.1}$$

Next, we define the one-dimensional kernel for an evaluation point $\bar{x} \in (a, b)$ as the following linear combination of $r + 1$ B-splines of order ℓ:

$$K(x) = \sum_{j=0}^{r} c_j \psi^{(\ell)}(x - x_j), \qquad \text{for all } x \in \mathbb{R}, \tag{5.2}$$

where the kernel nodes read (the additional small factor $\varepsilon > 0$ is discussed below):

$$x_j = -\frac{r}{2} + j + \lambda(\bar{x}), \qquad \text{for all } j = 0, ..., r, \tag{5.3}$$

$$\lambda(\bar{x}) = \begin{cases} \min\{0, -\frac{r+\ell+\varepsilon}{2} + \frac{\bar{x}-a}{h}\}, & \text{for } \bar{x} \in [a, \frac{a+b}{2}), \\ \max\{0, \frac{r+\ell+\varepsilon}{2} + \frac{\bar{x}-b}{h}\}, & \text{for } \bar{x} \in [\frac{a+b}{2}, b], \end{cases} \tag{5.4}$$

and the kernel coefficients c_j satisfy:

$$\sum_{j=0}^{r} c_j \int_{-\infty}^{\infty} \psi^{(\ell)}(x)(x + x_j)^k \, \mathrm{d}x = \begin{cases} 1, & \text{for } k = 0, \\ 0, & \text{else.} \end{cases} \tag{5.5}$$

Different evaluation points result in different kernels.

Now that we have specified the one-dimensional kernel, we can define the generalized post-processor for a d-dimensional domain $\Omega = (a_1, b_1) \times ... \times (a_d, b_d)$. To this end, we proceed as in Section 4.4.3: consider an evaluation point $\bar{x} = (\bar{x}_1, ..., \bar{x}_d) \in \Omega$. Next, let K_j denote the one-dimensional kernel for the evaluation point $\bar{x}_j \in (a_j, b_j)$ (as specified above), and set:

$$K(x) = K_1(x_1) ... \, K_d(x_d), \qquad \text{for all } x \in \mathbb{R}^d. \tag{5.6}$$

Using this definition, we can apply the generalized post-processor to a function $u \in L^2(\Omega)$ by computing the convolution with the scaled kernel K:

$$u^\star(\bar{x}) = \frac{1}{h^d} \int_\Omega K\left(\frac{\bar{x} - x}{h}\right) u(x) \, \mathrm{d}x. \tag{5.7}$$

Note that we have added a factor $\frac{\varepsilon}{2}$ in the definition of the shift function (5.4). In practice, we can set $\varepsilon = 0$ (cf. (4.12)). However, for the theoretical purposes in this chapter, we need $\varepsilon > 0$ to be arbitrarily small yet fixed (independent of h). We stress that the resulting post-processor is still applicable in the entire spatial domain, including the region near the boundary. A nonzero value $\varepsilon > 0$ simply means that the post-processor is slightly less 'symmetric' near the boundary (cf. Section 4.3.3).

5.2.2 DG discretization for hyperbolic problems

We study the generalized post-processor in both an abstract framework and in the context of DG schemes for linear periodic hyperbolic problems on uniform meshes with exact time integration. Below, we specify the latter. We acknowledge that the aforementioned assumptions are quite strong, and usually not valid in practice. Nevertheless, numerical experiments show that the position-dependent post-processor enhances the accuracy in a similar manner for other problems as well (cf. Chapter 4).

Altogether, we study the following d-dimensional linear hyperbolic problem on the spatial domain $\Omega = [0,1]^d$:

$$u_t + \sum_{j=1}^{d} A_j u_{x_j} + A_0 u = 0, \tag{5.8}$$

with initial condition u_0 and periodic boundary conditions. The coefficients A_j are assumed to be scalar and constant.

To obtain a DG approximation for this system, consider a uniform mesh for the spatial domain Ω with elements $E_1, ..., E_N$ of size $h \times ... \times h$. Next, define the test space V that contains each element of $L^2(\Omega)$ that is a polynomial of degree p or lower within each mesh element, and that may be discontinuous at the mesh element boundaries.

At the initial time $t = 0$, the DG approximation u_h is the L^2-projection of u_0 onto V. For $t > 0$, u_h is the function in V such that:

$$\int_\Omega (u_h)_t \, v + B(u_h, v) = 0, \qquad \forall v \in V,$$

where B is a bilinear form defined hereafter.

To specify B, let $(n_1^{(i)}, ..., n_d^{(i)})$ be the outward normal of E_i. Furthermore, let $\hat{u}_h$ denote the usual upwind flux (cf. Section 4.2). Now, the bilinear form B can be specified as follows:

$$B(u_h, v) = \sum_{i=1}^{N} \left(\sum_{e \in \partial E_i} \int_e \sum_{j=1}^{d} A_j n_j^{(i)} \hat{u}_h \, v + \int_{E_i} u_h \left(-\sum_{j=1}^{d} A_j v_{x_j} + A_0 v \right) \right).$$

5.2.3 Additional notation

This section specifies some additional notation that we use throughout this chapter.

Unless specified otherwise, Ω denotes a spatial domain of the form $(a_1, b_1) \times ... \times (a_d, b_d)$. The standard norms in $L^p(\Omega)$ are denoted as:

$$\|u\|_{L^p(\Omega)} = \left(\int_\Omega |u|^p \right)^{\frac{1}{p}}, \qquad 1 \leq p < \infty,$$

$$\|u\|_{L^\infty(\Omega)} = \operatorname*{ess\,sup}_{x\in\Omega} |u(x)|.$$

Furthermore, for integers $k \geq 0$ and $p = 2, \infty$, let $W^{k,p}(\Omega)$ denote the usual Sobolev space, i.e. the set of all functions u such that, for every d-dimensional multi-index[1] α with $|\alpha| \leq k$, the weak partial derivative $D^\alpha u$ belongs to $L^p(\Omega)$. For $p = 2$, we write $H^k(\Omega) = W^{k,2}(\Omega)$, which is equipped with the norm:

$$\|u\|_{H^k(\Omega)} = \left(\sum_{|\alpha|\leq k} \|D^\alpha u\|^2_{L^2(\Omega)} \right)^{1/2}. \tag{5.9}$$

Additionally, we define the negative-order space $H^{-k}(\Omega)$ for integers $k \geq 0$: this space is the closure of the smooth functions $C^\infty(\Omega)$ with respect to the so-called negative-order norm[2]:

$$\|u\|_{H^{-k}(\Omega)} = \sup_{v\in C_0^\infty(\Omega)} \frac{\int_\Omega uv}{\|v\|_{H^k(\Omega)}}. \tag{5.10}$$

A multi-variate B-spline of order ℓ is the tensor product of one-dimensional B-splines:

$$\psi^{(\ell)}(x) = \psi^{(\ell)}(x_1)...\,\psi^{(\ell)}(x_d), \qquad \forall x \in \mathbb{R}^d.$$

A scaled multi-variate B-spline is defined as:

$$\psi_h^{(\ell)}(x) = \frac{1}{h^d}\psi^{(\ell)}\left(\frac{1}{h}x\right), \qquad \forall x \in \mathbb{R}^d.$$

The usual central difference operator in the j^{th} direction is denoted as[3]

$$\partial^1_{h,j} u(x) = \frac{u(x + \frac{h}{2}e_j) - u(x - \frac{h}{2}e_j)}{h}.$$

Using this notation, we set for any d-dimensional multi-index α (writing $\partial^k_{h,j}u = \partial^{k-1}_{h,j}\partial^1_{h,j}u$ for integers $k \geq 2$):

$$\partial^\alpha_h u = \partial^{\alpha_1}_{h,1}...\partial^{\alpha_d}_{h,d}u.$$

Finally, we use the notation Ω_α for the largest subset of $\Omega = (a_1, b_1) \times ... \times (a_d, b_d)$ such that $\partial^\alpha_h u : \Omega_\alpha \to \mathbb{R}$ does not require information outside Ω:

$$\Omega_\alpha = \left(a_1 + \alpha_1\frac{h}{2}, b_1 - \alpha_1\frac{h}{2}\right) \times ... \times \left(a_d + \alpha_d\frac{h}{2}, b_d - \alpha_d\frac{h}{2}\right). \tag{5.11}$$

For scalars γ we set $\Omega_\gamma = \Omega_{(\gamma,...,\gamma)}$.

[1] A d-dimensional multi-index α is a d-tuple of nonnegative integers: $\alpha = (\alpha_1, ..., \alpha_d)$. Furthermore, $|\alpha| = \alpha_1 + ... + \alpha_d$ and $D^\alpha u = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} ... \left(\frac{\partial}{\partial x_d}\right)^{\alpha_d} u$.

[2] $C_0^\infty(\Omega)$ denotes the usual set of all functions in $C^\infty(\Omega)$ with compact support.

[3] Here, e_j is the multi-index whose j^{th} component is 1 and all other components are 0.

5.3 Auxiliary results

To obtain the main error estimates, we require two auxiliary results, which are discussed in this section. Section 5.3.1 derives an estimate for $\|u - u^\star\|$. Section 5.3.2 expresses derivatives of convolutions with B-splines in terms of central differences.

5.3.1 Estimating $\|u - u^\star\|$

In Section 4.3.2, we mentioned that the symmetric post-processor reproduces polynomials of degree $2p$. As a result, the difference between u and $u^\star$ is of order $2p+1$ (assuming that u is sufficiently smooth). Actually, Bramble and Schatz [11, Lemma 5.2] designed the symmetric kernel this way. In this section, we show that a similar result holds for the generalized post-processor. Unlike before, this estimate is valid in the entire spatial domain.

Lemma 5.3.1 *Consider the generalized post-processor with $r+1$ kernel nodes, as defined in Section 5.2.1. Let $k \leq r+1$ be a positive integer. Then*[4]*,*

$$\|u - u^\star\|_{L^\infty(\Omega)} \lesssim \sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)}\, h^k, \qquad \forall u \in W^{k,\infty}(\Omega). \tag{5.12}$$

Here, α denotes a d-dimensional multi-index. The constant involved depends on the L^1-norm of the kernels, as indicated in the proof below.

To show Lemma 5.3.1, we follow the same strategy sketched for the symmetric post-processor in [11, p. 103]: the main idea is to demonstrate (the proof is given below) that the generalized post-processor reproduces polynomials p of degree r (or lower, in each variable):

$$p^*(\bar{x}) = p(\bar{x}), \qquad \forall \bar{x} \in \Omega. \tag{5.13}$$

Using this property, the proof can be completed using Taylor's theorem: for $u \in W^{k,\infty}(\Omega)$ and $x, \bar{x} \in \Omega$, we have[5]:

$$\begin{aligned} u(x) \;=\; & \sum_{|\alpha|\leq k-1} \frac{(x-\bar{x})^\alpha}{\alpha!} D^\alpha u(\bar{x}) \\ & + k \sum_{|\alpha|=k} \frac{(x-\bar{x})^\alpha}{\alpha!} \int_0^1 s^{k-1} D^\alpha u\big(x + s(\bar{x}-x)\big)\, \mathrm{d}s. \end{aligned} \tag{5.14}$$

[4] Throughout this chapter, we use the symbol $\lesssim$ in expressions of the form "$F(x) \lesssim G(x)$ for all $x \in X$" to indicate that there exists a constant $C > 0$, independent of the variable x and the scaling h, such that $F(x) \leq CG(x)$ for all $x \in X$. The symbol $\gtrsim$ is defined similarly.

[5] For a d-dimensional multi-index α, we write $\alpha! = \alpha_1! \ldots \alpha_d!$ and $x^\alpha = x^{\alpha_1} \ldots x^{\alpha_d}$.

This follows from [14, p. 83, 100, 101] (using the chain rule and Lipschitz continuity of the derivatives of order $k-1$ and lower [30, p. 269]). We can now show Lemma 5.3.1:

PROOF (OF LEMMA 5.3.1) To show (5.13), first consider the one-dimensional case. Without loss of generality, we may assume that p is a monomial basis function, i.e. $p(x) = x^m$ (with $m = 0, ..., r$). Next, observe that, by definition of the post-processor,

$$p^\star(\bar{x}) := \frac{1}{h}\int_\Omega K\left(\frac{\bar{x}-x}{h}\right) x^m \,\mathrm{d}x := \sum_{j=0}^{r}\int_{-\infty}^{\infty} c_j \psi^{(\ell)}\Bigg(\underbrace{\frac{\bar{x}-x}{h} - x_j}_{=:y}\Bigg) x^m \,\mathrm{d}x.$$

$$= \sum_{j=0}^{r} c_j \int_{-\infty}^{\infty} \psi^{(\ell)}(y)\big(\bar{x} - h(y+x_j)\big)^m \,\mathrm{d}y.$$

Using the binomial theorem we may write:

$$p^\star(\bar{x}) = \sum_{n=0}^{m} \frac{m!}{(m-n)!n!}\,\bar{x}^n(-h)^{m-n} \underbrace{\sum_{j=0}^{r} c_j \int_{\mathbb{R}} \psi^{(\ell)}(y)(y+x_j)^{m-n}\,\mathrm{d}y}_{=\,1 \text{ for } m=n, \text{ and zero otherwise (5.5)}} = x^m = p(x).$$

This completes the proof of (5.13) for the one-dimensional case. For higher-dimensional problems, this result follows from (5.7) and repetitive application of (5.13) for one-dimensional kernels.

Now that we have obtained (5.13), we can show (5.12). To this end, choose an evaluation point $\bar{x} \in \Omega$. Because the post-processor reproduces polynomials (5.13), we may write for any polynomial p of degree r or lower (note that u, $u^\star$ and p are continuous):

$$|u - u^\star|(\bar{x}) = |(u-p)(\bar{x}) - (u-p)^\star(\bar{x})|.$$

In particular, we may choose p to be the Taylor polynomial of degree $k-1$ that approximates u near the point $\bar{x}$. As a consequence, $(u-p)(\bar{x}) = 0$, and we may write:

$$\begin{aligned}
|u-u^\star|(\bar{x}) &= |(u-p)^\star(\bar{x})| \\
&= \left|\frac{1}{h^d}\int_\Omega K\Big(\underbrace{\frac{\bar{x}-x}{h}}_{=:y}\Big)(u-p)(x)\,\mathrm{d}x\right| \\
&= \left|\int_{\mathrm{supp}\{K\}} K(y)(u-p)(\bar{x}-hy)\,\mathrm{d}y\right| \\
&\overset{\text{Hölder}}{\leq} \|K\|_{L^1(\mathbb{R}^d)}\,\|(u-p)(\bar{x}-h.)\|_{L^\infty(\mathrm{supp}(K))}\,. \qquad (5.15)
\end{aligned}$$

To estimate the second term, we apply Taylor's theorem (5.14) for all $y \in \mathrm{supp}(K)$:

$$\begin{aligned}
|u-p|(\bar{x}-hy) &= \left|k\sum_{|\alpha|=k}\frac{(-hy)^\alpha}{\alpha!}\int_0^1 s^{k-1} D^\alpha u\big(\bar{x}-hy+s(hy)\big)\,\mathrm{d}s\right| \\
&\leq \sum_{|\alpha|=k}\left|\frac{-hy^\alpha}{\alpha!}\right| \|D^\alpha u\|_{L^\infty(\Omega)}\, \underbrace{k\int_0^1 s^{k-1}\,\mathrm{d}s}_{=1} \\
&\leq \sum_{|\alpha|=k}\frac{|y^\alpha|}{\alpha!}\,\|D^\alpha u\|_{L^\infty(\Omega)}\,h^k
\end{aligned}$$

Substitution of this result into (5.15) yields:

$$|u-u^\star|(\bar{x}) \leq \|K\|_{L^1(\mathbb{R}^d)} \sup_{y\in\mathrm{supp}(K)} \left\{ \sum_{|\alpha|=k} \frac{|y^\alpha|}{\alpha!} \|D^\alpha u\|_{L^\infty(\Omega)} \right\} h^k$$
$$\lesssim \sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)} h^k, \qquad \forall \bar{x} \in \Omega.$$

Here we have used that K is independent of h for all $\bar{x}$ (although different $\bar{x}$ yield different kernels). We now arrive at (5.12), which completes the proof. ■

5.3.2 Derivatives of B-splines

The derivative of a B-spline ψ^ℓ can be expressed as the central difference of the lower-order B-spline $\psi^{\ell-1}$ [70, p. 12]. In [11, Lemma 5.3], this property was exploited to estimate norms of derivatives of $u^\star$ for the symmetric-post-processor. In this section, we obtain a similar result for the convolution of u with a single B-spline (without requiring continuity of u). This reduction to the core elements of the post-processor is convenient when handling different kernel types later on. Another difference with [11] is that we provide an explicit expression for the largest subdomain for which the estimates are valid (without requiring information outside Ω). These subdomains will turn out to be just large enough to ensure that our main estimates are applicable in the entire spatial domain. Altogether, we have the following result:

Lemma 5.3.2 *Let α be a d-dimensional multi-index whose entries are not larger than $\ell \geq 1$. If $u \in L^\infty(\Omega)$, then $\psi_h^{(\ell)} \star u \in W^{\ell,\infty}(\Omega_\ell)$, and*

$$\left\| D^\alpha \left(\psi_h^{(\ell)} \star u \right) \right\|_{L^\infty(\Omega_\ell)} \leq \|\partial_h^\alpha u\|_{L^\infty(\Omega_\alpha)}. \tag{5.16}$$

Similarly, if $u \in L^2(\Omega)$, then $\psi_h^{(\ell)} \star u \in H^\ell(\Omega_\ell)$, and, for $k \geq 0$:

$$\left\| D^\alpha \left(\psi_h^{(\ell)} \star u \right) \right\|_{H^{-k}(\Omega_\ell)} \leq \|\partial_h^\alpha u\|_{H^{-k}(\Omega_\alpha)}. \tag{5.17}$$

To show Lemma 5.3.2, the main idea is to demonstrate (the proof is given below)[6]:

$$D^\alpha \psi_h^{(\ell)} \star u = \psi_h^{(\ell-\alpha)} \star \partial_h^\alpha u. \tag{5.18}$$

Furthermore, we make use of Young's inequality for convolutions [9, Theorem 3.9.4]: for $1 \leq p, q, r \leq \infty$, $f \in L^p(R^d)$, and $g \in L^q(R^d)$:

$$\frac{1}{p} + \frac{1}{q} = \frac{1}{r} + 1 \qquad \Rightarrow \qquad \|f \star g\|_{L^r(\mathbb{R}^d)} \leq \|f\|_{L^p(\mathbb{R}^d)} \|g\|_{L^q(\mathbb{R}^d)}. \tag{5.19}$$

[6]Here, we use the notation $\psi_h^{(\ell-\alpha)}(x) = \psi_h^{(\ell-\alpha_1)}(x_1) \ldots \psi_h^{(\ell-\alpha_d)}(x_d)$, for all $x \in \mathrm{R}^d$. Furthermore, the notation $\psi_h^{(0)} \star u$ should be interpreted as u.

Additionally, we use that [70, p.3]:

$$\left\|\psi_h^{(\ell)}\right\|_{L^1(\mathbb{R}^d)} = 1. \tag{5.20}$$

We can now show Lemma 5.3.2:

Proof (of Lemma 5.3.2) To show (5.18), first, consider the one-dimensional case with $\ell = 1$:

$$\left(\psi_h^{(1)} \star u\right)(x) = \frac{1}{h}\int_{-\frac{h}{2}}^{\frac{h}{2}} u(\underbrace{x-y}_{=:s})\,\mathrm{d}y = \frac{1}{h}\int_{x-\frac{h}{2}}^{x+\frac{h}{2}} u(s)\,\mathrm{d}s.$$

As a result, $\psi_h^{(1)} \star u$ is continuous, and $D(\psi_h^{(1)} \star u) = \partial_h u$ almost everywhere [9, Theorem 5.4.2]. For $\ell \geq 1$, we may now write:

$$D\left(\psi_h^{(\ell)} \star u\right) \overset{(5.1)}{=} D\left(\psi_h^{(\ell-1)} \star \psi_h^{(1)} \star u\right) = \psi_h^{(\ell-1)} \star D\left(\psi_h^{(1)} \star u\right) = \psi_h^{(\ell-1)} \star \partial_h u.$$

For higher-order derivatives, we may repeat this strategy obtain (5.18) for the one-dimensional case. For the multi-dimensional case, we apply the above in each direction, which then completes the proof of (5.18).

To show (5.16), note that (5.18) implies that

$$\left\|D^\alpha\left(\psi_h^{(\ell)} \star u\right)\right\|_{L^\infty(\Omega_\ell)} = \left\|\psi_h^{(\ell-\alpha)} \star \partial_h^\alpha u\right\|_{L^\infty(\Omega_\ell)}.$$

Next, define $w \in L^\infty(\mathbb{R}^d)$ such that $w = \partial_h^\alpha u$ in Ω_α and zero everywhere else. Because of the local support of the B-spline, the convolution in the right hand side above requires only information of $\partial_h^\alpha u$ in $\Omega_\alpha \supseteq \Omega_\ell$. Hence, we may replace $\partial_h^\alpha u$ by w:

$$\left\|D^\alpha\left(\psi_h^{(\ell)} \star u\right)\right\|_{L^\infty(\Omega_\ell)} = \left\|\psi_h^{(\ell-\alpha)} \star w\right\|_{L^\infty(\Omega_\ell)}.$$

Next, we apply Young's inquality (5.19) with $p = 1$, and $q, r = \infty$:

$$\left\|D^\alpha\left(\psi_h^{(\ell)} \star u\right)\right\|_{L^\infty(\Omega_\ell)} \leq \left\|\psi_h^{(\ell-\alpha)}\right\|_{L^1(\mathbb{R}^d)} \|w\|_{L^\infty(\mathbb{R}^d)}.$$

Using (5.20) and the fact that $\|w\|_{L^\infty(\mathbb{R}^d)} = \left\|\partial_h^\alpha u\right\|_{L^\infty(\Omega_\alpha)}$ now yields (5.16).

To show (5.17), note that (5.18) implies that

$$\left\|D^\alpha\left(\psi_h^{(\ell)} \star u\right)\right\|_{H^{-k}(\Omega_\ell)} = \left\|\psi_h^{(\ell-\alpha)} \star \partial_h^\alpha u\right\|_{H^{-k}(\Omega_\ell)}$$

As before, we define $w \in L^\infty(\mathbb{R}^d)$ such that $w = \partial_h^\alpha u$ in Ω_α and zero everywhere else, so we may replace $\partial_h^\alpha u$ by w:

$$\left\|D^\alpha\left(\psi_h^{(\ell)} \star u\right)\right\|_{H^{-k}(\Omega_\ell)} = \left\|\psi_h^{(\ell-\alpha)} \star w\right\|_{H^{-k}(\Omega_\ell)}$$

Next, choose $v \in C_0^\infty(\Omega_\ell)$, and extend it to $\mathbb{R}^d$ by setting it equal to zero outside Ω_ℓ. We may now write:

$$\int_{\Omega_\ell}\left(\psi_h^{(\ell-\alpha)} \star w\right) v \overset{\text{Fubini}}{=} \int_{\Omega_\alpha} w\,\left(\psi_h^{(\ell-\alpha)} \star v\right).$$

Next, we note that $\psi_h^{(\ell-\alpha)} \star v \in C_0^\infty(\Omega_\alpha)$, so we may consider it as a test function in the definition of the negative-order norm (5.10) of w:

$$\int_{\Omega_\ell}\left(\psi_h^{(\ell-\alpha)} \star w\right) v = \underbrace{\frac{\int_{\Omega_\alpha} w\,\left(\psi_h^{(\ell-\alpha)} \star v\right)}{\left\|\psi_h^{(\ell-\alpha)} \star v\right\|_{H^k(\Omega_\alpha)}}}_{\leq \|w\|_{H^{-k}(\Omega_\alpha)}} \left\|\psi_h^{(\ell-\alpha)} \star v\right\|_{H^k(\Omega_\alpha)}$$

$$\leq \|w\|_{H^{-k}(\Omega_\alpha)} \left\| \psi_h^{(\ell-\alpha)} \star v \right\|_{H^k(\Omega_\alpha)}.$$

At the same time,

$$\begin{aligned}\left\| \psi_h^{(\ell-\alpha)} \star v \right\|^2_{H^k(\Omega_\alpha)} &= \sum_{|\beta|\leq k} \left\| D^\beta \left(\psi_h^{(\ell-\alpha)} \star v \right) \right\|^2_{L^2(\Omega_\alpha)} \\ &= \sum_{|\beta|\leq k} \left\| \psi_h^{(\ell-\alpha)} \star \left(D^\beta v \right) \right\|^2_{L^2(\Omega_\alpha)}.\end{aligned}$$

Next, apply Young's inequality (5.19) with $p = 1$ and $q, r = 2$ to obtain (using that v, and thus its derivatives, have compact support in Ω_ℓ):

$$\left\| \psi_h^{(\ell-\alpha)} \star v \right\|^2_{H^k(\Omega_\alpha)} \leq \sum_{|\beta|\leq k} \underbrace{\left\| \psi_h^{(\ell-\alpha)} \right\|^2_{L^1(\mathbb{R}^d)}}_{=1} \left\| D^\beta v \right\|^2_{L^2(\Omega_\ell)} \overset{(5.20)}{=} \|v\|^2_{H^k(\Omega_\ell)}.$$

Finally, we combine the results above to obtain:

$$\left\| D^\alpha \left(\psi_h^{(\ell)} \star u \right) \right\|_{H^{-k}(\Omega_\ell)} = \sup_{v \in C_0^\infty(\Omega_\ell)} \frac{\int_{\Omega_\ell} \left(\psi_h^{(\ell-\alpha)} \star w \right) v}{\|v\|_{H^k(\Omega_\ell)}} \leq \|w\|_{H^{-k}(\Omega_\alpha)}.$$

Replacing w by $\partial_h^\alpha u$ completes the proof of (5.17). ∎

5.4 The main result in abstract form

Using the auxiliary properties discussed in the previous section, we now derive an estimate for $\|u - v^\star\|$. This result is applicable for any (sufficiently smooth) functions u and v. The implications for DG approximations will be studied in Section 5.5. Section 5.4.1 expresses a post-processed function in terms of convolutions with B-splines. Section 5.4.2 estimates the remaining terms further. Section 5.4.3 combines these two results with Lemma 5.3.1 to obtain the final estimate for $\|u - v^\star\|$.

5.4.1 Reducing the post-processor to its building blocks

The first step is to express $v^\star$ in terms of convolutions with B-splines. This removes the dependency on the evaluation point and, as such, simplifies the analysis. As before, we provide explicit expressions for the subdomains involved, which are crucial to ensure that our final error estimates apply in the entire domain. Altogether, we show the following result:

Lemma 5.4.1 *Consider the generalized post-processor with B-splines of order $\ell \geq 1$ and $\varepsilon > 0$ small, as defined in Section 5.2.1. Then, for all $v \in L^2(\Omega)$:*

$$\|v^\star\|_{L^2(\Omega)} \lesssim \left\| \psi_h^{(\ell)} \star v \right\|_{L^2(\Omega_{\ell+\varepsilon})}, \tag{5.21}$$

$$\|v^\star\|_{L^\infty(\Omega)} \lesssim \left\| \psi_h^{(\ell)} \star v \right\|_{L^\infty(\Omega_{\ell+\varepsilon})}. \tag{5.22}$$

The constants involved depend on the kernel coefficients, as indicated in the proof below.

To show Lemma 5.4.1, the key is to observe that the kernel nodes are located within $\Omega_{\ell+\varepsilon}$ during the convolution (this is motivated below).

Proof (of Lemma 5.4.1) First, consider the one-dimensional case, and choose an evaluation point $\bar{x} \in \Omega$. Then,

$$v^\star(\bar{x}) = \sum_{j=0}^{r} c_j \int_\Omega \frac{1}{h}\psi^{(\ell)}\left(\frac{\bar{x}-x}{h} - x_j\right) v(x)\,\mathrm{d}x = \sum_{j=0}^{r} c_j \big(\psi_h^{(\ell)} \star v\big)\underbrace{(\bar{x} - hx_j)}_{\in\Omega_{\ell+\varepsilon}}.$$

Next, observe that $\bar{x} - hx_j \in \Omega_{\ell+\varepsilon}$ for all $\bar{x} \in \Omega$, which can be seen as follows. At the end points of the domain $\Omega = (a, b)$, the shift function (5.4) reads $\lambda(a) = -\frac{r+\ell+\varepsilon}{2}$ and $\lambda(b) = \frac{r+l+\varepsilon}{2}$ respectively. Hence, for $\bar{x} = a$, the right most kernel node (5.3) is $x_r(a) = -\frac{\ell+\varepsilon}{2}$. Similarly, for $\bar{x} = b$, the left-most kernel node is $x_0(b) = \frac{\ell+\varepsilon}{2}$. Hence, the quantity $\bar{x} - hx_j$ takes values in $[a + h\frac{\ell+\epsilon}{2}, b - h\frac{\ell+\varepsilon}{2}]$, which is precisely $\Omega_{\ell+\varepsilon}$ by definition (5.11).

Considering the L^2-norm of $v^\star$ now yields (recall the dependence of the kernel coefficients on the evaluation point):

$$\begin{aligned}
\|v^\star\|^2_{L^2(\Omega)} &= \int_\Omega \left|\sum_{j=0}^{r} c_j \big(\psi_h^{(\ell)} \star v\big)(\bar{x} - hx_j)\right|^2 \mathrm{d}\bar{x} \\
&\overset{\text{Cauchy-Schwartz}}{\leq} \int_\Omega \left(\sum_{j=0}^{r} |c_j|^2\right)\left(\sum_{j=0}^{r}\left|\big(\psi_h^{(\ell)} \star v\big)(\bar{x} - hx_j)\right|^2\right) \mathrm{d}\bar{x} \\
&\leq \sup_{\bar{x}\in\Omega}\left(\sum_{j=0}^{r}|c_j(\bar{x})|^2\right)\sum_{j=0}^{r}\int_\Omega \left|\big(\psi_h^{(\ell)} \star v\big)\underbrace{(\bar{x} - hx_j)}_{\in\Omega_{\ell+\varepsilon}}\right|^2 \mathrm{d}\bar{x} \\
&\leq \underbrace{r \sup_{\bar{x}\in\Omega}\left(\sum_{j=0}^{r}|c_j(\bar{x})|^2\right)}_{\text{constant}} \left\|\psi_h^{(\ell)} \star v\right\|^2_{L^2(\Omega_{\ell+\varepsilon})}.
\end{aligned}$$

Similarly, when we consider the L^∞-norm of $v^\star$, we obtain:

$$\begin{aligned}
\|v^\star\|_{L^\infty(\Omega)} &= \sup_{\bar{x}}\left\{\sum_{j=0}^{r} c_j \big(\psi_h^{(\ell)} \star v\big)\underbrace{(\bar{x} - hx_j)}_{\in\Omega_{\ell+\varepsilon}}\right\} \\
&\leq \underbrace{\sup_{\bar{x}\in\Omega}\left(\sum_{j=0}^{r}|c_j(\bar{x})|\right)}_{\text{constant}} \left\|\psi_h^{(\ell)} \star v\right\|_{L^\infty(\Omega_{\ell+\varepsilon})}.
\end{aligned}$$

This completes the proof for the one-dimensional case.

For general $d \geq 1$, the desired result follows by applying the strategy above for each spatial direction. To be more specific, recall that the kernel K is a tensor product of one-dimensional kernels $K_1, ..., K_d$ (cf. (5.6)). We can now write for $\bar{x} \in \Omega$ (the

super-scripted indices below indicate the corresponding one-dimensional kernel):

$$v^\star(\bar{x}) = \sum_{j_1=0}^{r} \dots \sum_{j_d=0}^{r} c_{j_1}^{(1)} \dots c_{j_d}^{(d)} \big(\psi_h^{(\ell)} \star v\big) \underbrace{\left(\bar{x} - h \begin{pmatrix} x_{j_1}^{(1)} \\ \vdots \\ x_{j_d}^{(d)} \end{pmatrix} \right)}_{\in \Omega_{\ell+\varepsilon}} .$$

Considering the L^2- and L^∞-norm yields:

$$\|v^\star\|_{L^2(\Omega)} \leq \underbrace{\sqrt{d\, r \sup_{\bar{x}\in\Omega} \left(\sum_{j_1=0}^{r} \dots \sum_{j_d=0}^{r} \left| c_{j_1}^{(1)}(\bar{x}) \dots c_{j_d}^{(d)}(\bar{x}) \right|^2 \right)}}_{\text{constant}} \left\| \psi_h^{(\ell)} \star v \right\|_{L^2(\Omega_{\ell+\varepsilon})} ,$$

$$\|v^\star\|_{L^\infty(\Omega)} \leq \underbrace{\sup_{\bar{x}\in\Omega} \left(\sum_{j_1=0}^{r} \dots \sum_{j_d=0}^{r} \left| c_{j_1}^{(1)}(\bar{x}) \dots c_{j_d}^{(d)}(\bar{x}) \right| \right)}_{\text{constant}} \left\| \psi_h^{(\ell)} \star v \right\|_{L^\infty(\Omega_{\ell+\varepsilon})} .$$

This completes the proof. ∎

5.4.2 Treating the remaining building blocks

Now that we have reduced the post-processor to its building blocks, in this section, we further estimate the latter. To this end, we follow Bramble and Schatz [11, p. 104–105], while considering B-splines rather than the symmetric kernel (and without any restriction to continuous functions). As before, another difference with [11] is that we keep a careful administration of the subdomains involved, as this is crucial for our final estimate to be applicable in the entire domain. Altogether, we show the following result:

Lemma 5.4.2 *Consider a B-spline of order $\ell \geq 1$, let $\varepsilon > 0$ small (cf. Section 5.2.1), and define*[7] *$d_0 = 1 + [d/2]$. Then, for all $v \in L^2(\Omega)$:*

$$\left\| \psi_h^{(\ell)} \star v \right\|_{L^2(\Omega_{\ell+\varepsilon})} \lesssim \sum_{|\alpha|\leq\ell} \|\partial_h^\alpha v\|_{H^{-\ell}(\Omega_\alpha)} . \tag{5.23}$$

Furthermore, for all $v \in L^\infty(\Omega)$:

$$\left\| \psi_h^{(\ell)} \star v \right\|_{L^\infty(\Omega_{\ell+\varepsilon})} \lesssim \sum_{|\alpha|\leq\ell+d_0} \|\partial_h^\alpha v\|_{H^{-\ell}(\Omega_\alpha)} + h^\ell \sum_{|\alpha|\leq\ell} \|\partial_h^\alpha v\|_{L^\infty(\Omega_\alpha)} . \tag{5.24}$$

[7] Here, $[d/2]$ denotes the integer part of $d/2$.

To show Lemma 5.4.2, the main idea is to switch from L^2-norms to negative-order norms at the cost of smoothness: for open bounded d-dimensional domains[8] $X_0 \Subset X_1$ and nonnegative integers k, we have:

$$\|u\|_{L^2(X_0)} \lesssim \sum_{|\alpha|\leq k} \|D^\alpha u\|_{H^{-k}(X_1)}, \qquad \forall u \in L^2(X_1). \tag{5.25}$$

This has been shown in [11, p. 96]. At the same time, we can switch from L^∞-norms to L^2-norms (again, at the cost of smoothness): for open bounded d-dimensional domains $X_0 \Subset X_1$ and $d_0 := [d/2]+1$, we have that $u \in H^{d_0}(X_1)$ is continuous almost everywhere in X_0, and:

$$\|u\|_{L^\infty(X_0)} \lesssim \|u\|_{H^{d_0}(X_1)}, \qquad \forall u \in H^{d_0}(X_1). \tag{5.26}$$

This result is given in [12, p. 679]. Combining these relations with the expression for derivatives of B-splines in Lemma 5.3.2, we can now show Lemma 5.4.2 as follows:

Proof (of Lemma 5.4.2) Relation (5.23) is obtained as follows:

$$\begin{aligned}\left\|\psi_h^{(\ell)} \star v\right\|_{L^2(\Omega_{\ell+\varepsilon})} &\overset{(5.25),\ \Omega_{\ell+\varepsilon}\Subset\Omega_\ell}{\lesssim} \sum_{|\alpha|\leq\ell} \left\|D^\alpha\left(\psi_h^{(\ell)} \star v\right)\right\|_{H^{-\ell}(\Omega_\ell)} \\ &\overset{\text{Lemma 5.3.2}}{\leq} \sum_{|\alpha|\leq\ell} \|\partial_h^\alpha v\|_{H^{-\ell}(\Omega_\alpha)}.\end{aligned}$$

Relation (5.24) requires a little more work: let $(\psi_h^{(\ell)} \star v)^\star$ denote the result of applying the generalized post-processor to $\psi_h^{(\ell)} \star v$ using $r+1$ B-splines of order d_0 (!) in the domain $\Omega_{\ell+\varepsilon}$ (cf. Section 5.2.1). The triangle inequality then gives:

$$\left\|\psi_h^{(\ell)} \star v\right\|_{L^\infty(\Omega_{\ell+\varepsilon})} \leq \left\|\psi_h^{(\ell)} \star v - \left(\psi_h^{(\ell)} \star v\right)^\star\right\|_{L^\infty(\Omega_{\ell+\varepsilon})} + \left\|\left(\psi_h^{(\ell)} \star v\right)^\star\right\|_{L^\infty(\Omega_{\ell+\varepsilon})}. \tag{5.27}$$

To estimate the first term in the right hand side of (5.27), we observe that $\psi_h^{(\ell)} \star v \in W^{\ell,\infty}(\Omega_\ell)$ by Lemma 5.3.2. Hence, we can apply Lemma 5.3.1 (substituting $\psi_h^{(\ell)} \star v$ for u) and rewrite the derivatives using Lemma 5.3.2:

$$\begin{aligned}\left\|\psi_h^{(\ell)} \star v - \left(\psi_h^{(\ell)} \star v\right)^\star\right\|_{L^\infty(\Omega_{\ell+\varepsilon})} &\overset{\text{Lemma 5.3.1}}{\lesssim} h^\ell \sum_{|\alpha|=\ell} \left\|D^\alpha\left(\psi_h^{(\ell)} \star v\right)\right\|_{L^\infty(\Omega_{\ell+\varepsilon})} \\ &\overset{\Omega_{\ell+\varepsilon}\subseteq\Omega_\ell}{\leq} h^\ell \sum_{|\alpha|=\ell} \left\|D^\alpha\left(\psi_h^{(\ell)} \star v\right)\right\|_{L^\infty(\Omega_\ell)} \\ &\overset{\text{Lemma 5.3.2}}{\leq} h^\ell \sum_{|\alpha|\leq\ell} \|\partial_h^\alpha v\|_{L^\infty(\Omega_\alpha)}\end{aligned} \tag{5.28}$$

To estimate the second term in the right hand side of (5.27), we apply Lemma 5.4.1, substituting $\Omega_{\ell+\varepsilon}$ for Ω and d_0 for ℓ. This results in a reduction from $\Omega_{\ell+\varepsilon}$ to $\Omega_{\ell+d_0+2\varepsilon}$:

$$\left\|\left(\psi_h^{(\ell)} \star v\right)^\star\right\|_{L^\infty(\Omega_{\ell+\varepsilon})} \overset{\text{Lemma 5.4.1}}{\lesssim} \left\|\psi_h^{(\ell+d_0)} \star v\right\|_{L^\infty(\Omega_{\ell+d_0+2\varepsilon})}$$

[8] We write $X_0 \Subset X_1$ to indicate that X_0 is a compactly embedded in X_1, i.e. the closure of X_0 is compact and a subset of the interior of X_1.

Next, we switch to L^2-norms using (5.26), after which we can proceed as before for the estimates in the L^2-norm:

$$\begin{aligned}
&\left\|\left(\psi_h^{(\ell)} \star v\right)^\star\right\|_{L^\infty(\Omega_{\ell+\varepsilon})} \\
\overset{(5.26),\ \Omega_{\ell+d_0+2\varepsilon}\Subset\Omega_{\ell+d_0+\varepsilon}}{\lesssim} &\sum_{|\alpha|\le d_0} \left\|D^\alpha\left(\psi_h^{(\ell+d_0)} \star v\right)\right\|_{L^2(\Omega_{\ell+d_0+\varepsilon})} \\
\overset{(5.25),\ \Omega_{\ell+d_0+\varepsilon}\Subset\Omega_{\ell+d_0}}{\lesssim} &\sum_{|\alpha|\le d_0}\sum_{|\beta|\le \ell} \left\|D^{\alpha+\beta}\left(\psi_h^{(\ell+d_0)} \star v\right)\right\|_{H^{-\ell}(\Omega_{\ell+d_0})} \\
\overset{\text{reordering}}{\lesssim} &\sum_{|\alpha|\le \ell+d_0} \left\|D^\alpha\left(\psi_h^{(\ell+d_0)} \star v\right)\right\|_{H^{-\ell}(\Omega_{\ell+d_0})}. \\
\overset{\text{Lemma 5.3.2}}{\lesssim} &\sum_{|\alpha|\le \ell+d_0} \|\partial_h^\alpha v\|_{H^{-\ell}(\Omega_\alpha)}. \qquad (5.29)
\end{aligned}$$

Substituting (5.28) and (5.29) into (5.27) yields (5.24), which then completes the proof.■

5.4.3 The main error estimate in abstract form

Combining the auxiliary results in the previous sections, we can now estimate $\|u - v^\star\|$ in both the L^2- and the L^∞-norm. These estimates are in abstract form in the sense that they are applicable for any (sufficiently smooth) functions u and v. To obtain these results, we follow [11, p. 104–106], while considering the generalized post-processor rather than the symmetric filter. Unlike before, our error estimates are valid in the entire domain, and the L^∞-estimates are not restricted to continuous v.

Theorem 5.4.3 *Consider the generalized post-processor using $r+1$ B-splines of order ℓ, as discussed in Section 5.2.1. Let $k \le r+1$. Then, for all $u \in W^{k,\infty}(\Omega)$ and $v \in L^2(\Omega)$:*

$$\|u - v^\star\|_{L^2(\Omega)} \lesssim \sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)}\, h^k + \sum_{|\alpha|\le\ell} \|\partial_h^\alpha (u-v)\|_{H^{-(\ell)}(\Omega_\alpha)}. \qquad (5.30)$$

Furthermore, for all $u \in W^{k,\infty}(\Omega)$ and $v \in L^\infty(\Omega)$:

$$\begin{aligned}
\|u - v^\star\|_{L^\infty(\Omega)} \lesssim &\sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)}\, h^k + \sum_{|\alpha|\le\ell+d_0} \|\partial_h^\alpha (u-v)\|_{H^{-\ell}(\Omega_\alpha)} \\
&+ \sum_{|\alpha|\le\ell} \|\partial_h^\alpha (u-v)\|_{L^\infty(\Omega_\alpha)}\, h^\ell. \qquad (5.31)
\end{aligned}$$

To show Theorem 5.4.3, the main idea is to apply the triangle inequality to write:

$$\|u - v^\star\| \le \|u - u^\star\| + \|(u-v)^\star\|. \qquad (5.32)$$

After that, we can apply Lemma 5.3.1 to the first term and Lemma 5.4.2 to the second. Altogether, the proof of Theorem 5.4.3 reads as follows:

Proof (of Theorem 5.4.3) To show (5.30), use the triangle inequality and the linearity of the post-processor to write:

$$\begin{aligned}\|u - v^\star\|_{L^2(\Omega)} &\leq \|u - u^\star\|_{L^2(\Omega)} + \|u^\star - v^\star\|_{L^2(\Omega)} \\ &\lesssim \|u - u^\star\|_{L^\infty(\Omega)} + \|(u-v)^\star\|_{L^2(\Omega)} \\ \|u - v^\star\|_{L^\infty(\Omega)} &\leq \|u - u^\star\|_{L^\infty(\Omega)} + \|(u-v)^\star\|_{L^\infty(\Omega)}\end{aligned}$$

Application of Lemma 5.4.1 to the last terms gives:

$$\begin{aligned}\|u - v^\star\|_{L^2(\Omega)} &\lesssim \|u - u^\star\|_{L^\infty(\Omega)} + \left\|\psi_h^{(\ell)} \star (u-v)\right\|_{L^2(\Omega_{\ell+\epsilon})} \\ \|u - v^\star\|_{L^\infty(\Omega)} &\lesssim \|u - u^\star\|_{L^\infty(\Omega)} + \left\|\psi_h^{(\ell)} \star (u-v)\right\|_{L^\infty(\Omega_{\ell+\epsilon})}\end{aligned}$$

Application of Lemma 5.3.1 to the first term and Lemma 5.4.2 to the second term in each inequality yields (5.30) and (5.31), which then completes the proof. ■

5.5 The main result for DG approximations

In the previous section, we obtained error estimates for arbitrary filtered functions. In this section, we study the implications for filtered DG approximations. Section 5.5.1 discusses the convergence of unfiltered DG approximations, including the superconvergence in the negative-order norm established in [22]. Section 5.5.2 uses these results to derive the main theorem of this chapter: the *generalized* post-processor improves the convergence rate of a DG approximation from order $p+1$ to order $2p+1$ in the L^2-norm, and to order $\min\{2p+1, 2p+2-d/2\}$ in the L^∞-norm. Section 5.5.3 discusses why the same convergence rates result for the *position-dependent* post-processor, and explains the accuracy improvement we observed earlier during the numerical experiments in Section 4.5.

5.5.1 Unfiltered DG convergence

A DG approximation with polynomial degree p typically converges at rate $p+1$ in the L^2-norm for sufficiently smooth problems. In the negative-order norm, superconvergence of order $2p+1$ has been shown for the linear periodic hyperbolic problems under consideration [22]. In this section, we summarize these results, including the implications for L^∞-norms and central differences of the error:

Lemma 5.5.1 *Consider the linear periodic hyperbolic problem* (5.8) *with exact solution u and initial data u_0. Suppose that u_h is the DG approximation for u with polynomial degree $p \geq 1$ and mesh size h, as discussed in*

Section 5.2.2. Let α be a d-dimensional multi-index. Then, for all initial data $u_0 \in W^{p+1+|\alpha|,\infty}(\Omega)$ and corresponding u and u_h:

$$\|\partial_h^\alpha (u - u_h)\|_{H^{-(p+1)}(\Omega_\alpha)} \lesssim \|u_0\|_{H^{p+1+|\alpha|}(\Omega)} h^{2p+1}. \tag{5.33}$$

Furthermore, for all $u_0 \in W^{p+2+|\alpha|,\infty}(\Omega)$ and corresponding u and u_h, if the DG scheme yields convergence of order $p+1$ in the sense that

$$\|u - u_h\|_{L^2(\Omega)} \lesssim \|u_0\|_{H^{p+2}(\Omega)} h^{p+1}, \tag{5.34}$$

then,

$$\|\partial_h^\alpha (u - u_h)\|_{L^2(\Omega_\alpha)} \lesssim \|u_0\|_{H^{p+2+|\alpha|}(\Omega)} h^{p+1}, \tag{5.35}$$

$$\|\partial_h^\alpha (u - u_h)\|_{L^\infty(\Omega_\alpha)} \lesssim \max_{|\beta|=p+1} \left\|D^{\beta+\alpha} u\right\|_{L^\infty(\Omega)} h^{p+1-\frac{d}{2}} + \|u_0\|_{H^{p+2+|\alpha|}(\Omega)} h^{p+1-\frac{d}{2}}. \tag{5.36}$$

Before we show these results, we have the following remarks regarding assumption (5.34). First of all, for one-dimensional problems with $A_0 = 0$ in (5.8), relation (5.34) has been shown in [21, p. 166, 189–199]. For *stationary* problems, the same result has been derived for certain two-dimensional triangulations (satisfying a so-called transversality condition) [61], and for d-dimensional meshes with a unique outflow edge per mesh element [19, 85, 20] (with a lower order Sobolev norm in the right hand side). For some stationary problems, a convergence rate of order $p + \frac{1}{2}$, as shown by Johnson and Pitkäranta [44], has shown to be sharp [62]. Nevertheless, convergence of order $p+1$ is usually observed in practice for (5.8) [65, 47]. In any case, assumption (5.34) is not needed to obtain the main error estimate in Section 5.5.2 hereafter in the L^2-norm; only the result in the L^∞-norm relies on it. Altogether, it seems reasonable to require (5.34) at this point.

To show Lemma 5.5.1, we use the following known error estimate in the negative-order norm[9] [22, Theorem 3.3]:

$$\|u - u_h\|_{H^{-(p+1)}(\Omega)} \lesssim \|u_0\|_{H^{p+1}(\Omega)} h^{2p+1}. \tag{5.37}$$

To obtain the L^∞-estimate, we use the following inverse inequality [12, p. 680]:

$$\|v\|_{L^\infty(\Omega)} \lesssim h^{-\frac{d}{2}} \|v\|_{L^2(\Omega)}, \qquad \forall v \in V, \tag{5.38}$$

and the following property of the L^2-projection P_h onto V [18, p. 121-129]:

$$\|v - P_h v\|_{L^\infty(\Omega)} \lesssim \max_{|\beta|=p+1} \left\|D^\beta v\right\|_{L^\infty(\Omega)} h^{p+1}, \quad \forall v \in W^{p+1,\infty}(\Omega). \tag{5.39}$$

[9] Actually, it follows from [22, Theorem 3.3] that $\|u - u_h\|_{H^{-(p+1)}(X_0)} \lesssim \|u_0\|_{H^{p+1}(\Omega)} h^{2p+1}$ for $X_0 \Subset \Omega$. However, the same result is true when we replace X_0 by Ω, due to the periodicity of the problem.

Altogether, Lemma 5.5.1 can be shown as follows:

Proof (of Lemma 5.5.1) Let Ω' be the result of translating Ω (and the corresponding mesh) by a distance $\frac{h}{2}$ in each direction, i.e.

$$\Omega' = \left(\frac{h}{2}, 1+\frac{h}{2}\right) \times ... \times \left(\frac{h}{2}, 1+\frac{h}{2}\right).$$

Next, consider $\partial_h^\alpha u$, $\partial_h^\alpha u_0$ and $\partial_h^\alpha u_h$ on this translated domain (using periodic extensions). Because the problem is linear and periodic, $\partial_h^\alpha u$ is a solution to (5.8) for the domain Ω' (with initial condition $\partial_h^\alpha u_0$). At the same time, $\partial_h^\alpha u_h$ is the DG approximation for $\partial_h^\alpha u$ (on the translated mesh). Hence, we can apply (5.34) and (5.37) for this translated problem on Ω' (also cf. [22, p. 590]):

$$\begin{aligned}\|\partial_h^\alpha (u-u_h)\|_{L^2(\Omega')} &\lesssim \|\partial_h^\alpha u_0\|_{H^{p+2}(\Omega')}\, h^{p+1},\\ \|\partial_h^\alpha (u-u_h)\|_{H^{-(p+1)}(\Omega')} &\lesssim \|\partial_h^\alpha u_0\|_{H^{p+1}(\Omega')}\, h^{2p+1}.\end{aligned}$$

Next, observe that it follows from (5.18), (5.19), (5.20) (and the periodicity) that, for $1 \le p \le \infty$:

$$\|\partial_h^\alpha u\|_{L^p(\Omega')} \lesssim \|D^\alpha u\|_{L^p(\Omega')}. \tag{5.40}$$

With this reasoning, we obtain:

$$\begin{aligned}\|\partial_h^\alpha (u-u_h)\|_{L^2(\Omega')} &\lesssim \|u_0\|_{H^{p+2+|\alpha|}(\Omega')}\, h^{p+1},\\ \|\partial_h^\alpha (u-u_h)\|_{H^{-(p+1)}(\Omega')} &\lesssim \|u_0\|_{H^{p+1+|\alpha|}(\Omega')}\, h^{2p+1}.\end{aligned} \tag{5.41}$$

Using periodicity and translation invariance again, we may replace Ω' by Ω, and we arrive at (5.35) and (5.33) (using $\Omega_\alpha \subseteq \Omega$).

To show (5.36), we consider the translated domain again, and use the triangle inequality and the inverse inequality (5.38) to write:

$$\begin{aligned}&\|\partial_h^\alpha (u-u_h)\|_{L^\infty(\Omega')}\\ \le\ &\|\partial_h^\alpha u - P_h\partial_h^\alpha u\|_{L^\infty(\Omega')} + \|P_h\partial_h^\alpha u - \partial_h^\alpha u_h\|_{L^\infty(\Omega')}\\ \overset{(5.38)}{\lesssim}\ &\|\partial_h^\alpha u - P_h\partial_h^\alpha u\|_{L^\infty(\Omega')} + h^{-\frac{d}{2}}\,\|P_h\partial_h^\alpha u - \partial_h^\alpha u_h\|_{L^2(\Omega')}.\end{aligned}$$

Next, apply the triangle inequality again, and use the property of the polynomial projection (5.39):

$$\begin{aligned}&\|\partial_h^\alpha (u-u_h)\|_{L^\infty(\Omega')}\\ \lesssim\ &\|\partial_h^\alpha u - P_h\partial_h^\alpha u\|_{L^\infty(\Omega')} + h^{-\frac{d}{2}}\,\|P_h\partial_h^\alpha u - \partial_h^\alpha u\|_{L^2(\Omega')}\\ &+h^{-\frac{d}{2}}\,\|\partial_h^\alpha u - \partial_h^\alpha u_h\|_{L^2(\Omega')}\\ \lesssim\ &\left(1+h^{-\frac{d}{2}}\right)\|\partial_h^\alpha u - P_h\partial_h^\alpha u\|_{L^\infty(\Omega')} + h^{-\frac{d}{2}}\,\|\partial_h^\alpha u - \partial_h^\alpha u_h\|_{L^2(\Omega')}\\ \overset{(5.39)}{\lesssim}\ &\left(1+h^{-\frac{d}{2}}\right)\max_{|\beta|=p+1}\left\|D^\beta\partial_h^\alpha u\right\|_{L^\infty(\Omega')} h^{p+1} + h^{-\frac{d}{2}}\,\|\partial_h^\alpha (u-u_h)\|_{L^2(\Omega')}\\ \overset{(5.41),\,(5.40)}{\lesssim}\ &\max_{|\beta|=p+1}\left\|D^{\beta+\alpha}u\right\|_{L^\infty(\Omega')} h^{p+1-\frac{d}{2}} + \|u_0\|_{H^{p+2+|\alpha|}(\Omega')}\, h^{p+1-\frac{d}{2}}.\end{aligned}$$

Using periodicity and translation invariance again, we may replace Ω' by Ω, and we arrive at (5.36) (using $\Omega_\alpha \subseteq \Omega$), which then completes the proof. ∎

5.5.2 Main result: extracting DG superconvergence

We now arrive at the main theorem of this chapter: the generalized post-processor improves the convergence rate of a DG approximation from order $p+1$ to order $2p+1$ in the L^2-norm, and to order $\min\{2p+1, 2p+2-d/2\}$ in the L^∞-norm. This theorem extends existing error estimates for the symmetric post-processor in the L^2-norm [22]. Below, we also include the L^∞-norm in the analysis.

Theorem 5.5.2 (Main result) *Consider the linear periodic hyperbolic problem* (5.8) *with exact solution u and initial data u_0. Suppose that u_h is the DG approximation for u with polynomial degree $p \geq 1$ and mesh size h, as discussed in Section 5.2.2. Furthermore, consider the generalized post-processor with at least $2p+1$ B-splines of order $p+1$ (cf. Section 5.2.1). Then, assuming $u_0, u \in W^{2p+3+[d/2],\infty}(\Omega)$:*

$$\|u - u_h^\star\|_{L^2(\Omega)} \lesssim h^{2p+1}. \tag{5.42}$$

If, furthermore, (5.34) *holds, then,*

$$\|u - u_h^\star\|_{L^\infty(\Omega)} \lesssim h^{\min\{2p+1, 2p+2-d/2\}}. \tag{5.43}$$

The constants involved may depend on u and u_0, but not on u_h (as indicated in the proof below).

To show Theorem 5.5.2, we substitute Lemma 5.5.1 into Theorem 5.4.3:

Proof (of Theorem 5.5.2) Let k be any positive integer not larger than the number of kernel nodes such that $u \in W^{k,\infty}(\Omega)$. Then, substitution of $v = u_h$ and $\ell = p+1$ into (5.30) and (5.31) yields:

$$\begin{aligned}
\|u - u_h^\star\|_{L^2(\Omega)} &\lesssim \sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)} h^k + \sum_{|\alpha|\leq p+1} \|\partial_h^\alpha (u - u_h)\|_{H^{-(p+1)}(\Omega_\alpha)} \\
\|u - u_h^\star\|_{L^\infty(\Omega)} &\lesssim \sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)} h^k + \sum_{|\alpha|\leq p+1+d_0} \|\partial_h^\alpha (u - u_h)\|_{H^{-(p+1)}(\Omega_\alpha)} \\
&\quad + \sum_{|\alpha|\leq p+1} \|\partial_h^\alpha (u - u_h)\|_{L^\infty(\Omega_\alpha)} h^{p+1}.
\end{aligned}$$

Application of Lemma 5.5.1 gives (using $u_0, u \in W^{2p+3+[d/2],\infty}(\Omega)$ by assumption):

$$\begin{aligned}
\|u - u_h^\star\|_{L^2(\Omega)} &\lesssim \sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)} h^k + \sum_{|\alpha|\leq p+1} \|u_0\|_{H^{p+1+|\alpha|}(\Omega)} h^{2p+1} \\
\|u - u_h^\star\|_{L^\infty(\Omega)} &\lesssim \sum_{|\alpha|=k} \|D^\alpha u\|_{L^\infty(\Omega)} h^k + \sum_{|\alpha|\leq p+1+d_0} \|u_0\|_{H^{p+1+|\alpha|}(\Omega)} h^{2p+1} \\
&\quad + \sum_{|\alpha|\leq p+1} \Big(\max_{|\beta|=p+1} \left\| D^{\beta+\alpha} u \right\|_{L^\infty(\Omega)} h^{2p+2-\frac{d}{2}} \\
&\quad + \|u_0\|_{H^{p+2+|\alpha|}(\Omega)} h^{2p+2-\frac{d}{2}} \Big).
\end{aligned}$$

Choosing $k = 2p + 1$, this can be simplified to (5.42) and (5.43), which then completes the proof. ■

5.5.3 Implications for the position-dependent post-processor

Now that we have established error estimates for the generalized post-processor, the same convergence rates follow automatically for the position-dependent post-processor. This is because the latter is based on a convex combination of two generalized post-processors with $2p + 1$ and $4p + 1$ B-splines respectively (cf. Section 4.4.2).

More precisely, we claim that Theorem 5.5.2 is also valid if $u_h^\star$ is the result of applying the position-dependent post-processor to u_h (the constants involved are typically different though). To see this, consider Theorem 5.5.2 and suppose that $u_{h,2p+1}^\star$ and $u_{h,4p+1}^\star$ are the result of applying the generalized post-processor to u_h with $2p + 1$ and $4p + 1$ B-splines respectively. Then, for $\theta : \Omega \to [0, 1]$, we have in any norm:

$$\begin{aligned} \|u - u_h^\star\| &:= \left\| u - \left(\theta\, u_{h,2p+1}^\star + (1-\theta) u_{h,4p+1}^\star \right) \right\| \\ &\leq \left\| u - u_{h,2p+1}^\star \right\| + \left\| u - u_{h,4p+1}^\star \right\|. \end{aligned}$$

Application of Theorem 5.5.2 to both terms then yields (5.42) and (5.43) for the position-dependent post-processor. This means that it improves the convergence rate of a DG approximation from order $p + 1$ to order $2p + 1$ in the L^2-norm, and to order $\min\{2p + 1, 2p + 2 - d/2\}$ in the L^∞-norm.

We can also use the analysis in this chapter to explain the accuracy improvement near the boundary we observed earlier during the numerical experiments in Section 4.5. Similar to (5.32), we may write for the pointwise error in $\bar{x} \in \Omega$:

$$|u - u_h^\star|(\bar{x}) \leq |u - u^\star|(\bar{x}) + |u^\star - u_h^\star|(\bar{x}). \tag{5.44}$$

Following the analysis in the previous sections, the accuracy of the second term is $O(h^{2p+1})$ (in the L^2-norm). For the first term, we can apply the bounds obtained in the proof of Lemma 5.3.1 (we omit the dependency on u for simplicity):

$$|u - u^\star|(\bar{x}) \lesssim \theta(\bar{x}) \|K_{2p+1}\|_{L^1(\mathbb{R}^d)} h^{2p+1} + \big(1 - \theta(\bar{x})\big) \|K_{4p+1}\|_{L^1(\mathbb{R}^d)} h^{4p+1}.$$

Here, K_{2p+1} and K_{4p+1} denote the kernels corresponding to the generalized post-processor with $2p+1$ and $4p+1$ B-splines respectively. From this expression it can be seen that this part of the error is influenced by the L^1-norm of the kernel. The latter becomes larger as the kernel becomes less symmetric. At the same time, we are forced to switch to non-symmetric kernels near the boundary. Using $\theta = 0$ in this region (cf. Section 4.4.2), we can compensate the relatively large L^1-norm of the kernel by a larger order ($4p+1$ rather than $2p+1$). This basically explains our observations in Section 4.5: the use of extra kernel near the boundary yields better accuracy.

5.6 Conclusion

This chapter derives theoretical error estimates for the position-dependent post-processor proposed in Chapter 4 for DG (upwind) approximations for linear hyperbolic problems. We have found that it enhances the accuracy from order $p+1$ to order $2p+1$ in the L^2-norm, and to order $\min\{2p+1, 2p+2-d/2\}$ in the L^∞-norm (where p is the polynomial degree and d is the spatial dimension). This expands the L^2-estimates in [22] for the symmetric post-processor, which cannot be applied near non-periodic boundaries and shocks. Altogether, our theory explains the superconvergence observed during the numerical experiments in Chapter 4, and guarantees similar results for a certain class of linear hyperbolic problems. Furthermore, our abstract formulation can be used to obtain similar error estimates for *any* approximation for which superconvergence in the negative-order norm can be shown.

6 Conclusion

6.1 Introduction

This thesis is focused on the linear systems and hidden accuracy of Discontinuous Galerkin (DG) discretizations. In particular, it discusses the two-level preconditioner in [24], investigates an alternative strategy in the form of a deflation method with the same coarse space, and studies the impact of both techniques on the convergence of the Conjugate Gradient (CG) method for Symmetric Interior Penalty (discontinuous) Galerkin (SIPG) discretizations for diffusion problems. Moreover, this thesis considers the one-sided post-processor in [64], proposes the position-dependent post-processor, and analyzes the impact of both strategies on the accuracy and smoothness of DG (upwind) approximations for hyperbolic problems.

The remaining of this chapter summarizes the main conclusions of this research in the following manner: Section 6.2 considers the two-level methods for solving the linear systems. Section 6.3 discusses post-processing for extracting the hidden accuracy. Finally, Section 6.4 provides suggestions for future research.

6.2 Linear DG systems

We have found that both the two-level preconditioner and the deflation variant yield scalable CG convergence (independent of the mesh element diameter). This has been shown theoretically for any polynomial degree $p \geq 1$, which extends the available analysis for the preconditioning variant for $p = 1$ in [24].

The scalability of both methods is also confirmed by our numerical experiments for various diffusion problems with extreme contrasts in the coefficients. These include problems mimicking bubbly flow, ground water flow, and the presence of layers of sand and shale in oil reservoir simulations. These ex-

periments also demonstrate that both two-level methods only yield fast CG convergence provided that the penalty parameter is chosen dependent on local values of the diffusion coefficient (using the largest limit value at discontinuities). The latter also benefits the accuracy of the SIPG discretization.

At the coarse level, both two-level methods need to solve the same coarse system. The latter is similar to a system resulting from a central difference scheme, for which very efficient solution techniques are readily available. In that sense, both two-level methods transform the original challenging DG system into a more familiar problem. It has been verified that the coarse systems can be solved efficiently by using an inexact solver with relatively low accuracy, such as the CG method combined with a scalable algebraic multigrid preconditioner.

The main difference between both methods is that the deflation method can be implemented by skipping one of the two smoothing steps in the algorithm for the preconditioning variant. This may be particularly advantageous for expensive smoothers, although the basic block Jacobi smoother was found to be highly effective for the problems under consideration. Despite the lower costs per iteration, we have found that the deflation method can require fewer iterations, especially for large problems. As a result, it can be up to 35% faster than the original preconditioner (in terms of the overall computational time in seconds). That is, when damping of the smoother is not taken into account. If an optimal damping parameter is used, both two-level strategies yield similar efficiency (deflation appears unaffected by damping). However, it remains an open question how the damping parameter can best be selected in practice.

Altogether, this work contributes to shorter computational times of DG discretizations, e.g. for oil reservoir simulations. This strengthens the increasing consensus that DG methods can be an effective alternative for classical discretizations schemes, such as the Finite Volume Method (FVM).

6.3 Hidden DG accuracy

We have found that the proposed position-dependent post-processor enhances the DG convergence from order $p+1$ to order $2p+1$. This result has been shown theoretically in both the L^2- and the L^∞-norm (with a slightly lower order in the L^∞-norm for higher-dimensional problems). This expands the L^2-estimates in [22] for the symmetric post-processor, which cannot be applied near non-periodic boundaries and shocks.

The aforementioned superconvergence of order $2p+1$ is also demonstrated by our numerical experiments. These include problems with non-periodic boundary conditions, problems with stationary shocks, a two-dimensional system, and a streamline visualization example. These experiments illustrate that both the position-dependent post-processor and the original one-sided technique produce the same results in the domain interior. In that region, both techniques apply the symmetric post-processor, resulting in the usual high level of accuracy and smoothness.

The differences occur at the boundary of the domain: in those regions, the position-dependent post-processor yields a more realistic smoothness without the previous artificial stair-stepping effect. Furthermore, unlike before, it improves the local accuracy of the DG approximation in the entire domain, not just in order but also in magnitude.

Altogether, this work contributes to more accurate visualization of DG approximations, e.g. in the form of streamlines. Furthermore, it sustains the idea that numerical approximations may contain more information than we originally thought.

6.4 Future research

Suggestions for future research include the following:

1. The comparison of both two-level methods could be continued for more advanced applications, e.g. for three-dimensional unstructured meshes, or a strongly anisotropic diffusion tensor.

2. According to the present study, the coarse systems in the two-level methods can be solved efficiently using an inexact CG solver. To improve the efficiency further, the Flexible CG (FCG) method could be studied to reduce the number of inner iterations (cf. Section 2.4.3).

3. The derived convergence theory for the two-level methods is based on the assumption that the scheme is coercive (or the condition used in Section 3.5.3). To ensure this in applications, practical conditions for the diffusion-dependent penalty parameter could be derived, possibly by applying available global strategies in a local fashion (cf. Section 2.2.3).

4. The application of the two-level deflation method could be extended to non-symmetric DG schemes, such as the Non-symmetric Interior Penalty Galerkin (NIPG) method. The latter could be less sensitive to the choice of the penalty parameter.

5. The theoretical error estimates for the post-processor apply for linear hyperbolic problems with constant coefficients and periodic boundary conditions. Nevertheless, the numerical results in this thesis suggest that it is effective for a larger class of problems, including Dirichlet boundary conditions and variable coefficients. Theoretical support for these findings could be a welcome step towards real-life applications. Additionally, the position-dependent post-processor could be analyzed for unstructured meshes and non-linear problems, e.g. by following [48, 42].

6. For large values of the polynomial degree p, further modification of the position-dependent post-processor may be required to avoid that the kernel support becomes too large to fit the geometric setting, and to avoid

that round-off errors start to dominate (as the magnitude of the one-sided kernel coefficients increases rapidly with p).

7. Numerical results suggest that there exists a relation between the post-processor and the L^2-projection onto the space of piecewise polynomials of degree $2p + 1$ (cf. Section 4.5.7). This could be studied theoretically.

Acknowledgments

For their contributions to this dissertation, I would like to express my sincere gratitude to ...

... Kees Vuik, for being my promotor and supervisor. Kees, we have been working together since I chose “Een ‘lastig’ probleem” for my Bachelor’s thesis. Over the years, you have become like an academic dad to me. Without your faith and support, this dissertation would not have been written.

... Ben de Pagter and Wim van Horssen, for supporting my position as a PhD student at DIAM.

... my committee, for taking the time to evaluate this thesis. Special thanks are due to Yvan Notay, for sending extensive and valuable comments on Chapter 3. Furthermore I am thankful to Jan Dirk Jansen, for giving relevant feedback on early versions of Chapter 2.

... Scott MacLachlan, for offering useful suggestions with respect to damping at Copper Mountain.

... Jan van Neerven, Markus Haase, and Mark Veraar, for helping me with derivations with an inspiring passion for functional analysis.

... Kees Lemmens, for turning all my computer issues into enjoyable mini classes on Linux. I can never go back to Windows now.

... my colleagues in the Numerical Analysis group, for sharing interesting and amusing ideas during tea talks and at the coffee machine. I have warm memories to the times we were celebrating Sinterklaas, having a

Pakistani dinner, and losing with style to our computer science colleagues at karting.

... Aletta Wubben and Louise van Swaaij, for teaching me the soft skills that are easily overlooked at a technical university.

... Cor Kraaikamp, for offering a cup of tea and alternative views at just the right moments.

... my friends and family, for supporting me, inviting me to fun outings, and reminding me of who I was before I started this research.

... Sonja Cox, for being my paranymph and for analyzing the world with me, including ourselves and, yes, our analyzing habits... Sonja, you are a wonderful friend and I hope you will move back to the Netherlands soon.

... my parents, Gé and Mary van Slingerland, for giving me "een goede basis" these last thirty years. Mom, dad, your unconditional love and support, like on that day when it was snowing heavily, move me every time. To quote the song, "You are the wind beneath my wings".

... Ewoud Marijt, for being my paranymph and for encouraging me to think in terms of possibilities. Ewoud, you always manage to make me smile, even if I, for some reason, try not to. I am incredibly lucky to have you by my side.

Paulien van Slingerland
Delft, May 2013

Curriculum Vitae

Paulien van Slingerland was born on May 19, 1983, in Leiderdorp, The Netherlands. After completing her secondary education at the Stedelijk Gymnasium Leiden in 2001, she enrolled for Applied Mathematics at Delft University of Technology. She was awarded the CIVI aanmoedigingsprijs for her propaedeutics diploma (cum laude) in 2002, after which she obtained her MSc. degree (cum laude) in 2007. The time integration scheme that was the result of her Master's thesis is still being used by Deltares for simulating water quality. In September 2007, Paulien started working as a PhD student at Delft University of Technology. Initially, she studied coastal wave modeling in the Fluid Mechanics group (Faculty of Civil Engineering and Geosciences). After a year she switched to the Numerical Analysis group (Faculty of Electrical Engineering, Mathematics and Computer Science), which has resulted in this thesis, four refereed journal papers (one is accepted, three are in submission), and a prize for the best poster presentation during the thirty-sixth Woudschoten conference of the Werkgemeenschap Scientific Computing. Since January 2013, Paulien is working at TNO as a trainee.

Publications

Journal papers

- P. van Slingerland, C. Vuik. *Scalable two-level preconditioning and deflation based on a piecewise constant subspace for (SIP)DG systems.* Submitted to JCAM.

- P. van Slingerland, C. Vuik. *Fast linear solver for pressure computation in layered domains.* Submitted to Comput. Geosci.

- L. Ji, P. van Slingerland, J.K. Ryan, C. Vuik. *Superconvergent error estimates for position-dependent smoothness-increasing accuracy-conserving (SIAC) post-processing of Discontinuous Galerkin Solutions.* Accepted for publication in Math. Comp.

- P. van Slingerland, J.K. Ryan, C. Vuik. *Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering for improving Discontinuous Galerkin solutions.* SIAM J. Sci. Comp., **33**(2011), pp 802–825.

Technical reports

- P. van Slingerland, C. Vuik. *Scalable two-level preconditioning and deflation based on a piecewise constant subspace for (SIP)DG systems.* DIAM report 12-11, Delft University of Technology, 2012.

- P. van Slingerland, C. Vuik. *Fast linear solver for pressure computation in layered domains.* DIAM report 12-10, Delft University of Technology, 2012.

- P. Slingerland, C. Vuik. *Spectral two-level deflation for DG: a preconditioner for CG that does not need symmetry.* DIAM report 11-12, Delft University of Technology, 2011.

- P. van Slingerland, J.K. Ryan, C. Vuik. *Smoothness-increasing convergence-conserving spline filters applied to streamline visualisation of DG approximations.* DIAM report 09-06, Delft University of Technology, 2009.

- P. van Slingerland, M. Borsboom, C. Vuik. *A local theta scheme for advection problems with strongly varying meshes and velocity profiles.* DIAM report 08-17, Delft University of Technology, 2008.

Talks at international conferences

- *Fast linear solver for pressure computation in layered domains.* 13th European conference on the mathematics of oil recovery (ECMOR). Biarritz, France, 2012.

- *A preconditioner for CG that does not need symmetry.* Twelfth Copper Mountain conference on iterative methods (COPPER). Copper Mountain (Colorado), United States of America, 2012.

- *Exploiting the nested block structure of DG matrices: a block ILU preconditioner with deflation and a spectral two-level strategy.* International conference on preconditioning techniques for scientific and industrial applications (PRECOND). Bordeaux, France, 2011.

- *A local theta scheme for advection problems with strongly varying meshes and velocity profiles.* The mathematics of finite elements and applications (MAFELAP). Brunel University, Londen, England, 2009.

- *A robust higher-order variable-θ scheme for the advection diffusion equation on unstructured grids.* 2nd international conference on high order non-oscillatory methods for wave propagation, transport and flow problems. Trento, Italy, 2007.

Other talks

- *Spectral two-level deflation for DG: a preconditioner for CG that does not need symmetry ....* Discontinuous Galerkin methods in computational electromagnetics: a workshop on recent developments in theory and applications. National Aerospace Laboratory of the Netherlands, Amsterdam, The Netherlands, 2011.

- *Extracting the hidden accuracy of DG solutions.* Spring meeting Werkgemeenschap Scientific Computing. Antwerp, Belgium, 2010.

- *A local theta scheme for advection (dominated) problems with strongly varying meshes and velocity profiles.* Invited talk at the Institute of Applied Mathematics. Dortmund University of Technology, Dortmund, Germany, 2009.

- *An accurate and robust local theta FCT scheme for the advection equation for strongly varying meshes and velocity profiles.* Meeting Kontactgroep Numerieke Stromingsleer. University of Twente, Enschede, The Netherlands, 2007.

Poster presentations

- *A preconditioner for CG that does not need symmetry.* NWO-JSPS joint seminar: numerical linear algebra - algorithms, applications, and training. Delft University of Technology, Delft, 2012.

- *A preconditioner for CG that does not need symmetry.* Thirty-sixth Woudschoten conference, Werkgemeenschap Scientific Computing. Zutphen, The Netherlands, 2011.

- *The hidden accuracy of DG.* Thirty-fifth Woudschoten conference, Werkgemeenschap Scientific Computing. Zutphen, The Netherlands, 2010.

- *Post-processing for DG: improving the accuracy near boundaries and shocks.* Opening symposium Applied Mathematics Institute. Delft University of Technology, Delft, The Netherlands, 2010.

- *Post-processing for DG.* Thirty-fourth Woudschoten conference, Werkgemeenschap Scientific Computing. Zutphen, The Netherlands, 2009.

- *Smoothness-increasing accuracy-conserving spline filters applied to streamline visualisation of DG approximations.* Burgersdag 2009, J.M. Burgerscentrum. Eindhoven University of Technology, Eindhoven, The Netherlands, 2009.

Bibliography

[1] S. Adjerid, K. D. Devine, J. E. Flaherty, and Lilia Krivodonova. A posteriori error estimation for discontinuous Galerkin solutions of hyperbolic problems. *Comput. Methods Appl. Mech. Engrg.*, 191(11-12):1097–1112, 2002.

[2] S. Adjerid and T. C. Massey. Superconvergence of discontinuous Galerkin solutions for a nonlinear scalar hyperbolic problem. *Comput. Methods Appl. Mech. Engrg.*, 195(25-28):3331–3346, 2006.

[3] P. F. Antonietti and B. Ayuso. Schwarz domain decomposition preconditioners for discontinuous Galerkin approximations of elliptic problems: non-overlapping case. *M2AN Math. Model. Numer. Anal.*, 41(1):21–54, 2007.

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

[5] O. Axelsson. *Iterative solution methods.* Cambridge University Press, Cambridge, 1994.

[6] O. Axelsson and P. S. Vassilevski. Variable-step multilevel preconditioning methods. I. Selfadjoint and positive definite elliptic problems. *Numer. Linear Algebra Appl.*, 1(1):75–101, 1994.

[7] B. Ayuso de Dios, M. Holst, Y. Zhu, and L. Zikatanov. Multilevel preconditioners for discontinuous Galerkin approximations of elliptic problems with jump coeffients. arXiv:1012.1287v2, 2012.

[8] D.H. Bailey, Y. Hida, X. S. Li, and B. Thompson. ARPREC: an arbitrary precision computation package. Technical Report 53651, Lawrence Berkeley National Laboratory, September 2002.

[9] V. I. Bogachev. *Measure theory*. Springer-Verlag, Berlin, 2007.

[10] J. H. Bramble and A. H. Schatz. Estimates for spline projections. *Rev. Française Automat. Informat. Recherche Opérationnelle*, 10(R-2):5–37, 1976.

[11] J. H. Bramble and A. H. Schatz. Higher order local accuracy by averaging in the finite element method. *Math. Comp.*, 31(137):94–111, 1977.

[12] James H. Bramble, Joachim A. Nitsche, and Alfred H. Schatz. Maximum-norm interior estimates for Ritz-Galerkin methods. *Math. Comput.*, 29:677–688, 1975.

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

[14] V. I. Burenkov. *Sobolev spaces on domains*, volume 137 of *Teubner-Texte zur Mathematik [Teubner Texts in Mathematics]*. B. G. Teubner Verlagsgesellschaft mbH, Stuttgart, 1998.

[15] E. Burman and P. Zunino. A domain decomposition method based on weighted interior penalties for advection-diffusion-reaction problems. *SIAM J. Numer. Anal.*, 44(4):1612–1638 (electronic), 2006.

[16] W. Cai, D. Gottlieb, and C.-W. Shu. On one-sided filters for spectral Fourier approximations of discontinuous functions. *SIAM J. Numer. Anal.*, 29(4):905–916, 1992.

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

[18] P. G. Ciarlet. *The finite element method for elliptic problems*, volume 4 of *Studies in mathematics and its applications*. North-Holland, New York, 1978.

[19] B. Cockburn, B. Dong, and J. Guzmán. Optimal convergence of the original DG method for the transport-reaction equation on special meshes. *SIAM J. Numer. Anal.*, 46(3):1250–1265, 2008.

[20] B. Cockburn, B. Dong, J. Guzmán, and J. Qian. Optimal convergence of the original DG method on special meshes for variable transport velocity. *SIAM J. Numer. Anal.*, 48(1):133–146, 2010.

[21] B. Cockburn, C. Johnson, C.-W. Shu, and E. Tadmor. *Advanced numerical approximation of nonlinear hyperbolic equations*, volume 1697 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1998. Papers from the C.I.M.E. Summer School held in Cetraro, June 23–28, 1997, Edited by Alfio Quarteroni, Fondazione C.I.M.E.. [C.I.M.E. Foundation].

[22] B. Cockburn, M. Luskin, C.-W. Shu, and E. Süli. Enhanced accuracy by post-processing for finite element methods for hyperbolic equations. *Math. Comp.*, 72(242):577–606, 2003.

[23] S. Curtis, R. M. Kirby, J. K. Ryan, and C.-W. Shu. Postprocessing for the discontinuous Galerkin method over nonuniform meshes. *SIAM J. Sci. Comput.*, 30(1):272–289, 2007.

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

[25] Veselin A. Dobrev, Raytcho D. Lazarov, and Ludmil T. Zikatanov. Preconditioning of symmetric interior penalty discontinuous Galerkin FEM for elliptic problems. In *Domain decomposition methods in science and engineering XVII*, volume 60 of *Lect. Notes Comput. Sci. Eng.*, pages 33–44. Springer, Berlin, 2008.

[26] Zdeněk Dostál. Conjugate gradient method with preconditioning by projector. *International Journal of Computer Mathematics*, 23(3-4):315–323, 1988.

[27] Maksymilian Dryja. On discontinuous Galerkin methods for elliptic problems with discontinuous coefficients. *Comput. Methods Appl. Math.*, 3(1):76–85 (electronic), 2003. Dedicated to Raytcho Lazarov.

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

[29] A. Ern, A.F. Stephansen, and P. Zunino. A discontinuous Galerkin method with weighted averages for advection-diffusion equations with locally small and anisotropic diffusivity. *IMA J. Numer. Anal.*, 29(2):235–256, 2009.

[30] L. C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1998.

[31] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov. On two-grid convergence estimates. *Numer. Linear Algebra Appl.*, 12(5-6):471–494, 2005.

[32] X. Feng and O. A. Karakashian. Two-level additive Schwarz methods for a discontinuous Galerkin approximation of second order elliptic problems. *SIAM J. Numer. Anal.*, 39(4):1343–1365 (electronic), 2001.

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

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

[35] D. Gottlieb, C.-W. Shu, A. Solomonoff, and H. Vandeven. On the Gibbs phenomenon. I. Recovering exponential accuracy from the Fourier partial sum of a nonperiodic analytic function. *J. Comput. Appl. Math.*, 43(1-2):81–98, 1992.

[36] S. Gottlieb and C.-W. Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation*, 67:73–85, 1998.

[37] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability preserving high-order time discretization methods. *SIAM Review*, 43:89–112, 2001.

[38] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time-dependent problems*, volume 21 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2007.

[39] J.S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods*, volume 54 of *Texts in Applied Mathematics*. Springer, New York, 2008. Algorithms, analysis, and applications.

[40] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, 1988.

[41] Liangyue Ji, Yan Xu, and Jennifer K. Ryan. Accuracy-enhancement of discontinuous Galerkin solutions for convection-diffusion equations in multiple-dimensions. *Math. Comp.*, 81(280):1929–1950, 2012.

[42] Liangyue Ji, Yan Xu, and Jennifer K Ryan. Negative-order norm estimates for nonlinear hyperbolic conservation laws. *J. Sci. Comput.*, 54(2-3):531–548, 2013.

[43] K. Johannsen. A symmetric smoother for the nonsymmetric interior penalty discontinuous Galerkin discretization. Technical Report ICES Report 05-23, University of Texas at Austin, 2005.

[44] C. Johnson and J. Pitkäranta. An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation. *Math. Comp.*, 46(173):1–26, 1986.

[45] R. B. Lowrie. *Compact higher-order numerical methods for hyperbolic conservation laws*. PhD thesis, University of Michigan, 1996.

[46] Lois Mansfield. On the use of deflation to improve the convergence of conjugate gradient iteration. *Communications in Applied Numerical Methods*, 4(2):151–156, 1988.

[47] H. Mirzaee, L. Ji, J. K. Ryan, and R. M. Kirby. Smoothness-increasing accuracy-conserving (SIAC) postprocessing for discontinuous Galerkin solutions over structured triangular meshes. *SIAM J. Numer. Anal.*, 49(5):1899–1920, 2011.

[48] H. Mirzaee, J. King, J.K. Ryan, and R.M. Kirby. Smoothness-increasing accuracy-conserving filters for discontinuous Galerkin solutions over unstructured triangular meshes. *SIAM J. Sci. Comput.*, 35(1):A212–A230, 2013.

[49] H. Mirzaee, J. K. Ryan, and R. M. Kirby. Quantification of errors introduced in the numerical approximation and implementation of smoothness-increasing accuracy conserving (SIAC) filtering of discontinuous Galerkin (DG) fields. *J. Sci. Comput.*, 45(1-3):447–470, 2010.

[50] H. Mirzaee, J. K. Ryan, and R. M. Kirby. Efficient implementation of smoothness-increasing accuracy-conserving (SIAC) filters for discontinuous Galerkin solutions. *J. Sci. Comput.*, 52(1):85–112, 2012.

[51] M. S. Mock and P. D. Lax. The computation of discontinuous solutions of linear hyperbolic equations. *Comm. Pure Appl. Math.*, 31(4):423–430, 1978.

[52] R. Nabben and C. Vuik. A comparison of deflation and coarse grid correction applied to porous media flow. *SIAM J. Numer. Anal.*, 42(4):1631–1647 (electronic), 2004.

[53] R. Nabben and C. Vuik. A comparison of deflation and the balancing preconditioner. *SIAM J. Sci. Comput.*, 27(5):1742–1759 (electronic), 2006.

[54] R. Nabben and C. Vuik. A comparison of abstract versions of deflation, balancing and additive coarse grid correction preconditioners. *Numerical Linear Algebra with Applications*, 15(4):355–372, 2008.

[55] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numer. Anal.*, 24(2):355–365, 1987.

[56] Y. Notay. Flexible conjugate gradients. *SIAM J. Sci. Comput.*, 22(4):1444–1460 (electronic), 2000.

[57] Yvan Notay. Algebraic analysis of two-grid methods: The nonsymmetric case. *Numer. Linear Algebra Appl.*, 17(1):73–96, 2010.

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

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

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

[61] G.R. Richter. An optimal-order error estimate for the discontinuous Galerkin method. *Math. Comp.*, 50(181):75–88, 1988.

[62] G.R. Richter. On the order of convergence of the discontinuous Galerkin method for hyperbolic equations. *Math. Comp.*, 77(264):1871–1885, 2008.

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

[64] J. K. Ryan and C.-W. Shu. On a one-sided post-processing technique for the discontinuous Galerkin methods. *Methods Appl. Anal.*, 10(2):295–307, 2003.

[65] J. K. Ryan, C.-W. Shu, and H. Atkins. Extension of a postprocessing technique for the discontinuous Galerkin method for hyperbolic equations with application to an aeroacoustic problem. *SIAM J. Sci. Comput.*, 26(3):821–843, 2005.

[66] J.K. Ryan and B. Cockburn. Local derivative post-processing for the discontinuous Galerkin method. *J. Comput. Phys.*, 228(23):8642–8664, 2009.

[67] Y. Saad. Iterative methods for sparse linear systems. This is a revised version of the book published in 1996 by PWS Publishing, Boston. It can be downloaded from http://www-users.cs.umn.edu/ saad/books.html, 2000.

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

[69] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM J. Sci. Comput.*, 21(5):1909–1926, December 1999.

[70] I. J. Schoenberg. *Cardinal spline interpolation*. SIAM, Philadelphia, Pa., 1973.

[71] L. L. Schumaker. *Spline functions: basic theory*. John Wiley & Sons Inc., New York, 1981.

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

[73] M. Steffen, S. Curtis, R. M. Kirby, and J. K. Ryan. Investigation of smoothness-increasing accuracy-conserving filters for improving streamline integration through discontinuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 14:680–692, 2008.

[74] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. W. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–532. Academic Press, 2001.

[75] J. M. Tang, S. P. MacLachlan, R. Nabben, and C. Vuik. A comparison of two-level preconditioners based on multigrid and deflation. *SIAM J. Matrix Anal. Appl.*, 31(4):1715–1739, 2010.

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

[77] V. Thomée. High order local approximations to derivatives in the finite element method. *Math. Comp.*, 31(139):652–660, 1977.

[78] V. Thomée. Negative norm estimates and superconvergence in Galerkin methods for parabolic problems. *Math. Comp.*, 34(149):93–113, 1980.

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

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

[81] C. Vuik, A. Segal, J.A. Meijerink, and G.T. Wijma. The construction of projection vectors for a Deflated ICCG method applied to problems with extreme contrasts in the coefficients. *Journal of Computational Physics*, 172:426–450, 2001.

[82] D. Walfish, J. K. Ryan, R. M. Kirby, and R. Haimes. One-sided smoothness-increasing accuracy-conserving filtering for enhanced streamline integration through discontinuous fields. *Journal of Scientific Computing*, 38:164–184, 2009.

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

[84] I. Yavneh. Why multigrid methods are so efficient. *Computing in Science & Engineering*, 8(6):12–22, 2006.

[85] T. Zhang and Z. Li. Optimal error estimate and superconvergence of the DG method for first-order hyperbolic problems. *J. Comput. Appl. Math.*, 235(1):144–153, 2010.