Interior penalty tensor-product preconditioners for high-order discontinuous Galerkin discretizations

Will Pazner*

Brown University, 182 George St., Providence, RI, 02912, U.S.A.

Per-Olof Persson†

University of California, Berkeley, Berkeley, CA 94720-3840, U.S.A.

In this paper, we introduce an interior penalty tensor-product preconditioner for the implicit time integration of discontinuous Galerkin discretizations of partial differential equations with second-order spatial derivatives. This preconditioner can be efficiently formed using a sum-factorized Lanczos algorithm for computing the Kronecker-product singular value decomposition of the diagonal blocks of the Jacobian matrix, and can be applied efficiently using a simultaneous triangularization procedure. In two spatial dimensions, the computational complexity for the overall method is linear per degree of freedom, which is the same as that of a sum-factorized explicit method. This preconditioner exactly reproduces the block Jacobi preconditioner for certain special cases, and compares favorably with the block Jacobi preconditioner for a range of test problems, including viscous compressible flow over a circular cylinder. This preconditioner shows greatly improved performance when compared with a Kronecker-product preconditioner that only incorporates first-order terms.

I. Introduction

The discontinuous Galerkin (DG) method, first introduced by Reed and Hill,[26] is a finite element method whose basis functions are piecewise polynomials with discontinuities along element boundaries. The DG method was subsequently extended to systems of hyperbolic conservation laws and elliptic problems.[8,9] In recent years, there has been considerable interest in the application of DG to computational fluid dynamics problems, including direct numerical simulation (DNS) and large eddy simulation (LES).[5,23,24] An important advantage of the DG method is the ability to use arbitrary-degree polynomial functions, and thus obtain arbitrarily high spatial order of accuracy. However, the use of high-degree polynomials results in a CFL stability condition that can render explicit time stepping methods impractical, motivating the use of implicit solvers. The linear systems resulting from the implicit time integration of DG discretizations have a natural block-sparse structure, with dense elementwise blocks whose size scales as $p^d \times p^d$, requiring $\mathcal{O}(p^{2d})$ storage, where p is the degree of polynomial approximation, and d is the number of spatial dimensions. Furthermore, the solution of such systems by means of standard iterative methods typically makes use of block preconditioners such as block Jacobi, block Gauss-Seidel, or block ILU.[7,10,25] These preconditioners require the inversion of dense elementwise blocks (at least along the diagonal), incurring a cost of $\mathcal{O}(p^{3d})$, which quickly becomes intractable for large p.

In our previous works,[20,21] we introduced a new Kronecker-product based preconditioner that exploits the tensor-product structure of nodal discontinuous Galerkin discretizations on quadrilateral or hexahedral meshes. This preconditioner generates optimal tensor-product approximations to the diagonal blocks of the Jacobian matrix, with the advantage that it greatly reduces the computational complexity associated with forming and applying it when compared with the exact block Jacobi method. This reduction in computational

*Ph.D. Student, Division of Applied Mathematics, Brown University, Providence, RI, 02912. E-mail: will_pazner@brown.edu. AIAA Student Member.

†Associate Professor, Department of Mathematics, University of California, Berkeley, Berkeley CA 94720-3840. E-mail: persson@berkeley.edu. AIAA Senior Member.

complexity makes it feasible to use implicit time integration methods together with DG discretizations with very high polynomial degree p. One limitation in the previous works is that the preconditioner uses only the inviscid fluxes in forming its approximation. This restriction is due to a technical limitation related to the lifting operators required by many DG discretizations of second-order terms, such as LDG, BR2, and CDG.[3] In this paper, we extend the tensor-product preconditioner to incorporate second-order terms by making use of the symmetric interior penalty method.[2,14] This method has the advantage that the lifting operators are not required in order to compute the primal form, and thus avoid the computational pitfalls of the other methods.

In this paper, we describe efficient algorithms for forming and applying the tensor-product preconditioner, fully incorporating the viscous fluxes. We analyze the effectiveness of the preconditioner by demonstrating that it exactly reproduces the diagonal blocks for several simple model problems such as the Poisson problem and convection-diffusion on Cartesian grids in 2D. Then we perform numerical experiments, applying this preconditioner to two benchmark convection-diffusion test cases, and to the compressible Navier-Stokes. A comparison with the traditional block Jacobi preconditioner allows us to evaluate the effectiveness of the preconditioner.

II. Governing equations and spatial discretization

Let the spatial domain $\Omega \subseteq \mathbb{R}^d$ be fixed, where $d = 2, 3$. We consider a second-order time-dependent system of partial differential equations in conservative form given by

$$\boldsymbol{u}_t + \nabla \cdot F(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{f}, \tag{1}$$

where $\boldsymbol{u}(\boldsymbol{x}, t)$ is a vector of n_c unknown functions, F is a given flux function, and $\boldsymbol{f}$ is a forcing term. Appropriate boundary conditions are specified on $\partial\Omega$. It is convenient to make use of the (non-unique) splitting of the flux into inviscid and viscous parts,

$$\boldsymbol{u}_t + \nabla \cdot (F_i(\boldsymbol{u}) + F_v(\boldsymbol{u}, \nabla \boldsymbol{u})) = \boldsymbol{f} \tag{2}$$

For the moment, we consider the case when $F_v = 0$, and we obtain a system of hyperbolic conservation laws. We discretize this system of equations by means of the discontinuous Galerkin method. Let $\mathcal{T}$ be a tessellation of the domain Ω by non-overlapping mesh elements, which in this work we will take to be mapped quadrilaterals in $\mathbb{R}^2$ or hexahedra in $\mathbb{R}^3$. For each mesh element $K \in \mathcal{T}$, let $\mathcal{P}^p_d(K)$ denote the space of d-fold tensor product polynomials of degree at most p on K. Then, we define our discontinuous finite element function space to be

$$V_h = \{v_h : v_h|_K \in \mathcal{P}^p_d(K) \text{ for all } K \in \mathcal{T}\}. \tag{3}$$

We look for an approximate solution $\boldsymbol{u}_h \in [V_h]^{n_c}$ to (2) by multiplying by a test function $\boldsymbol{v}_h \in [V_h]^{n_c}$, and integrating by parts over each element to obtain the standard DG method

$$\int_\Omega \partial_t \boldsymbol{u}_h \cdot \boldsymbol{v}_h \, dx - \int_\Omega F(\boldsymbol{u}_h) : \nabla \boldsymbol{v}_h \, dx + \int_\Gamma \widehat{F}(\boldsymbol{u}_h^-, \boldsymbol{u}_h^+) : [\![\boldsymbol{v}_h]\!] \, ds = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}_h \, dx, \tag{4}$$

where $\Gamma = \bigcup_{K \in \mathcal{T}} \partial K$ denotes the set of all interior and exterior edges of the tessellation $\mathcal{T}$. Here we introduce the numerical flux function $\widehat{F}$, which is defined in terms of the traces $\boldsymbol{u}_h^-$ and $\boldsymbol{u}_h^+$ of the solution $\boldsymbol{u}_h$ on the elements K^- and K^+ bordering the given edge. On an exterior edge, $\boldsymbol{u}_h^+$ is determined by the relevant boundary conditions. We also define the jump of a vector to be the scalar quantity given by $[\![\boldsymbol{q}]\!] = \boldsymbol{q}^- \cdot \boldsymbol{n}^- + \boldsymbol{q}^+ \cdot \boldsymbol{n}^+$, and the jump of a scalar quantity is a vector defined by $[\![\phi]\!] = \phi^- \boldsymbol{n}^- + \phi^+ \boldsymbol{n}^+$.

II.A. Treatment of second-order terms

In order to describe the discretization of the second-order term in equation (2), we first consider the simpler case of the model Poisson problem on the domain Ω,

$$\begin{aligned} -\nabla \cdot \nabla u &= f && \text{in } \Omega, \\ u &= g_D && \text{on } \partial\Omega_D, \\ \partial u / \partial n &= g_N && \text{on } \partial\Omega_N. \end{aligned} \tag{5}$$

We transform equation (5) into a system of first-order equations by introducing the auxiliary variable $\boldsymbol{\sigma} = \nabla u$ to obtain

$$-\nabla \cdot \boldsymbol{\sigma} = f, \tag{6}$$

$$\boldsymbol{\sigma} = \nabla u. \tag{7}$$

Repeating the above Galerkin procedure, we obtain the discretized system of equations

$$\int_\Omega \boldsymbol{\sigma}_h \cdot \nabla v_h \, dx - \int_\Gamma \widehat{\boldsymbol{\sigma}} \cdot [\![v_h]\!] \, ds = \int_\Omega f v_h \, dx, \tag{8}$$

$$\int_\Omega \boldsymbol{\sigma}_h \cdot \boldsymbol{\tau}_h \, dx = -\int_\Omega u_h \nabla \cdot \boldsymbol{\tau}_h \, dx + \int_\Gamma \widehat{u} [\![\boldsymbol{\tau}_h]\!] \, ds, \tag{9}$$

where $\widehat{u}$ and $\widehat{\boldsymbol{\sigma}}$ are yet-to-be-defined numerical flux functions. A comprehensive study of possible choices for these fluxes was performed by Arnold et al.,[3] and specific choices of stable and consistent fluxes include the local DG method,[8] the compact DG method,[22] the second method of Bassi and Rebay,[4] and the interior penalty method.[2] It is possible to generalize this technique to discretize equations of the form (2) with $F_v \neq 0$.

Equations (8) and (9) are often referred to as the *system flux formulation* of the method. It is generally useful to eliminate the auxiliary variable $\boldsymbol{\sigma}_h$ in order to obtain what is known as the *primal formulation* of the method. To this end, we apply elementwise integration by parts to obtain the identity

$$-\int_\Omega u_h \nabla \cdot \boldsymbol{\tau}_h \, dx = \int_\Omega \nabla(u_h) \cdot \boldsymbol{\tau}_h \, dx - \int_\Gamma [\![u_h]\!] \cdot \{ \boldsymbol{\tau}_h \} \, ds - \int_{\Gamma \setminus \partial\Omega} \{ u_h \} [\![\boldsymbol{\tau}_h]\!] \, ds. \tag{10}$$

Then, we replace the first term on the right-hand side of (9) with the right-hand side of (10) resulting in the equation

$$\int_\Omega \boldsymbol{\sigma}_h \cdot \boldsymbol{\tau}_h \, dx = \int_\Omega \nabla(u_h) \cdot \boldsymbol{\tau}_h \, dx - \int_\Gamma [\![u_h]\!] \cdot \{ \boldsymbol{\tau}_h \} \, ds - \int_{\Gamma \setminus \partial\Omega} \{ u_h - \widehat{u} \} [\![\boldsymbol{\tau}_h]\!] \, ds. \tag{11}$$

At this point we define the *lifting operators* $r : [L^2(\Gamma)]^d \to [V_h]^d$ and $\ell : L^2(\Gamma \setminus \partial\Omega) \to [V_h]^d$ by

$$\int_\Omega r(\boldsymbol{q}) \cdot \boldsymbol{\tau} \, dx = -\int_\Gamma \boldsymbol{q} \cdot \{ \boldsymbol{\tau} \} \, ds, \qquad \int_\Omega \ell(v) \cdot \boldsymbol{\tau} \, dx = -\int_{\Gamma \setminus \partial\Omega} v [\![\boldsymbol{\tau}]\!] \, ds. \tag{12}$$

Given these definitions, we see that equation (11) implies that

$$\boldsymbol{\sigma}_h = \nabla(u_h) + r([\![u_h]\!]) + \ell(\{ u_h - \widehat{u} \}). \tag{13}$$

Thus, having expressed $\boldsymbol{\sigma}_h$ explicitly in terms of u_h, we can rewrite the system of equations (8–9) as

$$B(u_h, v_h) = \int_\Omega f v_h \, dx, \tag{14}$$

where the bilinear form $B(\cdot, \cdot)$, called the *primal form*, is given by

$$B(u_h, v_h) = \int_\Omega \nabla(u_h) \cdot \nabla v_h \, dx - \int_\Gamma [\![u_h]\!] \cdot \{ \nabla v_h \} \, ds - \int_{\Gamma \setminus \partial\Omega} \{ u_h - \widehat{u} \} [\![\nabla v_h]\!] \, ds - \int_\Gamma \widehat{\boldsymbol{\sigma}} \cdot [\![v_h]\!] \, ds. \tag{15}$$

In the following section, we will discuss the impact of the choice of fluxes on the construction of approximate tensor-product preconditioners.

Of particular interest in this work is the interior penalty method, which can be obtained by choosing the numerical fluxes

$$\widehat{u} = \{ u_h \}, \qquad \widehat{\boldsymbol{\sigma}} = \{ \nabla u_h \} - \eta_e [\![u_h]\!], \tag{16}$$

for a given edge-specific penalty parameter η_e that scales as p^2/h, and can be chosen optimally by means of an explicit formula.[27] The primal form of the interior penalty method is given by

$$B(u_h, v_h) = \int_\Omega \nabla(u_h) \cdot \nabla v_h \, dx - \int_\Gamma [\![u_h]\!] \cdot \{ \nabla v_h \} \, ds - \int_\Gamma \{ \nabla u_h \} \cdot [\![v_h]\!] \, ds + \int_\Gamma \eta_e [\![u_h]\!] \cdot [\![v_h]\!] \, ds. \tag{17}$$

We point out that, in contrast to most of the methods presented in Reference 3, the primal form of the interior penalty method does not require the computation of lifting operators, which will prove to be computationally advantageous for the construction of tensor-product preconditioners.[12, 15]

II.A.1. Generalized formulation

It is possible to extend the interior penalty method to give discretizations for a wide range of equations, including the compressible Navier-Stokes equations.[14] We begin by assuming that the viscous flux F_v in (2) is linear in the gradient of $\boldsymbol{u}$, and hence can be written as

$$F_v^i(\boldsymbol{u}, \nabla \boldsymbol{u}) = \sum_{j=1}^d G_{ij}(\boldsymbol{u}) \frac{\partial \boldsymbol{u}}{\partial x_j}, \tag{18}$$

where $G_{ij}(\boldsymbol{u})$ is a $n_c \times n_c$ matrix whose entries are given by arbitrary functions of $\boldsymbol{u}$. We then transform (2) into the system

$$\boldsymbol{u}_t + \nabla \cdot (F_i(\boldsymbol{u}) + \boldsymbol{\sigma}) = \boldsymbol{f}, \tag{19}$$

$$\boldsymbol{\sigma} = F_v(\boldsymbol{u}, \nabla \boldsymbol{u}) = \sum_{j=1}^d G_{ij}(\boldsymbol{u}) \frac{\partial \boldsymbol{u}}{\partial x_j}. \tag{20}$$

As in the case of the Poisson problem, we obtain the flux formulation by multiplying by test functions $\boldsymbol{v}_h$ and $\boldsymbol{\tau}_h$, and integrating by parts

$$\begin{aligned} \int_\Omega \partial_t \boldsymbol{u}_h \cdot \boldsymbol{v}_h \, dx - \int_\Omega F_i(\boldsymbol{u}_h) : \nabla \boldsymbol{v}_h \, dx + \int_\Gamma \widehat{F}_i(\boldsymbol{u}_h^+, \boldsymbol{u}_h^-) : [\![\boldsymbol{v}_h]\!] \, ds \\ - \int_\Omega \boldsymbol{\sigma}_h : \nabla \boldsymbol{v}_h \, dx + \int_\Gamma \widehat{\boldsymbol{\sigma}} [\![\boldsymbol{v}_h]\!] \, ds = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}_h, \end{aligned} \tag{21}$$

$$\int_\Omega \boldsymbol{\sigma}_h : \boldsymbol{\tau}_h \, dx = -\int_\Omega \boldsymbol{u}_h \sum_{j=1}^d \frac{\partial}{\partial x_i} \sum_{i=1}^d G_{ij}^T(\boldsymbol{u}_h)(\boldsymbol{\tau}_h)_i \, dx + \int_\Gamma \widehat{\boldsymbol{u}} \sum_{j=1}^d n_j \sum_{i=1}^d G_{ij}^T(\boldsymbol{u}_h)(\boldsymbol{\tau}_h)_i \, ds, \tag{22}$$

where the numerical flux functions remain to be defined. $\widehat{F}_i$ can be chosen as any standard numerical flux for hyperbolic problems, and we define $\widehat{\boldsymbol{u}}$ and $\widehat{\boldsymbol{\sigma}}$ as

$$\widehat{\boldsymbol{u}} = \{\boldsymbol{u}_h\}, \qquad \widehat{\boldsymbol{\sigma}} = \{F_v(\boldsymbol{u}_h, \nabla \boldsymbol{u}_h)\} + \eta [\![\boldsymbol{u}_h]\!], \tag{23}$$

where η is a penalty parameter as in the Poisson case. Boundary conditions are enforced by appropriate modification of these numerical flux functions. By setting $\boldsymbol{\tau}_h = \nabla \boldsymbol{v}_h$ in equation (22) and inserting the resulting expression into (21), and subsequently integrating by parts again, we can eliminate $\boldsymbol{\sigma}_h$ to obtain the primal formulation

$$\begin{aligned} &\int_\Omega \partial_t \boldsymbol{u}_h \cdot \boldsymbol{v}_h \, dx - \int_\Omega F_i(\boldsymbol{u}_h) : \nabla \boldsymbol{v}_h \, dx + \int_\Gamma \widehat{F}_i(\boldsymbol{u}_h^+, \boldsymbol{u}_h^-) : [\![\boldsymbol{v}_h]\!] \, ds - \int_\Omega \sum_{j=1}^d G_{ij}(\boldsymbol{u}_h) \frac{\partial \boldsymbol{u}_h}{\partial x_j} : \nabla \boldsymbol{v}_h \, dx \\ &+ \int_\Gamma \left\{ \sum_{j=1}^d G_{ij}(\boldsymbol{u}_h) \frac{\partial \boldsymbol{u}_h}{\partial x_j} \right\} : [\![\boldsymbol{v}_h]\!] \, ds + \int_\Gamma \eta [\![\boldsymbol{u}_h]\!] : [\![\boldsymbol{v}_h]\!] \, ds + \int_\Gamma [\![\boldsymbol{u}_h]\!] : \left\{ \sum_{i=1}^d G_{ij}^T(\boldsymbol{u}_h) \frac{\partial \boldsymbol{v}_h}{\partial x_i} \right\} ds = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}_h. \end{aligned} \tag{24}$$

This general formulation of the interior penalty method is amenable to sum factorization, and can be incorporated into the Kronecker-product preconditioner framework in a straightforward manner.

II.B. Time integration

We use a standard method of lines approach to integrate the semidiscrete system (24) in time. We define the vector $\boldsymbol{u}$ whose entries consist of the degrees of freedom of the solution $\boldsymbol{u}_h$. Then, we write this equation in the form

$$M \boldsymbol{u}_t = \boldsymbol{r}(\boldsymbol{u}), \tag{25}$$

where M is the mass matrix, and the weighted residual $\boldsymbol{r}(\boldsymbol{u})$ is given by evaluating all but the first term in (24), with $\boldsymbol{v}_h$ chosen to be each of the basis functions of the space V_h.

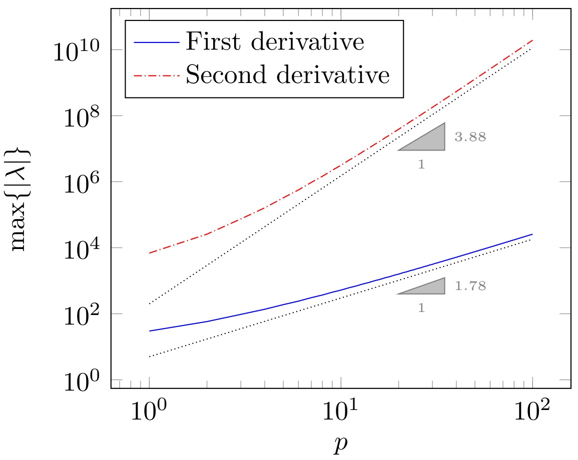


Figure 1: Spectral radius of DG spatial derivatives

The high-order DG discretization can result in a very restrictive CFL stability condition. It has been shown that the spectral radius of the DG spatial derivative operator is bounded above by p^2/h, and that it is in certain cases well-approximated by $p^{1.78}/h$, resulting in a highly restrictive stability condition when the polynomial degree p is taken to be large.[13,17,30] For second-order equations, this restriction is even more severe. For example, the spectral radius of the interior penalty Laplacian operator scales as approximately p^4/h^2, analogous to the case of spectral methods.[16] This fast growth in spectral radius can render explicit time integration methods impractical. The p-dependence of the spectral radius of one-dimensional DG spatial derivative operators is illustrated in Figure 1.

In order to avoid this restrictive time step condition, we make use of implicit time integration schemes. In this work, we consider the use of diagonally-implicit Runge-Kutta (DIRK) schemes.[1] At each stage we make use of Newton's method to solve the resulting nonlinear system of equations, which in turn gives rise to linear systems of the form

$$(M - \alpha \Delta t J)\boldsymbol{x} = \boldsymbol{b}, \tag{26}$$

where J is the Jacobian matrix $\partial \boldsymbol{r}(\boldsymbol{u})/\partial \boldsymbol{u}$. We solve this linear system by means of a preconditioned iterative method such as GMRES. The development of effective preconditioners for such discretizations arising from second-order equations is described in detail in the following sections.

II.C. Tensor-product structure and sum-factorization

An important advantage presented by using the tensor-product function space V_h defined by (3) is the possibility to greatly increase the efficiency of the method using the sum-factorization approach.[19] This approach has been previously applied to the discontinuous Galerkin method,[11,29] and the implementation details are described in detail in Reference 20. This approach allows for the efficient computation of the integrals present in equation (4), reducing the computational complexity from $\mathcal{O}(p^{2d})$ to $\mathcal{O}(p^{d+1})$.

In this section, we introduce some notation which will be useful in describing the tensor-product structure of the DG method. We begin by defining, for each element $K \in \mathcal{T}$, an isoparametric mapping function $T : \mathcal{R} \to K$, where the unit cube $\mathcal{R} = [0,1]^d$ is the reference element. We define $\phi_i(x), 1 \leq i \leq p$ to be a basis for $\mathcal{P}^p([0,1])$. Then, the tensor-product functions $\Phi_I = \bigotimes_{i=1}^d \phi_{I_i}$, for multi-index $I = (I_i), 1 \leq I_i \leq p$, form a basis for the space $\mathcal{P}^p_d(\mathcal{R})$. For a given element K, the basis functions for $\mathcal{P}^p_d(K)$ are then given by $\tilde{\Phi}_I = \Phi_I \circ T^{-1}$. It will be convenient to perform the integration of the quantities in (4) on the reference element, using the Jacobian of the transformation function J_T and its determinant. We consider a quadrature rule on the unit interval $[0,1]$ given by abscissa x_α and weights w_α, $1 \leq \alpha \leq \mu$. We use a tensor-product quadrature rule on $\mathcal{R}$, with abscissa $\boldsymbol{x}_A = (x_{\alpha_i})$ and weights $w_A = \prod_{i=1}^d w_{A_i}$, for multi-index $A = (\alpha_i)$, $1 \leq \alpha_i \leq \mu$.

Many of the important DG operations will be built from Kronecker products of one-dimensional operations. To this end, we define the one-dimensional Gauss point evaluation matrix of size $\mu \times (p+1)$ by

$$G_{\alpha,i} = \phi_i(x_\alpha), \tag{27}$$

and the one-dimensional differentiation matrix by

$$D_{\alpha,i} = \phi_i'(x_\alpha). \tag{28}$$

We also define the $\mu \times \mu$ diagonal weight matrix W by $W_{\alpha,\alpha} = w_\alpha$. These definitions allow us to write, for example, the two-dimensional operators as

$$G^{2\mathrm{D}}_{\alpha\beta,ij} = \phi_i(x_\alpha)\phi_j(x_\beta) = G \otimes G, \qquad D^{2\mathrm{D}}_x = G \otimes D, \qquad D^{2\mathrm{D}}_y = D \otimes G. \tag{29}$$

This process can, of course, be generalized to arbitrary spatial dimension d.

III. Tensor-product preconditioners

An important factor in achieving timely convergence of iterative linear solvers is the application of an effective preconditioner. The preconditioning of DG methods has been much studied, and block-based preconditioner such as block Jacobi, block Gauss-Seidel, and block ILU factorizations have found to be effective.[25] A challenge often encountered when using matrix-free methods such as the method described above is the construction of a preconditioner without having access to the entries of the matrix. In our previous works,[20, 21] we developed a strategy for approximating the diagonal blocks of the DG Jacobian matrix without needing to explicitly form the matrix.

In two spatial dimensions, we construct a preconditioner that approximates a diagonal block A by the sum of Kronecker products

$$A \approx P := A_1 \otimes B_1 + A_2 \otimes B_2. \tag{30}$$

In three spatial dimensions, the diagonal block is approximated by

$$A \approx P := A_1 \otimes B_1 \otimes C_1 + A_1 \otimes B_2 \otimes C_2, \tag{31}$$

where we emphasize that the same factor A_1 appears in both terms on the right-hand side. Finding the optimal such approximation is known as the "nearest Kronecker problem," and its solution can be found using a Kronecker-product singular value decomposition (KSVD). This KSVD can be computed efficiently using a Lanczos algorithm using matrix-free shuffled products. For certain classes of problems (for example, advection by a constant velocity field on a straight-sided mesh), the preconditioner matrix P is exactly equal to the diagonal block A. The solution of systems of equations of the form

$$P\boldsymbol{x} = \boldsymbol{b} \tag{32}$$

can be found efficiently using a simultaneous triangularization method. This preconditioner has been demonstrated to be effective when applied to a variety of problems, including the scalar advection equation and the compressible Euler equations.

III.A. Approximation of second-order terms

The preconditioner described above can be easily applied to DG discretizations of arbitrary hyperbolic conservation laws. However, the extension to equations with second-order terms such as the Navier-Stokes equations is not straightforward. Referring to the notation from Section II.A, we can choose to either make use of the system flux formulation or the primal formulation of the discretization. A significant drawback of implementing the system flux formulation directly is that it leads to a global system of equations with many more degrees of freedom than necessary. In fact, one of the advantageous aspects of methods such as the local DG method and the interior penalty method is that the gradient $\boldsymbol{\sigma}$ can be solved for element-by-element. Furthermore, these large linear systems often have a saddle-point structure, resulting in poor performance of iterative linear solvers.[6] For example, the local DG discretization of the Poisson problem (5) gives rise to a linear system of the form

$$\begin{pmatrix} M & -D \\ -D^T & E \end{pmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ \boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_{\boldsymbol{\sigma}} \\ \boldsymbol{b}_u \end{pmatrix}, \tag{33}$$

where D is a matrix corresponding to the divergence of $\boldsymbol{u}$, and D^T is the discrete gradient operator. The matrix E contains the stabilization terms corresponding to LDG coefficient $C_{11} > 0$.

The primal formulation given by (15) avoids this issue by eliminating the gradient $\boldsymbol{\sigma}$ and writing the bilinear form only in terms of the unknown function u. However, a difficulty arises in the sum factorization of the primal form. For most of the methods enumerated in Reference 3, the primal form requires the computation of the lifting operators of the form $r(u)$ defined by (12). These lifting operators are computed by inverting the mass matrices local to each element. However, on curved elements $K = T(\mathcal{R})$, where T is an isoparametric mapping, the inverse of the mass matrix cannot be expressed in tensor-product form. This restriction makes the sum factorization of the lifting operators significantly more challenging, and prevents the constructor of efficient shuffled matrix-vector products needed to compute the KSVD. To remedy these issues, we make use of the interior penalty method, whose primal form is given by (17), which does not require the computation of any lifting operators.

III.A.1. Exact representations

In the case of the scalar advection equation, it has been shown[20] that the representation (30) is exact for certain classes of problems. We would like repeat a similar analysis for the cases of the Poisson problem

$$-\Delta u = f \tag{34}$$

and scalar convection diffusion equation

$$u_t + \nabla \cdot (\boldsymbol{\beta} u - \nabla u) = 0 \tag{35}$$

in two spatial dimensions.

First we consider the Poisson problem. We consider the diagonal block A of the matrix associated with the bilinear form $B(\cdot,\cdot)$ given by (17), corresponding to an element K. We restrict ourselves to the case of a Cartesian grid, and so K is given by a translation of the unit square. Thus, the transformation Jacobian is equal to the identity matrix, and it suffices to consider the simple case of $K = \mathcal{R}$. The entries of A are then given by $B(\Phi_{k\ell}, \Phi_{ij})$. We can see that the terms corresponding to the volume integral can be written in the form

$$\left(\int_\Omega \nabla_h \Phi_{k\ell} \cdot \nabla_h \Phi_{ij} \; dx \right) = G^T W G \otimes D^T W D + D^T W D \otimes G^T W G, \tag{36}$$

using the notation from Section II.C. In a similar fashion, the penalty boundary terms can be written as

$$\left(\int_\Gamma \eta_e [\![u_h]\!] \cdot [\![v_h]\!] \right) = \eta_e \left(G^T W G \otimes G_0^T G_0 + G^T W G \otimes G_1^T G_1 + G_0^T G_0 \otimes G^T W G + G_1^T G_1 \otimes G^T W G \right), \tag{37}$$

where $(G_0)_i = \phi_i(0)$ and $(G_1)_i = \phi_i(1)$ are $1 \times (p+1)$ end-point evaluation matrices. The end-point differentiation matrices D_0 and D_1 are defined similarly. The remaining boundary terms can be treated similarly, resulting in the following form for the diagonal blocks,

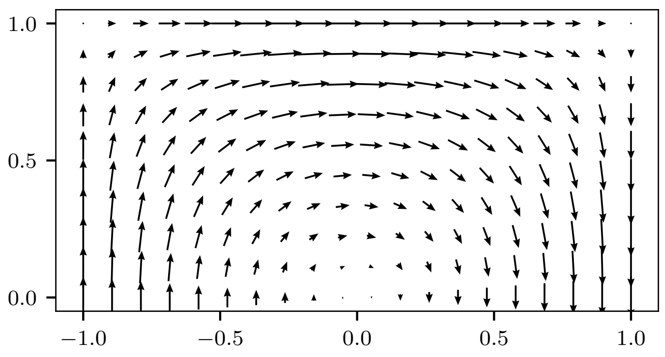
$$\begin{aligned} G^T W G \otimes \left(-D^T W D - \eta_e G_0^T G_0 - \eta_e G_1^T G_1 - D_0^T G_0 - G_0^T D_0 + D_1^T G_1 + G_1^T D_1 \right) \\ + \left(-D^T W D - \eta_e G_0^T G_0 - \eta_e G_1^T G_1 - D_0^T G_0 - G_0^T D_0 + D_1^T G_1 + G_1^T D_1 \right) \otimes G^T W G, \end{aligned} \tag{38}$$

which, in particular, demonstrates that the diagonal blocks can be written as the sum of two Kronecker products. Therefore, the tensor-product preconditioner is able to exactly reproduce the diagonal blocks of the DG discretization in the case of the Poisson problem on a Cartesian grid.

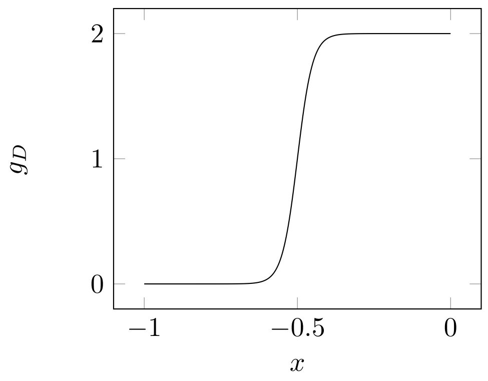
It can also be shown that the fully discrete system (26) for the scalar advection equation on a Cartesian grid with constant velocity field $\boldsymbol{\beta} = (\beta_x, \beta_y)$ gives rise to diagonal blocks of the form

$$\begin{aligned} G^T W G \otimes G^T W G - \alpha \Delta t (\beta_x G^T W G \otimes D^T W G + \beta_y D^T W G \otimes G^T W G \\ - \beta_x G^T W G \otimes G_1^T G_1 - \beta_y G_1^T G_1 \otimes G^T W G), \end{aligned} \tag{39}$$

where we assume for simplicity that $\beta_x, \beta_y > 0$, though this assumption is not necessary. Combining (38) and (39), we see that the diagonal blocks corresponding to the DG discretization of the convection-diffusion



(a) Prescribed velocity field $\boldsymbol{\beta}(x,y)$



(b) Inflow condition

Figure 2: Convection-diffusion test problem

equation (35) on a Cartesian grid with constant velocity field can be written as

$$\begin{aligned} G^TWG \otimes \Big(& G^TWG - \alpha\beta_x \Delta t (D^TWG - G_1^TG_1) - D^TWD - \eta_e G_0^T G_0 - \eta_e G_1^T G_1 \\ & - D_0^T G_0 - G_0^T D_0 + D_1^T G_1 + G_1^T D_1 \Big) + \Big(- \alpha\beta_y \Delta t (D^TWG - G_1^TG_1) - D^TWD \\ & - \eta_e G_0^T G_0 - \eta_e G_1^T G_1 - D_0^T G_0 - G_0^T D_0 + D_1^T G_1 + G_1^T D_1 \Big) \otimes G^TWG, \end{aligned} \tag{40}$$

which shows that the approximate tensor-product preconditioner (30) exactly reproduces the block Jacobi preconditioner in this case. In more general cases, we cannot expect the preconditioner to exactly reproduces the diagonal blocks. However, the KSVD construction guarantees the optimal (in Frobenius norm) such approximation, and the effectiveness of the preconditioner is demonstrated on a variety of test cases in the following sections.

IV. Numerical results

IV.A. Convection-diffusion

We consider the time-dependent scalar convection-diffusion equation,

$$\begin{aligned} u_t + \nabla \cdot (\boldsymbol{\beta} u - \epsilon \nabla u) = 0 \qquad & \text{in } \Omega, \\ u = g_D \qquad & \text{on } \partial\Omega_D, \\ \partial u / \partial n = g_N \qquad & \text{on } \partial\Omega_N, \end{aligned} \tag{41}$$

where $\boldsymbol{\beta}(x,y)$ is a given velocity field, and $\epsilon > 0$ is a constant diffusion coefficient. Also of interest to us is the steady version of this problem, where the first equation in (41) is replaced by

$$\nabla \cdot (\boldsymbol{\beta} u - \epsilon \nabla u) = 0. \tag{42}$$

In this section, we consider two benchmark convection-diffusion test cases studied in detail by Mackenzie and Morton.[18] The domain for both test cases is the rectangle $\Omega = [-1,1] \times [0,1]$. A prescribed velocity field (shown in Figure 2a) given by $\boldsymbol{\beta}(x,y) = (2y(1-x^2), -2x(1-y^2))$ is used.

IV.A.1. Test case 1

In the first test case,[28] we partition the boundary of the domain $\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. On $\Gamma_1 = \{-1 \leq x \leq 0, y = 0\}$, we specify a Dirichlet condition with a steep gradient given by $g_D(x) = 1 + \tanh(20x + 10)$. On $\Gamma_2 = \{0 \leq x \leq 1, y = 0\}$, we specific a homogeneous Neumann (outflow) condition. On the remaining tangential boundaries, Γ_3, we use a compatible Dirichlet condition $g_D = 1 - \tanh(10)$.

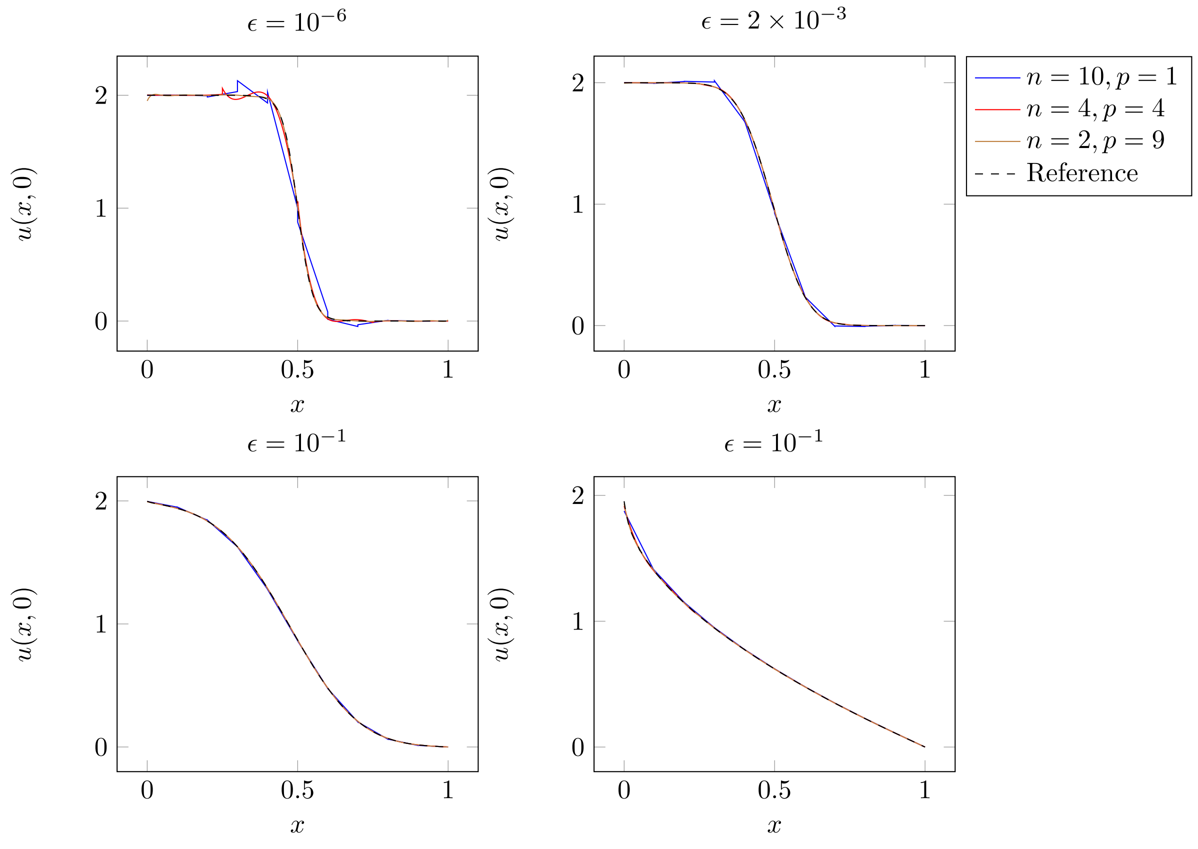


Figure 3: Computed outflow profiles for test problem 1.

We consider a wide range of diffusion coefficients $\epsilon = 10^{-6}, 2 \times 10^{-3}, 10^{-2}$, and 10^{-1}. The Péclet number ranges from 20 to 2×10^6. An important quantity of interest to study for this problem is the steady-state outflow profile $u(x, 0)$ for $0 \leq x \leq 1$. In order to motivate the use of high-order methods, we compare the computed outflow profiles for a variety of grids and polynomial degrees. The mesh is taken to be a regular grid of size $2n \times n$. The number of degrees of freedom is fixed at 800, and we consider three configurations: $(n = 10, p = 1)$, $(n = 4, p = 4)$, and $(n = 2, p = 9)$. The outflow profiles for all choices of diffusion coefficient ϵ are shown in Figure 3. We note that for the convection-dominated case of $\epsilon = 10^{-6}$, the lower-order methods severely underperform the higher-order methods. For $\epsilon = 10^{-2}$ this effect is much less dramatic, however for the most diffusive case of $\epsilon = 10^{-1}$, the second-order method does not accurately capture the steep gradient observed near the origin. This suggests that there is a benefit to using moderate to high polynomial degrees rather than low degree polynomials on h-refined meshes.

Now we turn our attention to solver and preconditioner performance. To study the effectiveness of the preconditioner, we compute the number of GMRES iterations required per linear solve. As before, we consider four choices of diffusion coefficient ϵ. We also consider the choice of time step Δt for the unsteady version of this problem. As a baseline for our comparisons, we use the exact block Jacobi preconditioner. We then compare both the approximate Kronecker-product preconditioner which incorporates second-order terms through the sum-factorized interior penalty method (which we denote KSVD-IP) and the Kronecker-product preconditioner used in previous works[21] that did not incorporate the diffusion terms (which we denote KSVD). The iteration counts are presented in Table 1. We observe that the block Jacobi and Kronecker-product preconditioners exhibit extremely similar convergence properties. However, for large diffusion coefficient, and in particular for high degree p, we see that Kronecker-product preconditioner that does not include the diffusion term does not result in fast convergence. In particular, for $\epsilon = 10^{-1}$ and $p = 4, p = 9$, GMRES did not converge to the steady-state solution in fewer than 2000 iterations. These results indicate a significant advantage to incorporating the diffusion terms using the interior penalty method.

Table 1: Number of GMRES iterations for Jacobi/KSVD-IP/KSVD preconditioner for convection-diffusion test case 1. A dash indicates no convergence in less than 2000 iterations.

$\epsilon = 10^{-6}$

Δt	$p = 1$	$p = 4$	$p = 9$
1×10^{-2}	7/7/7	6/6/6	6/8/8
2×10^{-2}	7/7/7	6/8/8	6/9/9
4×10^{-2}	8/8/8	6/8/8	5/10/10
8×10^{-2}	10/10/10	6/8/8	5/12/12
1.6×10^{-1}	15/15/15	7/9/9	6/14/14
Steady	25/25/25	12/12/12	9/9/9

$\epsilon = 2 \times 10^{-3}$

Δt	$p = 1$	$p = 4$	$p = 9$
1×10^{-2}	7/7/7	9/10/10	12/14/14
2×10^{-2}	9/9/9	11/12/12	15/17/19
4×10^{-2}	10/10/10	15/15/15	19/23/25
8×10^{-2}	14/14/14	19/20/21	23/30/33
1.6×10^{-1}	21/21/21	25/26/27	28/35/41
Steady	59/59/59	53/55/64	49/57/82

$\epsilon = 10^{-2}$

Δt	$p = 1$	$p = 4$	$p = 9$
1×10^{-2}	10/9/10	14/14/15	18/23/24
2×10^{-2}	12/12/12	18/18/18	22/26/29
4×10^{-2}	16/16/16	22/22/24	26/30/39
8×10^{-2}	22/22/22	28/28/34	31/36/58
1.6×10^{-1}	33/33/33	36/36/52	39/46/94
Steady	96/98/97	64/69/171	67/84/382

$\epsilon = 10^{-1}$

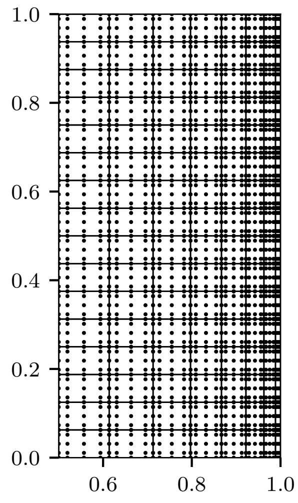
Δt	$p = 1$	$p = 4$	$p = 9$
1×10^{-2}	19/19/18	24/24/34	29/30/61
2×10^{-2}	24/24/24	29/29/48	34/34/86
4×10^{-2}	33/33/33	36/36/77	42/43/148
8×10^{-2}	48/48/48	46/46/138	51/52/296
1.6×10^{-1}	67/67/67	59/59/288	61/63/656
Steady	156/160/158	121/122/-	98/103/-

IV.A.2. Test case 2

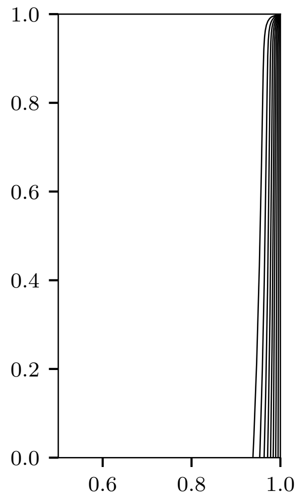
The second test case is a slight modification of the above problem. We decompose the boundary $\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. On the right boundary $\Gamma_1 = \{x = 1, 0 \leq y \leq 1\}$ we enforce a Dirichlet condition of $g_D = 100$. On the outflow boundary, $\Gamma_2 = \{0 \leq x \leq 1, y = 0\}$, we enforce a homogeneous Neumann condition. On the remaining boundaries, Γ_3, we enforce a homogeneous Dirichlet condition. The most important feature of this test case is the boundary layer that forms on the right boundary. In order to properly resolve this feature, we use an anisotropic mesh that is refined in the vicinity of the right boundary, see Figure 4b. We consider three mesh configurations, each with 3200 degrees of freedom, using degree 1, 4, and 9 polynomials. The anisotropy of the mesh, combined with the CFL condition resulting from the diffusion term, results in a severe time step restriction for explicit methods. In Table 2 we show the maximum stable time step for each configuration using the standard fourth-order explicit Runge-Kutta method. This motivates the use of implicit time integration methods.

We measure the number of GMRES iterations required per linear solve for both the time-dependent case (as a function of Δt), and for the steady case. The results are shown in Table 3. The number of iterations required with interior penalty Kronecker-product preconditioner is almost identical to the number of iterations required with the exact block Jacobi preconditioner for all cases considered. When comparing against the previous Kronecker-product preconditioner that did not include the diffusion term, the results are quite similar for either small diffusion coefficient ($\epsilon = 10^{-6}$), or for low polynomial degree ($p = 1$). However, for the more diffusive cases, and for higher degree polynomials, we see a dramatic difference in the number of iterations required. For $\epsilon = 10^{-2}$ or $\epsilon = 10^{-1}$ and $p = 4$ or $p = 9$, the solver did not converge in under 2000 iterations for most of the test cases.

These results demonstrate a marked improvement in preconditioner performance by including the diffusive terms in the Kronecker-product approximation.



(a) Anisotropic mesh with $p = 4$ nodes.



(b) Solution contours showing boundary layer, $\epsilon = 2 \times 10^{-3}$.

Figure 4: Convection-diffusion test problem 2

Table 2: Largest allowable time step for RK4

ϵ	$p = 1$	$p = 4$	$p = 9$
1×10^{-6}	1×10^{-2}	1×10^{-2}	4×10^{-3}
2×10^{-3}	2×10^{-4}	8×10^{-5}	2×10^{-5}
1×10^{-2}	6×10^{-5}	1×10^{-5}	4×10^{-6}
1×10^{-1}	6×10^{-6}	1×10^{-6}	4×10^{-7}

Table 3: Number of GMRES iterations for Jacobi/KSVD-IP/KSVD preconditioner for convection-diffusion test case 2. A dash indicates no convergence in less than 2000 iterations.

$\epsilon = 10^{-6}$

Δt	$p = 1$	$p = 4$	$p = 9$
10^{-2}	6/6/6	6/6/6	8/8/7
2×10^{-2}	9/9/9	8/8/8	10/10/9
4×10^{-2}	15/15/15	11/11/10	12/12/12
8×10^{-2}	23/23/23	14/14/14	14/14/14
1.6×10^{-1}	31/31/31	18/18/18	15/15/16
Steady	45/44/45	24/24/24	18/19/19

$\epsilon = 2 \times 10^{-3}$

Δt	$p = 1$	$p = 4$	$p = 9$
10^{-2}	28/28/28	45/45/106	52/53/280
2×10^{-2}	36/36/36	61/61/163	74/74/673
4×10^{-2}	49/49/49	84/84/281	102/103/-
8×10^{-2}	70/70/71	110/110/503	136/137/-
1.6×10^{-1}	103/103/103	136/136/755	178/180/-
Steady	178/178/179	202/202/-	279/283/-

$\epsilon = 10^{-2}$

Δt	$p = 1$	$p = 4$	$p = 9$
10^{-2}	42/42/42	61/61/258	78/78/-
2×10^{-2}	54/54/54	75/76/533	100/100/-
4×10^{-2}	72/72/73	91/91/-	151/160/-
8×10^{-2}	105/105/105	111/111/-	178/179/-
1.6×10^{-1}	141/141/142	134/134/-	204/206/-
Steady	230/230/231	197/197/-	312/305/-

$\epsilon = 10^{-1}$

Δt	$p = 1$	$p = 4$	$p = 9$
10^{-2}	83/83/85	89/89/-	120/120/-
2×10^{-2}	106/106/108	104/104/-	137/138/-
4×10^{-2}	137/137/139	126/126/-	167/168/-
8×10^{-2}	160/160/163	151/151/-	178/180/-
1.6×10^{-1}	193/193/196	176/176/-	194/194/-
Steady	310/310/313	259/258/-	290/299/-

IV.B. Navier-Stokes equations

In this section, we consider the compressible Navier-Stokes equations in two dimensions,

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0 \tag{43}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) + \frac{\partial p}{\partial x_i 1} = \frac{\partial \tau_{ij}}{\partial x_j} \qquad \text{for } i = 1, 2, \tag{44}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}\left(u_j(\rho E + p)\right) = -\frac{\partial q_j}{\partial x_j} + \frac{\partial}{\partial x_j}(u_j \tau_{ij}), \tag{45}$$

where ρ is the density, u_i is the ith component of the velocity, and E is the total energy. The viscous stress tensor and heat flux are given by

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) \qquad \text{and} \qquad q_j = -\frac{\mu}{\mathrm{Pr}}\frac{\partial}{\partial x_j}\left(E + \frac{p}{\rho} - \frac{1}{2}u_k u_k\right). \tag{46}$$

Here μ is the coefficient of viscosity, and Pr is the Prandtl number. The equation of state of an ideal gas is given by

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}u_k u_k\right), \tag{47}$$

where γ is the adiabatic gas constant.

We rewrite equations (43–45) in conservative form by defining the viscous and inviscid fluxes by

$$F_i^1(\boldsymbol{u}) = \begin{pmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ \rho(E + p/\rho)u_1 \end{pmatrix}, \qquad F_i^2(\boldsymbol{u}) = \begin{pmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ \rho(E + p/rho)u_2 \end{pmatrix}, \tag{48}$$

$$F_v^1(\boldsymbol{u}) = \begin{pmatrix} 0 \\ \tau_{11} \\ \tau_{21} \\ q_1 - u_j\tau_{1j} \end{pmatrix}, \qquad F_v^2(\boldsymbol{u}) = \begin{pmatrix} 0 \\ \tau_{12} \\ \tau_{22} \\ q_2 - u_j\tau_{j2} \end{pmatrix}. \tag{49}$$

In order to formulate the interior penalty method for the Navier-Stokes equations, we must write the viscous flux F_v in the form (18). Following the derivation of Hartmann,[14] we define the matrices

$$G_{11} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -\frac{4}{3}u_1 & \frac{4}{3} & 0 & 0 \\ -u_2 & 0 & 1 & 0 \\ -\left(\frac{4}{3}u_1^2 + u_2^2 + \frac{\gamma}{\mathrm{Pr}}(E - \boldsymbol{u}^2)\right) & \left(\frac{4}{3} - \frac{\gamma}{\mathrm{Pr}}\right)u_1 & \left(1 - \frac{\gamma}{\mathrm{Pr}}\right)u_2 & \frac{\gamma}{\mathrm{Pr}} \end{pmatrix},$$

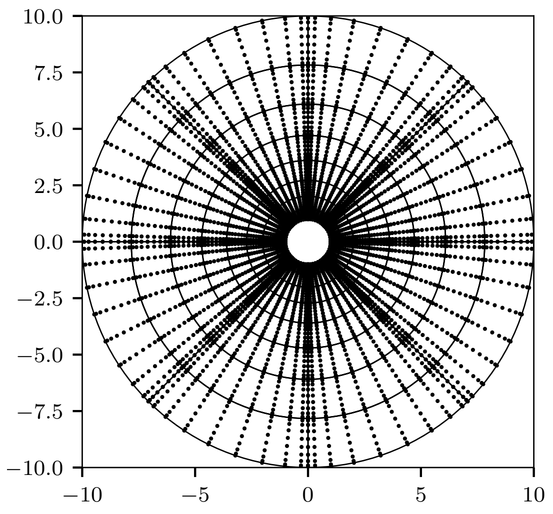
$$G_{12} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{2}{3}u_2 & 0 & -\frac{2}{3} & 0 \\ -u_1 & 1 & 0 & 0 \\ -\frac{1}{3}u_1u_2 & u_2 & -\frac{2}{3}u_1 & 0 \end{pmatrix}, \qquad G_{21} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -u_2 & 0 & 1 & 0 \\ \frac{2}{3}u_1 & -\frac{2}{3} & 0 & 0 \\ -\frac{1}{3}u_1u_2 & -\frac{2}{3}u_2 & u_1 & 0 \end{pmatrix},$$

$$G_{22} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -u_1 & 1 & 0 & 0 \\ -\frac{4}{3}u_2 & 0 & \frac{4}{3} & 0 \\ -\left(u_1^2 + \frac{4}{3}u_2^2 + \frac{\gamma}{\mathrm{Pr}}(E - \boldsymbol{u}^2)\right) & \left(1 - \frac{\gamma}{\mathrm{Pr}}\right)u_1 & \left(\frac{4}{3} - \frac{\gamma}{\mathrm{Pr}}\right)u_2 & \frac{\gamma}{\mathrm{Pr}} \end{pmatrix}$$

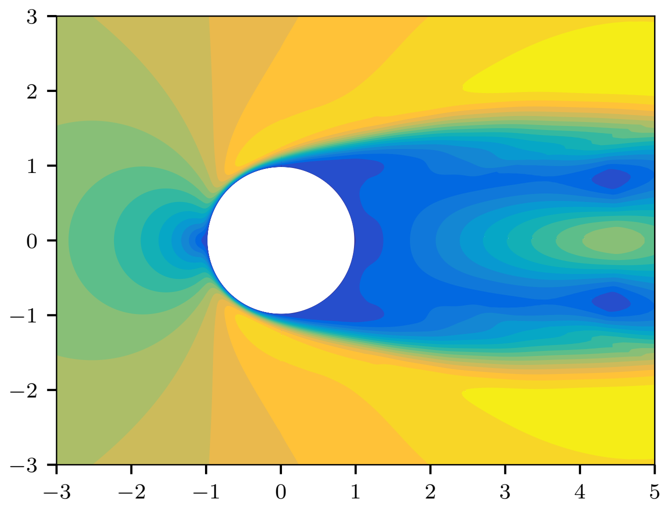
such that

$$F_v^i(\boldsymbol{u}, \nabla\boldsymbol{u}) = \sum_{j=1}^{2} G_{ij}(\boldsymbol{u})\frac{\partial \boldsymbol{u}}{\partial x_j}, \tag{50}$$

allowing us to use the interior penalty formulation given by (24).



(a) Mesh of annulus $A(1, 10)$ with $p = 9$ curved isoparametric elements.



(b) Solution (velocity magnitude) at $t = 50$ for flow over a cylinder at Re = 200.

Figure 5: Mesh and computed solution for viscous compressible flow over a circular cylinder.

IV.B.1. Flow over a circular cylinder

We consider viscous compressible flow over a circular cylinder. The domain is given by, $\Omega = A(1, 10)$, the annulus with inner radius 1 and outer radius 10. As in the previous examples, we consider three mesh configurations, corresponding to polynomial degrees $p = 1, p = 4, p = 9$. Each mesh has 6400 degrees of freedom per solution component. The coarsest mesh is shown in Figure 5a. A no-slip boundary condition is enforced at the inner boundary, and far-field conditions are enforced at the outer boundary. The Mach number is chosen to be $M = 0.2$, and we consider a range of Reynolds numbers: $\mathrm{Re} = 10, \mathrm{Re} = 200$, and $\mathrm{Re} = 1000$. We start from freestream conditions and integrate in time until $t = 1$ in order to obtain a representative solution.

At this point, we measure the number of GMRES iterations required per linear solve. We consider the time steps $\Delta t = 10^{-2}, 2 \times 10^{-2}, 4 \times 10^{-2}$, and 8×10^{-2}. As before, we compare the effectiveness of three preconditioners: exact block Jacobi, the interior penalty approximate Kronecker-product preconditioner (KSVD-IP), and the Kronecker-product preconditioner that does not include viscous terms (KSVD). Each of these preconditioners is applied component-wise to the Jacobian matrix (i.e. with block size $(p+1)^d \times (p+1)^d$, with n_c diagonal blocks per element). A comprehensive comparison of iteration counts in shown in Table 5.

We observe that the interior penalty Kronecker-product preconditioner is able to match the performance of the exact block Jacobi preconditioner for all polynomial degrees considered, and at all choices of Reynolds number. In contrast, the Kronecker-product preconditioner that did not include viscous terms results in highly decreased performance at both low Reynolds numbers and high polynomial degrees. For the low-degree case of $p = 1$, the effect was extremely modest for all Reynolds numbers. This suggests that the proper incorporation of second-order terms is important for good preconditioner performance at high degrees. For the convection-dominated case of $\mathrm{Re} = 1000$, the effect was modest except for at the largest time step $\Delta t = 8 \times 10^{-2}$. However, for the viscous-dominated case of $\mathrm{Re} = 10$, the increase in iterations was sizable for all time steps for degree $p \geq 4$.

V. Conclusion

In this work, we introduced an improved Kronecker-product preconditioner that uses the particular form of the interior penalty method to properly incorporate second-order derivative terms that arise from diffusive or viscous terms. This avoids the difficulty of computing the lifting operators required for other discretizations of such second-order terms. This preconditioner exactly reproduces the diagonal blocks of the discretized matrix in certain special cases, through an entirely algebraic and automatic approach. In cases where it is not exact, it is chosen to be optimal in the Frobenius norm. Numerical examples demonstrate

Table 5: Number of GMRES iterations for Jacobi/KSVD-IP/KSVD preconditioner for flow over a circular cylinder. A dash indicates no convergence in less than 2000 iterations.

Re = 10

Δt	$p = 1$	$p = 4$	$p = 9$
1×10^{-2}	16/16/14	28/28/47	46/46/125
2×10^{-2}	20/20/22	45/44/83	81/81/229
4×10^{-2}	35/35/40	73/74/160	146/145/652
8×10^{-2}	61/61/72	118/118/388	247/250/-

Re = 200

Δt	$p = 1$	$p = 4$	$p = 9$
1×10^{-2}	17/17/13	21/21/18	27/30/35
2×10^{-2}	21/21/21	32/33/37	56/60/78
4×10^{-2}	35/35/36	70/71/95	128/134/257
8×10^{-2}	65/65/67	176/179/383	371/379/1610

Re = 1000

Δt	$p = 1$	$p = 4$	$p = 9$
1×10^{-2}	17/17/14	22/22/19	29/31/31
2×10^{-2}	22/22/22	37/37/39	61/66/73
4×10^{-2}	36/36/36	90/91/100	213/218/384
8×10^{-2}	66/66/66	293/297/509	1344/1554/-

the effectiveness of this preconditioner when compared with block Jacobi on a range of problems, including convection diffusion and compressible Navier-Stokes. Comparisons with the previous Kronecker-product preconditioner, which did not incorporate diffusive terms, demonstrates a marked performance increase on a range of problems. Future work involves the development of Kronecker-product preconditioners that are suitable for use within a p-multigrid framework for the solution of elliptic steady-state problems.

VI. Acknowledgements

This work was supported by the AFOSR Computational Mathematics program under grant number FA9550-15-1-0010. The first author was supported by the Department of Defense through the National Defense Science & Engineering Graduate Fellowship Program and by the Natural Sciences and Engineering Research Council of Canada.

References

[1]R. Alexander. Diagonally implicit Runge-Kutta methods for stiff O.D.E.'s. *SIAM Journal on Numerical Analysis*, 14(6):1006–1021, 1977.

[2]D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis*, 19(4):742–760, 1982.

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

[4]F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 131(2):267–279, 1997.

[5]A. Beck, T. Bolemann, T. Hitz, V. Mayer, and C.-D. Munz. Explicit high-order discontinuous Galerkin spectral element

methods for LES and DNS. In *Recent Trends in Computational Engineering-CE2014*, pages 281–296. Springer, 2015.

[6]M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[7]P. Birken, G. Gassner, M. Haas, and C.-D. Munz. Preconditioning for modal discontinuous galerkin methods for unsteady 3D Navier-Stokes equations. *Journal of Computational Physics*, 240:20–35, 2013.

[8]B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, 1998.

[9]B. Cockburn and C.-W. Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998.

[10]L. T. Diosady and D. L. Darmofal. Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations. *Journal of Computational Physics*, 228(11):3917–3935, 2009.

[11]L. T. Diosady and S. M. Murman. Tensor-product preconditioners for higher-order spacetime discontinuous Galerkin methods. *Journal of Computational Physics*, 330:296 – 318, 2017.

[12]M. Drosson and K. Hillewaert. On the stability of the symmetric interior penalty method for the Spalart-Allmaras turbulence model. *Journal of Computational and Applied Mathematics*, 246:122–135, 2013.

[13]D. Gottlieb and E. Tadmor. The CFL condition for spectral approximations to hyperbolic initial-boundary value problems. *Mathematics of Computation*, 56(194):565–588, 1991.

[14]R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier-Stokes equations I: Method formulation. *International Journal of Numerical Analysis & Modeling*, 3(1):1–20, 2006.

[15]R. Hartmann and P. Houston. An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 227(22):9670–9685, 2008.

[16]D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations*. Springer Netherlands, 2009.

[17]L. Krivodonova and R. Qin. An analysis of the spectrum of the discontinuous Galerkin method. *Applied Numerical Mathematics*, 64:1–18, 2013.

[18]J. A. Mackenzie and K. W. Morton. Finite volume solutions of convection-diffusion test problems. *Mathematics of Computation*, 60(201):189, 1993.

[19]S. A. Orszag. Spectral methods for problems in complex geometries. *Journal of Computational Physics*, 37(1):70 – 92, 1980.

[20]W. Pazner and P.-O. Persson. Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods. *Journal of Computational Physics*, 354:344 – 369, 2017.

[21]W. Pazner and P.-O. Persson. High-order DNS and LES simulations using an implicit tensor-product discontinuous Galerkin method. In *Proceedings of the 23rd AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 2017.

[22]J. Peraire and P.-O. Persson. The compact discontinuous Galerkin (CDG) method for elliptic problems. *SIAM Journal on Scientific Computing*, 30(4):1806–1824, 2008.

[23]J. Peraire and P.-O. Persson. High-order discontinuous Galerkin methods for CFD. In Z. J. Wang, editor, *Adaptive High-Order Methods in Fluid Dynamics*, chapter 5, pages 119–152. World Scientific, 2011.

[24]P.-O. Persson. High-order LES simulations using implicit-explicit Runge-Kutta schemes. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting and Exhibit, AIAA*, volume 684, 2011.

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

[26]W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. *Los Alamos Report LA-UR-73-479*, 1973.

[27]K. Shahbazi. An explicit expression for the penalty parameter of the interior penalty method. *Journal of Computational Physics*, 205(2):401–407, 2005.

[28]R. M. Smith and A. G. Hutton. The numerical treatment of advection: A performance comparison of current methods. *Numerical Heat Transfer*, 5(4):439–461, 1982.

[29]P. E. J. Vos, S. J. Sherwin, and R. M. Kirby. From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low-and high-order discretisations. *Journal of Computational Physics*, 229(13):5161–5181, 2010.

[30]T. Warburton and T. Hagstrom. Taming the CFL number for discontinuous Galerkin methods on structured meshes. *SIAM Journal on Numerical Analysis*, 46(6):3151–3180, 2008.