AIAA AVIATION Forum
5-9 June 2017, Denver, Colorado
23rd AIAA Computational Fluid Dynamics Conference

High-order DNS and LES simulations using an implicit tensor-product discontinuous Galerkin method

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.

This paper describes an efficient tensor-product based preconditioner for the large linear systems arising from the implicit time integration of discontinuous Galerkin (DG) discretizations. A main advantage of the DG method is its potential for high-order accuracy, but the number of degrees of freedom per element scales as p^d, where p is the polynomial degree and d is the spatial dimension. Standard preconditioners such as block Jacobi and ILU factorizations rely on dense linear algebra, incurring a computational cost of $\mathcal{O}(p^{3d})$, which quickly becomes intractable. Our new preconditioner exploits the natural tensor-product structure of general quadrilateral and hexahedral meshes to reduce the computational complexity to $\mathcal{O}(p^3)$ in two dimensions, and $\mathcal{O}(p^5)$ in three dimensions. We apply this preconditioner to two benchmark fluid flow problems: the direct numerical solution of the compressible Taylor-Green vortex, and the large eddy simulation of flow over a NACA airfoil. These test cases demonstrate the effectiveness of the new tensor-product preconditioners on large-scale, high-order CFD problems.

I. Introduction

The discontinuous Galerkin (DG) method was introduced in 1973 by Reed and Hill for the neutron transport equation,[24] and in the 1990s was extended to nonlinear systems of conservation laws by Cockburn and Shu.[9] The DG method has successfully been applied a wide range of CFD problems,[20] including direct numerical simulation and large eddy simulation of turbulent flows, using both implicit and explicit time integration methods.[4,21] Although the DG method generalizes to arbitrary polynomial orders, there are several challenges preventing the use of very high degree polynomial bases. For explicit methods, the CFL stability condition requires that the time step satisfy approximately $\Delta t \leq Ch/p^2$, where h is the element size, and p is the degree of polynomial approximation.[14] For implicit methods, the number of degrees of freedom per element scales as p^d in d dimensions. The resulting linear system can be considered as a sparse matrix with dense $(p+1)^d \times (p+1)^d$ blocks for each element, requiring $\mathcal{O}(p^{3d})$ operations per linear solve using dense linear algebra, rendering the problem intractable for very large p. If it is possible to well approximate these $(p+1)^d \times (p+1)^d$ with certain sums of Kronecker products of smaller $(p+1) \times (p+1)$ matrices, it would allow for asymptotically much more computationally efficient linear algebra.

In this paper, we describe an implicit DG method with a specific tensor-product structure whose computational cost per degree of freedom scales linearly with the polynomial degree p in two dimensions, and as $\mathcal{O}(p^5)$ in three dimensions. This method requires a quadrilateral or hexahedral mesh, and a tensor-product basis. The matrix corresponding to the linear system is never explicitly constructed, but rather fast matrix-vector multiplications are performed as the kernel of the iterative GMRES algorithm. Such systems are traditionally preconditioned using *e.g.* block Jacobi, Gauss-Seidel, or ILU preconditioners.[17,22] In order to avoid inverting the diagonal blocks of the matrix and thus incurring the above-mentioned cost of $\mathcal{O}(p^{3d})$

*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.

operations, we make use the Kronecker product singular value decomposition to approximate this block by a sum of lower-dimensional tensor products.[30] In our previous work,[19] we described efficient algorithms for computing certain optimal approximations of these diagonal blocks, and implemented a matrix-free iterative framework for solving the large linear systems. This preconditioner has been applied effectively to systems of hyperbolic conservation laws, including the scalar advection equation and the Euler equations of gas dynamics, in two and three spatial dimensions.

In this work, we extend the application of these preconditioners to the compressible Navier-Stokes equations. In particular, we apply these new techniques to DNS and LES fluid flow problems. First, we consider the direct numerical simulation of the compressible Taylor-Green vortex at Re = 1600. This standard benchmark problem[33] is used to test the performance of high-order methods on transitional flows, and has been previous studied as a verification case for a wide variety of numerical methods, including discontinuous Galerkin methods.[6,7,11,12,27] Secondly, we consider the large eddy simulation of viscous flow over a NACA 0012 airfoil. This test problem is characterized by highly-refined elements around the boundary of the airfoil, resulting in a very restrictive CFL stability condition, and thus motivating the use of implicit time integration methods. The DG method has successfully been applied to similar test cases in several previous works.[3,28,29,32] The numerical results in the present work aim to demonstrate the feasibility of implicit time integration for DNS and LES problems using the discontinuous Galerkin method with very high polynomial degrees p.

II. Governing equations and discretization

The equations considered are the time-dependent, compressible Navier-Stokes equations,

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0 \tag{1}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) + \frac{\partial p}{\partial x_i 1} = \frac{\partial \tau_{ij}}{\partial x_j} \qquad \text{for } i = 1, 2, 3, \tag{2}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}\left(u_j(\rho E + p)\right) = -\frac{\partial q_j}{\partial x_j} + \frac{\partial}{\partial x_j}(u_j \tau_{ij}), \tag{3}$$

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

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) \qquad \text{and} \qquad q_j = -\frac{\mu}{\mathrm{Pr}}\frac{\partial}{\partial x_j}\left(E + \frac{p}{\rho} - \frac{1}{2}u_k u_k\right). \tag{4}$$

Here μ is the coefficient of viscosity, and Pr is the Prandtl number, which we assume to be constant. For an ideal gas, the pressure p has the form

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}u_k u_k\right), \tag{5}$$

where γ is the adiabatic gas constant.

We rewrite equations (1), (2), and (3) in the form

$$\frac{\partial u}{\partial t} + \nabla \cdot \boldsymbol{F}_i(u) - \nabla \cdot \boldsymbol{F}_v(u, \nabla u) = 0, \tag{6}$$

where u is a vector of the conserved variables, and $\boldsymbol{F}_i, \boldsymbol{F}_v$ are the inviscid and viscous flux functions, respectively. Equation (6) is discretized by means of the discontinuous Galerkin method, where the viscous terms are treated using the local DG (LDG) scheme.[8]

III. The discontinuous Galerkin method

We briefly summarize the tensor-product discontinuous Galerkin method for the conservation law

$$u_t + \nabla \cdot \boldsymbol{F}(u) = 0. \tag{7}$$

First, we discretize the spatial domain Ω using a quadrilateral or hexahedral mesh,

$$\mathcal{T}_h = \left\{ K_j : \bigcup_k K_j = \Omega \right\}. \tag{8}$$

Here, each element K_j is the image of the reference element (the unit cube $[0,1]^d$ in d dimensions), under a transformation map T_j. Then, we define our finite element function space,

$$V_h = \left\{ v_h : v_h|_{K_j} \in V(K_j) \text{ for all } K_j \in \mathcal{T}_h \right\}, \tag{9}$$

where the local function space $V(K_j)$ is defined on each element by means of a tensor-product basis. No continuity is enforced across element interfaces.

We now describe the construction of the space $V(K_j)$, where, for simplicity, we take $d = 2$. We fix a polynomial degree p, and consider $p+1$ nodes x_i in the unit interval $[0,1]$. For each node x_i, we define the basis function ϕ_i to be the unique polynomial of degree at most p satisfying $\phi_i(x_j) = \delta_{ij}$. We then define $(p+1)^2$ functions $\Phi_{ij}(x,y) = \phi_i(x)\phi_j(y)$. It is clear that these functions satisfy $\Phi_{ij}(x_k, x_\ell) = \delta_{ik}\delta_{j\ell}$, and thus give us a nodal basis on $[0,1]\times[0,1]$ with nodes (x_i, x_j). For a given element K, we have $K = T([0,1]^2)$, allowing us to write for each $(x,y) \in K$, $(x,y) = T(\xi,\eta)$, for some $(\xi,\eta) \in [0,1]^2$. Thus, we define the basis functions $\tilde{\Phi}_{ij}$ on the element K by $\tilde{\Phi}_{ij}(x,y) = \Phi_{ij}(\xi,\eta)$. In other words, $\tilde{\Phi}_{ij} = \Phi_{ij} \circ T^{-1}$. The local space $V(K)$ is defined as the span of the basis functions $\tilde{\Phi}_{ij}$.

In order to obtain the DG formulation for (7), we look for an approximate solution $u_h \in V_h$, multiply by a test function $v_h \in V_h$, and integrate by parts on each element $K \in \mathcal{T}_h$ to obtain

$$\int_K \partial_t(u_h) v_h \, dx = \int_K \boldsymbol{F}(u_h) \cdot \nabla v_h \, dx - \int_{\partial K} \widehat{F}(u_h^-, u_h^+, \boldsymbol{n}) v_h \, ds, \tag{10}$$

where $\widehat{F}$ is a *numerical flux function*, and u_h^- and u_h^+ are the traces of u_h on ∂K on the inside and outside of the element K, respectively. The integrals are approximated using a numerical quadrature rule, and the time derivative $\partial_t(u_h)$ is integrated using a method of lines approach. The second-order system in equation (6) is transformed into a system of first-order equations, with appropriate numerical fluxes chosen according to the LDG methodology.[8]

III.A. Sum-factorization approach

A main advantage of using tensor-product elements is the potential to greatly reduce the computational complexity of the method by means of the sum-factorization approach.[18] As an illustration of this approach, we consider the volume integral appearing on the right-hand side of (10). We approximate this integral by means of a tensor-product quadrature rule. We first consider the one-dimensional quadrature rule on $[0,1]$ with weights w_α and abscissa x_α, where $1 \le \alpha \le \mu$, and the number of quadrature points μ is taken to be a constant multiple of the polynomial degree p. These points give rise to the tensor-product quadrature

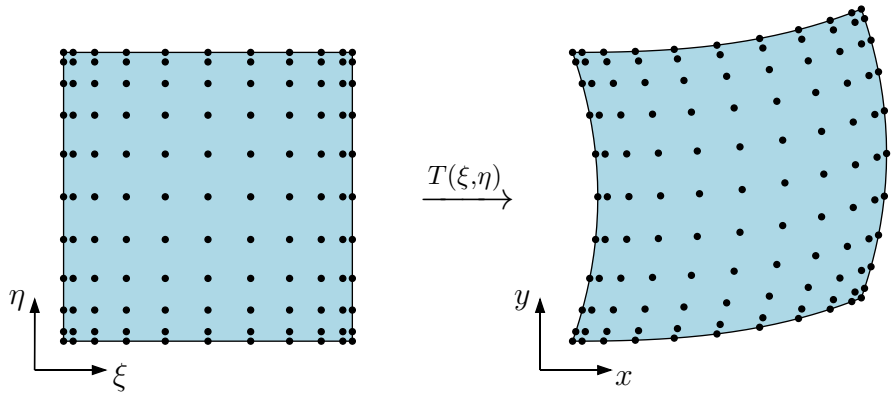


Figure 1: Example of a mapped quadrilateral element with $p = 10$ DG nodes.

rule on $[0,1]^2$, given by $w_{\alpha\beta} = w_\alpha w_\beta$, and $x_{\alpha\beta} = (x_\alpha, x_\beta)$. Then, for a fixed test function $v_h = \Phi_{k\ell}$, we approximate the given volume integral as

$$\sum_{\alpha,\beta} w_{\alpha,\beta} \boldsymbol{F}\left(\sum_{ij} u_{ij}\Phi_{ij}(x_\alpha, x_\beta)\right) \cdot \nabla\Phi_{k\ell}(x_\alpha, x_\beta), \tag{11}$$

where the approximate solution u_h is expanded in terms of the basis functions as

$$u_h(x,y) = \sum_{ij} u_{ij}\Phi_{ij}(x,y). \tag{12}$$

Computing the above sum first requires the evaluation of the solution function u_h at each of the quadrature nodes $x_{\alpha\beta}$. The naive computation requires $\mu^2(p+1)^2$ operations. However, this evaluation can be performed by factoring the sum as

$$\sum_{ij} u_{ij}\Phi_{ij}(x_\alpha, x_\beta) = \sum_i \sum_j u_{ij}\phi_i(x_\alpha)\phi_j(x_\beta) \tag{13}$$

$$= \sum_i \phi_i(x_\alpha) \sum_j u_{ij}\phi_j(x_\beta). \tag{14}$$

Each of the above summations has two free indices, and so the number of operations required to compute this sum can be reduced to $\mathcal{O}(\mu^2(p+1) + (p+1)^2\mu)$.

We can also describe this factorization operation using the convenient language of Kronecker products. If we define the $\mu \times (p+1)$ Vandermonde-type matrix G by

$$G_{\alpha,i} = \phi_i(x_\alpha), \tag{15}$$

then the double-summation in (13) can be written as

$$(G \otimes G)u, \tag{16}$$

where u is the vector of length $(p+1)^2$ whose entries are given by u_{ij}. The factorized version given by (14) can be written as the equivalent operation

$$GUG^T, \tag{17}$$

where U is the $(p+1) \times (p+1)$ matrix, whose columns concatenated give the vector u. It will also be useful to define the one-dimensional differentiation matrix D, which is a $\mu \times (p+1)$ matrix whose entries are given by

$$D_{\alpha,i} = \phi_i'(x_\alpha), \tag{18}$$

and the $\mu \times \mu$ diagonal weight matrix W, whose entries are given by the appropriate quadrature weights. Additionally, we take J_T to be the $\mu^d \times \mu^d$ diagonal matrix whose entries are given by the absolute Jacobian determinant of the transformation map T at each of the quadrature points. Using these matrices, it is possible to write many of main operations needed to implement a DG method in terms of Kronecker products, in a fashion similar to that described above. A short summary of these operations is shown in Table 1.

III.B. Implicit time integration

In order to avoid the restrictive explicit CFL condition, we use an implicit time integration method, such as backward differentiation formulas (BDF) or diagonally-implicit Runge-Kutta (DIRK) methods.[1] The main advantage of such methods is that they remain stable for larger time steps, even in the presence of highly anisotropic elements. Additionally, these methods avoid the restrictive p-dependent explicit stability condition mentioned above. Such implicit methods require the solution of systems of the form

$$Mu_h - \Delta t f(u_h) = r, \tag{19}$$

where f is a nonlinear function corresponding to the right-hand side of equation (10). This system of equations, when solved by means of Newton's method, give rise to linear systems of the form

$$(M - \Delta t J)u = b, \tag{20}$$

Table 1: Kronecker-product form of DG operations

Operation	2D	3D
Evaluate solution at quadrature points	$(G \otimes G)\, u$	$(G \otimes G \otimes G)\, u$
Integrate function f (known at quadrature points) against test functions	$\left(G^T W \otimes G^T W\right) J_T f$	$\left(G^T W \otimes G^T W \otimes G^T W\right) J_T f$
Integrate function $f = (f_1, \ldots, f_d)$ against gradient of test functions	$\left(G^T W \otimes D^T W\right) J_T f_1$ $\left(D^T W \otimes G^T W\right) J_T f_2$	$\left(G^T W \otimes G^T W \otimes D^T W\right) J_T f_1$ $\left(G^T W \otimes D^T W \otimes G^T W\right) J_T f_2$ $\left(D^T W \otimes G^T W \otimes G^T W\right) J_T f_3$

where the matrix J is the Jacobian of the function f with respect to the degrees of freedom of the solution variable u_h.

The most immediate challenge towards efficiently implementing an implicit method for high polynomial degree on tensor-product elements is forming the Jacobian matrix. In general, all the degrees of freedom within one element are coupled, and thus the diagonal blocks of the Jacobian matrix corresponding to a single element are dense $(p+1)^d \times (p+1)^d$ matrices. Therefore, it is impossible to explicitly form this matrix in less than $\mathcal{O}(p^{2d})$ time. To circumvent this, we use a a tensor-product based matrix-free approach combined with an iterative linear solver such as GMRES.[23] Each iteration requires performing a matrix-vector multiplication by the mass matrix and the Jacobian matrix. The matrix-free approach allows us to avoid explicitly forming these matrices, and instead we use sum-factorization techniques to compute the matrix-vector products.

For example, the mass matrix can be written in block-diagonal form. In the two-dimensional case, the entries of the block corresponding with the element K are given by

$$M_{ij,k\ell} = \int_K \Phi_{ij}(x,y)\Phi_{k\ell}(x,y)\; dx. \tag{21}$$

Using the notation from the preceding section, it is possible to see that this block can be written in the form

$$\left(G^T W \otimes G^T W\right) J_T \left(G \otimes G\right), \tag{22}$$

Taking advantage of the Kronecker-product structure of these blocks, it is therefore possible to multiply a vector by the mass matrix in $\mathcal{O}(p^3)$ time, rather than $\mathcal{O}(p^4)$ as dense linear algebra would suggest.

The case of the Jacobian matrix J is somewhat more complicated. J also has a natural block structure, with diagonal blocks corresponding to the coupling of the degrees of freedom within each element, and off-diagonal blocks corresponding to the coupling between neighboring elements. The entries of the diagonal block of J corresponding to element K are given by

$$J_{ij,k\ell} = \frac{\partial f_{ij}}{\partial u_{k\ell}}, \tag{23}$$

where f_{ij} is given by

$$f_{ij} = \int_K F(u) \cdot \nabla \Phi_{ij}\; dx - \int_{\partial K} \widehat{F}(u^-, u^+, n)\Phi_{ij}\; dA \tag{24}$$

Instead of evaluating each of the derivatives $\partial f_{ij}/\partial u_{k\ell}$, we instead consider the matrix-vector product

$$\left(\frac{\partial f_{ij}}{\partial u_{k\ell}}\right) v_{k\ell}, \tag{25}$$

for an arbitrary vector $v_{k\ell}$. Evaluating the integrals in (24) using a standard quadrature rule, and applying

the same sum-factorization technique as above, we obtain

$$\begin{aligned}\left(\frac{\partial f_{ij}}{\partial u_{k\ell}}\right) v_{k\ell} &= \sum_{k=1}^{p+1}\sum_{\ell=1}^{p+1}\sum_{\alpha=1}^{\mu}\sum_{\beta=1}^{\mu} w_\alpha w_\beta \phi_k(x_\alpha)\phi_\ell(x_\beta)\frac{\partial F}{\partial u}(x_\alpha, x_\beta)\cdot\nabla\left(\phi_i(x_\alpha)\phi_j(x_\beta)\right) v_{k\ell} \\ &\quad - \sum_{k=1}^{p+1}\sum_{\ell=1}^{p+1}\sum_{e\in\partial K}\sum_{\alpha=1}^{\mu} w_\alpha \phi_k(x_\alpha^e)\phi_\ell(y_\alpha^e)\frac{\partial \widehat{F}}{\partial u^-}(x_\alpha^e, y_\alpha^e)\phi_i(x_\alpha^e)\phi_j(y_\alpha^e) v_{k\ell} \qquad (26) \\ &= \sum_{\alpha=1}^{\mu} w_\alpha \phi_i'(x_\alpha)\sum_{\beta=1}^{\mu} w_\beta \frac{\partial F_1}{\partial u}(x_\alpha, x_\beta)\phi_j(x_\beta)\sum_{\ell=1}^{p+1}\phi_\ell(x_\beta)\sum_{k=1}^{p+1}\phi_k(x_\alpha) v_{k\ell} \\ &\quad + \sum_{\alpha=1}^{\mu} w_\alpha \phi_i(x_\alpha)\sum_{\beta=1}^{\mu} w_\beta \frac{\partial F_2}{\partial u}(x_\alpha, x_\beta)\phi_j'(x_\beta)\sum_{\ell=1}^{p+1}\phi_\ell(x_\beta)\sum_{k=1}^{p+1}\phi_k(x_\alpha) v_{k\ell} \qquad (27) \\ &\quad + \sum_{e\in\partial K}\sum_{\alpha=1}^{\mu} w_\alpha \frac{\partial \widehat{F}}{\partial u^-}(x_\alpha^e, y_\alpha^e)\phi_i(x_\alpha^e)\phi_j(y_\alpha^e)\sum_{\ell=1}^{p+1}\phi_\ell(y_\alpha^e)\sum_{k=1}^{p+1}\phi_k(x_\alpha^e) v_{k\ell},\end{aligned}$$

where $e \in \partial K$ represents an edge of the element K, and the points (x_α^e, y_α^e) are the quadrature nodes lying on the edge e. We note that each of the summations in the factored form of the expression in (27) has at most two free indices, and therefore it is possible to evaluate the factored form of this expression in $\mathcal{O}(p^3)$ time. A similar technique allows us to evaluated matrix-vector products of the form $(M - \Delta t J)v$ in three spatial dimensions in $\mathcal{O}(p^4)$ time. Therefore, each matrix-free evaluation can be performed in linear time per degree of freedom.

IV. Kronecker-product preconditioner

An effective preconditioner is an important component in obtaining an efficient iterative solver.[25] Typical preconditioners used with discontinuous Galerkin methods include block Jacobi, block Gauss-Seidel, and block ILU preconditioners.[17,22,23] These block-based preconditioners require the explicit computation of the entries of the matrix, together with the inversion of certain blocks. Performing these operations using dense linear algebra would incur a cost of $\mathcal{O}(p^{3d})$ operations, becoming prohibitively expensive as p is taken to be large. To address this issue, we make use of an approximate tensor-product preconditioner, introduced in Reference 19 for hyperbolic conservation laws, and apply it to the compressible Navier-Stokes equations.

This Kronecker-product preconditioner attempts to mimic the tensor-product structure often seen in finite difference and spectral method discretizations of partial differential equations. For example, the Laplacian operator on a cartesian grid can be written as

$$L_{1\mathrm{D}} = T_n, \qquad L_{2\mathrm{D}} = I \otimes T_n + T_n \otimes I, \qquad L_{3\mathrm{D}} = I \otimes I \otimes T_n + I \otimes T_n \otimes I + T_n \otimes I \otimes I, \qquad (28)$$

in one, two, and three spatial dimensions, respectively. DG methods for arbitrary equations will not, in general, give rise to such simple algebraic structures. However, many of the main operations required for a DG method do possess a similar tensor-product structure, as detailed in the previous section in Table 1. This observation motivates approximating the element-wise matrix blocks arising from the implicit time integration of DG discretizations by certain simple sums of Kronecker products. In order to precondition linear systems of the form,

$$(M - \Delta t J)u = b, \qquad (29)$$

we seek to approximate the block Jacobi preconditioner by finding tensor-product approximations to the diagonal blocks of the matrix $M - \Delta t J$. In two spatial dimensions, we are interested in approximating the diagonal block A with a matrix P of the form

$$A \approx P = \sum_{j=1}^{r} A_j \otimes B_j, \qquad (30)$$

for a fixed number of terms r, where each of the matrices A_j and B_j are of size $(p+1)\times(p+1)$. Similarly,

in three dimensions, we look for P of the form

$$A \approx P = \sum_{j=1}^{r} A_j \otimes B_j \otimes C_j. \tag{31}$$

The main tool used to find such approximations is the Kronecker-product singular value decomposition (KSVD),[30] which we describe in the following section.

IV.A. Kronecker-product singular value decomposition

The Nearest Kronecker Problem (NKP)[15,30] is defined as follows: given a matrix $A \in \mathbb{R}^{m\times n}$ (with $m = m_1 m_2$ and $n = n_1 n_2$), and given a fixed number r, find matrices $A_j \in \mathbb{R}^{m_1\times n_1}, B_j \in \mathbb{R}^{m_2\times n_2}$ that minimize the Frobenius norm

$$\left\| A - \sum_{j=1}^{r} A_j \otimes B_j \right\|_F. \tag{32}$$

Van Loan gives the solution to the NKP by means of the singular value decomposition of a *shuffled* matrix $\widetilde{A}$. Given the blocking of A,

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1,n_1} \\ A_{21} & A_{22} & \cdots & A_{2,n_1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m_1,1} & A_{m_1,2} & \cdots & A_{m_1,n_1} \end{pmatrix}, \tag{33}$$

where each block A_{ij} is a $m_2 \times n_2$ matrix, we then define $\widetilde{A}$ to be the $m_1 n_1 \times m_2 n_2$ matrix given by

$$\widetilde{A} = \begin{pmatrix} \widetilde{A}_1 \\ \widetilde{A}_2 \\ \vdots \\ \widetilde{A}_{n_1} \end{pmatrix}, \text{ where } \widetilde{A}_j \text{ is a block of rows given by } \widetilde{A}_j = \begin{pmatrix} \mathrm{vec}(A_{1j})^T \\ \mathrm{vec}(A_{2j})^T \\ \vdots \\ \mathrm{vec}(A_{m_1,j})^T \end{pmatrix}, \tag{34}$$

where the vec operator is defined so that $\mathrm{vec}(A_{ij})$ is the column vector of length $m_2 n_2$ obtained by stacking the columns of A_{ij}. It can be shown that this shuffled matrix has the useful property that, for matrices A_j and B_j,

$$\left\| A - \sum_{j=1}^{r} A_j \otimes B_j \right\|_F = \left\| \widetilde{A} - \sum_{j=1}^{r} \mathrm{vec}(A_j)\,\mathrm{vec}(B_j)^T \right\|_F. \tag{35}$$

This property establishes the equivalence between the r-term NKP and finding the best rank-r approximation to $\widetilde{A}$, which is immediately given by the SVD,

$$\widetilde{A} = U \Sigma V^T. \tag{36}$$

The matrices A_j and B_j from equation (32) are then given by reshaping the left and right singular vectors of $\widetilde{A}$, so that we have

$$\mathrm{vec}(A_j) = \sqrt{\sigma_j} U_j, \qquad \mathrm{vec}(B_j) = \sqrt{\sigma_j} V_j. \tag{37}$$

Although the computation of the SVD is, in general, an expensive operation, it is possible to obtain accurate approximations of the first r singular values and vectors by means of a Lanczos algorithm.[13] This iterative procedure, combined with matrix-free evaluations of products by the shuffles matrices $\widetilde{A}$ and $\widetilde{A}^T$ allow for the efficient computation of optimal (in Frobenius norm) approximations of the form (30).

IV.B. Two- and three-dimensional preconditioners

Using the KSVD as described above, we can find Kronecker-product approximations to the diagonal blocks of the matrix $M - \Delta t J$. In two dimensions, we look for two-term approximations of the form

$$P = A_1 \otimes B_1 + A_2 \otimes B_2. \tag{38}$$

This particular case (*i.e.* when $r = 2$) has the advantage that linear systems of the form $Px = b$ can be efficiently solved by adapting a matrix diagonalization technique.[16,26] Such a system can be transformed by left-multiplication by $A_2^{-1} \otimes B_1^{-1}$ to obtain

$$(A_2^{-1}A_1 \otimes I + I \otimes B_1^{-1}B_2)x = (A_2^{-1} \otimes B_1^{-1})b. \tag{39}$$

Then, we employ the Schur factorizations,

$$A_2^{-1}A_1 = Q_1 T_1 Q_1^T, \tag{40}$$

$$B_1^{-1}B_2 = Q_2 T_2 Q_2^T, \tag{41}$$

where the matrices Q_1 and Q_2 are orthogonal, and T_1 and T_2 are quasi-triangular, to reformulate the above system as

$$(Q_1 \otimes Q_2)(T_1 \otimes I + I \otimes T_2)(Q_1^T \otimes Q_2^T)x = (A_2^{-1} \otimes B_1^{-1})b, \tag{42}$$

which can be solved efficiently by standard means.

In three spatial dimensions, we look for a preconditioner that has the form

$$A \approx P = A_1 \otimes B_1 \otimes C_1 + A_1 \otimes B_2 \otimes C_2, \tag{43}$$

where it is important to note that the same matrix A_1 is present in both terms on the right-hand side. Left-multiplying the linear system $Px = b$ by $A_1^{-1} \otimes B_2^{-1} \otimes C_1^{-1}$ gives

$$(I \otimes B_2^{-1}B_1 \otimes I + I \otimes I \otimes C_1^{-1}C_2)x = (A_1^{-1} \otimes B_2^{-1} \otimes C_1^{-1})b. \tag{44}$$

The same Schur factorization technique as in the two-dimensional case can then be applied, resulting in the system

$$(I \otimes Q_1 \otimes Q_2)(I \otimes T_1 \otimes I + I \otimes I \otimes T_2)(I \otimes Q_1^T \otimes Q_2^T)x = (A_1^{-1} \otimes B_2^{-1} \otimes C_1^{-1})b. \tag{45}$$

As in the previous case, is also possible to efficiently solve such systems by standard means.

IV.C. Application to Navier-Stokes

In this paper, we are interested in the application of the above techniques to the specific case of the compressible Navier-Stokes equations. We first note that in this case, the solution consists of $d+2$ components, $(\rho, \rho\boldsymbol{u}, \rho E)$, where $\boldsymbol{u} \in \mathbb{R}^d$ is the velocity vector. We let n_c indicate the number of solution components. Then, we can consider the Jacobian matrix to be composed of blocks of size $n_c(p+1)^d \times n_c(p+1)^d$. In the two dimensional case, these blocks are approximated by the sum $A_1 \otimes B_1 + A_2 \otimes B_2$, where A_1 and A_2 are of size $n_c(p+1) \times n_c(p+1)$, and the matrices B_1 and B_2 are of size $(p+1) \times (p+1)$. In the three-dimensional case, we recall that the preconditioner takes the form $A_1 \otimes B_1 \otimes C_1 + A_1 \otimes B_2 \otimes C_2$. Here, A_1 is of size $n_c(p+1) \times n_c(p+1)$, and the remaining matrices are all $(p+1) \times (p+1)$. Additionally, the Navier-Stokes equations differs from previous applications of these preconditioners because of the second-order terms corresponding to the viscous fluxes. In this paper, we discretize these second-order terms using the LDG method, whose primal form involves volume integrals of certain lifting operators.[2] The computation of these integrals is not immediately amenable to sum-factorization, and therefore, in this work, we choose to include only the inviscid flux in the preconditioner. This choice results in a preconditioner that captures less of the dynamics of the equations, in particular at low Reynolds numbers, but we have found that for moderate to high Reynolds cases, it remains effective. The question of how to effectively capture second-order terms in the tensor-product preconditioner remains an open area of research.

V. Results

In order to demonstrate the application of the approximate Kronecker-product preconditioner to the compressible Navier-Stokes equations, we use two benchmark test cases. The first test case is the direct numerical simulation of the compressible Taylor-Green vortex at Reynolds number 1600, a widely used example with available reference data and diagnostics.[33] This test case features a cartesian hexahedral grid, and periodic boundary conditions. The second test case is the large eddy simulation of viscous flow over a NACA 0012 airfoil, demonstrating the applicability of the method to unstructured meshes with anisotropic elements, complex geometries, boundary-layer phenomena, and strong viscous effects.

V.A. Direct numerical simulation of Taylor-Green vortex

The direct numerical simulation of the compressible Taylor-Green vortex at Re = 1600 was selected as a difficult benchmark test problem for the first International Workshop on High-Order CFD Methods.[33] This problem has been much studied,[5,6,10,31] and fully-resolved reference data are available, making this test case particularly useful.

V.A.1. Geometry and initial conditions

The spatial domain is taken to be the periodic cube $-\pi \leq x, y, z \leq \pi$. The initial conditions are given by

$$u(x,y,z) = u_0 \sin(x)\cos(y)\cos(z) \tag{46}$$
$$v(x,y,z) = -u_0 \cos(x)\sin(y)\cos(z) \tag{47}$$
$$w(x,y,z) = 0 \tag{48}$$
$$p(x,y,z) = p_0 + \frac{\rho_0 u_0^2}{16}\left(\cos(2x)+\cos(2y)\right)\left(\cos(2z)+2\right), \tag{49}$$

where we take the parameters to be $\gamma = 1.4$, $\mathrm{Pr} = 0.71$, $u_0 = 1$, $\rho_0 = 1$, with Mach number $M_0 = u_0/c_0 = 0.10$, where c_0 is the speed of sound computed in accordance with the pressure p_0. The initial density distribution is then given by $\rho = p\rho_0/p_0$. The characteristic convective time is given by $t_\mathrm{c} = 1$, and the final time is $t_\mathrm{f} = 20t_\mathrm{c}$. We discretize the geometry using a sequence of regular hexahedral grids, and consider polynomial degrees varying from $p = 3$ to $p = 15$. The number of numerical degrees of freedom per solution component ranges from 64^3 to 240^3. A table of configurations, including grid size (number of elements per dimension), polynomial degree, and total number of numerical degrees of freedom per solution component is listed in Table 2.

Table 2: Grid configurations for Taylor-Green problem

Grid size n_x	p	Numerical DOFs
8	7	64^3
8	15	128^3
16	3	64^3
16	7	128^3
32	3	128^3
48	4	240^3

V.A.2. Diagnostics

We compare our computed solution at several resolutions with a known reference solution computed using a pseudo-spectral method with 512^3 degrees of freedom per solution component. In order to compare the results, we consider several quantities integrated over the domain. First, we consider the mean kinetic energy,

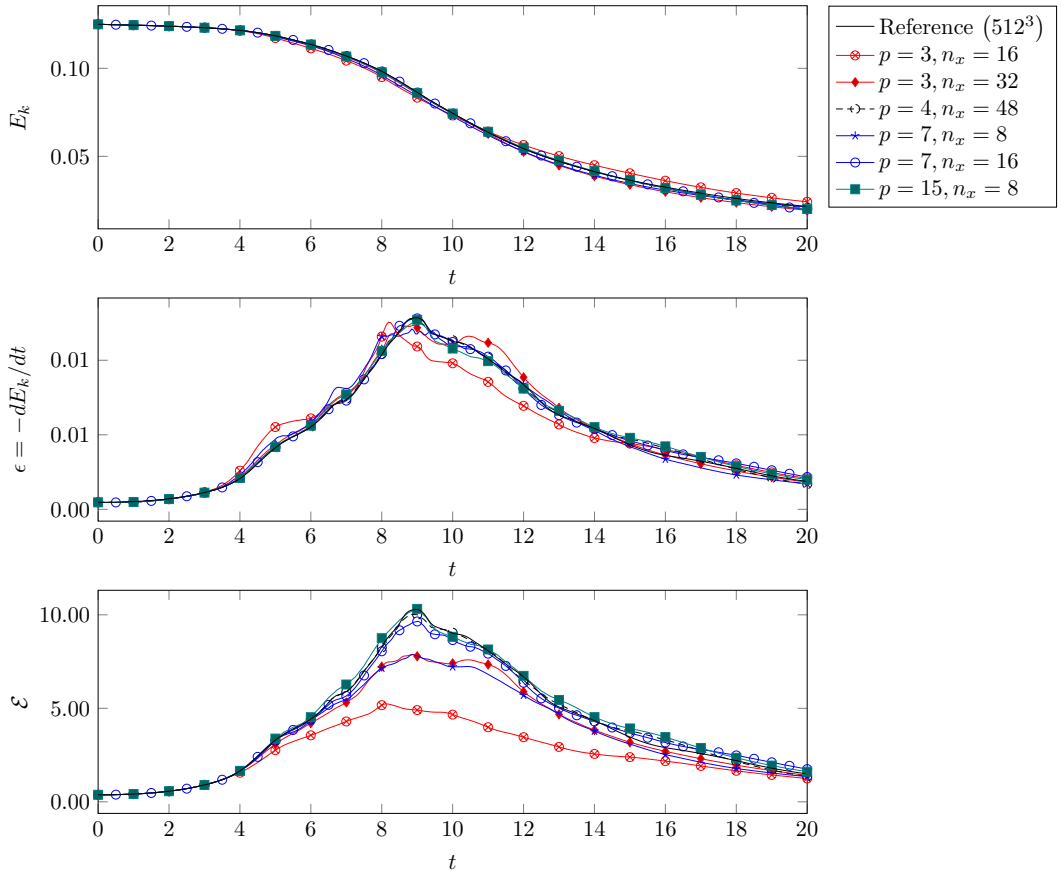
$$E_k(t) = \frac{1}{\rho_0|\Omega|}\int_\Omega \rho \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2}\,dx, \tag{50}$$

where $|\Omega| = (2\pi)^3$ is the volume of the domain. From the time-evolution of this quantity, we can also compute the kinetic energy dissipation rate (KEDR), given by

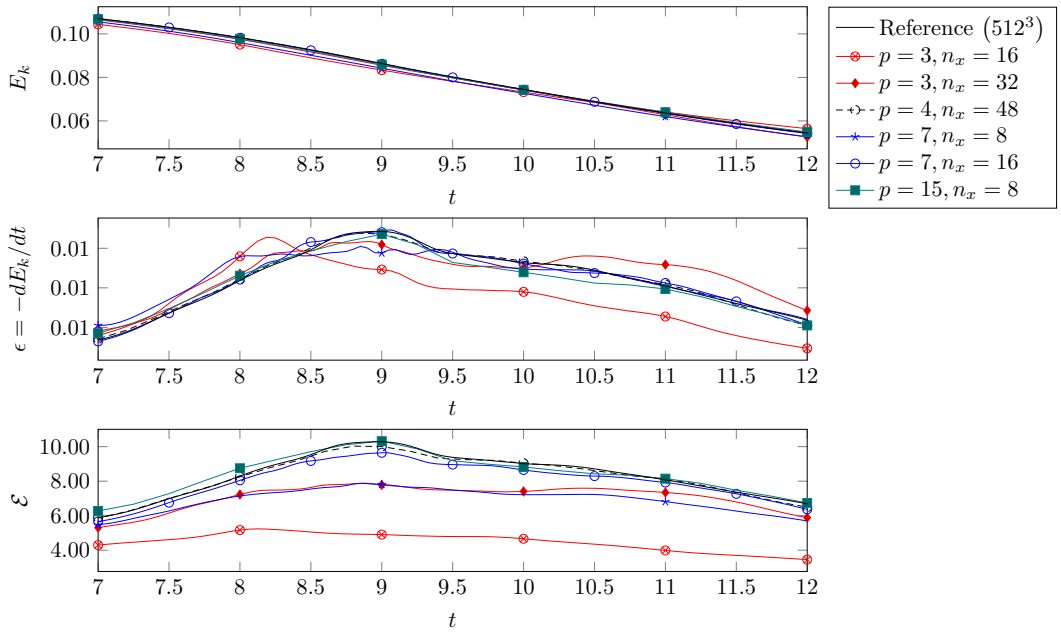
$$\epsilon(t) = -\frac{dE_k}{dt}(t). \tag{51}$$

Additionally, we consider the mean enstrophy over the domain,

$$\mathcal{E}(t) = \frac{1}{\rho_0|\Omega|}\int_\Omega \rho \frac{\omega\cdot\omega}{2}\,dx. \tag{52}$$



(a) Time evolution of diagnostics for $t \in [0, 20t_c]$.



(b) Zoom-in around point of peak dissipation.

Figure 2: Time evolution of kinetic energy E_k, kinetic energy dissipation rate (KEDR) ϵ, and enstrophy $\mathcal{E}$, for Taylor-Green test case. Comparison of various DG configurations with reference pseudo-spectral solution.

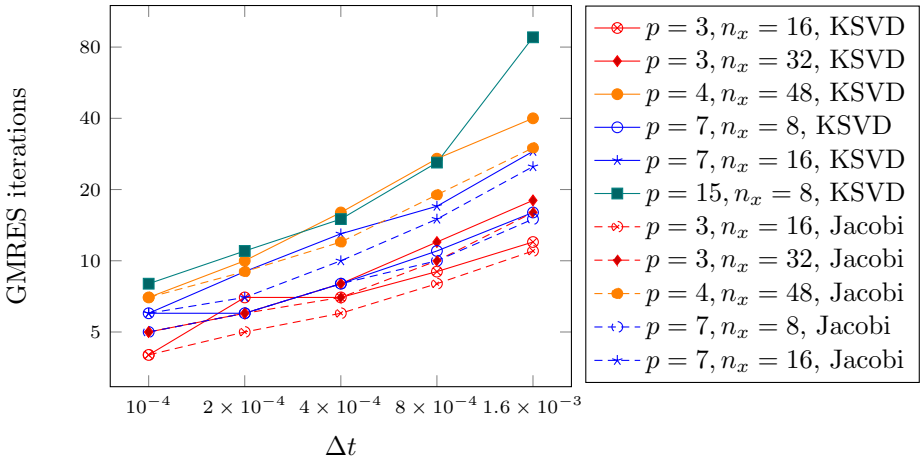


Figure 3: Number of GMRES iterations required for approximate Kronecker-product preconditioner and exact block Jacobi preconditioner vs. timestep, Taylor-Green test case. Kronecker-product results shown with solid lines, block Jacobi with dashed lines.

For incompressible flow, the relationship $\epsilon = 2\mu\mathcal{E}/\rho_0$ holds, and for low-Mach compressible flow, this relationship holds approximately. Indeed, we notice that the profiles of $\mathcal{E}$ and ϵ are close to indistinguishable, indicating that compressible effects do not play a large role in this test problem. For each of the configurations listed above, we compare the results with the pseudo-spectral reference solution in Figure 2a. In Figure 2b, we present plots of the same quantities, zoomed in around the point of peak dissipation at about $t = 9t_c$. Additionally, the isosurfaces of the vorticity norm $|\omega|$ at times $t = 0, 2, 4, 6, 8, 10\ t_c$ are shown in Figure 4, illustrating the transition to turbulence, and subsequent decay.

The simulation with 240^3 degrees of freedom per component ($p = 4, n_x = 48$) reproduces almost exactly all three of the profiles. We note that even for the severely under-resolved cases with 64^3 degrees of freedom, the kinetic energy profiles closely match the reference data. However, the low-order $p = 3$ discretizations tend to under-predict the enstrophy and KEDR. With an equal number of degrees of freedom, the $p = 7, n_x = 8$ discretization more closely matches the reference profiles than the $p = 3, n_x = 16$ configuration, providing accuracy comparable to the $p = 3, n_x = 32$ case, with one eighth the number of degrees of freedom. Similarly, despite the exceedingly coarse mesh, the $p = 15, n_x = 8$ configuration captured the diagnostics more accurately than all other configurations considered with equal number of degrees of freedom. This $p = 15$ discretization successfully captured the peak enstrophy and energy dissipation rate, as shown in Figure 2b, and, in fact, matches the reference data comparably to the finest discretization with $n_x = 48$. The improved accuracy per degree of freedom for higher polynomial degree p motivates the use of very high order methods.

V.A.3. Preconditioner efficiency

In this section we examine the efficiency of the approximate tensor-product preconditioner for the Taylor-Green test case, for each of the configurations listed in Table 2. For each configuration, we choose a range of timesteps, the smallest timestep corresponding to $\Delta t = 10^{-4}$. We then increase the timestep four times in multiples of two, until we obtain a largest timestep of $\Delta t = 1.6 \times 10^{-3}$. We measure the number of iterations required to perform one backward Euler step (*i.e.* Newton solve) to find the average number of iterations per linear solve. We also compute the average number of iterations per linear solve using the exact block Jacobi preconditioner, for all configurations except for $p = 15, n_x = 8$, for which the test did not complete because of its excessive memory and runtime requirements. The iteration counts are displayed in Figure 3. Using this methodology, we can compare the effectiveness of the approximate Kronecker-product preconditioner with the exact block Jacobi preconditioner.

As expected, for both the Jacobi and Kronecker preconditioners, we observe that the number of iterations per linear solve increases with Δt. We also observe that in both cases, this dependence is sublinear, with the exception of the case $p = 15, n_x = 8, \Delta t = 1.6 \times 10^{-3}$. For a majority of cases, the Kronecker-product preconditioner resulted in very similar (or sometimes identical) iteration counts. We conclude that the

Kronecker-product preconditioners are able to achieve iteration counts very similar to the exact block Jacobi preconditioner, but with large savings in terms of computational complexity and memory requirements. For the block Jacobi preconditioner, storing the diagonal block requires $(n_c(p+1)^3)^2$ memory per block, and computing the LU factorization requires $\mathcal{O}(p^9)$ operations. On the other hand, the memory requirements for the Kronecker-product preconditioner grows linearly in p with the number of degrees of freedom, while the computational complexity scales as $\mathcal{O}(p^5)$, indicating that a large savings in computational complexity can be achieved at the cost of a modest increase in iteration count.

Table 3: Average number of GMRES iterations per Newton solve for Taylor-Green test case

(a) $p = 3, n_x = 16$

Δt	Jacobi	KSVD
1×10^{-4}	4	4
2×10^{-4}	5	7
4×10^{-4}	6	7
8×10^{-4}	8	9
1.6×10^{-3}	11	12

(b) $p = 3, n_x = 32$

Δt	Jacobi	KSVD
1×10^{-4}	5	5
2×10^{-4}	6	6
4×10^{-4}	7	8
8×10^{-4}	10	12
1.6×10^{-3}	16	18

(c) $p = 4, n_x = 48$

Δt	Jacobi	KSVD
1×10^{-4}	7	7
2×10^{-4}	9	10
4×10^{-4}	12	16
8×10^{-4}	19	27
1.6×10^{-3}	30	40

(d) $p = 7, n_x = 8$

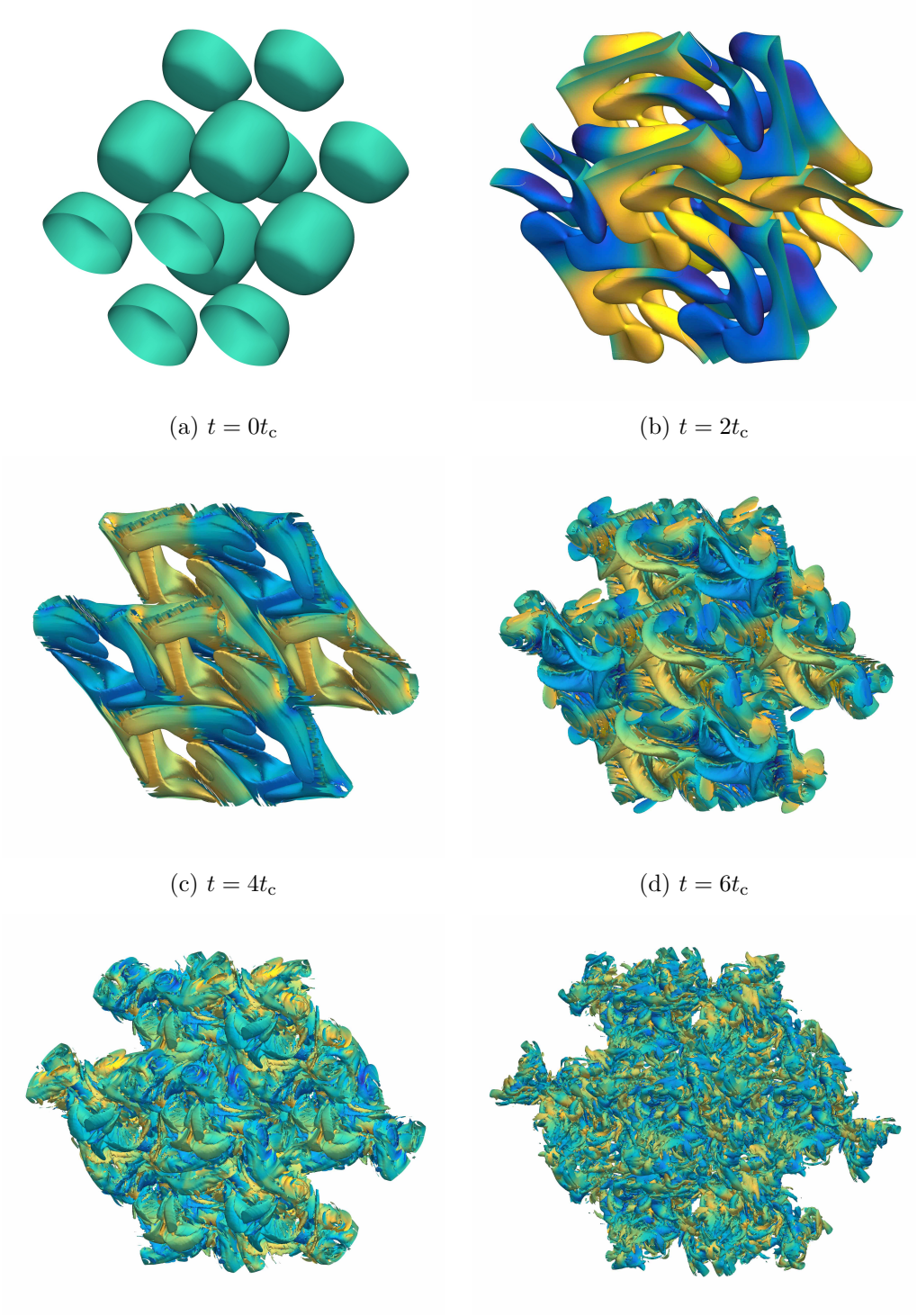
Δt	Jacobi	KSVD
1×10^{-4}	5	6
2×10^{-4}	6	6
4×10^{-4}	8	8
8×10^{-4}	10	11
1.6×10^{-3}	15	16

(e) $p = 7, n_x = 16$

Δt	Jacobi	KSVD
1×10^{-4}	6	6
2×10^{-4}	7	9
4×10^{-4}	10	13
8×10^{-4}	15	17
1.6×10^{-3}	25	29

(f) $p = 15, n_x = 8$

Δt	Jacobi	KSVD
1×10^{-4}	–	8
2×10^{-4}	–	11
4×10^{-4}	–	15
8×10^{-4}	–	26
1.6×10^{-3}	–	88



(a) $t = 0t_c$ (b) $t = 2t_c$

(c) $t = 4t_c$ (d) $t = 6t_c$

(c) $t = 8t_c$ (d) $t = 10t_c$

Figure 4: Compressible Taylor-Green vortex at Re = 1600, isosurfaces of vorticity norm $|\omega|$, colored by helicity $H = \omega \cdot \boldsymbol{u}$.

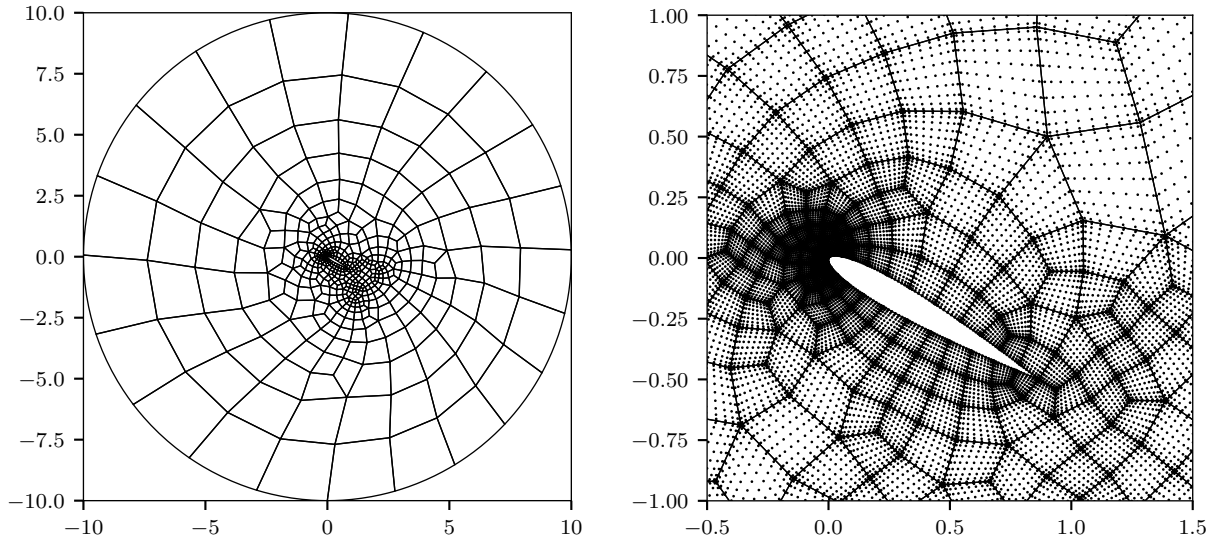


Figure 5: Unstructured NACA mesh (zoom-in around airfoil shown with $p = 10$ DG nodes).

V.B. Large eddy simulation of viscous flow over NACA 0012 airfoil

In this section, we consider the viscous, compressible, two-dimensional flow over a NACA 0012 airfoil at $\mathrm{Re} = 1600$, with angle of attack 30°. The domain is taken to be a disk of radius 10 centered at $(0, 0)$, and the leading edge of the airfoil is placed at the origin. A no-slip condition is enforced on the surface of the airfoil, and far-field conditions are enforced on the exterior domain boundary. The Mach number is set to be $M_0 = 0.2$, and the far-field velocity is set to unity in the freestream direction. The domain is discretized using an unstructured quadrilateral mesh, which is refined around the edges of the airfoil and in its wake. Isoparametric mappings are used to curve the elements on the domain boundaries. The mesh is shown in Figure 5.

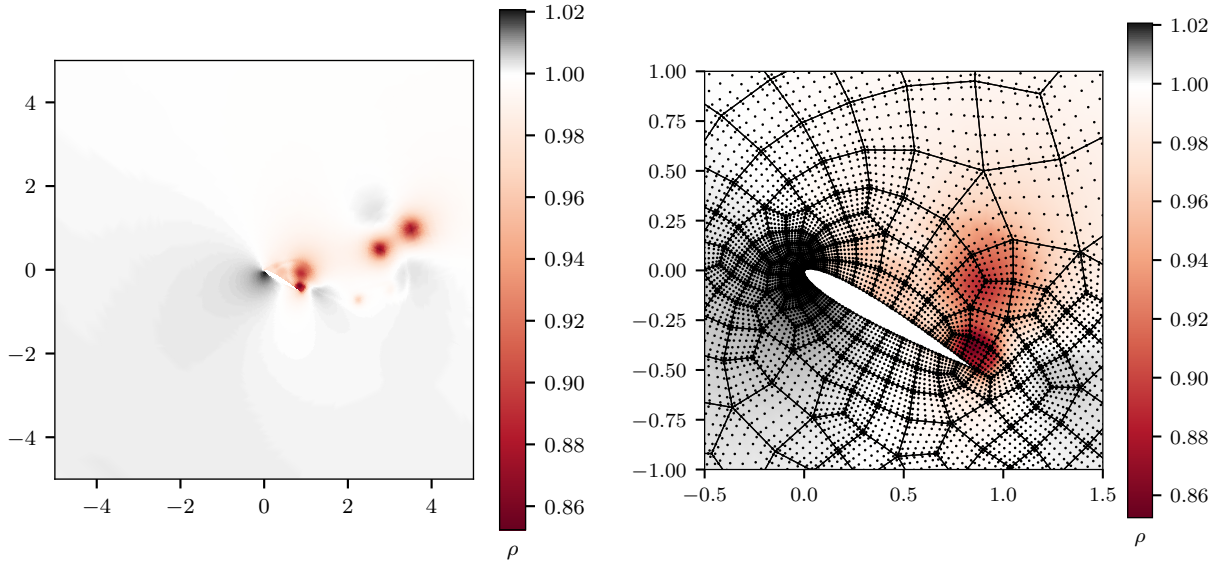


Figure 6: Solution (density) to NACA test problem at $t = 7.5$ s, with $p = 7$ polynomial degree.

In order to obtain a representative solution, we integrate the equations for 7.5 s, at which point vortices have begun to form in the wake of the wing. The solution at this time (computed using polynomial degree of $p = 7$) is displayed in Figure 6. Then, in order to compare solver performance, we linearize about this solution. We compare the number of iterations required to perform one backward Euler step with $p = 3, 4, 7, 10$, for a range of timesteps. For each p, the smallest timestep is determined by the explicit

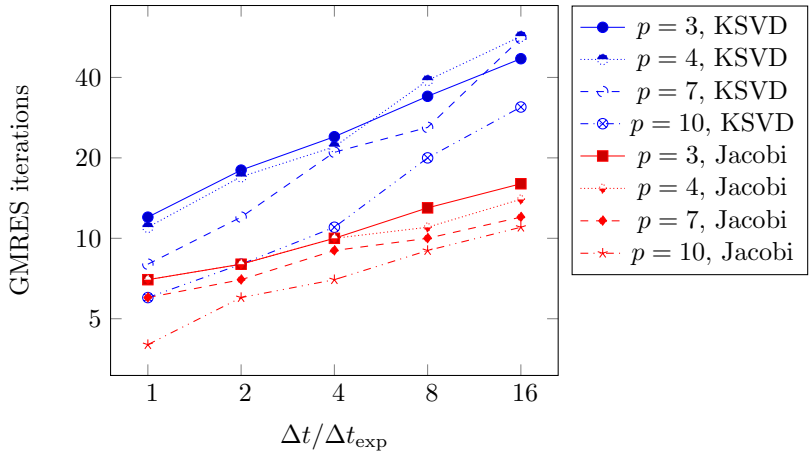


Figure 7: Number of GMRES iterations required for approximate Kronecker-product preconditioner and exact block Jacobi preconditioner vs. timestep ratio $\Delta t/\Delta t_{\text{exp}}$, NACA test case, for $p = 3, 4, 7, 10$.

stability condition, denoted by Δt_{exp}. Then, we increase Δt four times, each by a factor of two, to obtain a final timestep of $16\Delta t_{\text{exp}}$. The average number of iterations required per Newton solve are shown in Table 4 and Figure 7. In this case, we notice that for the smallest timestep, Δt_{exp}, the number of iterations for both preconditioners is quite similar. For both the block Jacobi and Kronecker-product preconditioners, the iteration count increases with Δt, although the rate of increase for the KSVD-based preconditioner is greater than that of the Jacobi preconditioner. For $\Delta t = 4\Delta t_{\text{exp}}$, about two times as many iterations are required, and for $\Delta t = 16\Delta t_{\text{exp}}$, about three to four times as many iterations are required. Of course, for large polynomial degrees p, the increased cost due to the larger number of iterations will be offset by the decrease in computational complexity associated with forming and applying the preconditioner.

Table 4: Average number of GMRES iterations per Newton solve for 2D NACA test case

(a) $p = 3$

Δt	Jacobi	KSVD
0.0005	7	12
0.0010	8	18
0.0020	10	24
0.0040	13	34
0.0080	16	47

(b) $p = 4$

Δt	Jacobi	KSVD
0.00025	7	11
0.00050	8	17
0.00100	10	22
0.00200	11	39
0.00400	14	57

(c) $p = 7$

Δt	Jacobi	KSVD
0.00005	6	8
0.00010	7	12
0.00020	9	21
0.00040	10	26
0.00080	12	56

(d) $p = 10$

Δt	Jacobi	KSVD
0.00001	4	6
0.00002	6	8
0.00004	7	11
0.00008	9	20
0.00016	11	31

VI. Conclusion and Future Work

In this paper, we have applied approximate Kronecker-product preconditioners to two DNS and LES benchmark test cases, in two and three spatial dimensions. These preconditioners improve upon the p-dependence of the computational complexity associated with traditional block-based preconditioners such as block Jacobi or block Gauss-Seidel by exploiting the tensor-product structure of DG methods with quadrilateral or hexahedral elements. We have found that for moderate timesteps, these approximate preconditioners result in comparable iteration counts when compared with exact block Jacobi, even for high polynomial degrees p. These results demonstrate a promising approach for the implicit time integration of very high-order DG methods for CFD problems.

For larger timesteps Δt and high polynomial degree p, we see a degradation in the performance of the approximate preconditioner. Improving the iteration counts in these cases is a topic of future research. Possible methods for such improvements include systematic treatment of the viscous fluxes, incorporating the approximate preconditioner as a smoother for p-multigrid methods, and application of ILU-based approaches.

Acknowledgments

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The first author was supported in part 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. The second author was supported in part by the AFOSR Computational Mathematics program under grant number FA9550-15-1-0010.

References

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

[2]Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and L. Donatella Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[3]F. Bassi, L. Botti, A. Colombo, A. Crivellini, A. Ghidoni, and F. Massa. On the development of an implicit high-order discontinuous Galerkin method for DNS and implicit LES of turbulent flows. *European Journal of Mechanics. B. Fluids*, 55(part 2):367–379, 2016.

[4]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.

[5]Marc E. Brachet, Daniel I. Meiron, Steven A. Orszag, B. G. Nickel, Rudolf H. Morf, and Uriel Frisch. Small-scale structure of the Taylor-Green vortex. *Journal of Fluid Mechanics*, 130:411–452, 1983.

[6]C. Carton de Wiart, K. Hillewaert, M. Duponcheel, and G. Winckelmans. Assessment of a discontinuous Galerkin method for the simulation of vortical flows at high Reynolds number. *International Journal for Numerical Methods in Fluids*, 74(7):469–493, 2014.

[7]Jean-Baptiste Chapelier, Marta De La Llave Plata, and Florent Renac. Inviscid and viscous simulations of the Taylor-Green vortex flow using a modal discontinuous Galerkin approach. In *42nd AIAA Fluid Dynamics Conference and Exhibit*. American Institute of Aeronautics and Astronautics, June 2012.

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

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

[10]James DeBonis. Solutions of the Taylor-Green vortex problem using high-resolution explicit finite difference methods. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, January 2013.

[11]Laslo T. Diosady and Scott M. Murman. Design of a variational multiscale method for high Reynolds number compressible flows. In *21st AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, June 2013.

[12]Gregor J. Gassner and Andrea D. Beck. On the accuracy of high-order discretizations for underresolved turbulence simulations. *Theoretical and Computational Fluid Dynamics*, 27(3):221–237, 2013.

[13]Gene H. Golub, Franklin T. Luk, and Michael L. Overton. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software (TOMS)*, 7(2):149–169, 1981.

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

[15]Charles F. Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(12):85 – 100, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

[16]Robert E. Lynch, John R. Rice, and Donald H. Thomas. Direct solution of partial difference equations by tensor product methods. *Numerische Mathematik*, 6(1):185–199, 1964.

[17]K.-A. Mardal, T. K. Nilssen, and G. A. Staff. Order-optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs. *SIAM Journal on Scientific Computing*, 29(1):361–375, 2007.

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

[19]Will Pazner and Per-Olof Persson. Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods. *Submitted to Journal of Computational Physics (Under Review)*, abs/1704.04549, 2017.

[20]Jaime Peraire and Per-Olof 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.

[21]Per-Olof 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.

[22]Per-Olof Persson and Jaime Peraire. An efficient low memory implicit DG algorithm for time dependent problems. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 113, 2006.

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

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

[25]Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[26]Jie Shen, Tao Tang, and Li-Lian Wang. *Separable Multi-Dimensional Domains*, pages 299–366. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[27]Chi-Wang Shu, Wai-Sun Don, David Gottlieb, Oleg Schilling, and Leland Jameson. Numerical convergence study of nearly incompressible, inviscid Taylor-Green vortex flow. *Journal of Scientific Computing*, 24(1):1–27, 2005.

[28]A. Uranga, P.-O. Persson, M. Drela, and J. Peraire. Implicit large eddy simulation of transition to turbulence at low Reynolds numbers using a discontinuous Galerkin method. *International Journal for Numerical Methods in Engineering*, 87(1-5):232–261, 2011.

[29]Alejandra Uranga, Per-Olof Persson, Mark Drela, and Jaime Peraire. Implicit large eddy simulation of transitional flows over airfoils and wings. In *19th AIAA Computational Fluid Dynamics*. American Institute of Aeronautics and Astronautics, June 2009.

[30]Charles F. Van Loan and Nikos Pitsianis. Approximation with Kronecker products. In *Linear Algebra for Large Scale and Real-Time Applications*, pages 293–314. Springer, 1993.

[31]Wim M. Van Rees, Anthony Leonard, D. I. Pullin, and Petros Koumoutsakos. A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds numbers. *Journal of Computational Physics*, 230(8):2794–2805, 2011.

[32]Brian C. Vermeire, Siva Nadarajah, and Paul G. Tucker. Implicit large eddy simulation using the high-order correction procedure via reconstruction scheme. *International Journal for Numerical Methods in Fluids*, 82(5):231–260, 2016.

[33]Z.J. Wang, Krzysztof Fidkowski, Rémi Abgrall, Francesco Bassi, Doru Caraeni, Andrew Cary, Herman Deconinck, Ralf Hartmann, Koen Hillewaert, H.T. Huynh, Norbert Kroll, Georg May, Per-Olof Persson, Bram van Leer, and Miguel Visbal. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.