Tensor-Product Preconditioners for Higher-Order Space-Time Discontinuous Galerkin Methods

Laslo T. Diosady[a,*], Scott M. Murman[b]

[a]*Science & Technology Corp., NASA Ames Research Center, Mountain View, CA, 94035*
[b]*NASA Ames Research Center, Mountain View, CA, 94035*

Abstract

A space-time discontinuous-Galerkin spectral-element discretization is presented for direct numerical simulation of the compressible Navier-Stokes equations. An efficient solution technique based on a matrix-free Newton-Krylov method is developed in order to overcome the stiffness associated with high solution order. The use of tensor-product basis functions is key to maintaining efficiency at high-order. Efficient preconditioning methods are presented which can take advantage of the tensor-product formulation. A diagonalized Alternating-Direction-Implicit (ADI) scheme is extended to the space-time discontinuous Galerkin discretization. A new preconditioner for the compressible Euler/Navier-Stokes equations based on the fast-diagonalization method is also presented. Numerical results demonstrate the effectiveness of these preconditioners for the direct numerical simulation of subsonic turbulent flows.

1. Introduction

Higher-order methods are efficient numerical tools for simulations requiring high spatial and temporal resolution, allowing for solutions with fewer degrees of freedom and lower computational cost than traditional second-order CFD methods[1]. In this work, we use a space-time discontinuous-Galerkin (DG) finite-element method, which extends to arbitrary order of accuracy in both space and time. Higher-order DG methods have been widely used for the solution of the compressible Euler and Navier-Stokes equations [2, 3, 4, 5]. Higher-order DG methods are particularly attractive due to the possibility of using local h- and p-adaptation. In particular, the use of a space-time formulation allows for local adaptation in both the spatial and temporal directions, potentially leading to a significant reduction in cost as the increased resolution in time is only applied where necessary.

*Corresponding author
Email addresses: laslo.diosady@nasa.gov (Laslo T. Diosady), scott.murman@nasa.gov (Scott M. Murman)

While space-time DG formulations have been previously developed for the compressible Navier-Stokes equations [6, 7, 8]; higher-order space-time formulations are often considered too expensive for practical engineering simulations, as they require the solution of a globally coupled system of equations for each time-slab. For large three-dimensional simulations the cost of storing the linearization for a single step of an implicit scheme may be prohibitively expensive. Using a space-time formulation, this storage cost is further scaled by the order of the basis used in the temporal direction. We overcome this limitation by using a matrix-free Newton-Krylov method. Efficient preconditioning schemes are necessary to overcome the stiffness associated with large polynomial orders. In addition, the use of tensor-product basis functions is key to maintaining efficiency at high-order. We present two classes of efficient preconditioning methods which take advantage of the tensor-product formulation: a diagonalized Alternating-Direction-Implicit scheme, and a scheme based on the fast-diagonalization method.

Alternating-Direction-Implicit (ADI) schemes were first introduced for the solution of elliptic partial differential equations by Peaceman and Rachford [9]. The scheme presented in this work derives from that of Beam and Warming [10] who presented ADI methods for finite-difference discretization of hyperbolic systems of equations. Specifically, we present a diagonalized ADI scheme, which has previously been used in finite-difference simulations of the compressible Euler and Navier-Stokes equations [11]. We previously extended the diagonalized ADI scheme to a discontinuous-Galerkin finite-element discretization [12]. The diagonalized ADI scheme has been extended to the solution of time-spectral finite-difference discretizations [13]. In this work, we extend the diagonalized-ADI scheme to the space-time DG discretization using an entropy-stable formulation.

The fast diagonalization method (FDM) was presented by Lynch et. al [14] for the direct solution of finite-difference discretizations of scalar partial differential equations. As an exact solver, the FDM has limited applicability to constant-coefficient problems on rectangular geometries. However, Lottes and Fischer[15] used FDM as an effective block preconditioner for a spectral-element discretization of the incompressible Navier-Stokes equations. Specifically, FDM was applied to the scalar elliptic problems arising in a semi-implicit time-stepping scheme. The use of FDM as a preconditioner was extended to scalar advection-diffusion problems by Lott and Elman[16], who used it as part of an iterative structuring preconditioner for a spectral-element discretization. In this work we extend the use of FDM-based preconditioners to space-time DG discretizations of the scalar advection-diffusion equation and the compressible Navier-Stokes equations.

In Section 2, we present the space-time discontinuous-Galerkin discretization and discuss the use of tensor-products basis. In Section 3, we present the solution technique based on a matrix-free Newton-Krylov scheme. In Section 4, we introduce the ADI and FDM preconditioners for the scalar-advection diffusion equations. In Section 5, we present the extension of these schemes to the compressible Navier-Stokes equations. In Section 6, we apply the ADI and FDM

preconditioners for the direct numerical simulation of compressible turbulent flows. Lastly, Section 7 provides summary and conclusions.

2. Discretization

The compressible Navier-Stokes equations are written in conservative form as:

$$\boldsymbol{u}_{,t} + \nabla \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V) = 0, \tag{1}$$

where $(\cdot)_{,t}$ denotes partial differentiation with respect to time. The conservative state vector is

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho \boldsymbol{V} \\ \rho E \end{bmatrix}, \tag{2}$$

where ρ is the density, $\boldsymbol{V}$ is the velocity vector, and E the total energy. The inviscid and viscous fluxes are given, respectively, by:

$$\boldsymbol{f}^I = \begin{bmatrix} \rho \boldsymbol{V} \\ \rho \boldsymbol{V}\boldsymbol{V}^T + p\boldsymbol{I} \\ \rho \boldsymbol{V} H \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{f}^V = \begin{bmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau}\boldsymbol{V} - \kappa_T \nabla T \end{bmatrix}, \tag{3}$$

where p is the static pressure, $H = E + \frac{p}{\rho}$ is the total enthalpy, $\boldsymbol{\tau}$ the viscous stress tensor, κ_T is the thermal conductivity, $T = p/\rho R$ is the temperature, and R is the gas constant. The pressure is given by:

$$p = (\gamma - 1)\left(\rho E - \tfrac{1}{2}\rho \boldsymbol{V}^2\right), \tag{4}$$

where γ is the specific heat ratio. The viscous stress tensor, $\boldsymbol{\tau}$, is given by:

$$\boldsymbol{\tau} = \mu\left(\nabla \boldsymbol{V} + \nabla \boldsymbol{V}^T\right) - \lambda(\nabla \cdot \boldsymbol{V})\boldsymbol{I}, \tag{5}$$

where μ is the viscosity, and $\lambda = \frac{2}{3}\mu$ is the bulk viscosity.

Applying a change of variables $\boldsymbol{u} = \boldsymbol{u}(\mathbf{v})$, where $\mathbf{v}$ are the entropy variables:

$$\mathbf{v} = \begin{bmatrix} -\frac{s}{\gamma-1} + \frac{\gamma+1}{\gamma-1} - \frac{\rho E}{p} \\ \frac{\rho \boldsymbol{V}}{p} \\ -\frac{\rho}{p} \end{bmatrix}, \tag{6}$$

we rewrite the Navier-Stokes equations as:

$$\boldsymbol{A}_0 \mathbf{v}_{,t} + \bar{\boldsymbol{A}} \nabla \mathbf{v} - \nabla \cdot (\bar{\bar{\boldsymbol{K}}} \nabla \mathbf{v}) \quad = \quad 0, \tag{7}$$

with symmetric $\boldsymbol{A}_0 = \boldsymbol{u}_{,\mathbf{v}}$, $\bar{\boldsymbol{A}} = \boldsymbol{f}^I_{,\boldsymbol{u}} \boldsymbol{A}_0 = \boldsymbol{f}^I_{,\mathbf{v}}$ and $\bar{\bar{\boldsymbol{K}}} = \boldsymbol{f}^V_{,\nabla \boldsymbol{u}} \boldsymbol{A}_0 = \boldsymbol{f}^V_{,\nabla \mathbf{v}}$ [17].

We proceed to discretize (7) as follows. The domain, Ω, is partitioned into non-overlapping hexahedral elements, κ, while the time is partitioned into intervals (time-slabs), $I^n = [t^n, t^{n+1}]$. Define $\mathcal{V}_h = \left\{\boldsymbol{w}, \boldsymbol{w}|_\kappa \in [\mathcal{P}(\kappa \times I)]^5\right\}$, the

space-time finite-element space consisting of piece-wise polynomial functions in both space and time on each element. We seek a solution $\mathbf{v} \in \mathcal{V}_h$ such that the weak form:

$$\begin{aligned} \boldsymbol{r}(\mathbf{v}, \boldsymbol{w}) \quad = \quad & \sum_{\kappa} \Big\{ \int_{I^n} \int_{\kappa} - \Big(\boldsymbol{w}_{,t} \cdot \boldsymbol{u} + \nabla \boldsymbol{w} \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V) \Big) \\ & + \int_{I^n} \int_{\partial\kappa} \boldsymbol{w} \cdot (\widehat{\boldsymbol{f}^I \cdot \boldsymbol{n}} - \widehat{\boldsymbol{f}^V \cdot \boldsymbol{n}}) \\ & + \int_{\kappa} \boldsymbol{w}(t_-^{n+1}) \cdot \boldsymbol{u}(t_-^{n+1}) - \boldsymbol{w}(t_+^n) \cdot \boldsymbol{u}(t_-^n) \Big\} = 0, \end{aligned} \tag{8}$$

is satisfied for all $\boldsymbol{w} \in \mathcal{V}_h$. Here $\widehat{\boldsymbol{f}^I \cdot \boldsymbol{n}}$ and $\widehat{\boldsymbol{f}^V \cdot \boldsymbol{n}}$ denote numerical flux functions approximating the inviscid and viscous fluxes, respectively, while $\boldsymbol{n}$ is the outward pointing normal vector. In this work, the inviscid flux is discretized using the method of Ismail and Roe [18], while the viscous flux is discretized using the second method of Bassi and Rebay [2].

The space $\mathcal{V}_h$ is represented using a tensor-product basis such that on each element $\mathbf{v}$ is given by the product of Lagrange polynomials:

$$\mathbf{v}(\mathbf{x}(\xi), t(\tau)) = \mathbf{v}_{ijkl} \Phi_{ijkl} \qquad \text{with} \qquad \Phi_{ijkl} = \phi_i(\xi_1)\phi_j(\xi_2)\phi_k(\xi_3)\phi_l(\tau), \tag{9}$$

where $\mathbf{x}(\xi)$ defines a mapping from the reference cube, $\xi \subset [-1,1]^3$, to physical space, while $t(\tau)$ is the mapping from the reference interval $[-1,1]$ to the time interval $[t^n, t^{n+1}]$. ϕ_i are one-dimensional Lagrange basis functions defined at Gauss-Legendre (GL) points, while $\mathbf{v}_{ijkl}$ are the corresponding nodal values of the entropy variables. The integrals in (8) are evaluated using numerical quadrature. For example:

$$\begin{aligned} & \frac{2}{\Delta t} \int_{I^n} \int_{\kappa} - \Big(\boldsymbol{w}_{,t} \cdot \boldsymbol{u} + \nabla \boldsymbol{w} \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V)) \Big) \\ \simeq \quad & \Big\{ - \Big(\tau_{,t} \boldsymbol{w}_{,\tau} \cdot \boldsymbol{u} + \nabla_\xi \boldsymbol{w} \cdot (\tilde{\boldsymbol{f}}^I - \tilde{\boldsymbol{f}}^V)) \Big) |J| \Big\}_{\xi_p \xi_q \xi_r \tau_s} w_p w_q w_r w_s, \end{aligned} \tag{10}$$

where $\xi_p, \xi_q, \xi_r, \tau_s$ are one-dimensional GL quadrature points, and w_p, w_q, w_r and w_s are the associated quadrature weights. $|J|$ denotes the Jacobian of the mapping from element reference space to physical space, ∇_ξ denotes differentiation with respect to the reference coordinate ξ, while $\tilde{\boldsymbol{f}}^I = \xi_{,\mathbf{x}} \boldsymbol{f}^I$ and $\tilde{\boldsymbol{f}}^V = \xi_{,\mathbf{x}} \boldsymbol{f}^V$ are the fluxes mapped to the local element coordinate system. In this work we use a quadrature rule with twice as many quadrature points as nodal points in order to reduce the quadrature error (we ensure exact integration of cubic nonlinearities) thereby improving the nonlinear stability of our scheme [19, 12].

The remaining integrals appearing in (8) are evaluated in a similar manner, which may be described as a sequence of three steps:

1. Evaluate the state ($\mathbf{v}$) and gradient ($\nabla_\xi \mathbf{v}$) at the quadrature points.

2. Evaluate the fluxes ($\tilde{\boldsymbol{f}}^I$ and $\tilde{\boldsymbol{f}}^V$) at the quadrature points.
3. Multiply the fluxes with the basis functions ($\boldsymbol{w}$) or gradients ($\nabla_\xi \boldsymbol{w}$).

A key requirement for efficiency at high polynomial order is the evaluation of the first and third steps using the sum-factorization approach [12, 20], which allows the multiplication of the basis functions to be performed as a sequence of one-dimensional operations. This results in a residual evaluation cost which scales as $O(N^{d+1})$ for each space-time element where N is the solution order while d is the number of spatial-temporal dimensions (for unsteady 3D simulations $d = 4$). The cost per degree of freedom for the residual evaluation scales linearly with the solution order, however for moderate solution orders, $N = 4 - 16$, the increased operation count with solution order may be offset by the use of optimized numerical kernels [12].

3. Solution Strategy

Given the choice of basis functions and quadrature rule, Equation (8) represents a globally coupled system of nonlinear equations which need to be solved for each time-slab. For large three-dimensional simulations the cost of storing the linearization for a single step of an implicit scheme may be prohibitively expensive. Using a space-time formulation, this storage cost is further scaled by the order of the basis used in the temporal direction. In general, all degrees of freedom within an element are coupled, leading to a storage cost which scales as $O(N^d)$ per degree of freedom. We overcome the memory requirement limitation by using a matrix-free Newton-Krylov method.

A restarted GMRES method is used as Krylov solver, which does not require the explicit storage of the Jacobian matrix [21]. GMRES requires only the application of the linearization to each search direction, i.e. we need to compute the linearized residual in the search direction. In this work, the linearized residual is computed directly. As with the residual evaluation, the terms in the evaluation of the linearized residual in a search direction, $\boldsymbol{s}$, are computed as a sequence of three steps:

1. Evaluate the state ($\mathbf{v}$), gradient ($\nabla_\xi \mathbf{v}$), linearized state ($\boldsymbol{s}$) and linearized gradient ($\nabla_\xi \boldsymbol{s}$) at the quadrature points.
2. Evaluate the linearized fluxes ($\tilde{\boldsymbol{f}}^{\text{Lin}} = \frac{\partial \tilde{\boldsymbol{f}}}{\partial \mathbf{v}} \boldsymbol{s} + \frac{\partial \tilde{\boldsymbol{f}}}{\partial \nabla_\xi \mathbf{v}} \nabla_\xi \boldsymbol{s}$) at the quadrature points.
3. Multiply of the linearized fluxes with the basis functions ($\boldsymbol{w}$) or gradients ($\nabla_\xi \boldsymbol{w}$).

This approach is more expensive than the finite-difference approach often used in Jacobian-free method [22], however is insensitive to a step-size parameter, gives the exact linearization, and may be used to compute the solution of adjoint (dual) problems. In addition, the sum factorization approach is used in the application of the linearization, such that the cost of applying the linearization scales as $O(N)$ per degree of freedom. We note that with increasing solution

order this is more efficient than storing the linearization and computing the linearized residual as a matrix-vector product whose cost scales as $O(N^d)$ per DOF (even if there was no cost associated with forming the linearization).

We note that this approach also allows us to use different quadrature rules for computing the linearized and nonlinear residuals. In particular, we use a lower order or collocated quadrature rule in the evaluation of the linearized residuals of our Newton-Krylov scheme while driving to zero the nonlinear residual computed with more accurate quadrature. This reduced quadrature approach may be viewed as an inexact Newton scheme.

Preconditioning is necessary for stiff problems in order to obtain good performance in a Newton-Krylov scheme [22]. Elemental block-Jacobi, block-Gauss-Seidel and block-ILU preconditioners have been successfully used for the discontinuous Galerkin discretization of compressible flows at moderate solution orders $N = 2 - 5$ [4, 5]. However, the memory required for the storage of the factorization of elemental blocks scales as $O(N^d)$ per degree of freedom. This implies that at high order ($N > 5$) block preconditioners are prohibitively expensive. For example, we note that storing the elemental block Jacobian for a single 3D space-time element at $N = 16$ would require on the order of 1Tb of memory. Alternative DG formulations have been proposed in order to reduce the memory required to store the linearization (see ref [23]). However, factorization of the linearization introduces significant fill-in, implying that the memory required for a block preconditioner may still be prohibitively expensive for high polynomial orders.

In this work, we develop preconditioners with memory requirements no larger than that required for residual evaluations. In particular, we develop element-wise block-Jacobi preconditioners where the elemental blocks are solved approximately taking advantage of the tensor-product formulation of our finite-element scheme. Our preconditioners rely on the factorization of one-dimensional operators which require no more memory than the temporary arrays used in the evaluation of the residual on a single element. We present two different preconditioners: a diagonalized Alternating-Direction-Implicit (ADI) preconditioner and a preconditioner based on the Fast Diagonalization Method (FDM). In Section 4, we present simplified versions of our preconditioners for the case of a constant-coefficient linear advection-diffusion problem. In Section 5 we extend the preconditioners to the compressible Euler and Navier-Stokes equations on general curvilinear domains.

4. Preconditioning: Scalar Advection-Diffusion

Consider the constant-coefficient linear advection-diffusion equation:

$$v_{,t} + \boldsymbol{a} \cdot \nabla v - k\Delta v \quad = \quad f, \tag{11}$$

where v is the conserved scalar, $\boldsymbol{a}$ is a divergence free velocity field, k the diffusivity and f is a forcing term. For now consider only domains with rectangular parallelepiped elements for which the mapping, $\mathbf{x}(\xi)$, from reference to physical

space is a constant on each element. General curvilinear grids are considered in Section 5 along with the case of spatially varying coefficients.

Applying our space-time discontinuous-Galerkin discretization using an upwind-flux for the inviscid terms and the BR2 flux for the viscous terms we obtain an elemental block Jacobian which corresponds to the following operator:

$$\begin{aligned}\boldsymbol{r}(v|_\kappa, w|_\kappa) &= -\int_{I^n}\int_\kappa (w_{,t}v + \nabla w\cdot\boldsymbol{a}v - k\nabla w\cdot\nabla v) \\ &\quad + \int_{I^n}\int_{\partial\kappa} a_n^+ wv - \tfrac{1}{2}k(w\nabla_{\boldsymbol{n}}v + \nabla_{\boldsymbol{n}}wv - \eta_e wv) \\ &\quad + \int_\kappa w(t_-^{n+1})v(t_-^{n+1}),\end{aligned} \tag{12}$$

where $a_n = \boldsymbol{a}\cdot\boldsymbol{n}$, $a_n^\pm = \frac{1}{2}(a_n \pm |a_n|)$, $\nabla_{\boldsymbol{n}}$ denotes differentiation in the normal direction, while η_e is a lifting parameter from the BR2 scheme. We note that for tensor-product elements the BR2 scheme is equivalent to an interior-penalty scheme with a solution-order and geometry-dependent penalty parameter. It is convenient to write the elemental block Jacobian for a single element as:

$$\begin{aligned}&\frac{2}{\Delta t|J|}\boldsymbol{r}(v|_\kappa, w|_\kappa) = \\ &\int_\tau\int_{\xi_2}\int_{\xi_3}\left(-\int_{\xi_1}(\tilde{a}_1 w_{,\xi_1}v - \tilde{k}_{11}w_{,\xi_1}v_{,\xi_1}) + \left[\tilde{a}_1^+ wv - \tfrac{1}{2}\tilde{k}_{11}(wv_{,\xi_1} + w_{,\xi_1}v - \eta_e^+ wv)\right]_{\xi_1=-1}^{\xi_1=1}\right) \\ &+\int_\tau\int_{\xi_3}\int_{\xi_1}\left(-\int_{\xi_2}(\tilde{a}_2 w_{,\xi_2}v - \tilde{k}_{22}w_{,\xi_2}v_{,\xi_2}) + \left[\tilde{a}_2^+ wv - \tfrac{1}{2}\tilde{k}_{22}(wv_{,\xi_2} + w_{,\xi_2}v - \eta_e^+ wv)\right]_{\xi_2=-1}^{\xi_2=1}\right) \\ &+\int_\tau\int_{\xi_2}\int_{\xi_3}\left(-\int_{\xi_3}(\tilde{a}_3 w_{,\xi_3}v - \tilde{k}_{33}w_{,\xi_3}v_{,\xi_3}) + \left[\tilde{a}_3^+ wv - \tfrac{1}{2}\tilde{k}_{33}(wv_{,\xi_3} + w_{,\xi_3}v - \eta_e^+ wv)\right]_{\xi_3=-1}^{\xi_3=1}\right) \\ &+\int_{\xi_3}\int_{\xi_2}\int_{\xi_1}\left(-\int_\tau \tilde{a}_0 w_{,\tau}v + [\tilde{a}_0 wv]^{\tau=1}\right),\end{aligned} \tag{13}$$

where $\tilde{a}_i = \xi_{i,x_j}a_j$,

$$\tilde{a}_i^+ = \left\{ \begin{array}{ll} \frac{1}{2}(\tilde{a}_i + |\tilde{a}_i|) & \text{at} \quad \xi_i = 1 \\ \frac{1}{2}(\tilde{a}_i - |\tilde{a}_i|) & \text{at} \quad \xi_i = -1 \end{array} \right\}, \tag{14}$$

$\tilde{k}_{ii} = \xi_{i,x_j}\xi_{i,x_j}k$, while $\eta_e^+ = \eta_e \text{sign}(\xi_i)$. The index "0" corresponds to the temporal direction, with $\tilde{a}_0 = \frac{2}{\Delta t}$. We recognize that each line on the right-hand side of (13) corresponds to an advection-diffusion operator along an axis of our reference element. Employing a tensor-product basis, the matrix system for the elemental block Jacobian is rewritten as:

$$\begin{aligned}\frac{2}{\Delta t|J|}\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{V}}\right|_\kappa &= (D_1\otimes M_2\otimes M_3\otimes M_0) + (M_1\otimes D_2\otimes M_3\otimes M_0) \\ &\quad +(M_1\otimes M_2\otimes D_3\otimes M_0) + (M_1\otimes M_2\otimes M_3\otimes D_0),\end{aligned} \tag{15}$$

where M_i are one-dimensional mass matrices, whose entries are given by:

$$M_i \;=\; \int_{\xi_i} wv, \tag{16}$$

while D_i are one-dimensional advection diffusion operators:

$$D_i \;=\; -\int_{\xi_i} (\tilde{a}_i w_{,\xi_i} v - \tilde{k}_{ii} w_{,\xi_i} v_{,\xi_i}) + \left[\tilde{a}_i^+ wv - \tfrac{1}{2}\tilde{k}_{ii}(wv_{,\xi_i} + w_{,\xi_i} v - \eta_e^+ wv)\right]_{\xi_i=-1}^{\xi_i=1}, \tag{17}$$

Factoring out the elemental mass matrix gives:

$$\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{V}}\right|_\kappa \;=\; \frac{\Delta t|J|}{2}(M_1 \otimes M_2 \otimes M_3 \otimes M_0)\{(\tilde{D}_1 \otimes I \otimes I \otimes I) + (I \otimes \tilde{D}_2 \otimes I \otimes I) + (I \otimes I \otimes \tilde{D}_3 \otimes I) + (I \otimes I \otimes I \otimes \tilde{D}_0)\}, \tag{18}$$

where $\tilde{D}_i = M_i^{-1} D_i$, while I denotes the identity matrix. We seek to efficiently compute an inverse of $\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{V}}\right|_\kappa$. We note that the term $\frac{\Delta t|J|}{2}(M_1 \otimes M_2 \otimes M_3 \otimes M_0)$ appearing in (18) is simply the space-time elemental mass matrix, which is diagonal and thus easily inverted. It remains to invert the following matrix:

$$A \;=\; (\tilde{D}_1 \otimes I \otimes I \otimes I) + (I \otimes \tilde{D}_2 \otimes I \otimes I) + (I \otimes I \otimes \tilde{D}_3 \otimes I) + (I \otimes I \otimes I \otimes \tilde{D}_0). \tag{19}$$

We proceed to use the ADI or FDM schemes to invert A.

4.1. Alternating-Direction-Implicit (ADI)

Exact inversion of A requires the factorization and solution of an $N^d \times N^d$ matrix system. As noted previously, the cost of the factorization is prohibitively expensive with increasing solution order. The ADI scheme approximately inverts (19) by successively solving one-dimensional problems of size $N \times N$ in each coordinate direction. First, we modify the underlying PDE problem by adding a pseudo-temporal evolution. The ADI scheme may then be viewed as a single step applied to marching the modified PDE to steady state in pseudo-time. The pseudo-time-step is denoted by τ. The modified elemental Jacobian is:

$$\tau\hat{A} \;=\; (I \otimes I \otimes I \otimes I) + (\tau\tilde{D}_1 \otimes I \otimes I \otimes I) + (I \otimes \tau\tilde{D}_2 \otimes I \otimes I) + (I \otimes I \otimes \tau\tilde{D}_3 \otimes I) + (I \otimes I \otimes I \otimes \tau\tilde{D}_0). \tag{20}$$

This modified elemental Jacobian is inverted approximately by:

$$\tfrac{1}{\tau}\hat{A}^{-1} \approx \tfrac{1}{\tau}\tilde{A}^{-1} \;=\; (I \otimes I \otimes I \otimes \tau\tilde{D}_0 + I)^{-1}(I \otimes I \otimes \tau\tilde{D}_3 + I \otimes I)^{-1} (I \otimes \tau\tilde{D}_2 + I \otimes I \otimes I)^{-1}(\tau\tilde{D}_1 + I \otimes I \otimes I \otimes I)^{-1}, \tag{21}$$

where the errors introduced as a result of the approximate inversion are second order in pseudo-time [10]. We note the matrix inversions in (21) correspond to the solution of one-dimensional advection-diffusion problems in each coordinate direction. In particular, each inversion corresponds to N^{d-1} solves of systems of size $N \times N$. The dominant cost of applying (21), is the factorization of the one-dimensional systems, which must be performed dN^{d-1} times. In our implementation we choose not to store any factorizations thus these must be recomputed each application of the preconditioner. This gives a total cost of a single application of the ADI preconditioner which scales as $O(dN^{d+2})$. In the constant-coefficient case, the factorization needs to be performed only once in each coordinate direction, such that the cost is dominated by the forward- and back-substitution resulting in a cost which scales as $O(dN^{d+1})$. Thus, this "single-factorization" approach gives a factor of N reduction in the cost of applying the ADI preconditioner, resulting in a cost which is, to leading order, the same as a residual evaluation. Additionally, we note that the single-factorization inversion may be performed using the optimized matrix-matrix multiplication routines developed for the residual evaluation.

The choice of the pseudo-time step can have a significant impact on the convergence rate of the ADI preconditioner. In this work we choose a pseudo-time step based on maintaining a pseudo-time CFL number on the order of 1. In the advective limit, the characteristic time-scale along each axis of the reference element is $\frac{1}{|\tilde{a}_i|N}$, while in the diffusive limit the characteristic time is $\frac{1}{\tilde{k}_{ii}N^3}$. We set the pseudo-time step such that:

$$\frac{1}{\tau^2} = c^2\sqrt{1-\frac{\max(c_i^2)}{c^2}}, \tag{22}$$

where $c_i = \max(|\tilde{a}_i|N, \tilde{k}_{ii}N^3)$, while $c = \sqrt{c_1^2+c_2^2+c_3^2+c_0^2}$. The pseudo-time step scales with the characteristic time $1/c$ as well as a factor which takes into account the alignment of the flow with the axes of the reference element. In particular, when the flow is aligned with a particular coordinate direction, the pseudo-time goes to infinity and the ADI-scheme results in an exact block inversion.

4.2. Fast Diagonalization Method (FDM)

The fast diagonalization method may be used to efficiently compute exact inverses of certain separable tensor-product matrices by taking advantage of the eigenvector factorization of the one-dimensional operators. The system (19) may be rewritten as:

$$\begin{aligned} A = {} & (X_1\otimes X_2\otimes X_3\otimes X_0)\{(\Lambda_1\otimes I\otimes I\otimes I)+(I\otimes\Lambda_2\otimes I\otimes I) \\ & +(I\otimes I\otimes\Lambda_3\otimes I)+(I\otimes I\otimes I\otimes\Lambda_0)\}(X_1\otimes X_2\otimes X_3\otimes X_0)^{-1}, \end{aligned} \tag{23}$$

provided the one-dimensional advection-diffusion operators $\tilde{D}_i$ are diagonalizable. The factorization is $\tilde{D}_i = X_i\Lambda_i X_i^{-1}$, where Λ_i is the diagonal matrix of eigenvalues, while X_i is the matrix whose columns are the eigenvectors of $\tilde{D}_i$.

Using the eigenvector factorization A is easily inverted as:

$$\begin{aligned} A^{-1} &= (X_1 \otimes X_2 \otimes X_3 \otimes X_0)\{(\Lambda_1 \otimes I \otimes I \otimes I) + (I \otimes \Lambda_2 \otimes I \otimes I) \\ &\quad +(I \otimes I \otimes \Lambda_3 \otimes I) + (I \otimes I \otimes I \otimes \Lambda_0)\}^{-1}(X_1 \otimes X_2 \otimes X_3 \otimes X_0)^{-1}. \end{aligned} \tag{24}$$

The first and third matrices corresponding to the tensor-product of the eigenvectors matrices can be efficiently applied using the sum-factorization approach. The second matrix in (24), corresponds to a diagonal matrix with the eigenvalues of A, and hence easily invertible. The dominant cost of applying the FDM scheme is due to the sum-factorization approach used to multiply the eigenvector matrices, leading to a dominant cost of $O(dN^{d+1})$. Thus the cost of applying the FDM preconditioner is, to leading order, the same as a single residual evaluation. We note that this is also the same cost as the single-factorization ADI approach, however, in this case the elemental block inversion is exact.

In the development of the FDM preconditioner we have assumed that the one-dimensional scalar advection-diffusion operators, $\tilde{D}_i$, have a full set of linearly independent eigenvectors. In the diffusion-dominated limit, the one-dimensional operators are symmetric positive-definite and thus guaranteed to have an orthogonal set of eigenvectors. Unfortunately, in the advection-dominated limit, no such guarantee exists. In particular, in the continuous limit, an advection operator is not diagonalizable and has only a single eigenvalue with geometric multiplicity of one. In the discrete case, the scalar advection operator is in general diagonalizable, however with increasing solution order, the eigenvectors become aligned, leading to exponential growth of the condition number of the eigenvector matrices. Figure 1 plots the condition number of the eigenvector matrix for the one-dimensional scalar advection operator as a function of solution order. The exponential growth of the condition number with solution order implies that the basic FDM scheme cannot be used beyond $N = 10$. In order to reduce the conditioning issue, an artificial viscosity, $\hat{\tilde{k}}_{ii} = \tilde{a}_i \epsilon / N^2$ is added to the one-dimensional advection-diffusion operators used in the preconditioning matrix. The use of these modified operators implies that the FDM preconditioner does not exactly invert the elemental block-Jacobian, however it still remains effective as a preconditioner. We use an artifical viscosity with a value of $\epsilon = 10^{-2}$, which is sufficient to bound the condition number of the eigenvector matrices while not significantly modifying the elemental block-Jacobian. Figure 1 demonstrates that using this value of ϵ the modified operators with artificial viscosity have bounded condition number with increasing solution order.

4.3. Numerical Results

We now present numerical results using the ADI and FDM preconditioner for the solution of both steady and unsteady scalar advection-diffusion problems. We solve (11) on the domain $\Omega = [0, 1]^3$ with uniform advection velocity $a = [\frac{3}{4}, \frac{1}{2}, \frac{\sqrt{3}}{4}]^T$ and forcing f chosen such that the exact solution has the form:

$$v = \alpha_0 + \sin(\alpha_t + \beta_t t)\sin(\alpha_x + \beta_x x)\sin(\alpha_y + \beta_y y)\sin(\alpha_z + \beta_z z), \tag{25}$$

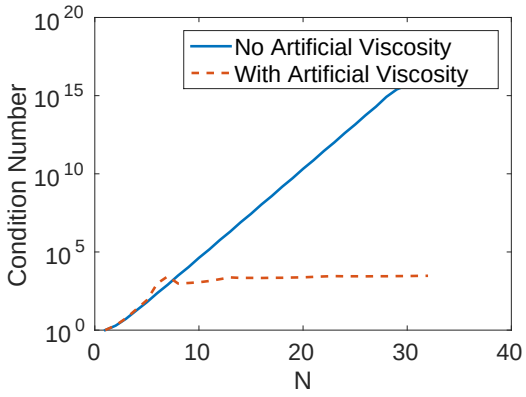

Figure 1: Condition number of eigenvector matrices for one-dimensional advection-operators with and without artificial viscosity.

with constants $\alpha_0 = 1$, $\alpha_t = 0.25$, $\alpha_x = 0.15$, $\alpha_y = 0.47$, $\alpha_z = 0.65$, $\beta_t = 2.5$, $\beta_x = 7.6$, $\beta_y = 3.4$ and $\beta_z = 2.3$. For steady state problems β_t is set to zero. The character of the system is dictated by the Péclet number, $Pe = \frac{|a|}{k}$, where $Pe >> 1$ is advection dominated, while $Pe << 1$ is diffusion dominated. The results in this section are reported in terms of number of residual evaluations required to converge the residual to a tolerance of 10^{-12}. The linear solver is a preconditioned restarted GMRES algorithm using 20 Krylov vectors. We compare the ADI and FDM preconditioners presented in the previous section along with a simple mass-matrix preconditioner.

4.3.1. Steady-State $Pe = 10^3$

We first consider the solution of a steady-steady state advection diffusion problem $Pe = 1000$. We solve the steady-state advection problem using polynomial orders $N = 2, 4, 8, 16$, on meshes with varying degress of freedom, $1/h$, in each direction. Table 1 reports the number of iterations required for convergence using mass-matrix, ADI and FDM preconditioners.

In general, the mass-matrix preconditioner performs the worst with the tensor-product preconditioners providing greater improvement with increasing solution order and mesh size. In particular, the mass-matrix preconditioning is insufficient to ensure convergence of the Krylov scheme for large solution order or using large numbers of degrees of freedom (denoted by "*" in Table 1). When GMRES stalls, using a larger Krylov subspace (i.e. increasing the GMRES restart value to 50 or 100) typically allows for an additional order of magnitude reduction in the residual, though GMRES still stalls.

Using the ADI preconditioner allows for convergence of the Krylov scheme for larger solution orders and more degrees of freedom, though the algorithm still stalls at 16th order for large numbers of degrees of freedom. W

The FDM preconditioner is sufficiently strong to allow for convergence of the Krylov method at all solution orders and degrees of freedom considered.

In order to better understand the performance of the different preconditioners for this test case we consider two additional steady-state problems at $Pe = \infty$ and $Pe = 1$.

4.3.2. Steady-State $Pe = \infty$

In this case we focus on the performance of the preconditioners in the advection-dominated limit. As shown in Table 2, when using the mass-matrix as the preconditioner, the number of iterations increases with both solution order and number of degrees of freedom. With large number of degrees of freedom, the mass-matrix preconditioner is not sufficiently effective, and the restarted GMRES algorithm stalls before reaching the desired convergence tolerance.

Using the ADI preconditioner, the number of iterations increases linearly with number of degrees of freedom. For this convection-dominated problem the number of iterations scales roughly with the number of degrees of freedom through which a characteristic passes. The ADI preconditioner ensures that for a given number of degrees of freedom, the number of iterations does not grow with solution order, N, (in fact the number of iterations may even decrease). An explicit scheme, on the other hand, would impose a stability restriction requiring the number of iterations to scale linearly with solution order. Thus, the implicit scheme with ADI preconditioner can effectively overcome the stiffness associated with high solution order.

The FDM preconditioner converges in the fewest number of iterations for all combinations of solution order and number of degrees of freedom presented. At a fixed solution order, N, the number of iterations scales linearly with the number of degrees of freedom. In this advection-dominated limit, the FDM preconditioner converges in a number of iterations proportional to the number of elements through which a characteristic passes (as seen along the diagonals of Table 2). Thus for a fixed number of degrees of freedom, the number of iterations required for convergence using the FDM preconditioner scales inversely with solution order, N.

4.3.3. Steady-State $Pe = 1$

Next we consider $Pe = 1$, where diffusive effects become important. Table 3 presents the number of iterations required for convergence using the different preconditioners considered. As with the advection-dominated case, the mass-matrix preconditioner performs worst. The stiffness of the problem increases both with degrees of freedom and solution order. The mass matrix preconditioning is insufficient to enable convergence prior to the stalling of the GMRES algorithm at $N = 16$ even for a problem with a single element.

The ADI preconditioner gives improved performance relative to the mass matrix preconditioner, however the number of iterations required to converge grows superlinearly with the number of degrees of freedom at a given solution order. Furthermore, using a fixed number of degrees of freedom the number of iterations grows with increasing solution order.

(a) Mass-Matrix

	N			
$1/h$	2	4	8	16
2	20	-	-	-
4	43	54	-	-
8	158	199	313	-
16	283	334	769	1597
32	439	604	*	*
64	712	*	*	*
128	*	*	*	*

(b) ADI

	N			
$1/h$	2	4	8	16
2	11	-	-	-
4	25	22	-	-
8	56	60	60	-
16	139	134	389	303
32	243	277	889	*
64	374	475	1084	*
128	653	722	*	*

(c) FDM

	N			
$1/h$	2	4	8	16
2	6	-	-	-
4	19	6	-	-
8	42	34	10	-
16	93	65	74	16
32	238	174	133	82
64	377	288	245	190
128	644	508	384	316

Table 1: Number of residual evaluations for steady state advection-diffusion problem with $Pe = 10^3$ using mass-matrix, ADI and FDM preconditioners. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denote failure of convergence of restarted GMRES algorithm.)

(a) Mass-Matrix

	N			
$1/h$	2	4	8	16
2	17	-	-	-
4	43	53	-	-
8	139	177	194	-
16	304	365	430	404
32	413	563	693	798
64	669	*	*	*
128	*	*	*	*

(b) ADI

	N			
$1/h$	2	4	8	16
2	12	-	-	-
4	27	24	-	-
8	53	49	46	-
16	126	130	114	95
32	237	218	249	235
64	328	317	312	316
128	479	500	472	443

(c) FDM

	N			
$1/h$	2	4	8	16
2	6	-	-	-
4	13	6	-	-
8	35	13	6	-
16	72	30	12	6
32	229	75	30	12
64	280	196	74	30
128	400	298	193	74

Table 2: Number of residual evaluations for steady state advection-diffusion problem with $Pe = \infty$ using mass-matrix, ADI and FDM preconditioners. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denote failure of convergence of restarted GMRES algorithm.)

Once again, the FDM preconditioner performs the best of the three options considered. In the single element case, the FDM preconditioner converges in a small number of iterations independent of the solution order as given by first the diagonal of Table 3. In the general multi-element case, the number of iterations grows almost quadratically with the number of degrees of freedom. However, at a fixed number of degrees of freedom, the number of iterations remains roughly constant with increasing solution order, N. Thus the FDM preconditioner is able to overcome the stiffness associated with the higher-order solution.

In the diffusion dominated case, a coarse correction or multigrid procedure may be used to obtain convergence rates which are independent of the number of degrees of freedom. In this work we are primarily interested in advection-dominated problems, thus we have not considered the use of a coarse grid correction.

(a) Mass-Matrix

	N			
$1/h$	2	4	8	16
2	13	-	-	-
4	47	96	-	-
8	252	474	718	-
16	774	1756	4232	*
32	*	*	*	*

(b) ADI

	N			
$1/h$	2	4	8	16
2	9	-	-	-
4	46	39	-	-
8	141	187	123	-
16	378	494	707	677
32	998	1433	*	*

(c) FDM

	N			
$1/h$	2	4	8	16
2	8	-	-	-
4	34	17	-	-
8	125	169	18	-
16	374	357	360	18
32	1076	1097	1082	1267

Table 3: Number of residual evaluations for steady state advection-diffusion problem with $Pe = 1$ using mass-matrix, ADI and FDM preconditioners. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denote failure of convergence of restarted GMRES algorithm.)

4.3.4. Unsteady $Pe = 10^3$

Finally, we consider unsteady advection diffusion problem solved using our space-time DG scheme. For this unsteady problem we consider performance as a function of solution order, number of degrees of freedom as well as the CFL number, where $CFL = \frac{|a|\Delta t}{hN}$. We report convergence in terms of the number of residual evaluations required to converge a single time slab in Tables 4 - 6.

In general, the number of iterations required to converge a single time slab grows with increasing CFL number. At very small CFL the number of iterations is independent of the number of degrees of freedom, and grows only slightly with solution order. With increasing CFL the performance begins to resemble the steady case, with the FDM preconditioner performing better than mass-matrix or ADI preconditioning. At very large CFL and high solution orders only the FDM preconditioner is able to converge all cases.

The cost for performing a simulation for a fixed period of time may be estimated by the cost for a single time slab divided by the CFL number. Thus, performing a simulation at large CFL number is generally more efficient. In particular, the time step will be dictated by the physics which needs to be resolved as opposed to a stability restriction.

(a) Mass-Matrix, $N = 2$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	10	18	33	100	234	276	303
32	10	18	33	118	319	411	458
64	10	18	34	126	479	699	708
128	12	19	37	135	818	*	*

(b) Mass-Matrix, $N = 4$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	21	33	59	187	372	389	364
32	21	35	61	235	553	591	672
64	25	36	68	276	896	*	*
128	23	40	77	318	1680	*	*

(c) Mass-Matrix, $N = 8$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	49	67	114	399	776	799	802
32	48	71	131	578	*	*	*
64	47	75	166	705	*	*	*
128	52	85	195	897	*	*	*

(d) Mass-Matrix, $N = 16$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	123	224	247	773	1679	1665	1641
32	220	242	357	1395	*	*	*
64	165	265	442	*	*	*	*
128	184	235	512	*	*	*	*

Table 4: Number of residual evaluations for unsteady advection-diffusion problem with $Pe = 10^3$ using mass-matrix preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denote failure of convergence of restarted GMRES algorithm.)

(a) ADI, $N = 2$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	11	14	22	43	113	147	149
32	11	14	22	47	172	250	273
64	11	14	23	53	220	399	425
128	10	13	26	66	339	610	662

(b) ADI, $N = 4$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	13	19	35	66	153	148	138
32	13	20	40	92	244	255	265
64	14	22	50	138	359	494	497
128	15	25	59	154	488	724	768

(c) ADI, $N = 8$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	16	27	74	231	436	366	405
32	18	33	115	529	1028	963	944
64	20	40	149	479	1064	1088	1130
128	22	51	201	478	1057	1337	*

(d) ADI, $N = 16$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	21	50	239	695	477	321	341
32	28	85	556	*	*	*	*
64	34	122	763	1964	*	*	*
128	43	189	980	*	*	*	*

Table 5: Number of residual evaluations for unsteady advection-diffusion problem with $Pe = 10^3$ using ADI preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denote failure of convergence of restarted GMRES algorithm.)

(a) FDM, $N = 2$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	9	10	14	31	90	104	96
32	9	10	16	36	139	217	218
64	9	11	18	43	209	364	359
128	9	11	21	54	334	597	674

(b) FDM, $N = 4$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	10	12	21	41	73	71	70
32	10	12	24	53	151	196	197
64	10	13	30	67	189	287	289
128	10	15	39	77	282	517	523

(c) FDM, $N = 8$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	11	15	35	67	78	77	77
32	11	18	47	83	136	124	125
64	12	21	61	94	219	225	228
128	13	27	77	110	270	401	403

(d) FDM, $N = 16$

	CFL						
$1/h$	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
16	10	13	16	18	19	19	19
32	15	36	104	96	84	84	83
64	18	48	121	121	224	211	200
128	23	69	132	184	311	326	380

Table 6: Number of residual evaluations for unsteady advection-diffusion problem with $Pe = 10^3$ using FDM preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denote failure of convergence of restarted GMRES algorithm.)

5. Preconditioning: Compressible Navier-Stokes

In this section, we extend the ADI and FDM preconditioners to the compressible Navier-Stokes equations. As in Section 4 we initially consider a constant-coefficient problem, then discuss modifications to the algorithm in the general varying coefficient/curvilinear geometry case. In particular, we consider the linearized compressible Euler equations given by:

$$\boldsymbol{A}_0 \mathbf{v}_{,t} + \bar{\boldsymbol{A}} \nabla \mathbf{v} \quad = \quad 0. \tag{26}$$

Applying the discontinuous-Galerkin discretization, using the Roe flux [24] we obtain an elemental block Jacobian which corresponds to the following operator:

$$\begin{aligned} \boldsymbol{r}(\mathbf{v}|_\kappa, \boldsymbol{w}|_\kappa) \quad = \quad & -\int_{I^n}\int_\kappa (\boldsymbol{w}_{,t}\boldsymbol{A}_0\mathbf{v} + \nabla \boldsymbol{w} \cdot \bar{\boldsymbol{A}}\mathbf{v}) \\ & + \int_{I^n}\int_{\partial\kappa} \boldsymbol{w}\boldsymbol{A}_n^+\mathbf{v} \\ & + \int_\kappa \boldsymbol{w}(t_-^{n+1})\boldsymbol{A}_0\mathbf{v}(t_-^{n+1}). \end{aligned} \tag{27}$$

As in the scalar case, it is convenient to write this in the following form:

$$\begin{aligned} \frac{2}{\Delta t|J|}\boldsymbol{r}(\mathbf{v}|_\kappa, \boldsymbol{w}|_\kappa) \quad = \quad & + \int_\tau\int_{\xi_3}\int_{\xi_2}\left(-\int_{\xi_1} \boldsymbol{w}_{,\xi_1}\tilde{\boldsymbol{A}}_1\mathbf{v} + \left[\boldsymbol{w}\tilde{\boldsymbol{A}}_1^+\mathbf{v}\right]_{\xi_1=-1}^{\xi_1=1}\right) \\ & + \int_\tau\int_{\xi_3}\int_{\xi_1}\left(-\int_{\xi_2} \boldsymbol{w}_{,\xi_2}\tilde{\boldsymbol{A}}_2\mathbf{v} + \left[\boldsymbol{w}\tilde{\boldsymbol{A}}_2^+\mathbf{v}\right]_{\xi_2=-1}^{\xi_2=1}\right) \\ & + \int_\tau\int_{\xi_2}\int_{\xi_1}\left(-\int_{\xi_3} \boldsymbol{w}_{,\xi_3}\tilde{\boldsymbol{A}}_3\mathbf{v} + \left[\boldsymbol{w}\tilde{\boldsymbol{A}}_3^+\mathbf{v}\right]_{\xi_3=-1}^{\xi_3=1}\right) \\ & + \int_{\xi_3}\int_{\xi_2}\int_{\xi_1}\left(-\int_\tau \boldsymbol{w}_{,\tau}\tilde{\boldsymbol{A}}_0\mathbf{v} + \left[\boldsymbol{w}\tilde{\boldsymbol{A}}_0\mathbf{v}\right]^{\tau=1}\right), \end{aligned} \tag{28}$$

where $\tilde{\boldsymbol{A}}_i = \xi_{i,x_j}\boldsymbol{A}_j$,

$$\tilde{\boldsymbol{A}}_i^+ = \left\{ \begin{array}{lll} \frac{1}{2}(\tilde{\boldsymbol{A}}_i + |\tilde{\boldsymbol{A}}_i|) & \text{at} & \xi_i = 1 \\ \frac{1}{2}(\tilde{\boldsymbol{A}}_i - |\tilde{\boldsymbol{A}}_i|) & \text{at} & \xi_i = -1 \end{array} \right\}, \tag{29}$$

while $\tilde{\boldsymbol{A}}_0 = \frac{2}{\Delta t}\boldsymbol{A}_0$. We recognize that each line on the right-hand side of (28) corresponds to an one-dimensional hyperbolic system along an axis of our reference element. Unlike the scalar case we cannot write the discrete system corresponding to (28) as a separable tensor-product matrix. However we may write:

$$\begin{aligned} \frac{2}{\Delta t}\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{V}} \quad = \quad & (D_1^+ \otimes M_2 \otimes M_3 \otimes M_0 \otimes \tilde{\boldsymbol{A}}_1^+) + (D_1^- \otimes M_2 \otimes M_3 \otimes M_0 \otimes \tilde{\boldsymbol{A}}_1^-) \\ & (M_1 \otimes D_2^+ \otimes M_3 \otimes M_0 \otimes \tilde{\boldsymbol{A}}_2^+) + (M_1 \otimes D_2^- \otimes M_3 \otimes M_0 \otimes \tilde{\boldsymbol{A}}_2^-) \\ & (M_1 \otimes M_2 \otimes D_3^+ \otimes M_0 \otimes \tilde{\boldsymbol{A}}_3^+) + (M_1 \otimes M_2 \otimes D_3^- \otimes M_0 \otimes \tilde{\boldsymbol{A}}_3^-) \\ & (M_1 \otimes M_2 \otimes M_3 \otimes D_0^+ \otimes \tilde{\boldsymbol{A}}_0), \end{aligned} \tag{30}$$

where $D_i^{\pm}$ are one-dimensional advection operators with advective velocity ± 1. As in the scalar case we may factor out the space-time mass matrix to get a system of the form:

$$\begin{aligned} A &= (\tilde{D}_1^+ \otimes I \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_1^+) + (\tilde{D}_1^- \otimes I \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_1^-) \\ &\quad (I \otimes \tilde{D}_2^+ \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_2^+) + (I \otimes \tilde{D}_2^- \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_2^-) \\ &\quad (I \otimes I \otimes \tilde{D}_3^+ \otimes I \otimes \tilde{\boldsymbol{A}}_3^+) + (I \otimes I \otimes \tilde{D}_3^- \otimes I \otimes \tilde{\boldsymbol{A}}_3^-) \\ &\quad (I \otimes I \otimes I \otimes \tilde{D}_0^+ \otimes \tilde{\boldsymbol{A}}_0). \end{aligned} \tag{31}$$

Exact inversion of A requires the factorization and solution of a $5N^d \times 5N^d$ matrix system. The cost of the factorization is prohibitively expensive with increasing solution order. We now proceed to develop the diagonalized-ADI and FDM-based scheme for approximately inverting A.

5.1. Diagonalized ADI

As in the scalar case, the development of an ADI scheme relies on modifying the underlying PDE scheme by introducing a pseudo-time evolution. An ADI scheme may be viewed as approximating the original system marching to steady-state in pseudo-time.

$$\begin{aligned} \tau\tilde{A} &= \Big\{(I \otimes I \otimes I \otimes I \otimes I) + (I \otimes I \otimes I \otimes \tau\tilde{D}^+ \otimes \tilde{\boldsymbol{A}}_0 \boldsymbol{A}_0^{-1}) \\ &\quad +(I \otimes I \otimes \tau\tilde{D}^+ \otimes I \otimes \tilde{\boldsymbol{A}}_3^+ \boldsymbol{A}_0^{-1}) + (I \otimes I \otimes \tau\tilde{D}^- \otimes I \otimes \tilde{\boldsymbol{A}}_3^- \boldsymbol{A}_0^{-1}) \\ &\quad +(I \otimes \tau\tilde{D}^+ \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_2^+ \boldsymbol{A}_0^{-1}) + (I \otimes \tau\tilde{D}^- \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_2^- \boldsymbol{A}_0^{-1}) \\ &\quad +(\tau\tilde{D}^+ \otimes I \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_1^+ \boldsymbol{A}_0^{-1}) + (\tau\tilde{D}^- \otimes I \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_1^- \boldsymbol{A}_0^{-1})\Big\} \\ &\quad \times (I \otimes I \otimes I \otimes I \otimes \boldsymbol{A}_0) \end{aligned} \tag{32}$$

Following the scalar case, we may write the approximate inverse of (32) as:

$$\begin{aligned} \tfrac{1}{\tau}\tilde{A}^{-1} &= (I \otimes I \otimes I \otimes I \otimes \boldsymbol{A}_0^{-1}) \times \\ &\quad \{(I \otimes I \otimes I \otimes I \otimes I) + (I \otimes I \otimes I \otimes \tau\tilde{D}^+ \otimes \tilde{\boldsymbol{A}}_0 \boldsymbol{A}_0^{-1})\}^{-1} \\ &\quad \{(I \otimes I \otimes I \otimes I \otimes I) + (I \otimes I \otimes \tau\tilde{D}^+ \otimes I \otimes \tilde{\boldsymbol{A}}_3^+ \boldsymbol{A}_0^{-1}) + (I \otimes I \otimes \tau\tilde{D}^- \otimes I \otimes \tilde{\boldsymbol{A}}_3^- \boldsymbol{A}_0^{-1})\}^{-1} \\ &\quad \{(I \otimes I \otimes I \otimes I \otimes I) + (I \otimes \tau\tilde{D}^+ \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_2^+ \boldsymbol{A}_0^{-1}) + (I \otimes \tau\tilde{D}^- \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_2^- \boldsymbol{A}_0^{-1})\}^{-1} \\ &\quad \{(I \otimes I \otimes I \otimes I \otimes I) + (\tau\tilde{D}^+ \otimes I \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_1^+ \boldsymbol{A}_0^{-1}) + (\tau\tilde{D}^- \otimes I \otimes I \otimes I \otimes \tilde{\boldsymbol{A}}_1^- \boldsymbol{A}_0^{-1})\}^{-1} \end{aligned} \tag{33}$$

Each inversion in (33) corresponds to solving advective systems along the principle axis of the reference element. We highlight this fact by writing (33), using

an abuse of notation, as:

$$\begin{aligned}\tfrac{1}{\tau}\tilde{A}^{-1} &= (I\otimes I\otimes I\otimes I\otimes \boldsymbol{A}_0^{-1})\times \\ &\quad (I\otimes I\otimes I\otimes \tau\tilde{D}_0^*\otimes \tilde{\boldsymbol{A}}_0\boldsymbol{A}_0^{-1})^{-1} \\ &\quad (I\otimes I\otimes \tau\tilde{D}_3^*\otimes I\otimes \tilde{\boldsymbol{A}}_3\boldsymbol{A}_0^{-1})^{-1} \\ &\quad (I\otimes \tau\tilde{D}_2^*\otimes I\otimes I\otimes \tilde{\boldsymbol{A}}_2\boldsymbol{A}_0^{-1})^{-1} \\ &\quad (\tau\tilde{D}_1^*\otimes I\otimes I\otimes I\otimes \tilde{\boldsymbol{A}}_1\boldsymbol{A}_0^{-1})^{-1}, \end{aligned} \tag{34}$$

where $\tilde{D}_i^*$ are fictitious operators such that (33) and (34) are equivalent. With this notation it may be easily seen that each inversion in (34) corresponds to solving N^{d-1} systems of size $5N\times 5N$. As in the scalar case, the dominant cost is the factorization of the one-dimensional systems such that the cost, to leading order, is $O(5^3 dN^{d+2})$.

Following Pulliam and Chaussee [11], a diagonalized form of the ADI scheme may be obtained using an eigenvalue decomposition of the flux Jacobians, $\tilde{\boldsymbol{A}}_i = \boldsymbol{R}_i\tilde{\boldsymbol{\Lambda}}_i\boldsymbol{R}_i^T$. Here, R_i are the eigenvectors, while $\tilde{\boldsymbol{\Lambda}}_i$ is the matrix of eigenvalues of $\boldsymbol{A}_i$. We also note that $\boldsymbol{A}_0 = \boldsymbol{R}_1\boldsymbol{R}_1^T = \boldsymbol{R}_2\boldsymbol{R}_2^T = \boldsymbol{R}_3\boldsymbol{R}_3^T$ [6]. We may write (34) as:

$$\begin{aligned}\tfrac{1}{\tau}\tilde{A}^{-1} &= (I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-T}\boldsymbol{R}_3^{-1})\times \\ &\quad (I\otimes I\otimes I\otimes \tau\tilde{D}_0^*\otimes \tilde{\boldsymbol{\Lambda}}_0)^{-1} \\ &\quad (I\otimes I\otimes \tau\tilde{D}_3^*\otimes I\otimes R_3\tilde{\boldsymbol{\Lambda}}_3\boldsymbol{R}_3^{-1})^{-1} \\ &\quad (I\otimes \tau\tilde{D}_2^*\otimes I\otimes I\otimes R_2\tilde{\boldsymbol{\Lambda}}_2\boldsymbol{R}_2^{-1})^{-1} \\ &\quad (\tau\tilde{D}_1^*\otimes I\otimes I\otimes I\otimes R_1\tilde{\boldsymbol{\Lambda}}_1\boldsymbol{R}_1^{-1})^{-1} \\ &= (I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-T})(I\otimes I\otimes I\otimes \tau\tilde{D}_0^*\otimes \tilde{\boldsymbol{\Lambda}}_0)^{-1} \\ &\quad (I\otimes I\otimes \tau\tilde{D}_3^*\otimes I\otimes \tilde{\boldsymbol{\Lambda}}_3)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-1}\boldsymbol{R}_2) \\ &\quad (I\otimes \tau\tilde{D}_2^*\otimes I\otimes I\otimes \tilde{\boldsymbol{\Lambda}}_2)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_2^{-1}\boldsymbol{R}_1) \\ &\quad (\tau\tilde{D}_1^*\otimes I\otimes I\otimes I\otimes \tilde{\boldsymbol{\Lambda}}_1)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_1^{-1}). \end{aligned} \tag{35}$$

We point out that $\tilde{D}_i^*\otimes\tilde{\boldsymbol{\Lambda}}_i$ corresponds to a block-diagonal matrix where each block corresponds to a one-dimensional scalar advection operator with advection velocity given by an eigenvalue $\tilde{\lambda}_i$. In this form the inversions in (35) correspond to solutions of independent 5 N^{d-1} scalar advection problems in each coordinate direction. The application of $\boldsymbol{R}_3^{-T}$, $\boldsymbol{R}_3^{-1}\boldsymbol{R}_2$, $\boldsymbol{R}_2^{-1}\boldsymbol{R}_1$, and $\boldsymbol{R}_1^{-1}$ correspond to transformations to and from characteristic variables, which are performed locally at each nodal point. Additionally, we note that $\boldsymbol{R}_3^{-1}\boldsymbol{R}_2$, $\boldsymbol{R}_2^{-1}\boldsymbol{R}_1$ depend only upon local geometry information [11].

We now discuss the extension of the diagonalized-ADI scheme to the case of variable coefficient compressible Navier-Stokes equations on curvilinear grids. Extension to the Navier-Stokes equations is achieved by adding a spectral radius approximation for the viscous terms. The one-dimensional scalar advection systems are replaced with advection-diffusion systems with viscosity $\tilde{\nu}_{ii} = \nu\xi_{i,x_j}\xi_{i,x_j}$. In this case of variable coefficients or curvilinear geometry, the

elemental block Jacobian may not be written as a sum of tensor-product matrices. However, we may recognize the diagonalized-ADI scheme as a sequence of steps, and apply these steps in the general case. The steps involved in the diagonalized-ADI scheme are given by:

1. Multiply by the inverse of the space-time element mass matrix
2. Transform to characteristic variables in the ξ_1-direction
3. Solve one-dimensional scalar systems along lines in the ξ_1-direction
4. Transform to characteristic variables in the ξ_2-direction
5. Solve one-dimensional scalar systems along lines in the ξ_2-direction
6. Transform to characteristic variables in the ξ_3-direction
7. Solve one-dimensional scalar systems along lines in the ξ_3-direction
8. Solve one-dimensional scalar systems along lines in the τ-direction
9. Transform back to entropy variables

We outline the modifications to the diagonalized-ADI scheme in the variable coefficient case. In step 1 the space-time element mass matrix is approximated using collocation resulting in an easily invertible diagonal matrix. In steps 2, 4, 6 and 9 variable transformations are performed locally at each collocation point using the point-wise values for the state and geometry information. Similarly, the scalar systems solved in 3, 5 and 7 correspond to variable-coefficient advection problems with advection velocity corresponding to the eigenvalues at the collocation points.

In the constant-coefficient case, the diagonalized ADI scheme, (35), is identical to the full ADI scheme, (34). However, the diagonalized ADI scheme allows for a significant reduction in computational cost. As opposed to solving one-dimensional systems of size $5N \times 5N$, 5 scalar systems of size $N \times N$ need to be solved. However, due to repeated eigenvalues in the Euler system only 3, as opposed to 5, factorizations need to be performed along each N^{d-1} lines in each coordinate direction. This leads to a cost which scales as $O(3dN^{d+2})$ which represents a savings of about 40 relative to the full ADI scheme.

In the above estimate we have assumed that a separate factorization is performed for each eigenvalue on each line where the scalar systems depend upon the local variable-coefficients. We refer to this scheme as the "variable-coefficient" diagonalized-ADI scheme. We also consider a second scheme we refer to as the "single-factorization" diagonalized-ADI scheme in which we take advantage of the fact that in the constant-coefficient case, the scalar systems in each coordinate direction are identical for each eigenvalue. In the single-factorization scheme we approximate the variable coefficients by an average in the directions normal to each line, such that only 3 factorizations (corresponding to the 3 eigenvalues) are required to be performed in each coordinate direction. The dominant cost for applying this single-factorization diagonalized-ADI preconditioner then becomes the forward- and back- substitutions which gives a cost which scales as $O(5dN^{d+1})$. We note that this cost is, to leading order, the same order as a residual evaluation.

5.2. Fast-Diagonalization Method

For the compressible Euler system, the fast diagonalization method cannot be applied directly, since the system (31) is not a separable tensor-product system. In particular, the flux Jacobians, $\tilde{\boldsymbol{A}}_i$, cannot be simultaneously diagonalized, (i.e. there does not exist a set of eigenvectors $\boldsymbol{R}$, such that $\boldsymbol{R}^{-1}\boldsymbol{A}_i\boldsymbol{R}^{-T}$ is diagonal for $i = 0, \ldots, 3$). Thus it is not possible to rearrange (31) into a block diagonal form where each block corresponding to a characteristic variable is in separable form. However, we recognize that for the Euler system there is a particular choice of eigenvectors $\boldsymbol{R}$, for which $\boldsymbol{R}^{-1}\boldsymbol{A}_i\boldsymbol{R}^{-T}$ has non-zero values on the diagonal and only in the last row and column. We use this transformation to develop an approximate inverse of (31) using FDM and an approximate Schur complement. We proceed to describe this preconditioner in the rest of this section.

First, we present the transformation of variables required in order to modify the system (31) into the desired form. We write $\boldsymbol{R}$ as $\boldsymbol{R} = \boldsymbol{u}_{,\boldsymbol{w}}\boldsymbol{r}$ where $\boldsymbol{w} = \begin{bmatrix} \rho & \boldsymbol{V}^T & p \end{bmatrix}^T$ is the vector of primitive variables, while $\boldsymbol{r}$ is given by:

$$\boldsymbol{r} = \begin{bmatrix} \frac{\rho\sqrt{\gamma-1}}{\sqrt{\gamma\rho}} & 0 & \frac{\rho}{\sqrt{\gamma\rho}} \\ 0 & \frac{c}{\sqrt{\gamma\rho}}\boldsymbol{N} & 0 \\ 0 & 0 & \frac{\rho c^2}{\sqrt{\gamma\rho}} \end{bmatrix}, \tag{36}$$

where $c = \sqrt{\gamma p/\rho}$ is the speed of sound, while $\boldsymbol{N} = \begin{bmatrix} \boldsymbol{N}_1 & \boldsymbol{N}_2 & \boldsymbol{N}_3 \end{bmatrix}$ is a matrix of unit vectors with $N_i = \frac{\nabla\xi_i}{|\nabla\xi_i|}$. The corresponding flux Jacobians are given by:

$$\begin{aligned} \tilde{\Gamma}_i &= \boldsymbol{R}^{-1}\tilde{\boldsymbol{A}}_i\boldsymbol{R}^{-T} \\ &= \begin{bmatrix} (\boldsymbol{V}\cdot\nabla\xi_i) & 0 & 0 \\ 0 & (\boldsymbol{V}\cdot\nabla\xi_i)\boldsymbol{N}^T\boldsymbol{N} & c\boldsymbol{N}^T\nabla\xi_i \\ 0 & c\nabla\xi_i^T\boldsymbol{N} & \boldsymbol{V}\cdot\nabla\xi_i \end{bmatrix}, \end{aligned} \tag{37}$$

for $i = 1, 2, 3$. For elements which are regular parallelepipeds, this simplifies to:

$$\tilde{\Gamma}_i = \begin{bmatrix} (\boldsymbol{V}\cdot\nabla\xi_i) & 0 & 0 \\ 0 & (\boldsymbol{V}\cdot\nabla\xi_i)\boldsymbol{I} & c|\nabla\xi_i|\mathbf{e}_i \\ 0 & c|\nabla\xi_i|\mathbf{e}_i^T & (\boldsymbol{V}\cdot\nabla\xi_i) \end{bmatrix}, \tag{38}$$

where $\mathbf{e}_i$ is the vector with all zeros except for a 1 in the ith coordinate. As noted previously, $\tilde{\boldsymbol{\Gamma}}_i$ are diagonal except for a single entry in the last row/column due to the acoustic coupling terms. We note that the absolute value of the flux Jacobians have a similar structure. For subsonic flow:

$$\begin{aligned} |\tilde{\boldsymbol{\Gamma}}_i| &= \boldsymbol{R}^{-1}|\tilde{\boldsymbol{A}}_a|\boldsymbol{R}^{-T} \\ &= \begin{bmatrix} |\boldsymbol{V}\cdot\nabla\xi_i| & 0 & 0 \\ 0 & |\boldsymbol{V}\cdot\nabla\xi_i|(\boldsymbol{I}-\mathbf{e}_i\mathbf{e}_i^T) + c|\xi_{i,x_k}|\mathbf{e}_i\mathbf{e}_i^T & (\boldsymbol{V}\cdot\nabla\xi_i)\mathbf{e}_i \\ 0 & (\boldsymbol{V}\cdot\nabla\xi_i)\mathbf{e}_i^T & c|\nabla\xi_i| \end{bmatrix}, \end{aligned} \tag{39}$$

while for supersonic flow $|\tilde{\mathbf{\Gamma}}_i| = \pm\tilde{\mathbf{\Gamma}}_i$ depending upon the direction of the flow. These properties allows us to rewrite the system (31) as:

$$\begin{aligned} A &= (I \times I \times I \times I \times \boldsymbol{R}) \times \\ &\quad \{(\tilde{D}_1^+ \otimes I \otimes I \otimes I \otimes \tilde{\mathbf{\Gamma}}_1^+) + (\tilde{D}_1^- \otimes I \otimes I \otimes I \otimes \tilde{\mathbf{\Gamma}}_1^-) \\ &\quad (I \otimes \tilde{D}_2^+ \otimes I \otimes I \otimes \tilde{\mathbf{\Gamma}}_2^+) + (I \otimes \tilde{D}_2^- \otimes I \otimes I \otimes \tilde{\mathbf{\Gamma}}_2^-) \\ &\quad (I \otimes I \otimes \tilde{D}_3^+ \otimes I \otimes \tilde{\mathbf{\Gamma}}_3^+) + (I \otimes I \otimes \tilde{D}_3^- \otimes I \otimes \tilde{\mathbf{\Gamma}}_3^-) \\ &\quad (I \otimes I \otimes I \otimes \tilde{D}_0^+ \otimes \boldsymbol{I})\} \\ &\quad \times (I \times I \times I \times I \times \boldsymbol{R}^T), \end{aligned} \tag{40}$$

where the key observation is that the matrix corresponding to the second through fifth lines of (40) have a similar block structure as the transformed flux Jacobians. Namely, we may write this system as a block matrix of the form:

$$\hat{A} = \begin{bmatrix} A_{00} & & & & \\ & A_{11} & & & B_{14} \\ & & A_{22} & & B_{24} \\ & & & A_{33} & B_{34} \\ & B_{14} & B_{24} & B_{34} & A_{44} \end{bmatrix}, \tag{41}$$

where:

$$\begin{aligned} A_{00} &= (\tilde{D}_1 \otimes I \otimes I \otimes I) + (I \otimes \tilde{D}_2 \otimes I \otimes I) \\ &\quad + (I \otimes I \otimes \tilde{D}_3 \otimes I) + (I \otimes I \otimes I \otimes \tilde{D}_t) \\ A_{11} &= (\tilde{D}_1^\ddagger \otimes I \otimes I \otimes I) + (I \otimes \tilde{D}_2 \otimes I \otimes I) \\ &\quad + (I \otimes I \otimes \tilde{D}_3 \otimes I) + (I \otimes I \otimes I \otimes \tilde{D}_t) \\ A_{22} &= (\tilde{D}_1 \otimes I \otimes I \otimes I) + (I \otimes \tilde{D}_2^\ddagger \otimes I \otimes I) \\ &\quad + (I \otimes I \otimes \tilde{D}_3 \otimes I) + (I \otimes I \otimes I \otimes \tilde{D}_t) \\ A_{33} &= (\tilde{D}_1 \otimes I \otimes I \otimes I) + (I \otimes \tilde{D}_2 \otimes I \otimes I) \\ &\quad + (I \otimes I \otimes \tilde{D}_3^\ddagger \otimes I) + (I \otimes I \otimes I \otimes \tilde{D}_t) \\ A_{44} &= (\tilde{D}_1^\ddagger \otimes I \otimes I \otimes I) + (I \otimes \tilde{D}_2^\ddagger \otimes I \otimes I) \\ &\quad + (I \otimes I \otimes \tilde{D}_3^\ddagger \otimes I) + (I \otimes I \otimes I \otimes \tilde{D}_t) \\ B_{14} &= (\tilde{D}_1^\dagger \otimes I \otimes I \otimes I) \\ B_{24} &= (I \otimes \tilde{D}_2^\dagger \otimes I \otimes I) \\ B_{34} &= (I \otimes I \otimes \tilde{D}_3^\dagger \otimes I). \end{aligned} \tag{42}$$

Here $\tilde{D}_i = M_i^{-1} D_i$, $\tilde{D}_i^\ddagger = M_i^{-1} D_i^\ddagger$ and $\tilde{D}_i^\dagger = M_i^{-1} D_i^\dagger$ are one-dimensional

advection operators corresponding to:

$$D_i[m,n] = -\int_{\xi_i} \tilde{a}_i w_{,\xi_i} v + \left[\tfrac{1}{2}(\tilde{a}_i + \xi_i|\tilde{a}_i|)wv\right]_{\xi_i=-1}^{\xi_i=1} \quad (43)$$

$$D_i^{\ddagger}[m,n] = -\int_{\xi_i} \tilde{a}_i w_{,\xi_i} v \left[\tfrac{1}{2}(\tilde{a}_i + \xi_i\tilde{c}_i)wv\right]_{\xi_i=-1}^{\xi_i=1} \quad (44)$$

$$D_i^{\dagger}[m,n] = -\int_{\xi_i} \tilde{c}_i w_{,\xi_i} v + \left[\tfrac{1}{2}(\tilde{c}_i + \xi_i\tilde{a}_i)wv\right]_{\xi_i=-1}^{\xi_i=1} \quad (45)$$

where $\tilde{c}_i = c|\nabla\xi_i|$.

We can now perform a block-LDU factorization of $\hat{A}$ appearing in (41):

$$\hat{A} = \begin{bmatrix} I & & & & \\ & I & & & \\ & & I & & \\ & & & I & \\ & B_{14}A_{11}^{-1} & B_{24}A_{22}^{-1} & B_{34}A_{33}^{-1} & I \end{bmatrix} \times \begin{bmatrix} A_{00} & & & & \\ & A_{11} & & & \\ & & A_{22} & & \\ & & & A_{33} & \\ & & & & S_{44} \end{bmatrix} \begin{bmatrix} I & & & & \\ & I & & & A_{11}^{-1}B_{14} \\ & & I & & A_{22}^{-1}B_{24} \\ & & & I & A_{33}^{-1}B_{34} \\ & & & & I \end{bmatrix}, \quad (46)$$

where S_{44} is the Schur complement given by:

$$S_{44} = A_{44} - B_{14}A_{11}^{-1}B_{14} - B_{24}A_{22}^{-1}B_{24} - B_{34}A_{33}^{-1}B_{34}. \quad (47)$$

The inverse of $\hat{A}$ is given by:

$$\hat{A}^{-1} = \begin{bmatrix} I & & & \\ & I & & -A_{11}^{-1}B_{14} \\ & & I & -A_{22}^{-1}B_{24} \\ & & & -A_{33}^{-1}B_{34} \\ & & & I \end{bmatrix} \begin{bmatrix} A_{00}^{-1} & & & & \\ & A_{11}^{-1} & & & \\ & & A_{22}^{-1} & & \\ & & & A_{33}^{-1} & \\ & & & & S_{44}^{-1} \end{bmatrix} \times \begin{bmatrix} I & & & & \\ & I & & & \\ & & I & & \\ & & & I & \\ & -B_{14}A_{11}^{-1} & -B_{24}A_{22}^{-1} & -B_{34}A_{33}^{-1} & I \end{bmatrix}. \quad (48)$$

We note that A_{ii}^{-1}, $i = 0,\ldots,3$ may be efficiently applied using the fast-diagonalization method, while multiplication by B_{i4}, $i = 1,\ldots,3$ may be efficiently performed using the sum factorization approach. So far we have made no approximations other than those assumptions relied upon to develop the diagonalized ADI preconditioner: namely, constant-coefficient problem on parallelepiped elements. Thus exact inversion of the Schur complement would give

an exact elemental Jacobian inversion. Unfortunately, the Schur complement matrix is in general full, and, as we have discussed numerous times, is prohibitively expensive to store at high polynomial order. Instead of forming and inverting the Schur complement S_{44} directly, we invert an approximate Schur complement using the fast-diagonalization method.

We note that S_{44} is not a separable tensor-product matrix due to the presence of the terms $B_{i4}A_{ii}^{-1}B_{i4}$. Instead we form an approximation to the Schur complement $\tilde{S}^{44}$ which is in separable form:

$$\tilde{S}_{44} = A_{44} - B_{14}\tilde{A}_{11}^{-1}B_{14} - B_{24}\tilde{A}_{22}^{-1}B_{24} - B_{34}\tilde{A}_{33}^{-1}B_{34}, \tag{49}$$

where $\tilde{A}_{ii}$ are spectral radius approximations given by:

$$\begin{aligned}
\tilde{A}_{11} &= (\tilde{D}_1^{\ddagger} + (\lambda_{\min}(\tilde{D}_2) + \lambda_{\min}(\tilde{D}_3) + \lambda_{\min}(\tilde{D}_t))I \otimes I \otimes I) && (50)\\
\tilde{A}_{22} &= (I \otimes \tilde{D}_2^{\ddagger} + (\lambda_{\min}(\tilde{D}_1) + \lambda_{\min}(\tilde{D}_3) + \lambda_{\min}(\tilde{D}_t))I \otimes I \otimes I) && (51)\\
\tilde{A}_{33} &= (I \otimes I \otimes \tilde{D}_3^{\ddagger} + (\lambda_{\min}(\tilde{D}_1) + \lambda_{\min}(\tilde{D}_2) + \lambda_{\min}(\tilde{D}_t))I \otimes I), && (52)
\end{aligned}$$

where $\lambda_{\min}(\tilde{D}_i)$ denotes the smallest real component of the eigenvalues of $\tilde{D}_i$. We note that $\tilde{S}_{44}$ is not explicitly formed; only the application of its inverse is required which is performed using FDM.

We note several important features of the FDM-based preconditioner for the constant-coefficient linearized Euler equations. First, the FDM preconditioner provides an exact block inversion for the convective (entropy) mode, corresponding to the first block row of (41), as this mode decouples from the other variables. Second, $\frac{4}{5}^{\text{ths}}$ of the eigenvectors of the operator $\tilde{A}^{-1}A - I$, corresponding to a Richardson iteration on the element block Jacobian, are identically zero as the lower and upper factors in (46) are computed exactly.

Extension of the FDM-based preconditioner to the compressible Navier-Stokes equations is performed by including the diagonal of the viscous flux Jacobian. Applying the transformations $\boldsymbol{R}^{-1}\tilde{\boldsymbol{K}}_{ij}\boldsymbol{R}^{-T}$ gives viscous flux Jacobians:

$$\boldsymbol{R}^{-1}\tilde{\boldsymbol{K}}_{ij}\boldsymbol{R}^{-T} = \nu|\nabla\xi_i||\nabla\xi_j| \times$$
$$\begin{bmatrix}
\frac{1}{Pr}\boldsymbol{N}_i^T\boldsymbol{N}_j & 0 & -\frac{\sqrt{\gamma-1}}{Pr}\boldsymbol{N}_i^T\boldsymbol{N}_j \\
0 & \boldsymbol{N}_i^T\boldsymbol{N}_j\boldsymbol{N}^T\boldsymbol{N} - (\boldsymbol{N}\times\boldsymbol{N})(\boldsymbol{N}_i\times\boldsymbol{N}_j) + \frac{1}{3}(\boldsymbol{N}^T\boldsymbol{N}_i)^T(\boldsymbol{N}^T\boldsymbol{N}_j) & 0 \\
-\frac{\sqrt{\gamma-1}}{Pr}\boldsymbol{N}_i^T\boldsymbol{N}_j & 0 & \frac{(\gamma-1)}{Pr}\boldsymbol{N}_i^T\boldsymbol{N}_j
\end{bmatrix}.$$

where, Pr is the Prandtl number. For parallelepiped elements, the diagonals of the viscous flux Jacobian simplifies to:

$$R^{-1}\tilde{\boldsymbol{K}}_{ii}R^{-T} = \tilde{\nu}_{ii}\begin{bmatrix}
\frac{1}{Pr} & 0 & -\frac{\sqrt{\gamma-1}}{Pr} \\
0 & \boldsymbol{I} + \frac{1}{3}\mathbf{e}_i\mathbf{e}_i^T & 0 \\
-\frac{\sqrt{\gamma-1}}{Pr} & 0 & \frac{(\gamma-1)}{Pr}
\end{bmatrix}$$

Thus, inclusion of the diagonal of the viscous flux Jacobian does not alter the block-structure of (41) except that the convective mode no longer decouples

from the other modes. Additionally, omitting $\frac{1}{3}\mathbf{e}_i\mathbf{e}_i^T$ from the diagonal terms and all off-diagonal terms $\boldsymbol{K}_{ij}$, $i \neq j$, we can recover the exact Jacobian in the limit of incompressible flow. Thus omission of these terms may only when significant compressibility effects are present.

As with the diagonalized-ADI scheme, we present the FDM-based preconditioner as a sequence of operations and discuss the modifications to these operations in the case of variable-coefficient problems on general curvilinear grids. The application of the FDM preconditioner is given by the following steps:

1. Multiply by the inverse of the space-time element mass matrix
2. Transform to "characteristic variables" using the transformation $\boldsymbol{R}^{-1}$
3. Apply A_{ii}^{-1}, $i = 0, \ldots, 3$ using FDM
4. Apply B_{i4} using the sum-factorization approach to compute the residual for the Schur complement problem
5. Solve the Schur complement problem by applying $\tilde{S}_{44}^{-1}$ using FDM
6. Multiplying the results by $A_{ii}^{-1}B_{i4}$ by applying the sum-factorization approach followed by FDM in order to update the remaining state
7. Transform back to entropy variables

In the variable-coefficient case, the elemental flux Jacobian does not have the simple form given in (31), however we may apply the FDM-based preconditioner in the general case, by making the following modifications. In step 1 the space-time element mass matrix is approximated using collocation resulting in an easily invertible diagonal matrix. In steps 2 and 6 variable transformations are performed locally at each collocation point using the point-wise values for the state and geometry information. The system matrices in 3-6 are formed based on elemental means of the eigenvalues computed at the collocation points, in a manner similar to the single-factorization ADI approach.

Finally we discuss the cost of applying the FDM-based preconditioner. The dominant cost is due to the application of the eigenvector matrices using the sum-factorization approach in the FDM inversions. In total 9 FDM inversions are performed for a single application of the preconditioner leading to a total cost which scales as $O(9dN^{d+1})$. Thus, to leading order, the cost of the FDM-based preconditioner is the same as a single residual evaluation similar to the single-factorization ADI approach.

6. Numerical Results

We now present numerical results showing the performance of the diagonalized ADI and FDM-based preconditioners for solving the unsteady Navier-Stokes equations. We consider two problems: direct numerical simulation of the Taylor-Green vortex evolution and direct numerical simulation of a turbulent channel flow.

6.1. Taylor-Green Vortex

The Taylor-Green vortex flow is simulated using the compressible Navier-Stokes equations at Mach number $M_0 = 0.1$ and Reynolds number $Re = \frac{\rho_0 V_0 L}{\mu} = 1600$. The flow is solved on an isotropic domain which spans $[0, 2\pi L]$ in each coordinate direction. The initial conditions are given by:

$$u = V_0 \sin(x/L)\cos(y/L)\cos(z/L) \quad (53)$$

$$v = -V_0 \cos(x/L)\sin(y/L)\cos(z/L) \quad (54)$$

$$w = 0 \quad (55)$$

$$p = \rho_0 V_0^2 \left[\frac{1}{\gamma M_0^2} + \frac{1}{16}\left(\cos(2x) + \sin(2y)\right)\left(\cos(2z) + 2\right)\right] \quad (56)$$

where u, v and w are the components of the velocity in the x, y and z-directions, p is the pressure and ρ is the density. The flow is initialized to be isothermal ($\frac{p}{\rho} = \frac{p_0}{\rho_0} = RT_0$). Starting from the simple initial condition, the flow becomes turbulent through repeated vortex stretching leading to progressively smaller eddies, which are then dissipated through the action of molecular diffusion.

As in the case of scalar advection-diffusion we examine the performance of our preconditioners as a function of solution order, degrees of freedom, and CFL number based on the accoustic speed, $CFL = \frac{c\Delta t}{hN}$. We consider four preconditioners: mass-matrix, diagonalized-ADI with variable coefficients, diagonalized-ADI with a single factorization, and the FDM-based preconditioner. Tables 7 - 10 report the number of iterations required to converge the residual to 10^{-14} for a single time slab starting at the point of peak dissipation. Tables 7 - 10 also give the corresponding CPU time per spatial degree of freedom for a simulation time corresponding to CFL = 1.

The mass matrix preconditioner is too weak to converge the space-time system with increasing CFL number or solution order. Using the tensor-product preconditioners allows for larger time-steps to be taken. In particular, the tensor-product preconditioners are able to overcome the increased stiffness associated with high-order to allow for the time step to be dictated by the physics of the problem as opposed to an explicit stability limit. For very large time steps, the combination of insufficiently strong preconditioning and the nonlinearity present in the problem prevent the Newton-Krylov scheme from converging. In studies not presented here, an improved nonlinear localization scheme using pseudo-transient continuation has allowed for convergence with somewhat larger time steps using ADI and FDM preconditioners. However, in the scope of this paper we do not consider other nonlinear localization techniques.

The two variants of the ADI preconditioner have very similar performance in terms of the number of iterations required to converge. The two ADI preconditioners are identical for constant-coefficient problems. It is initially somewhat surprising that the performance of the two preconditioners are so similar given the large spatial variations in the velocity near the point of peak dissipation. However, for this low Mach number flow, the accoustic speed is essentially constant within the domain and it is likely that the convergence is limited by the accoustic modes. In terms of CPU time, the variable-coefficient diagonalized-ADI

scheme has a cost about twice that of the single-factorization diagonalized-ADI scheme, justifying the single-factorization approximation.

The FDM-based preconditioner performs somewhat better than the single-factorization ADI scheme in terms of the number of iterations required to converge. In terms of CPU time, however, the FDM-based preconditioner is somewhat worse than the ADI based preconditioner as the cost of applying the FDM preconditioner is more than the single-factorization diagonalized-ADI scheme. At $CFL = 1 - 4$ the single-factorization ADI preconditioning is approximately 10%-30% faster than the FDM-based preconditioner (provided both schemes converge).

The ADI-based preconditioners are slightly more robust than the FDM preconditioner, resulting in more reliable convergence for large CFL numbers when $N = 8$. However, as noted before, at very large CFL numbers, (particularly at higher-order) lack of convergence of the Newton-Krylov scheme appears to be driven by a combination of both weak preconditioning and the nonlinearity of the problem. We have noticed that at high solution order and large CFL numbers the FDM preconditioner gives a much larger reduction in linear residual than the ADI preconditioners in the first step of the Newton scheme (using 20 Krylov vectors). However, this generally results in a poorer update with a larger nonlinear residual after the first iteration when compared to the ADI scheme. Currently, a simple line-search is employed at each Newton-Krylov iteration to ensure decrease in the non-linear residual. A more advanced nonlinear localization strategy may be necessary in the limit of large time-steps at higher-order. Conclusions about the robustness of the preconditioning scheme in the large CFL limit may be premature prior to a more complete investigation of nonlinear continuation strategies.

Finally, the performance of the ADI and FDM preconditioners are compared with an exact element-wise block-Jacobi preconditioner commonly used for DG methods at low polynomial order. As noted previously the memory required for storing the block-Jacobian and its factorization may be prohibitive at high polynomial order. In order to overcome this memory limitation the block-Jacobi preconditioner is implemented in a matrix-free manner using subiterations to converge each block system to machine zero. In this simple implementation, the cost of the block-Jacobi preconditioner is about two orders of magnitude more expensive than the ADI or FDM preconditioners at 8th order. Thus, we consider only the coarsest mesh resolution and report convergence only in terms of the number of iterations as given in Table 11. At 2nd order, the performance of the Block-Jacobi preconditioner is nearly identical to the FDM preconditioner. With increasing polynomial order the performance of the block-Jacobi preconditioner improves relative to the ADI and FDM preconditioners, particularly with increasing CFL number. At 8th-order the block-Jacobi preconditioner converges using 2-3 times fewer iterations than the ADI preconditioner at the largest CFL numbers considered. However, in term of CPU time the ADI and FDM preconditioners are must more efficient than the block-Jacobi preconditioner which is significantly more expensive to apply then the tensor-product preconditioners. Thus the ADI and FDM preconditioners provide an efficient low-memory

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	66	108	195	370	714	*	*
64	65	108	196	373	714	*	*
96	64	109	197	375	735	*	*
128	63	109	197	374	735	*	*

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	272.8	204.3	169.7	153.6	144.4	*	*
64	416.8	284.6	264.1	215.5	202.9	*	*
96	468.0	358.7	305.9	272.2	265.0	*	*
128	462.8	374.3	304.6	281.8	274.0	*	*

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	138	231	448	916	*	*	*
64	158	231	430	897	*	*	*
96	137	231	427	905	*	*	*
128	136	209	410	894	*	*	*

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	574.9	457.6	435.5	443.2	*	*	*
64	698.7	451.7	450.3	438.7	*	*	*
96	747.1	590.2	525.2	556.3	*	*	*
128	787.8	596.5	601.3	627.1	*	*	*

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	311	519	1032	*	*	*	*
64	313	521	1027	*	*	*	*
96	304	516	1024	*	*	*	*
128	285	225	1002	987	*	*	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	1609.2	1316.8	1298.0	*	*	*	*
64	1740.8	1388.4	1344.8	*	*	*	*
96	2171.9	1778.5	1754.3	*	*	*	*
128	2134.7	1842.2	1808.8	854.7	*	*	*

Table 7: Number of GMRES Iterations and CPU time (μs) for Taylor-Green vortex problem using the mass-matrix preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	41	68	90	159	293	567	*
64	40	68	90	159	293	566	*
96	40	67	90	159	294	587	*
128	39	58	86	159	294	567	*

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	538.6	418.1	269.5	230.8	208.9	200.1	*
64	748.2	478.5	280.4	237.3	212.8	203.3	*
96	582.6	453.5	294.7	251.8	228.7	226.8	*
128	605.0	406.3	303.8	267.5	239.6	229.8	*

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	77	103	162	231	400	949	*
64	78	101	157	231	402	821	*
96	76	93	138	210	401	802	*
128	72	97	140	189	397	787	*

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	943.5	591.7	442.4	309.9	266.8	316.2	*
64	975.2	609.3	445.0	319.1	275.9	284.8	*
96	993.1	599.8	442.1	321.8	305.6	305.1	*
128	1029.4	679.8	479.3	314.4	333.8	324.6	*

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	104	156	248	355	680	*	*
64	110	159	239	409	724	*	*
96	107	149	232	337	608	*	*
128	101	101	240	139	568	*	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	1475.5	1058.4	818.2	579.3	549.2	*	*
64	1667.0	1137.1	843.9	687.6	597.0	*	*
96	1779.4	1148.6	878.8	628.0	561.4	*	*
128	1740.3	864.3	849.2	273.6	524.1	*	*

Table 8: Number of GMRES Iterations and CPU time (μs) for Taylor-Green vortex problem using the variable-coefficient diagonalized-ADI preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	41	69	90	159	293	567	*
64	40	68	90	159	293	566	*
96	40	67	90	159	294	587	*
128	39	58	86	159	294	567	*

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	305.5	239.5	149.0	124.4	111.4	102.6	*
64	533.2	263.6	162.8	135.3	117.2	110.2	*
96	377.7	284.9	177.8	151.1	134.3	133.8	*
128	425.1	257.6	180.6	154.1	138.5	134.1	*

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	77	103	162	231	400	953	*
64	78	101	157	231	402	821	*
96	76	93	138	210	401	801	*
128	71	97	139	189	397	786	*

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	424.2	263.9	187.2	127.4	109.5	129.2	*
64	498.6	320.2	193.4	144.6	115.4	112.3	*
96	473.4	266.9	187.9	144.1	134.7	130.1	*
128	486.4	321.0	217.0	149.2	156.2	147.3	*

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	104	156	247	355	679	*	*
64	111	159	240	400	720	*	*
96	109	151	233	337	587	*	*
128	101	*	226	139	743	*	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	701.7	485.4	357.8	250.7	233.3	*	*
64	816.5	519.7	373.3	303.2	260.9	*	*
96	927.9	599.9	447.9	313.9	269.0	*	*
128	928.1	*	444.0	175.1	270.5	*	*

Table 9: Number of GMRES Iterations and CPU time (μs) for Taylor-Green vortex problem using the single-factorization diagonalized-ADI preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	30	52	80	114	183	377	687
64	30	51	79	113	184	394	747
96	30	50	78	114	183	396	749
128	29	42	75	109	183	377	731

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	408.6	323.7	235.6	163.5	128.6	129.8	117.5
64	482.8	406.6	240.0	178.6	135.5	137.9	129.1
96	475.6	348.1	252.7	178.5	145.2	149.3	140.5
128	476.3	311.2	279.7	189.3	150.8	151.4	146.8

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	59	92	132	199	284	481	*
64	62	94	135	197	263	549	*
96	62	84	117	185	244	476	*
128	56	86	121	171	229	401	*

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	420.2	305.7	209.2	151.0	106.8	88.9	*
64	555.9	369.4	237.7	163.3	118.1	114.5	*
96	626.9	381.2	250.3	191.9	125.7	119.7	*
128	636.1	424.3	296.7	196.0	128.9	112.1	*

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	112	159	225	327	*	*	*
64	113	165	218	354	*	*	*
96	114	158	202	315	*	*	*
128	199	104	362	147	932	*	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
48	915.8	603.8	414.8	299.4	*	*	*
64	1100.5	734.9	463.5	370.4	*	*	*
96	1265.2	815.2	513.6	375.3	*	*	*
128	1297.1	598.5	592.6	204.5	536.2	*	*

Table 10: Number of GMRES Iterations and CPU time (μs) for Taylor-Green vortex problem using the FDM-based preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

alternative to the block-Jacobi preconditioner.

(a) $N = 2$

	CFL						
	$\frac{1}{2}$	1	2	4	8	16	32
Mass-matrix	66	108	195	370	714	*	*
ADI (variable)	41	68	90	159	293	567	*
ADI (single)	41	69	90	159	293	567	*
FDM	30	52	80	114	183	377	687
Block-Jacobi	30	51	78	112	205	375	717

(b) $N = 4$

	CFL						
	$\frac{1}{2}$	1	2	4	8	16	32
Mass-matrix	138	231	448	916	*	*	*
ADI (variable)	77	103	162	231	400	949	*
ADI (single)	77	103	162	231	400	953	*
FDM	59	92	132	199	284	481	*
Block-Jacobi	50	75	112	161	248	358	646

(c) $N = 8$

	CFL						
	$\frac{1}{2}$	1	2	4	8	16	32
Mass-matrix	311	519	1032	*	*	*	*
ADI (variable)	104	156	248	355	680	*	*
ADI (single)	104	156	247	355	679	*	*
FDM	112	159	225	327	*	*	*
Block-Jacobi	79	99	144	198	306		

Table 11: Number of GMRES Iterations for Taylor-Green vortex problem with $1/h = 48$ using mass-matrix, ADI, FDM and Block-Jacobi preconditioners. (* denotes failure of convergence of restarted GMRES algorithm.)

6.2. Turbulent Channel Flow

As a second test case, we consider the flow in a channel at $Re_\tau = 180$, where $Re_\tau = \frac{u_\tau \delta}{\nu}$ is the Reynolds number based on the wall shear velocity, $u_\tau = \sqrt{\tau_w/\rho}$, the channel half-width, δ, and the kinematic viscosity, $\nu = \mu/\rho$. The simulations performed are nearly incompressible with a Mach number of approximately 0.1 based on the bulk velocity and mean speed of sound. Following Kim et al. [25] the domain is of size $4\pi\delta \times 2\delta \times 2\pi\delta$, corresponding to stream-wise, normal, and span-wise directions, respectively. The domain is periodic in the stream-wise and span-wise directions while adiabatic no-slip boundary conditions are applied at the channel walls. The flow is driven by a constant body force applied to the stream-wise momentum equation.

As with the previous test cases, we examine the performance of our preconditioners as a function of solution order, degrees of freedom, and CFL number based on the accoustic speed. We consider three set of meshes corresponding to $96 \times 64 \times 80$, $144 \times 96 \times 128$ and $192 \times 128 \times 160$ degrees of freedom in stream-wise, wall-normal and span-wise directions. Tables 12 - 15 report the number of GMRES iterations and CPU time required to converge to a residual of 10^{-13} for each time slab. The results for the turbulent channel flow case are similar to that for the Taylor-Green vortex problem. Namely, the mass-matrix preconditioner is sufficient only at small CFL or low solution order. The diagonalized-ADI and FDM-based preconditioners allow convergence at higher solution orders and larger CFL numbers. Once again, both variants of the diagonalized-ADI preconditioner perform similarly in terms of the number of iterations. The single-factorization diagonalized-ADI is about a factor of 2 faster in terms of CPU time. The FDM-based preconditioner tends to converge in the fewest number of iterations, but the cost in terms of CPU time is about the same for FDM and single-factorization diagonalized-ADI schemes. Just as we have observed with the Taylor-Green vortex test case, the convergence at high solution order and large CFL numbers is limited by a combination of insufficiently strong preconditioning and nonlinearity. Using the Newton-Krylov scheme with ADI or FDM-based preconditioners allow for an 8th order simulation to be performed at a $CFL = 8$ in a CPU time that is approximately 10 times faster than using the mass-matrix preconditioner at the largest CFL number for which the Newton-Krylov scheme would converge.

7. Conclusions

We have presented a higher-order space-time discontinuous Galerkin method for the direct numerical simulation of subsonic turbulent compressible flows. An efficient matrix-free Newton-Krylov method is used for the solution of the nonlinear system of equations arising at each space-time slab. The use of a tensor-product formulation is vital in order to maintain an efficient implementation. The sum-factorization ensures efficient residual evaluation as well as linearized residual evaluation used in the matrix-free Krylov scheme.

Two classes of preconditioners were presented which take advantage of the tensor-product formulation: a diagonalized-Alternating Direction Implicit (ADI) scheme and a preconditioner based on the Fast Diagonalization Method (FDM). For advection-dominated scalar advection-diffusion problems, the ADI preconditioner demonstrates convergence in a number of iterations independent of the solution order for a fixed number of degrees of freedom, thus over-coming the stiffness associated with higher-order methods. The FDM preconditioner essentially gives an exact elemental solve for scalar problems, providing a very efficient scheme at high-order.

The ADI and FDM-based preconditioners were extended to the solution of the unsteady compressible Navier-Stokes equations. The performance of the ADI and FDM preconditioners were assessed for direct numerical simulation of the Taylor-Green vortex problem and turbulent flow in a channel. The

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	58	104	214	395	805	*	*
$144 \times 96 \times 120$	66	150	255	505	*	*	*
$192 \times 128 \times 160$	66	129	255	524	1032	*	*

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	519.7	385.9	368.8	276.6	263.1	*	*
$144 \times 96 \times 120$	837.0	865.0	579.4	448.5	*	*	*
$192 \times 128 \times 160$	1031.7	649.2	575.3	434.3	416.8	*	*

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	194	343	663	*	*	*	*
$144 \times 96 \times 120$	194	342	697	*	*	*	*
$192 \times 128 \times 160$	173	341	702	*	*	*	*

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	946.4	812.0	748.6	*	*	*	*
$144 \times 96 \times 120$	1412.6	1105.6	920.8	*	*	*	*
$192 \times 128 \times 160$	1227.8	1100.3	1032.7	*	*	*	*

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	461	839	*	*	*	*	*
$144 \times 96 \times 120$	446	845	*	*	*	*	*
$192 \times 128 \times 160$	430	909	*	*	*	*	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	2488.2	2199.8	*	*	*	*	*
$144 \times 96 \times 120$	3178.7	2956.6	*	*	*	*	*
$192 \times 128 \times 160$	3049.1	3219.6	*	*	*	*	*

Table 12: Number of GMRES Iterations and CPU time (μs) for turbulent channel DNS using the mass-matrix preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	47	61	68	117	189	336	696
$144 \times 96 \times 120$	51	62	89	135	247	441	857
$192 \times 128 \times 160$	51	62	90	134	244	462	882

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	851.6	450.9	322.9	188.0	162.1	136.6	127.2
$144 \times 96 \times 120$	1673.4	541.8	527.0	341.7	264.5	218.4	179.3
$192 \times 128 \times 160$	1016.2	582.2	464.5	273.8	216.4	199.9	175.9

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	93	97	110	164	315	592	*
$144 \times 96 \times 120$	77	99	111	164	315	592	*
$192 \times 128 \times 160$	80	83	112	185	315	612	*

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	1191.9	626.4	374.3	264.1	233.4	216.8	*
$144 \times 96 \times 120$	1519.4	876.7	410.6	269.0	280.1	241.1	*
$192 \times 128 \times 160$	1271.3	691.0	490.3	340.2	265.0	254.4	*

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	105	135	166	228	402	853	*
$144 \times 96 \times 120$	110	119	167	245	417	895	*
$192 \times 128 \times 160$	110	120	189	284	460	895	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	1518.0	947.4	561.8	381.4	332.3	345.5	*
$144 \times 96 \times 120$	1845.4	1001.1	680.9	496.9	415.1	410.7	*
$192 \times 128 \times 160$	1888.6	1002.2	736.1	555.5	451.8	428.1	*

Table 13: Number of GMRES Iterations and CPU time (μs) for turbulent channel DNS using the variable-coefficient diagonalized-ADI preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	47	61	68	117	189	336	696
$144 \times 96 \times 120$	51	62	89	135	247	441	857
$192 \times 128 \times 160$	51	62	90	134	244	462	882

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	645.6	319.2	228.6	132.0	100.7	84.9	74.7
$144 \times 96 \times 120$	1557.3	564.8	476.0	239.8	144.8	114.5	102.3
$192 \times 128 \times 160$	847.4	547.0	255.1	184.3	139.6	118.1	106.3

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	93	97	110	164	315	592	*
$144 \times 96 \times 120$	77	99	111	164	315	592	*
$192 \times 128 \times 160$	80	83	112	185	315	612	*

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	642.8	275.5	211.6	127.0	102.9	95.4	*
$144 \times 96 \times 120$	633.7	682.6	226.7	130.8	161.6	106.9	*
$192 \times 128 \times 160$	800.7	440.5	216.8	167.6	132.8	119.4	*

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	105	135	166	228	402	853	*
$144 \times 96 \times 120$	110	119	167	245	417	895	*
$192 \times 128 \times 160$	110	120	189	284	460	916	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	721.9	429.6	260.0	173.7	149.3	151.8	*
$144 \times 96 \times 120$	1041.3	500.3	325.6	249.2	200.2	207.2	*
$192 \times 128 \times 160$	1051.6	557.4	388.3	288.2	224.2	216.5	*

Table 14: Number of GMRES Iterations and CPU time (μs) for turbulent channel DNS using the single-factorization diagonalized-ADI preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

(a) Iterations, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	33	43	60	70	124	206	413
$144 \times 96 \times 120$	36	48	61	92	162	294	525
$192 \times 128 \times 160$	37	52	61	112	159	294	567

(b) CPU Time, $N = 2$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	774.1	494.6	238.2	137.4	103.8	74.3	74.1
$144 \times 96 \times 120$	678.0	975.0	445.1	300.6	160.2	132.6	108.7
$192 \times 128 \times 160$	804.9	589.4	284.0	279.8	141.8	118.3	119.2

(c) Iterations, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	83	95	107	139	252	516	*
$144 \times 96 \times 120$	76	98	108	139	210	356	*
$192 \times 128 \times 160$	76	82	109	138	209	315	630

(d) CPU Time, $N = 4$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	790.2	394.8	214.7	120.8	112.3	105.3	*
$144 \times 96 \times 120$	1057.6	787.4	368.5	186.2	147.4	94.9	*
$192 \times 128 \times 160$	785.7	411.4	283.2	176.6	108.0	84.5	81.3

(e) Iterations, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	108	134	160	273	662	*	*
$144 \times 96 \times 120$	111	134	180	252	441	*	*
$192 \times 128 \times 160$	111	134	181	251	399	*	*

(f) CPU Time, $N = 8$

	CFL						
$1/h$	$\frac{1}{2}$	1	2	4	8	16	32
$96 \times 64 \times 80$	906.9	574.1	324.3	262.8	316.3	*	*
$144 \times 96 \times 120$	1129.5	689.4	444.6	294.4	256.1	*	*
$192 \times 128 \times 160$	1182.0	695.5	430.0	285.9	227.0	*	*

Table 15: Number of GMRES Iterations and CPU time (μs) for turbulent channel DNS using the FDM-based preconditioner. N is the solution order, while $1/h$ is the number of degrees of freedom in each coordinate direction. (* denotes failure of convergence of restarted GMRES algorithm.)

tensor-product preconditioners allow for higher-order solutions to be performed at larger CFL numbers, overcoming the stiffness associated with higher-order. The tensor-product preconditioners allow for 8th order turbulent flow simulations to be performed approximately an order of magnitude faster than using a simple mass-matrix preconditioner.

Acknowledgements

This work was partially funded by the Revolutionary Computational Aerosciences sub-project within NASAs Aeronautics Sciences project.

References

[1] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845.

[2] Bassi, F. and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by K. Cockburn and Shu, Springer, Berlin, 2000, pp. 197–208.

[3] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid solution of high-order discontiunous Galerkin discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 207, No. 1, 2005, pp. 92–113.

[4] Persson, P.-O. and Peraire, J., "An efficient low memory implicit DG algorithm for time dependent problems," AIAA 2006-0113, 2006.

[5] Diosady, L. T. and Darmofal, D. L., "Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations," *Journal of Computational Physics*, Vol. 228, 2009, pp. 3917–3935.

[6] Barth, T. J., "Numerical Methods for Gasdynamic Systems on Unstructured Meshes," *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, edited by D. Kroner, M. Olhberger, and C. Rohde, Springer-Verlag, 1999, pp. 195 – 282.

[7] van der Vegt, J. and van der Ven, H., "Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. I. General formulation," *Journal of Computational Physics*, Vol. 182, 2002, pp. 546–585.

[8] C.M. Klaij, J. v. d. V. and van der Ven, H., "Space-time discontinuous Galerkin methods for the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 217, 2006, pp. 589–611.

[9] Peaceman, D. W. and Rachford Jr., H. H., "The numerical solution of parabolic and elliptic differential equations," *SIAM J.*, Vol. 3, No. 1, 1955, pp. 28–41.

[10] Beam, R. and Warming, R., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 393 – 402.

[11] Pulliam, T. and Chaussee, D., "A Diagonal Form of an Implicit Approximate-Factorization Algorithm," *Journal of Computational Physics*, Vol. 39, 1981, pp. 347–363.

[12] Diosady, L. T. and Murman, S. M., "Design of a variational multiscale method for turbulent compressible flows," AIAA Paper 2013-2870, 2013.

[13] Leffell, J. I., Murman, S. M., and Pulliam, T. H., "An Extension of the Time-Spectral Method to Overset Solvers," AIAA Paper 2013-0637, 2013.

[14] Lynch, R. E., Rice, J. R., and Thomas, D. H., "Direct solution of partial difference equations by tensor product methods," *Numerische Mathematik*, Vol. 6, No. 1, 1964, pp. 185–199.

[15] Lottes, J. W. and Fischer, P. F., "Hybrid Multigrid/Schwarz Algorithms for the Spectral Element Method," *Journal of Scientific Computing*, Vol. 24, No. 1, 2005, pp. 45–78.

[16] Lott, P. A. and Elman, H., "Fast iterative solver for convection-diffusion systems with spectral elements," *Numerical Methods for Partial Differential Equations*, Vol. 27, No. 2, 2011, pp. 231–254.

[17] Hughes, T. J. R., Franca, L., and Mallet, M., "A new finite element formulation for computational fluid dynamics: I Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics," Vol. 54, 1986, pp. 223–234.

[18] Ismail, F. and Roe, P. L., "Affordable, Entropy-consistent Euler flux functions II: entropy production at shocks," *J. Comput. Phys.*, Vol. 228, No. 15, Aug. 2009, pp. 5410–5436.

[19] Kirby, R. M. and Karniadakis, G. E., "De-aliasing on non-uniform grids: algorithms and applications," *Journal of Computational Physics*, Vol. 191, 2003, pp. 249–264.

[20] Vos, P., Sherwin, S., and Kirby, R., "From h to p Efficiently: Implementing finite and spectral/hp element discretizations to achieve optimal performance at low and high order approximations." *Journal of Computational Physics*, Vol. 229, No. 13, 2010, pp. 5161–5181.

[21] Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[22] Knoll, D. A. and Keyes, D. E., "Jacobian-free Newton-Krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, Vol. 193, No. 1, 2004, pp. 357–397.

[23] Persson, P.-O., "A sparse and high-order accurate line-based discontinuous Galerkin method for unstructured meshes," *Journal of Computational Physics*, Vol. 233, 2013, pp. 414 – 429.

[24] Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[25] Kim, J., Moin, P., and Moser, R., "Turbulence statistics in fully developed channel flow at low Reynolds number," *Journal of Fluid Mechanics*, Vol. 177, 1987, pp. 133–166.