INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2016; **00**:1–23
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld

A Parallel Adaptive Numerical Method with Generalized Curvilinear Coordinate Transformation for Compressible Navier-Stokes Equations

X. Gao*, L. D. Owen, and S. M. J. Guzik

Computational Fluid Dynamics and Propulsion Laboratory, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO, 80523, USA

SUMMARY

A fourth-order finite-volume method for solving the Navier-Stokes equations on a mapped grid with adaptive mesh refinement is proposed, implemented, and demonstrated for the prediction of unsteady compressible viscous flows. The method employs fourth-order quadrature rules for evaluating face-averaged fluxes. Our approach is freestream preserving, guaranteed by the way of computing the averages of the metric terms on the faces of cells. The standard Runge-Kutta marching method is used for time discretization. Solutions of a smooth flow are obtained in order to verify that the method is formally fourth-order accurate when applying the nonlinear viscous operators on mapped grids. Solutions of a shock tube problem are obtained to demonstrate the effectiveness of adaptive mesh refinement in resolving discontinuities. A Mach reflection problem is solved to demonstrate the mapped algorithm on a non-rectangular physical domain. The simulation is compared against experimental results. Future work will consider mapped multiblock grids for practical engineering geometries. Copyright © 2016 John Wiley & Sons, Ltd.

Received …

KEY WORDS: Generalized Curvilinear Coordinate Transformation, Fourth-Order Finite-Volume Method, Compressible Navier-Stokes Algorithm, Mapped Grids, Finite-Volume Method on Mapped Grids, Adaptive Mesh Refinement for Mapped Grids

1. INTRODUCTION

The past a few years have seen a great deal of research interest in high-order finite-volume methods on Cartesian grids [1, 2, 3, 5, 7, 9, 10, 11, 13, 14]. In particular, fluid flow problems involving turbulence and combustion that require long-time integration are candidates for high order methods (fourth-order and beyond). In addition, there is less impact of loss of accuracy at boundaries where mesh is not smooth, for example, resulting from applications of adaptive mesh refinement (AMR) or multiblock grids. AMR allows for the mesh resolution to change in response to the characteristics of the solution. Therefore, it is extremely advantageous for applications involving discontinuities, shock waves [15], space plasma physics [16], and combustion [17] or in any regions that require high resolution. Together with a high-order scheme, the AMR technique can enable full benefits of computational efficiency and accuracy. Finally, high-order methods increase the computation per unit memory which makes better use of modern and upcoming computer architectures.

Often, we need solve the fluid flow problems with complex geometries. One can resort to applying the finite-volume method directly to structured curvilinear meshes or unstructured meshes [18, 19], using cut-cell [20] or embedded boundary [21] techniques, or mapping [22] a structured grid in physical space to a Cartesian grid in computational space. In the present paper, we employ mapping

*Correspondence to: Xinfeng Gao at the Department of Mechanical Engineering of Colorado State University, Fort Collins, CO, 80523, USA. E-mail: Xinfeng.Gao@colostate.edu

Prepared using fldauth.cls *[Version: 2010/05/13 v2.00]*

to handle complex geometry. In doing so, we can exploit the computational efficiency of Cartesian methods, where finite-differencing operators are easily implemented and cell or face neighbors are easily known. Although the governing equations are complicated by the transformation with grid metrics, computation of smooth surfaces becomes efficient. Major drawback is grid generation but very complex geometries are still feasible. A previous fourth-order finite-volume method developed for Cartesian grids [7] was successfully applied to the case of mapped grids [9, 10] for solving time-dependent hyperbolic system of conservation laws. Notably, the method for computing the averages of the metric terms on faces was established such that freestream preservation is automatically satisfied. This was extended to support AMR by Guzik et al. [11].

In the present work, we build a new algorithm based on the methodologies verified in our previous work [11] for solving a time-dependent parabolic system of conservation laws. The present algorithm solves the full compressible Navier-Stokes equations, using a fourth-order finite-volume method and adaptive mesh refinement on mapped grids. Altogether it is a challenging task to ensure the solution maintains three characteristics: a freestream-preservation property, conservation, and fourth-order accuracy in smooth regions. Accordingly, the new strategies, already designed, implemented, and verified for the fourth-order hyperbolic operators, will be verified for the fourth-order viscous operators. In addition, new stencils for spatial discretizations will be designed and implemented for the fourth-order viscous operators. These strategies will be validated by solving the full Navier-Stokes equations. This paper will detail the strategies and demonstrate their validity with solutions of the unsteady Couette flow, a shock tube problem, and a Mach reflection problem. The solution accuracy of the viscous operators are verified with the transient Couette flow and the freestream preservation is confirmed on a smooth flow. Solutions of the Sod's shock tube are obtained to demonstrate the effectiveness of adaptive mesh refinement in resolving discontinuities. Finally, the Mach reflection problem demonstrates the utility of the mapped grids method on a non-rectangular physical domain. The overarching goal of our study is to model and simulate practical engineering problems, which nearly all involve complex geometries. More complex geometries, such as the example of a gas-turbine fan system shown in Figure 1, require a multiblock Cartesian space with non-trivial connectivity and mapped blocks [8]. Our future work involves extending the capabilities described herein so that they can be applied to the mapped multiblock grids shown in Figure 1.

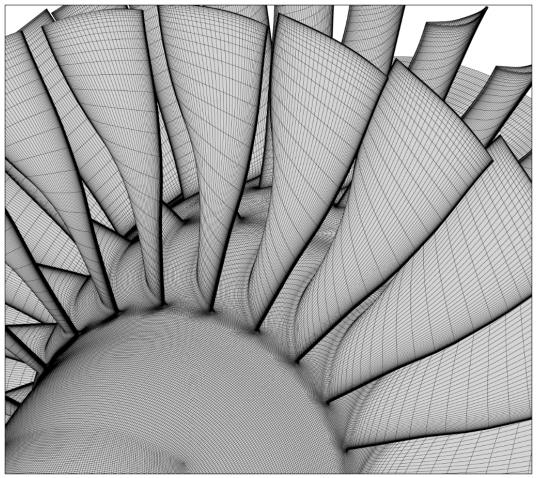

Figure 1. Meshing of a gas-turbine fan system with a mapped multiblock grid [12].

The paper is organized as follows. In Section 2, we first describe the mathematical modeling and finite-volume method for Navier-Stokes equations in Cartesian coordinates, and then modify the method for mapped grids. The numerical algorithm is given in Section 3, including stencil formulations, physical boundary conditions, time discretization, and stability constraints. Adaptive mesh refinement for mapped grids is described in Section 4. Results are presented in Section 5 with discussions focusing on solutions accuracy and freestream preservation properties. Finally, we draw conclusions in Section 6 where also, future work is proposed.

2. COMPUTATIONAL MODELING OF COMPRESSIBLE NAVIER-STOKES EQUATIONS

2.1. *Governing Equations and Finite-Volume Method in the Cartesian Coordinate System*

We describe the governing equations and the finite-volume method in a Cartesian coordinate system, since our algorithm discretizes the equations on a rectilinear coordinate system that is the base

DOI: 10.1002/fld

for both Cartesian and mapped grids. The system of governing equations including the continuity, momentum, and energy equations in the Cartesian coordinate system can be written in a conservative form as

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \left(\vec{\mathbf{F}} - \vec{\mathbf{G}} \right) = \mathbf{S} \,. \tag{1}$$

The solution vector, $\mathbf{U}$, of the conserved variables for the three-dimensional (3D) case is given by $[\rho,\, \rho u_x,\, \rho u_y,\, \rho u_z,\, \rho e]^{\mathrm{T}}$, where ρ is the density, u_x, u_y, and u_z are the component of velocity, $\boldsymbol{u}$, in each coordinate direction, and $e = |\boldsymbol{u}|^2/2 + h - p/\rho$ is the total specific energy with h being the enthalpy and p the pressure. The inviscid and viscous flux dyads, $\vec{\mathbf{F}}$ and $\vec{\mathbf{G}}$, are given in detail as

$$\vec{\mathbf{F}} = \left[\begin{bmatrix} \rho u_x \\ \rho u_x^2 + p \\ \rho u_x u_y \\ \rho u_x u_z \\ (\rho e + p) u_x \end{bmatrix}, \begin{bmatrix} \rho u_y \\ \rho u_y u_x \\ \rho u_y^2 + p \\ \rho u_y u_z \\ (\rho e + p) u_y \end{bmatrix}, \begin{bmatrix} \rho u_z \\ \rho u_z u_x \\ \rho u_z u_y \\ \rho u_z^2 + p \\ (\rho e + p) u_z \end{bmatrix} \right], \quad \text{and}$$

$$\vec{\mathbf{G}} = \begin{bmatrix} \begin{bmatrix} 0 & \tau_{xx} & \tau_{xy} & \tau_{xz} & u_x\tau_{xx} + u_y\tau_{xy} + u_z\tau_{xz} + \kappa \frac{\partial T}{\partial x} \end{bmatrix}^{\mathrm{T}}, \\ \begin{bmatrix} 0 & \tau_{yx} & \tau_{yy} & \tau_{yz} & u_x\tau_{yx} + u_y\tau_{yy} + u_z\tau_{yz} + \kappa \frac{\partial T}{\partial y} \end{bmatrix}^{\mathrm{T}}, \\ \begin{bmatrix} 0 & \tau_{zx} & \tau_{zy} & \tau_{zz} & u_x\tau_{zx} + u_y\tau_{zy} + u_z\tau_{zz} + \kappa \frac{\partial T}{\partial z} \end{bmatrix}^{\mathrm{T}} \end{bmatrix}, \quad \text{respectively.}$$

The source vector, $\mathbf{S}$, contains terms related to the body force and has the form of $\mathbf{S} = \left[0,\, \rho f_x,\, \rho f_y,\, \rho f_z,\, \rho \vec{f} \cdot \boldsymbol{u}\right]$. Additionally, $\vec{\nabla}$ is the vector differential operator. $\vec{\vec{\tau}}$ is the molecular stress tensor and $\vec{f}$ is the body force. The pressure is given by the ideal gas law. The molecular fluid stress tensor, is defined as

$$\vec{\vec{\tau}} = 2\mu(\vec{\vec{S}} - \frac{1}{3}\vec{\vec{I}}\,\vec{\nabla} \cdot \boldsymbol{u}) \,,$$

where μ is the molecular viscosity and $\vec{\vec{S}}$ is the strain rate tensor defined by $\left(\vec{\nabla}\boldsymbol{u} + (\vec{\nabla}\boldsymbol{u})^{\mathrm{T}}\right)/2$. The molecular heat flux is modeled using Fourier's law, where κ is thermal conductivity and T is the temperature.

The finite-volume approximation to the governing equations starts from the integral form as

$$\frac{\partial}{\partial t} \int_{\vec{x}(V_i)} \mathbf{U} \, \mathrm{d}\vec{x} + \int_{\vec{x}(V_i)} \left(\vec{\nabla} \cdot (\vec{\mathbf{F}} - \vec{\mathbf{G}}) - \mathbf{S} \right) \mathrm{d}\vec{x} = 0 \,.$$

We consider Cartesian-grid finite-volume methods where the cell centers are marked by the points $(i_0, ..., i_{D-1}) = \boldsymbol{i} \in \mathbb{Z}^D$ and the faces by $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$. Symbols, $D, \boldsymbol{i}, \boldsymbol{e}^d$, are the number of space dimensions, the integer vector for cell indices, and the unit vector in direction d, respectively. Representing the integrals as averages and applying the divergence theorem of Gauss, we arrive at a semi-discrete form

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle \mathbf{U} \rangle_{\boldsymbol{i}} = -\frac{1}{\Delta x_d} \sum_{d=1}^{D} \left(\left(\langle \vec{\mathbf{F}} \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \vec{\mathbf{F}} \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d} \right) - \left(\langle \vec{\mathbf{G}} \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \vec{\mathbf{G}} \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d} \right) \right) + \langle \mathbf{S} \rangle_{\boldsymbol{i}} \,, \tag{2}$$

with $\langle \bullet \rangle \equiv \frac{1}{V_{\boldsymbol{i}}} \int_{\vec{x}(V_i)} \bullet \, \mathrm{d}\vec{x}$ representing the cell-averaged or face-averaged quantities.

2.2. Governing Equations and Finite-Volume Method on Mapped Grids

In the present algorithm, we transform the physical space in which the mesh has curved, potentially non-orthogonal mesh lines into a computational space in which the mesh in rectilinear through a generalized curvilinear coordinate transformation. Such a transformation enables the straightforward application of our Cartesian algorithm and the treatment of more complex geometries.

Prepared using fldauth.cls
DOI: 10.1002/fld

Assume a smooth mapping from some abstract coordinate space into physical space $\vec{x} = \vec{x}(\vec{\xi})$, $\vec{x} : [0,1]^D \rightarrow \mathbb{R}^D$, where $\vec{x}$ denotes the physical space, eg. (x, y, z) for 3D in space, and $\vec{\xi}$ represents the computational space, eg. (ξ, η, ζ) for 3D in space. The divergence of a vector field, $\vec{\mathcal{F}}$, in physical space can be expressed in computational space as

$$\vec{\nabla}_x \cdot \vec{\mathcal{F}} = \frac{1}{J} \vec{\nabla}_\xi \cdot (\mathrm{N}^{\mathrm{T}} \vec{\mathcal{F}}) \,, \tag{3}$$

where J is the metrics Jacobian and N^{T} contains grid metrics. $\vec{\nabla}_x$ and $\vec{\nabla}_\xi$ are the vector differential operator in the physical and the computational space, respectively. Assuming the coordinates are time invariant, we obtain

$$J \equiv \det(\vec{\nabla}_\xi \vec{x}) \,, \quad \mathrm{N}^{\mathrm{T}} = J \, \vec{\nabla}_x \vec{\xi} \,, \quad \mathrm{N} = J \, (\vec{\nabla}_x \vec{\xi})^{\mathrm{T}} \,, \quad \text{and} \;\; \mathrm{N}^{\mathrm{T}}_{d,s} \equiv \det \left((\vec{\nabla}_\xi \vec{x})^{\mathrm{T}} (d | \boldsymbol{e}^s) \right) \,,$$

where $\mathrm{A}(d|\boldsymbol{e}^s)$ denotes a modification of matrix A by replacing row d with vector $\boldsymbol{e}^s$. Using above relations, Equation (1) is then transformed from physical space to computational space as

$$\frac{\partial}{\partial t} \int_{V_i} J\mathbf{U} \, \mathrm{d}\vec{\xi} + \int_{V_i} \vec{\nabla}_\xi \cdot (\mathrm{N}^{\mathrm{T}} (\vec{\mathbf{F}} - \vec{\boldsymbol{\mathcal{G}}})) \, \mathrm{d}\vec{\xi} = \int_{V_i} J\mathbf{S} \, \mathrm{d}\vec{\xi} \,,$$

where $\vec{\boldsymbol{\mathcal{G}}}$ is the mapped viscous flux dyad. Unlike the hyperbolic flux dyad ($\vec{\mathbf{F}}$), the viscous flux dyad ($\vec{\mathbf{G}}$) contains gradient fields for quantities such as velocity and temperature; consequently, the gradient fields must be transformed as well. The details are presented next, since the viscous operators are essential to the present algorithm solving the full Navier-Stokes equations. Following the same derivation procedure for Equation (2), the semi-discrete form on mapped grids is

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \langle J\mathbf{U} \rangle_{\boldsymbol{i}} \quad = \quad & -\frac{1}{\Delta\xi_d} \sum_{d=1}^{D} \Bigg(\left(\langle \mathrm{N}^{\mathrm{T}}_d \vec{\mathbf{F}} \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \mathrm{N}^{\mathrm{T}}_d \vec{\mathbf{F}} \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d} \right) \\ & - \left(\langle \mathrm{N}^{\mathrm{T}}_d \vec{\boldsymbol{\mathcal{G}}} \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \mathrm{N}^{\mathrm{T}}_d \vec{\boldsymbol{\mathcal{G}}} \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d} \right) \Bigg) + \langle J\mathbf{S} \rangle_{\boldsymbol{i}} \,, \end{aligned} \tag{4}$$

where the subscript d denotes the d^{th} row of N^{T}, $\langle \vec{\boldsymbol{\mathcal{G}}} \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ is the mapped viscous flux dyad at cell faces and is given by

$$\begin{bmatrix} \mathbf{0} \\ \langle \mu \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \left(\left\langle (\vec{\nabla}_\xi \boldsymbol{u}) \, \frac{\mathrm{N}^{\mathrm{T}}}{J} \right\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + \left\langle \left((\vec{\nabla}_\xi \boldsymbol{u}) \, \frac{\mathrm{N}^{\mathrm{T}}}{J} \right)^{\mathrm{T}} \right\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \frac{1}{3} I \left\langle J^{-1} \vec{\nabla}_\xi \cdot (\mathrm{N}^{\mathrm{T}} \boldsymbol{u}) \right\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \right) \\ \langle \vec{\vec{\mathcal{T}}} \cdot \boldsymbol{u} \rangle + \left\langle \kappa \frac{\mathrm{N}}{J} \vec{\nabla}_\xi T \right\rangle \end{bmatrix} , \tag{5}$$

where the bold $\mathbf{0}$ indicates a row vector with D components, each being zero, and $\vec{\vec{\mathcal{T}}}$ is the mapped stress tensor. The molecular fluid stress tensor is mapped as

$$\vec{\vec{\mathcal{T}}} = 2\mu \left(\vec{\vec{\mathcal{S}}} - \frac{1}{3} J^{-1} \vec{\vec{I}} \vec{\nabla}_\xi \cdot \left(\mathrm{N}^{\mathrm{T}} \boldsymbol{u} \right) \right) ,$$

where $\vec{\vec{\mathcal{S}}}$ is the strain rate tensor on mapped grids given by

$$\vec{\vec{\mathcal{S}}} = \frac{1}{2} \left((\vec{\nabla}_\xi \boldsymbol{u}) \, \frac{\mathrm{N}^{\mathrm{T}}}{J} + \left((\vec{\nabla}_\xi \boldsymbol{u}) \, \frac{\mathrm{N}^{\mathrm{T}}}{J} \right)^{\mathrm{T}} \right) .$$

In addition, the divergence of the velocity in physical space is transformed to the computational space using the relation given by Equation (3). The last row in Equation (5) presents the viscous flux

DOI: 10.1002/fld

vector in the energy equation, a vector with D elements. In this study, we assume fluid transport and thermodynamic properties, such as molecular viscosity, μ, and thermal conductivity, κ, are constant, and therefore, the average sign, $\langle\rangle$, is dropped for them from this point on.

Our algorithm by design is fourth order in both space and time. It is critical that we ensure the averages of $\langle \mathrm{N}_d^{\mathrm{T}} \vec{\mathbf{F}} \rangle$ and $\langle \mathrm{N}_d^{\mathrm{T}} \vec{\boldsymbol{\mathcal{G}}} \rangle$ on the faces in Equation (4) are fourth-order accurate. We employ a product rule of the form

$$\langle uv \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \langle u \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \langle v \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + \frac{\Delta\xi_d^2}{12} \sum_{d' \neq d} \frac{\partial u}{\partial \xi_{d'}} \frac{\partial v}{\partial \xi_{d'}} + O(\Delta\xi_d^4)\,, \tag{6}$$

which is valid on rectilinear grids and where fourth-order estimates of $\langle \vec{\mathbf{F}} \rangle$ and $\langle \vec{\mathbf{G}} \rangle$ or $\langle \vec{\boldsymbol{\mathcal{G}}} \rangle$ are known. In the present work, the spacing is uniform on the same grid level; for example, $\Delta\xi_d{}^2 = \Delta\xi_d \Delta\xi_{d'}$. For a single solution variable, the averages can be obtained from

$$\left\langle \mathrm{N}_d^{\mathrm{T}} \vec{\mathrm{F}} \right\rangle = \sum_{s=1}^{D} \langle \mathrm{N}_{d,s}^{\mathrm{T}} \rangle \langle \mathrm{F}_s \rangle + \frac{\Delta\xi_d^2}{12} \sum_{s=1}^{D} \sum_{d' \neq d} \frac{\partial \mathrm{N}_{d,s}^{\mathrm{T}}}{\partial \xi_{d'}} \frac{\partial \mathrm{F}_s}{\partial \xi_{d'}} + O(\Delta\xi_d^4)\,, \tag{7}$$

where $\vec{\mathrm{F}}$ is the flux vector for a scalar equation as an example. For the system of equations considered in our case, the product rule given by Equation (6) is applied to the dyad described by Equation (5) as an example to illustrate how the averages are evaluated. Nearly all terms in this equation involve the computation of the average of a product of two quantities, which can be scalar, vector, or matrix. Fourth-order spatial accuracy at a face, $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$, is required when computing, $\left\langle (\vec{\nabla}_\xi \boldsymbol{u}) \, \frac{\mathrm{N}^{\mathrm{T}}}{J} \right\rangle$, $\left\langle \left((\vec{\nabla}_\xi \boldsymbol{u}) \, \frac{\mathrm{N}^{\mathrm{T}}}{J} \right)^{\mathrm{T}} \right\rangle$, $\left\langle J^{-1} \vec{\nabla}_\xi \cdot (\mathrm{N}^{\mathrm{T}} \boldsymbol{u}) \right\rangle$, $\langle \vec{\vec{\mathcal{T}}} \cdot \boldsymbol{u} \rangle$, and $\langle \frac{\mathrm{N}}{J} \vec{\nabla}_\xi T \rangle$.

Following Equation (6) or (7), we evaluate $\langle (\vec{\nabla}_\xi \boldsymbol{u}) \frac{\mathrm{N}^{\mathrm{T}}}{J} \rangle$ on face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$ by

$$\left\langle \left(\vec{\nabla}_\xi \boldsymbol{u} \right) \frac{\mathrm{N}^{\mathrm{T}}}{J} \right\rangle = \langle \vec{\nabla}_\xi \boldsymbol{u} \rangle \left\langle \frac{\mathrm{N}^{\mathrm{T}}}{J} \right\rangle + \frac{\Delta\xi_d^2}{12} \sum_{d' \neq d} \frac{\partial \left(\vec{\nabla}_\xi \boldsymbol{u} \right)}{\partial \xi_{d'}} \frac{\partial \frac{\mathrm{N}^{\mathrm{T}}}{J}}{\partial \xi_{d'}} + O(\Delta\xi_d^4)\,. \tag{8}$$

Note that Equation (8) is a tensor of $D \times D$ and each entry of the left-hand side on the face can be expressed as

$$\langle \cdot \rangle_{rk} = \left\langle \sum_{s=1}^{D} \frac{\partial u_r}{\partial \xi_s} \frac{\mathrm{N}_{sk}^{\mathrm{T}}}{J} \right\rangle,$$

where r and k are indices for rows and columns, respectively. On the right-hand side (RHS) of the equation, the face-averaged quantity, $\langle \frac{\mathrm{N}^{\mathrm{T}}}{J} \rangle$, is evaluated by the same form of a product rule

$$\left\langle \frac{\mathrm{N}^{\mathrm{T}}}{J} \right\rangle = \langle \mathrm{N}^{\mathrm{T}} \rangle \langle J^{-1} \rangle + \frac{\Delta\xi_d^2}{12} \sum_{d' \neq d} \frac{\partial \mathrm{N}^{\mathrm{T}}}{\partial \xi_{d'}} \frac{\partial J^{-1}}{\partial \xi_{d'}} + O(\Delta\xi_d^4)\,. \tag{9}$$

The second term on the RHS of Equation (8) is written for multiple dimensions as

$$\frac{\Delta\xi_d^2}{12} \sum_{d' \neq d} \frac{\partial \left(\vec{\nabla}_\xi \boldsymbol{u} \right)}{\partial \xi_{d'}} \frac{\partial \frac{\mathrm{N}^{\mathrm{T}}}{J}}{\partial \xi_{d'}} \equiv \frac{\Delta\xi_d^2}{12} \sum_{s=1}^{D} \sum_{d' \neq d} \left(\frac{\partial \left(\frac{\partial u_r}{\partial \xi_s} \right)}{\partial \xi_{d'}} \frac{\partial \frac{\mathrm{N}_{sk}^{\mathrm{T}}}{J}}{\partial \xi_{d'}} \right). \tag{10}$$

Finally, each entry of Equation (8) is expressed and implemented for face, $\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}^d$, by

$$\left\langle \sum_{s=1}^{D} \frac{\partial u_r}{\partial \xi_s} \frac{\mathrm{N}_{sk}^{\mathrm{T}}}{J} \right\rangle \equiv \sum_{s=1}^{D} \left\langle \frac{\partial u_r}{\partial \xi_s} \right\rangle \left\langle \frac{\mathrm{N}_{sk}^{\mathrm{T}}}{J} \right\rangle + \frac{\Delta\xi_d^2}{12} \sum_{s=1}^{D} \sum_{d' \neq d} \left(\frac{\partial \left(\frac{\partial u_r}{\partial \xi_s} \right)}{\partial \xi_{d'}} \frac{\partial \frac{\mathrm{N}_{sk}^{\mathrm{T}}}{J}}{\partial \xi_{d'}} \right). \tag{11}$$

DOI: 10.1002/fld

It is worth mentioning that we compute the second term on the RHS of Equation (8) using the face averages instead of the face-centered values as shown in the equation. In doing so, we simplify the algorithm implementation but without destroying the spatial accuracy. The fourth-order accuracy of this term is maintained, since there is a second-order difference between the face-centered data and the face-averages, ($(\cdot)_{\boldsymbol{i}} = \langle\cdot\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + O(\Delta\xi_d^2)$), and multiplication by $\Delta\xi_d^2$, in the second term on the RHS of Equation (8) ensures fourth-order accuracy.

On face $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d$, the matrix multiplication term, $\left\langle \left((\vec{\nabla}_\xi \boldsymbol{u}) \frac{\mathrm{N}^\mathrm{T}}{J} \right)^\mathrm{T} \right\rangle$, is readily obtained by transposing $\left\langle (\vec{\nabla}_\xi \boldsymbol{u}) \frac{\mathrm{N}^\mathrm{T}}{J} \right\rangle$. The divergence term, $\left\langle \vec{\nabla}_\xi \cdot (\mathrm{N}^\mathrm{T}\boldsymbol{u}) \right\rangle$, is evaluated by

$$\left\langle \vec{\nabla}_\xi \cdot \left(\mathrm{N}^\mathrm{T}\boldsymbol{u}\right) \right\rangle = \left\langle \sum_{s=1}^{D} \left(\langle \boldsymbol{u}_s \rangle \sum_{d'=1}^{D} \frac{\partial \langle \mathrm{N}^\mathrm{T}_{d',s} \rangle}{\partial \xi_{d'}} \right) + \sum_{d'=1}^{D} \sum_{s=1}^{D} \left(\langle \mathrm{N}^\mathrm{T}_{d',s} \rangle \frac{\partial \langle \boldsymbol{u}_s \rangle}{\partial \xi_{d'}} \right) \right\rangle. \tag{12}$$

Further, using the fact that the columns of N^T are divergence free, we can drop the first term on the RHS of Equation (12) and the equation is simplified as

$$\left\langle \vec{\nabla}_\xi \cdot \left(\mathrm{N}^\mathrm{T}\boldsymbol{u}\right) \right\rangle = \sum_{s=1}^{D} \left(\langle \mathrm{N}^\mathrm{T}_{d'=d,s} \rangle \langle \frac{\partial u_s}{\partial \xi_d} \rangle + \sum_{d' \neq d} \langle \mathrm{N}^\mathrm{T}_{d',s} \rangle \langle \frac{\partial u_s}{\partial \xi_{d'}} \rangle \right), \tag{13}$$

where separating $d^{'} = d$ from $d^{'} \neq d$ makes it apparent whether the derivatives are transverse or normal to the direction of the face. This separation helps during code implementation. In the energy equation, terms related to the work done by the fluid stresses and the heat flux are fairly straightforward since the mapped stress tensor has been calculated. For example, having already computed

$$\langle \vec{\mathcal{T}} \rangle = \mu \left(\left\langle (\vec{\nabla}_\xi \boldsymbol{u}) \frac{\mathrm{N}^\mathrm{T}}{J} \right\rangle + \left\langle \left((\vec{\nabla}_\xi \boldsymbol{u}) \frac{\mathrm{N}^\mathrm{T}}{J} \right)^\mathrm{T} \right\rangle - \frac{1}{3} I \left\langle J^{-1} \vec{\nabla}_\xi \cdot (\mathrm{N}^\mathrm{T}\boldsymbol{u}) \right\rangle \right),$$

and $\langle \boldsymbol{u} \rangle$, we can readily compute $\langle \vec{\mathcal{T}} \cdot \boldsymbol{u} \rangle$. The heat flux is evaluated by $\langle \kappa \frac{\mathrm{N}}{J} \vec{\nabla}_\xi T \rangle$. Again, the fourth-order accuracy must be maintained whenever there is a computation of the average of a product of two quantities.

2.3. *Freestream Preserving*

Freestream preservation is an important requirement for the discretization of conservation laws in mapped coordinates. This property ensures that a uniform flow is unaffected by the choice of mapping and discretization. In particular, a uniform flow in physical space may appear with gradients when mapped to computational space; despite this, the discretization must be designed to preserve the uniform flow. Such a method has been designed and proven to be freestream preserving for hyperbolic system in previous work [9, 11]. Adopting this methodology, we will prove mathematically that the present algorithm is freestream preserving for solving the Navier-Stokes equations.

Rather than repeating the methodology in detail, which can be found in the work by Guzik et al. [11], we should provide the main concept herein. In that work, the freestream preservation was demonstrated using Equation (7); if the flux, $\vec{\mathrm{F}}$, is constant, then the second term vanishes and we only need to derive quadrature formulas for $\langle \mathrm{N}^\mathrm{T} \rangle$ so that the discrete divergence of a constant vector field is zero. Such quadratures exist due to Stokes' theorem and the Poincaré lemma. The columns of the matrix N^T are divergence-free as can be proven by a direct calculation. Representing the column vectors of N^T by the row vectors of N, denoted by N_s. The Poincaré lemma implies the existence of vectors $\vec{\mathcal{N}}_s$ such that, in three dimensions, $\mathrm{N}_s = \vec{\nabla}_\xi \times \vec{\mathcal{N}}_s$, and thus $\vec{\nabla}_\xi \cdot (\vec{\nabla}_\xi \times \vec{\mathcal{N}}_s) = 0$.

The above freestream condition holds true for a constant flux, whether $\vec{\mathrm{F}}$ is inviscid (hyperbolic) or viscous (elliptic). In the hyperbolic case, flux is a function of the flow variables, so a constant

DOI: 10.1002/fld

flow field will result in a constant flux. While in the elliptic case, the viscous flux is a function of both the flow variable and their gradients, we must verify that the resulting flux remains constant for a given uniform flow field. Examine Equation (5) and take the momentum flux as an example. Assuming the velocity field is uniform, $\vec{\nabla}_\xi \boldsymbol{u} = 0$ is true and as a consequence, the first two terms, $\left\langle (\vec{\nabla}_\xi \boldsymbol{u})\ \frac{\mathrm{N}^\mathrm{T}}{J} \right\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ and $\left\langle \left((\vec{\nabla}_\xi \boldsymbol{u})\ \frac{\mathrm{N}^\mathrm{T}}{J} \right)^\mathrm{T} \right\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ vanish. The flux only contains $-\frac{1}{3} I \left\langle J^{-1} \vec{\nabla}_\xi \cdot (\mathrm{N}^\mathrm{T} \boldsymbol{u}) \right\rangle$, and what matters is $\vec{\nabla}_\xi \cdot (\mathrm{N}^\mathrm{T} \boldsymbol{u})$. From Equation (13), it can be seen $\vec{\nabla}_\xi \cdot (\mathrm{N}^\mathrm{T} \boldsymbol{u}) = 0$ for a uniform velocity field. Therefore, the viscous flux is zero for a constant velocity flow field. Further, we will provide numerical verification of the freestream preservation in Section 5.

3. DISCRETIZATION SCHEMES

3.1. Interior Stencils for Computing Gradients and Transverse Derivatives

Gradients of the solution variables must be computed for cells and faces, since they are required for evaluating viscous fluxes. For hyperbolic flux evaluation, gradients are also needed by the high-order reconstruction process, including within limiters and dissipation mechanisms for strong shocks. Previous works [7, 10, 11] contain details on the inviscid flux evaluation. For conciseness, the procedures are not repeated herein, but interested readers can consult the references for more information. For elliptic flux evaluation, the viscous flux is a function of $\vec{\boldsymbol{\mathcal{G}}}\left(\langle \mu \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}, \langle w \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}, \langle \vec{\nabla}_\xi w \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \right)$, where w is the primitive variable, such as velocity or temperature. Consequently, the gradient fields must be calculated first in order to evaluate fluxes. Next, we present the interior stencils for gradient computation. For convenience, the illustration is shown in 2D using velocity as an example; however, extension to 3D is straightforward.

For the interior cells, a fourth-order center-differencing method was developed by Gao et al. [23]. Namely, the face-averaged normal gradient is computed from cell-averaged velocities, while the face-averaged tangent gradient is calculated from cell-averaged gradients; they are given by Equations (14a) and (14b), respectively

$$\left\langle \frac{\partial w}{\partial \xi_d} \right\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^{d'}} \begin{cases} \dfrac{1}{12\Delta\xi_d} \left(15\langle w \rangle_{\boldsymbol{i}+\boldsymbol{e}^{d'}} - 15\langle w \rangle_{\boldsymbol{i}} - \langle w \rangle_{\boldsymbol{i}+2\boldsymbol{e}^{d'}} + \langle w \rangle_{\boldsymbol{i}-\boldsymbol{e}^{d'}} \right) & d = d' \quad (14a) \\ \dfrac{7}{12} \left(\left\langle \dfrac{\partial w}{\partial \xi_d} \right\rangle_{\boldsymbol{i}} + \left\langle \dfrac{\partial w}{\partial \xi_d} \right\rangle_{\boldsymbol{i}+\boldsymbol{e}^{d'}} \right) - \dfrac{1}{12} \left(\left\langle \dfrac{\partial w}{\partial \xi_d} \right\rangle_{\boldsymbol{i}-\boldsymbol{e}^{d'}} + \left\langle \dfrac{\partial w}{\partial \xi_d} \right\rangle_{\boldsymbol{i}+2\boldsymbol{e}^{d'}} \right) & d \neq d' \quad (14b) \end{cases}$$

where ξ_d is the d^{th} component of $\vec{\xi}$ and $\Delta\xi_d$ is the spacing in d direction and commonly has a value of unity on the base grid. In Equation (14b), the cell-averaged gradients are formulated by the fourth-order stencil

$$\left\langle \frac{\partial w}{\partial \xi_d} \right\rangle_{\boldsymbol{i}} = \frac{1}{12\Delta\xi_d} \left(8\langle w \rangle_{\boldsymbol{i}+\boldsymbol{e}^d} - 8\langle w \rangle_{\boldsymbol{i}-\boldsymbol{e}^d} - \langle w \rangle_{\boldsymbol{i}+2\boldsymbol{e}^d} + \langle w \rangle_{\boldsymbol{i}-2\boldsymbol{e}^d} \right). \quad (15)$$

Note for a given face, the normal gradient parallels the face normal, while the tangent gradient is orthogonal to the face normal in the computational space.

Often, for the viscous operators, we need to compute transverse derivatives of quantities ($\boldsymbol{u}$, T, $\vec{\nabla}_\xi \boldsymbol{u}$, or $\vec{\nabla}_\xi T$). For example, a second-order accurate approximation of $\frac{\partial \left(\vec{\nabla}_\xi \boldsymbol{u} \right)}{\partial \xi_{d'}}$ is required in the second term on the RHS of Equation (8). This can be achieved by simply performing center differencing and Figure 2 shows the fourth-order stencil for this computation. As a reminder, for finite-volume methods, we start with the known knowledge of cell-averages of solution variables, and with this information, the face- and cell-averaged gradients of solution variables are now known from Equations (14a), (14b), and (15).

DOI: 10.1002/fld

At this point, information of $\langle w\rangle_{\boldsymbol{i}}$, $\langle\vec{\nabla}_\xi w\rangle_{\boldsymbol{i}}$, $\langle w\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$, and $\langle\vec{\nabla}_\xi w\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ is known, with w being the velocity component in each coordinate direction. To find the second-order tangential derivative of the face-averaged values at location $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^1$, marked with a filled diamond in Figure 2, a center differencing is performed

$$\left.\frac{\partial(\frac{\partial w}{\partial\xi})}{\partial\xi}\right|_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^1} = \frac{\langle\frac{\partial w}{\partial\xi}\rangle_{\boldsymbol{i}+\boldsymbol{e}^0+\frac{1}{2}\boldsymbol{e}^1} - \langle\frac{\partial w}{\partial\xi}\rangle_{\boldsymbol{i}-\boldsymbol{e}^0+\frac{1}{2}\boldsymbol{e}^1}}{2\Delta\xi}, \tag{16}$$

where $\langle\frac{\partial w}{\partial\xi}\rangle_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}^1}$ are used in the stencil with $\boldsymbol{j}\in[\boldsymbol{i}-1,\boldsymbol{i}+1]$. Additionally, in this calculation we use face-averages instead of face-centered values. In doing so, we can simplify the algorithm implementation while still maintaining fourth-order spatial accuracy. Similarly, for a ξ-face, interior stencils can be formulated for computing the gradients of the solution variables. In like manner, $\frac{\partial(\mathrm{N}^{\mathrm{T}}/J)}{\partial\xi_d}$ in Equation (8) is calculated at the $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^{d'}$ face.

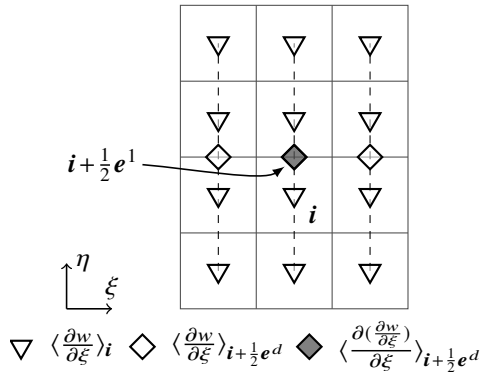

Figure 2. The fourth-order stencil for computing gradient fields in interior cells.

In summary, having obtained the gradient fields of the solution variables and the grid metrics on faces, the fluxes can be evaluated using the relations described in Section 2.2.

3.2. Boundary Schemes for the Gradients and the Tangent Derivatives

It is worth mentioning that in our algorithm we employ two layers of ghost cells outside the computational domain to facilitate the stencil calculations. Consequently, the centered stencils as described above are used to compute the cell-averaged gradients everywhere within the computational domain. Also, the face-averaged gradients are computed using center schemes everywhere, including both the interior faces and the physical boundaries [23]. In ghost cells orthogonal to a d direction domain boundary, gradients are required only in the $d\neq d'$ directions, and interested readers can consult the same reference for detailed stencil calculations.

However, the tangent derivative to a η-face as in Equation (16) must be modified from the center scheme to the one-sided scheme for faces orthogonal to the physical boundary. For example, Figure 3 shows such a face, on which a filled diamond is marked. This example is shown for the case where one physical boundary is present at the domain extent in the ξ direction. Nonetheless, the concept is the same for cases where one physical boundary presents at the domain extent in other coordinate directions. When multiple physical boundaries intersect, transverse derivatives on faces orthogonal to any physical boundary are computed using one-sided schemes. Specifically, for the face as shown in Figure 3, Equation (16) is modified as

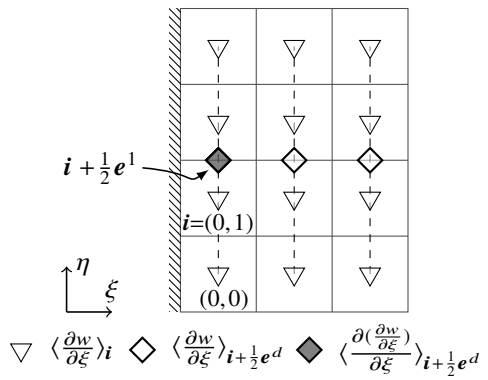

Figure 3. One-sided tangential gradients on an adjacent face, normal to the physical boundary.

$$\left.\frac{\partial(\frac{\partial w}{\partial\xi})}{\partial\xi}\right|_{\mathbf{0},\frac{1}{2}} = \frac{1}{2\Delta\xi}\left(-\frac{3}{2}\langle\frac{\partial w}{\partial\xi}\rangle\Big|_{0,\frac{1}{2}} + 2\langle\frac{\partial w}{\partial\xi}\rangle\Big|_{1,\frac{1}{2}} - \frac{1}{2}\langle\frac{\partial w}{\partial\xi}\rangle\Big|_{2,\frac{1}{2}}\right).$$

Needless to say, the computation of temperature or pressure gradients and their tangent derivatives should follow the same formulas and methodology. We will not repeat the solution procedure here.

DOI: 10.1002/fld

3.3. Physical Boundary Conditions

At a body surface, no-slip condition must be satisfied for viscous flow. For clarity, assume that body surfaces are mapped to η = constant coordinates. Define a vector $\vec{r} = x\vec{i} + y\vec{j} + z\vec{k}$. Given $\vec{\xi}(\vec{x})$, the covariant basis vectors are tangent to the ξ, η, and ζ axes and given by

$$\vec{e}_1 = \frac{\frac{\partial \vec{r}}{\partial \xi}}{|\frac{\partial \vec{r}}{\partial \xi}|}, \quad \vec{e}_2 = \frac{\frac{\partial \vec{r}}{\partial \eta}}{|\frac{\partial \vec{r}}{\partial \eta}|}, \text{ and } \quad \vec{e}_3 = \frac{\frac{\partial \vec{r}}{\partial \zeta}}{|\frac{\partial \vec{r}}{\partial \zeta}|}. \tag{17}$$

The contravariant basis vectors are then defined

$$\vec{E}^1 = \frac{\vec{e}_2 \otimes \vec{e}_3}{\vec{e}_1 \cdot (\vec{e}_2 \otimes \vec{e}_3)}, \quad \vec{E}^2 = \frac{\vec{e}_3 \otimes \vec{e}_1}{\vec{e}_2 \cdot (\vec{e}_3 \otimes \vec{e}_1)}, \text{ and } \quad \vec{E}^3 = \frac{\vec{e}_1 \otimes \vec{e}_2}{\vec{e}_3 \cdot (\vec{e}_1 \otimes \vec{e}_2)}. \tag{18}$$

The covariant and contravariant basis vectors are illustrated in Figure 4 for general curvilinear coordinate systems. Here, the coordinates (ξ, η, ζ) are shown as orthogonal, although this is not necessary. In the figure, the gray planes are the coordinate surfaces; for example, $\vec{E}^1$ is orthogonal to the coordinate surface, i.e., the η-ζ surface or the ξ-plane.

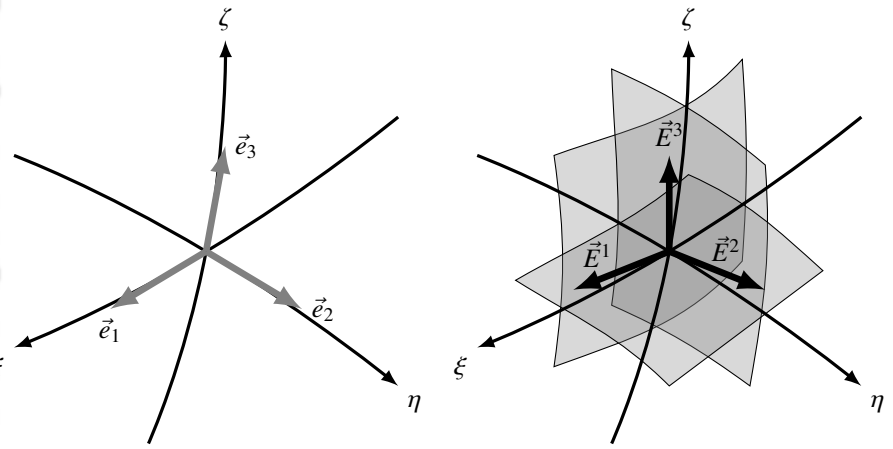

Figure 4. The covariant basis vectors, $\vec{e}_i$, and the contravariant basis vectors, $\vec{E}^i$, are shown for a curvilinear coordinate system, where $i = 1, \cdots, D$.

Therefore, we can obtain the contravariant components of the velocity by the dot product of $\boldsymbol{u}$ with the contravariant basis vectors:

$$U = \boldsymbol{u} \cdot \vec{E}^1, \quad V = \boldsymbol{u} \cdot \vec{E}^2, \text{ and } \quad W = \boldsymbol{u} \cdot \vec{E}^3.$$

In the application of boundary conditions, one often needs expressions for the velocity components normal and tangential to the boundary in terms of the Cartesian velocity components. In this case, we need to preserve the magnitude of the velocity. Recall that the body surface follows a grid surface of constant η, and $\vec{e}_1$ and $\vec{e}_3$ are tangent to the body surface, while $\vec{E}^2$ is normal to the body surface. Therefore, we can write

$$u_x\vec{i} + u_y\vec{j} + u_z\vec{k} = u_{t_\xi}\vec{e}_1 + u_{n_\eta}\vec{E}^2 + u_{t_\zeta}\vec{e}_3,$$

where u_{t_ξ} and u_{t_ζ} are the tangential velocity components and u_{n_η} is the normal velocity component. Using the metric relations, $\vec{e}_1 \cdot \vec{E}^2 = 0$, $\vec{e}_3 \cdot \vec{E}^2 = 0$, and $\vec{e}_1 \cdot \vec{e}_1 = \vec{E}^2 \cdot \vec{E}^2 = \vec{e}_3 \cdot \vec{e}_3 = 1$, we can express the unit vectors in grid metrics. With the covariant and contravariant unit vectors (See Appendix 8), we can easily correlate the tangential and normal velocity components for solving the Cartesian velocity components at the body surface, which are required by the algorithm.

For an inviscid flow, the normal velocity, u_{n_η}, is set to zero, u_{t_ξ} and u_{t_ζ} are determined from an extrapolation. For a viscous flow, the no-slip condition gives $u_x = u_y = u_z = 0$. There is an

DOI: 10.1002/fld

additional boundary condition related to heat transfer that determines the temperature or its gradient normal to the surface. If the wall remains at a constant temperature, then this temperature must be specified. More commonly, an adiabatic condition is appropriate. In this case, there is no heat transfer to or from the wall, giving $\frac{\partial T}{\partial n_\eta} = 0$. When a heat flux is defined for the wall, the wall temperature can be determined by extrapolation from the interior. The gradient of temperature at the surface can be expressed in terms of the basis vectors $\vec{e}_1$, $\vec{E}^2$, and $\vec{e}_3$ as follows:

$$\vec{\nabla} T = \frac{\partial T}{\partial x}\,\vec{i} + \frac{\partial T}{\partial y}\,\vec{j} + \frac{\partial T}{\partial z}\,\vec{k} = \frac{\partial T}{\partial t_\xi}\,\vec{e}_1 + \frac{\partial T}{\partial n_\eta}\,\vec{E}^2 + \frac{\partial T}{\partial t_\zeta}\,\vec{e}_3\,.$$

Note that t_ξ and n_ξ represent the tangent and normal direction to ξ, respectively. For example, $\frac{\partial T}{\partial n_\eta}$ can be expressed in terms of $\vec{\nabla}_x T$ using the the relation

$$\frac{\partial T}{\partial n_\eta} = \vec{E}^2 \cdot \vec{\nabla} T = \frac{\eta_x \frac{\partial T}{\partial x} + \eta_y \frac{\partial T}{\partial y} + \eta_z \frac{\partial T}{\partial z}}{\sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2}}\,.$$

Similarly, $\frac{\partial T}{\partial t_\xi}$ and $\frac{\partial T}{\partial t_\zeta}$ can be correlated to $\vec{\nabla}_x T$ through the relations, $\vec{e}_1 \cdot \vec{\nabla} T$ and $\vec{e}_3 \cdot \vec{\nabla} T$, respectively. The Cartesian temperature gradient at the body surface can be computed by

$$\left[\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}, \frac{\partial T}{\partial z}\right]^{\mathrm{T}} = A \left[\frac{\partial T}{\partial t_\xi}, \frac{\partial T}{\partial t_\zeta}, \frac{\partial T}{\partial n_\eta}\right]^{\mathrm{T}},$$

where A is provided in Appendix 8. If the wall pressure needs to be determined, the same procedure is applied. The conservative variables can then be found from the values of $\boldsymbol{u}$, T, and p.

3.4. Time Discretization and Stability Constraints

In order to evolve the solution in time, we discretize the semi-discrete system, Equation (4), using the standard explicit, four-stage, fourth-order Runge-Kutta scheme. Stability constraints are analyzed and derived for the hyperbolic system on mapped grids by Colella et al. [9] and for the linear convection-diffusion equation on Cartesian grids by Gao et al. [23]. Next, we complete the stability constraints based on previous works for the full Navier-Stokes equations on mapped grids.

Colella et al. [9] found the constraint of purely first-order upwind fluxes appears to be the most severe among several combinations of low- and higher-order fluxes. The stability constraint is

$$\frac{\Delta t}{\Delta x} \sum_{d=1}^{D} \mid \vec{\vec{\lambda}}_{\boldsymbol{x}} \cdot \boldsymbol{e}^d \mid \leq 1.3925\,,$$

where $\vec{\vec{\lambda}}_{\boldsymbol{x}}$ is the tensor of characteristic speeds in physical space, and for example, $\vec{\vec{\lambda}}_{\boldsymbol{x}}$ in 3D is $\vec{\vec{\lambda}}_{\boldsymbol{x}} = [\vec{\lambda}_x, \vec{\lambda}_y, \vec{\lambda}_z]^{\mathrm{T}}$. The size of each vector equals to the number of governing equations. Naturally, the stability constraint is extended to the mapped grids by transforming $\vec{\vec{\lambda}}_{\boldsymbol{x}}$ to

$$\frac{\Delta t}{\Delta \xi} \sum_{d=1}^{D} \mid \vec{\vec{\lambda}}_{\boldsymbol{\xi}} \cdot \boldsymbol{e}^d \mid \leq 1.3925\,,$$

where $\vec{\vec{\lambda}}_{\boldsymbol{\xi}} = \frac{\mathrm{N}^{\mathrm{T}}}{J} \vec{\vec{\lambda}}_{\boldsymbol{x}}$. For the elliptic component, Gao et al. [23] derived the stability limit as

$$\frac{\nu \Delta t}{\Delta x^2} \sum_{d=1}^{D} \mid \vec{\vec{\lambda}}_{\boldsymbol{x}} \cdot \boldsymbol{e}^d \mid \leq 2.5$$

where ν is the kinematic viscosity and $\frac{\nu \Delta t}{\Delta x^2}$ is the Von Neumann number. Similarly, we extend the stability constraint to mapped grids, and it takes the form of

$$\frac{\nu \Delta t}{\Delta \xi^2} \sum_{d=1}^{D} \mid \vec{\vec{\lambda}}_{\boldsymbol{\xi}} \cdot \boldsymbol{e}^d \mid \leq 2.5\,.$$

Prepared using fldauth.cls
DOI: 10.1002/fld

We recommend a conservative time step given by

$$\frac{1}{\Delta t} = \frac{1}{\Delta t_{\text{hyperbolic}}} + \frac{1}{\Delta t_{\text{elliptic}}}, \tag{19}$$

where $\Delta t_{\text{hyperbolic}}$ and $\Delta t_{\text{elliptic}}$ are the time steps constrained by the stability conditions by the hyperbolic and elliptic components, respectively. Note that the stability analysis was performed on a linear OΔE's with periodic boundary conditions. It serves a reasonably reliable necessary stability condition, but it is by no means a sufficient one.

4. ADAPTIVE MESH REFINEMENT FOR MAPPED GRIDS

4.1. Properly Nested Adaptive Mesh Refinement

AMR calculations are performed on a hierarchy of nested meshes denoted by Ω^l for grid level l as indicated in Figure 5. Each mesh can be decomposed into a disjoint union of rectangles in order to perform calculations in parallel. To implement adaptive mesh refinement, we make use of the Chombo library for parallel AMR [24]. Although we follow the strategies used therein, certain data operations need to be introduced for specific algorithms. For completeness, a few major elements are provided next.

In particular, the control volume corresponding to a cell in Ω^{l-1} is either completely contained in the control volumes defined by Ω^l or its intersection has zero volume. We assume that there are a sufficient number of cells on level ℓ separating the level $\ell+1$ cells from the level $\ell-1$ cells. Grid hierarchies that meet these conditions are referred as being properly nested. Two boxes are shown on the finer grid levels in Figure 5 and are outlined with thick lines. Any relationships between boxes on the same level, or between different levels, are known simply through the vectors describing the corner locations on the integer lattice. Consequently, there is no need for tracking connectivity between boxes.

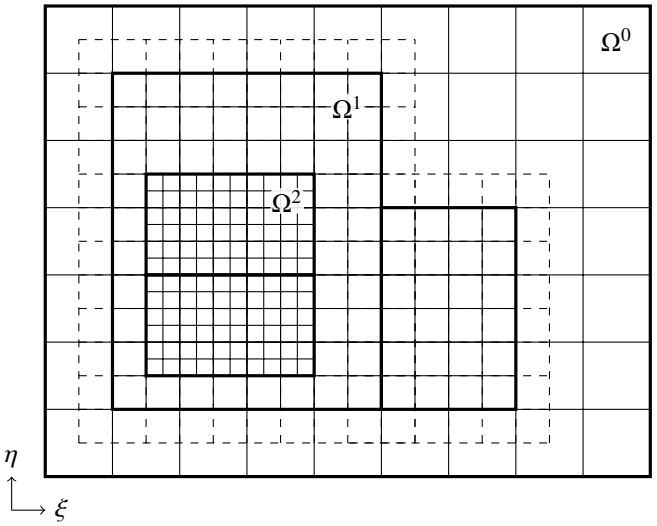

Figure 5. A three-level grid with $n_{ref} = 2$ and nesting sufficient for one cell to separate level $\ell+1$ from level $\ell-1$. A single layer of invalid ghost cells surrounding the middle grid level is shown by dashed lines.

There exists a rich literature on AMR [15, 25]. We do not attempt to duplicate material which is thoroughly covered in the literature. Rather, herein we focus on describing the typical work-flow for advancing level ℓ and stress some important issues in the context of mapped grids. As mentioned earlier, the grid Ω^ℓ on level ℓ is partitioned into *boxes* that are distributed among processors to perform calculations in parallel. In order to keep the calculations independent, ghost cells surround each box. Three types of ghost cells are defined:

1. A valid ghost is a ghost cell within the valid region of Ω^ℓ. These result from partitioning and are filled by means of exchange.
2. An invalid ghost is a ghost cell outside the valid region of Ω^ℓ. These result from grid refinement and are used to couple a coarser grid level to level l. They are filled by interpolation from the coarser level.
3. A domain ghost is a ghost cell outside the valid computational domain region. They are used to enforce boundary conditions.

DOI: 10.1002/fld

In the present mapped algorithm, 7 layers of ghost cells of $\langle \mathbf{U} \rangle$ are required to compute the flux in computational space for viscous flows. Fewer ghosts cells are needed for the metric terms and $\langle J\mathbf{U} \rangle$. A summary of the ghost cells at an AMR interface is presented in Figure 6 and the ghost cells are distinguished by different shades. In general, 1 layer of ghost cells is required for $\langle J \rangle$ and $\langle J\mathbf{U} \rangle$ in order to compute the gradients,

$$\frac{\partial \bar{\mathbf{U}}}{\partial \xi_d} \text{ and } \frac{\partial \langle J \rangle}{\partial \xi_d},$$

everywhere using a centered stencil. The gradients will be used later in a cell-centered product rule. Note that $\bar{\mathbf{U}}$ is cell-centered values. As $\langle J \rangle$ itself is computed using a product formula involving $\langle \mathrm{N}^T \rangle$, the latter is required on the faces of two layers of ghost cells. Note that $\vec{\mathcal{N}}_s$ is only required on the same ghosts as $\langle \mathrm{N}^{\mathrm{T}} \rangle$ because it is one codimension higher. Although $\langle \mathrm{N}^{\mathrm{T}} \rangle$ is also one codimension higher than $\langle J \rangle$, it is needed on an extra ghost in order to compute the volume flux, $\langle \mathrm{N}^{\mathrm{T}} \vec{\chi} \rangle$, via a product rule, which is then used to compute $\langle J \rangle$. Here, $\vec{\chi}$ is defined by $\vec{\chi}(\vec{\xi}) = \vec{x}(\vec{\xi})/D$ (see Guzik et al. [10]).

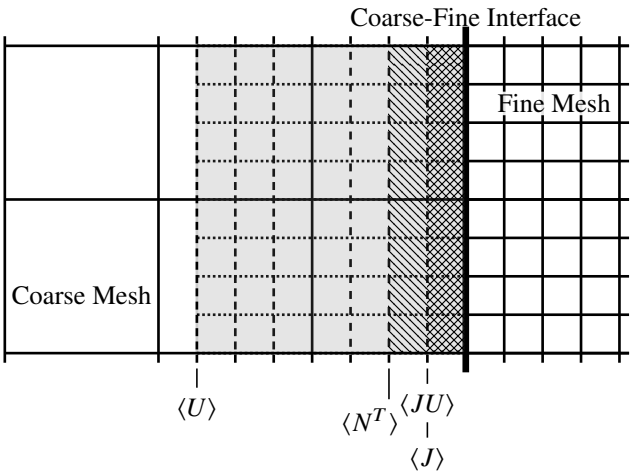

Figure 6. Extent of invalid ghost cells at an AMR interface for several solution variables and grid metrics.

The calculation of $\langle \mathbf{U} \rangle$ in valid cells is preceded by an exchange of $\langle J\mathbf{U} \rangle$ so that it is available in one layer of valid ghost cells. Equation (20) is applied to obtain $\langle \mathbf{U} \rangle$ in valid cells. A second exchange, this time of $\langle \mathbf{U} \rangle$, is used to fill all seven layers of valid ghost cells. The filling of invalid ghosts is achieved through interpolation from the coarser level. The interpolator must fill $\langle \mathbf{U} \rangle$ in the seven ghost cells and obtain $\langle J\mathbf{U} \rangle$ in one ghost cell adjacent to the interface. From the invalid ghost cells, $\langle \mathbf{U} \rangle$ is only used to help compute the flux in the valid cells, and $\langle J\mathbf{U} \rangle$ is only used to compute $\bar{\mathbf{U}}$, so that a centered difference of $\bar{\mathbf{U}}$ can be used in Equation (20) to compute $\langle \mathbf{U} \rangle$ in the valid cells adjacent to the interface.

The typical work-flow for advancing level ℓ is:

1. Regrid levels finer than ℓ if required. This involves tagging all cells which should compose the finer levels, often based on the magnitudes of solution gradients, and constructing a new, properly nested mesh hierarchy. In regions where new fine cells appear, the solution is interpolated from the coarser level. We organize the regrid process into three steps:
   ① Generate the geometric information for the new grid hierarchy.
   ② Adjust the values of the solution on the old hierarchy to be consistent with the new grid geometry. This step is critical for ensuring conservation of the solution.
   ③ Construct the solution on the new grid hierarchy by constrained interpolation.
2. Advance level ℓ. The solution state is first computed using a cell-centered product rule (a rearrangement of Equation (6)),

$$\langle \mathbf{U} \rangle = \frac{1}{\langle J \rangle} \left(\langle J\mathbf{U} \rangle - \frac{{\Delta\xi_d}^2}{12} \sum_d^D \frac{\partial \bar{\mathbf{U}}}{\partial \xi_d} \frac{\partial J}{\partial \xi_d} \right). \tag{20}$$

   Note that only second-order accuracy is required for the gradients. Therefore, $\bar{\mathbf{U}}$ can be obtained from $\langle J\mathbf{U} \rangle / \langle J \rangle$ and gradients of J can be evaluated using $\langle J \rangle$.
3. Interpolate to the invalid ghost cells surrounding level $\ell + 1$. Interpolation in time requires a careful tuning to the Runge-Kutta advancement on the coarser level [7]. In space, a least-squares algorithm is used to compute a conservative interpolating polynomial in each coarse cell.

DOI: 10.1002/fld

4. Start level $\ell+1$ at step 1. Level $\ell+1$ is refined in time (sub-cycled) with a time step $\Delta t^{\ell+1} = \Delta t^{\ell}/n^{\ell}_{ref}$.
5. Average the solution from $\ell+1$ to underlying cells on level ℓ and correct fluxes at coarse-fine interfaces to ensure conservation [26, 15]. In the latter correction, the fluxes computed on the coarse grid are replaced by fluxes computed on the fine grid at the interface between the two grid levels.

5. RESULTS AND DISCUSSIONS

We now discuss the mapping relation used for the computational grid generation and present solutions of two transient viscous flow problems on the mapped grids. In particular, the results demonstrate that the freestream preservation is maintained and verify that the solution is fourth-order accurate. Additionally, contours and profiles of the solution variables are provided.

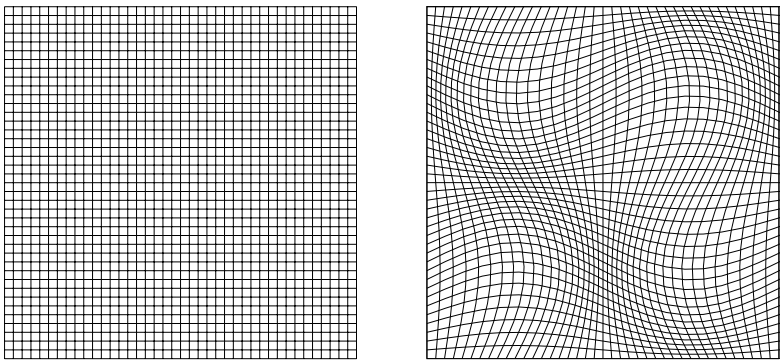

Figure 7. A single level grid on a xy-plane in the computational and physical space, respectively.

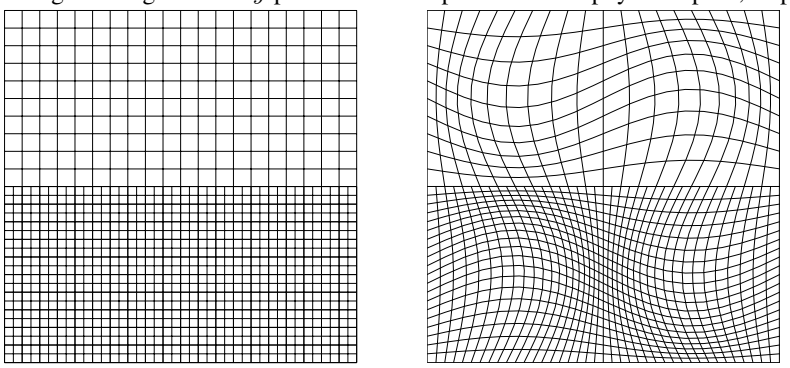

Figure 8. A two-level AMR grid on a xy-plane in the computational and physical space, respectively. The fine level has the same spatial resolution as that of the single grid in Figure 7.

5.1. Mapping

This study employs a nonlinear mapping, a specialization of the mapping used in Persson et al. [28]. This mapping is generated by perturbing a uniform Cartesian mesh using

$$x_d = \xi_d + c_d \prod_{p=1}^{D} \sin(2\pi\xi_p), \quad d = 1, ..., D. \tag{21}$$

To ensure that the perturbed mesh does not tangle, it is sufficient to take $\forall d,\ 0 \leq 2\pi c_d \leq 1$. We use $c_d = 0.1$. Figure 7 illustrates patches of the computational and physical meshes generated using the above analytical relation, while Figure 8 shows the meshes with a level of refinement. This mapping is used for all our calculations in this paper.

DOI: 10.1002/fld

5.2. Conservation and Freestream Preservation

To test conservation and freestream preservation of the algorithm, a two-dimensional case was run with initial velocity in both x and y directions of 0.2 m/s and periodic boundary conditions for all the boundaries. The physical domain is a 1 m by 1 m square domain with 256 by 256 cells. The mesh was warped according to the warped meshing method described in Equation (21). The case was run for 15,000 time steps with a time step size of 3.433×10^{-6}. The initial and final conservative quantities for the flow are tabulated in Table I, indicating the conservative quantities are indeed conserved. Moreover, we computed L_1-norm for the difference of the conservative quantity and the norms are all close to machine zero. This verifies that the freestream preservation is maintained.

Table I. Initial and final conservative quantities for constant periodic flow problem

$J\mathbf{U}$	Initial	Final	Difference
$J\rho$	8.028160000000258e+04	8.028160000000261e+04	0.000000000000003e+04
$J\rho u$	1.605631999999906e+04	1.605631999999990e+04	0.000000000000083e+04
$J\rho v$	1.605631999999906e+04	1.605631999999985e+04	0.000000000000078e+04
$J\rho E$	1.659803494809955e+10	1.659803494809955e+10	0.000000000000000e+10

5.3. Fourth-Order Solution Accuracy

We apply our algorithm to solve a transient Couette flow and compare the numerical solution to the exact analytical solution for a numerical error convergence study. The analytical transient solution of a Couette flow between two infinite plates may be found in the texts by Batchelor [29], Sherman [30], or White [31], and is often in the form [32]

$$\frac{u(y,t)}{U_0} = \left\{(1-\frac{y}{H})-2\sum_{n=1}^{\infty}\frac{\sin(n\pi y/H)}{n\pi}e^{-\frac{n^2\pi^2\nu t}{H^2}}\right\},$$

where H is the channel height, ν is the kinematic viscosity, U_0 is the moving velocity of the bottom wall, n is an integer, y is the location normal to the wall direction, and t is time. In this paper, the fluid is air at 80°C, with a kinematic viscosity of 2.09e-05 m^2/s and a density of 1.0 kg/m^3. The channel height is 0.1m and the moving wall velocity is 0.0418 m/s. The Reynolds number is 200 based on the channel height and the moving wall velocity. Boundary conditions on x- and z-direction are periodic.

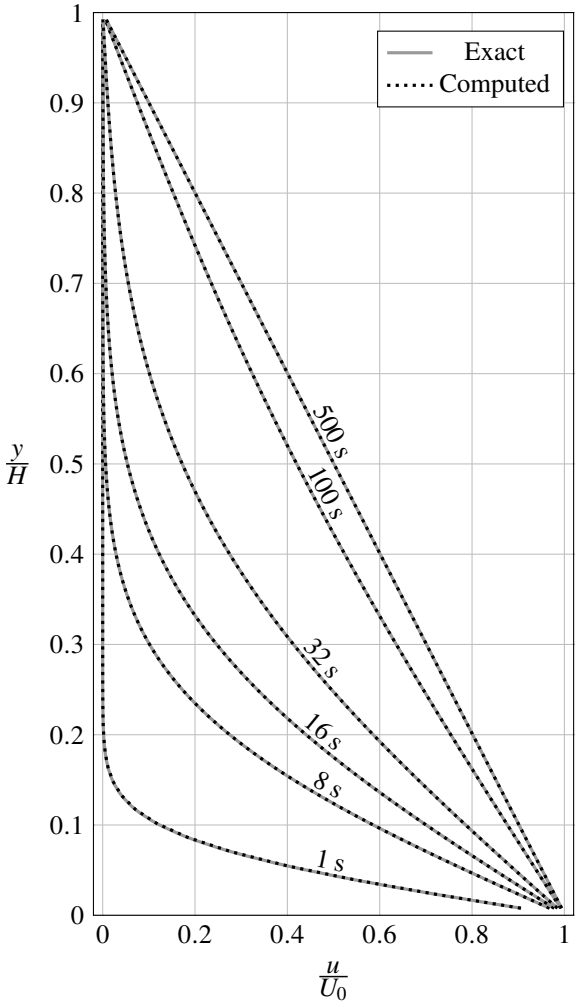

Figure 9. Comparison of the computed solution on mapped grids and analytical solution for the transient Couette flow at time = 1s, 8s, 16s, 32s, 100s, and 500s.

DOI: 10.1002/fld

Table II. Numerical values of the transient Couette flow solution errors measured with the L_∞-, L_1-, and L_2-norms at 2.5004 seconds and the convergence rates between consecutive grid resolutions.

L norm	64	128	rate	256	rate	512	rate
L_∞	5.065e-09	2.313e-10	4.45	7.147e-12	5.01	2.139e-13	5.06
L_1	1.207e-12	3.477e-14	5.11	1.059e-15	5.03	4.565e-17	4.53
L_2	6.670e-11	2.164e-12	4.94	5.060e-14	5.41	1.333e-15	5.24

The problem has an exact analytical solution allowing for error to be described by a particular norm given by

$$L_p = \left(\frac{\int |u - u_{\text{exact}}|^p \mathrm{d}\Omega}{\Omega} \right)^{\frac{1}{p}},$$

where $\mathrm{d}\Omega$ is the area associated with a discrete error measure. By evaluating a sequence of grids, the error can be fit to the relation

$$L_p = \alpha {N_D}^{\beta},$$

where N_D is the grid size in dimension D, β denotes the spatial order of accuracy (for the fourth-order, $\beta \approx -4$) and α describes the absolute magnitude of the error. In this work, the grid size is equal in all dimensions.

Table II lists the solution errors on mapped grids measured by the L-norms, showing the convergence rates on the fine mesh beyond 4.0. Considering the velocity field is initialized with an exact solution at 2.5 s, a super convergence is observed. It verifies that a fourth-order accurate viscous flux operator is achieved.

The computed transient Couette flow solution on the size of 64 grid and the analytical solution are compared for times at 1.0, 8.0, 16.0, 32.0, 100.0, and 500.0 seconds and plotted in Figure 9. The computed and the exact solutions overlap. The solution reaches steady state after about 500 seconds. Figure 10 shows the norms of the solution error norms and the slopes for all the norms fall between -5 and -4. Therefore, the viscous operators are at least fourth-order accurate.

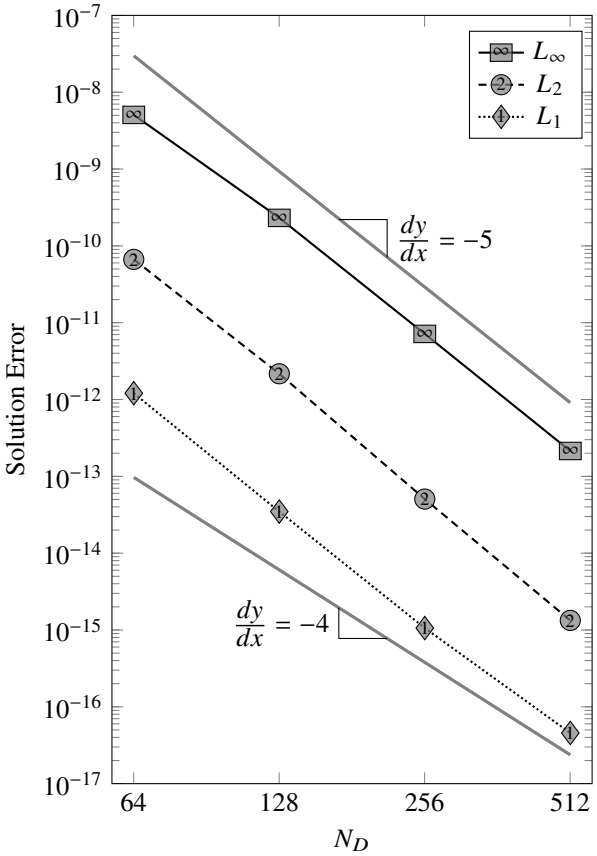

Figure 10. Solution accuracy on mapped grids measured by the L-norms for transient Couette flow.

Previously, Gao et al. [23] verified the solution accuracy of a fourth-order finite-volume method with adaptive mesh refinement on Cartesian grids using the Couette flow case. Comparison of the solution on the mapped grids in this work was made to the solution on Cartesian grids from the previous study, and an excellent agreement was observed.

5.4. *Sod's Shock Tube*

The ability of the algorithm to accurately resolve shock and rarefaction waves was validated by solving Sod's classical shock tube problem [33]. The problem is solved in multi-dimensions. The

DOI: 10.1002/fld

1D Sod's shock tube is represented in Figure 11, where a diaphragm is imagined along the centered dashed line. On the left side, the pressure and density are both given values of 1.0. On the right side, the pressure is 0.1 and the density is 0.125. The diaphragm is instantly removed resulting in a Riemann problem. Rarefaction waves propagate into the high-pressure region on the left. A shock wave propagates into the low-pressure region on the right. The shock wave is followed by a contact surface moving at the flow velocity. The pressure remains constant across the contact surface while the there is a discontinuous change in density.

In order to demonstrate the mapping, a 2D grid is used as in Figure 11 showing the mesh at time $t = 0.2$. AMR is used to resolve the discontinuities and it is clearly shown in the regions of the contact surface and the shock wave. Each new grid level refines the mesh by a factor of two. Cells are tagged for refinement when the magnitude of the relative gradient of density exceeds some threshold, for instance,

$$\text{refine cell } \boldsymbol{i} \text{ if } \sqrt{\left(\sum_{d=1}^{D}\left(\frac{\rho_{\boldsymbol{i}+\boldsymbol{e}^d}-\rho_{\boldsymbol{i}-\boldsymbol{e}^d}}{\rho_{\boldsymbol{i}+\boldsymbol{e}^d}+\rho_{\boldsymbol{i}-\boldsymbol{e}^d}}\right)^2\right)} \geq \epsilon\,.$$

This operation is applied to the primitive variables on the grid in computational space. In this case, the threshold, ϵ, is set to 0.15.

A trace of the density along the line $y = 0.5$, marked in Figure 11, is plotted in Figure 12. Along with an analytical solution, a result obtained from solving the Euler equations on the same AMR grid is also plotted in the figure to compare with the solution from solving the full compressible Navier-Stokes equations. The Navier-Stokes solution represented by the dashed line and the Euler solution denoted by the dotted line are nearly identical in smooth regions, while, as expected, the Navier-Stokes solution is slightly more dissipated near discontinuities than the Euler solution. AMR is shown to be quite effective at reducing the error near the shock and contact discontinuities.

The Sod problem is also solved in 2D and 3D with the interface for the initial condition laid diagonally across the grid. This configuration also allows us to examine the influence of cross flow errors. For effective illustration of the mapped AMR patches, we will present the profiles using the 2D result, with which the 3D solution agrees. In Figure 13, the initial condition laid diagonally across the grid from the left bottom corner to the right upper corner. The diagram is placed along the dashed line, marked by initial discontinuity. A 3-level AMR grid is shown, and the figure also shows three solid lines along which the density profiles are compared at the same time. Figure 14 compares the three profiles obtained from the numerical solution of the Navier-Stokes equations to the exact 1D Euler solution. They agree well in general with the exact Euler solution considering the viscous effect, while the profiles show minor cross-flow error in comparison to that in Figure 12.

It is worth noting that a piece-wise parabolic (PPM) limiter [7] is used to solve the Sod problem. Once the limiter is engaged in the solution process, the solution accuracy is reduced to the first order. For smooth flows, the algorithm demonstrates fourth-order accurate, such as the Couette flow as shown in the previous section. Indeed, the order of accuracy is controlled by the use of limiters. In addition, our discretization operators are dimensionally split, hence the computational cost varies linearly with the number of space dimensions. Nevertheless, it can be affected by the time step, for example, the von Neumann number for the viscous flows.

5.5. Mach Reflection Problem

To demonstrate the mapped algorithm on a non-rectangular physical domain, a Mach reflection problem is considered. The simulation is compared against experimental results published by Ben-Dor and Glass [4]. The experimental test gas is argon; real gas effects are thus strongly inhibited. The gas is assumed to be ideal with $\gamma = 1.6667$ and a gas constant of $208.1\ \mathrm{J/(Kg \cdot K)}$. Given the monatomic structure of the gas, the specific heat ratio is invariant with temperature and the gas can be modeled as calorically perfect.

A sketch of the geometry is shown in Figure 15. Initial conditions consist of a shock wave positioned before the wedge and moving right to left. From the experiment, the shock has a Mach

DOI: 10.1002/fld

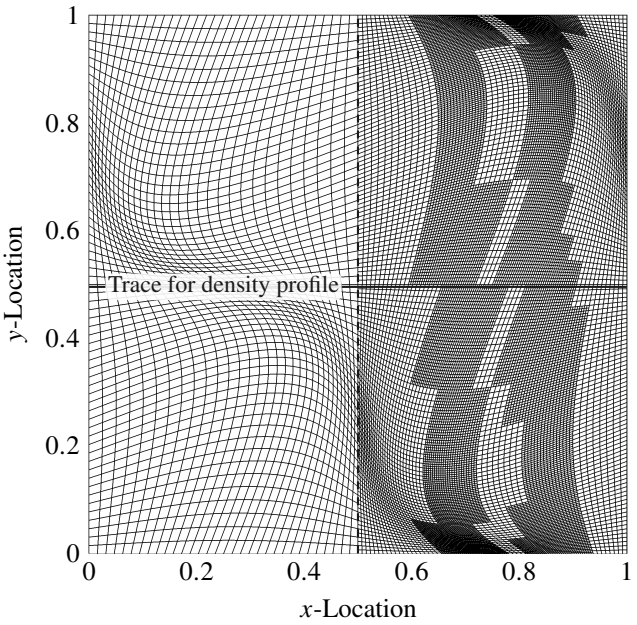

Figure 11. AMR mesh with three levels at $t = 0.2$. The base grid is 64×64.

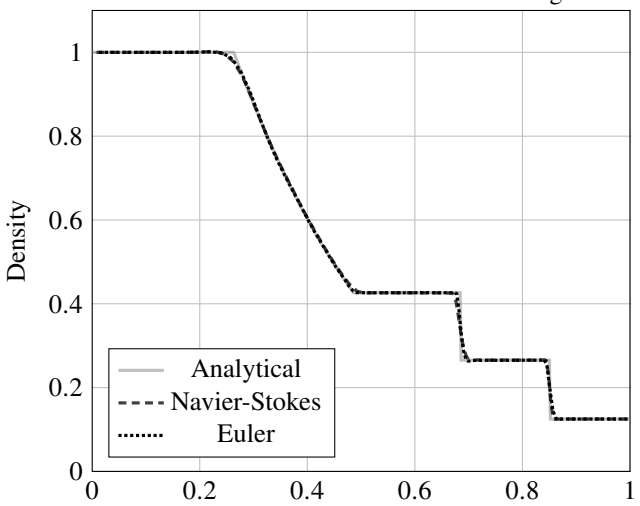

Figure 12. Density at $y = 0.5$ and $t = 0.2$. Two numerical results, the dashed and the dotted lines, are obtained on a mesh sized 64×64 by solving the Navier-Stokes and the Euler equations, respectively. The AMR results add two levels of refinement, each with a 2 times increase in resolution.

number of 2.82 and the flow ahead of it is quiescent. In [4], three conditions are given ahead of the shock, which are inconsistent with respect to the ideal gas law: $T_1 = 295$–299 K, $p_1 = 2000$ Pa, and $\rho_1 = 4.354 \times 10^{-2}$ kg/m^3. We selected $T_1 = 299$ K, $\rho_1 = 4.354 \times 10^{-2}$ kg/m^3 and calculated $p_1 = 2709$ Pa. From unsteady shock relations for a calorically perfect gas, the conditions behind the shock are $T_1 = 998$ K, $\rho_2 = 1.265 \times 10^{-1}$ kg/m^3 and $p_2 = 26250$ Pa. The induced velocity is $u_2 = -595$ m/s which is slightly supersonic. The post-shock conditions were entered into CEA [6] to compute a dynamic viscosity of 5.543×10^{-5} kg/m $\cdot$ s and a thermal conductivity of 4.341×10^{-2} W/m $\cdot$ K. These values are then assumed constant in the simulation. Considering that velocity gradients only appear behind the shock, this is a reasonable approximation.

The wedge has an angle of $20°$. The dimensions of the domain are sized to match the experiment where, we are given the distance the shock has proceeded along the ramp. This distance, from P–Q in Fig. 18 is 55.2 mm. From this, the dimensions in Figure 15 are set as follows: $x_1 = 0.02$ m, $x_2 = 0.07$ m, and $x_3 = 0.155$ m. The long lead length before the wedge is used to ensure that the

DOI: 10.1002/fld

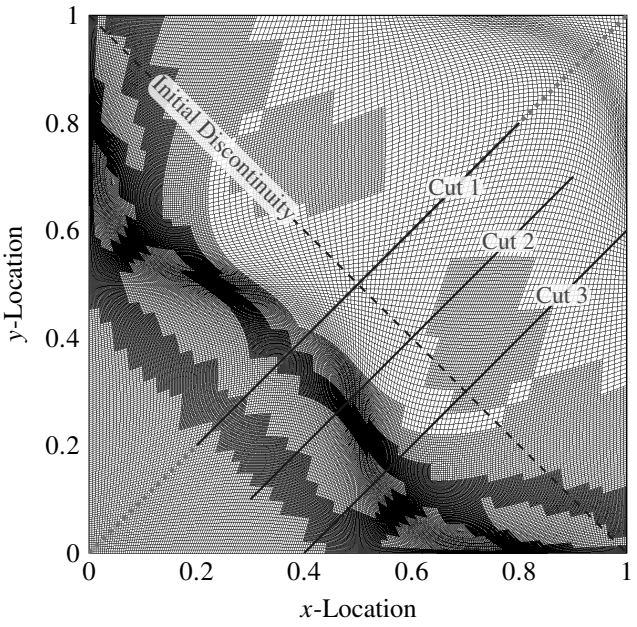

Figure 13. Solid lines showing the locations where the profiles are plotted.

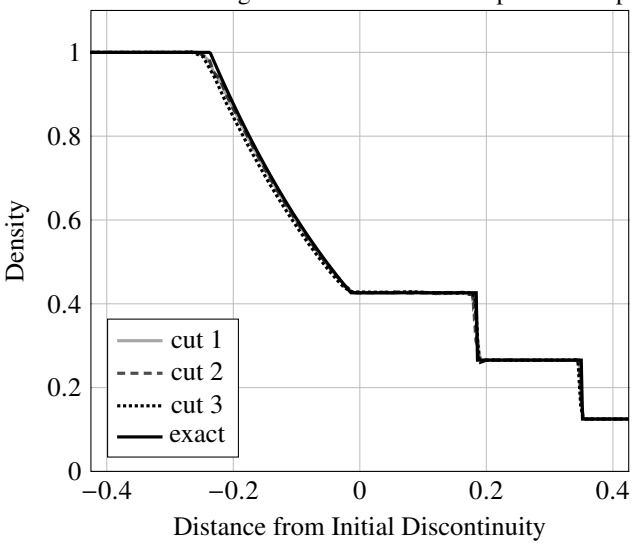

Figure 14. Density profiles at the three cuts and $t = 0.2$. The x coordinate is the distance away from the initial discontinuity where the origin is assumed.

fluid flowing into the reflected shock has been been visited by the shock over the duration of the simulation. In other words, the boundary layer is developed as much as is possible, in the vicinity of the shocks, over the duration of the simulation. The simulation is run for 0.107 ms at which point the Mach stem is positioned 55.2 mm from the wedge corner.

The computational grid has a 96×24 base grid with 2 levels of AMR, each refined by a factor of 4. Figure 16 shows the AMR patches for the finest grid along with density contours. Cells are tagged for refinement based on an undivided gradient of density. All cells near the boundary are also tagged to ensure resolution of the boundary layer. Figure 17 shows the AMR patches on all levels in the computational space along with density contours.

The shock reflection shown in Figure 16 is known as simple Mach reflection. A discussion of various shock reflection patterns is available in Ben-Dor and Glass [4] and the references therein. In Figure 18, the density contours are quantitatively compared against the experimental results

DOI: 10.1002/fld

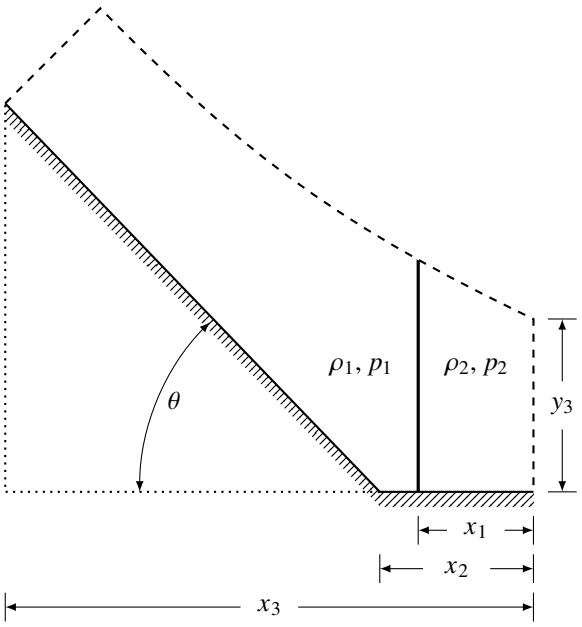

Figure 15. Shock ramp geometry. Subscripts, 1 and 2, indicate states, before and after the shock, respectively.

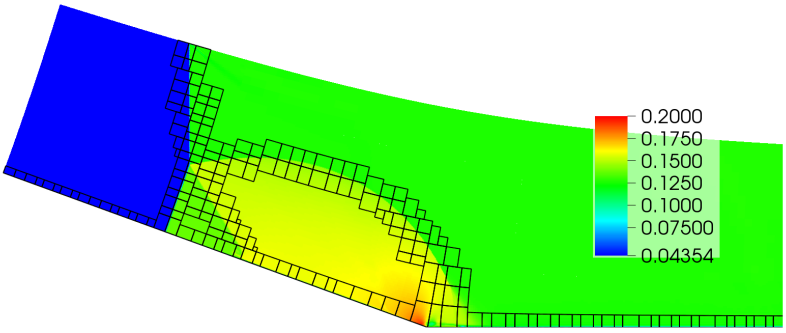

Figure 16. The AMR patches are shown on the finest level in the physical space along with contours of density ($\mathrm{kg/m^3}$).

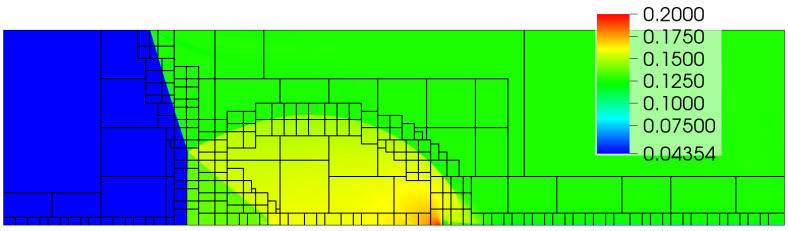

Figure 17. The AMR patches are shown on all levels in the computational space along with contours of density ($\mathrm{kg/m^3}$).

published by Ben-Dor and Glass [4]. Note in this figure that the density contours are relative to the freestream density $\rho_1 = \rho_0 = 4.354 \times 10^{-2}\ \mathrm{kg/m^3}$. Scaling is required to overlay the results; we matched both point Q and the horizontal location of the incident shock (between labels 1 and 2). There is general agreement although some notable differences are observed. In particular, flow separation is observed in the simulation before the wedge (point Q in Figure 18). This is the primary reason for the disparate contours near point Q. The separation induces a larger displacement of the boundary layer, creating both a lambda shock where the reflected shock intersects the wall, and

DOI: 10.1002/fld

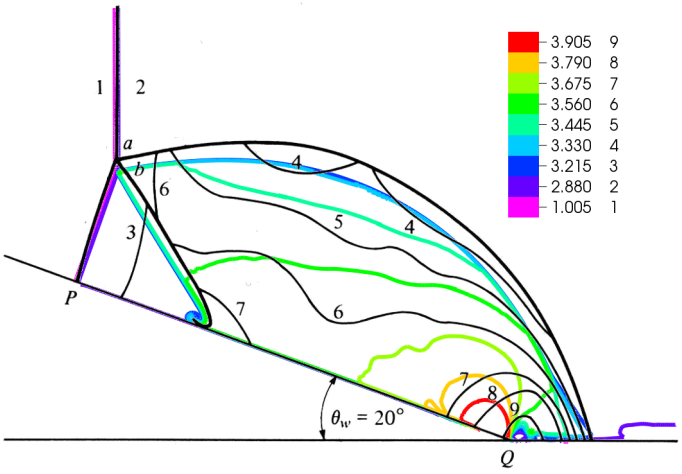

Figure 18. Computed relative density (ρ/ρ_0) contours are shown in color. Experimental results are shown in black contours and are reproduced from Ben-Dor and Glass [4] (used with permission).

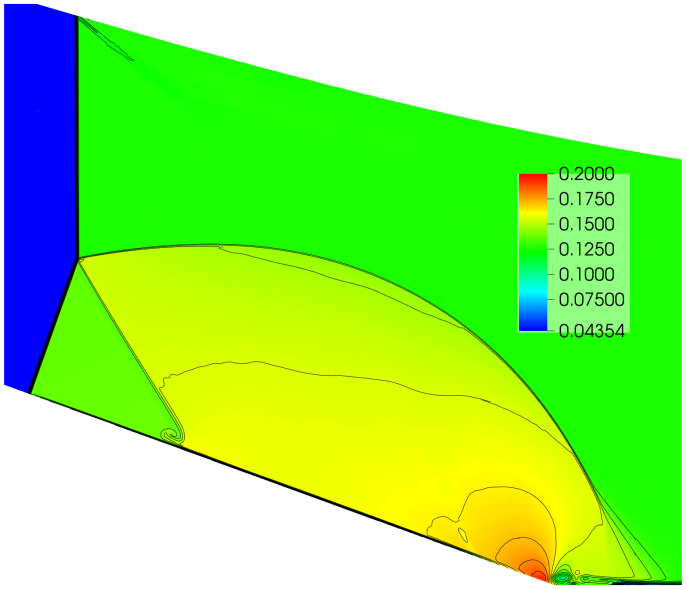

Figure 19. Density contours ($\mathrm{kg/m^3}$) showing the lambda shock where the reflected shock intersects the wall.

a shorter Mach stem (distance from P to a). The former is commonly observed when an oblique shock intersects a boundary layer [27]. The lambda shock is more evident in the density contours of Figure 19. With respect to the latter, the effect of the boundary layer displacement is to reduce the apparent wedge angle. In summary, the comparison suggests that the simulated boundary layer is too thick. The disparate contours near point Q suggest that this difference is a result of separation in the numerical results whereas no separation is observed in experiment. In future work, we hope to investigate this issue more thoroughly. Avenues to explore include allowing viscosity to vary with temperature and increasing the resolution of the grid. Separation is also observed in other numerical results [21] (albeit for very different cases), yet is always very difficult to identify in experiment.

DOI: 10.1002/fld

6. CONCLUSIONS AND FUTURE WORK

A mapped, fourth-order finite-volume algorithm with adaptive mesh refinement was developed to solve the full compressible Navier-Stokes equations. The free-stream preservation property is maintained. Fourth-order accuracy was verified with the transient Couette flow. Further, the ability of the algorithm to accurately resolve shock and rarefaction waves was validated by solving Sod's classical shock tube problem. In addition, the mapped algroithm is demonstrated on a non-rectangular physical domain with the Mach reflection on a ramp. It is worth mentioning here that our algorithm is built upon Chombo and it has been shown that Chombo scales up to 2×10^5 cores for explicit schemes and 1×10^5 cores for implicit schemes [34]. Our algorithm is expected to exhibit similar parallel performance. A follow-up study to evaluate the scalability of our parallel adaptive numerical method on mapped grids for solving compressible viscous flows will be performed. Further validation will be carried out for fully three-dimensional flows.

7. ACKNOWLEDGMENT

This research at Colorado State University was supported by U.S. Department of Energy under Award Number DE-EE0006086, and by the National Science Foundation under Award Number CCF-1422725. The authors would also like to thank Nathaniel Overton from the Computational Fluid Dynamics and Propulsion Laboratory at Colorado State University for making figures with the tikz drawing language. The authors would like to thank the anonymous reviewers for their valuable suggestions to improve the quality of the paper.

8. APPENDIX

Using shorthand notations, such as $x_\xi = \frac{\partial x}{\partial \xi}$, the covariant unit basis vectors, by definition from Equation (17), are expanded as

$$\vec{e}_1 = \frac{(x_\xi \vec{i} + y_\xi \vec{j} + z_\xi \vec{k})}{\sqrt{x_\xi^2 + y_\xi^2 + z_\xi^2}}, \quad \vec{e}_2 = \frac{(x_\eta \vec{i} + y_\eta \vec{j} + z_\eta \vec{k})}{\sqrt{x_\eta^2 + y_\eta^2 + z_\eta^2}}, \quad \vec{e}_3 = \frac{(x_\zeta \vec{i} + y_\zeta \vec{j} + z_\zeta \vec{k})}{\sqrt{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}}.$$

Note that only grid metrics are expressed in such shorthand notation.

Often, when it comes to the implementation of boundary conditions, writing the basis vectors in grid metrics (derivatives of $\vec{\xi}$ with respect to $\boldsymbol{x}$) is useful and they are expressed as

$$\begin{aligned}
\vec{e}_1 &= \frac{(\eta_y\zeta_z - \zeta_y\eta_z)\vec{i} - (\eta_x\zeta_z - \zeta_x\eta_z)\vec{j} + (\eta_x\zeta_y - \zeta_x\eta_y)\vec{k}}{\sqrt{(\eta_y\zeta_z - \zeta_y\eta_z)^2 + (\zeta_x\eta_z - \eta_x\zeta_z)^2 + (\eta_x\zeta_y - \zeta_x\eta_y)^2}}, \\
\vec{e}_2 &= \frac{(\zeta_y\xi_z - \xi_y\zeta_z)\vec{i} - (\zeta_x\xi_z - \xi_x\zeta_z)\vec{j} + (\zeta_x\xi_y - \xi_x\zeta_y)\vec{k}}{\sqrt{(\zeta_y\xi_z - \xi_y\zeta_z)^2 + (\xi_x\zeta_z - \zeta_x\xi_z)^2 + (\zeta_x\xi_y - \xi_x\zeta_y)^2}}, \quad \text{and} \\
\vec{e}_3 &= \frac{(\xi_y\eta_z - \eta_y\xi_z)\vec{i} - (\xi_x\eta_z - \eta_x\xi_z)\vec{j} + (\xi_x\eta_y - \eta_x\xi_y)\vec{k}}{\sqrt{(\xi_y\eta_z - \eta_y\xi_z)^2 + (\eta_x\xi_z - \xi_x\eta_z)^2 + (\xi_x\eta_y - \eta_x\xi_y)^2}}.
\end{aligned}$$

The contravariant unit basis vectors defined by Equation (18) are written as

$$\vec{E}^1 = \frac{(\xi_x \vec{i} + \xi_y \vec{j} + \xi_z \vec{k})}{\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}}, \quad \vec{E}^2 = \frac{(\eta_x \vec{i} + \eta_y \vec{j} + \eta_z \vec{k})}{\sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2}}, \quad \vec{E}^3 = \frac{(\zeta_x \vec{i} + \zeta_y \vec{j} + \zeta_z \vec{k})}{\sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}}.$$

DOI: 10.1002/fld

We find the tangential and normal velocity components to be

$$u_{t_\xi} = \vec{e_1} \cdot \left(u_x\vec{i} + u_y\vec{j} + u_z\vec{k} \right) = \frac{(\eta_y\zeta_z - \zeta_y\eta_z)u_x - (\eta_x\zeta_z - \zeta_x\eta_z)u_y + (\eta_x\zeta_y - \zeta_x\eta_y)u_z}{\sqrt{(\eta_y\zeta_z - \zeta_y\eta_z)^2 + (\zeta_x\eta_z - \eta_x\zeta_z)^2 + (\eta_x\zeta_y - \zeta_x\eta_y)^2}} ,$$

$$u_{t_\zeta} = \vec{e_3} \cdot \left(u_x\vec{i} + u_y\vec{j} + u_z\vec{k} \right) = \frac{(\xi_y\eta_z - \eta_y\xi_z)u_x - (\xi_x\eta_z - \eta_x\xi_z)u_y + (\xi_x\eta_y - \eta_x\xi_y)u_z}{\sqrt{(\xi_y\eta_z - \eta_y\xi_z)^2 + (\eta_x\xi_z - \xi_x\eta_z)^2 + (\xi_x\eta_y - \eta_x\xi_y)^2}} ,$$

$$u_{n_\eta} = \vec{E^2} \cdot \left(u_x\vec{i} + u_y\vec{j} + u_z\vec{k} \right) = \frac{\eta_x u_x + \eta_y u_y + \eta_z u_z}{\sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2}} .$$

From above, a mathematical relation can be easily established for solving the Cartesian velocity components at the body surface, which are required by the algorithm,

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = A \begin{bmatrix} u_{t\xi} \\ u_{t\zeta} \\ u_{n\eta} \end{bmatrix} ,$$

where

$$A = \begin{bmatrix} m_1(\eta_y\zeta_z - \zeta_y\eta_z) & -m_1(\eta_x\zeta_z - \zeta_x\eta_z) & m_1(\eta_x\zeta_y - \zeta_x\eta_y) \\ m_2(\xi_y\eta_z - \eta_y\xi_z) & -m_2(\xi_x\eta_z - \eta_x\xi_z) & m_2(\xi_x\eta_y - \eta_x\xi_y) \\ m_3\eta_x & m_3\eta_y & m_3\eta_z \end{bmatrix}^{-1}$$

with factors, m_1, m_2, and m_3, being given by

$$m_1 = 1/\sqrt{(\eta_y\zeta_z - \zeta_y\eta_z)^2 + (\zeta_x\eta_z - \eta_x\zeta_z)^2 + (\eta_x\zeta_y - \zeta_x\eta_y)^2} ,$$
$$m_2 = 1/\sqrt{(\xi_y\eta_z - \eta_y\xi_z)^2 + (\eta_x\xi_z - \xi_x\eta_z)^2 + (\xi_x\eta_y - \eta_x\xi_y)^2} ,$$
$$m_3 = 1/\sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2} .$$

REFERENCES

1. Henshaw WD. A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids. *Journal of Computational Physics* 1994; 113(1):13-25.
2. Yee HC. Explicit and Implicit Multidimensional Compact High-Resolution Shock-Capturing Methods. *Journal of Computational Physics* 1997; 131(1):216-232.
3. Barad M, Colella P. A Fourth-Order Accurate Local Refinement Method for Poisson's Equation. *Journal of Computational Physics* 2005; 209(1):1-18.
4. Ben-Dor G, Glass I. Domains and Boundaries of Non-Stationary Oblique Shock-Wave Reflexions. 2. Monatomic Gas. *Journal of Fluid Mechanics* 1980; 96(4):735-756.
5. Colella P, Dorr M, Hittinger J, Martin D, and McCorquodale P. High-Order Finite-Volume Adaptive Methods on Locally Rectangular Grids. *Journal of Physics Conference Series* 180 012010, 2009.
6. Gordon S and McBride B. Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications. NASA Reference Publication 1311, 1994.
7. McCorquodale P and Colella P. A High-Order Finite-Volume Method for Conservation Laws on Locally Refined Grids. *Communications in Applied Mathematics and Computational Science* 2011; 6(1): 1-25.
8. McCorquodale P, Ullrich P, Johansen H, and Colella P. An Adaptive Multiblock High-Order Finite-Volume Method for Solving the Shallow Water Equations on the Sphere. *Communications in Applied Mathematics and Computational Science* 2015; 10(2): 121-162.
9. Colella P, Dorr MR, Hittinger J, and Martin DF. High-Order, Finite-Volume Methods in Mapped Coordinates. *Journal of Computational Physics* 2011; 230:2952-2976.
10. Guzik S, McCorquodale P, and Colella P. A Freestream-Preserving High-Order Finite-Volume Method for Mapped Grids with Adaptive-Mesh Refinement. *50th AIAA Aerospace Sciences Meeting* 2012; AIAA 2012-0574.
11. Guzik S, Gao X, Owen L, McCorquodale P, and Colella P. A Freestream-Preserving Fourth-Order Finite-Volume Method in Mapped Coordinates with Adaptive-Mesh Refinement. *Computers and Fluids* 2015; 123:202-217.
12. Spotts N. Unsteady Reynolds-Averaged Navier-Stokes Simulations of Inlet Flow Distortion in the Fan System of a Gas-Turbine Engine. *Masters Thesis*, Colorado State University, 2015.
13. Zhang Q, Johansen H, and Colella P. A Fourth-Order Accurate Finite-Volume Method with Structured Adaptive Mesh Refinement for Solving the Advection-Diffusion Equation. *SIAM Journal on Scientific Computing* 2012; 34(2): B179B201.

Prepared using fldauth.cls
DOI: 10.1002/fld

14. Lani A, SjÖgreen B, Yee HC, and Henshaw W. Variable High-Order Multiblock Overlapping Grid Methods for Mixed Steady and Unsteady Multiscale Viscous Flows, part II: hypersonic nonequilibrium flows. *Communications in Computational Physics* 2012. 13(2):583-602.
15. Berger MJ and Colella P. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics* 1989; 82(1):64-84.
16. Groth C, De Zeeuw DL, Powell KG, Gombosi TI, and Stout QF. A Parallel Solution-Adaptive Scheme for Ideal Magnetohydrodynamics.*14th Computational Fluid Dynamics Conference* 1999; AIAA 99-3273.
17. Day MS and Bell JB. Numerical Simulation of Laminar Reacting Flows with Complex Chemistry. *Combustion Theory Modelling* 2000; 4:535-556.
18. Mavriplis DJ. Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes. *AIAA Journal* 1990; 28:213-221.
19. Lawlor OS, Chakravorty S, Wilmarth TL, Choudhury N, Dooley,I, Zheng G, and Kalé L. ParFUM: A Parallel Framework for Unstructured Meshes for Scalable Dynamic Physics Applications. *Engineering with Computers* 2006; 22:215-235.
20. Berger MJ and Aftosmis MJ. Progress Towards a Cartesian Cut-Cell Method for Viscous Compressible Flow. *50th AIAA Aerospace Sciences Meeting* 2012; AIAA 2012-1301.
21. Graves DT, Colella P, Modiano D, Johnson J, SjÖgreen B, and Gao X. A Cartesian Grid Embedded Boundary Method for the Compressible Navier Stokes Equations. *Communications in Applied Mathematics and Computational Sciences* 2013; 8(1):99-122.
22. Steger JL. Implicit Finite Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries, *16th AIAA Aerospace Sciences Meeting*1978; AIAA 1978-665.
23. Gao X, Guzik S, and Colella P. Fourth Order Boundary Treatment for Viscous Fluxes on Cartesian Grid Finite-Volume Methods. *52nd AIAA Aerospace Sciences Meeting* 2014; AIAA 2014-1277.
24. Adams M, Colella P, Graves DT, Johnson JN, Johansen HS, Keen ND, Ligocki TJ, Martin DF, McCorquodale PW, Modiano D, Schwartz PO, Sternberg TD, and Van Straalen B. *Chombo Software Package for AMR Applications - Design Document*, Lawrence Berkeley National Laboratory, LBNL-6616E, 2014.
25. Berger MJ and Oliger J. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics* 1989; 53:484-512.
26. Berger MJ. On Conservation at Grid Interfaces. *SIAM Journal of Numerical Analysis* 1987; 24(5):967-984.
27. Liepmann H, Roshko A, and Dhawan S. On Reflection of Shock Waves from Boundary Layers. NACA Technical Report 1100, 1952.
28. Persson PO, Bonet J, and Peraire J. Discontinuous Galerkin Solution of the Navier-Stokes Equations on Deformable Domains. *Computer Methods in Applied Mechanics and Engineering* 2009; 198:1585-1595.
29. Batchelor GK. *Introduction to Fluid Dynamics*. Cambridge University Press, 1970.
30. Sherman F. *Viscous Flows*. McGraw-Hill, 1991.
31. White FM. *Viscous Fluid Flow*. McGraw-Hill, 1991.
32. Muzychka YS and Yovanovich MM. Unsteady Viscous Flows and Stoke's First Problem. *International Journal of Thermal Sciences* 2010; 49: 820-828.
33. Sod GA. A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws. *Journal of Computational Physics* 1978; 27:1-31.
34. Van Straalen B, Colella P, Graves DT, and Keen N. Petascale Block-Structured AMR Applications Without Distributed Meta-data. *Proceedings, Part II. Lecture Notes in Computer Science* 2011; Springer,ISBN 978-3-642-23396-8.
35. McBride BJ and Gordon S. Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications II. Users Manual and Program Description.*NASA Report* 1996; NASA 96-1211.

DOI: 10.1002/fld

A Parallel Adaptive Numerical Method with Generalized Curvilinear Coordinate Transformation for Compressible Navier-Stokes Equations

X. Gao*, L. D. Owen, and S. M. J. Guzik

A fourth-order finite-volume method for solving the Navier-Stokes equations on a mapped grid with adaptive mesh refinement is proposed, implemented, and demonstrated for the prediction of unsteady compressible viscous flows. Shown here, a Mach reflection problem is solved to demonstrate the effectiveness of the mapped algorithm on a non-rectangular physical domain. AMR patches on the finest mesh level are outlined.