AIAA 2016-1101

FINAL VERSION: 54th AIAA Aerospace Sciences Meeting, 4 - 8 January, San Diego, California, 2016

Hyperbolic Navier-Stokes Solver for Three-Dimensional Flows

Yoshitaka Nakashima*and Norihiko Watanabe†

Software Cradle Co., Ltd, Osaka, JAPAN

Hiroaki Nishikawa‡

National Institute of Aerospace, Hampton, VA 23666, USA

In this paper, we present a hyperbolic Navier-Stokes solver for three-dimensional compressible viscous flows. Hyperbolic Navier-Stokes systems, which have been constructed and demonstrated in two dimensions, are extended here to three dimensions and the eigenstructure of the hyperbolized viscous terms is derived. The system is discretized by the second-order node-centered edge-based method on unstructured grids. The resulting residual equations are solved by an implicit defect-correction solver based on a compact Jacobian derived exactly from a first-order edge-based discretization. Numerical results are presented to examine the practical applicability and potential advantages of the hyperbolic Navier-Stokes method, accurate gradients in particular, through three-dimensional viscous flow problems with unstructured grids.

I. Introduction

Computational Fluid Dynamics (CFD) has been widely used for analysis and design in a broad range of industrial applications from aeronautical engineering to biomedical science, especially since the advent of three-dimensional general-purpose unstructured-grid solvers during the 1990's. Software CRADLE introduced a fully parallelized unstructured-grid CFD software called SC/Tetra in 1998, which is a comprehensive CFD package with grid generation, flow solvers, and post-processing/visualization. Since then, it has been under continuous and active development, including a recent development of an improved fully-implicit density-based compressible-flow solver [1]; and SC/Tetra has become one of the most successful CFD packages ever developed at CRADLE since its inception in 1984. Practical unstructured-grid CFD codes such as SC/Tetra have been constructed and sustained by a number of technologies developed through a large amount of fundamental research work in the past decades. Today, the current state-of-the-art unstructured-grid CFD codes have matured, undeniably, to the level at which they provide useful answers to a wide range of complex real-world applications. Nevertheless, the unstructured-grid CFD technology is still developing, and the future of practical unstructured-grid CFD codes is promised only through continuous algorithmic improvements. Therefore, in addition to efficient optimization methods, improved physical modeling, and uncertainty quantifications, advancements in discretization methods and linear/nonlinear solvers are also needed to improve robustness and efficiency for complex industrial applications.

Among various CFD research efforts, high-order methods have been one of the most active areas for algorithm development. Despite the progress made in the past decades, however, second-order algorithms still dominate practical three-dimensional CFD codes, including both commercial and government codes, especially for applications involving complex geometries. As pointed out in Refs.[2,3], one of the bottlenecks in the practical application of high-order methods is the requirement of high-order grids with curved elements for curved geometries, which is by no means trivial although progress is being made (e.g., Refs.[4–6]). It

*Software Engineer (nakashima@cradle.co.jp), Software Cradle Co., Ltd, Umeda Kita-ku, Osaka, Japan

†Software Engineer (watanabe@cradle.co.jp), Software Cradle Co., Ltd, Umeda Kita-ku, Osaka, Japan

‡Associate Research Fellow (hiro@nianet.org), 100 Exploration Way, Hampton, VA 23666 USA, Associate Fellow AIAA

Copyright © 2016 by Yoshitaka Nakashima, Norihiko Watanabe, Hiroaki Nishikawa. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

should be mentioned that grid generation for complex geometries, at least from our perspective, remains one of the major difficulties in industrial CFD applications even for second-order methods. Another issue is a lack of efficiency and robustness in high-order methods. As high-order methods are always more expensive, on a given grid, than second-order methods, grid adaptation is expected to play a critical role for achieving target accuracy on a relatively coarse grid so that the high-order methods can be applied at a comparable or possibly lower cost than that of second-order methods. However, a full potential of grid adaptation technique has not been realized, again even for second-order methods, especially in three-dimensional viscous flow problems where grid adaptation is applied exclusively outside a boundary layer, requiring high-quality anisotropic grid generation around viscous surfaces. The robustness and efficiency issue also arises in nonlinear solvers for high-order methods, which is required for steady as well as unsteady applications with implicit time-stepping. In particular, the construction of a robust and efficient nonlinear solver may be practically expensive in some methods due to an extremely large memory requirement for residual Jacobians (e.g., 50×50 Jacobian block for a third-order discontinuous Galerkin method). Explicit time-stepping schemes do not require residual Jacobians, and therefore are significantly cheaper per time step than implicit time-stepping schemes. But they can be useful only if they allow a practically large time step on highly stretched viscous grids typically used in practical CFD simulations. In this work, our main interest is implicit time-stepping schemes, which, as our previous study [1] indicates, are significantly more efficient for many of the current industrial applications at which SC/Tetra is targeting.

A possible alternative to conventional high-order methods, which can potentially bring immediate improvements to existing unstructured-grid CFD solvers, is the Hyperbolic Navier-Stokes (HNS) method. It is a first-order hyperbolic reformulation approach to diffusion originally developed in a series of papers [7, 8]. Later, it was extended to the Navier-Stokes (NS) equations in Ref.[9], and since then, there has been an increasing interest in developing numerical algorithms based on a hyperbolic formulation of diffusion and viscous terms: high-order steady/unsteady residual-distribution schemes [10–12], high-order edge-based schemes [13–18], third-order steady/unsteady active-flux schemes [19], and so on. In particular, we are interested in the edge-based discretization because it is considered as one of the most efficient discretization methods [20], and also because it is relatively easy to implement into SC/Tetra where the node-centered edge-based discretization is already available. More specifically, we are interested in the second-order HNS scheme, which is still entirely based on second-order algorithms, but has a potential for providing unique advantages over conventional second-order schemes:

1. Second-order accuracy is achieved in the gradient quantities (e.g., viscous stresses, vorticity, heat fluxes) on irregular grids, which are inherently first-order accurate in conventional second-order methods [14–16, 18].

2. Accurate and non-oscillatory gradients can be obtained on severely distorted grids such as adapted viscous grids, thus allowing accurate viscous simulations with full grid adaptation through a boundary layer [12, 16, 18].

3. Hessian (second-order partial derivatives), which is often utilized to guide anisotropic grid adaptation, can be obtained with first-order accuracy on irregular grids; it is inconsistent (zero-th order) and involves numerical noise that does not diminish in the grid refinement in conventional second-order methods [21].

4. A consistent viscous Jacobian can be constructed from a first-order upwind viscous scheme, thus potentially leading to a robust and efficient implicit solver [16, 18].

5. Third-order accuracy can be achieved for the inviscid terms by second-order algorithms [18]; and moreover, a single flux evaluation per edge is still sufficient for third-order accuracy.

6. Higher-order accuracy in the inviscid terms and in the gradient quantities can be achieved on existing grids; high-order grids with curved elements are not needed [17].

7. Improved iterative convergence is achieved due to the elimination of second-derivatives from the viscous terms [9, 16, 18].

These advantages, which have been demonstrated in a series of papers [9, 16, 18] for the NS equations in two dimensions, appear to be highly attractive for practical CFD codes. Firstly, physical quantities of interest are

often related to the gradient quantities such as the viscous stresses, the strain tensor, the vorticity, and the heat fluxes. Accurate prediction of the gradient quantities, however, is often a difficult task on unstructured grids. A typical resolution is to locally generate a structured-type grid in a region where accurate gradients are sought, e.g., a prismatic layer in a boundary layer. But grid adaptation performs less efficiently on such grids than unstructured grids, thus limiting the unstructured-grid technology. In this context, higher-order accuracy and high quality in the gradient prediction on irregular grids (with existing second-order methods), as demonstrated for the NS equations in Refs.[16, 18], is definitely an advantage, and could be a game-changing technology to unstructured-grid CFD methods. Secondly, the construction of a nonlinear solver can potentially be much easier for second-order methods than for high-order methods because, at least for now, there exist a bulk of research and decades of experience in second-order methods. Thirdly, a third-order inviscid scheme (by second-order algorithms!) is expected to be highly beneficial in high-Reynolds-number turbulent flows, which dominate target industrial applications. In fact, some improved inviscid schemes have been demonstrated to provide enhanced resolutions for such problems [22, 23] despite the fact that these schemes are not third-order accurate on general unstructured grids. And finally, the fact that higher-order gradients and third-order accuracy in the inviscid terms can be achieved on existing grids used in second-order computations would be a great advantage; many other third-order methods require high-order grids that are not even available at the moment, at least for many cases we currently deal with. It is also highly noteworthy that third-order accuracy in the inviscid terms is achieved without increasing the number of quadrature points; the residual can still be computed by looping over edges with a single flux evaluation per edge [17]. The objective of the present work is, therefore, to extend HNS systems to three dimensions, and to examine the practical applicability and the potential advantages of the HNS method through relatively simple but realistic three-dimensional viscous flow problems with unstructured grids.

It may be pointed out that there exist other approaches based on a first-order system formulation of the NS equations [24–26]. However, their formulations are fundamentally different from those in the HNS method. The most distinctive feature is that the viscous terms are written as a hyperbolic system in the HNS method whereas their systems are not hyperbolic. A similar hyperbolic viscous formulation is considered in Ref.[27] for the one-dimensional NS equations. Their system is deliberately designed to preserve the equivalence with the original NS system at any instant of physical time with a relaxation time carefully defined based on a mesh spacing. As a result, the stiffness due to second derivatives is not eliminated. Also, high-order accuracy of gradients remains to be demonstrated. On the other hand, the relaxation time is a mesh-independent constant in the HNS method, and the effect of second derivatives is completely eliminated. Finally, the HNS method is based on a pseudo-time formulation, and therefore time-accurate schemes can be constructed only by implicit time-stepping schemes or space-time schemes [10,11,19]; if a method is time accurate with explicit time-stepping schemes, it is not the HNS method considered here.

The paper is organized as follows. In Sections II and III, we present the extension of HNS systems to three dimensions. Eigenvalues, right-eigenvectors, left-eigenvectors, and the absolute flux Jacobian are presented for the hyperbolized viscous terms. Then, in Section IV, we discuss the discretization and the solver construction in details. The hyperbolized viscous terms are discretized with an upwind flux, and the dissipation terms are expressed with a face-normal (directed-area) vector only, not requiring ambiguous tangent vectors. In Section V, numerical results for three dimensional test cases will be presented and discussed. Finally, in Section VI, the paper concludes with remarks on future directions.

II. Hyperbolic Compressible NS System: HNS17

Consider the compressible NS equations:

$$\partial_t \rho + \mathrm{div}(\rho \mathbf{v}) = 0, \tag{II.1}$$

$$\partial_t (\rho \mathbf{v}) + \mathrm{div}(\rho \mathbf{v} \otimes \mathbf{v}) = -\mathrm{grad}\, p + \mathrm{div}\, \boldsymbol{\tau}, \tag{II.2}$$

$$\partial_t (\rho E) + \mathrm{div}(\rho \mathbf{v} H) = \mathrm{div}\,(\boldsymbol{\tau}\mathbf{v}) - \mathrm{div}\, \mathbf{q}, \tag{II.3}$$

where t is the physical time, $\otimes$ denotes the dyadic product, ρ is the density, $\mathbf{v}$ is the velocity vector, p is the pressure, E is the specific total energy, and $H = E + p/\rho$ is the specific total enthalpy. The viscous stress tensor $\boldsymbol{\tau}$ and the heat flux $\mathbf{q}$ are given, under Stokes' hypothesis, by

$$\boldsymbol{\tau} = -\frac{2}{3}\mu(\mathrm{div}\,\mathbf{v})\mathbf{I} + \mu\left(\mathrm{grad}\,\mathbf{v} + (\mathrm{grad}\,\mathbf{v})^t\right), \quad \mathbf{q} = -\frac{\mu}{Pr(\gamma-1)}\mathrm{grad}\,T, \tag{II.4}$$

where $\mathbf{I}$ is the identity matrix, T is the temperature, γ is the ratio of specific heats, Pr is the Prandtl number, μ is the viscosity defined by Sutherland's law, and the superscript t denotes the transpose. In this paper, all vectors are defined as column vectors, and therefore, if a vector has the superscript t, it is typically understood as a row vector. All the quantities are assumed to have been nondimensionalized by their free-stream values except that the velocity and the pressure are scaled by the free-stream speed of sound and the free-stream dynamic pressure, respectively (see Ref.[28]). Thus, the viscosity is given by the following form of Sutherland's law:

$$\mu = \frac{M_\infty}{Re_\infty}\frac{1+C/\tilde{T}_\infty}{T+C/\tilde{T}_\infty}T^{\frac{3}{2}}, \tag{II.5}$$

where $\tilde{T}_\infty$ is the dimensional free stream temperature, and $C = 110.5$ [K] is the Sutherland constant. The ratio of the free stream Mach number, M_∞, to the free stream Reynolds number, Re_∞, arise from the nondimensionalization. The system is closed by the nondimensionalized equation of state for ideal gases:

$$\gamma p = \rho T. \tag{II.6}$$

In the HNS method, the viscous terms are written as a hyperbolic system such that it recovers the NS equations in the pseudo steady state. The first hyperbolic formulation introduced in Ref.[9], called HNS14, is constructed by adding the viscous stresses and the heat fluxes as additional variables. Alternative hyperbolic formulations, HNS17 and HNS20, have been introduced in Ref.[18] to enable accurate computations of the velocity gradients (HNS17/20) and the density gradients (HNS20), and also an efficient implementation of a third-order inviscid scheme (HNS20). We first consider HNS17:

$$\partial_\tau \rho + \mathrm{div}(\rho \mathbf{v}) = 0, \tag{II.7}$$

$$\partial_\tau (\rho \mathbf{v}) + \mathrm{div}(\rho \mathbf{v} \otimes \mathbf{v}) = -\mathrm{grad}\, p + \mathrm{div}\, \boldsymbol{\tau}, \tag{II.8}$$

$$\partial_\tau (\rho E) + \mathrm{div}(\rho \mathbf{v} H) = \mathrm{div}\,(\boldsymbol{\tau}\mathbf{v}) - \mathrm{div}\,\mathbf{q}, \tag{II.9}$$

$$\frac{T_v}{\mu_v}\partial_\tau \mathbf{g} = \mathrm{grad}\,\mathbf{v} - \frac{\mathbf{g}}{\mu_v}, \tag{II.10}$$

$$\frac{T_h}{\mu_h}\partial_\tau \mathbf{q} = -\frac{1}{\gamma(\gamma-1)}\mathrm{grad}\,T - \frac{\mathbf{q}}{\mu_h}, \tag{II.11}$$

where τ is a pseudo time, μ_v and μ_h are scaled viscosities,

$$\mu_v = \frac{4}{3}\mu, \quad \mu_h = \frac{\gamma\mu}{Pr}, \tag{II.12}$$

and T_v and T_h are relaxation times associated with the velocity gradients and the heat flux. The system is constructed such that we have in the pseudo steady state,

$$\mathbf{q} = -\frac{\mu_h}{\gamma(\gamma-1)}\mathrm{grad}\,T, \quad \mathbf{g} = \mu_v\,\mathrm{grad}\,\mathbf{v}, \tag{II.13}$$

and thus the viscous stress tensor is expressed by

$$\boldsymbol{\tau} = -\frac{1}{2}tr(\mathbf{g})\mathbf{I} + \frac{3}{4}\left(\mathbf{g}+\mathbf{g}^t\right), \tag{II.14}$$

where $tr(\mathbf{g})$ denotes the trace of $\mathbf{g}$. In the HNS formulation, the variables related to gradients such as $\mathbf{g}$ and $\mathbf{q}$ are referred to as the gradient variables. The physical time derivatives have been ignored here, but can be discretized by an implicit time-stepping scheme and incorporated as source terms [10, 11, 19]. This work focuses on a steady solver development, which is directly relevant to a sub-iteration problem for implicit time-stepping schemes. Note that the viscous stress tensor has six independent components by symmetry but $\mathbf{g}$ has nine independent components, and thus, Equation (II.14) cannot be solved for $\mathbf{g}$. This is the reason that the HNS14 system does not lead to accurate velocity gradients; and the HNS17 system was introduced [18]. The relaxation times, T_v and T_h, are free parameters defined as in Refs.[9, 14, 15] by

$$T_v = \frac{L^2}{\nu_v}, \quad T_h = \frac{L^2}{\nu_h}, \quad \nu_v = \frac{\mu_v}{\rho}, \quad \nu_h = \frac{\mu_h}{\rho}, \quad L = \frac{1}{2\pi}. \tag{II.15}$$

In Ref.[16], it is reported that their solver experiences a convergence difficulty on high-aspect-ratio grids and a modification to the length scale L (a local cell-aspect ratio incorporated in the denominator) was proposed to resolve the issue. However, we have actually never experienced such a convergence difficulty. In fact, an analysis for the model diffusion equation shows that it is a problem with explicit solvers and implicit solvers do not have issues with high-aspect-ratio grids. It should be noted also that the length scale L is a nondimensionalized length, i.e., nondimensionalized by the same reference length used to define the free stream Reynolds number Re_∞. In other words, it is defined per unit grid length. If the reference length is not unity in a given grid, the above L must be multiplied by the reference length measured in the grid unit. With the free stream Reynolds number Re_∞ adjusted in a similar manner (i.e., a target Reynolds number divided by the reference length in the grid unit), which is typically done in viscous simulations [29, 30], we will be solving the same problem and the results will be independent of the grid scaling. In the case that the NS system is solved without nondimensionalization, the length scale L needs to be carefully chosen such that L is a constant independent of a mesh spacing h [7], but L/h is sufficiently large. Further details of the HNS method for the dimensional NS system are left as future work.

Following Ref.[9], we write HNS17 as a preconditioned conservative system:

$$\mathbf{P}^{-1}\partial_\tau \mathbf{U} + \mathrm{div}\mathbf{F} = \mathbf{S}, \tag{II.16}$$

where, with the notation $\mathbf{g} = [\mathbf{g}_u, \mathbf{g}_v, \mathbf{g}_w]^t$,

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \rho E \\ \mathbf{g}_u \\ \mathbf{g}_v \\ \mathbf{g}_w \\ \mathbf{q} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho\mathbf{v}^t \\ \rho\mathbf{v}\otimes\mathbf{v} + p\mathbf{I} - \boldsymbol{\tau} \\ \rho\mathbf{v}^t H - (\boldsymbol{\tau}\mathbf{v})^t + \mathbf{q}^t \\ -u\mathbf{I} \\ -v\mathbf{I} \\ -w\mathbf{I} \\ \dfrac{T}{\gamma(\gamma-1)}\mathbf{I} \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 0 \\ \mathbf{0} \\ 0 \\ -\mathbf{g}_u/\mu_v \\ -\mathbf{g}_v/\mu_v \\ -\mathbf{g}_w/\mu_v \\ -\mathbf{q}/\mu_h \end{bmatrix}, \tag{II.17}$$

$$\mathbf{P}^{-1} = \mathrm{diag}\left(1,1,1,1,1,1,\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_h}{\mu_h},\frac{T_h}{\mu_h},\frac{T_h}{\mu_h}\right). \tag{II.18}$$

The local-preconditioning matrix, $\mathbf{P}$, has been introduced to simplify the construction of numerical schemes [9], not specifically to accelerate pseudo-steady convergence. Existing local-preconditioners [31–34] can be incorporated into $\mathbf{P}$ for accelerating convergence as well as preserving accuracy in low-Mach-number flows, and should be explored in future.

The projected flux $\mathbf{F}_n = \mathbf{F}\mathbf{n}$ in the direction of an arbitrary unit vector $\mathbf{n}$, which is relevant to the edge-based discretization, is written as a sum of the inviscid flux $\mathbf{F}_n^i$ and the viscous flux $\mathbf{F}_n^v$:

$$\mathbf{F}_n = \mathbf{F}_n^i + \mathbf{F}_n^v, \tag{II.19}$$

where

$$\mathbf{F}_n^i = \begin{bmatrix} \rho u_n \\ \rho u_n\mathbf{v} + p\mathbf{n} \\ \rho u_n H \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{F}_n^v = \begin{bmatrix} 0 \\ -\boldsymbol{\tau}_n \\ -\tau_{nv} + q_n \\ -u\mathbf{n} \\ -v\mathbf{n} \\ -w\mathbf{n} \\ \dfrac{T}{\gamma(\gamma-1)}\mathbf{n} \end{bmatrix}, \tag{II.20}$$

$$u_n = \mathbf{v}\cdot\mathbf{n}, \quad q_n = \mathbf{q}\cdot\mathbf{n}, \quad \boldsymbol{\tau}_n = \boldsymbol{\tau}\mathbf{n}, \quad \tau_{nv} = \boldsymbol{\tau}_n\cdot\mathbf{v}, \tag{II.21}$$

which are given, in the three-dimensional Cartesian coordinate system, with the following notations:

$$\mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \mathbf{g}_u & \mathbf{g}_v & \mathbf{g}_w \end{bmatrix}^t = \begin{bmatrix} g_{ux} & g_{uy} & g_{uz} \\ g_{vx} & g_{vy} & g_{vz} \\ g_{wx} & g_{wy} & g_{wz} \end{bmatrix}, \tag{II.22}$$

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}, \quad \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{II.23}$$

as

$$u_n = \mathbf{v}\cdot\mathbf{n} = un_x + vn_y + wn_z, \quad q_n = \mathbf{q}\cdot\mathbf{n} = q_xn_x + q_yn_y + q_zn_z, \tag{II.24}$$

$$\boldsymbol{\tau}_n = \begin{bmatrix} \tau_{nx} \\ \tau_{ny} \\ \tau_{nz} \end{bmatrix} = \boldsymbol{\tau}\mathbf{n} = \begin{bmatrix} \tau_{xx}n_x + \tau_{xy}n_y + \tau_{xz}n_z \\ \tau_{yx}n_x + \tau_{yy}n_y + \tau_{yz}n_z \\ \tau_{zx}n_x + \tau_{zy}n_y + \tau_{zz}n_z \end{bmatrix}, \quad \tau_{nv} = \boldsymbol{\tau}_n\cdot\mathbf{v} = \tau_{nx}u + \tau_{ny}v + \tau_{nz}w. \tag{II.25}$$

In principle, the wave structure of HNS17 can be analyzed by the projected Jacobian:

$$\mathbf{PA}_n = \mathbf{P}\frac{\partial\mathbf{F}_n}{\partial\mathbf{U}}. \tag{II.26}$$

However, it is very difficult, to say the least, to analyze the above Jacobian, and the full eigen-structure has not been found yet. To overcome the difficulty, a simplified approach was proposed in Ref.[9], where the Jacobian is split into the inviscid and viscous parts:

$$\mathbf{PA}_n = \mathbf{PA}_n^i + \mathbf{PA}_n^v, \tag{II.27}$$

or, equivalently, since $\mathbf{P}$ has no effect on the inviscid Jacobian,

$$\mathbf{PA}_n = \mathbf{A}_n^i + \mathbf{PA}_n^v, \tag{II.28}$$

where $\mathbf{A}_n^i$ and $\mathbf{A}_n^v$ are the flux Jacobians derived from the inviscid and viscous fluxes, respectively:

$$\mathbf{A}_n^i = \frac{\partial\mathbf{F}_n^i}{\partial\mathbf{U}}, \quad \mathbf{A}_n^v = \frac{\partial\mathbf{F}_n^v}{\partial\mathbf{U}}. \tag{II.29}$$

It is well known that the inviscid Jacobian has real eigenvalues and linearly independent eigenvectors, and therefore the inviscid part is hyperbolic in the pseudo time [35]. It can be shown that the viscous Jacobian also has real eigenvalues and linearly independent eigenvectors, and therefore the viscous part is also hyperbolic in the pseudo time [9, 16, 18]. Eigenvalues of $\mathbf{PA}_n^v$ are given by

$$\lambda_1 = -a_{nv}, \quad \lambda_2 = a_{nv}, \quad \lambda_3 = -a_{mv}, \quad \lambda_4 = a_{mv}, \tag{II.30}$$

$$\lambda_5 = -a_{lv}, \quad \lambda_6 = a_{lv}, \quad \lambda_7 = -a_h, \quad \lambda_8 = a_h, \tag{II.31}$$

and $\lambda_{9,\ldots,17} = 0$, where

$$a_{nv} = \sqrt{\frac{\nu_v}{T_v}}, \quad a_{mv} = a_{lv} = \sqrt{\frac{3\nu_v}{4T_v}}, \quad a_h = \sqrt{\frac{\nu_h}{T_h}}, \tag{II.32}$$

corresponding to the normal viscous, shear viscous, and heating waves, respectively [9]. For air, as pointed out in Ref.[9], the largest eigenvalue is given by a_h since

$$\frac{a_{nv}}{a_h} = \frac{4Pr}{3\gamma} < 1, \quad \frac{a_{mv}}{a_h} = \frac{a_{lv}}{a_h} = \frac{2\sqrt{3}Pr}{3\gamma} < 1, \tag{II.33}$$

for a wide range of temperature. Right eigenvectors associated with the non-zero eigenvalues, which are relevant to the construction of a numerical flux, are given by

$$\mathbf{r}_1 = \begin{bmatrix} 0 \\ \mathbf{n} \\ u_n + \dfrac{\tau_{nn} Pr_n}{\rho a_h (Pr_n^2 - 1)} \\ a_{nv} n_x \, \mathbf{n} \\ a_{nv} n_y \, \mathbf{n} \\ a_{nv} n_z \, \mathbf{n} \\ -\dfrac{\tau_{nn}}{\rho (Pr_n^2 - 1)} \mathbf{n} \end{bmatrix}, \quad \mathbf{r}_2 = \begin{bmatrix} 0 \\ \mathbf{n} \\ u_n - \dfrac{\tau_{nn} Pr_n}{\rho a_h (Pr_n^2 - 1)} \\ -a_{nv} n_x \, \mathbf{n} \\ -a_{nv} n_y \, \mathbf{n} \\ -a_{nv} n_z \, \mathbf{n} \\ -\dfrac{\tau_{nn}}{\rho (Pr_n^2 - 1)} \mathbf{n} \end{bmatrix}, \quad \mathbf{r}_3 = \begin{bmatrix} 0 \\ \mathbf{m} \\ u_m + \dfrac{\tau_{nm} Pr_m}{\rho a_h (Pr_m^2 - 1)} \\ \dfrac{4}{3} a_{mv} m_x \, \mathbf{n} \\ \dfrac{4}{3} a_{mv} m_y \, \mathbf{n} \\ \dfrac{4}{3} a_{mv} m_z \, \mathbf{n} \\ -\dfrac{\tau_{nm}}{\rho (Pr_m^2 - 1)} \mathbf{n} \end{bmatrix}, \tag{II.34}$$

$$\mathbf{r}_4 = \begin{bmatrix} 0 \\ \mathbf{m} \\ u_m - \dfrac{\tau_{nm} Pr_m}{\rho a_h (Pr_m^2 - 1)} \\ -\dfrac{4}{3} a_{mv} m_x \, \mathbf{n} \\ -\dfrac{4}{3} a_{mv} m_y \, \mathbf{n} \\ -\dfrac{4}{3} a_{mv} m_z \, \mathbf{n} \\ -\dfrac{\tau_{nm}}{\rho (Pr_m^2 - 1)} \mathbf{n} \end{bmatrix}, \quad \mathbf{r}_5 = \begin{bmatrix} 0 \\ \mathbf{l} \\ u_l + \dfrac{\tau_{nl} Pr_l}{\rho a_h (Pr_l^2 - 1)} \\ \dfrac{4}{3} a_{lv} l_x \, \mathbf{n} \\ \dfrac{4}{3} a_{lv} l_y \, \mathbf{n} \\ \dfrac{4}{3} a_{lv} l_z \, \mathbf{n} \\ -\dfrac{\tau_{nl}}{\rho (Pr_l^2 - 1)} \mathbf{n} \end{bmatrix}, \quad \mathbf{r}_6 = \begin{bmatrix} 0 \\ \mathbf{l} \\ u_l - \dfrac{\tau_{nl} Pr_l}{\rho a_h (Pr_l^2 - 1)} \\ -\dfrac{4}{3} a_{lv} l_x \, \mathbf{n} \\ -\dfrac{4}{3} a_{lv} l_y \, \mathbf{n} \\ -\dfrac{4}{3} a_{lv} l_z \, \mathbf{n} \\ -\dfrac{\tau_{nl}}{\rho (Pr_l^2 - 1)} \mathbf{n} \end{bmatrix}, \tag{II.35}$$

$$\mathbf{r}_7 = \begin{bmatrix} 0 \\ \mathbf{0} \\ -1 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ a_h \mathbf{n} \end{bmatrix}, \quad \mathbf{r}_8 = \begin{bmatrix} 0 \\ \mathbf{0} \\ 1 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ a_h \mathbf{n} \end{bmatrix}, \tag{II.36}$$

where $\mathbf{m} = [m_x, m_y, m_z]^t$ and $\mathbf{l} = [l_x, l_y, l_z]^t$ are unit vectors satisfying $\mathbf{m} \cdot \mathbf{l} = 0$, $\mathbf{m} \cdot \mathbf{n} = 0$, $\mathbf{l} \cdot \mathbf{n} = 0$, and

$$\tau_{nl} = \tau_{nx} l_x + \tau_{ny} l_y + \tau_{nz} l_z, \quad \tau_{nm} = \tau_{nx} m_x + \tau_{ny} m_y + \tau_{nz} m_z, \quad \tau_{nn} = \tau_{nx} n_x + \tau_{ny} n_y + \tau_{nz} n_z, \tag{II.37}$$

$$u_l = ul_x + vl_y + wl_z, \quad u_m = um_x + vm_y + wm_z, \tag{II.38}$$

$$Pr_l = \frac{a_{lv}}{a_h}, \quad Pr_m = \frac{a_{mv}}{a_h}, \quad Pr_n = \frac{a_{nv}}{a_h}. \tag{II.39}$$

The corresponding left eigenvectors, $\boldsymbol{\ell}_k^t$, $k = 1, 2, 3, 4, 5, 6, 7, 8$, can also be found:

$$\boldsymbol{\ell}_1 = \frac{1}{2}\begin{bmatrix} -u_n \\ \mathbf{n} \\ 0 \\ \dfrac{3n_x\mathbf{n} - \boldsymbol{\delta}_1}{2a_{nv}} \\ \dfrac{3n_y\mathbf{n} - \boldsymbol{\delta}_2}{2a_{nv}} \\ \dfrac{3n_z\mathbf{n} - \boldsymbol{\delta}_3}{2a_{nv}} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\ell}_2 = \frac{1}{2}\begin{bmatrix} -u_n \\ \mathbf{n} \\ 0 \\ \dfrac{-3n_x\mathbf{n} + \boldsymbol{\delta}_1}{2a_{nv}} \\ \dfrac{-3n_y\mathbf{n} + \boldsymbol{\delta}_2}{2a_{nv}} \\ \dfrac{-3n_z\mathbf{n} + \boldsymbol{\delta}_3}{2a_{nv}} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\ell}_3 = \frac{1}{2}\begin{bmatrix} -u_m \\ \mathbf{m} \\ 0 \\ \dfrac{3(m_x\mathbf{n} + n_x\mathbf{m})}{4a_{mv}} \\ \dfrac{3(m_y\mathbf{n} + n_y\mathbf{m})}{4a_{mv}} \\ \dfrac{3(m_z\mathbf{n} + n_z\mathbf{m})}{4a_{mv}} \\ \mathbf{0} \end{bmatrix}, \tag{II.40}$$

$$\boldsymbol{\ell}_4 = \frac{1}{2}\begin{bmatrix} -u_m \\ \mathbf{m} \\ 0 \\ \dfrac{-3(m_x\mathbf{n} + n_x\mathbf{m})}{4a_{mv}} \\ \dfrac{-3(m_y\mathbf{n} + n_y\mathbf{m})}{4a_{mv}} \\ \dfrac{-3(m_z\mathbf{n} + n_z\mathbf{m})}{4a_{mv}} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\ell}_5 = \frac{1}{2}\begin{bmatrix} -u_l \\ \mathbf{l} \\ 0 \\ \dfrac{3(l_x\mathbf{n} + n_x\mathbf{l})}{4a_{lv}} \\ \dfrac{3(l_y\mathbf{n} + n_y\mathbf{l})}{4a_{lv}} \\ \dfrac{3(l_z\mathbf{n} + n_z\mathbf{l})}{4a_{lv}} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\ell}_6 = \frac{1}{2}\begin{bmatrix} -u_l \\ \mathbf{l} \\ 0 \\ \dfrac{-3(l_x\mathbf{n} + n_x\mathbf{l})}{4a_{lv}} \\ \dfrac{-3(l_y\mathbf{n} + n_y\mathbf{l})}{4a_{lv}} \\ \dfrac{-3(l_z\mathbf{n} + n_z\mathbf{l})}{4a_{lv}} \\ \mathbf{0} \end{bmatrix}, \tag{II.41}$$

$$\boldsymbol{\ell}_7 = \frac{1}{2}\begin{bmatrix} \dfrac{a^2}{\gamma(\gamma-1)} - \dfrac{\mathbf{v}^2}{2} - \dfrac{1}{\rho a_h}\left(\dfrac{u_n\tau_{nn}}{Pr_n^2-1} + \dfrac{u_m\tau_{nm}}{Pr_m^2-1} + \dfrac{u_l\tau_{nl}}{Pr_l^2-1}\right) \\ \mathbf{v} + \dfrac{1}{\rho a_h}\left(\dfrac{\tau_{nn}}{Pr_n^2-1}\mathbf{n} + \dfrac{\tau_{nm}}{Pr_m^2-1}\mathbf{m} + \dfrac{\tau_{nl}}{Pr_l^2-1}\mathbf{l}\right) \\ -1 \\ \dfrac{3}{4\rho a_h^2}\left(\dfrac{2n_x\mathbf{n} - 2\boldsymbol{\delta}_1/3}{Pr_n^2-1}\tau_{nn} + \dfrac{m_x\mathbf{n} + n_x\mathbf{m}}{Pr_m^2-1}\tau_{nm} + \dfrac{l_x\mathbf{n} + n_x\mathbf{l}}{Pr_l^2-1}\tau_{nl}\right) \\ \dfrac{3}{4\rho a_h^2}\left(\dfrac{2n_y\mathbf{n} - 2\boldsymbol{\delta}_2/3}{Pr_n^2-1}\tau_{nn} + \dfrac{m_y\mathbf{n} + n_y\mathbf{m}}{Pr_m^2-1}\tau_{nm} + \dfrac{l_y\mathbf{n} + n_y\mathbf{l}}{Pr_l^2-1}\tau_{nl}\right) \\ \dfrac{3}{4\rho a_h^2}\left(\dfrac{2n_z\mathbf{n} - 2\boldsymbol{\delta}_3/3}{Pr_n^2-1}\tau_{nn} + \dfrac{m_z\mathbf{n} + n_z\mathbf{m}}{Pr_m^2-1}\tau_{nm} + \dfrac{l_z\mathbf{n} + n_z\mathbf{l}}{Pr_l^2-1}\tau_{nl}\right) \\ \dfrac{\mathbf{n}}{a_h} \end{bmatrix}, \tag{II.42}$$

$$
\boldsymbol{\ell}_8 = \frac{1}{2}\begin{bmatrix}
-\dfrac{a^2}{\gamma(\gamma-1)} + \dfrac{\mathbf{v}^2}{2} - \dfrac{1}{\rho a_h}\left(\dfrac{u_n\tau_{nn}}{Pr_n^2-1} + \dfrac{u_m\tau_{nm}}{Pr_m^2-1} + \dfrac{u_l\tau_{nl}}{Pr_l^2-1}\right) \\
-\mathbf{v} + \dfrac{1}{\rho a_h}\left(\dfrac{\tau_{nn}}{Pr_n^2-1}\mathbf{n} + \dfrac{\tau_{nm}}{Pr_m^2-1}\mathbf{m} + \dfrac{\tau_{nl}}{Pr_l^2-1}\mathbf{l}\right) \\
1 \\
-\dfrac{3}{4\rho a_h^2}\left(\dfrac{2n_x\mathbf{n} - 2\boldsymbol{\delta}_1/3}{Pr_n^2-1}\tau_{nn} + \dfrac{m_x\mathbf{n} + n_x\mathbf{m}}{Pr_m^2-1}\tau_{nm} + \dfrac{l_x\mathbf{n} + n_x\mathbf{l}}{Pr_l^2-1}\tau_{nl}\right) \\
-\dfrac{3}{4\rho a_h^2}\left(\dfrac{2n_y\mathbf{n} - 2\boldsymbol{\delta}_2/3}{Pr_n^2-1}\tau_{nn} + \dfrac{m_y\mathbf{n} + n_y\mathbf{m}}{Pr_m^2-1}\tau_{nm} + \dfrac{l_y\mathbf{n} + n_y\mathbf{l}}{Pr_l^2-1}\tau_{nl}\right) \\
-\dfrac{3}{4\rho a_h^2}\left(\dfrac{2n_z\mathbf{n} - 2\boldsymbol{\delta}_3/3}{Pr_n^2-1}\tau_{nn} + \dfrac{m_z\mathbf{n} + n_z\mathbf{m}}{Pr_m^2-1}\tau_{nm} + \dfrac{l_z\mathbf{n} + n_z\mathbf{l}}{Pr_l^2-1}\tau_{nl}\right) \\
\dfrac{\mathbf{n}}{a_h}
\end{bmatrix}, \tag{II.43}
$$

where $\boldsymbol{\delta}_1 = [1,0,0]^t$, $\boldsymbol{\delta}_2 = [0,1,0]^t$, $\boldsymbol{\delta}_3 = [0,0,1]^t$, and $a^2 = \gamma p/\rho$. It is remarked that the 4th, 5th, and 6th elements of the left eigenvectors expressed as above are the column vectors of the following matrices:

$$
\pm\frac{1}{4a_{nv}}\left(3\mathbf{n}\otimes\mathbf{n} - \mathbf{I}\right), \quad \text{for } \boldsymbol{\ell}_{1/2}, \tag{II.44}
$$

$$
\pm\frac{3}{8a_{mv}}\left(\mathbf{m}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{m}\right), \quad \text{for } \boldsymbol{\ell}_{3/4}, \tag{II.45}
$$

$$
\pm\frac{3}{8a_{lv}}\left(\mathbf{l}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{l}\right), \quad \text{for } \boldsymbol{\ell}_{5/6}, \tag{II.46}
$$

$$
\pm\frac{3}{8\rho a_h^2}\left(\frac{2\mathbf{n}\otimes\mathbf{n} - \frac{2}{3}\mathbf{I}}{Pr_n^2-1}\tau_{nn} + \frac{\mathbf{m}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{m}}{Pr_m^2-1}\tau_{nm} + \frac{\mathbf{l}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{l}}{Pr_l^2-1}\tau_{nl}\right), \quad \text{for } \boldsymbol{\ell}_{7/8}. \tag{II.47}
$$

The characteristic variables are obtained as follows:

$$
\boldsymbol{\ell}_1^t d\mathbf{U} = \frac{\rho\, du_n}{2} + \frac{d\tau_{nn}}{2a_{nv}}, \quad \boldsymbol{\ell}_2^t d\mathbf{U} = \frac{\rho\, du_n}{2} - \frac{d\tau_{nn}}{2a_{nv}}, \tag{II.48}
$$

$$
\boldsymbol{\ell}_3^t d\mathbf{U} = \frac{\rho\, du_m}{2} + \frac{d\tau_{nm}}{2a_{mv}}, \quad \boldsymbol{\ell}_4^t d\mathbf{U} = \frac{\rho\, du_m}{2} - \frac{d\tau_{nm}}{2a_{mv}}, \tag{II.49}
$$

$$
\boldsymbol{\ell}_5^t d\mathbf{U} = \frac{\rho\, du_l}{2} + \frac{d\tau_{nl}}{2a_{lv}}, \quad \boldsymbol{\ell}_6^t d\mathbf{U} = \frac{\rho\, du_l}{2} - \frac{d\tau_{nl}}{2a_{lv}}, \tag{II.50}
$$

$$
\boldsymbol{\ell}_7^t d\mathbf{U} = \frac{1}{2a_h}\left[dq_n - \frac{\rho a_h dT}{\gamma(\gamma-1)} + \frac{\tau_{nn}}{Pr_n^2-1}\left(du_n + \frac{d\tau_{nn}}{\rho a_h}\right) + \frac{\tau_{nm}}{Pr_m^2-1}\left(du_m + \frac{d\tau_{nm}}{\rho a_h}\right) + \frac{\tau_{nl}}{Pr_l^2-1}\left(du_l + \frac{d\tau_{nl}}{\rho a_h}\right)\right], \tag{II.51}
$$

$$
\boldsymbol{\ell}_8^t d\mathbf{U} = \frac{1}{2a_h}\left[dq_n + \frac{\rho a_h dT}{\gamma(\gamma-1)} + \frac{\tau_{nn}}{Pr_n^2-1}\left(du_n - \frac{d\tau_{nn}}{\rho a_h}\right) + \frac{\tau_{nm}}{Pr_m^2-1}\left(du_m - \frac{d\tau_{nm}}{\rho a_h}\right) + \frac{\tau_{nl}}{Pr_l^2-1}\left(du_l - \frac{d\tau_{nl}}{\rho a_h}\right)\right], \tag{II.52}
$$

where

$$
dT = \frac{\gamma dp - a^2 d\rho}{\rho}. \tag{II.53}
$$

The linearly independent eigenvectors associated with the zero eigenvalues can also be found but not needed for constructing a numerical flux, and therefore not shown. It may be noted that some of the eigenvectors are unbounded for $Pr_n = 1$ or $Pr_m = Pr_l = 1$, but that is not the case for air because of Equation (II.33).

In fact, there is no concern for any gas because the singularity will disappear in the final form of a numerical flux as we will show later, and also in the absolute Jacobian as shown below.

The left and right eigenvectors involve the vectors $\mathbf{m}$ and $\mathbf{l}$. These vectors are defined just as orthogonal to each other and orthogonal to $\mathbf{n}$, and not uniquely defined for a given $\mathbf{n}$. It is often convenient to eliminate these vectors from numerical schemes, where only $\mathbf{n}$ can be defined clearly (e.g., the unit vector normal to a face of a computational cell). The elimination is actually possible by using the properties of three mutually orthogonal vectors (See Ref.[28]). For example, the absolute Jacobian can be expressed in terms of $\mathbf{n}$ only:

$$
\begin{aligned}
|\mathbf{PA}_n^v| &= \sum_{k=1}^{8} |\lambda_k|\, \mathbf{r}_k \boldsymbol{\ell}_k^t \\
&= \begin{bmatrix}
0 & \mathbf{0}^t & 0 & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t \\
-u_n \Delta a_{nm}\mathbf{n} - a_{mv}\mathbf{v} & \Delta a_{nm}\mathbf{n}\otimes\mathbf{n} + a_{mv}\mathbf{I} & 0 & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\
-u_n^2 \Delta a_{nm} - a_{mv}\mathbf{v}^2 + a_h(E - \mathbf{v}^2) & \Delta a_{nm}\mathbf{n}^t - \Delta a_{hm}\mathbf{v}^t & a_h & \mathbf{a}_u^t & \mathbf{a}_v^t & \mathbf{a}_w^t & \mathbf{0}^t \\
\mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{B}_u & \mathbf{B}_v & \mathbf{B}_w & \mathbf{O} \\
\mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{C}_u & \mathbf{C}_v & \mathbf{C}_w & \mathbf{O} \\
\mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{D}_u & \mathbf{D}_v & \mathbf{D}_w & \mathbf{O} \\
\mathbf{z} & \mathbf{Z} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & a_h\mathbf{n}\otimes\mathbf{n}
\end{bmatrix},
\end{aligned}
\tag{II.54}
$$

where $\mathbf{O} = \mathbf{0}\otimes\mathbf{0}$, and

$$
\Delta a_{nm} = a_{nv} - a_{mv}, \quad \Delta a_{hm} = a_h - a_{mv}, \tag{II.55}
$$

$$
[\mathbf{a}_u, \mathbf{a}_v, \mathbf{a}_w] = \frac{1}{\rho a_h}\left\{\frac{\tau_{nn}}{2}\frac{3\mathbf{n}\otimes\mathbf{n} - \mathbf{I}}{Pr_n + 1} + \frac{3}{4}\frac{\mathbf{n}\otimes\boldsymbol{\tau}_n - 2\mathbf{n}\otimes\mathbf{n} + \boldsymbol{\tau}_n\otimes\mathbf{n}}{Pr_m + 1}\right\}, \tag{II.56}
$$

$$
\mathbf{B}_u = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_x\mathbf{n}\otimes(3n_x\mathbf{n} - \boldsymbol{\delta}_1) + a_{mv}\mathbf{n}\otimes\left(\mathbf{n} + \frac{1}{3}n_x\boldsymbol{\delta}_1\right), \tag{II.57}
$$

$$
\mathbf{B}_v = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_x\mathbf{n}\otimes(3n_y\mathbf{n} - \boldsymbol{\delta}_2) + a_{mv}\mathbf{n}\otimes\left(n_y\boldsymbol{\delta}_1 - \frac{2}{3}n_x\boldsymbol{\delta}_2\right), \tag{II.58}
$$

$$
\mathbf{B}_w = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_x\mathbf{n}\otimes(3n_z\mathbf{n} - \boldsymbol{\delta}_3) + a_{mv}\mathbf{n}\otimes\left(n_z\boldsymbol{\delta}_1 - \frac{2}{3}n_x\boldsymbol{\delta}_3\right), \tag{II.59}
$$

$$
\mathbf{C}_u = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_y\mathbf{n}\otimes(3n_x\mathbf{n} - \boldsymbol{\delta}_1) + a_{mv}\mathbf{n}\otimes\left(n_x\boldsymbol{\delta}_2 - \frac{2}{3}n_y\boldsymbol{\delta}_1\right), \tag{II.60}
$$

$$
\mathbf{C}_v = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_y\mathbf{n}\otimes(3n_y\mathbf{n} - \boldsymbol{\delta}_2) + a_{mv}\mathbf{n}\otimes\left(\mathbf{n} + \frac{1}{3}n_y\boldsymbol{\delta}_2\right), \tag{II.61}
$$

$$
\mathbf{C}_w = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_y\mathbf{n}\otimes(3n_z\mathbf{n} - \boldsymbol{\delta}_3) + a_{mv}\mathbf{n}\otimes\left(n_z\boldsymbol{\delta}_2 - \frac{2}{3}n_y\boldsymbol{\delta}_3\right), \tag{II.62}
$$

$$\mathbf{D}_u = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_z \mathbf{n}\otimes(3n_x\mathbf{n} - \boldsymbol{\delta}_1) + a_{mv}\mathbf{n}\otimes\left(n_x\boldsymbol{\delta}_3 - \frac{2}{3}n_z\boldsymbol{\delta}_1\right), \tag{II.63}$$

$$\mathbf{D}_v = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_z \mathbf{n}\otimes(3n_y\mathbf{n} - \boldsymbol{\delta}_2) + a_{mv}\mathbf{n}\otimes\left(n_y\boldsymbol{\delta}_3 - \frac{2}{3}n_z\boldsymbol{\delta}_2\right), \tag{II.64}$$

$$\mathbf{D}_w = \frac{1}{2}\left(a_{nv} - \frac{4}{3}a_{mv}\right) n_z \mathbf{n}\otimes(3n_z\mathbf{n} - \boldsymbol{\delta}_3) + a_{mv}\mathbf{n}\otimes\left(\mathbf{n} + \frac{1}{3}n_z\boldsymbol{\delta}_3\right), \tag{II.65}$$

$$\mathbf{z} = \frac{a_h}{\rho}\left(\frac{\tau_{nn}u_n}{Pr_n+1} + \frac{\tau_{nv} - \tau_{nn}u_n}{Pr_m+1}\right)\mathbf{n}, \tag{II.66}$$

$$\mathbf{Z} = \frac{a_h}{\rho}\left(-\frac{\mathbf{n}\otimes\boldsymbol{\tau}_n}{Pr_n+1} - \frac{\tau_{nn}\mathbf{n}\otimes\mathbf{n} - \mathbf{n}\otimes\boldsymbol{\tau}_n}{Pr_m+1}\right). \tag{II.67}$$

Observe that the vectors $\mathbf{m}$ and $\mathbf{l}$ have been completely eliminated. In the construction of numerical schemes, the left and right eigenvectors are usually not explicitly required; it is the absolute Jacobian $|\mathbf{PA}_n^v|$ that is often needed. Therefore, many numerical schemes can be constructed without $\mathbf{m}$ and $\mathbf{l}$. Note also that the terms $\frac{1}{Pr_n^2-1}$, $\frac{1}{Pr_m^2-1}$, and $\frac{1}{Pr_l^2-1}$ have disappeared.

Since the inviscid and viscous terms are both hyperbolic in the pseudo time, and numerical schemes can be constructed straightforwardly and in the same way for both. Based on the individual eigen-structure, the viscous scheme can be constructed independently from the inviscid terms; and thus any inviscid scheme can be employed in the HNS method. This strategy was first introduced for the NS equations in Ref.[9], and successfully applied in the subsequent studies [16, 18]. Note that independent construction of inviscid and viscous terms is common in many numerical methods for the NS equations; interactions of the inviscid and viscous terms are rarely taken into account in numerical methods. It would be very interesting and might be useful if a full eigen-structure of the full HNS Jacobian (II.26) were found. However, it is not necessary for constructing numerical schemes.

Finally, we remark that the HNS17 system is a building block for a more advanced system, HNS20. To extend it to HNS20, we only need to add an artificial hyperbolic-mass-diffusion system [18], which can be analyzed and discretized independently from other terms. Hence, the eigen-structure of the HNS17 system as presented above is directly relevant to the HNS20 system. To construct an HNS20 scheme, we construct a numerical scheme for the artificial hyperbolic-mass-diffusion system independently, and add it to an HNS17 scheme [14].

III. Advanced Hyperbolic Compressible NS System: HNS20

The HNS20 system was introduced in Ref.[18] in order to to obtain accurate density gradients, thereby achieving accurate gradients for all the primitive variables. It is constructed by adding an artificial hyperbolic-mass-diffusion system to HNS17 [18]:

$$\partial_\tau\rho + \operatorname{div}(\rho\mathbf{v}) = \operatorname{div}\mathbf{r}, \tag{III.1}$$

$$\partial_\tau(\rho\mathbf{v}) + \operatorname{div}(\rho\mathbf{v}\otimes\mathbf{v}) = -\operatorname{grad}p + \operatorname{div}\boldsymbol{\tau}, \tag{III.2}$$

$$\partial_\tau(\rho E) + \operatorname{div}(\rho\mathbf{v}H) = \operatorname{div}(\boldsymbol{\tau}\mathbf{v}) - \operatorname{div}\mathbf{q}, \tag{III.3}$$

$$\frac{T_v}{\mu_v}\partial_\tau\mathbf{g} = \operatorname{grad}\mathbf{v} - \frac{\mathbf{g}}{\mu_v}, \tag{III.4}$$

$$\frac{T_h}{\mu_h}\partial_\tau\mathbf{q} = -\frac{1}{\gamma(\gamma-1)}\operatorname{grad}T - \frac{\mathbf{q}}{\mu_h}, \tag{III.5}$$

$$\frac{T_\rho}{\nu_\rho}\partial_\tau\mathbf{r} = \operatorname{grad}\rho - \frac{\mathbf{r}}{\nu_\rho}, \tag{III.6}$$

where ν_ρ is an artificial hyperbolic-mass-diffusion coefficient, $T_\rho = L^2/\nu_\rho$, and $\mathbf{r}/\nu_\rho$ is a vector of gradient variables equivalent to the density gradient in the pseudo steady state or as soon as we set $\partial_\tau \mathbf{r} = 0$. The artificial mass diffusion term in the continuity equation and Equation (III.6) form a hyperbolic diffusion system. In Ref.[18], the artificial hyperbolic-mass-diffusion coefficient ν_ρ was given a nearly machine-zero level value of 10^{-12} so that it will not affect the continuity equation. In this work, it is defined as the minimum control volume:

$$\nu_\rho = V_{min}, \tag{III.7}$$

where V_{min} denotes the minimum dual control volume for a given grid. In this way, the coefficient is in the level of $O(h^3)$, which is comparable to or smaller than the truncation error of third- or second-order discretizations, respectively. Thus, the mass diffusion term is artificial and will not affect the continuity equation. There exists a formulation of the continuity equation with a mass (or volume, to be more precise) diffusion term, called Brenner's modification [36, 37]. It is noted that Brenner's modification has been introduced for a physical reason, but the mass diffusion in HNS20 has been introduced for a sole purpose of computing accurate density gradients and it is meaningful only in the form of a hyperbolic diffusion system. If Brenner's modification were employed, then the diffusive term could be directly reformulated as a hyperbolic diffusion system without introducing an artificial diffusion. While the artificial mass diffusion term is made negligibly small, the artificial hyperbolic-mass-diffusion system allows us to obtain density gradients to the same order of accuracy as that of other variables as demonstrated in Ref.[18]. And therefore, we obtain accurate gradients for all the primitive variables in the pseudo steady state or when the pseudo time derivatives are dropped:

$$\operatorname{grad}\rho = \frac{\mathbf{r}}{\nu_\rho}, \quad \operatorname{grad}\mathbf{v} = \frac{\mathbf{g}}{\mu_v}, \quad \operatorname{grad}T = -\frac{\gamma(\gamma-1)\mathbf{q}}{\mu_h}. \tag{III.8}$$

For second-order schemes, these gradients will be obtained with second-order accuracy on arbitrary unstructured grids. This is a special feature of the HNS method. In conventional second-order methods, the gradients, which are computed from numerical solutions, are inherently first-order accurate on general unstructured grids. To achieve second-order accuracy in the gradients, the algorithm for computing the gradients, such as a least-squares (LSQ) method, must be exact for quadratic functions. However, the quadratic exactness cannot be achieved because the numerical solution obtained by a second-order method (which is typically exact only for linear functions) is not exact for quadratic functions in the first place. Hence, any high-order LSQ fit will not be able to produce second-order gradients as long as it is applied to second-order solutions. See [16] for such results for the Navier-Stokes computations. On the other hand, the gradients obtained by the HNS method are not reconstructed from numerical solutions, rather they are computed as numerical solutions. In the case of a second-order HNS scheme, they are computed directly by a second-order numerical scheme, not by a recovery algorithm, and therefore achieve second-order accuracy. Second-order accurate gradients are useful also for obtaining accurate second-order derivatives (Hessian). Second-order derivatives are often employed to guide anisotropic grid adaptation [38–40], but these quantities are zero-th order accurate when computed from second-order solutions [21]. This is because Hessian recovery algorithms must be exact for quadratic functions to achieve first-order accuracy but second-order numerical solutions are generally exact only for linear functions. On the other hand, in the HNS method, a first-order accurate Hessian can be obtained simply by applying a linear LSQ fit to the second-order gradients. Hessian computed by a LSQ fit can be noisy on irregular grids, but as it is first-order accurate we can expect accurate estimates on fine or adapted grids whereas accurate estimates cannot be expected on any grid in conventional methods because it is a zero-th order quantity.

Accurate gradients for the primitive variables are useful not only for predicting derivative quantities (e.g., the viscous stresses, the vorticity, and the heat fluxes) and second-order derivatives, but also for improving inviscid schemes. The inviscid terms depend only on the primitive variables, and therefore the discretization can be improved by utilizing the accurate gradients of the primitive variables. In the case of the node-centered edge-based discretization, it is known that third-order accuracy can be achieved with second-order gradients and a linear flux extrapolation on triangular and tetrahedral grids [41]. Therefore, the inviscid part of a second-order HNS scheme can be immediately upgraded to third-order. Third-order accuracy in the inviscid terms has already been demonstrated for a second-order HNS scheme in two dimensions [18]. For other discretization methods, a similar upgrade is possible. For example, in cell-centered finite-volume methods, a quadratic polynomial can be constructed for the primitive variables within a cell by using the accurate

gradients and their LSQ gradients used as second derivatives. Naturally, the same applies to discontinuous Galerkin methods, which may be considered as an extension of the cell-centered finite-volume methods.

As mentioned in the previous section, the HNS20 system can be conveniently analyzed and discretized by treating the HNS17 part and the artificial hyperbolic-mass-diffusion part separately. Write the HNS20 system in the vector form:

$$\mathbf{P}^{-1}\partial_\tau \mathbf{U} + \mathrm{div}\mathbf{F} = \mathbf{S}, \tag{III.9}$$

where

$$\mathbf{P}^{-1} = \mathrm{diag}\left(1,1,1,1,1,\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_v}{\mu_v},\frac{T_h}{\mu_h},\frac{T_h}{\mu_h},\frac{T_h}{\mu_h},\frac{T_\rho}{\nu_\rho},\frac{T_\rho}{\nu_\rho},\frac{T_\rho}{\nu_\rho}\right), \tag{III.10}$$

$$\mathbf{U}=\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \rho E\\ \mathbf{g}_u\\ \mathbf{g}_v\\ \mathbf{g}_w\\ \mathbf{q}\\ \mathbf{r}\end{bmatrix},\quad \mathbf{F}=\begin{bmatrix}\rho\mathbf{v}^t-\mathbf{r}^t\\ \rho\mathbf{v}\otimes\mathbf{v}+p\mathbf{I}-\boldsymbol{\tau}\\ \rho\mathbf{v}^tH-(\boldsymbol{\tau}\mathbf{v})^t+\mathbf{q}^t\\ -u\mathbf{I}\\ -v\mathbf{I}\\ -w\mathbf{I}\\ \frac{T}{\gamma(\gamma-1)}\mathbf{I}\\ -\rho\mathbf{I}\end{bmatrix},\quad \mathbf{S}=\begin{bmatrix}0\\ \mathbf{0}\\ 0\\ -\mathbf{g}_u/\mu_v\\ -\mathbf{g}_v/\mu_v\\ -\mathbf{g}_w/\mu_v\\ -\mathbf{q}/\mu_h\\ -\mathbf{r}/\nu_\rho\end{bmatrix}. \tag{III.11}$$

The projected flux is given by

$$\mathbf{F}_n = \mathbf{F}_n^i + \mathbf{F}_n^v + \mathbf{F}_n^a, \tag{III.12}$$

where $\mathbf{F}_n^i$ and $\mathbf{F}_n^v$ are the inviscid and viscous fluxes exactly as in the HNS17 system except that they have now three additional zero entries in the 18th, 19th, and 20th components, and $\mathbf{F}_n^a$ denotes the contribution from the artificial hyperbolic-mass-diffusion system:

$$\mathbf{F}_n^i=\begin{bmatrix}\rho u_n\\ \rho u_n\mathbf{v}+p\mathbf{n}\\ \rho u_n H\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix},\quad \mathbf{F}_n^v=\begin{bmatrix}0\\ -\boldsymbol{\tau}_n\\ -\tau_{nv}+q_n\\ -u\mathbf{n}\\ -v\mathbf{n}\\ -w\mathbf{n}\\ \frac{T}{\gamma(\gamma-1)}\mathbf{n}\\ \mathbf{0}\end{bmatrix},\quad \mathbf{F}_n^a=\begin{bmatrix}-r_n\\ \mathbf{0}\\ 0\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ -\rho\mathbf{n}\end{bmatrix}, \tag{III.13}$$

where

$$r_n = \mathbf{r}\cdot\mathbf{n} = (r_x, r_y, r_z)\cdot(n_x, n_y, n_z) = r_x n_x + r_y n_y + r_z n_z. \tag{III.14}$$

The viscous flux $\mathbf{F}_n^v$ has already been analyzed in the previous section. Here, we focus on the artificial hyperbolic-mass-diffusion part. The preconditioned flux Jacobian for $\mathbf{F}_n^a$ is given by

$$\mathbf{P}\mathbf{A}_n^a = \mathbf{P}\frac{\partial \mathbf{F}_n^a}{\partial \mathbf{U}}, \tag{III.15}$$

which has the following two non-zero eigenvalues:

$$\pm\sqrt{\frac{\nu_\rho}{T_\rho}}, \tag{III.16}$$

and all other eigenvalues are zero. As the eigenvalues indicate, the artificial hyperbolic-mass-diffusion system is a straightforward extension of the two-dimensional hyperbolic diffusion system [14] to three dimensions. Therefore, the corresponding eigenvectors can be found straightforwardly, and the absolute Jacobian, which is relevant to the construction of numerical schemes, is obtained as

$$|\mathbf{PA}_n^a| = \frac{\nu_\rho}{L}\begin{bmatrix} 1 & \mathbf{0}^t & 0 & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ 0 & \mathbf{0}^t & 0 & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{n}\otimes\mathbf{n} \end{bmatrix}. \tag{III.17}$$

In the node-centered edge-based discretization, an upwind flux can be constructed with $|\mathbf{PA}_n^a|$ and added to the inviscid and viscous fluxes of HNS17. It is therefore very simple to extend an HNS17 scheme to HNS20. Note that the matrix $|\mathbf{PA}_n^a|$ looks singular, but it is not singular as a 4×4 matrix for the relevant variables (ρ, r_x, r_y, r_z) and thus can be inverted if necessary.

IV. Node-Centered Edge-Based Discretization

In this section, we describe the discretization of the HNS20 system. All vectors and matrices below refer to Section III unless otherwise stated. The discretization of the HNS17 system can be obtained by ignoring the last three components in all vectors and matrices, and the artificial mass diffusion term in the continuity equation.

IV.A. Discretization

The three-dimensional HNS20 system is discretized in space by the node-centered edge-based method on a general unstructured grid [42, 43]. The spatial residual at node j is obtained as

$$\mathbf{Res}_j = \mathbf{P}_j\left(\sum_{k\in\{k_j\}} \mathbf{\Phi}_{jk}A_{jk} - \mathbf{S}_j V_j\right), \tag{IV.1}$$

where $\mathbf{P}_j$ is the local preconditioning matrix (III.10) evaluated at the node j, V_j is the measure of the dual control volume around the node j, $\{k_j\}$ is a set of edge-connected neighbors of j, $\mathbf{\Phi}_{jk}$ is a numerical flux in the direction of the directed area vector $\mathbf{n}_{jk}$, and A_{jk} is the magnitude of the directed area vector:

$$A_{jk} = |\mathbf{n}_{jk}|. \tag{IV.2}$$

In the rest of the paper, we drop the subscript and denote the unit directed area vector by $\mathbf{n}$:

$$\mathbf{n} = \mathbf{n}_{jk}/A_{jk}. \tag{IV.3}$$

The directed area vector is defined per edge as a sum of the triangular dual-volume faces, each of which is defined by the edge midpoint, the centroid of a face of an adjacent element having the edge as a side,

and the centroid of the element (see, e.g., figures in Appendix B of Ref.[43]). The dual control volume is, therefore, a polyhedron with many faces. However, we do not treat each dual-volume face individually, but lump them into one around each edge and define the directed area vector. This process yields a highly efficient discretization with a single numerical flux evaluation per edge, called the edge-based discretization [17, 41, 44–47]. The order of accuracy of the edge-based discretization is truly second-order on tetrahedral grids, but may deteriorate to first-order for hexahedral, prismatic, pyramidal, or mixed grids without certain geometrical regularities [43, 48] although it is typically still much more accurate than first-order schemes by the effect of a linear solution reconstruction to the face.

The order of accuracy of the edge-based discretization is largely dictated by the edge-based formula with the lumped dual-volume face normals, and thus higher-order extensions are not straightforward. A recent study by Katz and Sankaran [41], however, revealed that the node-centered edge-based discretization achieves third-order accuracy for hyperbolic conservation laws on triangular grids with two minor modifications: second-order gradients and linear flux extrapolation. In three dimensions, third-order accuracy can be achieved on tetrahedral grids. This is a highly efficient third-order scheme compared with other third-order methods because the residual is computed in a loop over edges with a single numerical flux evaluation per edge. The third-order edge-based scheme has been extended to the NS equations by third-order gradients (with a cubic LSQ fit) and high-order elements in Ref.[49], and by the HNS method in Refs.[16, 18]. The present work is closely related to the latter, where the third-order edge-based scheme is directly applicable without major algorithmic changes (because the viscous terms are reformulated as a hyperbolic system).

To close the residual at a boundary node, a suitable boundary-flux quadrature formula, depending on the element type, needs to be employed in order to preserve the design order of accuracy. The boundary flux quadrature is perhaps a unique feature of the node-centered edge-based discretization. Note that the boundary flux is not merely integrated over the boundary faces, but it has to be approximated in such a way as to guarantee that the resulting residual at the boundary node is exact for linear fluxes to guarantee second-order accuracy. Consequently, the boundary flux quadrature is not independent of the interior edge-based quadrature, and thus highly dependent on the grid element type. A comprehensive list of boundary-flux quadrature formulas, which preserve second-order accuracy, in two (triangles, quadrilaterals, and mixed) and three dimensions (tetrahedra, hexahedra, prisms, pyramids, and mixed) is given in Appendix B of Ref.[43]. Formulas that can preserve third-order accuracy has been derived for triangular and tetrahedral grids in Ref.[17]. In this work, we employ a second-order formula, but plan to implement the third-order formula in future.

Boundary conditions are imposed with a combination of weak and strong procedures as described in Ref.[16]. We first compute the residual at all boundary nodes with boundary conditions imposed on the right (ghost) state that is sent to a numerical flux [18]. At an inflow boundary, the right state is set by the free stream condition; at an outflow boundary, the right state is set by the free stream pressure and other variables as the same as the left state (i.e., the values at a boundary node); at a viscous wall, the right state is set by zero velocity components, the heat flux $\mathbf{q}$ and the density gradient $\mathbf{r}$ with zero normal components, and the velocity gradient $\mathbf{g}$ with zero tangential components. Note that the zero normal density gradient at a viscous wall is consistent with the zero normal temperature gradient (adiabatic condition) and the zero normal pressure gradient from a laminar boundary layer theory [50]. After the residuals are computed, we strongly impose the no-slip condition and the zero normal or tangential gradient conditions by replacing residual components by zero-velocity, zero normal heat flux and density gradients, and tangential velocity gradients equations. Note that these equations are all algebraic equations imposed with the gradient variables, and there is no need to discretize derivatives at a boundary node. The equations for the normal velocity gradient, tangential heat flux, and density gradient equations are formed by projecting their original residuals in the corresponding directions. Further study on boundary conditions is left as a subject for future research.

IV.B. Numerical Flux

The numerical flux is constructed as a sum of the upwind inviscid flux, the upwind viscous flux, and the upwind artificial mass diffusion flux [18]:

$$\mathbf{\Phi}_{jk} = \mathbf{\Phi}^i_{jk} + \mathbf{\Phi}^v_{jk} + \mathbf{\Phi}^a_{jk}, \tag{IV.4}$$

$$\mathbf{\Phi}^i_{jk} = \frac{1}{2}\left[\mathbf{F}^i_n(\mathbf{U}_L) + \mathbf{F}^i_n\mathbf{U}_R)\right] - \frac{1}{2}\left|\mathbf{A}^i_n\right|\Delta\mathbf{U}, \tag{IV.5}$$

$$\mathbf{\Phi}^v_{jk} = \frac{1}{2}\left[\mathbf{F}^v_n(\mathbf{U}_L) + \mathbf{F}^v_n(\mathbf{U}_R)\right] - \frac{1}{2}\mathbf{P}^{-1}\left(|\mathbf{PA}^v_n| + |\mathbf{PA}^{av}_n|\right)\Delta\mathbf{U}, \tag{IV.6}$$

$$\mathbf{\Phi}^a_{jk} = \frac{1}{2}\left[\mathbf{F}^a_n(\mathbf{U}_L) + \mathbf{F}^a_n(\mathbf{U}_R)\right] - \frac{1}{2}\mathbf{P}^{-1}\left|\mathbf{PA}^a_n\right|\Delta\mathbf{U}, \tag{IV.7}$$

where the subscripts L and R indicate the values at the left and right sides of the edge midpoint, and

$$\Delta\mathbf{U} = \mathbf{U}_R - \mathbf{U}_L. \tag{IV.8}$$

The dissipation terms in $\mathbf{\Phi}^v_{jk}$ and $\mathbf{\Phi}^a_{jk}$ are constructed by following a standard procedure in the local-preconditioning method [32], i.e., constructed with the preconditioned Jacobian and multiplied by $\mathbf{P}^{-1}$ to cancel the effect of $\mathbf{P}$ in Equation (IV.1). For second-order accuracy, it suffices to evaluate the left and right fluxes by the extrapolated solution values as indicated in the above equations. However, we actually perform a linear flux extrapolation for the inviscid flux to further improve the accuracy as we will explain later. It should be pointed out at this point that the central flux, although seemingly legitimate for diffusion, should not be used for the viscous fluxes because it is known to yield a conventional viscous discretization and thus destroy all the benefits of the HNS method [14]. In other words, the viscous discretization is more accurate and in particular the gradients are one-order-higher accurate with upwind fluxes; the dissipative terms provide a strong coupling between the primitive and gradient variables, which is essential to preserve the benefits of the HNS method. Upwind schemes are typically considered as dissipative and less accurate than central schemes, but it does not apply to the hyperbolized viscous terms.

The upwind viscous flux has two contributions in the dissipation term: $|\mathbf{PA}^v_n|\,\Delta\mathbf{U}$ and $|\mathbf{PA}^{av}_n|\,\Delta\mathbf{U}$. The first one is the upwind viscous dissipation for the HNS17 system, which is constructed by λ_k, $\boldsymbol{\ell}^t_k\Delta\mathbf{U}$, and $\mathbf{r}_k$ from Section II with the right-eigenvector $\mathbf{r}_k$ padded with three zeroes:

$$|\mathbf{PA}^v_n|\,\Delta\mathbf{U} = \sum_{k=1}^{8}|\lambda_k|\left(\boldsymbol{\ell}^t_k\Delta\mathbf{U}\right)\mathbf{r}_k, \tag{IV.9}$$

which simplifies to

$$|\mathbf{PA}^v_n|\Delta\mathbf{U} = \begin{bmatrix} 0 \\ \rho\{(a_{nv}-a_{mv})\Delta u_n\mathbf{n} + a_{mv}\Delta\mathbf{v}\} \\ \rho\mathbf{v}\cdot\{(a_{nv}-a_{mv})\Delta u_n\mathbf{n}+a_{mv}\Delta\mathbf{v}\} + \dfrac{\rho a_h\Delta T}{\gamma(\gamma-1)} + \dfrac{1}{\rho a_h}\left\{\dfrac{\tau_{nn}\Delta\tau_{nn}}{Pr_n+1} + \dfrac{\boldsymbol{\tau}_n\cdot\Delta\boldsymbol{\tau}_n - \tau_{nn}\Delta\tau_{nn}}{Pr_m+1}\right\} \\ \left\{\left(a_{nv} - \dfrac{4}{3}a_{mv}\right)\Delta\tau_{nn}n_x + \dfrac{4}{3}a_{mv}\Delta\tau_{nx}\right\}\mathbf{n} \\ \left\{\left(a_{nv} - \dfrac{4}{3}a_{mv}\right)\Delta\tau_{nn}n_y + \dfrac{4}{3}a_{mv}\Delta\tau_{ny}\right\}\mathbf{n} \\ \left\{\left(a_{nv} - \dfrac{4}{3}a_{mv}\right)\Delta\tau_{nn}n_z + \dfrac{4}{3}a_{mv}\Delta\tau_{nz}\right\}\mathbf{n} \\ a_h\left(\Delta q_n - \dfrac{\tau_{nn}\Delta u_n}{Pr_n+1} - \dfrac{\boldsymbol{\tau}_n\cdot\Delta\mathbf{v} - \tau_{nn}\Delta u_n}{Pr_m+1}\right)\mathbf{n} \\ \mathbf{0} \end{bmatrix}, \tag{IV.10}$$

where we have eliminated the vectors $\mathbf{m}$ and $\mathbf{l}$ by using the properties of three mutually orthogonal vectors as explained in Section II. These vectors are perpendicular to the unit directed area vector $\mathbf{n}$, and not uniquely defined. The above dissipation term is therefore defined unambiguously. The dissipation term can also be obtained directly by using the absolute Jacobian (II.54), which is already independent of $\mathbf{m}$ and $\mathbf{l}$. Note also from Equation (IV.9) that components with zero eigenvalues ($\lambda_k = 0$) do not contribute to the absolute Jacobian $|\mathbf{PA}^v_n|\,\Delta\mathbf{U}$; this is the reason that their eigenvectors are not needed in the construction of

the dissipation term. Observe also that the terms $\frac{1}{Pr_n^2-1}$, $\frac{1}{Pr_m^2-1}$, and $\frac{1}{Pr_l^2-1}$, which appear in the eigenvectors, have disappeared in the dissipation term.

The other contribution, $|\mathbf{PA}_n^{av}|\,\Delta\mathbf{U}$, is the artificial hyperbolic dissipation introduced in Ref.[18] to improve the order of accuracy of $\mathbf{g}$, which is here extended to three dimensions:

$$|\mathbf{PA}_n^{av}|\,\Delta\mathbf{U} = \frac{\nu_v}{L}\begin{bmatrix} 0 & \mathbf{0}^t & 0 & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ 0 & \mathbf{0}^t & 0 & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{n}\otimes\mathbf{n} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{n}\otimes\mathbf{n} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{n}\otimes\mathbf{n} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \end{bmatrix}\Delta\mathbf{U} = \frac{\nu_v}{L}\begin{bmatrix} 0 \\ \Delta(\rho\mathbf{v}) \\ 0 \\ \Delta g_{un}\mathbf{n} \\ \Delta g_{vn}\mathbf{n} \\ \Delta g_{wn}\mathbf{n} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{IV.11}$$

where

$$\Delta g_{un} = \Delta\mathbf{g}_u\cdot\mathbf{n} = \Delta g_{ux}n_x + \Delta g_{uy}n_y + \Delta g_{uz}n_z, \tag{IV.12}$$

$$\Delta g_{vn} = \Delta\mathbf{g}_v\cdot\mathbf{n} = \Delta g_{vx}n_x + \Delta g_{vy}n_y + \Delta g_{vz}n_z, \tag{IV.13}$$

$$\Delta g_{wn} = \Delta\mathbf{g}_w\cdot\mathbf{n} = \Delta g_{wx}n_x + \Delta g_{wy}n_y + \Delta g_{wz}n_z. \tag{IV.14}$$

It is claimed and demonstrated in Ref.[18] that this term provides a strong coupling between $\mathbf{v}$ and $\mathbf{g}$, and ensures the same order of accuracy for $\mathbf{v}$ and $\mathbf{g}$. See Ref.[18] for details. The dissipation term of the upwind flux for the artificial hyperbolic-mass-diffusion term has a very similar structure:

$$|\mathbf{PA}_n^{a}|\,\Delta\mathbf{U} = \frac{\nu_\rho}{L}\begin{bmatrix} 1 & \mathbf{0}^t & 0 & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ 0 & \mathbf{0}^t & 0 & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t & \mathbf{0}^t \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{0} & \mathbf{O} & \mathbf{0} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{n}\otimes\mathbf{n} \end{bmatrix}\Delta\mathbf{U} = \frac{\nu_\rho}{L}\begin{bmatrix} \Delta\rho \\ \mathbf{0} \\ 0 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \Delta r_n\mathbf{n} \end{bmatrix}. \tag{IV.15}$$

The interface quantities needed to evaluate the dissipation matrices are computed by the Roe-averages [51] in the inviscid flux, and otherwise by the arithmetic averages. Then, the inviscid flux $\mathbf{\Phi}_{jk}^i$ in Equation (IV.5) is equivalent to the Roe flux [51]. However, any other inviscid numerical flux can be employed. In our code, a hybrid-rotated Riemann solver called the Rotated-RHLL flux [52] is implemented as a default inviscid flux for robust shock capturing [1]. For test problems considered in this work, it is equivalent to the Roe flux.

Finally, it is pointed out that a conventional viscous discretization can be derived from the upwind viscous flux (IV.6) by ignoring components corresponding to the gradient variables as described in Ref.[53], and the

second, third, and fourth components of the dissipation term (IV.10) serve as a high-frequency damping term essential to robust and accurate viscous discretizations.

IV.C. Kappa Scheme for Reconstruction

For second-order accuracy, the solution needs to be reconstructed at the midpoint of each edge. The reconstruction is performed in the following variables:

$$\mathbf{W} = [\,\rho,\, \mathbf{v}^t,\, T,\, \mathbf{g}_u^t,\, \mathbf{g}_v^t,\, \mathbf{g}_w^t,\, \mathbf{q}^t,\, \mathbf{r}^t\,]^t, \tag{IV.16}$$

to define two states, $\mathbf{W}_L$ and $\mathbf{W}_R$, which are reconstructed from the two end nodes of the edge, j and k, respectively, and sent to the numerical flux function. In our code, we employ the kappa scheme [54, 55]:

$$\mathbf{W}_L = \mathbf{W}_j + \left[\frac{1-\kappa}{2}(\partial_{jk}\mathbf{W}_j - \Delta\mathbf{W}_{jk}) + \frac{1+\kappa}{2}\Delta\mathbf{W}_{jk}\right], \tag{IV.17}$$

$$\mathbf{W}_R = \mathbf{W}_k - \left[\frac{1-\kappa}{2}(\partial_{jk}\mathbf{W}_k - \Delta\mathbf{W}_{jk}) + \frac{1+\kappa}{2}\Delta\mathbf{W}_{jk}\right], \tag{IV.18}$$

where κ is a parameter, and

$$\partial_{jk}\mathbf{W}_j = \nabla\mathbf{W}_j \cdot \Delta\mathbf{l}_{jk}, \quad \partial_{jk}\mathbf{W}_k = \nabla\mathbf{W}_k \cdot \Delta\mathbf{l}_{jk}, \quad \Delta\mathbf{W}_{jk} = \frac{1}{2}\left(\mathbf{W}_k - \mathbf{W}_j\right), \tag{IV.19}$$

and $\Delta\mathbf{l}_{jk} = (x_k - x_j, y_k - y_j, z_k - z_j)$, $\nabla\mathbf{W}_j$ is the gradient of $\mathbf{W}$ computed by a linear LSQ method at j, and similarly for $\nabla\mathbf{W}_k$. For the choice of κ, we set $\kappa = 1/2$ as recommended in Ref.[18]. As we will explain below, in the HNS20 discretization, LSQ gradient computations are not necessary for the primitive variables since they can be obtained directly from the gradient variables.

IV.D. Scheme-II and Third-Order Accuracy in Inviscid Terms

It is known from Ref.[14] that replacing the LSQ gradients by those obtained from the gradient variables results in a superior scheme. The resulting scheme is called Scheme-II, and in contrast, the one based on all LSQ gradients is called Scheme-I [14]. For HNS20, the gradients of the primitive variables (ρ, u, v, w, T) can be obtained directly from the gradient variables:

$$\nabla\rho_j = \frac{\mathbf{r}_j}{\nu_\rho}, \quad \nabla\mathbf{v}_j = \frac{\mathbf{g}_j}{(\mu_v)_j}, \quad \nabla T_j = -\gamma(\gamma-1)\frac{\mathbf{q}_j}{(\mu_h)_j}. \tag{IV.20}$$

Therefore, we do not need to perform the LSQ gradient computation for the primitive variables. The LSQ gradients are computed only for the gradient variables, $\mathbf{g}_u$, $\mathbf{g}_v$, $\mathbf{g}_w$, $\mathbf{q}$, and $\mathbf{r}$. This is Scheme-II for HNS20. Note that the gradients of the primitive variables are then second-order accurate while the LSQ gradients are first-order accurate on unstructured grids.

As shown for the diffusion equation in Ref.[14], Scheme-II gives lower-level errors for both the solution and the gradient variables, compared with Scheme-I. But the most significant feature is that it gives third-order accuracy in the inviscid terms. As mentioned earlier, the node-centered edge-based discretization achieves third-order accuracy with second-order gradients and a linear flux extrapolation [41]. For the inviscid terms, the flux depends only on the primitive variables and their gradients are obtained from the gradient variables with second-order accuracy. Therefore, third-order accuracy can be achieved if the inviscid flux is computed with a linear flux extrapolation:

$$\mathbf{\Phi}_{jk}^i = \frac{1}{2}\left[\,\left(\mathbf{F}_n^i\right)_L + \left(\mathbf{F}_n^i\right)_R\,\right] - \frac{1}{2}\left|\mathbf{A}_n^i\right|\Delta\mathbf{U}, \tag{IV.21}$$

where

$$(\mathbf{F}_n^i)_L = \mathbf{F}_n^i(\mathbf{U}_j) + \frac{1}{2}\left(\frac{\partial\mathbf{F}_n^i}{\partial\mathbf{W}}\right)_j \nabla\mathbf{W}_j \cdot \Delta\mathbf{l}_{jk}, \tag{IV.22}$$

$$(\mathbf{F}_n^i)_R = \mathbf{F}_n^i(\mathbf{U}_k) - \frac{1}{2}\left(\frac{\partial\mathbf{F}_n^i}{\partial\mathbf{W}}\right)_k \nabla\mathbf{W}_k \cdot \Delta\mathbf{l}_{jk}. \tag{IV.23}$$

This feature was demonstrated first in Ref.[15] for the advection-diffusion equation, and in Ref.[18] for the HNS20 system in two dimensions. As pointed out in Ref.[18], third-order accuracy for the inviscid terms cannot be obtained with HNS17 because the density gradient is computed by a linear LSQ method, which is only first-order accurate. It may be possible to obtain second-order density gradients by a quadratic LSQ fit. However, HNS20 is more attractive in that third-order accuracy in the inviscid terms can be achieved by second-order algorithms only, without introducing high-order algorithms such as a quadratic LSQ fit.

IV.E. Implicit Defect-Correction Solver

The discretization leads to a global system of residual equations:

$$0 = \mathbf{Res}(\mathbf{U}_h), \tag{IV.24}$$

where $\mathbf{U}_h$ denotes the global solution vector for which the system is to be solved. To solve the nonlinear system, we consider an implicit defect-correction method in the form:

$$\mathbf{U}_h^{n+1} = \mathbf{U}_h^n + \Delta\mathbf{U}_h, \tag{IV.25}$$

where n is the iteration counter, and the correction $\Delta\mathbf{U}_h$ is obtained as the solution to the linearized system:

$$\left(\mathbf{D} + \frac{\partial\widetilde{\mathbf{Res}}}{\partial\mathbf{U}_h}\right)\Delta\mathbf{U}_h = -\mathbf{Res}(\mathbf{U}_h^n), \tag{IV.26}$$

where $\mathbf{D}$ is a diagonal matrix corresponding to the pseudo time term, and $\widetilde{\mathbf{Res}}$ is the global residual vector based on a first-order spatial discretization. The matrix $\mathbf{D}$ has a diagonal block given at node j by

$$\mathbf{D}_{jj} = \mathrm{diag}\left(\frac{V_j}{\Delta\tau},\ldots,\frac{V_j}{\Delta\tau},\frac{V_j}{\Delta\tau_g},\ldots,\frac{V_j}{\Delta\tau_g},\frac{V_j}{\Delta\tau_q},\frac{V_j}{\Delta\tau_q},\frac{V_j}{\Delta\tau_q},\frac{V_j}{\Delta\tau_r},\frac{V_j}{\Delta\tau_r},\frac{V_j}{\Delta\tau_r}\right), \tag{IV.27}$$

where $\Delta\tau$ denotes a local pseudo-time step for the first five equations (the continuity, momentum, and energy equations), and $\Delta\tau_g$, $\Delta\tau_q$, and $\Delta\tau_r$ for Equations of $\mathbf{g}$ (III.4), of $\mathbf{q}$ (III.5), and of $\mathbf{r}$ (III.6), respectively. The local pseudo-time steps are defined by

$$\Delta\tau = \frac{2\,\mathrm{CFL}}{\sum_{k\in\{k_j\}}(|u_n| + a + a_h)A_{jk}}V_j, \quad \Delta\tau_g = \frac{\mathrm{CFL}_g}{\mathrm{CFL}}\Delta\tau, \quad \Delta\tau_q = \frac{\mathrm{CFL}_q}{\mathrm{CFL}}\Delta\tau, \quad \Delta\tau_r = \frac{\mathrm{CFL}_r}{\mathrm{CFL}}\Delta\tau, \tag{IV.28}$$

and controlled by the CFL numbers: CFL, CFL_g, CFL_q, CFL_r. The Jacobian matrix consists of 20×20 diagonal and off-diagonal blocks defined per node: the diagonal block at node j is given by

$$\left(\frac{\partial\widetilde{\mathbf{Res}}}{\partial\mathbf{U}_h}\right)_{jj} = \mathbf{P}_j\sum_{k\in\{k_j\}}\left(\frac{\partial\mathbf{\Phi}_{jk}}{\partial\mathbf{U}_j}A_{jk} - \frac{\partial\mathbf{S}_j}{\partial\mathbf{U}_j}V_j\right) + \frac{\partial\mathbf{P}_j}{\partial\mathbf{U}_j}\left(\sum_{k\in\{k_j\}}\mathbf{\Phi}_{jk}A_{jk} - \mathbf{S}_jV_j\right), \tag{IV.29}$$

and the corresponding off-diagonal blocks are given by

$$\left(\frac{\partial\widetilde{\mathbf{Res}}}{\partial\mathbf{U}_h}\right)_{jk} = \mathbf{P}_j\frac{\partial\mathbf{\Phi}_{jk}}{\partial\mathbf{U}_k}A_{jk} \quad \text{for } k\in\{k_j\}, \tag{IV.30}$$

where the numerical flux $\mathbf{\Phi}_{jk}$ is defined with the two nodal solutions, $\mathbf{U}_L = \mathbf{U}_j$ and $\mathbf{U}_R = \mathbf{U}_k$, i.e., the first-order spatial discretization (Scheme-I), except the second term in Equation (IV.29). Note that all derivatives, including $\frac{\partial\mathbf{\Phi}_{jk}}{\partial\mathbf{U}_j}$ and $\frac{\partial\mathbf{\Phi}_{jk}}{\partial\mathbf{U}_k}$:

$$\frac{\partial\mathbf{\Phi}_{jk}}{\partial\mathbf{U}_j} = \frac{1}{2}\frac{\partial}{\partial\mathbf{U}_j}\left\{\mathbf{F}_n(\mathbf{U}_j) - \left(\left|\mathbf{A}_n^i\right| + \mathbf{P}^{-1}\left|\mathbf{PA}_n^v\right| + \mathbf{P}^{-1}\left|\mathbf{PA}_n^{av}\right| + \mathbf{P}^{-1}\left|\mathbf{PA}_n^a\right|\right)(\mathbf{U}_k - \mathbf{U}_j)\right\}, \tag{IV.31}$$

$$\frac{\partial\mathbf{\Phi}_{jk}}{\partial\mathbf{U}_k} = \frac{1}{2}\frac{\partial}{\partial\mathbf{U}_k}\left\{\mathbf{F}_n(\mathbf{U}_k) - \left(\left|\mathbf{A}_n^i\right| + \mathbf{P}^{-1}\left|\mathbf{PA}_n^v\right| + \mathbf{P}^{-1}\left|\mathbf{PA}_n^{av}\right| + \mathbf{P}^{-1}\left|\mathbf{PA}_n^a\right|\right)(\mathbf{U}_k - \mathbf{U}_j)\right\}, \tag{IV.32}$$

have been derived exactly and hand-coded. Therefore, the solver becomes Newton's method if the LSQ gradients are ignored and the pseudo-time steps are sufficiently large: $\mathrm{CFL}, \mathrm{CFL}_g, \mathrm{CFL}_q, \mathrm{CFL}_r \to \infty$. For simplicity, the automatic differentiation technique [56] may be employed, but the hand-coded Jacobian is employed here for the sake of computational efficiency. The Jacobian blocks are computed by looping over edges and accumulating the contributions at nodes, just like the residual computations, except for the source term and the local-preconditioning matrix contributions, which are computed more conveniently by looping over nodes.

In practice, we do not solve but relax the linear system by a relaxation scheme. In this study, we employ the sequential Gauss-Seidel (GS) under-relaxation scheme which can be written at node j as

$$\Delta \mathbf{U}_j^{n+1} = (1-\omega)\Delta \mathbf{U}_j^n + \omega \Delta \mathbf{U}_j^{GS}, \tag{IV.33}$$

$$\Delta \mathbf{U}_j^{GS} = -\left(\frac{\partial \widetilde{\mathbf{Res}}}{\partial \mathbf{U}_h}\right)_{jj}^{-1} \left\{ \sum_{k\in\{k_j^n\}} \left(\frac{\partial \widetilde{\mathbf{Res}}}{\partial \mathbf{U}_h}\right)_{jk} \Delta \mathbf{U}_k^n + \sum_{k\in\{k_j^{n+1}\}} \left(\frac{\partial \widetilde{\mathbf{Res}}}{\partial \mathbf{U}_h}\right)_{jk} \Delta \mathbf{U}_k^{n+1} + \mathbf{Res}(\mathbf{U}_h^n) \right\}, \tag{IV.34}$$

where $\{k_j^n\}$ and $\{k_j^{n+1}\}$ denote sets of neighbor nodes where the update (IV.33) has not been performed and has already performed before updating at the node j, respectively, and ω is a relaxation parameter typically taken as $\omega = 0.9$. The relaxation is performed to a specified tolerance or to a specified maximum number of relaxations.

It should be noted that it is possible to employ an explicit solver such as a pseudo-time marching scheme, but it will require some fix for high-Reynolds-number cases. This is because the elements of $\mathbf{P}$ become vanishingly small for high Reynolds numbers, and thus give very small weights to the equations for the gradient variables, i.e., Equations (II.10) and (II.11). Explicit solvers will then suffer from extremely slow convergence for these equations. The same is true for the density gradients because the coefficient ν_ρ is small; in this case, slow convergence would be expected for any Reynolds number. Furthermore, explicit solvers would suffer from a severe convergence deterioration for highly-stretched high-aspect-ratio grids. These problems may be resolved by a proper definition of the pseudo time step or a suitable modification to the local-preconditioning matrix. But the implicit solver does not suffer, as long as the CFL numbers are sufficiently large, apparently because the inversion of the diagonal block as in Equation (IV.34) cancels such problematic scalings. In this work, to avoid slow convergence, we define the CFL numbers as

$$\mathrm{CFL}_g = \mathrm{CFL}_q = \mathrm{CFL}_r = 10^{10}\,\mathrm{CFL}, \tag{IV.35}$$

for a given CFL. In effect, these CFL numbers may be considered as infinite.

V. Numerical Results

We present preliminary results for a laminar flow over a sphere of unit diameter with $M_\infty = 0.2$, $Pr = 0.72$, $\gamma = 1.4$, $Re_\infty = 40$, and $T_\infty = 300$ [K]. The viscosity is taken as a constant. The outer boundary is taken to be a cube centered at the origin with each side 100 times the diameter; and the sphere is also centered at the origin. Free stream boundary condition is applied at the inflow and side boundary planes, and the back-pressure condition is applied at the outflow boundary plane. On the surface of the sphere, the no-slip and adiabatic conditions are imposed. For all cases, we employ the implicit defect-correction solver with $\mathrm{CFL} = 100$ for the pseudo time step. The linear relaxation is performed until the linear residual is reduced by one order of magnitude or it reaches the maximum relaxations of 100. For the cases presented below, the linear relaxation typically converged at around 50 relaxations. The residual Jacobian is computed at every nonlinear iteration. The implicit solver is taken to be converged when the iterative solution difference, which is normalized by a reference value, is reduced by five orders of magnitude in the L_1 norm for all the variables. The performance of the HNS solver are examined in comparison with a convention solver, which is a second-order node-centered edge-based scheme based on the same upwind flux for the inviscid terms and the alpha-damping flux for the viscous terms with the parameter $\alpha = 4/3$ [43, 57]. An implicit solver is constructed also for the conventional scheme. The inviscid Jacobian is exact for the first-order inviscid scheme, but the viscous Jacobian is based on the inconsistent edge-terms-only scheme [58] because a consistent and compact version is not available, as is often the case for conventional viscous schemes, for

	TNS		HNS17		HNS20	
Grid	Cycle	CPU Time	Cycle	CPU Time	Cycle	CPU Time
Coarse	46	5.00	43	21.86	43	27.32
Medium	66	54.19	54	230.52	53	311.36
Fine	119	931.72	82	3380.77	82	4721.31

Table 1: Iterative convergence results. CPU time is measured in seconds. TNS with unweighted LSQ gradients. Results are very similar for TNS with weighted LSQ gradients, and therefore not shown.

the alpha-damping scheme. For LSQ gradients, both unweighted and weighted LSQ methods are considered. But the results for the HNS solver are very similar, and therefore only the results obtained with unweighted LSQ gradients are shown. The conventional scheme is referred to as Traditional NS (TNS) in the rest of the section. Both the HNS and TNS solvers have been implemented into the fully-parallelized SC/Tetra package. Three levels of unstructured grids have been generated for this problem as shown in Figure 1. All calculations were performed by SC/Tetra with 48 processors.

First, we focus on the comparison between the TNS (unweighted LSQ) solver and the HNS20 solver. Pressure and vorticity contours are compared in Figures 2 and 3, respectively. As can be seen, the HNS20 solver yields much more accurate distributions of the pressure as well as the vorticity. The vorticity obtained with the TNS solver exhibits numerical noise even on the fine grid. It clearly demonstrates the superior feature of the HNS method for accurately predicting gradient quantities on three-dimensional unstructured grids. Shown in Table 1 are the iterative convergence results. Focusing on the iteration number, the HNS20 solver converges with much fewer numbers of iterations for the same grid as the grid is refined although it takes larger total CPU times. It indicates that the HNS20 solver will get faster in CPU time than the TNS solver eventually as the grid is further refined. As observed Refs.[16, 18] for two-dimensional problems, the HNS20 solver begins to yield faster convergence in CPU time for a given grid once the iterative convergence speed-up dominates over the high per-iteration cost of the HNS20 solver. Note that a comparison of convergence speeds for a given grid is just one aspect of efficiency comparison, and actually rarely discussed since improved solvers are likely to be more expensive for the same grid. If we focus on a target accuracy, the story is quite different. In fact, as typical in high-order methods, it is more sensible to discuss efficiency for a target accuracy, not for a given grid. If we compare the pressure contours in Figure 2, we find that the results obtained by the HNS20 solver (medium) and the TNS solver (fine) are of similar quality. Moreover, comparing the vorticity contours in Figure 3, we observe that the HNS20 solver yields accurate vorticity distribution even on the medium grid while the TNS solver produces noisy vorticity contours even on the finest grid. Therefore, if we compare the HNS20 solver (medium) and the TNS solver(fine), the HNS20 solver is more efficient and the speed-up factor in CPU time is $931.72/311.36 \approx 3.0$, and much greater in terms of accurate vorticity prediction. It should be noted, finally, that the possibility of achieving faster convergence in CPU time for a given fine grid over a conventional second-order solver is a special feature of the HNS method, which cannot be generally expected for methods that can provide third-order accuracy in the inviscid terms and second-order accurate gradients on general unstructured grids.

Results obtained with the TNS (weighted LSQ) and HNS17 solvers are shown in Figures 4 and 5. For the TNS solver, the use of weighted LSQ gradients does not have an impact on the pressure contours but improves the vorticity contours significantly. However, the results are still not as accurate as those obtained by the HNS solver: the fine-grid result of TNS is similar to the coarse-grid result of HNS. Results obtained by the HNS17 solver are very similar to those obtained by the HNS20 solver. The effect of the third-order inviscid scheme (HNS20) is not clearly observed for this particular problem. It should be investigated for high-Reynolds-number flows, where the accuracy of the inviscid terms is expected to have a large impact. The cost comparison in Table 1 shows that the HNS17 and HNS20 solvers converged in almost the same number of iterations and the HNS17 solver takes less CPU time as expected.

Finally, further numerical experiments indicate that the HNS solver is more robust than the TNS solver for highly-skewed viscous grids. Although not shown here, the HNS solver was tested for the sphere case with highly-stretched irregular tetrahedral grids (purely tetrahedral grids with no prismatic layer). It was observed that the HNS solver converged for all such grids and produce accurate gradients whereas the TNS solver diverges and cannot produce results even for grids with the element-aspect-ratio of 8. These results will be reported in future.

VI. Concluding Remarks

The HNS method has been extended to three dimensions. The eigen-structure of the hyperbolized viscous systems have been presented, including left and right eigenvectors relevant to the construction of the upwind viscous flux. It is also pointed out that the length scale does not require an aspect-ratio-fix in implicit solvers, and it should be understood as a nondimensionalized length. An advanced HNS system called HNS20 is discretized by the second-order node-centered edge-based method on unstructured grids in three dimensions. A superior construction called Scheme-II, where accurate gradient variables are used to perform the linear solution extrapolation, has been employed to upgrade the inviscid scheme to third-order. An implicit defect-correction solver is developed based on a compact Jacobian derived exactly for a first-order version of the scheme. Numerical results show that the HNS solver has a potential for providing immediate improvements to conventional second-order solvers especially in gradient predictions on unstructured grids in three dimensions.

This paper is a report on work in progress, and the presented results are by no means exhaustive. The following items are currently explored for increased robustness, efficiency, and accuracy:

Advanced linear solver: More advanced linear solvers, than the under-relaxed GS relaxation scheme, should be implemented. First, the under-relaxation parameter ω should be controlled adaptively based on changes in the linear residuals. Second, a multi-color GS scheme is our next candidate, which can be made partition-independent and thus the results will not depend on the number of processors. Finally, employing the relaxation scheme as a smoother, a multigrid method may be constructed to further improve the efficiency.

Simplified Jacobian and decoupled relaxation strategy: It is possible to reduce the memory requirement for the Jacobian and the operation count for the relaxation by employing a loosely-coupled approach where the 20×20 Jacobian block is broken into 5×5, 3×3, 3×3, 3×3, 3×3, 3×3 sub-blocks, with all other off-diagonal elements ignored. Although such a simplified Jacobian generally leads to slower convergence, it may still serve as an effective preconditioner for a Newton-Krylov solver.

Advanced nonlinear solver: A Jacobian-Free Newton-Krylov (JFNK) solver should be employed not only for efficiency but also for robustness. JFNK solvers are efficient in the sense that a quadratic convergence can be achieved in principle, but an effective preconditioner is required to be efficient. The type of implicit solver considered in this paper is known to serve as an effective preconditioner [59]. It is particularly attractive for robustness that the JFNK solver cannot diverge even if the preconditioner diverges. In developing a nonlinear solver, an adaptive CFL strategy is known to increase robustness and should be implemented [59]. Also, the residual Jacobian may be frozen and updated only at every order of magnitude reduction in the nonlinear residual to further save computational time. Ultimately, a nonlinear multigrid solver may be developed with such an advanced nonlinear solver employed as a smoother [60].

Boundary treatments: A high-order boundary flux quadrature [17] should be used to fully take advantage of the third-order inviscid terms. Also, a high-order quadrature should be explored for evaluating integrated quantities such as the drag coefficient, which would be possible because the HNS method produces second-order accurate viscous stresses and their gradients are also available with first-order accuracy.

Along with these potential algorithmic enhancements, additional numerical experiments are being planned to further investigate the performance of the second-order HNS solver, including high-Reynolds-number unsteady flows. In doing so, an assessment of efficiency will need to be made carefully for a target accuracy, e.g., in the drag coefficient. Finally, we conclude the paper with certain areas where the HNS method has a potential for bringing improvements over existing methods:

(1) Gradient prediction: CFD simulations are often performed to predict physical quantities involving gradients. A typical example is the drag coefficient, which requires accurate prediction of the viscous stresses. Another would be the prediction of heating rates in heat-conducting flows. Yet, accurate vorticity prediction for unsteady vortical flows such as a flow over rotorcraft is also an important example. For such applications, the high-accurate and high-quality gradient prediction capability of the HNS method is a very attractive feature especially on adaptive unstructured grids over complex geometries. It is attractive not only for accuracy but also for efficiency because accurate gradients can be obtained with a coarser grid, thus

reducing the computational cost.

(2) Grid adaptation with tetrahedral grids: High-order and high-quality gradient prediction capability of the HNS method has a potential for simplifying a grid generation process with fully tetrahedral elements throughout a domain. Furthermore, it has a potential for achieving accurate viscous flow simulations with anisotropic grid adaptation, which is considered as an essential mechanism for high-order methods to be practical [3], and is typically more efficient with tetrahedral elements [38–40]. In addition, the HNS method has the ability to produce first-order accurate second-order derivatives, which are often used as a guide for anisotropic grid adaptation but inconsistent in conventional second-order methods on general unstructured grids [21].

(3) High-Reynolds-number turbulent flows: Turbulence models in Reynolds-Averaged NS (RANS) methods involve gradient quantities typically in the source terms. It is well known that inaccurate gradients in the source terms can cause a significant deterioration in accuracy and robustness of RANS simulations. The HNS method, as it has the ability to generate accurate gradients, is expected to provide improvements in this regard. Accurate velocity gradients can also be used directly for evaluating the Reynolds stresses modeled as a quadratic function of velocity gradients [61], and furthermore for providing accurate strain tensor and vorticity in transition models [62]. Accurate gradients of turbulence variables can also be obtained because turbulence model equations can be formulated and discretized as a hyperbolic advection-diffusion system. Furthermore, in large-eddy simulations (LES), accurate gradients generated by the HNS method would lead to accurate evaluation of sub-grid models constructed based on gradient quantities. For highly refined grids as typical in LES and direct numerical simulations (DNS), the improved conditioning by eliminating the effect of second derivatives from the NS equations may prove to be advantageous in terms of both efficiency and robustness. This is true even with wall functions, which are often employed to reduce computational requirements by modeling, instead of resolving, a near-wall behavior in turbulent flow simulations. Wall functions indeed allow a large cell-spacing off the wall, but the grid needs to be as fine (and stretched, and high-aspect-ratio) as a wall-integration grid beyond the first cell for accurate predictions [63]. The HNS method also has a potential for improving wall functions with accurate gradients. For example, the wall shear stress can be oriented more accurately by a quadratic extrapolation of a wall-tangential velocity, which is possible with a second-order HNS scheme. Also, accurate pressure gradients can be directly used in wall functions that incorporate the pressure gradient [64].

(4) High-order methods: The HNS method is a method of reformulating the NS system, and therefore, applicable to any discretization method, including high-order discretization methods. High-order methods are desired for many applications, but generally computationally very expensive (without grid adaptation) on unstructured grids. Then, the HNS method does not seem an attractive option because it would further increase the computational cost by additional variables and equations. However, that may not be entirely true. In fact, there is a possibility that some high-order discretizations may be constructed for the HNS system without introducing additional degrees of freedom. This is possible if we can use the gradient variables to construct a high-order representation of the primitive variables and also unify some of the high-order moments of the gradient variables that approximate the same derivatives. It can be considered as an extension of Scheme-II to high-order methods. A similar construction was presented also for the residual-distribution method [12]. High-order methods successfully combined with the HNS method thus would not necessarily require additional cost, and offer the following potential advantages. First, it eliminates the stiffness of the second-derivatives in the viscous terms, and improves iterative convergence for viscous dominated regions. The improvement would be observed immediately in the case there is no increase in the number of unknowns. Second, it will generate high-quality gradients (e.g., viscous stresses, vorticity, heat fluxes), whose order of accuracy is the same as the solution variables, on fully irregular adaptive grids. Third, accurate gradients can be used also to improve the order of accuracy in the inviscid terms by upgrading the polynomial order in the primitive variables. If high-order methods were to replace the current state-of-the-art second-order methods in future, they might very well end up as high-order HNS methods along with fully arbitrary grid adaptation.

References

[1]Nakashima, Y., Watanabe, N., and Nishikawa, H., "Development of an Effective Implicit Solver for General-Purpose Unstructured CFD Software," *The 28th Computational Fluid Dynamics Symposium*, C08-1, Tokyo, Japan, 2014.

[2]Vincent, P. E. and Jameson, A., "Facilitating the Adoption of Unstructured High-Order Methods Amongst a Wider Community of Fluid Dynamicists," *Math. Model. Nat. Phenom.*, Vol. 6, No. 3, 2011, pp. 97–8140.

[3]Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T., Kroll, N., May, G., Persson, P.-O., Van Leer, B., and Visbal, M., "High-Order CFD Methods: Current status and Perspective," *Int. J. Numer. Meth. Fluids*, Vol. 72, 2012, pp. 811–845.

[4]Ims, J., Duan, Z., and Wang, Z. J., "meshCurve: An Automated Low-Order to High-Order Mesh Generator," *Proc. of 22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-2293, Dallas, TX, 2015.

[5]Fortunato, M. and Persson, P.-O., "High-order unstructured curved mesh generation using the Winslow equations," *J. Comput. Phys.*, Vol. 307, 2016, pp. 1–14.

[6]Gargallo-Peiró, A., Roca, X., Peraire, J., and Sarrate, J., "Optimization of a Regularized Distortion Measure to Generate Curved High-Order Unstructured Tetrahedral Meshes," *Int. J. Numer. Meth. Fluids*, Vol. 103, 2015, pp. 342–363.

[7]Nishikawa, H., "A First-Order System Approach for Diffusion Equation. I: Second Order Residual Distribution Schemes," *J. Comput. Phys.*, Vol. 227, 2007, pp. 315–352.

[8]Nishikawa, H., "A First-Order System Approach for Diffusion Equation. II: Unification of Advection and Diffusion," *J. Comput. Phys.*, Vol. 229, 2010, pp. 3989–4016.

[9]Nishikawa, H., "New-Generation Hyperbolic Navier-Stokes Schemes: $O(1/h)$ Speed-Up and Accurate Viscous/Heat Fluxes," *Proc. of 20th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2011-3043, Honolulu, Hawaii, 2011.

[10]Mazaheri, A. and Nishikawa, H., "First-Order Hyperbolic System Method for Time-Dependent Advection-Diffusion Problems," *NASA-TM–2014–218175*, March 2014.

[11]Mazaheri, A. and Nishikawa, H., "Very efficient high-order hyperbolic schemes for time-dependent advection-diffusion problems: Third-, fourth-, and sixth-order," *Computers and Fluids*, Vol. 102, 2014, pp. 131–147.

[12]Mazaheri, A. and Nishikawa, H., "Improved second-order hyperbolic residual-distribution scheme and its extension to third-order on arbitrary triangular grids," *J. Comput. Phys.*, Vol. 300, 2015, pp. 455–491.

[13]Nishikawa, H., "Divergence Formulation of Source Term," *J. Comput. Phys.*, Vol. 231, 2012, pp. 6393–6400.

[14]Nishikawa, H., "First-, Second-, and Third-Order Finite-Volume Schemes for Diffusion," *J. Comput. Phys.*, Vol. 256, 2014, pp. 791–805.

[15]Nishikawa, H., "First, Second, and Third Order Finite-Volume Schemes for Advection-Diffusion," *J. Comput. Phys.*, Vol. 273, 2014, pp. 287–309.

[16]Nishikawa, H., "First, Second, and Third Order Finite-Volume Schemes for Navier-Stokes Equations," *Proc. of 7th AIAA Theoretical Fluid Mechanics Conference, AIAA Aviation and Aeronautics Forum and Exposition 2014*, AIAA Paper 2014-2091, Atlanta, GA, 2014.

[17]Nishikawa, H., "Accuracy-Preserving Boundary Flux Quadrature for Finite-Volume Discretization on Unstructured Grids," *J. Comput. Phys.*, Vol. 281, 2015, pp. 518–555.

[18]Nishikawa, H., "Alternative Formulations for First-, Second-, and Third-Order Hyperbolic Navier-Stokes Schemes," *Proc. of 22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-2451, Dallas, TX, 2015.

[19]Nishikawa, H. and Roe, P. L., "Third-Order Active-Flux Scheme for Advection Diffusion: Hyperbolic Diffusion, Boundary Condition, and Newton Solver," *Computers and Fluids*, Vol. 125, 2016, pp. 71–81.

[20]Löhner, R., "Improved Error and Work Estimates for High-Order Elements," *Int. J. Numer. Meth. Fluids*, Vol. 72, 2013, pp. 1207–1218.

[21]Kamenski, L. and Huang, W., "How a Nonconvergent Recovered Hessian works in Mesh Adaptation," *SIAM J. Numer. Anal.*, Vol. 52, No. 4, 2014, pp. 1692–1708.

[22]Yang, H. Q. and Harris, R. E., "Development of Vertex-Centered, High-Order Schemes and Implementation in FUN3D," *Proc. of 22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-3192, Dallas, TX, 2015.

[23]Abalakin, I., Kozubskaya, T., and Dervieux, A., "High Accuracy Finite Volume Method for Solving Nonlinear Aeroacoustics Problems on Unstructured Meshes," *Chinese Journal of Aeroacoustics*, Vol. 19, No. 2, 2006, pp. 97–104.

[24]Girault, V. and Raviart, P. A., "An analysis of a mixed finite element method for the Navier-Stokes equations," *Numer. Math.*, Vol. 33, 1979, pp. 235–271.

[25]Heys, J., Lee, E., Manteuffel, T., and McCormick, S., "An Alternative Least-Squares Formulation of the Navier-Stokes Equations with Improved Mass Conservation," *J. Comput. Phys.*, Vol. 226, 2008, pp. 944–1006.

[26]Jiang, B.-N., *The Least-Squares Finite Element Method*, Springer, 1998.

[27]Toro, E. F. and Montecinos, G. I., "Advection-diffusion-reaction equations: Hyperbolisation and high-order ADER discretizations," *SIAM J. Sci. Comput.*, Vol. 36, 2014, pp. A2423–A2457.

[28]Masatsuka, K., "I do like CFD, VOL.1, Second Edition," `http://www.cfdbooks.com`, 2013.

[29]"CFL3D online manual," `http://cfl3d.larc.nasa.gov`.

[30]"FUN3D online manual," `http://fun3d.larc.nasa.gov`.

[31]Weiss, J. M. and Smith, W. A., "Preconditioning Applied to Variable and Constant Density Flows," *AIAA J.*, Vol. 33, No. 11, 1995, pp. 2050–2057.

[32]van Leer, B., Lee, W.-T., and Roe, P. L., "Characteristic Time-Stepping or Local Preconditioning of the Euler Equations," *Proc. of 10th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 91-1552, Hawaii, 1991.

[33]Turkel, E., "Preconditioning Methods for Solving the Incompressible and Low-Speed Compressible Equations," *J. Comput. Phys.*, Vol. 72, 1987, pp. 277–298.

[34]H., Y. and Merkle, C. L., "The Applicaiton of Preconditioning in Viscous Flows," *J. Comput. Phys.*, Vol. 105, 1993, pp. 207–223.

[35]Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2, A Wiley - Interscience Publications, 1990.

[36]Brenner, H., "Navier-Stokes Revisited," *Physics A*, Vol. 349, 2005, pp. 60–132.

[37]Brenner, H., "Diffusive Volume Transport in Fluids," *Physics A*, Vol. 389, 2010, pp. 4026–4045.

[38]Yano, M. and Darmofal, D. L., "An Optimization Framework for Anisotropic Simplex Mesh Adaptation: Application to Aerodynamic Flows," *Proc. of 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA Paper 2012-0079, Nashville, Tennessee, 2012.

[39]Michal, T. and Krakos, J., "Anisotropic Mesh Adaptation Through Edge Primitive Operations," *Proc. of 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA Paper 2012-0159, Nashville, Tennessee, 2012.

[40]Park , M. A. and Darmofal, D. L., "Parallel Anisotropic Tetrahedral Adaptation," *Proc. of 46th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2008-917, Reno, Nevada, 2008.

[41]Katz, A. and Sankaran, V., "Mesh Quality Effects on the Accuracy of CFD Solutions on Unstructured Meshes," *J. Comput. Phys.*, Vol. 230, 2011, pp. 7670–7686.

[42]Anderson, W. K. and Bonhaus, D. L., "An implicit upwind algorithm for computing turbulent flows on unstructured grids," *Comput. Fluids*, Vol. 23, 1994, pp. 1–21.

[43]Nishikawa, H., "Beyond Interface Gradient: A General Principle for Constructing Diffusion Schemes," *Proc. of 40th AIAA Fluid Dynamics Conference and Exhibit*, AIAA Paper 2010-5093, Chicago, 2010.

[44]Barth, T. J., "Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes," AIAA Paper 91-0721, 1991.

[45]Luo, H., Baum, J. D., and Löhner, R., "An Improved Finite Volume Scheme for Compressible Flows on Unstructured Grids," *Proc. of 33rd Aerospace Sciences Meeting and Exhibit*, AIAA Paper 95-0348, 1995.

[46]Haselbacher, A., McGuirk, J. J., and Page, G. J., "Finite Volume Discretization Aspects for Viscous Flows on Mixed Unstructured Grids," *AIAA J.*, Vol. 37, No. 2, 1999, pp. 177–184.

[47]Diskin, B., Thomas, J. L., Nielsen, E. J., Nishikawa, H., and White, J. A., "Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Viscous Fluxes," *AIAA J.*, Vol. 48, No. 7, July 2010, pp. 1326–1338.

[48]Diskin, B. and Thomas, J. L., "Accuracy Analysis for Mixed-Element Finite-Volume Discretization Schemes," *NIA Report No. 2007-08*, 2007.

[49]Pincock, B. and Katz, A., "High-Order Flux Correction for Viscous Flows on Arbitrary Unstructured Grids," *J. Sci. Comput.*, Vol. 61, 2014, pp. 454–476.

[50]Schlichting, H. and Gersten, K., *Boundary Layer Theory*, Springer, eighth ed., 2000.

[51]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *J. Comput. Phys.*, Vol. 43, 1981, pp. 357–372.

[52]Nishikawa, H. and Kitamura, K., "Very Simple, Carbuncle-Free, Boundary-Layer Resolving, Rotated-Hybrid Riemann Solvers," *J. Comput. Phys.*, Vol. 227, 2007, pp. 2560–2581.

[53]Nishikawa, H., "Two Ways to Extend Diffusion Schemes to Navier-Stokes Schemes: Gradients or Upwinding," *20th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2011-3044, Honolulu, Hawaii, 2011.

[54]van Leer, B., "Upwind-Difference Methods for Aerodynamic Problems Governed by the Euler Equations," *Fluid Mechanics, Lectures in Applied Mathematics*, Vol. 22, 1985, pp. 327–335.

[55]Burg, C. O. E., "Higher Order Variable Extrapolation For Unstructured Finite Volume RANS Flow Solvers," AIAA Paper 2005-4999, 2005.

[56]Rall, L. B. and Corliss, G. F., "An Introduction to Automatic Differentiation," *Computational Differentiation: Techniques, Applications, and Tools*, edited by M. Berz, C. H. Bischof, G. F. Corliss, and A. Griewank, SIAM, Philadelphia, Penn., 1996, pp. 1–17.

[57]Nishikawa, H., "Robust and Accurate Viscous Discretization via Upwind Scheme - I: Basic Principle," *Comput. Fluids*, Vol. 49, No. 1, October 2011, pp. 62–86.

[58]Thomas, J. L., Diskin, B., and Nishikawa, H., "A Critical Study Agglomerated Multigrid Methods for Diffusion on Highly-Stretched Grids," *Comput. Fluids*, Vol. 40, No. 1, 2010, pp. 82–93.

[59]Pandya, M. J., Diskin, B., and James L. Thomas, N. T. F., "Improved Convergence and Robustness of USM3D Solutions on Mixed Element Grids," *Proc. of 53rd AIAA Aerospace Sciences Meeting*, AIAA Paper 2015-1747, Kissimmee, Florida, January 2015.

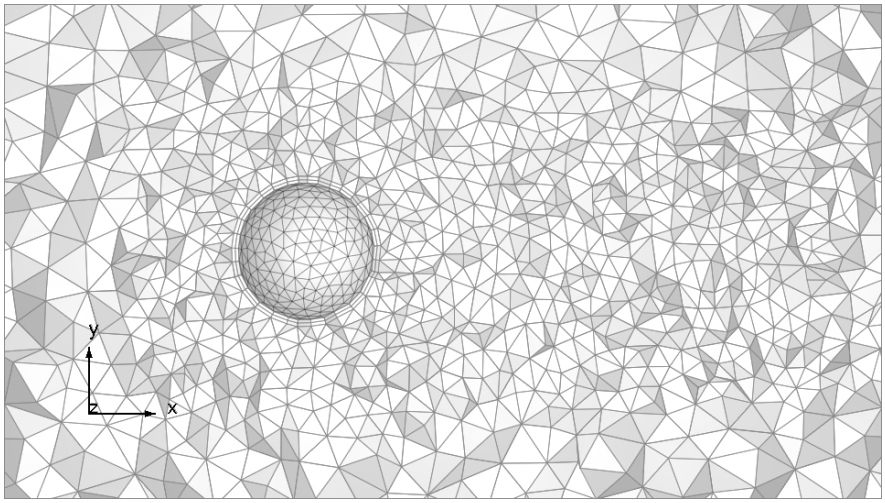
[60]Nishikawa, H., Diskin, B., Thomas, J. L., and Hammond, D. H., "Recent Advances in Agglomerated Multigrid," *51st AIAA Aerospace Sciences Meeting*, AIAA Paper 2013-863, Texas, 2013.

[61]Mani, M., Babcock, D. A., Winkler, C. M., and Spalart, P. R., "Predictions of a Supersonic Turbulent Flow in a Square Duct," *51st AIAA Aerospace Sciences Meeting*, AIAA Paper 2013-0860, Texas, 2013.

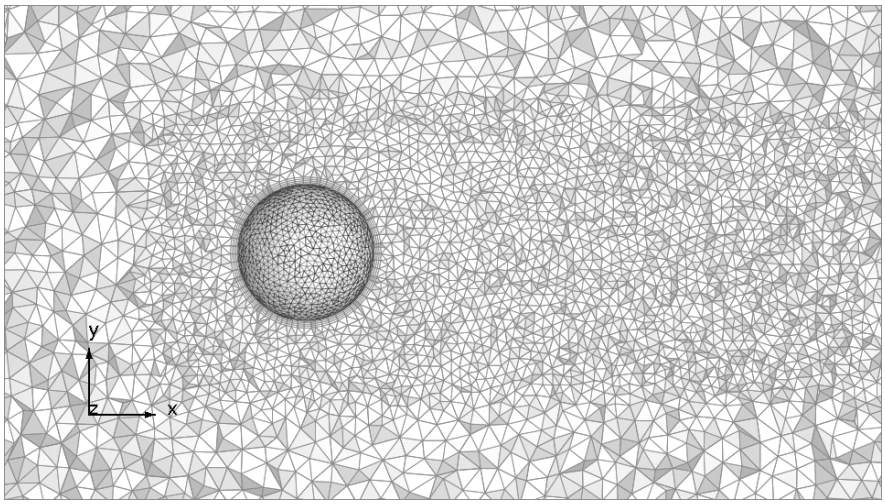
[62]Menter, F. R., Langtry, R., and Völker, S., "Transition Modelling for General Purpose CFD Codes," *Flow, Turbulence Combustion*, Vol. 77, 2006, pp. 277–303.

[63]Carlson, J.-R., Vatsa, V. N., and White, J., "Validation of a Node-Centered Wall Function Model for the Unstructured Flow Code FUN3D," *Proc. of 22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-2758, Dallas, TX, 2015.

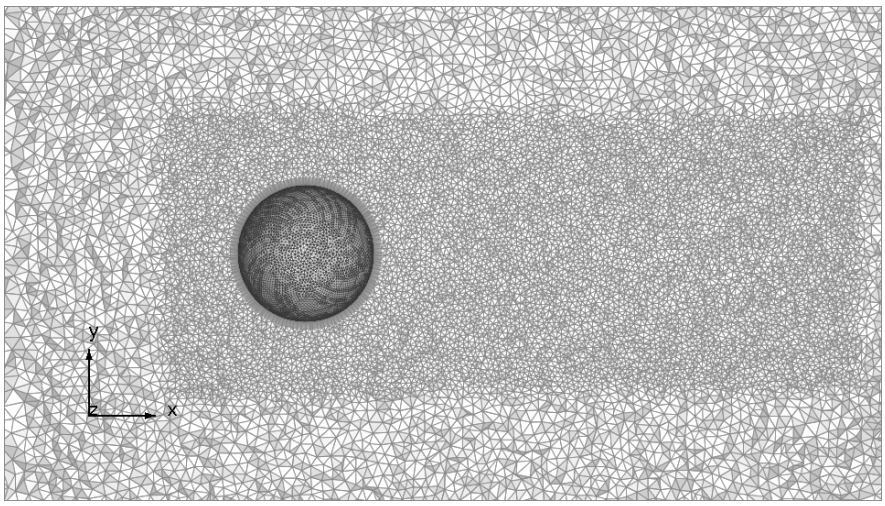
[64]Wilcox, D. C., "Wall Matching, A Rational Alternative to Wall Functions," *Proc. of 27th Aerospace Sciences Meeting*, AIAA Paper 89-0611, Reno, Nevada, 1989.



(a) Coarse Grid: 29,126 nodes and 162,085 elemetns.

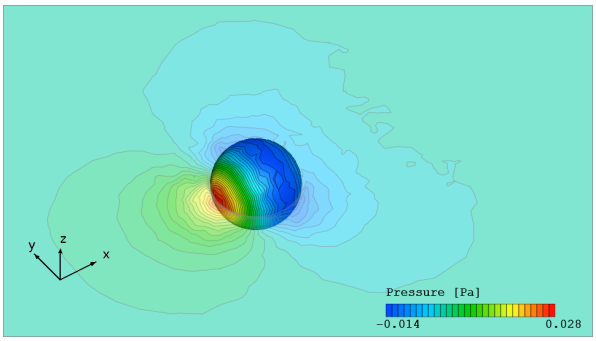


(b) Medium Grid: 201,107 nodes and 1,129,637 elements.

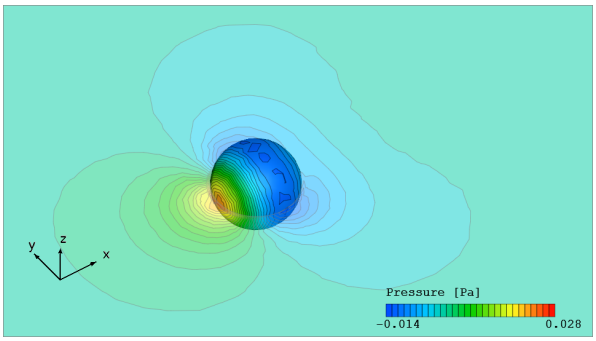


(c) Fine Grid: 1,527,768 nodes and 8,628,989 elements.

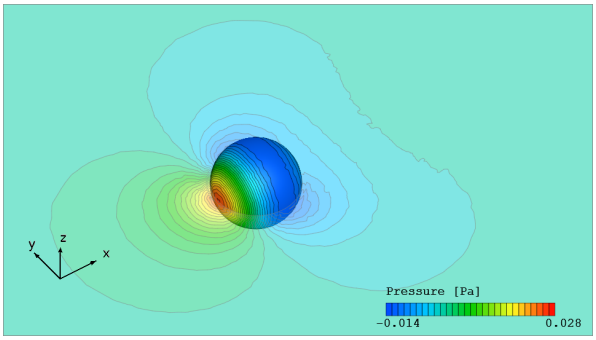
Figure 1: Three levels of unstructured grids generated over a sphere: a prismatic layer near the surface wrapped by a fully irregular tetrahedral grid.



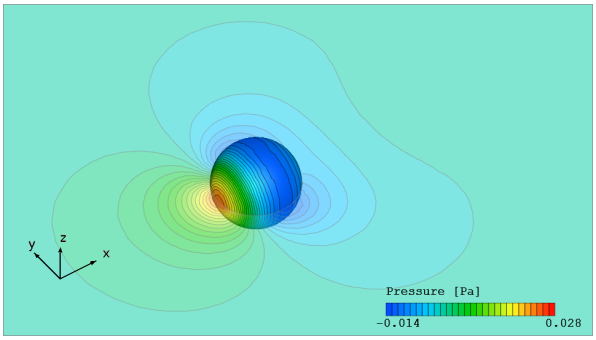
(a) TNS (unweighted LSQ) on the coarse grid.



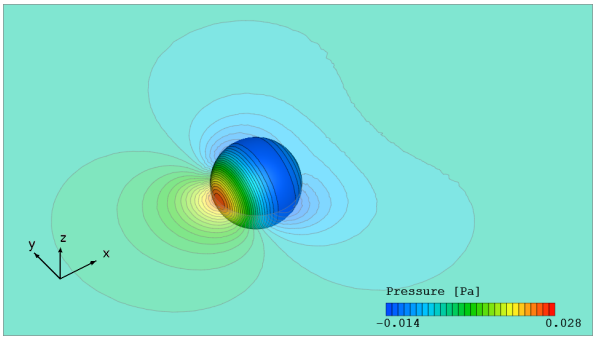
(b) HNS20 on the coarse grid.



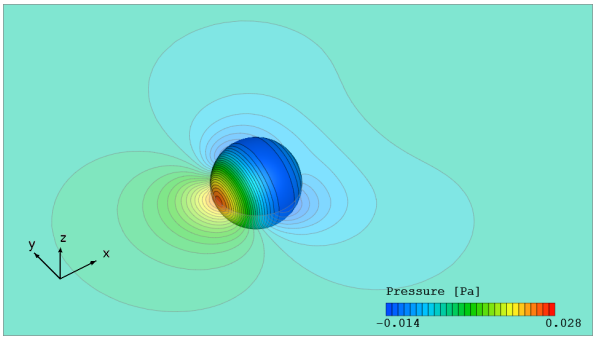
(c) TNS (unweighted LSQ) on the medium grid.



(d) HNS20 on the medium grid.

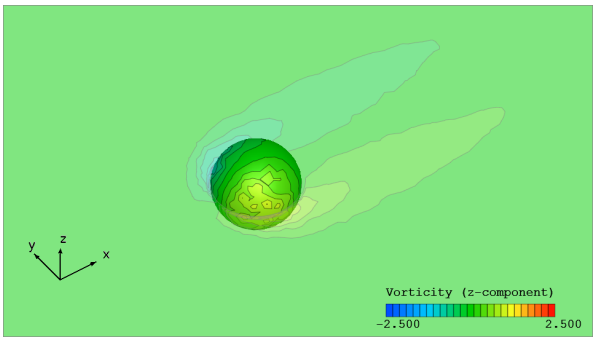


(e) TNS (unweighted LSQ) on the fine grid.



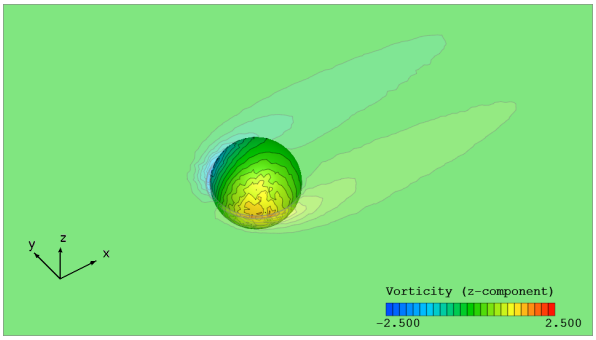
(f) HNS20 on the fine grid.

Figure 2: Comparison of the pressure contours for TNS (unweighted LSQ) and HNS20.

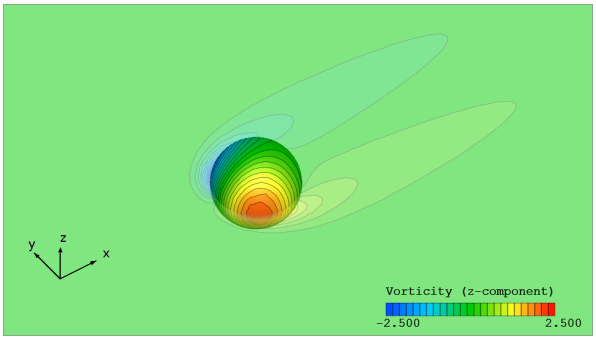


(a) TNS (unweighted LSQ) on the coarse grid.

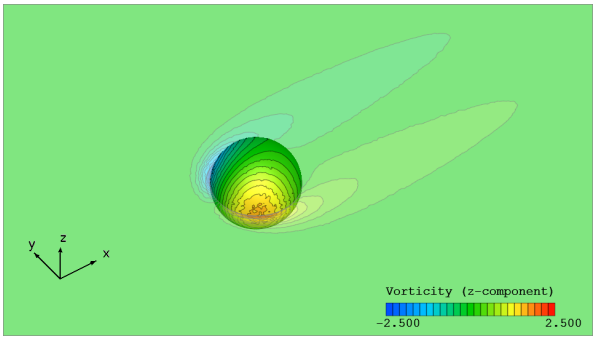
(b) HNS20 on the coarse grid.



(c) TNS (unweighted LSQ) on the medium grid.



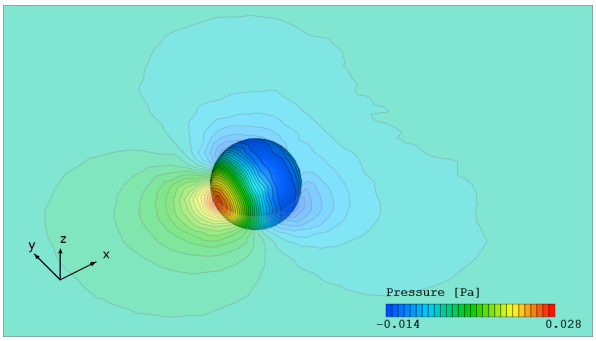
(d) HNS20 on the medium grid.



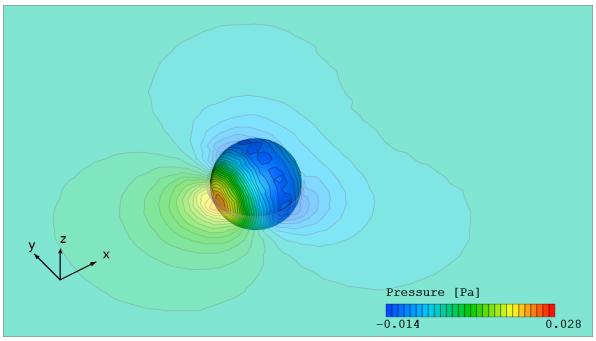
(e) TNS (unweighted LSQ) on the fine grid.

(f) HNS20 on the fine grid.

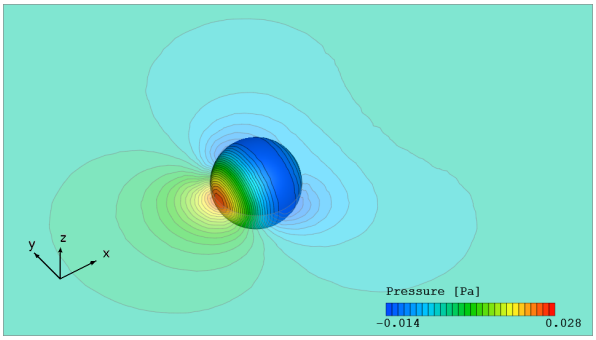
Figure 3: Comparison of the vorticity contours for TNS (unweighted LSQ) and HNS20.



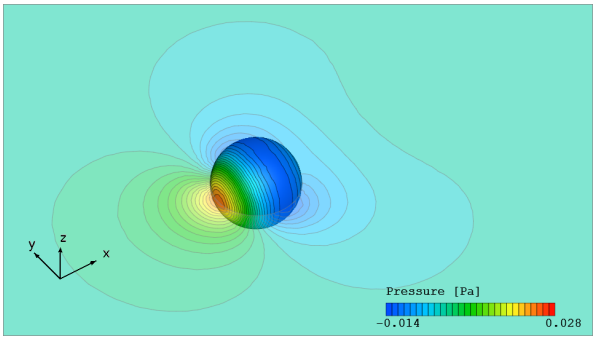
(a) TNS (weighted LSQ) on the coarse grid.



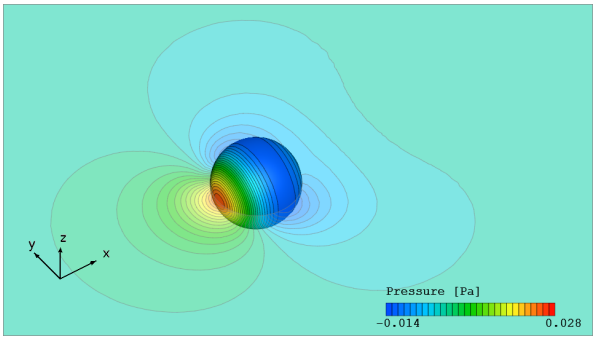
(b) HNS17 on the coarse grid.



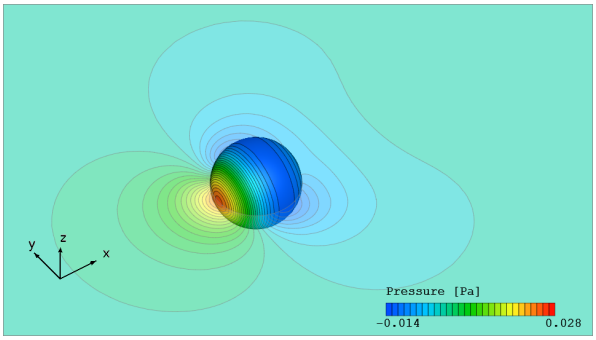
(c) TNS (weighted LSQ) on the medium grid.



(d) HNS17 on the medium grid.

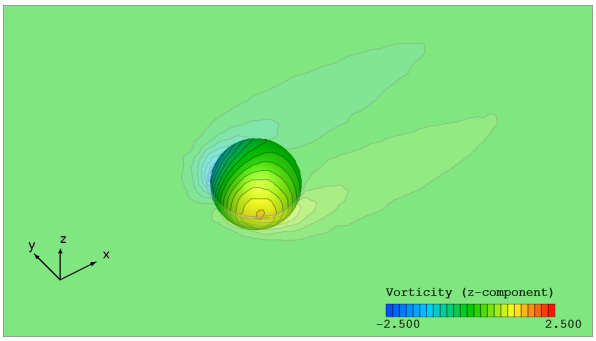


(e) TNS (weighted LSQ) on the fine grid.



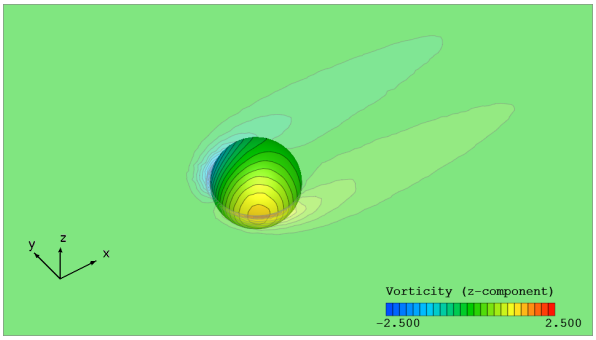
(f) HNS17 on the fine grid.

Figure 4: Comparison of the pressure contours for TNS (weighted LSQ) and HNS17.

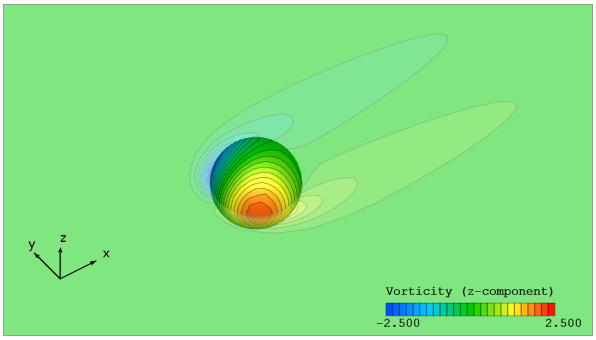


(a) TNS (weighted LSQ) on the coarse grid.

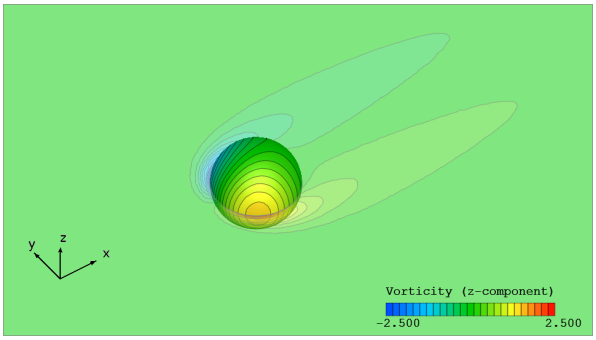
(b) HNS17 on the coarse grid.



(c) TNS (weighted LSQ) on the medium grid.



(d) HNS17 on the medium grid.



(e) TNS (weighted LSQ) on the fine grid.

(f) HNS17 on the fine grid.

Figure 5: Comparison of the vorticity contours for TNS (weighted LSQ) and HNS17.