Efficient Steady-State Convergence for a Higher-Order Unstructured Finite Volume Solver for Compressible Flows

Shayan Hoshyari [*] , Ehsan Mirzaee [†] , and Carl Ollivier-Gooch [‡]
The University of British Columbia, Vancouver, BC, V6T 1Z4, Canada

We present a three-dimensional higher-order-accurate finite volume algorithm for the solution of steady-state compressible flow problems. Higher-order accuracy is achieved by constructing a piecewise continuous representation of the average solution values using the k-exact reconstruction scheme. The pseudo-transient continuation method is employed to reduce the solution of the discretized system of nonlinear equations into the solution of a series of linear systems, which are subsequently solved using the GMRES method. We consider several preconditioning methods in conjunction with different matrix reordering algorithms, and show that our proposed preconditioner based on inner GMRES iterations can enhance the convergence speed and reduce the memory cost of the solver. Moreover, when starting from a lower-order solution as the initial condition, we show that ramping up the CFL number accelerates the convergence rate. Finally, we verify the developed finite volume algorithm by solving a set of test problems, where we attain optimal solution convergence with mesh refinement.

Nomenclature

All the quantities are dimensionless, unless otherwise stated.

$\mathbf{a}_\kappa^i$	=	ith reconstruction coefficient for control volume κ
CFL	=	Courant-Friedrichs-Lewy number
E	=	total energy
Dest	=	destruction term in the Spalart-Allmaras turbulence model
Diff	=	diffusion term in the Spalart-Allmaras turbulence model
$\boldsymbol{F}$	=	inviscid flux matrix
$\mathcal{F}$	=	numerical flux function
H	=	enthalpy
$\boldsymbol{I}$	=	identity matrix

[*]M.Sc. Student, Department of Computer Science, hoshyari@cs.ubc.ca, AIAA Member
[†]Ph.D. Candidate, Department of Mechanical Engineering, mirzaee@alumni.ubc.ca
[‡]Professor, Department of Mechanical Engineering, cfog@mech.ubc.ca, AIAA Associate Fellow

Ma	=	Mach number
n	=	time step index
N_{rec}	=	number of cells in the reconstruction stencil
P	=	pressure
Pr	=	Prandtl number
Pr_T	=	turbulent Prandtl number
$Prod$	=	production term in the Spalart-Allmaras turbulence model
LST(s)	=	CPU time spent on the linear solver
$\boldsymbol{Q}$	=	viscous flux matrix
$\mathcal{Q}$	=	numerical flux function
$\mathbf{R}$	=	discrete residual operator
Re	=	Reynolds number
$\mathbf{S}$	=	source term vector
T	=	temperature
TST(s)	=	total computational time
$Trip$	=	trip term in the Spalart-Allmaras turbulence model
$\mathbf{u}$	=	solution vector
$\mathbf{U}_h$	=	vector of average solution values
$\mathbf{u}_h$	=	discrete solution function
$\mathbf{v}$	=	velocity vector
$\mathbf{x}_{\kappa-}$, $\mathbf{x}_{\kappa+}$	=	reference locations of the adjacent control volumes
β	=	CFL growth factor
γ	=	specific heat ratio
Δt_{κ}^{n}	=	time step size for control volume κ
$\delta \mathbf{U}_h^n$	=	change in the solution average values
η	=	jump term
κ	=	control volume index
κ_{LS}	=	parameter in the line search algorithm
flux Jacobian in κ	=	
μ	=	viscosity
μ_T	=	eddy viscosity
$\tilde{\nu}$	=	turbulence viscosity

ρ	=	density
$\boldsymbol{\tau}$	=	viscous stress tensor
$\phi^i_\kappa(\mathbf{x})$	=	ith basis function for control volume κ
Ω_κ	=	volume of control volume κ
ω	=	under-relaxation parameter
$(.)^T$	=	matrix transpose

I. Introduction

Despite the significant advancement of computational fluid dynamics, there is still a high demand for computational resources when complicated problems are of interest. The high performance computation community believes that the employment of high-order solution schemes—as opposed to the traditional second-order accurate methods—has the potential to significantly reduce the computational costs [1–3]. Higher-order accurate methods can attain a practical level of accuracy on coarser meshes, which can be more efficient both in terms of solution time and memory consumption. Various approaches have been devised for achieving higher-order accuracy for aerodynamic flow problems, most notably: continuous [4] and discontinuous [5] Galerkin finite element methods, the correction procedure via reconstruction formulations of the discontinuous Galerkin and spectral volume methods [6], and finite volume schemes [7] (see reference [8] for a detailed comparison of the numerical results). We are interested in the finite volume method because it is a physically intuitive and mathematically straightforward approach, while providing the most obvious upgrade path to higher-order accuracy for the existing second-order industrial solvers. Furthermore, our emphasis is on unstructured meshes for the flexibility they offer in terms of mesh adaptivity and handling complex geometries.

In the presence of shock waves (discontinuities), during the transient state or at the steady state, high-order methods lose their monotonicity and would experience oscillations that violate the physical constraints. The intrinsic dissipation level in the numerical scheme is not sufficient to eliminate the high-frequency modes in the solution; therefore, for these problems, some shock capturing scheme such as a slope limiter [9] or artificial diffusion [10] should be used in conjunction with the k-exact reconstruction method to ensure convergence to the correct solution.

In two dimensions, unstructured high-order finite volume methods have been successfully applied to a range of aerodynamic problems: the Euler equations [11, 12], laminar Navier-Stokes equations [3, 13], and turbulent Reynolds Averaged Navier-Stokes (RANS) equations [7]. Three-dimensional results, however, are scarce and limited to the solution of the Euler and laminar Navier-Stokes equations [13, 14]. In the long run, we are interested in developing a high-order three-dimensional finite volume solver for the solution of both inviscid and viscous (laminar and turbulent) flow problems, which provides approximate error bounds on target unknowns such as lift and drag. The results obtained from solving the RANS equations contain two types of errors: modeling error and numerical error. While eliminating the unknown modeling error is out of our control, higher-order methods help us to decrease the numerical error and reduce computational costs. In this paper, we take the first steps towards our goal by considering the solution of some well-known three-dimensional benchmark problems. Moreover, due to high memory requirement of three-dimensional problems, we seek a memory-lean yet effective preconditioning scheme for the linear systems that must be solved during the simulation process.

High-order discretization schemes require high-order representations of curved boundaries [15] to maintain their nominal order of accuracy. In turbulent flow simulations, however, merely curving the boundary faces is not sufficient, for the mesh consists of highly anisotropic cells near the walls, and if we do not curve the interior faces according to the curvature of the boundaries, we will end up with self-intersecting, invalid cells. To propagate the boundary curvature into the interior mesh in 2D, we currently use a solid mechanics analogy and model the faceted mesh as an elastic solid subject to boundary deformation [7]. The same approach could be extended to handle fully three-dimensional meshes with highly anisotropic cells.

To achieve high-order accuracy, we use the k-exact reconstruction method. On anisotropic meshes, using the Cartesian coordinate system for the k-exact solution reconstruction causes numerical problems, which leads to inaccurate solution. To prevent this issue for highly anisotropic cells in 2D, we reconstruct the solution on a curvilinear coordinate system [7]. In 3D, however, the choice of this new coordinate system is not trivial, and a new technique should be devised to reconstruct the solution.

This paper begins by reviewing the Euler and turbulent RANS equations, followed by an introduction to our numerical discretization scheme, the k-exact reconstruction finite volume method. We then present the solution procedure for the discretized system of equations and report the numerical results.

II. Governing Equations

The compressible RANS equations can be written in divergence form:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\boldsymbol{F}(\mathbf{u}) - \boldsymbol{Q}(\mathbf{u}, \nabla \mathbf{u})) = \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}), \tag{1}$$

where $\mathbf{u}$ is the solution vector, $\boldsymbol{F}$ the inviscid flux matrix, $\boldsymbol{Q}$ the viscous flux matrix, and $\mathbf{S}$ the source term vector. In this work, the negative Spalart-Allmaras (S-A neg) turbulence model [16] is used in conjunction with the RANS equations.

The dimensionless solution vector and the flux matrices for the RANS + S-A neg equations are given as

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ E \\ \rho\tilde{\nu} \end{bmatrix} \quad \boldsymbol{F} = \begin{bmatrix} \rho\mathbf{v}^T \\ \rho\mathbf{v}\mathbf{v}^T + P\boldsymbol{I} \\ (E+P)\mathbf{v}^T \\ \tilde{\nu}\rho\mathbf{v}^T \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} 0 \\ \frac{Ma}{Re}\boldsymbol{\tau} \\ (E+P)\boldsymbol{\tau}\mathbf{v} + \frac{1}{\gamma-1}\left(\frac{\mu}{Pr} + \frac{\mu_T}{Pr_T}\right)\nabla T \\ -\frac{Ma}{Re\,\sigma}(\mu+\mu_T)\nabla\tilde{\nu} \end{bmatrix}, \tag{2}$$

where ρ is the fluid density, $\mathbf{v}$ the velocity vector, $\tilde{\nu}$ the S-A neg turbulence working variable, E the total energy, P the fluid pressure, T the fluid temperature, $\boldsymbol{\tau}$ the viscous stress tensor, Ma the Mach number, Re the Reynolds number, Pr the Prandtl number, Pr_T the turbulent Prandtl number, $\boldsymbol{I}$ the identity matrix, γ the specific heat ratio, σ a parameter in the S-A neg model, and $(.)^T$ denotes matrix transpose. The emphasis is on air as the working fluid, so the values $\gamma = 1.4$, $Pr = 0.72$, and $Pr_T = 0.9$ are used. The pressure is related to the other variables via the following formula:

$$P = (\gamma - 1)\left(E - \frac{1}{2}\rho(\mathbf{v}\cdot\mathbf{v})\right). \tag{3}$$

Similarly, temperature is related to pressure and density in the following form:

$$T = \frac{\gamma P}{\rho}. \tag{4}$$

In Equation (2), μ is the viscosity, and is found from Sutherland's law; μ_T is the eddy viscosity, and is found from the S-A neg model [16]. Finally, the stress tensor $\boldsymbol{\tau}$ is found via the following equation:

$$\boldsymbol{\tau} = 2(\mu + \mu_T)\left(\frac{1}{2}\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right) - \frac{1}{3}tr(\nabla\mathbf{v})\boldsymbol{I}\right). \tag{5}$$

The source term vector for the RANS + S-A neg equations is

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ Diff + \rho(Prod - Dest + Trip) \end{bmatrix}, \tag{6}$$

where *Diff*, *Prod*, *Dest*, and *Trip* represent diffusion, production, destruction, and trip terms, respectively, and are found from the S-A neg model [16].

By neglecting the turbulence model equation, the viscous flux matrix, and the source term vector in Equation (2), the Euler equations are recovered. The Euler equations govern the flow of inviscid fluids and are characterized by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ E \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \rho\mathbf{v}^T \\ \rho\mathbf{v}\mathbf{v}^T + P\boldsymbol{I} \\ (E+P)\mathbf{v}^T \end{bmatrix}, \quad \boldsymbol{Q} = \mathbf{0}, \quad \mathbf{S} = \mathbf{0}. \tag{7}$$

III. Discretization Scheme

A. The Finite Volume Method

The finite volume method seeks the vector of average solution values inside every control volume, denoted as $\mathbf{U}_h$. First, a discrete solution function, $\mathbf{u}_h = \mathbf{u}_h(\mathbf{x}, \mathbf{U}_h)$, is constructed corresponding to every set of average solution values, which is known as solution reconstruction. Herein, we use the k-exact reconstruction method that will be introduced in Section III.B. After employing a solution reconstruction method, by casting equation (1) into the integral form, and replacing $\mathbf{u}$ with $\mathbf{u}_h$, a system of ordinary differential equations is derived for $\mathbf{U}_h$:

$$\frac{d\mathbf{U}_{h,\kappa}}{dt} + \frac{1}{\Omega_\kappa}\int_{\partial\kappa}\left(\mathcal{F}(\mathbf{u}_h^+, \mathbf{u}_h^-) - \mathcal{Q}(\mathbf{u}_h^+, \nabla\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla\mathbf{u}_h^-)\right) dS - \frac{1}{\Omega_\kappa}\int_\kappa \mathbf{S}(\mathbf{u}_h, \nabla\mathbf{u}_h) d\Omega = 0, \tag{8}$$

for every control volume κ. Here, $\mathbf{U}_{h,\kappa}$ is the sub-vector of average solution values, Ω_κ the volume of κ. Although the discrete solution $\mathbf{u}_h$ and its gradient $\nabla\mathbf{u}_h$ have to be continuous inside every control volume, they can be discontinuous on the control volume boundaries $\partial\kappa$. These discontinuous values are shown using the $(.)^+$ and $(.)^-$ notations. $\mathcal{F}$ and $\mathcal{Q}$ represent the numerical flux functions, and are designed to imitate the product of the original flux matrices and the normal vector, while taking into account the discontinuity of the discrete solution. Proper quadrature formulas should be used to numerically compute the integrals in Equation (8) to a certain order of accuracy. To find the quadrature point locations, we start by mapping each element to its reference version which resides in a nondimensional space. The mapping is a polynomial of degree l that must guarantee geometric continuity across neighboring elements; thus it is usually constructed from Lagrangian basis functions (polynomials). The quadrature rules for different orders and types of reference elements are tabulated in the literature, and by a change of variables they can be constructed for elements in the physical space. Note that for meshes with curved anisotropic cells near the boundary, the quadrature point locations get updated when we solve the linear elasticity problem to propagate the boundary curvature into the domain [7].

Looking back at Equation (8), we can derive the following system of ODEs for control volume averages:

$$\frac{d\mathbf{U}_h}{dt} + \mathbf{R}(\mathbf{U}_h) = 0. \tag{9}$$

Although only the steady-state solution is of interest in this work, the unsteady term can be used to improve the robustness of the solver with respect to the initial solution guess. This method, known as pseudo-transient continuation (PTC), will be discussed in Section IV.A.

B. K-exact Reconstruction

In this work, higher-order accuracy is achieved using the k-exact reconstruction scheme [17], which results in a nominal discretization error of order $O(h^{(k+1)})$ [18]. In this method, the solution in every control volume is defined as the superposition of a set of basis functions in the following form:

$$\mathbf{u}_h(\mathbf{x};\mathbf{U}_h)|_{x\in\kappa} = \mathbf{u}_{h,\kappa}(\mathbf{x};\mathbf{U}_h) = \sum_{i=1}^{N_{\text{rec}}} \mathbf{a}_\kappa^i(\mathbf{U}_h)\phi_\kappa^i(\mathbf{x}), \tag{10}$$

where $\phi_\kappa^i(\mathbf{x})$ and $\mathbf{a}_\kappa^i$ represent the ith basis function and reconstruction coefficient for every control volume κ, respectively. Commonly, the basis functions are chosen as monomials in Cartesian coordinates with origin at the control volume's centroid of volume, $\mathbf{x}_\kappa$. Therefore, the set of basis functions can be expressed as

$$\left\{\phi_\kappa^i(\mathbf{x})|i = 1\ldots N_{\text{rec}}\right\} = \left\{\frac{1}{a!b!c!}(x_1 - x_{\kappa 1})^a(x_2 - x_{\kappa 2})^b(x_3 - x_{\kappa 3})^c|a+b+c \leq k\right\}. \tag{11}$$

Here, N_{rec} is the number of cells in the reconstruction stencil, and this particular choice of basis functions has the advantage that the discrete solution $\mathbf{u}_h$ will resemble the Taylor expansion of the exact solution $\mathbf{u}$, which is particularly useful in theoretical analysis [19].

The k-exact reconstruction requires $\mathbf{u}_{h,\kappa}$ to closely predict the average values of a specific set of the neighbors of κ, *Stencil*(κ). Moreover, the discrete solution must satisfy the conservation of the mean constraint; the reconstruction coefficients are thus found from the following constrained minimization problem:

$$\begin{aligned} \underset{\mathbf{a}_\kappa^1\ldots\mathbf{a}_\kappa^{N_{\text{rec}}}}{\text{minimize}} \quad & \sum_{\sigma\in Stencil(\kappa)} \left(\frac{1}{\Omega_\sigma}\int_\sigma \mathbf{u}_{h,\kappa}(\mathbf{x})d\Omega - \mathbf{U}_{h,\sigma}\right)^2 \\ \text{subject to} \quad & \frac{1}{\Omega_\kappa}\int_\kappa \mathbf{u}_h(\mathbf{x})d\Omega = \mathbf{U}_{h,\kappa}, \end{aligned} \tag{12}$$

which is first transformed into an unconstrained optimization problem by a change of variables [19], and then solved using the singular value decomposition (SVD) algorithm [20]. To construct *Stencil*(κ), all the neighbors at a given topological distance from κ are added to the stencil until the number of stencil members exceeds $1.5N_{\text{rec}}$. This strategy ensures that the minimization problem of Equation (12) does not become undetermined or ill-conditioned. Although it is possible to increase the number of stencil members, it would make the minimization process unnecessarily expensive.

On curved anisotropic meshes, using the Cartesian coordinate system for the k-exact solution reconstruction

(Equation (11)) leads to two related numerical issues: first, the least-squares system of Equation (12) becomes terribly ill-conditioned, and as a result, the error in estimating the reconstruction coefficients becomes of order $O(1)$. Second, the reconstruction scheme is not k-exact any more, and the discretization error is not of order $O(h^{k+1})$. To alleviate these issues in two-dimensions, we use a new curvilinear coordinate system around which the geometric entries in the least-squares system are calculated, so that the system is well-conditioned for highly anisotropic control volumes [7]. Furthermore, we use column scaling to reduce the condition number of the left-hand side matrix in the least-squares system. In three-dimensions, however, the choice of the new coordinate system is not trivial, and one might design another method to properly reconstruct the solution on anisotropic cells.

C. Numerical Flux Functions

We evaluate the inviscid flux function using a computationally efficient formulation [21] of the approximate Riemann solver of Roe [22] because of its effectiveness and simplicity.

The viscous flux functions are evaluated using an intermediate state $\mathbf{u}_h^*$, such that $Q = \boldsymbol{Q}(\mathbf{u}_h^*, \nabla\mathbf{u}_h^*)\mathbf{n}$. Finding the intermediate state as the numerical average of the left and right states, $\mathbf{u}_h^* = \frac{1}{2}\left(\mathbf{u}_h^+ + \mathbf{u}_h^-\right)$, results in a sufficiently accurate solution [23]. When evaluating the intermediate state gradient, however, simple averaging may lead to spurious solutions and instabilities. Nishikawa [24] suggested the following modified formula:

$$\nabla\mathbf{u}_h^* = \frac{1}{2}\left(\nabla\mathbf{u}_h^+ + \nabla\mathbf{u}_h^-\right) + \eta\left(\frac{\mathbf{u}_h^+ - \mathbf{u}_h^-}{\|\mathbf{x}_{\kappa+} - \mathbf{x}_{\kappa-}\|_2}\right)\mathbf{n}, \tag{13}$$

where $\mathbf{x}_{\kappa-}$ and $\mathbf{x}_{\kappa+}$ are the reference locations of the adjacent control volumes, and η is a heuristic damping factor, known as the jump term. Jalali and Ollivier-Gooch [25] have numerically tested different values of η in high-order finite volume simulations, and have recommended a value of $\eta = 1$, which is also used in this work.

IV. Solving the Discretized System of Equations

A. Pseudo-Transient Continuation (PTC)

In finding the steady-state solution of the semi-discrete system in Equation (9), time accuracy is not of interest; the stability of the time advance scheme, on the other hand, is crucial, which makes the backward Euler method an appealing choice. For the backward Euler method to be stable, all eigenvalues of the flux Jacobian matrix should reside on the left-half plane. To study this in our solver, Jalali *et al* [26] performed an eigenanalysis for diffusive fluxes on isotropic and anisotropic meshes, and showed that, for the simplified cases considered, the eigenvalues lie on the left-half plane.

The PTC method is a modification of backward Euler with nonuniform time-stepping to accelerate the global convergence. Using PTC, we can write the fully discrete version of Equation (9) as

$$\boldsymbol{V}^n \left(\mathbf{U}_h^{n+1} - \mathbf{U}_h^n\right) + \mathbf{R}(\mathbf{U}_h^{n+1}) = 0, \tag{14}$$

where n is the time step, $\mathbf{R}$ the discrete residual operator, $\boldsymbol{V}^n$ a diagonal matrix with coefficients given by $\boldsymbol{V}_\kappa^n = \frac{1}{\Delta t_\kappa^n}$, corresponding to each control volume κ. A Newton-iteration algorithm gives the linear version of Equation (14):

$$\left(\boldsymbol{V}^n + \frac{\partial \mathbf{R}}{\partial \mathbf{U}_h}\bigg|_{\mathbf{U}_h^n}\right) \delta\mathbf{U}_h^n = -\mathbf{R}(\mathbf{U}_h^n), \tag{15}$$

where $\delta\mathbf{U}_h^n$ is the change in the solution average values between the current and next iterations, and CFL is the Courant-Friedrichs-Lewy number. Moreover, $\frac{\partial \mathbf{R}}{\partial \mathbf{U}_h}\Big|_{\mathbf{U}_h^n}$ is the residual Jacobian matrix at the nth time step and is evaluated using the algorithm proposed by Michalak and Ollivier-Gooch [27]. For local time advance, at each nonlinear step n, we set a global CFL number and calculate the nth time step for each control volume κ:

$$\Delta t_\kappa^n = CFL^n \frac{h_\kappa}{\lambda_\kappa^{\max}}, \tag{16}$$

where h_κ is the hydraulic diameter of κ, and $\lambda_\kappa^{\max}$ the maximum eigenvalue of the inviscid flux Jacobian over all the surface quadrature points of κ. After $\delta\mathbf{U}_h^n$ is solved for, we update the solution:

$$\mathbf{U}_h^{n+1} = \mathbf{U}_h^n + \omega^n \delta\mathbf{U}_h^n, \tag{17}$$

where, serving as a step size along the direction $\delta\mathbf{U}_h^n$, $\omega^n \in (0 \quad 1]$ is an under-relaxation parameter calculated by a line search algorithm at each nonlinear iteration. Every time, we initialize ω^n with a physical under relaxation factor ω_{phys}, computed by imposing the physical constraints on pressure and density. If ω_{phys} satisfies the residual decrease condition *and* the new solution state is physical, then we continue with $\omega^n = \omega_{phys}$; if not, we halve ω^n repeatedly until both conditions are satisfied. The residual decrease condition ensures that at each nonlinear step, the residual does not increase beyond a certain limit:

$$\left\|\omega^n \boldsymbol{V}^n \delta\mathbf{U}_h^n + \mathbf{R}(\mathbf{U}_h^n + \omega^n \delta\mathbf{U}_h^n)\right\|_2 \le \kappa_{LS} \|\mathbf{R}(\mathbf{U}_h^n)\|_2. \tag{18}$$

Here, κ_{LS} controls the strictness of the line search algorithm, and $\kappa_{LS} = 1.2$ performs well for both viscous and inviscid flows [28]. In the presence of turbulence, the residual of the turbulence working variable is usually ill-scaled and may accordingly affect the efficiency of the line search algorithm [28]. To enhance convergence, one potential approach is to leave out the vector entries corresponding to the turbulence working variable [29] in the evaluation of the norms in

Equation (18). This we shall explore in Section V.A.

The backward Euler scheme is unconditionally stable for all time steps, and, in contrast to explicit methods, time advance is not limited by the CFL condition. Using large time steps can potentially lead to fast convergence. At transient states of the nonlinear system, however, a small time step is desirable to accelerate the convergence of Newton's method. Finding a balance between these two conflicting objectives is one of the challenges in developing efficient implicit techniques. With small CFL numbers, many time steps are required for convergence; large CFL numbers, on the other hand, may lead to breakdown of the iteration process. There are two possible reasons for the breakdown. First, too much time has been elapsed in the corresponding time step so that some flow features could not be resolved correctly, and the iteration process diverges towards a non-physical state. Second, the condition number of the coefficient matrix in Equation (15) is too large such that the linear solver cannot solve the corresponding system within a prescribed number of iterations [30].

Updating the global CFL number during nonlinear iterations helps to resolve transient states at earlier stages and enhance the convergence rate near the steady-state by recovering Newton's method. A common CFL evolution strategy is based on the value of the under-relaxation parameter used for updating the solution at each nonlinear step:

$$\mathrm{CFL}^{n+1} = \begin{cases} \beta \cdot \mathrm{CFL}^n \quad \text{for } \beta > 1 & \text{if } \omega^n = 1 \\ \mathrm{CFL}^n & \text{if } \omega_{min} < \omega^n < 1 \\ \alpha \cdot \mathrm{CFL}^n \quad \text{for } \alpha < 1 & \text{if } \omega^n < \omega_{min} \end{cases} \cdot \tag{19}$$

Here, we use $\omega_{min} = 0.01$ and $\alpha = 0.1$. The initial CFL number will be denoted as CFL^0 in this work. As mentioned earlier, we increase CFL gradually to avoid non-physical states. At the beginning of the solution process, if we use free-stream initial conditions, the approximate solution is far from its physical steady-state; small CFL numbers would then prevent divergence by making the approximate solution follow a physical transient path. Therefore, for higher order cases ($k = 3$) starting from free-stream initial conditions, we use $\beta = 1.5$. Although it increases slowly here, CFL still becomes large enough to recover Newton's iterations near the steady-state. On the other hand, when we run the solver sequentially and use the converged solution of lower-order schemes as the initial conditions for higher-order ones, since a physical solution is already at hand, we can increase CFL dramatically and still avoid divergence. In any case, CFL will be capped at 10^{10} in this work. In Section V.A, we will study the effect of increasing β on the convergence rate for sequential runs.

B. Linear Solver

Every PTC iteration requires the solution of the linear system $\boldsymbol{A}\mathbf{x} = \mathbf{b}$, where

$$\boldsymbol{A} = \left(\boldsymbol{V}^n + \frac{\partial \mathbf{R}}{\partial \mathbf{U}_h} \bigg|_{\mathbf{U}_h^n} \right), \quad \mathbf{x} = \delta \mathbf{U}_h^n, \quad \mathbf{b} = -\mathbf{R}(\mathbf{U}_h^n). \tag{20}$$

We use the generalized minimal residual (GMRES) method [31] to solve this system because of its successful history in solving various aerodynamic problems [28, 32–34]; it is also readily available as a black box solver in many numerical computation libraries, such as our library of choice: PETSc [35]. The behavior of the GMRES method is strongly dependent on the eigenvalue spectrum of $\boldsymbol{A}$—the more compact the eigenvalue spectrum, the fewer iterations required for convergence. As the left-hand-side (LHS) matrices arising from a k-exact finite volume discretization usually lack a compact eigenvalue distribution, the performance of GMRES can be greatly improved by means of preconditioning.

For right preconditioning, we need a matrix $\boldsymbol{P}$ such that the condition number of the product matrix $\boldsymbol{AP}$ is smaller than that of $\boldsymbol{A}$. Then, the original equation is changed to

$$\boldsymbol{AP}\mathbf{y} = \boldsymbol{b} \quad \mathbf{x} = \boldsymbol{P}\mathbf{y}. \tag{21}$$

Here, we first solve for the vector $\mathbf{y}$ and then use it to find $\mathbf{x}$. The preconditioning matrix can be constructed either directly from $\boldsymbol{A}$, or from the LHS matrix of a lower-order discretization scheme. The matrix from which the preconditioner is constructed will be denoted as $\boldsymbol{A}^*$. Also, note that the GMRES solver only requires the matrix-by-vector product of $\boldsymbol{P}$ and does not need $\boldsymbol{P}$ in its explicit form.

The Point Gauss-Seidel (PGS) iterative solver is a reliable preconditioner for solving linear systems arising from spatial discretization on isotropic meshes [36]. Therefore, PGS is used for the solution of Euler equations in this work, where an isotropic mesh correctly captures the solution. Nevertheless, resolving viscous flow problems requires anisotropic meshes and is a more challenging problem. In this work, we first use the incomplete lower-upper factorization of fill level p (ILUp) to form the matrix $\boldsymbol{P}$ for preconditioning the linear system in viscous turbulent problems. In this method, a lower triangular matrix, $\tilde{\boldsymbol{L}}$, and an upper triangular matrix, $\tilde{\boldsymbol{U}}$, are constructed based on $\boldsymbol{A}^*$. These matrices have small nonzero structures and satisfy $\boldsymbol{A}^* \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$ [31]. The preconditioner matrix is then set to $\boldsymbol{P} = (\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}})^{-1}$. A larger fill level results in $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ matrices with bigger nonzero structures but is likely to construct a more effective preconditioner. Since the ILU preconditioner is based on matrix factoring, its performance is dramatically affected by the ordering of the unknowns in the matrix $\boldsymbol{A}^*$. Quotient minimum degree (QMD) and reverse Cuthill-McKee (RCM) are among the reordering methods we consider in this work. QMD reduces the fill of the factored matrices, RCM the bandwidth of the matrix $\boldsymbol{A}^*$ itself. Furthermore, we will demonstrate in Section V.A that lines of control volumes with strong coupling can make an effective basis for matrix reordering.

Jalali and Ollivier-Gooch [7] observed that a fill level of three or larger is required for preconditioning of two-dimensional viscous turbulent flow problems if the $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ matrices are factored from the higher-order LHS matrix. Although this method is a practical preconditioner for two-dimensional problems, its memory cost soars drastically in three dimensions, hindering its implementation for such problems. An alternative is to factor the $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ matrices from the LHS matrix of the 0-exact (first-order) discretization scheme [32–34]. Hereinafter, the ILU preconditioner of fill level p will be denoted as LO-ILUp if factored from the 0-exact LHS matrix, and as HO-ILUp if factored from the full-order LHS matrix. Despite being conservative on memory, LO-ILU0 may behave poorly for high-order reconstruction schemes [37]. This motivates us to seek more effective preconditioners.

C. Improved Preconditioning Schemes

To improve the poor performance of the ILU preconditioning method for large higher-order problems, some strategies will be introduced: a preconditioning method based on inner GMRES iterations, (complete) LU factorization of the 0-exact LHS matrix, and LO-ILUp with larger levels of fill. Moreover, alongside these preconditioners, we introduce and examine a matrix reordering method based on lines of control volumes with strong coupling (*line* reordering, in short).

1. Inner GMRES Iterations

The 0-order LHS matrix, some researchers have argued, can construct an effective preconditioner compared to its full-order counterpart because the structure of the 0-order LHS matrix only includes the immediate neighbors of every control volume [32–34]. Nevertheless, as will be shown in Section V.A, the LO-ILU method can behave poorly for high-order reconstruction schemes applied to viscous flow problems. To mitigate this issue, we conjecture that the exact inverse of the lower-order LHS matrix, $\boldsymbol{P} = (\boldsymbol{A}^{*})^{-1}$ rather than $\boldsymbol{P} = (\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}})^{-1}$, makes a more efficient preconditioner. In this case, the matrix-by-vector product $\mathbf{z} = \boldsymbol{P}\mathbf{v}$ can be found by solving the following system:

$$\boldsymbol{A}^{*}\mathbf{z} = \mathbf{v}. \tag{22}$$

The linear system of Equation (22), however, can be as large as the original linear system $\boldsymbol{A}\mathbf{x} = \mathbf{b}$, and it might be expensive, in time and memory, to carry out the solution exactly. Instead of seeking the exact solution, the proposal is to use further inner ILU preconditioned GMRES iterations to find an approximate solution for Equation (22). The resulting preconditioners will be referred to as LO-GMRES-ILUp in this work, and their performances will be examined in Section V.A. It must be noted that when a nonlinear operator such as an inner GMRES solver is used as the preconditioner, the outer solver has to be replaced with a Flexible GMRES (FGMRES) solver [38]. The FGMRES solver can only be right preconditioned.

Finding the exact inverse matrix in $\boldsymbol{P} = (\boldsymbol{A}^*)^{-1}$ is thought to be cumbersome—especially for large problems; thus LO-GMRES-ILUp preconditioners attempt to find an approximation of $(\boldsymbol{A}^*)^{-1}$. To evaluate the efficiency of these preconditioners, it would then be helpful to introduce another preconditioner that uses (complete) LU decomposition to find the exact value of $(\boldsymbol{A}^*)^{-1}$; that is, $\boldsymbol{P} = (\boldsymbol{A}^*)^{-1} = (\boldsymbol{LU})^{-1}$. We call this preconditioner LO-LU. In serial runs, PETSc implements the LU decomposition.

2. Lines of Control Volumes with Strong Coupling

Forming non-overlapping lines that contain control volumes with strongly coupled degrees of freedom is beneficial in constructing effective preconditioners. Mavriplis [39] introduced the concept of lines in anisotropic unstructured meshes to enhance the performance of multigrid solvers via implicit smoothing. His approximate algorithm for finding such lines was purely geometric-based, and only considered control volume aspect ratios. Thus, his method only captured coupling through diffusion. Okusanya *et al* [40] later developed an algorithm that considered both the advection and diffusion phenomena. Such an algorithm, Fidkowski *et al* [41] proved, results in a set of unique lines; they used the lines for nonlinear p-multigrid solution of the Euler equations. Diosady and Darmofal [42] showed that reordering the unknowns, such that members of a line have consecutive numbers, is an effective reordering strategy for the ILU preconditioner. We implement the line reordering method of Diosady and Darmofal to study its effect on the speed and robustness of the linear solver.

In the first step of the line creation algorithm, a weight is assigned to every face inside the mesh. This weight is derived from the Jacobian of the 0-exact discretization of the advection-diffusion equation,

$$\nabla \cdot (\mathbf{v}u - \mu_L \nabla u) = 0, \tag{23}$$

where u is the unknown variable, $\mathbf{v}$ the velocity taken from the initial conditions, and μ_L an input parameter that controls the sensitivity of the lines to mesh anisotropy. The weight of a face f is then defined as

$$W(f) = \max\left(\frac{\partial R_\sigma}{\partial u_\tau}, \frac{\partial R_\tau}{\partial u_\sigma}\right), \tag{24}$$

where W is the weight of the face, σ and τ the adjacent control volumes of the face, and $\mathbf{R}$ the residual vector. If a face is located on the boundary, Equation (24) will not directly be applicable to it. Thus, a mirrored ghost control volume is placed adjacent to every boundary face. After finding the face weights, calling Algorithm 1 will construct the lines. In this algorithm, $F(\tau)$ is the set of faces of the control volume τ, and $\sigma = C(\tau, f)$ the control volume which has the face f in common with the control volume τ. This algorithm ensures that two adjacent control volumes are connected to each other if and only if they have their first or second highly weighted face lying between them [41].

Algorithm 1 Line creation

```
function CreateLines                                   ▷ Calling this function creates all lines.
    Unmark all control volumes.
    while there a exists an unmarked control volume $\tau$ do
        MakePath($\tau$)                               ▷ Forward path creation
        MakePath($\tau$)                               ▷ Backward path creation

function MakePath($\tau$)                              ▷ Helper function.
    repeat
        For control volume $\tau$, pick the face $f \in F(\tau)$ with the highest weight, such that control volume $\sigma = C(\tau, f)$
        is not part of the current line.
        if $f$ is a boundary face then
            Terminate
        else if $\sigma$ is marked then
            Terminate
        else if $f$ is not the first or second ranked face of $\sigma$ in weight then
            Terminate
        Mark $\sigma$.
        Add $\sigma$ to the current line.
        $\tau \leftarrow \sigma$
    until Termination
```

To illustrate the performance of Algorithm 1, an anisotropic mesh for the geometry of a NACA 0012 airfoil is considered, as shown in Figure 1. The velocity has been taken from the 2-exact solution of the flow resulting from a Mach number of 0.15, a Reynolds number of 6×10^6, and an angle of attack of 10 degrees. The parameter μ_L is set to the heuristic value of 10^{-5}; this value is chosen based on the ratio $\frac{Ma}{Re}$ to get an estimate of the boundary layer. As desired, the lines follow the direction of mesh anisotropy near wall boundaries; their pattern changes and follows the flow direction in other parts of the geometry.

V. Results

First, the performance of different preconditioning methods is demonstrated in computing the flow around a NACA 0012 airfoil; these computations are carried out by a single core of an Intel i7-4790 (3.60 GHz) processor. Next, the inviscid flow around a sphere is considered. Finally, our finite volume solution is verified for fully turbulent flows over a three-dimensional flat plate and an extruded NACA 0012 airfoil. All the three-dimensional flow simulations are performed using the Cedar cluster from the WestGrid computing facilities [43]. Each node of the cluster consists of two 16-core Intel Xeon E5-2683 v4 (2.1 GHz) processors, and has a minimum of 128GB memory. The wall times and memory costs are reported when using 8 processes, each running on a separate node.

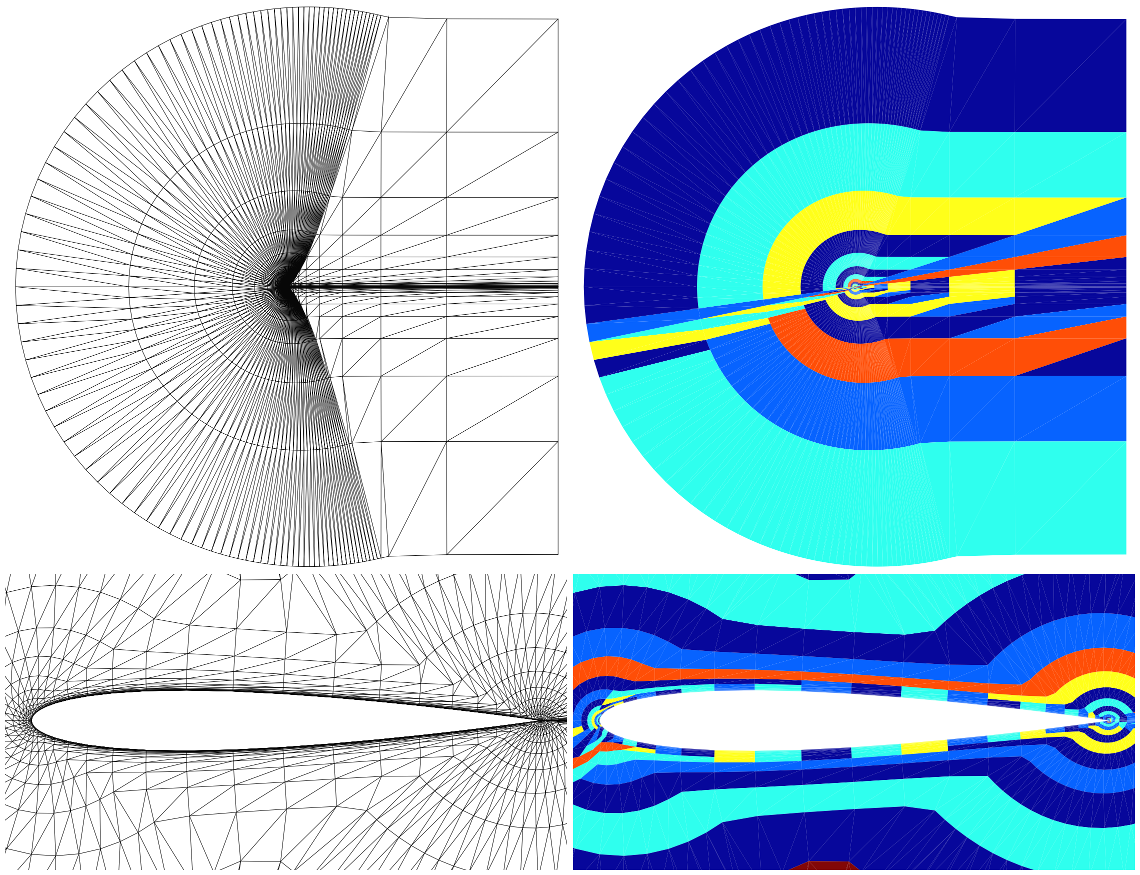


Fig. 1 Mesh and lines of control volumes with strong coupling for the geometry of a NACA 0012 airfoil

A. Preconditioner Study

The performance of different preconditioning schemes is studied by considering the turbulent flow around a NACA 0012 airfoil, with $Ma = 0.15$, $Re = 6 \times 10^6$, and an angle of attack of 10 degrees. Three nested meshes with approximately 25K, 100K, and 400K control volumes (N_{CV} = 25K, 100K, and 400K) are considered, where the coarsest mesh and its corresponding lines of control volumes with strong coupling are shown in Figure 1. The farfield boundaries are located almost 500 chords away from the airfoil, and the chord length has a nondimensional value of one. This problem is taken from the NASA Turbulence Modeling Resource (TMR) website [44], and has been previously studied by Jalali and Ollivier-Gooch [7], who used a HO-ILU3 preconditioner with QMD reordering.

The different preconditioning methods tested are shown in Table 1. In all cases, the outer GMRES solver stops when the linear residual of Equation (15) is reduced by a factor of 10^{-3}, or a maximum number of 500 iterations is performed. For inner GMRES, we do not want to spend so much time calculating an exact approximation; thus we just perform a few iterations (see Table 1). Furthermore, the simulation ends when the norm of the residual vector $||\mathbf{R}(\mathbf{U}_h)||_2$ is reduced by a factor of 10^{-8}. All cases work efficiently as *right* preconditioners, except for case A, where left preconditioning is superior. In preconditioners which involve inner GMRES iterations (cases D, E, D*, and E*), the inner GMRES solver

Table 1 Preconditioning methods considered

Case name	Preconditioning method	Reordering method	Preconditioner side	Inner GMRES preconditioner side	Number of inner GMRES iterations
A	HO-ILU3	QMD	left	—	—
B	LO-ILU0	RCM	right	—	—
C	LO-ILU0	line	right	—	—
D	LO-GMRES-ILU0	RCM	right	left	10
E	LO-GMRES-ILU0	line	right	left	10
F	LO-LU	QMD	right	—	—
B^*	LO-ILU3	RCM	right	—	—
D^*	LO-GMRES-ILU3	RCM	right	left	5
E^*	LO-GMRES-ILU3	line	right	left	5

is better off with a *left* ILU preconditioner. Note that other combinations of preconditioner and reordering schemes that are not considered in Table 1, either had inferior performances or did not result in convergence even for the coarse mesh.

The residual history of the solver for each preconditioning scheme is shown in Figure 2, where the highest-order discretization scheme (3-exact) is considered, starting from free-stream initial conditions with $\beta = 1.5$ and $\text{CFL}^0 = 10^{-2}$; here the residual entries of the turbulence working variable are excluded in the line search process (Equation 18). For the cases plotted in Figure 2, Table 2 shows the number of PTC iterations (N-PTC), the number of outer GMRES iterations (N-GMRES), total memory consumption, CPU time spent on the linear solver (LST), and total computational time (TST). On the coarse mesh ($N_{\text{CV}} = 25\text{K}$), most preconditioners work alike in terms of the number of PTC iterations, but cases B^* and D^* run in shorter times, and show a speed-up of three compared to case A. There is approximately a twofold reduction in memory consumption for the cases using the 0-order LHS matrix rather than full-order. This reduction in the required memory is also evident from the data acquired on the medium mesh.

It is the medium mesh ($N_{\text{CV}} = 100\text{K}$) results that reveal more about the differences between all these preconditioners. Here, cases B and C fail to converge; these two LO-ILU0 cases apparently lack enough information to sufficiently reduce the condition number of the coefficient matrix in the linear system. On the other hand, cases D, E, B^*, and D^* converge and are on a par in terms of PTC iterations, while cases B^* and D^* stand out with regard to timing: case B^* runs approximately eight times faster than case A, case D^* five times faster.
Strikingly, Case F takes more PTC iterations to converge compared to LO-GMRES-ILU cases (except E^*), which are trying to get an approximation of $(\boldsymbol{A}^*)^{-1}$, not the exact value. Case B^* is a modification to case B with three levels of fill, and it performs almost twice as many total linear iterations as D^*, but still is faster. Linear iterations are cheaper for B^* because each GMRES iteration at case D^* comprises five inner iterations itself. We suspend our judgment about the best preconditioning scheme until we see the convergence history for sequential runs over the medium mesh and the results on the fine mesh ($N_{\text{CV}} = 400\text{K}$).

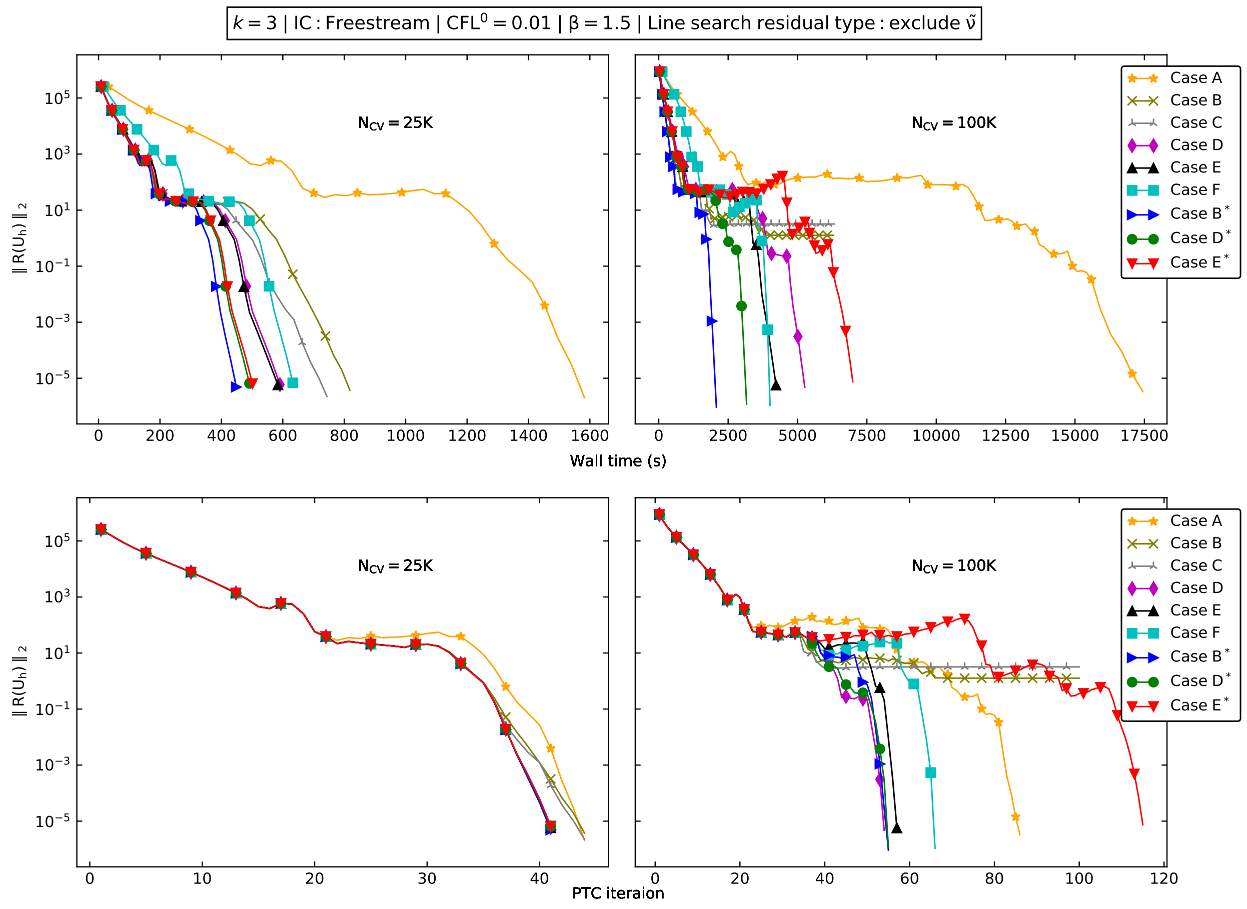


Fig. 2 Comparison of residual histories obtained using different preconditioning schemes (NACA 0012 problem, 1 process)

Using the line reordering method, case E outperforms its corresponding preconditioner with RCM reordering (case D) in speed, but still comes short compared to cases B* or D*. The inner ILU preconditioner uses a fill level of zero in case E and three in case D*; thus case D* provides a better approximation of $(\boldsymbol{A}^*)^{-1}$, requiring a smaller number of inner iterations, which in turn makes it faster. Case E*, which combines the line reordering and an inner ILU3 preconditioner, struggles to converge and requires many PTC iterations—but is still faster than case A after all. We might conclude that line reordering has some shortcomings when a higher level of fill is required.

The turbulence closure equation is not usually well-scaled. Hence, the residuals of the turbulence working variable may have different magnitudes compared to those of the other variables. In evaluating the residual in the line search algorithm (Equation (18)), therefore, it might be beneficial to omit these ill-scaled residual components. Figure 3 depicts the behavior of different preconditioning schemes when omitting or considering $\tilde{\nu}$ residuals in the line search. In most cases, the advantage of omitting those residuals is evident both in number of PTC iterations and in wall time. We therefore continue to exclude $\tilde{\nu}$ residuals in the line search algorithm.

Gradually increasing the CFL number helps ensure the solution follows a physical path towards the steady state. On

Table 2 Performance comparison for different preconditioning schemes (NACA 0012 problem, 1 process). Cases with entries marked by "–" did not converge to the steady state solution.

$k = 3$	$\text{CFL}^0 = 0.01$	$\beta = 1.5$	Initial condition: Freestream		
Preconditioner	N-PTC	N-GMRES	Memory(GB)	LST(s)	TST(s)
			$N_{\text{CV}} = 25\text{K}$		
A	44	1, 104	1.55	1, 254	1, 582
B	44	12, 360	0.72	452	818
C	44	9, 963	0.73	363	744
D	41	2, 193	0.90	246	591
E	41	2, 163	0.84	234	584
F	41	1, 752	0.88	289	632
B*	41	2, 759	0.74	102	449
D*	41	1, 868	0.84	147	491
E*	41	1, 904	0.86	158	501
			$N_{\text{CV}} = 100\text{K}$		
A	86	11, 272	5.81	14, 756	17, 444
B	–	–	2.73	–	–
C	–	–	2.80	–	–
D	54	7, 130	3.18	3, 362	5, 264
E	57	4, 987	3.26	2, 210	4, 226
F	66	3, 309	3.66	2, 357	4, 011
B*	55	6, 803	3.74	745	2, 077
D*	55	3, 833	3.22	1, 275	3, 166
E*	115	6, 432	3.67	2, 163	6, 992

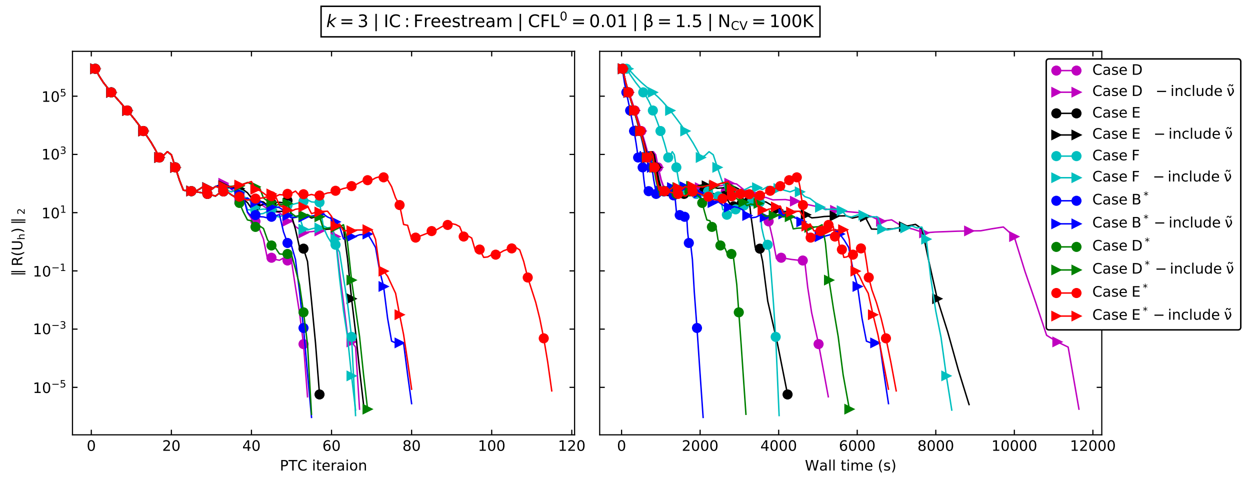


Fig. 3 Effect of excluding the $\tilde{\nu}_T$ residual in the line search algorithm on convergence using different preconditioning schemes (NACA 0012 problem, 1 process)

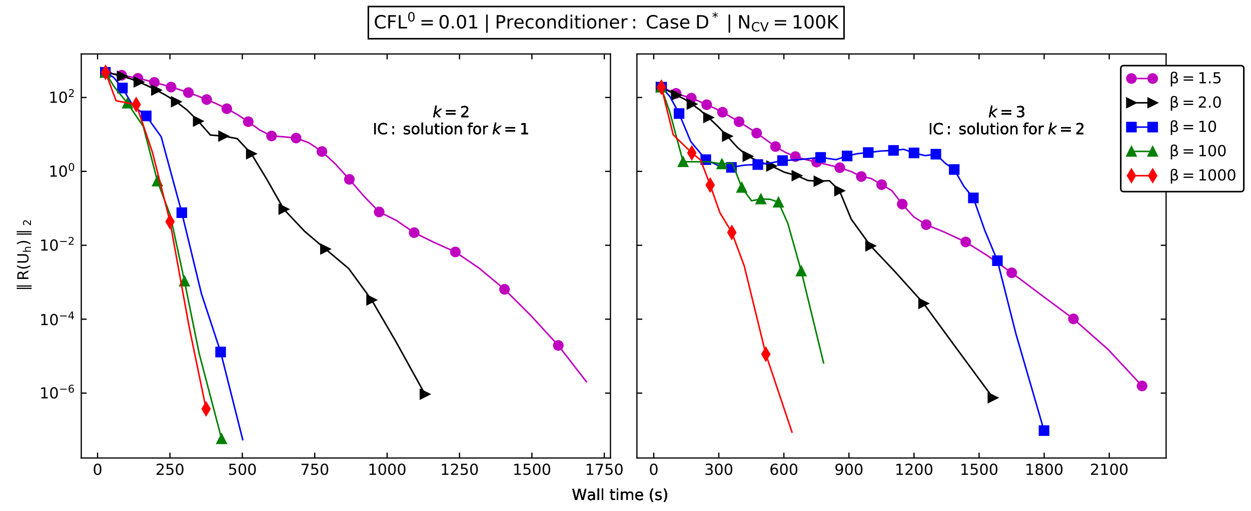


Fig. 4 Effect of increasing CFL growth factor on convergence for 2- and 3-exact discretizations starting from a lower-order converged solution (NACA 0012 problem, 1 process)

the other hand, running the high-order solver sequentially helps preserve the robustness of the scheme; that is, first running a lower-order scheme to the steady state, and then using the lower-order solution as the initial condition for the higher-order scheme. In the sequential run, since a physical solution is available as the initial conditions for 2- and 3-exact schemes, we argue that the CFL growth factor (β) can be increased to boost up the convergence rate. Figure 4 shows the effect of increasing β on the convergence of the 2-exact scheme starting from the 1-exact solution, and the 3-exact scheme starting from the 2-exact solution. In both cases, the solver converges faster as we increase β—a fourfold increase with $\beta = 1000$ compared to $\beta = 1$. In the 3-exact case, the combination of CFL^0 and β drives the solution to a more challenging path, but the solver is sufficiently robust to eventually converge to the steady state.

The sequential run histories in Figure 5, for B* and D* preconditioners, show that although the case B* is faster for the 1-exact portion, starting from free-stream initial conditions, the overall wall time is less for case D*. Furthermore, Figure 6 demonstrates the superior performance of case D* on the fine mesh, where case B* fails to converge to the 3-exact steady state. Our speculation is that case B*, since it only performs an incomplete factorization on the lower-order matrix, lacks enough information to reduce the condition number of the coefficient matrix for the finest mesh, whereas case D*, since it approximates the exact inverse of the lower order matrix, extracts more information from the matrix, which in turn reduces the condition number more effectively. Now is the time to make an informed judgment about the most promising preconditioning method, and use the case D* preconditioner for the three-dimensional problems accordingly.

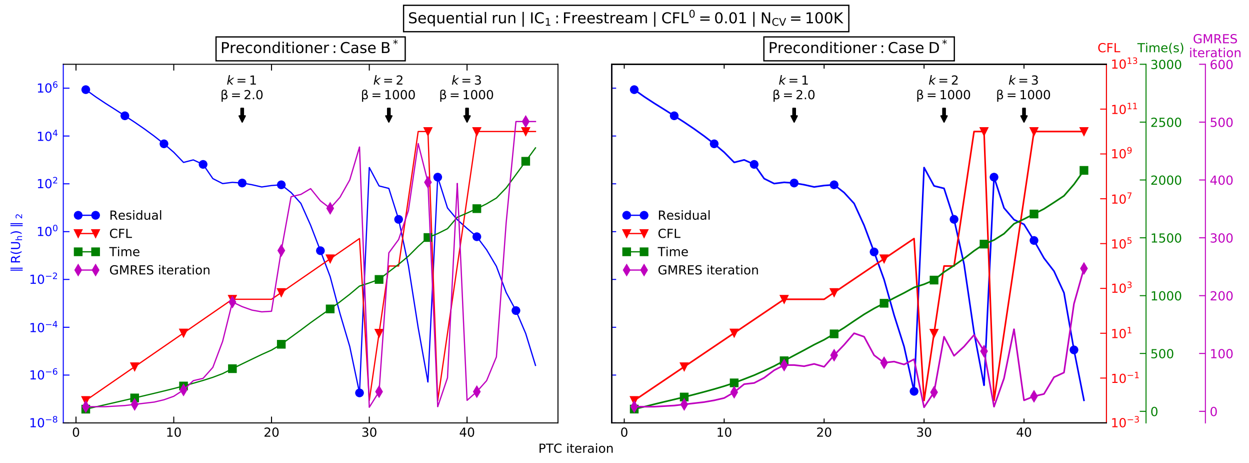


Fig. 5 Sequential convergence history for B* and D* preconditioning schemes (NACA 0012 problem, 1 process)

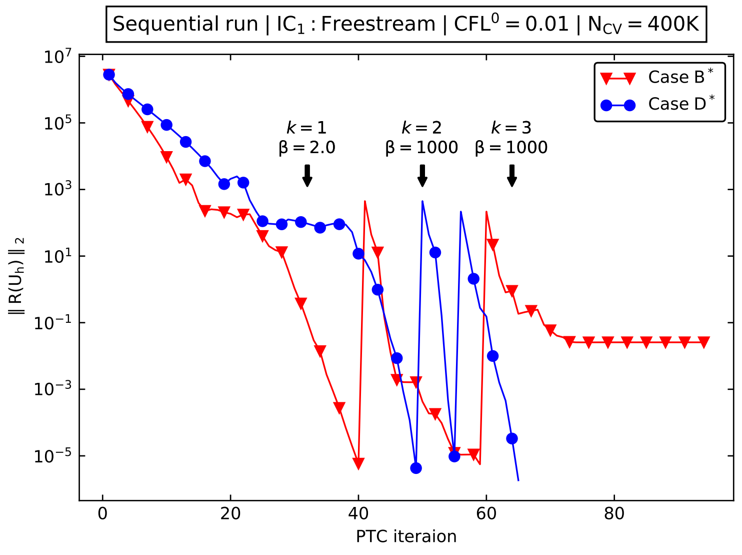


Fig. 6 Comparison of Sequential residual history for B* and D* preconditioning schemes (NACA 0012 problem, 1 process)

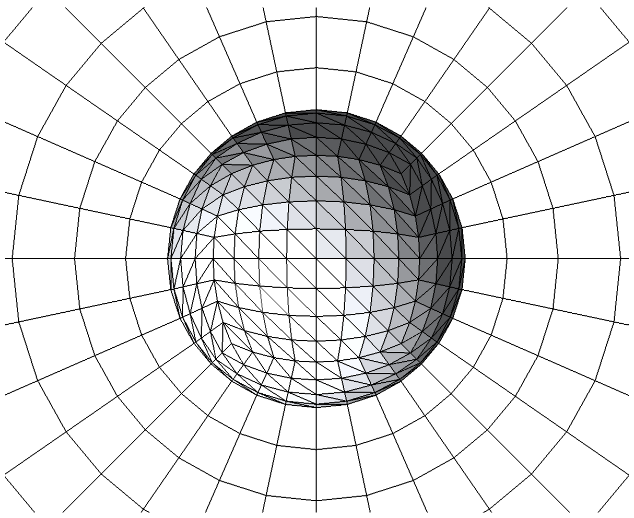


Fig. 7 Symmetric cut of the coarsest mesh near the sphere surface

B. Inviscid Flow Over a Sphere

The second test problem is the inviscid flow around a sphere with unit radius. The Mach number is equal to $Ma = 0.38$, and the free-stream flow is in the x_1 direction. This problem is chosen as a three-dimensional extension of the inviscid flow around a circle, the higher-order solution of which was studied by Bassi and Rebay [15]. Three prismatic grids (64K, 322K, and 1M control volumes) are used, where the farfield is located at a distance of 100 dimensionless units from the sphere center. A symmetric cut of the coarsest mesh near the sphere surface is shown in Figure 7. The faces and control volumes adjacent to the wall boundaries are curved using second-order Lagrange polynomials. To ensure that numerical quadrature does not introduce additional error in the solution, a quadrature scheme accurate up to order $k + 1$ is employed.

As the flow does not contain any discontinuities, the entropy must be constant throughout the domain. Therefore, the L_2 norm of the entropy relative to the free-stream conditions, $||S - S_\infty||_2$, has an exact value of zero. The convergence of this quantity versus mesh refinement is shown in Figure 8, where the mesh length scale is defined as $h = (N_{\text{CV}})^{1/3}$, and N_{CV} is the number of control volumes. The $k = 1$ solution converges even faster than its expected ratios of 2, $k = 3$ faster than the ratio of 4. The convergence ratio for the $k = 2$ solution, however, is half an order smaller than its expected value of 3.

The effect of the reconstruction order k on solution accuracy is also evident from the qualitative Mach contours on the $x_3 = 0$ symmetry plane, which are plotted in Figure 9 for the finest mesh. Not surprisingly, only the $k = 3$ solution has been able to capture the symmetry of the flow to a great extent. Furthermore, the artificial wake behind the sphere seems small for the $k = 2$ solution compared to that of $k = 1$.

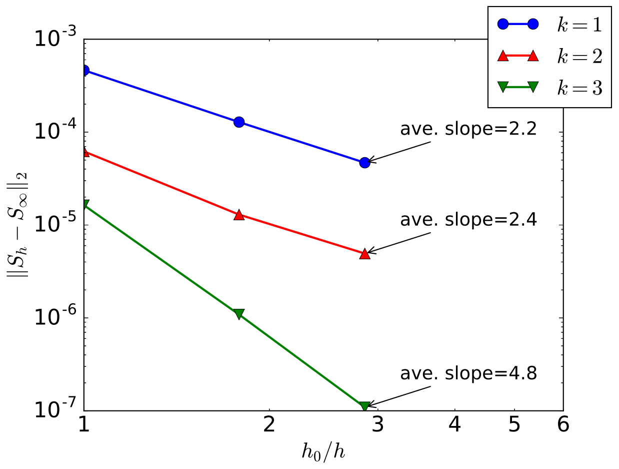


Fig. 8 Relative entropy norm versus mesh size for the sphere problem

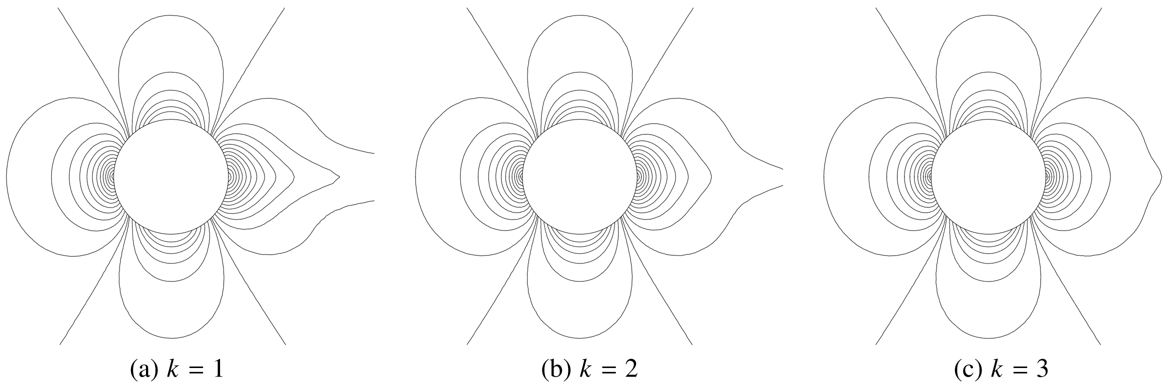


Fig. 9 Computed Mach contours on the $x_3 = 0$ symmetry plane for the sphere problem on the 1M control volume mesh

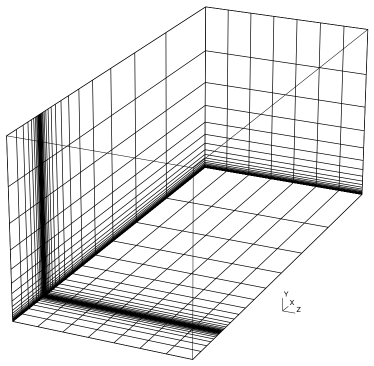


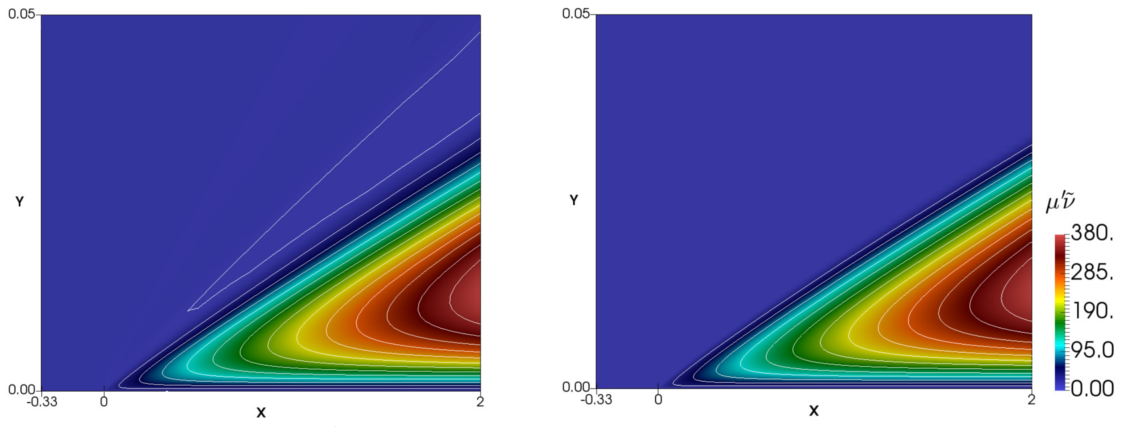
Fig. 10 Coarsest mesh for the flat plate problem

C. Turbulent Flow Over a Flat Plate

Now, we consider a three-dimensional extension of the flat plate verification case from the NASA TMR website. The computational domain is $\Omega = [-0.33, 2] \times [0, 1] \times [0, 1]$, the nondimensional parameters are $Re = 5 \times 10^6$ and $Ma = 0.2$, and the flow is in the x_1 direction. The flat plate is located on the $(0 \leq x_1 \leq 2) \wedge (x_2 = 0)$ boundary, where adiabatic solid wall conditions are imposed. Symmetry boundary conditions are imposed on the $(x_3 = 0)$, $(x_3 = 1)$, and $(-0.33 \leq x_1 \leq 0) \wedge (x_2 = 0)$ boundaries, while other boundaries are considered as farfield. Since we use symmetry boundary conditions on all the boundaries normal to the x_3-axis, the exact solution of this problem must be constant in the x_3 direction. Therefore, the solutions obtained in this section can be compared to the two-dimensional results from the NASA TMR website, which are obtained using the CFL3D solver [45] on a 544×384 grid.

The two-dimensional meshes provided by the NASA TMR website are extruded in the x_3 direction to construct a series of nested three-dimensional grids with the following dimensions: $60 \times 34 \times 7$, $120 \times 68 \times 14$, and $240 \times 136 \times 28$. The coarsest mesh is depicted in Figure 10. The medium and fine meshes are uniform refinements of the coarse and medium meshes, respectively. An anisotropic mesh is necessary to correctly capture the flow pattern in the boundary layer and at the plate leading edge. On each mesh, the problem is solved sequentially, first for $k = 1$, then $k = 2$, and finally $k = 3$. The free-stream conditions are used as the initial conditions for $k = 1$, while the initial solution for each $k \geq 2$ is taken from the converged solution of the $(k - 1)$-exact scheme. For better performance, here we combine the preconditioning method of case D* with a block Jacobi preconditioner, for the domain is decomposed between eight processes. Moreover, $\beta = 1000$ is used for $k = 2$ and $k = 3$ portions of the sequential run.

To schematically demonstrate the correctness of the solution, the distribution of the turbulence working variable obtained from $k = 3$ on the finest mesh is plotted on the $x_3 = 0.5$ plane, and shown in Figure 11a. The results for a 544×384 grid provided by the NASA TMR are also shown in Figure 11b. There is a close agreement between the reference solution of NASA TMR and the solution of this work, although a few undershoots of small magnitude are present in the latter. The next quantity of interest is the distribution of the nondimensional eddy viscosity $\frac{\mu_T}{\mu}$ on the line $(x_1 = 0.97) \wedge (x_3 = 0.5)$, which is depicted in Figure 12 along with the reference solution from the NASA TMR website.



(a) 3-exact: min : 7×10^{-3}, max : 379 (b) NASA TMR: min : 0, max : 378

Fig. 11 Distribution of the turbulence working variable on the plane $x_3 = 0.5$ **for the flat plate problem**

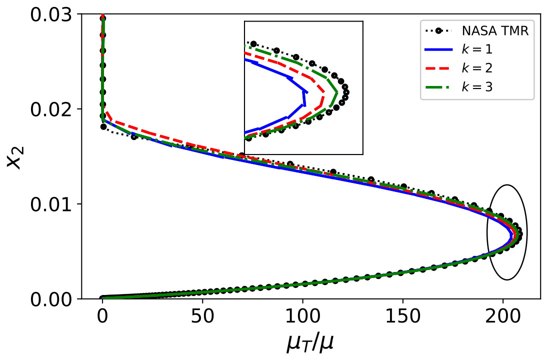


Fig. 12 Distribution of the nondimensional eddy viscosity on the line $(x_1 = 0.97) \wedge (x_3 = 0.5)$ **for the flat plate problem on the** $240 \times 136 \times 28$ **grid**

Our computed results agree with the reference values. Also, it is evident that a higher-order reconstruction scheme leads to a more accurate estimate of the eddy viscosity.

To further verify the numerical solution, the convergence of the drag coefficient C_D and the skin friction coefficient C_f at the point $\mathbf{x} = (0.97, 0, 0.5)$ are studied with mesh refinement. Table 3 shows the computed values on different meshes. The computed values converge faster for a higher-order reconstruction scheme, such that only $k = 3$ offers an accurate solution on the medium mesh. The reference values from the NASA TMR library are also shown in the same table. As desired, the computed values in this work agree with the reference values of the NASA TMR website. The convergence orders of C_D and C_f are computed using the procedure of Celik *et al* [46], and are shown in Table 3 along with the cumulative convergence time for each case. All the reconstruction schemes have attained optimal convergence rates. Nevertheless, care must be taken when interpreting the estimated convergence rates because the procedure of Celik *et al* is most reliable when the error has already achieved an asymptotic behavior on the coarsest mesh, which might not necessarily be the case here.

The sequential residual history for different mesh sizes is plotted in Figure 13. Increasing the number of control

Table 3 Computed value and convergence order of the drag coefficient and the skin friction coefficient at the point $\mathbf{x} = (0.97, 0, 0.5)$ for the flat plate problem

	C_D			C_f			Cumulative CPU time (s)		
NASA TMR	0.00286			0.00271			—		
Mesh \ k	1	2	3	1	2	3	1	2	3
$60 \times 34 \times 7$	0.00396	0.00233	0.00233	0.00350	0.00228	0.00222	36	65	97
$120 \times 68 \times 14$	0.00301	0.00281	0.00285	0.00283	0.00268	0.00271	401	634	1061
$240 \times 136 \times 28$	0.00287	0.00286	0.00286	0.00274	0.00273	0.00273	4614	6238	9466
Convergence order	2.8	3.3	5.4	3	3	5.1			

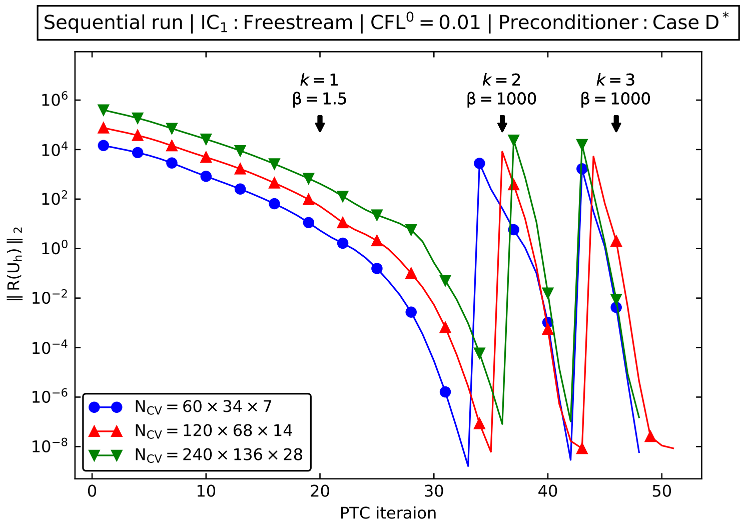


Fig. 13 Sequential residual history for different meshes (flat plate problem, 8 processes)

volumes does not affect the overall residual history with respect to PTC iterations; the cost of each PTC iteration, on the other hand, increases as we refine the mesh. The convergence history for the sequential run on the fine mesh is shown in Figure 14.

The specifics and resource consumption information of the flow solver for the flat plate problem are listed in Table 4. For the $k = 2$ and $k = 3$ cases, the results are taken when a lower order solution is used as initial condition. Note that memory consumption increases linearly with mesh refinement. For $k = 1$ the majority of the total time is spent on linear solves, but for $k = 2$ and $k = 3$, the linear solve share decreases dramatically due to the increase in β and the subsequent lower number of PTC iterations. Most importantly, the benefit of higher-order methods can be observed in Table 3 and Table 4: while the $k = 3$ solution on the medium mesh offers the same level of accuracy for C_D as the $k = 1$ solution on the fine mesh (for C_f, the $k = 3$ solution on the medium mesh is even more accurate), the cumulative computational time of the former is smaller than that of the latter by a factor of four, and the memory consumption is smaller by a factor of three.

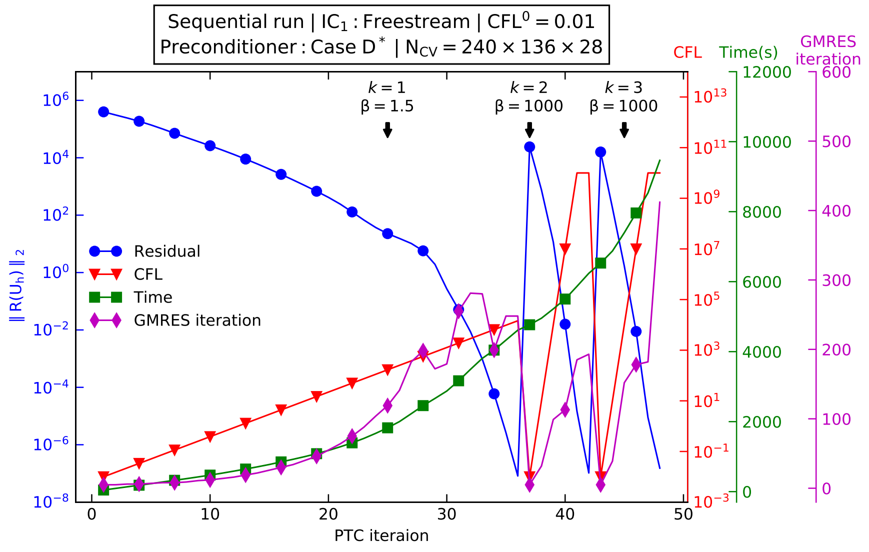


Fig. 14 Sequential convergence history (flat plate problem, 8 processes)

Figure 15 shows the condition number of the linear matrix at different nonlinear iterations. The results are taken by solving the flat plate problem on the coarse mesh, for $k = 1$, $k = 2$, and $k = 3$, with or without preconditioning. Case E is used as the preconditioner, and for all cases we use $CFL^0 = 0.1$ and $\beta = 1.5$. Although we usually ramp up the CFL number for higher-order cases when they start from a lower-order converged solution, but here we keep $\beta = 1.5$ for all simulations to prevent fast convergence of the higher-order cases to keep track of the condition number. A GMRES solver is used to report the condition number and the maximum number of linear iterations is set to 500, so for iterations where the solver reaches this maximum without converging, as is the case for most unpreconditioned PTC iterations, the reported condition number is not accurate. The maximum number of PTC iterations is set to 35, and none of the unpreconditioned cases converge in this range. According to Figure 15, preconditioning dramatically reduces the condition number of the linear system for earlier PTC iterations, but as we go forward the linear system becomes stiffer. On the other hand, for the cases considered here, the effect of reconstruction order on the stiffness of the linear system seems to be marginal.

To test the parallel scalability of the flow solver, a strong parallel scaling test is conducted for the flat plate test case, i.e., the same problem is solved by using different numbers of processors ranging from 1 to 10 while the resulting speedup is recorded. The test is done for the $k = 3$ scheme on the $120 \times 68 \times 14$ mesh. The parallel speedup of the total solution process, the Jacobian evaluation algorithm, and the linear solver scheme were computed separately, and are shown in Figure 16. Not surprisingly, the linear solver algorithm is less scalable than that of the Jacobian integrator mainly because the efficiency of the block Jacobi preconditioner decreases with an increase in the number of processors. The sudden dip of the parallel speedup for 9 processors may be due to a less favorable mesh partitioning, or communication problems on the cluster. With regard to weak scaling, where the problem size assigned to each processor

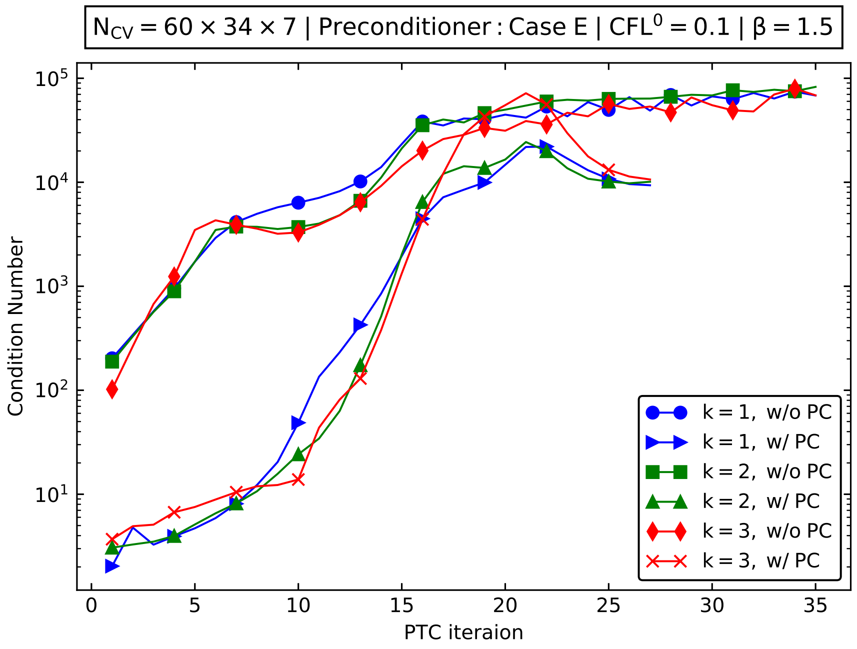


Fig. 15 Effect of preconditioning and reconstruction order on the the stiffness of the linear system (flat plate problem, 1 process)

is fixed and each time a larger problem is solved by more processors, our tests for the flat plate case show the weak scaling efficiency to be approximately 94% for different number of processors ranging from 1 to 10.

D. Turbulent Flow Over an Extruded Airfoil

For the last test case, we consider a three-dimensional extension of the flow around the NACA 0012 airfoil. The computational domain of Section V.A is extruded in the x_3 direction with an extrusion length of one nondimensional unit. Symmetry boundary conditions are imposed on the two boundaries normal to the x_3-axis, while other boundaries retain their conditions from Section V.A. The nondimensional parameters are set to $Re = 6 \times 10^6$ and $Ma = 0.15$, and the angles of attack and side slip are equal to 10 and zero degrees, respectively. Since the exact solution must be constant in the x_3 direction, the solutions are compared to the reference values provided by NASA TMR, which are obtained using the FUN3D [47] solver on a 7169×2049 grid.

Two meshes are employed: a hexahedral mesh with $N_{\text{CV}} = 100$K, and a mixed prismatic-hexahedral mesh with $N_{\text{CV}} = 176$K. In both cases, quadrature points are obtained from the tensor product of one-dimensional quadrature formulas and the quadrature points of the curved two-dimensional meshes. A portion of the meshes is depicted in Figure 17. In both cases, there are 7 layers of extruded control volumes in the x_3 direction. Again, we run the solver sequentially here, as we did for the NACA 0012 and flat plate problems, increasing β while we increase the order of accuracy. As for the flat plate problem, the preconditioning scheme of case D* is combined with a block Jacobi preconditioner when the domain is decomposed and the solver uses multiple processes.

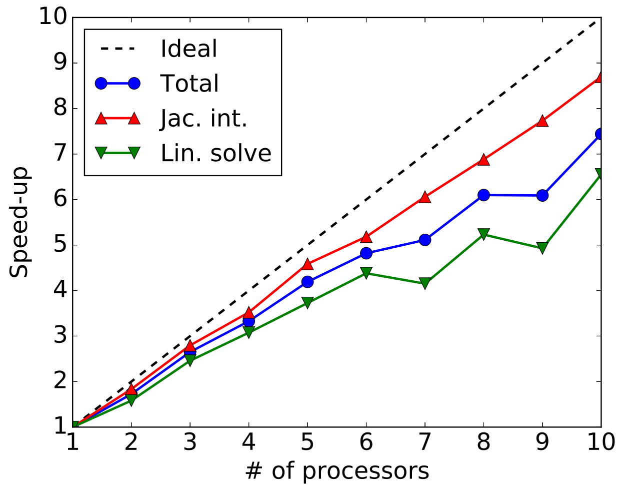


Fig. 16 The parallel speedup of the solver for the flat plate problem: $120 \times 68 \times 14$ **mesh and** $k = 3$**.**

Table 4 Specifics and resource consumption of the flow solver (flat plate problem, 8 **processes)**

Preconditioner: Case D*							
k	CFL^0	β	N-PTC	N-GMRES	Memory(GB)	LST(s)	TST(s)
$60 \times 34 \times 7$ mesh							
1	0.01	1.5	33	901	0.35	12	36
2	0.01	1,000	9	351	1.13	6	29
3	0.01	1,000	6	259	1.75	5	32
$120 \times 68 \times 14$ mesh							
1	0.01	1.5	35	1,578	4.57	219	401
2	0.01	1,000	8	445	6.45	66	233
3	0.01	1,000	8	662	11.47	128	428
$240 \times 136 \times 28$ mesh							
1	0.01	1.5	36	3,158	32.51	3,236	4,614
2	0.01	1,000	6	618	49.48	729	1,624
3	0.01	1,000	6	955	96.05	1,510	3,228

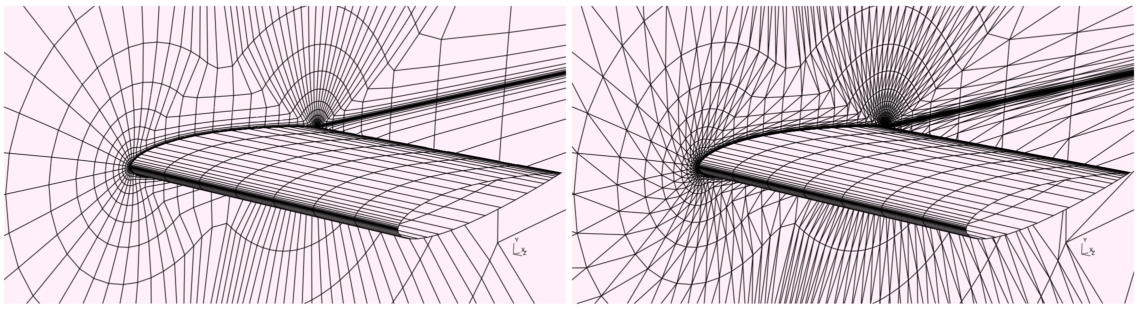

(a) Hexahedral mesh (b) Mixed prismatic-hexahedral mesh

Fig. 17 Meshes for the extruded NACA 0012 problem

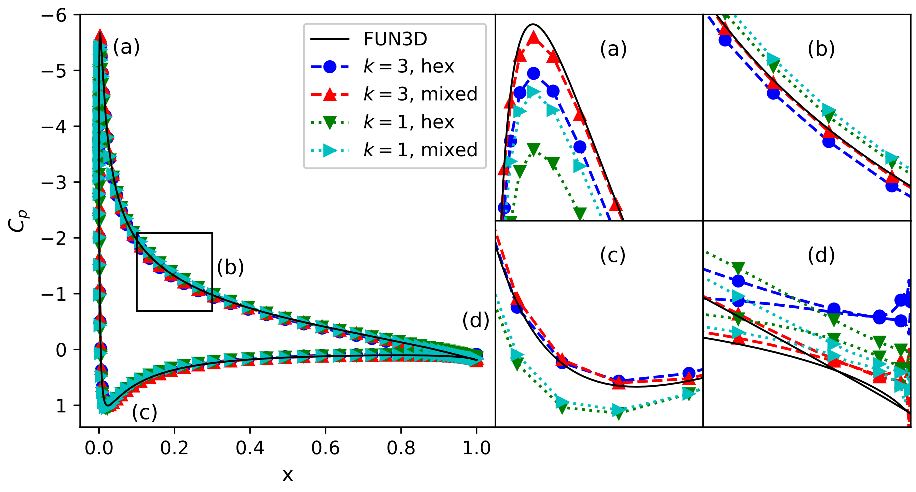


Fig. 18 Distribution of surface pressure coefficient on the intersection of the extruded NACA 0012 airfoil and the $x_3 = 0.5$ plane

The distribution of the surface pressure coefficient on the intersection of the airfoil and the $x_3 = 0.5$ plane is computed using the $k = 1$ and 3 solutions, and is depicted in Figure 18. Generally, the computed distributions are consistent with the reference values obtained by FUN3D. Closer views of the plots are shown for four distinct points to better compare the results. As expected, the $k = 3$ scheme predicts the most accurate values, particularly on the mixed mesh. For example, the FUN3D results show that the pressure coefficient on the lower surface becomes bigger than that of the upper surface, near the trailing edge of the airfoil. This phenomenon is captured by the $k = 3$ solutions, but not observed by those of $k = 1$.

The computed and the reference lift and drag coefficients are shown in Table 5. It seems that the hexahedral mesh is too coarse, as none of the schemes can obtain accurate enough solutions. For the mixed mesh, the $k = 3$ scheme gives satisfactory results with less than 10% error for all the coefficients. The solution of the other schemes, however, is quite off, particularly for the pressure drag coefficient C_{D_P}. Note that the reference results are obtained using a two-dimensional 7169×2049 mesh, which has approximately a thousand times more cells than the meshes in each layer of our extruded cases.

The sequential residual history for different mesh sizes is shown in Figure 19. Convergence with respect to PTC iterations is marginally affected by the mesh type . Moreover, the effect of increasing β on reducing the number of PTC iterations for $k = 2$ and 3 is obvious. The convergence history for the sequential run on the mixed mesh is shown in Figure 20.

Finally, parameters related to the iterative convergence of the solver are listed in Table 6. The linear solve time makes up around 30% of the total computation time. The total computation time is dependent on k, but does not differ considerably between the two meshes for the same reconstruction order, especially when measured per degree of

Table 5 Computed drag and lift coefficients for the extruded NACA 0012 airfoil problem

k	C_{Dp}	C_{Dv}	C_L
NASA TMR			
–	0.00607	0.00621	1.0910
Hex mesh			
1	0.01703	0.00582	1.0619
2	0.01702	0.00497	1.0507
3	0.00301	0.00472	1.0417
Mixed mesh			
1	0.01129	0.00574	1.0735
2	0.00365	0.00565	1.0776
3	0.00550	0.00536	1.0869

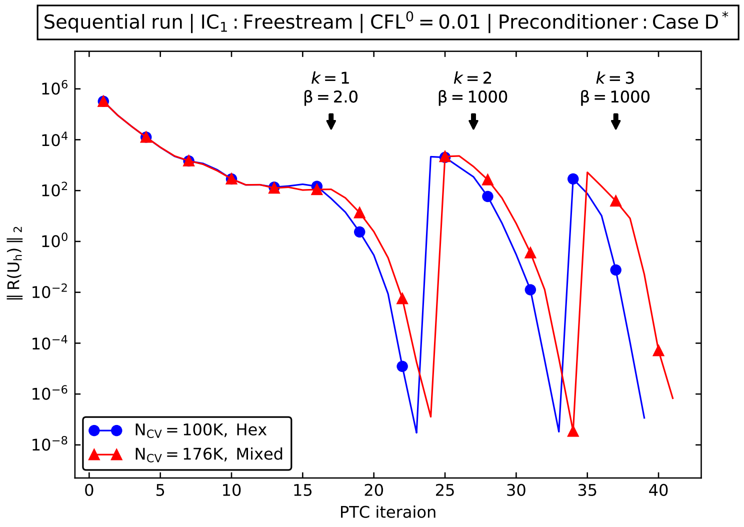


Fig. 19 Sequential residual history for different meshes (extruded NACA 0012 airfoil problem, 8 processes)

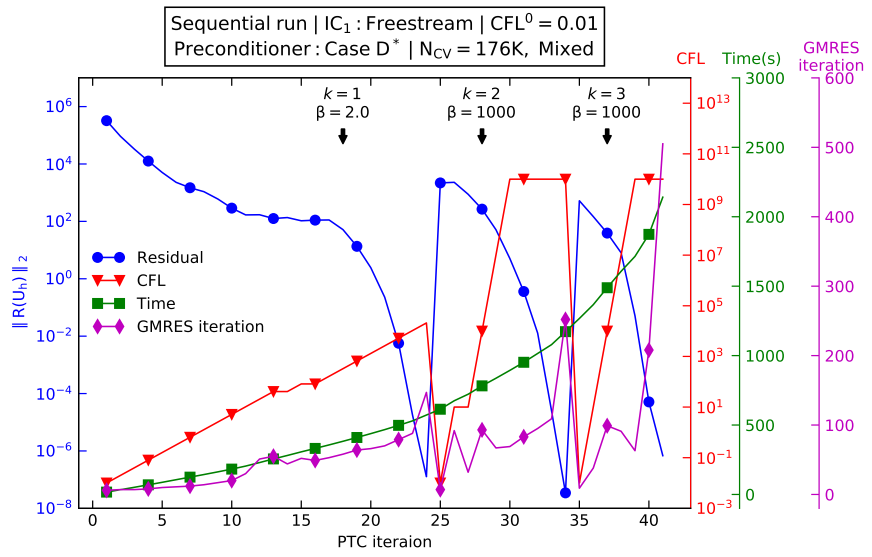


Fig. 20 Sequential convergence history (extruded NACA 0012 airfoil problem, 8 processes)

Table 6 Specifics and resource consumption of the flow solver (extruded NACA 0012 problem, 8 processes)

Preconditioner: Case D*							
k	CFL^0	β	N-PTC	N-GMRES	Memory(GB)	LST(s)	TST(s)
Hex mesh, N_{CV} = 100K							
1	0.01	2.0	23	794	3.99	86	322
2	0.01	1,000	10	896	5.86	130	482
3	0.01	1,000	6	815	9.43	145	507
Mixed mesh, N_{CV} = 176K							
1	0.01	2.0	24	1,018	7.11	179	572
2	0.01	1,000	10	899	8.76	188	601
3	0.01	1,000	6	1,013	20.53	352	968

freedom. Moreover, the memory consumption seems to be almost linearly increasing with both k and N_{CV}.

VI. Conclusions

A higher-order solution framework was presented for steady-state three-dimensional compressible flows. The highlights of this solution framework include the following: a k-exact finite volume discretization, a pseudo-transient continuation (PTC) solution algorithm, and a memory-lean preconditioner based on inner GMRES iterations.

Higher-order accuracy was achieved by constructing a piecewise continuous representation of the control volume average values. This piecewise continuous representation, also referred to as the discrete solution, was defined as the superposition of a set of basis functions, defined as monomials of the Cartesian coordinates.

The PTC method was revisited for the solution of the discretized nonlinear equations, where each iteration required the solution of a linear system. The challenges involved in the solution of these linear systems were addressed for three-dimensional problems, and a novel method was introduced to mitigate these issues. In this method, the FGMRES linear solver was preconditioned using inner GMRES iterations, which were subsequently preconditioned by the ILU method. The $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ matrices were factored from the LHS matrix of the 0-exact discretization scheme. By considering the 2-D fully turbulent flow around the NACA 0012 airfoil, it was shown that the winning preconditioning algorithm (case D*) is efficient in terms of solution time, memory consumption, and robustness, compared to the other preconditioning methods considered. Furthermore, in higher-order schemes, our results showed that we can ramp up the CFL number when starting from a lower-order converged solution as the initial condition. This reduces the total number of PTC iterations and enhances the speed of convergence.

Moreover, inviscid and viscous turbulent test problems were considered, where the solution was verified against reference values either obtained analytically or provided by the NASA TMR website. In all cases, a satisfactory agreement was observed between the solutions of this work and the reference values. Accuracy tests were also performed with mesh refinement, and optimal convergence order was attained for reconstruction orders $k = 1$ and 3. The $k = 2$ scheme, however, was slightly inferior in performance compared to its theoretical convergence order.

Timing and memory consumption results demonstrated the benefit and practicality of using higher-order methods for obtaining a certain level of accuracy. Compared to a higher-order discretization scheme, for the cases tested, a second-order scheme required a finer grid and more computational time to attain the same level of accuracy.

VII. Acknowledgments

Funding for this research is provided by the University of British Columbia and the Canadian Natural Sciences and Engineering Research Council (NSERC) under grant 2015-04005; the authors acknowledge their support and also appreciate the computational resources provided by Compute Canada.

References

[1] Leicht, T., and Hartmann, R., "Anisotropic mesh refinement for discontinuous Galerkin methods in two-dimensional aerodynamic flow simulations," *International Journal for Numerical Methods in Fluids*, Vol. 56, No. 11, 2008, pp. 2111–2138. doi:10.1002/fld.1608.

[2] Mavriplis, D., Darmofal, D., Keyes, D., and Turner, M., "Petaflops opportunities for the NASA fundamental aeronautics program," *Proceedings of the Eighteenth AIAA Computational Fluid Dynamics Conference*, 2007, p. 4084. doi:10.2514/6.2007-4084.

[3] Jalali, A., and Ollivier Gooch, C. F., "An hp-Adaptive Ustructured Finite Volume Solver for Compressible Aerodynamic Flows," *Proceedings of the Fifty-Fifth AIAA Aerospace Sciences Meeting*, 2017, p. 0082. doi:10.1002/fld.4396.

[4] Anderson, W. K., Wang, L., Kapadia, S., Tanis, C., and Hilbert, B., "Petrov–Galerkin and discontinuous-Galerkin methods for time-domain and frequency-domain electromagnetic simulations," *Journal of Computational Physics*, Vol. 230, No. 23, 2011, pp. 8360–8385. doi:10.1016/j.jcp.2011.06.025.

[5] Hartmann, R., and Leicht, T., "Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration," *International Journal for Numerical Methods in Fluids*, 2016. doi:10.1002/fld.4219.

[6] Huynh, H., Wang, Z. J., and Vincent, P. E., "High-order methods for computational fluid dynamics: a brief review of compact differential formulations on unstructured grids," *Computers and Fluids*, Vol. 98, 2014, pp. 209–220. doi:10.1016/j.compfluid.2013.12.007.

[7] Jalali, A., and Ollivier-Gooch, C., "Higher-Order Unstructured Finite Volume RANS Solution of Turbulent Compressible Flows," *Computers and Fluids*, Vol. 143, 2017, pp. 32–47. doi:10.1016/j.compfluid.2016.11.004.

[8] Andren, J., Gao, H., Yano, M., Darmofal, D., Ollivier Gooch, C., and Wang, Z., "A comparison of higher-order methods on a set of canonical aerodynamics applications," *Proceedings of the Twentieth AIAA Computational Fluid Dynamics Conference*, 2011, p. 3230. doi:10.2514/6.2011-3230.

[9] Sweby, P. K., "High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws," *SIAM Journal on Numerical Analysis*, Vol. 21, No. 5, 1984, pp. 995–1011. doi:10.1137/0721062.

[10] Barter, G. E., and Darmofal, D. L., "Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation," *Journal of Computational Physics*, Vol. 229, No. 5, 2010, pp. 1810–1827. doi:10.1016/j.jcp.2009.11.010.

[11] Haider, F., Brenner, P., Courbet, B., and Croisille, J.-P., "Parallel implementation of k-exact finite volume reconstruction on unstructured grids," *High-Order Nonlinear Numerical Schemes for Evolutionary PDEs*, Springer, 2014, pp. 59–75. doi:10.1007/978-3-319-05455-1_4.

[12] Michalak, C., and Ollivier-Gooch, C., "Accuracy preserving limiter for the high-order accurate solution of the Euler equations," *Journal of Computational Physics*, Vol. 228, No. 23, 2009, pp. 8693–8711. doi:10.1016/j.jcp.2009.08.021.

[13] Li, W., *Efficient Implementation of High-Order Accurate Numerical Methods on Unstructured Grids*, Springer, 2014. doi:10.1007/978-3-662-43432-1.

[14] Haider, F., Croisille, J.-P., and Courbet, B., "Stability analysis of the cell centered finite-volume MUSCL method on unstructured grids," *Numerische Mathematik*, Vol. 113, No. 4, 2009, pp. 555–600. doi:10.1007/s00211-009-0242-6.

[15] Bassi, F., and Rebay, S., "High-order accurate discontinuous finite element solution of the 2D Euler equations," *Journal of Computational Physics*, Vol. 138, No. 2, 1997, pp. 251–285. doi:10.1006/jcph.1997.5454.

[16] Allmaras, S. R., Johnson, F. T., and Spalart, P., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," *Proceedings of the Seventh International Conference on CFD*, 2012.

[17] Barth, T. J., and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction," *Twenty-eighth AIAA Aerospace Sciences Meeting*, 1990. doi:10.2514/6.1990-13, aIAA paper 90-0013.

[18] Barth, T. J., and Larson, M. G., "A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes," *Finite Volumes for Complex Applications III, London*, 2002.

[19] Ollivier-Gooch, C. F., and Van Altena, M., "A High-order Accurate Unstructured Mesh Finite-Volume Scheme for the Advection-Diffusion Equation," *Journal of Computational Physics*, Vol. 181, No. 2, 2002, pp. 729–752. doi:10.1006/jcph.2002.7159.

[20] Golub, G. H., and Loan, C. F. V., *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1983.

[21] Roe, P., and Pike, J., "Efficient construction and utilisation of approximate Riemann solutions," *Proceedings of the Sixth International Symposium on Computing Methods in Applied Sciences and Engineering, VI*, North-Holland Publishing Co., 1985, pp. 499–518.

[22] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372. doi:10.1016/0021-9991(81)90128-5.

[23] Nishikawa, H., "Two ways to extend diffusion schemes to Navier-Stokes schemes: Gradient formula or upwind flux," *Proceedings of the Twentieth AIAA Computational Fluid Dynamics Conference*, 2011, p. 3044. doi:10.2514/6.2011-3044.

[24] Nishikawa, H., "Beyond Interface Gradient: A General Principle for Constructing Diffusion Scheme," *Proceedings of the Fortieth AIAA Fluid Dynamics Conference*, 2010. doi:10.2514/6.2010-5093, aIAA paper 2010-5093.

[25] Jalali, A., and Ollivier Gooch, C., "Accuracy assessment of finite volume discretizations of diffusive fluxes on unstructured meshes," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2012, p. 608. doi:10.2514/6.2012-608.

[26] Jalali, A., Sharbatdar, M., and Ollivier-Gooch, C., "Accuracy analysis of unstructured finite volume discretization schemes for diffusive fluxes," *Computers and Fluids*, Vol. 101, 2014, pp. 220–232. doi:10.1016/j.compfluid.2014.06.008.

[27] Michalak, C., and Ollivier-Gooch, C., "Globalized matrix-explicit Newton-GMRES for the high-order accurate solution of the Euler equations," *Computers and Fluids*, Vol. 39, 2010, pp. 1156–1167. doi:10.1016/j.compfluid.2010.02.008.

[28] Ceze, M., and Fidkowski, K. J., "Constrained pseudo-transient continuation," *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 11, 2015, pp. 1683–1703. doi:10.1002/nme.4858.

[29] Wallraff, M., Hartmann, R., Leicht, T., Kroll, N., Hirsch, C., Bassi, F., Johnston, C., and Hillewaert, K., "Multigrid solver algorithms for DG methods and applications to aerodynamic flows," *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, Vol. 128, 2015, pp. 153–178. doi:10.1007/978-3-319-12886-3_9.

[30] Bücker, H. M., Pollul, B., and Rasch, A., "On CFL evolution strategies for implicit upwind methods in linearized Euler equations," *International journal for numerical methods in fluids*, Vol. 59, No. 1, 2009, pp. 1–18. doi:10.1002/fld.1798.

[31] Saad, Y., *Iterative Methods for Sparse Linear Systems*, SIAM, 2003. doi:10.1137/1.9780898718003.ch4.

[32] Pueyo, A., and Zingg, D. W., "Efficient Newton-Krylov solver for aerodynamic computations," *AIAA Journal*, Vol. 36, No. 11, 1998, pp. 1991–1997. doi:10.2514/2.326.

[33] Luo, H., Baum, J. D., and Löhner, R., "A fast, matrix-free implicit method for compressible flows on unstructured grids," *Sixteenth International Conference on Numerical Methods in Fluid Dynamics*, Springer, 1998, pp. 73–78. doi:10.1007/BFb0106564.

[34] Wong, P., and Zingg, D. W., "Three-dimensional aerodynamic computations on unstructured grids using a Newton–Krylov approach," *Computers and Fluids*, Vol. 37, No. 2, 2008, pp. 107–120. doi:10.1016/j.compfluid.2007.04.005.

[35] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K., Smith, B. F., Zampini, S., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015. URL `http://www.mcs.anl.gov/petsc`.

[36] Mavriplis, D. J., "Directional agglomeration multigrid techniques for high-Reynolds-number viscous flows," *AIAA Journal*, Vol. 37, No. 10, 1999, pp. 1222–1230. doi:10.2514/2.590.

[37] Nejat, A., and Ollivier-Gooch, C., "Effect of discretization order on preconditioning and convergence of a high-order unstructured Newton-GMRES solver for the Euler equations," *Journal of Computational Physics*, Vol. 227, No. 4, 2008, pp. 2366–2386. doi:10.1016/j.jcp.2007.10.024.

[38] Saad, Y., "A flexible inner-outer preconditioned GMRES algorithm," *SIAM Journal on Scientific Computing*, Vol. 14, No. 2, 1993, pp. 461–469. doi:10.1137/0914028.

[39] Mavriplis, D. J., "Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes," *Journal of Computational Physics*, Vol. 145, No. 1, 1998, pp. 141–165. doi:10.1006/jcph.1998.6036.

[40] Okusanya, T., Darmofal, D., and Peraire, J., "Algebraic multigrid for stabilized finite element discretizations of the Navier–Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 193, No. 33, 2004, pp. 3667–3686. doi:10.1016/j.cma.2004.01.025.

[41] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 207, No. 1, 2005, pp. 92–113. doi:10.1016/j.jcp.2005.01.005.

[42] Diosady, L. T., and Darmofal, D. L., "Preconditioning methods for discontinuous Galerkin solutions of the Navier–Stokes equations," *Journal of Computational Physics*, Vol. 228, No. 11, 2009, pp. 3917–3935.

[43] "WestGrid computing resources," , 2017. URL `https://www.westgrid.ca/`.

[44] Rumsey, C., "Turbulence Modeling Resource," , 2014. URL `http://turbmodels.larc.nasa.gov/`.

[45] "Computational Fluids Laboratory 3-D," , 2011. URL `https://cfl3d.larc.nasa.gov/`.

[46] Celik, I. B., Ghia, U., and Roache, P. J., "Procedure for estimation and reporting of uncertainty due to discretization in CFD applications," *Journal of Fluids Engineering-Transactions of the ASME*, Vol. 130, No. 7, 2008.

[47] "Fully Unstructured Navier–Stokes in 3-D," , 2011. URL `http://fun3d.larc.nasa.gov/`.