Scalable Solution Strategies for Stabilized Finite-Element Flow Solvers on Unstructured Meshes

Behzad R. Ahrabi[1] and Dimitri J. Mavriplis[2]

Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071

The objective of the present study is to investigate and develop robust, efficient, and scalable multilevel solution strategies and preconditioning techniques for stabilized finite-element flow solvers. The proposed solution strategy is essentially a p-multigrid approach in which the solution on the mesh with lowest polynomial degree (p=1) is solved using a preconditioned Newton-Krylov method. Seeking robustness, the Newton-Krylov methods are commonly preconditioned using incomplete factorization methods such as ILU(k). However, it is well known that these methods are not scalable. To overcome this problem, we have developed an implicit line preconditioner which can be properly distrusted among processing elements, without affecting the convergence behavior of the linear system and the non-linear path. The lines are extracted from a stiffness matrix based on strong connections. To improve the robustness of the implicit-line relaxation, a dual CFL strategy, with a lower CFL number in the preconditioner matrix, has been developed. In this study, we also employ an algebraic multigrid (AMG) preconditioner and augment it with the dual-CFL strategy. To reach high-order discretizations, a set of hierarchical basis functions are employed and a non-linear p-multigrid approach is developed. In order to test the performance of the newly developed algorithms, a new flow solver for two and three dimensional mixed-element unstructured meshes has been developed in which the Reynolds averaged Navier-Stokes (RANS) equations and negative Spalart-Almaras (SA) turbulence model are discretized, in a coupled form, using the Streamline Upwind Petrov-Galerkin (SUPG) scheme. Several two- and three-dimensional numerical examples including turbulent flows over the 30P30N three element airfoil and a wing-body-tail configuration from the 4th AIAA Drag Prediction Workshop are presented in which the performance of the implicit line relaxation and algebraic multigrid preconditioners are compared with the incomplete lower upper factorization (ILU(k)) method. Results show positive steps toward development of scalable solution techniques.

I. Introduction

During last three decades, second-order finite-volume (FV) schemes have played a crucial role in advancements of aerodynamic simulations, particularly in the context of complex geometries. However, due to practical limitations on order of accuracy, these methods require excessively fine grids to provide high-fidelity simulations. This shortcoming has motivated many computational fluid dynamics (CFD) researchers in recent years to develop high-order finite-element (FE) schemes in order to reach the desired accuracy at minimal computational cost. However, despite significant progress in development of high-order discretizations such as Petrov-Galerkin (PG) and discontinuous Galerkin (DG) schemes, development of robust, efficient, and scalable solution techniques for the discretized equations resulting from these methods has generally been lagging. The importance of this problem is better realized by noting that concurrently, hardware technology has progressed rapidly and now, high performance computing (HPC) power has reached a level which is not

[1]AIAA member, Post-Doctoral Research Associate, brezaahr@uwyo.edu

[2]AIAA Associate Fellow, Professor, mavripl@uwyo.edu

readily usable by non-specialized solution techniques. This stems from the fact that modern parallelization paradigms have been designed based on minimal communication between processing elements. In the following, it is described why multigrid and multilevel methods are powerful candidates to get the most benefit out of modern HPC architectures and also, how the elements of these methods form the theme of this paper.

Historically, multigrid and multilevel methods have played a dominant role in aerospace CFD codes. A major characteristic of these methods is optimality, meaning that these methods are capable of computing the solution to a problem with N unknowns in $O(N)$ operations. Lately, interest in and use of these methods for aerospace applications has fallen off mainly due to automation and robustness issues [1]. Seeking robustness, there has been a recent trend towards incomplete factorization matrix methods, generally using these methods with various levels of fill as preconditioners to Newton-Krylov methods for current FV and emerging FE discretizations. However, due to the approximate nature of these methods, they need to be applied iteratively (as the preconditioner) and the overall number of iterations increases with the problem size. In addition, for stiffer problems resulting from higher-order discretizations, higher fill levels are required. This results in a non-optimal solver which is difficult to parallelize. Note that in the context of domain-decomposition, most often factorization methods are applied locally within each partition, which leads to decreased convergence rates with increasing levels of parallelization. Consequently, the robustness offered by these methods on smaller problems may not be feasibly reachable on larger problems. On the other hand, multigrid and multilevel methods, in addition to being theoretically optimal, are well suited for current and emerging massively parallel heterogeneous architectures largely due to the fact that these methods rely on local smoothers on each level of the multilevel sequence. Thus, global implicitness is achieved through reduced long-range communication on coarser levels, while local errors are reduced effectively in parallel on finer levels. Furthermore, the flexibility afforded in the design of multilevel smoothers enables the numerical analyst to optimize these methods for emerging HPC hardware which include high-latency limited bandwidth hierarchical memory configurations.

The principal objective of this work is to investigate and develop robust, efficient, and scalable multilevel solution strategies and preconditioning techniques for stabilized finite-element methods. The focus will be put on the different aspects and requirements of steady-state problems, while seeking to maximize parallel scalability. It should be noted that by stabilized finite-element methods we mean Petrov-Galerkin (PG) formulations in which the discrete solution space is assumed to be continuous. In this approach, the lowest polynomial function space is made by linear basis functions which results in a second-order accurate scheme. To build higher-order spaces, in this study, hierarchical basis functions are employed. Having this setup, the proposed multilevel solution strategy can be viewed as a spectral multigrid method in which the solution on the mesh with the lowest polynomial degree (coarsest mesh) is solved to the desired accuracy using a preconditioned Newton-Krylov method. In this study, we investigate the components of this multilevel system. In particular, efficient smoothers for each level are investigated and developed. For preconditioning on the coarsest mesh, implicit line-relaxation and algebraic multigrid (AMG) methods are employed and their performances are compared with incomplete lower upper (ILU) factorization method. For AMG preconditioning, Hyper/BoomerAMG [2] is used. For implicit-line relaxation, the lines are extracted from a stiffness matrix based on strong connections. In order to maintain the robustness of the line-relaxation methods in parallel processing, special attention is devoted to domain decomposition so that partitioning is done without cutting the lines.

It should be mentioned that examples of developments of spectral and algebraic multigrid methods in PG discretizations of Euler and Navier-Stokes (NS) equations can be found in studies by Okusanya et al. [3] and Helenbrook [4], respectively. In the present work, application of multigrid methods for Reynolds averaged Navier-Stokes (RANS) is studied. In summary, the long-term goal of this work is the development of a solution algorithm which is able to deliver h- (mesh resolution) and p- (degree of polynomial space) independent

convergence rates and minimizes the cost of each iteration. Although this goal has not been fully accomplished, positive and effective steps have been taken.

In the following sections, first, the features of the newly developed flow solver for this study is briefly reviewed. Next, the governing equations and PG discretization are described. Different solution methodologies and preconditioning techniques are described next, and finally, the numerical results and conclusions are presented.

II. Description of the Flow Solver

We refer to the flow solver developed in the present work as the HOMA (High-Order Multilevel Adaptive) solver. In this flow solver, the compressible Reynolds averaged Navier-Stokes (RANS) equations are discretized, in a coupled form, using Petrov-Galerkin finite-element method, and the solution space is assumed to be continuous. The lowest polynomial space is constructed using a linear basis and high-order discretizations are built using a hierarchical basis. The solver is capable of handling two- and three-dimensional unstructured meshes with mixed element types and iso-, sub-, and super-parametric geometric mappings. The time integration for steady-state and unsteady problems is fully implicit and the linearization is done exactly using automatic differentiation (AD). Unsteady problems are solved using the second-order backward difference formula (BDF2) scheme. For steady-state problems, a preconditioned Newton-Krylov method based on pseudo-transient continuation (PTC), and a non-linear spectral multigrid method based on the full approximation scheme (FAS) are used. For preconditioning of the linear systems, incomplete lower upper factorization with arbitrary levels of fill (ILU(k)), implicit line smoothers, and Additive Schwarz methods with various local solvers have been developed as the built-in options of the code. Also, the PETSc toolkit [5-7] has been integrated into the solver, through which the external packages as Hyper/BoomerAMG [2] and Trilinos/ML [8] are accessible for the preconditioning.

A. History

The two-dimensional implementation of the mentioned multilevel flow solver originates from the FUNSAFE framework [9-13]. The original code is able to perform adjoint-based hp-adaptations and for this purpose it utilizes a hierarchical set of basis functions [12, 14]. Having the capability to perform the desired p- and h- refinements, this code was chosen as the starting point to study and develop p-multigrid methods in the present work. Afterwards, the three-dimensional implementation of the code was developed by the first author. In the HOMA solver, the discretization is naturally inspired by the FUNSAFE framework. Also, the implicit line preconditioning and multilevel solution techniques are inspired by the NSU3D flow solver [15-17] and other multigrid discontinuous Galerkin solvers [18, 19] developed in the CFD lab of the University of Wyoming.

III. Governing Equations

The governing equations consist of the compressible Reynolds Averaged Navier-Stokes (RANS) equations coupled with the negative version of the one equation Spalart-Allmaras (SA) turbulence model [20]. In the conservative form, these equations can be written as

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} = \mathbf{S} \tag{1}$$

Here, the bold letters denote vector variables due to multiple equations, and i indexes the spatial dimension. Also, as seen in the following, $\overrightarrow{()}$ denotes a vector in n_{sd} spatial dimensions. The vector of the conservative

flow variables $\mathbf{Q}$, the source term $\mathbf{S}$, and the flux vector $\mathbf{F}_i$ which consists of inviscid and viscous parts, $\mathbf{F}_i^E$ and $\mathbf{F}_i^v$, are given by

$$\mathbf{F}_i = \mathbf{F}_i^E - \mathbf{F}_i^v \tag{2}$$

$$\mathbf{Q} = \begin{Bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \\ \rho\tilde{\nu} \end{Bmatrix}, \mathbf{F}_i^E = \begin{Bmatrix} \rho u_i \\ \rho u_1 u_i + \delta_{1i} p \\ \rho u_2 u_i + \delta_{2i} p \\ \rho u_3 u_i + \delta_{3i} p \\ \rho H u_i \\ \rho\tilde{\nu} u_i \end{Bmatrix}, \mathbf{F}_i^v = \begin{Bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij} u_j - q_i \\ \frac{1}{\sigma}\mu(1+f_n\chi)\frac{\partial\tilde{\nu}}{\partial x_i} \end{Bmatrix}, \mathbf{S} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ S_T \end{Bmatrix} \quad i = 1,2,3 \tag{3}$$

where ρ is the density, p is the static pressure, u_i is the velocity component in the direction of the Cartesian coordinate x_i, E is the specific total energy, $H = E + \frac{p}{\rho}$ is the specific total enthalpy, and δ_{ij} is the Kronecker delta. With the assumption of a perfect gas, the pressure is related to the state variables by the constitutive relation,

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho u_i u_i\right) \tag{4}$$

where γ is the ratio of specific heats and it is set to 1.4. Also, τ is the shear stress tensor and q is the heat flux which are given by

$$\begin{aligned} \tau_{ij} &= (\mu + \mu_T)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) \\ q_i &= -c_p\left(\frac{\mu}{Pr} + \frac{\mu_T}{Pr_T}\right)\frac{\partial T}{\partial x_i} \end{aligned} \tag{5}$$

where $Pr = 0.72$ and $Pr_T = 0.9$ are Prandtl and turbulent Prandtl numbers, respectively. μ is the dynamic viscosity which is obtained by the Sutherland's Law and μ_T is the turbulent eddy viscosity which is given by

$$\mu_T = \begin{cases} \rho\tilde{\nu} f_{v_1} & \tilde{\nu} \geq 0 \\ 0 & \tilde{\nu} < 0 \end{cases}, \quad f_{v_1} = \frac{\chi^3}{\chi^3 + c_{v_1}^3}, \quad \chi \equiv \frac{\tilde{\nu}}{\nu} \tag{6}$$

where ν is the kinematic viscosity and $\tilde{\nu}$ is working variable of the turbulence model. The function f_n in the diffusion term of the turbulence model equation is given by

$$f_n = \begin{cases} 1 & \tilde{\nu} \geq 0 \\ \dfrac{c_{n_1} + \chi^3}{c_{n_1} - \chi^3} & \tilde{\nu} < 0 \end{cases} \tag{7}$$

The source term of the turbulence model equation is given by

$$S_T = \rho(P_n - D_n) + \frac{c_{b_2}\rho}{\sigma}\left(\frac{\partial\tilde{\nu}}{\partial x_i}\frac{\partial\tilde{\nu}}{\partial x_i}\right) - \frac{1}{\sigma}(\nu + \tilde{\nu})\left(\frac{\partial\rho}{\partial x_i}\frac{\partial\tilde{\nu}}{\partial x_i}\right) \tag{8}$$

where

$$P_n = \begin{cases} c_{b_1}(1 - f_{t_2})\tilde{S}\tilde{\nu} & \tilde{\nu} \geq 0 \\ c_{b_1}(1 - c_{t_3})S\tilde{\nu} & \tilde{\nu} < 0 \end{cases} \qquad D_n = \begin{cases} \left(c_{w_1} f_w - \frac{c_{b_1}}{\kappa^2} f_{t_2}\right)\left(\frac{\tilde{\nu}}{d}\right)^2 & \tilde{\nu} \geq 0 \\ -c_{w_1}\left(\frac{\tilde{\nu}}{d}\right)^2 & \tilde{\nu} < 0 \end{cases} \tag{9}$$

$$\tilde{S} = \begin{cases} S + \bar{S} & \bar{S} \geq -c_{v_2} S \\ \dfrac{S + S(c_{v_2}^2 S + c_{v_3} \bar{S})}{(c_{v_3} - 2c_{v_2})S - \bar{S}} & \bar{S} < -c_{v_2} S \end{cases} \tag{10}$$

Here, d is the distance to the wall and S is the magnitude of vorticity

$$S = \sqrt{\omega_i \omega_i} \quad , \quad \boldsymbol{\omega} = \boldsymbol{\nabla} \times \mathbf{u} \tag{11}$$

The remaining functions are

$$\bar{S} = \frac{\tilde{\nu} f_{v_2}}{\kappa^2 d^2} \quad , \quad f_{v_2} = 1 - \frac{\chi}{1 + \chi f_{v_1}} \quad , \quad f_{t_2} = c_{t_3} e^{-c_{t_4} \chi^2} \tag{12}$$

$$f_w = g \left(\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6} \right)^{\frac{1}{6}} \quad , \quad g = r + c_{w_2}(r^6 - r) \quad , \quad r = \min \left(\frac{\tilde{\nu}}{\tilde{S} \kappa^2 d^2}, r_{lim} \right) \tag{13}$$

And finally, the constants are $\sigma = 2/3$, $\kappa = 0.41$, $c_{v_1} = 7.1$, $c_{v_2} = 0.7$, $c_{v_3} = 0.9$, $c_{n_1} = 16.0$, $c_{b_1} = 0.1355$, $c_{b_2} = 0.622$, $c_{w_1} = \frac{c_{b_1}}{\kappa^2} + (1 + c_{b_2})\sigma$, $c_{w_2} = 0.3$, $c_{w_3} = 2.0$, $c_{t_3} = 1.2$, $c_{t_4} = 0.5$, and $r_{lim} = 10.0$.

B. Artificial viscosity

For shock capturing, artificial viscosity is added to the governing equations in order to stabilize the solution near the shock waves. To this end, the viscous flux is augmented as [21]

$$\widehat{\mathbf{F}}_i^v = \mathbf{F}_i^v + \mathbf{F}_i^{av} \tag{14}$$

$$\mathbf{F}_i^{av} = h_{sh} \lambda_{max} \epsilon_{sh} \frac{\partial \widetilde{\mathbf{Q}}}{\partial x_i} \tag{15}$$

where $\widetilde{\mathbf{Q}}$ is a vector of the conservative flow variables which includes ρH instead of ρE. This choice is expected to conserve the total enthalpy across the shock which is required by Rankine-Hugoniot shock jump relations (see Refs. [22, 23]). Also, in above, the maximum eigenvalue (or wave speed) of the system is $\lambda_{max} = \sqrt{u_i u_i} + c$, where c is the speed of sound. h_{sh} is an element length for shock capturing which is calculated at each quadrature point as

$$h = \left[\sum_{j=1}^{n_s^e} \left(\frac{\partial N_j}{\partial x_i} \frac{\partial N_j}{\partial x_i} \right) \right]^{-1/2}$$

where N_j is the j^{th} shape function in the element and n_s^e is number of shape functions (or number of modes) in the element. Finally, ϵ_{sh} is a shock indicator which is defined as

$$\epsilon_{sh} = \frac{\left(u_i \dfrac{\partial p}{\partial x_i} \right) h_{sh}}{\left(u_i \dfrac{\partial p}{\partial x_i} \right) h_{sh} + \kappa_{sh} c \, p}$$

where κ_{sh} is a tunable parameter based on shock strength that varies from 0.1 for a strong shock and 1.0 for a weak shock. In this study, the artificial viscosity has been used only for the first numerical example (see section VII.A) for which the value of κ_{sh} has been set to 0.1.

C. Spatial Discretization

To start, the *strong form* of the problem is written as an initial boundary value problem:

$$\mathcal{L}(\mathbf{q}) = \mathbf{S} \qquad \vec{x} \in \Omega \text{ and } t \in [0, \infty) \tag{16.a}$$

$$\mathbf{F}_i = \mathbf{F}_i^b \qquad \vec{x} \in \Gamma_F \text{ and } t \in [0, \infty) \tag{22.b}$$

$$\mathbf{q}(\boldsymbol{x}, t) = \mathbf{q}_D \qquad \vec{x} \in \Gamma_D \text{ and } t \in [0, \infty) \tag{22.c}$$

$$\mathbf{q}(\boldsymbol{x}, 0) = \mathbf{q}_0(\boldsymbol{x}) \qquad \vec{x} \in \Omega \tag{22.d}$$

where, $\Omega \subset \mathbb{R}^{n_{sd}}$ is a bounded domain with Lipschitz-continuous boundary Γ, $\mathbf{F}_i^b$ are the prescribed boundary fluxes through Γ_F portion of the boundary, and $\mathbf{q}_D$ is the Dirichlet boundary condition on the Γ_D portion of the boundary, and the operator $\mathcal{L}$ is defined as (see also Refs. [24, 25])

$$\mathcal{L} \coloneqq [\mathbf{A}^q]\frac{\partial}{\partial t} + [\mathbf{A}_i^E]\frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\mathbf{G}_{\text{ij}}\frac{\partial}{\partial x_j}\right) \tag{17}$$

where $[\mathbf{A}^q] = \frac{\partial \mathbf{Q}}{\partial \mathbf{q}}$ is the variable transformation matrix, $[\mathbf{A}_i^E] = \frac{\partial \mathbf{F}_i^E}{\partial \mathbf{q}}$ is the Euler flux Jacobian matrix, and $[\mathbf{G}_{ij}]$ is the diffusivity matrix which is defined such that $\mathbf{F}_i^v = [\mathbf{G}_{ij}]\frac{\partial \mathbf{q}}{\partial x_j}$. Here, $\mathbf{q}$ is the vector of the dependent variables, which may be chosen over the conservative variables $\mathbf{Q}$ to facilitate the implementation. In the present work, it is the vector of state variables $\mathbf{q} = [\rho, u_i, T, \tilde{\nu}]^T$. This choice is based on the need for modeling fluids with nonlinear equations of state that typically provide the pressure and other thermodynamic variables in terms of density and temperature.

For discretization, Ω is approximated by a *computational domain* Ω^h with piecewise-polynomial boundary Γ^h. Then, the *finite element mesh* $\mathcal{T}^h = \{\Omega^1, \Omega^2, ..., \Omega^{n_{el}}\}$ is defined as the division of Ω^h into a finite number of non-overlapping elements such that $\Omega^h = \bigcup_{e=1}^{n_{el}} \Omega^e$. Accordingly, the boundary is partitioned as $\Gamma^h = \bigcup_{e=1}^{n_{el}} \Gamma^e \cap \Gamma$. Next, each element e is equipped with a polynomial order $1 \leq P(\Omega^e) = P^e$. At this point, spatial approximation spaces can be precisely defined as

$$\boldsymbol{\mathcal{S}}_t^h \coloneqq \left\{\mathbf{q} | \mathbf{q}(\cdot, t) \in [\mathcal{H}^1(\Omega^h)]^{n_Q}, \mathbf{q}(\cdot, t)|_{\Omega^e} \in [\mathcal{P}_{P^e}(\Omega^e)]^{n_Q}, t \in [0, \infty)\ \forall e \text{ and } \mathbf{q}(\cdot, t) = \mathbf{q}_D^h \text{ on } \Gamma_D^h\right\} \tag{18}$$

$$\mathcal{W}^h \coloneqq \left\{w |\ w \in \mathcal{H}^1(\Omega^h);\ w|_{\Omega^e} \in \mathcal{P}_{P^e}(\Omega^e)\ \forall e \text{ and } w = 0 \text{ on } \Gamma_D^h\right\} \tag{19}$$

where $\mathcal{H}^1$ is the usual Sobolev space of weakly differentiable functions, $[\mathcal{H}^1]^{n_Q}$ is the corresponding space for vector functions with n_Q components, and $\mathcal{P}_P$ is the polynomial space, complete to the order P. Now, the discrete solution to the weak form of the problem can be expressed as: for any $t \in [0, \infty)$ find $\mathbf{q}^h \in \boldsymbol{\mathcal{S}}_t^h$ such that for all $w^h \in \mathcal{W}^h$,

$$\iint_{\Omega^h} \left(w^h [\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t} - \frac{\partial w^h}{\partial x_i}\mathbf{F}_i - w^h \mathbf{S}\right) d\Omega + \int_{\Gamma_F^h} w^h (\mathbf{F}_i^b n_i)\ d\Gamma = \mathbf{0} \tag{20}$$

where the superscript h denotes discretized variables and n_i are the components of the unit outward-normal on Γ. The discrete solution $\mathbf{q}^h$ is expanded as:

$$\mathbf{q}^h = \sum_{i=1}^{n_{DOF}} \mathbf{q}_i N_i \quad \text{on } \Omega^h \tag{21}$$

where $\mathbf{q}_i$'s are the solution's coefficients or the Degrees of Freedom (DOFs), N_i's are the basis functions for the finite-dimension space $\boldsymbol{\mathcal{S}}_t^h$, and n_{DOF} is the dimension of that space as well as the number of DOFs. If

the weight function w^h is constructed using the same class as the solution basis functions N, the original Bubnov-Galerkin discretization is derived. It is well known that in situations where advection fluxes dominate diffusion fluxes, the original Galerkin method will suffer from spurious oscillations that lead to instability of the method. In present work, the Streamline-Upwind Petrov-Galerkin (SUPG) [26-28] scheme has been used for stabilization. In this scheme, a stabilization term is added to the Galerkin discretization as

$$\underbrace{\iint_{\Omega^h}\left(N[\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t}-\frac{\partial N}{\partial x_i}\mathbf{F}_i-N\mathbf{S}\;d\Omega\right)d\Omega+\int_{\Gamma_F^h}N(\mathbf{F}_i^b n_i)\;d\Gamma}_{Galerkin\ Discretization}+\underbrace{\sum_{e=1}^{n_{el}}\iint_{\Omega^e}[\mathbf{P}^e]\left([\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t}+\frac{\partial \mathbf{F}_i}{\partial x_i}-\mathbf{S}\right)d\Omega}_{Stabilization\ Term}=\mathbf{0} \quad (22)$$

In above equation, N has been used instead of w to emphasize that hereafter the weight functions are chosen from the same class as the solution basis functions. The stabilization term is calculated over all the elements in the computational domain. The value in the parentheses of this term is the residual of the original PDE and this is why these methods are known as residual-based stabilization methods [29]. The term $[\mathbf{P}]$ is called the Perturbation to the test function space as it modifies the original Galerkin method to a Petrov-Galerkin method with $N[\boldsymbol{I}]+[\boldsymbol{P}]$ as the weight function [27]. For the SUPG method,

$$[\mathbf{P}^e]=\left[\frac{\partial \mathbf{F}_i^E}{\partial \mathbf{Q}}\right]\frac{\partial N}{\partial x_i}[\boldsymbol{\tau}^e] \quad (23)$$

where $[\boldsymbol{\tau}]$ is called the stabilization matrix. It has the dimension of time and it can be obtained based on the eigensystem decomposition of the projection of the flux Jacobian matrices onto the spatial gradients of the basis functions. However, the stabilization may also be derived from flux-vector splitting formulations. Advantages of such an approach are that differentiability, positivity, and total enthalpy conservation can be maintained [30-32]. In the present study, only the stabilization based on eigensystem decomposition has been used [33] with viscous scaling similar to that described in Ref. [34]. This term for element e is given by

$$[\boldsymbol{\tau}^e]^{-1}=\sum_{j=1}^{n_s^e}\left|\frac{\partial N_j^e}{\partial x_i}\left[\frac{\partial \mathbf{F}_i^E}{\partial \mathbf{Q}}\right]\right|+\frac{\partial N_j^e}{\partial x_i}[\mathbf{G}_{ik}]\frac{\partial N_j^e}{\partial x_k} \quad (24)$$

where N_j^e and n_s^e are the *shape functions* and number of modes within the element e, respectively. Here, a shape function within an element is considered as the restriction of a basis function to that element (see Ref. [14]). In the above equation,

$$\left|\frac{\partial N_j^e}{\partial x_i}\left[\frac{\partial \mathbf{F}_i^E}{\partial \mathbf{Q}}\right]\right|=[\mathbf{T}][\boldsymbol{\Lambda}][\mathbf{T}]^{-1} \quad (25)$$

where $[\mathbf{T}]$ and $[\boldsymbol{\Lambda}]$ denote the matrix of right eigenvectors and the diagonal matrix of absolute values of the eigenvalues of the left hand side of the above equation, respectively.

D. Initial and Boundary Conditions

In this study, initial conditions are set to free-stream conditions with the exception of the no-slip walls, where the no-slip condition is applied. The non-dimensional value of the free-stream turbulence working variable $\tilde{\nu}$ is set to 3 for fully turbulent flows [35].

Regarding boundary conditions, far-field, inviscid wall, and no-slip wall boundaries are considered. The walls are assumed to be adiabatic. For the far-field boundaries, the boundary flux vector $\mathbf{F}_i^b$ only includes the inviscid part and is constructed using the Roe scheme [36] based on the free-stream and interior state values. For inviscid walls, the boundary flux vector only takes the pressure from interior and thus $\mathbf{F}_i^b = [0, \delta_{1i}p, \delta_{2i}p, \delta_{3i}, 0]^T$.

1. Weakly Imposed Dirichlet Boundary conditions

To apply no-slip condition on the walls, the discrete weak form in Eq. (21) may be augmented with weak implementation of Dirichlet boundary conditions [37]. Bazilevs and Hughes proposed such an approach in Ref. [38] for the advection diffusion and incompressible Navier–Stokes and then they extended their method in Ref. [39]. It should be mentioned that this method is essentially based on Nitsche's method [40]. Here the same idea is followed, although the utilized formulation is based on a DG discretization for compressible flows. This formulation is obtained by ignoring interior stabilization from the Symmetric Interior Penalty Galerkin (SIPG) method [10, 41, 42]. To impose weak boundary conditions, Eq. (21) is augmented as following,

$$\begin{aligned}&\iint_{\Omega^h}\left(N[\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t}-\frac{\partial N}{\partial x_i}\mathbf{F}_i-N\mathbf{S}\;d\Omega\right)d\Omega+\int_{\Gamma_F^h}N(\mathbf{F}_i^b n_i)\;d\Gamma\\&+\sum_{e=1}^{n_{el}}\iint_{\Omega^e}[\mathbf{P}^e]\left([\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t}+\frac{\partial \mathbf{F}_i}{\partial x_i}-\mathbf{S}\right)d\Omega+\mathcal{N}_{\Gamma_{\mathrm{D}}}=\mathbf{0}\end{aligned}\tag{26}$$

where

$$\begin{aligned}\mathcal{N}_{\Gamma_{\mathrm{D}}}=&\int_{\Gamma\cap\Gamma_D^h}N\left(\mathbf{F}_i^E(\mathbf{q}^b)-\mathbf{F}_i^v(\mathbf{q}^b,\nabla\mathbf{q}^h)\right)n_i\;d\Gamma\\&-\int_{\Gamma\cap\Gamma_D^h}\left([\mathbf{G}_{i1}(\mathbf{q}^b)]\frac{\partial N}{\partial x_i},[\mathbf{G}_{i2}(\mathbf{q}^b)]\frac{\partial N}{\partial x_i},[\mathbf{G}_{i3}(\mathbf{q}^b)]\frac{\partial N}{\partial x_i}\right)\cdot(\mathbf{q}^h-\mathbf{q}^b)\vec{n}\;d\Gamma\\&+\int_{\Gamma\cap\Gamma_D^h}\eta[\mathbf{G}(\mathbf{q}^b)](\mathbf{q}^h-\mathbf{q}^b)\vec{n}\cdot N\vec{n}\;d\Gamma\end{aligned}\tag{27}$$

where $\mathbf{q}^b$ is a state vector which is constructed based on the Dirichlet boundary conditions and interior solution. Adiabatic and no-slip conditions yield $\mathbf{q}^b=[\rho,0,0,0,T,0]^T$ and thus the component of the boundary viscous fluxes $\mathbf{F}_i^v$ associated with the energy equation vanishes. Note that to calculate $\mathbf{q}^b$, ρ and T are calculated based on the interior solution on the element adjacent to the boundary. Also, the turbulence working variable $\tilde{\nu}$ is set to be zero at no-slip walls. The last term in Eq. (27) is referred to as a penalty term, in which the penalty parameter η is given by

$$\eta=n_s^e\,\frac{|\Gamma^e|}{|\Omega^e|}\tag{28}$$

where $|\Gamma^e|$ and $|\Omega^e|$ are the surface and volume of the element e which is adjacent to the boundary.

V. Principal Solution Strategies

As mentioned in the introduction, the objective of this work is the investigation and development of scalable multilevel solution strategies and preconditioning techniques for high-order PG schemes. The focus is put on steady-state problems. In this section, the different elements of the proposed multilevel solution strategy are briefly reviewed. The multilevel solution strategy can be viewed as a p-multigrid method in which the solution on the P1 mesh ((p=1), coarsest level) is solved to the desired accuracy using a preconditioned Newton-Krylov method. Note that the solver on the coarsest level performs as an exact coarse grid corrector (CGC) for the finer levels. Therefore, the performance and scalability of the whole multilevel strategy greatly depends on the performance and scalability of the solver on the coarsest level.

To drive the solution on the P1 mesh toward the steady state, a pseudo-transient continuation (PTC) method is utilized which is explained in the next sub-section. Next, different implicit line relaxation and algebraic multigrid (AMG) methods used for preconditioning of the linear systems on the P1 mesh are discussed. Finally, the strategy for p-multigrid method is described.

A. Pseudo-Transient Continuation (PTC)

By applying a Petrov-Galerkin (PG) discretization on the spatial derivatives, the equations can be written in the following semi-discrete form:

$$[\mathbf{M}^h]\frac{\partial \mathbf{q}^h}{\partial t} + \mathbf{R}^h(\mathbf{q}^h) = \mathbf{0} \tag{29}$$

where $\mathbf{R}^h$ corresponds to the discretized spatial residual, and $[\mathbf{M}^h]$ denotes the mass matrix. Applying backward difference formula (BDF) on Eq. (27) results in the following implicit formulation:

$$\mathbf{Res}^{h,n+1}(\mathbf{q}^{h,n+1}) = \frac{[\mathbf{M}^h]}{\Delta t}(\mathbf{q}^{h,n+1} - \mathbf{q}^{h,n}) + \mathbf{R}^h(\mathbf{q}^{h,n+1}) = 0 \tag{30}$$

where $\mathbf{Res}^{h,n+1}$ represents the unsteady flow residual at time step $n+1$. The implicit system is linearized using an automatic differentiation implementation and the vector of the conservative variables is updated in a Newton-iteration algorithm similar to that in Refs. [43, 44]. The linearized system and the update equation are given by

$$[\mathbf{J}^{h,n}(\mathbf{q}^{h,n})]\Delta\mathbf{q}^{h,n} = -\mathbf{R}^{h,n}(\mathbf{q}^{h,n}) \tag{31}$$

$$\mathbf{q}^{h,n+1} = \mathbf{q}^{h,n} + w_{opt}\Delta\mathbf{q}^{h,n} \tag{32}$$

where $[\mathbf{J}^{h,n}]$ denotes the Jacobian matrix. The w_{opt} is a nominal optimum relaxation factor which is determined in a line search process. In order to accelerate the global convergence, the utilized Newton algorithm is modified to include local time-steps which are amplified by a CFL number. Thus,

$$[\mathbf{J}^h] = \left[\frac{[\mathbf{M}^h]}{\text{CFL }\Delta t} + \frac{\partial \mathbf{R}^h}{\partial \mathbf{Q}^h}\right] \tag{33}$$

At small CFL numbers, the algorithm essentially becomes an explicit method, whereas at high CFL numbers, the algorithm approaches Newton's method. To enhance the robustness, a limiting relaxation factor $w_{\rho,T}$ is determined such that the relative changes in density and temperature are limited by maximum value which is called θ. Next, $w_{\rho,T}$ serves as the maximum relaxation factor during the line search that is used to determine the optimal relaxation value w_{opt}. In this process, the RMS of the unsteady residual $\mathbf{Res}^{n+1}$ is evaluated at four relaxation factors: 0, w_{min},$(w_{min} + w_{\rho,T})/2$, and $w_{\rho,T}$. The optimal value is found by locating the minimum of a fitted cubic polynomial within the range of [0, $w_{\rho,T}$]. If a full step, as characterized by a relaxation factor of $w_{opt} = 1.0$, is taken the CFL is amplified by the factor β_{CFL}. If the optimum relaxation factor falls below the minimal value w_{min}, the step is rejected and the CFL number is divided by $5\beta_{CFL}$. In other cases, the CFL remains at the previous value. For the results presented in this paper, $w_{min} = 0.1$, for subsonic flows: $\theta = 0.2$, $\beta_{CFL} = 2.0$, and for supersonic flows: $\theta = 0.4$, $\beta_{CFL} = 1.5$.

The linear system in Eq. (31) is solved using GMRES [45]. The utilized preconditioning techniques are descried in section VI.

B. Non-linear P-Multigrid using Full Approximation Scheme (FAS-PMG)

To solve high-order ($P \geq 2$) problems, a p-multigrid (PMG) algorithm is applied to the nonlinear system of equations, using the full approximation scheme (FAS). In this approach, the coarser levels are formed by reducing the polynomial order of the function space while the number of elements in the computational mesh remains constant. The problem on the coarsest level ($P = 1$) is solved to the desired accuracy using the PTC algorithm. A key element in the p-multigrid approach is to employ a set of hierarchical basis functions with

a modal method. The notion of hierarchy comes from the fact that higher-order polynomials are incrementally built from the lower-order ones. In other words, to obtain higher order expansions, new modes, which can be viewed as corrections, are added to the old expansion. As described in Refs. [18, 19], the use of hierarchical basis functions simplifies the inter-grid restriction and prolongation operators significantly. In addition, the basis functions associated with high-order modes ($P \geq 2$) can be freely scaled. This feature can be exploited to condition the linear system of equations and devise more robust smoothers.

Algorithms 1 and 2 describe the details of the developed non-linear PMG algorithm. In this algorithm, l indexes the solution level, n_l is the number of solution levels which is set to the maximum polynomial order P, and n_{MGCyc} is the number of multigrid cycles. *NSmoother* denotes a non-linear smoother (see Algorithm 4) which is formed based on the linearized system in Eq. (4). Here, $\mathbf{n}_R = \{n_R^l\}$ is the number of non-linear residual updates, and $\mathbf{f}_J = \{f_J^l\}$ is the frequency of the Jacobian-matrix updates. In Algorithm 4, *LSmoother* denotes a linear smoother. For inviscid flows this smoother is point Jacobi and for the laminar and turbulent flows, it is pre-conditioned implicit line Jacobi (PILJ) which is defined in section VI.B.

Algorithm 1. Non-Linear P-Multigrid (V-Cycle): $\mathbf{q}^{n_l} = PMG(n_l, n_{MGCyc}, \mathbf{n}_R, \mathbf{f}_J, \mathbf{n}_o, \mathbf{n}_i, \boldsymbol{\omega}, \mathbf{q}^{n_l,0})$

0: # This algorithm uses Full Approximation Scheme (FAS) for non-linear multigrid iterations #
1: for $k = 1, n_{MGCyc}$
2: # Pre-smoothing #
3: for $l = n_l, 2, -1$ # Descending counter #
4: $\mathbf{q}^l = NSmoother(n_R^l, f_J^l, n_o^l, n_i^l, \omega^l, \mathbf{q}^{l,0}, \mathbf{S}^l)$ # See Algorithm 2 #
5: $\mathbf{q}^{l-1,0} = I_l^{l-1}\mathbf{q}^l$
6: $\mathbf{S}^{l-1} = \mathbf{R}^{l-1}(I_l^{l-1}\mathbf{q}^l) + \hat{I}_l^{l-1}\mathbf{R}^l(\mathbf{q}^l)$
7: end for
8: # Solve the coarsest level #
9: $\mathbf{q}^1 = Solve(\mathbf{q}^{1,0}, \mathbf{S}^1)$ # Using PTC #
10: # Post-smoothing #
11: for $l = 2, n_l$
12: $\mathbf{q}^{l,0} = \mathbf{q}^l + I_{l-1}^l(\mathbf{q}^{l-1} - \mathbf{q}^{l-1,0})$
13: $\mathbf{q}^l = NSmoother(n_R^l, f_J^l, n_o^l, n_i^l, \omega^l, \mathbf{q}^{l,0}, \mathbf{S}^l)$ # See Algorithm 4 #
14: end for
15: end for
16: return $\mathbf{q}^{n_l}$

Algorithm 2. Non-Linear Smoother: $\mathbf{q}^{n_R} = \boldsymbol{NSmoother}(n_R, f_J, n_o, n_i, \omega, \mathbf{q}^0, \mathbf{S})$

1: for $k = 0, n_R - 1$
2: if $(k\%f_J == 0)$ $[\mathbf{J}] = [\mathbf{J}(\mathbf{q}^k)], [\mathbf{P}] = [\mathbf{P}(\mathbf{q}^k)]$
3: $\hat{\mathbf{R}}^k = -(\mathbf{R}(\mathbf{q}^k) - \mathbf{S})$
4: $d\mathbf{Q} = LSmoother\big(n_o, n_i, \omega, [\mathbf{J}], [\mathbf{P}], \hat{\mathbf{R}}^k\big)$ # $LSmoother = \begin{cases} Point\ Jocobi \text{ for inviscid flows} \\ PILJ \text{ for laminar and turbulent flows} \end{cases}$ #
5: $\mathbf{q}^{k+1} = \mathbf{q}^k + d\mathbf{q}$ # For *PILJ* see Algorithms 3 and 4 #
6: end for
7: return $\mathbf{q}^{n_R}$

VI. Preconditioners and Smoothers

Preconditioners and smoothers are some of the building blocks of the PTC and PMG algorithms. This section describes the details of the several techniques which have been developed in the present work for this purpose.

A. Incomplete Lower Upper (ILU) Factorization

The first preconditioner considered is Incomplete Lower Upper factorization with k level of fills, which is also known as ILU(k) [40]. In order to reduce the number of fill-ins in this approach, the DOFs are reordered using Reverse Cuthill–McKee (RCM) algorithm. As mentioned in the introduction, in the context of domain-decomposition, most often incomplete factorization methods are applied locally within each partition. As a result, the coupling between partitions are lost and the convergence rates are decreased with increasing levels of parallelization. Therefore, the robustness offered by these methods on smaller problems (or fewer partition numbers) may not be feasibly reachable on larger problems (or higher partition numbers). In this paper, we show that implicit line smoothers can be reliable substitutes for incomplete factorization methods.

B. Implicit-Line Smoother

The key to enhance the robustness of the preconditioner is to solve for strongly coupled unknowns in a coupled form. If the strong couplings occur in detectible geometrical directions, the unknowns can be grouped in lines, ordered sequentially, and solved for implicitly using a tridiagonal solver. In diffusion-dominated regions, where anisotropic meshes are used (e.g. boundary layers), the strong couplings follow the clustering directions and thus the construction of the lines is straightforward. On the other hand, in convection-dominated regions, the strong couplings happen in characteristic directions which cannot be intuitively recognized before starting the solution process. The implicit-line smoother presented in this paper is a generalization of an approach initially developed by Mavriplis [16]. The construction of the lines in the original approach is purely mesh based. In anisotropic regions, the lines are constructed in the clustering directions and the unknowns on the lines are solved implicitly. In isotropic regions, the length of the lines reduces to zero and the solution algorithm converts to a point explicit method. Okusanya et al. [3] extended Mavriplis' approach by introducing a matrix-based system to measure nodal coupling used in forming the implicit lines. In particular, they used a matrix derived from a convection-diffusion discretization to form the implicit lines in both convention- and diffusion-dominated regions and utilized their method in two-dimensional Euler and Navier–Stokes solutions. A difficulty in this method is that the lines in the convection-dominated areas cannot be anticipated a priori, and thus, to get the most benefit from generality of the method, the line construction procedures need to be recalled as the solution evolves. When it comes to domain decomposition for distributed-memory parallel implementations, this difficulty is more elaborate. The following subsection clarifies this subject. In this paper, we also utilize a matrix-based approach. However, to construct the lines at the preprocessing step, we use a matrix which is derived from the Laplace operator (see also Ref. [46]).

1. Parallel Implementation

Traditionally, graph partitioning techniques have been used to partition the unstructured finite-element grids. In these techniques, the DOFs (nodal or modal) and their interconnections are modeled by a graph. Then, the graph is partitioned such that the computational load is equally distributed among the partitions and the entailed intercommunications are minimized. The use of implicit-line solvers demands special attention in the grid partitioning. Note that the classical tridiagonal line solver is an inherently sequential operation and thus, if a line is divided among multiple partitions, a running processor should either become idle while the off-portions of the line is being solved by the other processors, or treat the broken line as a separate line.

The first approach is computationally inefficient and the second one can drastically reduce the effectiveness of the numerical algorithm. To overcome this issue, Mavriplis [16] developed a domain decomposition technique in which each line is completely contained in one partition. In this technique, the original unweighted graph is contracted along the implicit lines to produce a weighted graph. To be more specific, in the original graph, all vertices and edges are assigned by unity weight. Then in the contraction process, all of the vertices that are associated with a line are merged into a new vertex. Connections between merged vertices also produce merged edges and the weights associated with the merged vertices and edges are taken as the sum of the weights of the constituent vertices or edges. The contracted weighted graph is then partitioned using a weighted graph partitioner such as METIS [47] or CHACO [48]. To obtain the partitioning for the original unweighted graph, the partitioned weighted graph is de-contracted, meaning that all constituent vertices of a merged vertex are assigned the partition number of that vertex. Since the implicit lines reduce to a single vertex in the contracted graph, they can never be broken by the partitioning process. The weighting assigned to the contracted graph ensures load balancing and communication optimization of the final uncontracted graph. In this study, the same technique is utilized to partition the computational grid while the implicit lines are extracted from the matrix-based approach. Since the utilized matrix is based on the Laplace operator, the configuration of the implicit lines and consequently the partitioning do not change during the solution process. It should be noted that if line configurations change during the solution process, in large-scale simulations, an efficient parallel mesh partitioner should be recalled for each reconfiguration. Although this approach is practical, it requires substantial code developments that complicate the solution algorithms. For the sake of simplicity, this approach has been reserved for the future work.

2. Dual CFL Strategy

An important consideration in the application of an implicit-line smoother is the diagonal dominancy of the utilized matrix. As mentioned in section V.A, during the PTC algorithm, the CFL number is increased aggressively and thus, the implicit line smoother may become unstable and ineffective. To overcome this problem and to improve the robustness, here, the line smoother itself is preconditioned with a secondary Jacobian matrix [**P**] which is calculated based on a capped CFL number (referred to as CFLCap). We refer to this approach as dual CFL strategy or Preconditioned Implicit Line Jacobi (PILJ) method. Algorithms 3 and 4 describe the details of this approach. Note that in algorithm 3, in an outer loop (with n_o sweeps) the residual of the system ([**A**]**x**=**b**) is calculated based on the original matrix [**A**] and then in an inner loop (with n_i sweeps), the updates $d\mathbf{x}$ are calculated based on the pre-conditioner matrix [**P**]. The relaxation factor ω is used only in the inner loop. The pre-conditioner matrix [**P**] is calculated as

$$\begin{cases} \text{if CFL} \le \text{CFLCap} & [\mathbf{P}] = [\mathbf{A}] \\ \text{if CFL} > \text{CFLCap} & [\mathbf{P}] = [\mathbf{A}_{\text{CFLCap}}] \ \text{ or } \ [\mathbf{P}] = [\mathbf{A}] + \dfrac{\left[\widehat{\mathbf{M}}\right]}{\text{CFLCap } \Delta t} \end{cases} \tag{34}$$

where $[\mathbf{A}_{\text{CFLCap}}]$ is a Jacobian matrix evaluated at the capped CFL number and $\left[\widehat{\mathbf{M}}\right]$ is a lumped mass matrix. The lumped mass matrix is a block-diagonal matrix in which each row is the absolute row sum of the original mass matrix. Self-evidently, the second approach (using $\left[\widehat{\mathbf{M}}\right]$) is computationally more efficient and besides, it can avoid the storage of the second Jacobian matrix which is required by the first approach (using $[\mathbf{A}_{\text{CFLCap}}]$). Finally, it should be noted that an example of use of dual CFL numbers in the context of the spectral-element time discretization can be found in a study by Mundis [49].

Algorithm 3. Pre-conditioned implicit line Jacobi: $\mathbf{x}^{n_o} = PILJ(n_o, n_i, \omega, [\mathbf{A}], [\mathbf{P}], \mathbf{b}, \mathbf{x}^0)$

0: # This algorithm solves $[\mathbf{A}]\ \mathbf{x} = \mathbf{b}$. Here, $[\mathbf{P}]$ is a preconditioner matrix #
1: for $k = 1, n_o$
2: $\quad \mathbf{r}^{k-1} = \mathbf{b} - [\mathbf{A}]\mathbf{x}^{k-1}$
3: $\quad d\mathbf{x}^k = ILJ(n_i, \omega, [\mathbf{P}], \mathbf{r}^{k-1}, \mathbf{0})$ # See Algorithm 4 #
4: $\quad \mathbf{x}^k = \mathbf{x}^{k-1} + d\mathbf{x}^k$
5: end for
6: return $\mathbf{x}^{n_o}$

Algorithm 4. Implicit line Jacobi: $\mathbf{x}^{n_i} = ILJ(n_i, \omega, [\mathbf{A}], \mathbf{b}, \mathbf{x}^0)$

0: # This algorithm solves $[\mathbf{A}]\mathbf{x} = \mathbf{b}$ using a defect correction scheme #
1: for $k = 1, n_i$
2: $\quad \mathbf{r}^{k-1} = \mathbf{b} - [\mathbf{A}]\mathbf{x}^{k-1}$ # $[\mathbf{A}] = [\mathbf{T}] + [\mathbf{O}]$, $[\mathbf{T}]$ includes the tridiagonal part of $[\mathbf{A}]$ #
2: $\quad$ *Solve* $[\mathbf{T}]d\mathbf{x}^k = \mathbf{r}^{k-1}$ # Using tridiagonal matrix algorithm - TDMA (Thomas algorithm) #
3: $\quad \mathbf{x}^k = \mathbf{x}^{k-1} + \omega \cdot d\mathbf{x}^k$
4: end for
5: return $\mathbf{x}^{n_i}$

C. Scaling of the High-Order Modes

As mentioned earlier, in the present work, high-order discretizations are built using a hierarchical basis. The particular shape functions used in this study are similar to those presented in Ref. [50]. The traces of these shape functions on the edges of mesh coincide with scaled Lobatto polynomials. Note that the basis functions associated with high-order modes ($P \geq 2$) can be freely scaled. This feature can be exploited to condition the linear system of the equations. In the present work, this idea has been explored heuristically, while an analytical approach has been reserved for the future work. For a pure Galerkin discretization, scaling of modes can be viewed as a concurrent left and right preconditioning:

$$\begin{aligned} [\mathbf{P}][\mathbf{A}][\mathbf{P}]\ \mathbf{y} &= [\mathbf{P}]\mathbf{b} \\ \mathbf{y} &= [\mathbf{P}^{-1}]\mathbf{x} \end{aligned} \tag{35}$$

where $[\mathbf{P}]$ is a diagonal matrix consisting of scaling factors (less than 1.0) of the DOFs, and $\mathbf{y}$ is the vector of scaled solution coefficients. However, it should be noted that in the utilized SUPG formulation, the stabilization matrix is also affected by the scaling of the modes. For the results presented in this paper, only the basis functions associated with P2 modes have been scaled even for P3 solutions.

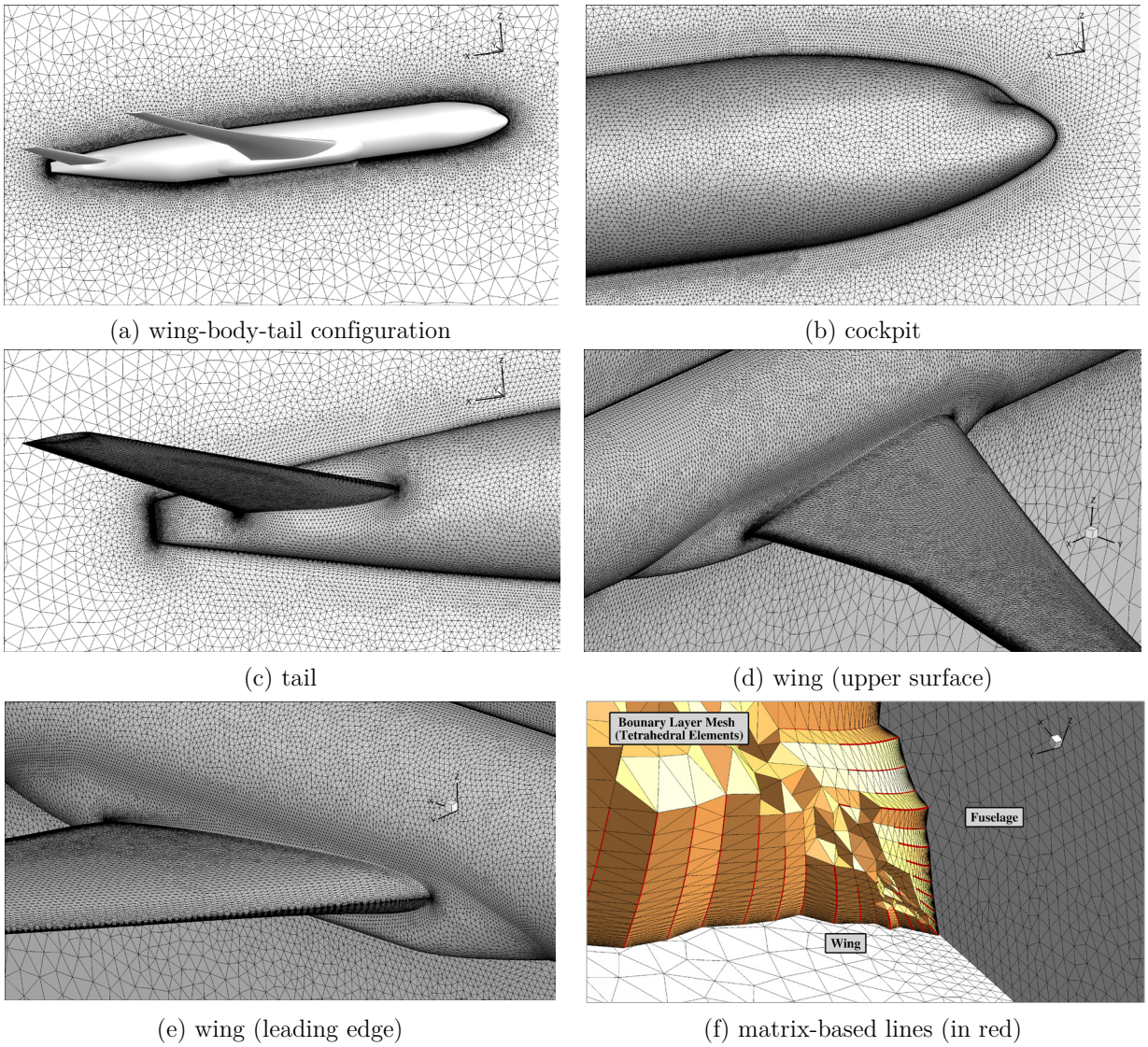
1. Effect of Scaling of High-Order Modes on Matrix-Based Line Generation

As mentioned earlier, in this study, the implicit lines are generated based on strong connections in a stiffness matrix $[\mathbf{A}]$ derived from the Laplace operator. In such a matrix, the entry a_{ij}, which characterizes the influence of node j on node i, is proportional to $\int \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}dx$, where N denotes a basis function. By scaling down the basis functions of the high-order nodes, the corresponding entries in the stiffness matrix are reduced and thus these nodes are not chosen as strong connections. Consequently, when high-order modes are scaled, the implicit line, mainly include unscaled low-order modes. Thus, the smoothing on high-order modes is mainly point explicit.

VII. Numerical Results

A. Instance 1: Transonic Turbulent Flow over DPW-4 Wing-Body-Tail Geometry

The first numerical example compares the performance of the implicit line Jacobi (PILJ) and ILU(k) preconditioners in the PTC algorithm for a three-dimensional case with P1 discretization. To provide a reasonable level of challenge, a transonic turbulent flow over a wing-body-tail configuration, taken from 4th AIAA CFD Drag Prediction Workshop (DPW-4) [51], is considered. In this test case, the Mach number is 0.85, the Reynolds number is 5.0e+6, the tail incidence angle is 0.0 degree, and the angle of attack is 2.0 degrees. The computational mesh, which has been generated using Pointwise software, includes 36,613,818 tetrahedral elements and 6,208,401 vertices. In this mesh, the surface grids have been made using advancing front algorithm, and the boundary layers have been generated using the T-Rex tool. Figure 1 shows different parts of the mesh. Figure 1.f shows a sample of the matrix-based generated lines in the boundary layer. In this figure the lines have been colored in red. Compared to the meshes used in DPW-4, the used mesh here is considered as a coarse mesh.



(a) wing-body-tail configuration

(b) cockpit

(c) tail

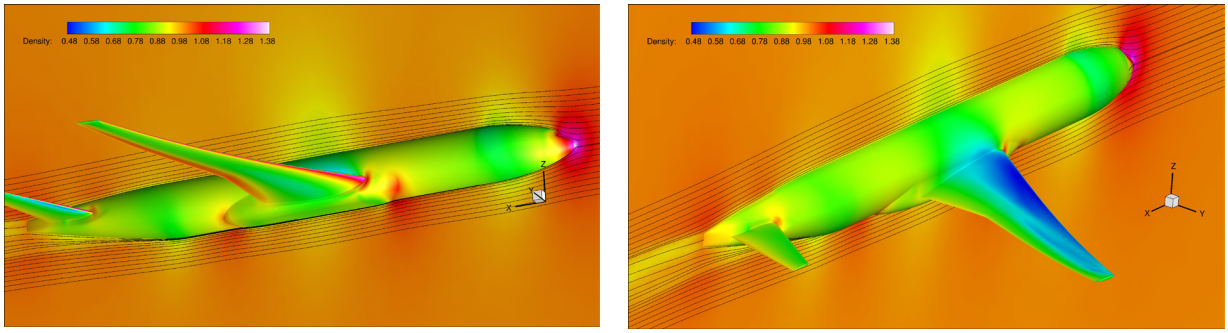
(d) wing (upper surface)

(e) wing (leading edge)

(f) matrix-based lines (in red)

Figure 1. Instance 1, computational mesh for the turbulent flow over DPW-4 geometry.

Figure 2 shows the contours of the density and streamlines solved on this mesh. As seen, the structure of the shock is clearly detectible on the wing. Figure 3 shows the contours of the turbulence working variable in the wake region. As seen, a finer mesh is required to properly resolve the wake region, but the focus of this paper is on the convergence behavior rather than accuracy of the simulation.



(a) (b)

Figure 2. Instance 1, contours of the density and streamlines for the transonic turbulent flow over DPW-4 geometry.

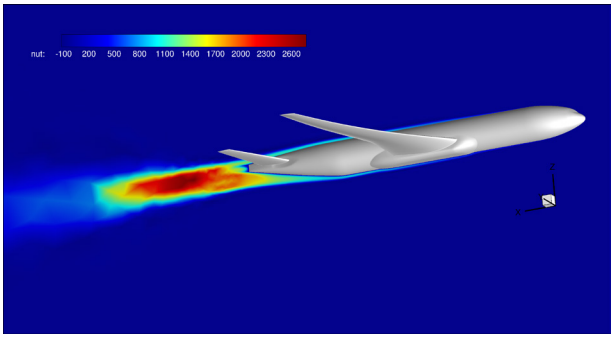


Figure 3. Instance 1, contours of the turbulence working variable for the transonic turbulent flow over DPW-4 geometry.

In order to investigate the scalability and robustness of the mentioned preconditioners in the PTC algorithm, their convergence behavior on four domain decompositions, with 80, 160, 320, and 640 partitions, have been studied. Both preconditioners have been tested on the same partitions made using the contracted graphs described in section VI.B.1. Also, for both preconditioners, at each non-linear step, the convergence tolerance of the linear system has been set to 10^{-4} (for relative residual), and at most 200 Krylov vectors have been allocated for residual minimization. For PILJ, the number of outer sweeps, number of inner sweeps, relaxation factor of the inner sweeps, and CLFCap have been set to 10, 20, 0.2, 1000, respectively. For ILU, 2 to 4 fill levels have been used.

Figures 4 and 5 demonstrate the operation of the PTC algorithm during the non-linear convergence for the studied cases. These figures plot the total steady-state residual (in black), the number of used Krylov vectors (in blue), the CFL number (in red), and the elapsed time (in green). Also, Table 1 summarizes the final number of non-linear steps, the maximum number of used Krylov vectors, and the total elapsed time for the studied cases. As seen in the Fig. 4, a notable advantage of PILJ is that the convergence behavior is independent of the number of partitions. That is for all partition numbers, the plots of total residual, CFL, and number of Krylov vectors are identical. This shows that the preconditioning function has been properly

distributed among processors without affecting the convergence behavior. On the other hand, the performance of the ILU severely depends on the number of partitions. For example, as seen in Figs. 5.b to 5.d, while the PTC algorithm with ILU(3) can solve the problem on 320 partitions, it stalls on 640 partitions, where by increasing the fill level to 4, again it can solve the problem.

Clearly, the ILU preconditioner requires significantly more memory compared to the implicit lines, not only to store the factored matrix, but also to store more Krylov vectors. As seen in Table 1, due to memory limitations, the smaller partition numbers did not allow using ILU. Also, note that for all partitions, PILJ has used at most 36 Krylov vectors to reach the specified tolerance (10^{-4}), whereas ILU has recurrently used the maximum allowable number of Krylov vectors (200). Also in Table 1, it seen that for the partitions which both preconditioners have been used, the implicit lines have resulted in fewer non-linear steps and overall faster convergence. Figure 6 plots the strong scaling for PILJ. It should be noted that this plot has been generated based on the total computation times shown in Table 1. Considering that the performance of an implicit flow solver is studied here, Fig. 6 indicates a good scaling behavior.

In this numerical instance, it was shown that the implicit-line preconditioner can outperform the commonly-used ILU(k) preconditioner in any measure, i.e. robustness, memory requirement, scalability, and speed. Although this technique is not fully matured yet, the current results motivates us to peruse it.

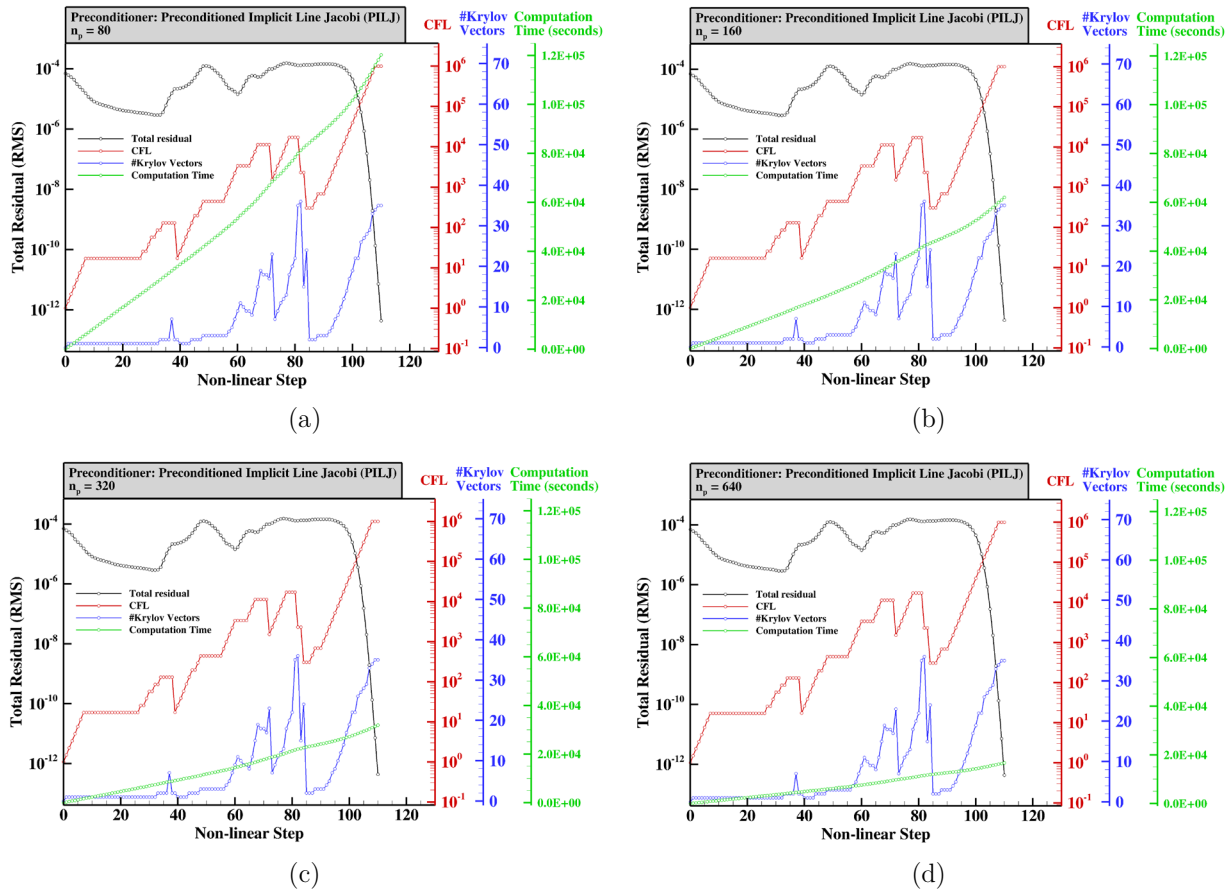


Figure 4. Instance 1, convergence behavior of the PTC algorithm with the implicit-line preconditioner at different partition numbers for the transonic turbulent flow over DPW-4 geometry.

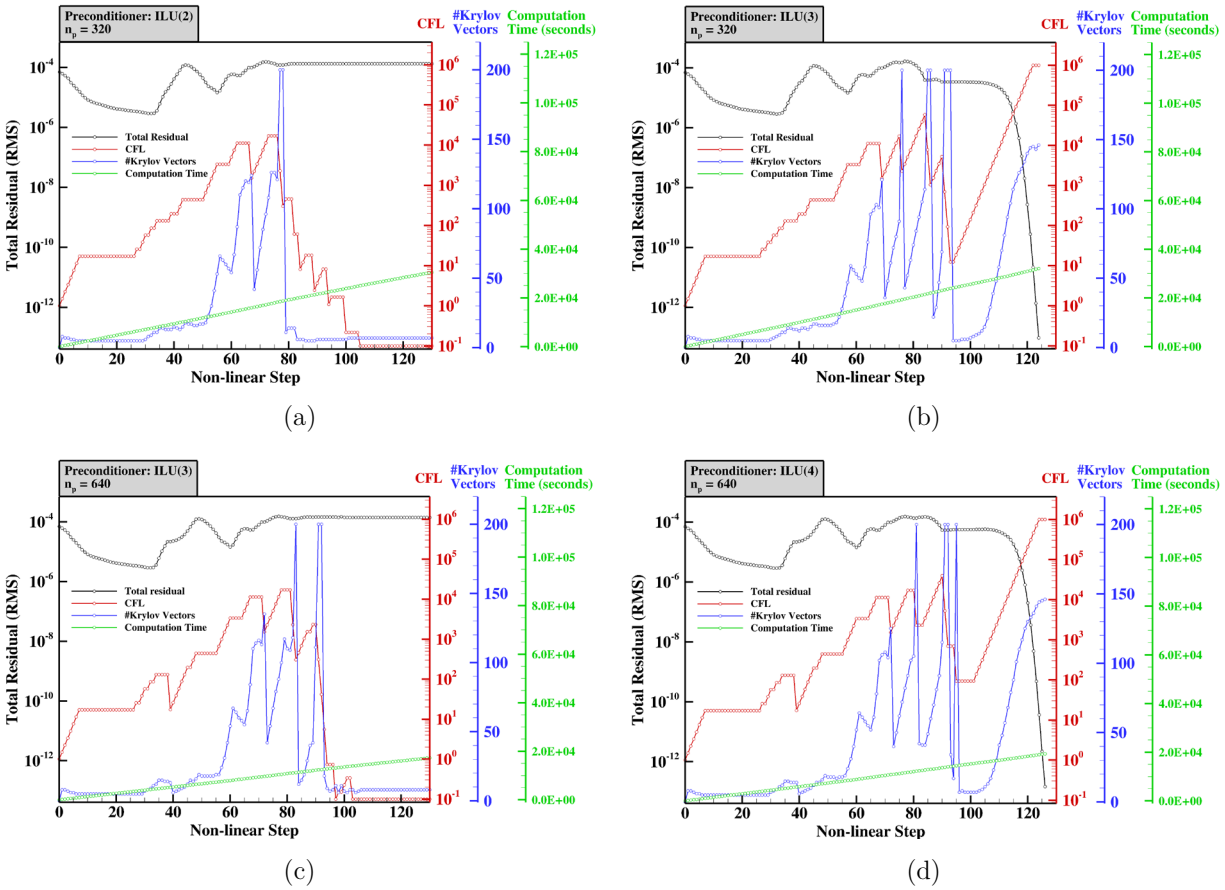


Figure 5. Instance 1, convergence behavior of the PTC algorithm with ILU(k) preconditioner at different partition numbers for the transonic turbulent flow over DPW-4 geometry.

Table 1. Instance 1, number of non-linear steps, maximum number of Krylov vectors, and total computational time for all of the studied cases.

	PILJ			ILU(2)			ILU(3)			ILU(4)		
n_p	n_{NL}	k_{max}	t_{tot}	n_{NL}	k_{max}	t_{tot}	n_{NL}	k_{max}	t_{tot}	n_{NL}	k_{max}	t_{tot}
80	110	36	1.202e+5	NM	NM	NM	NM	NM	NM	NM	NM	NM
160	110	36	6.203e+4	NC	NC	NC	NM	NM	NM	NM	NM	NM
320	110	36	3.188e+4	NC	NC	NC	124	200	3.204e+4	NM	NM	NM
640	110	36	1.671+4	NC	NC	NC	NC	NC	NC	126	200	1.936e+4

n_p = Number of processors (and partitions)

n_{NL} = Number of non-linear steps

k_{max} = Maximum of number of Krylov vector used in the solution

t_{tot} = Total elapsed time in seconds

NM = Not enough memory to run

NC = No convergence

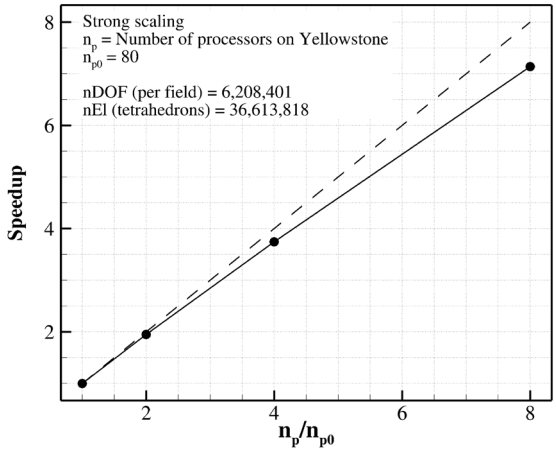


Figure 6. Instance 1, strong scaling of the HOMA solver with the implicit-line preconditioner for the transonic turbulent flow over DPW-4 geometry for a mesh with 6.2 million DOFs per field.

Table 2. Instance 2, PETSc command line options for the preconditioning with BoomerAMG for laminar flow over NACA0012.

Option	Value	Description
ksp_type	fgmres	Sets the Krylov space method
ksp_max_it	200	Maximum number of linear iterations (or Krylov vectors)
ksp_pc_side	right	Sets the preconditioning side
ksp_gmres_modifiedgramschmidt		Uses modified Gram-Schmidt in the orthogonalization (more stable, but slower)
pc_type	hypre	Sets the type of the preconditioner
pc_hypre_type	boomeramg	Sets the type of the hypre preconditioner
pc_hypre_boomeramg_relax_type_all	FCF-jacobi	Sets the relaxation method for BoomerAMG
pc_hypre_boomeramg_max_iter	5	Maximum number of application of BoomerAMG on each Krylov vector
pc_hypre_boomeramg_grid_sweeps_all	2	Number of sweeps for the up and down grid levels
pc_hypre_boomeramg_relax_weight_all	0.3	Relaxation weight for all levels
pc_hypre_boomeramg_outer_relax_weight_all	0.3	Outer relaxation weight for all levels
pc_hypre_boomeramg_tol	1.0e-3	Convergence tolerance per hypre call
pc_hypre_boomeramg_strong_threshold	0.25	Threshold for being strongly connected

B. Instance 2: Subsonic Laminar Flow over NACA0012 Airfoil

The next numerical example compares the performance of ILU(0) and Hyper/BoomerAMG in a laminar flow over NACA0012 airfoil at free stream Mach number of 0.5, Reynolds number of 5000.0, and the angle of attack of 1.0 degree. The simulations have been performed on a series of three successively h-refined meshes. Table 2 shows the command line options used to configure the PETSc toolkit for preconditioning with BoomerAMG. The PETSc toolkit allows to choose a separate matrix for the preconditioning of the Krylov

solver. This feature has been exploited here to implement the dual CFL strategy. For BoomerAMG, the preconditioner matrix has been calculated based on a capped CFL number, which is 200. Figure 7 shows the behavior of both preconditioners during the non-linear convergence. As seen for all meshes, ILU(0) requires more Krylov vectors to reach the convergence tolerance of the linear system, which is 10^{-3}. In particular, for the mesh with two level of h-refinements, after non-linear iteration 14, when the CFL number passes 1000, the linear solver uses the maximum number of Krylov vectors (which is 200) and does not reach the convergence tolerance. Thus, in this test case, BoomerAMG with used settings outperforms ILU(0) in terms of robustness and memory requirements. However, for all meshes, ILU(0) results in lower CPU time for convergence. The cost of the BoomerAMG algorithm can be explained by the slow rates of coarsening achieved by the AMG algorithm, and high levels of matrix fill which occur on the coarser levels. This is described in Table 3, where the statistics for an inviscid flow over NACA0012 test case have been reproduced. This table clearly shows the slow coarsening rates and growth in the average number of non-zero per row in the intermediate level matrices.

Note that the computation presented in this test case have been performed using a single processor. Thus, using an optimized and scalable AMG approach it might be possible to outperform the inherently sequential ILU in parallel. However, alternative coarsening strategies and coarse level operator construction may need to be considered for improved AMG performance.

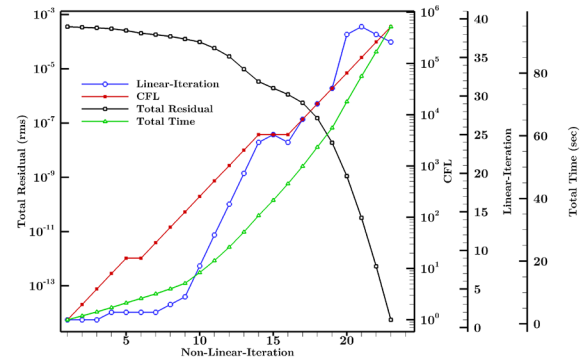
Table 3. Instance 2, Statistics of different grid levels used in the BoomerAMG for an inviscid flow over NACA0012 with 380,256 DOFs.

Level	Rows	Non-zero entries	Sparse	Entries per row			Row sums	
				min	max	avg	min	max
0	1,521,024	42,516,992	0.000	16	40	28	-3.60E-01	6.03E+00
1	714,510	31,690,616	0.000	12	76	44.4	-1.01E+00	8.30E+00
2	391,286	24,276,284	0.000	11	126	62	-1.71E+00	1.31E+01
3	221,241	17,623,887	0.000	7	172	79.7	-2.08E+00	1.14E+01
4	125,468	12,106,562	0.001	17	211	96.5	-2.54E+00	1.22E+01
5	70,356	7,758,122	0.002	22	240	110.3	-1.58E+00	1.37E+01
6	35,498	3,735,578	0.003	22	254	105.2	-3.21E+00	1.50E+01
7	16,937	1,551,843	0.005	17	247	91.6	-1.98E+00	1.91E+01
8	7,966	635,340	0.010	11	184	79.8	-1.17E+00	1.68E+01
9	3,470	227,250	0.019	14	158	65.5	-5.66E-01	1.59E+01
10	1,007	36,681	0.036	4	84	36.4	-2.09E-02	1.02E+01
11	109	1,001	0.084	1	18	9.2	2.02E-03	7.39E+00
12	2	2	0.500	1	1	1	1.42E-02	3.22E-01

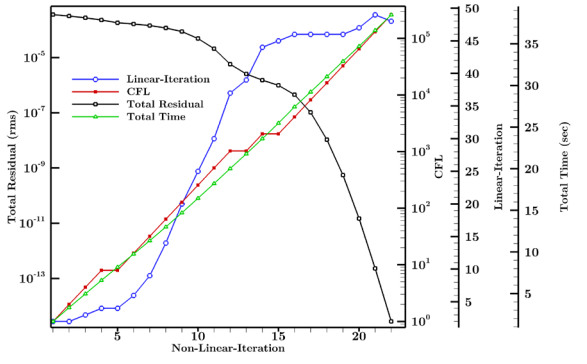
C. Instance 3: Inviscid Flow over NACA0012 Airfoil

This test case is presented to evaluate the performance of the developed FAS-PMG algorithm on Euler equations. For this purpose, an inviscid flow over NACA0012 airfoil at free stream Mach number of 0.5 and the angle-of-attack of 2 degrees is studied. In order to investigate the effect of the discretization order and mesh size on the convergence behavior, high-order simulations (P2 and P3) have been performed on two computational meshes. The second mesh has been generated by uniform refinement of the first mesh. The multigrid iteration on P2 and P3 problems have started from a partially converged P1 solution, in which the non-linear residual has been dropped to 1.0e-6. Table 4 shows the settings used for these simulations. As seen in this table, the modes associated with P2 have been scaled. Figures 8a and 8b show the P2 and P3 multigrid

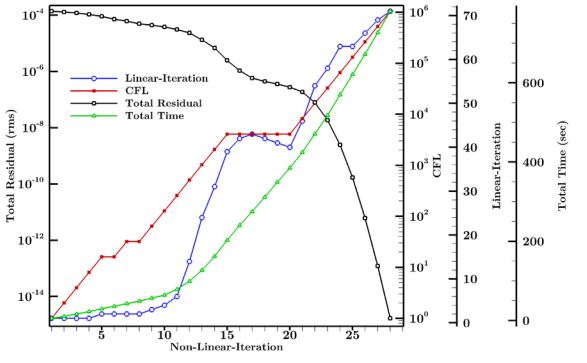
solutions for both meshes. Note that at each multigrid step, the P1 problem is solved exactly using the PTC algorithm. Also, note that for P3, two settings have been used. In the second setting, more residual updates have been used on the P2 level and thus the coarse level solution has been enhanced. In Figs. 8c and 8d, the convergence of the multigrid algorithm has been compared to that of the single grid solver. This comparison has been done by changing the mesh size as well as the discretization order. Based on Figs. 8a to 8d, the developed PMG shows an almost h- and P-independent convergence behavior.



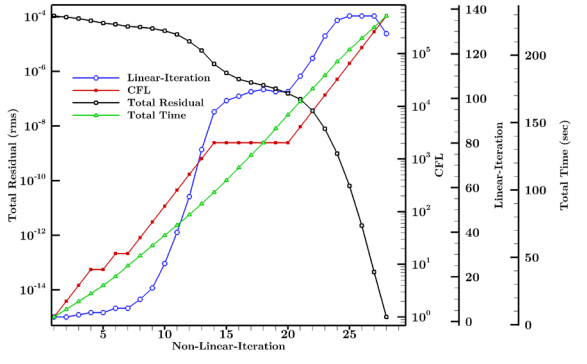
(a) BoomerAMG, Grid $h/2^0$ (4,524 DOFs)



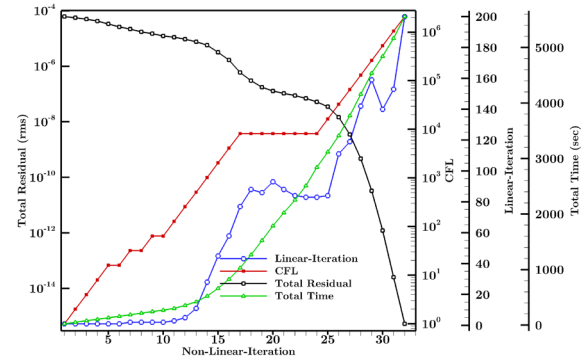
(b) ILU(0), Grid $h/2^0$ (4,524 DOFs)



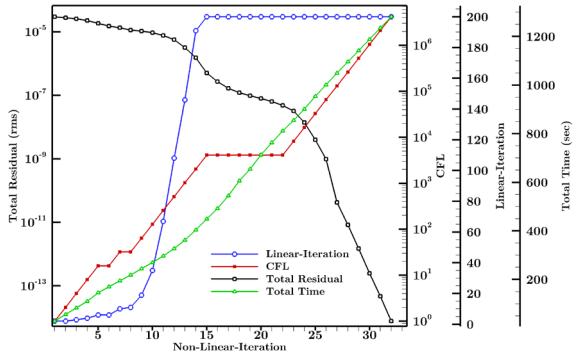
(c) BoomerAMG, Grid $h/2^1$ (18,096 DOFs)



(d) ILU(0), Grid $h/2^1$ (18,096 DOFs)



(e) BoomerAMG, Grid $h/2^2$ (72,384 DOFs)



(f) ILU(0), Grid $h/2^2$ (72,384 DOFs)

Figure 7. Instance 2, convergence history during the PTC process for the BoomerAMG and ILU(0) preconditioners on three h-refined grids for the laminar flow over NACA0012 airfoil.

Table 4. Instance 3, settings used in the p-multigrid and single grid solutions for the inviscid flow over NACA0012 airfoil.

Polynomial order	Solution algorithm	Scaling factor for P2 modes	Number of residual updates at each pre- and post-smoothing
P2	MG	0.8	$f_J^2 = 1, n_R^2 = 1$
P3	MG	0.3	$f_J^3 = 1, n_R^3 = 1, f_J^2 = 1, n_R^2 = 1$
P3	MG_ECL	0.3	$f_J^3 = 1, n_R^3 = 1, f_J^2 = 2, n_R^2 = 2$
P2	SG	0.7	$f_J^2 = 1, n_R^2 = 1$
P3	SG	0.2	$f_J^3 = 1, n_R^3 = 1$

f_J^l = Frequency of update for Jacobian matrix at level l
n_R^l = Number of residual updates at level l
Number of point-Jacobi sweeps for all levels at all cases = 10
ECL: Enhanced Coarse Level [correction]

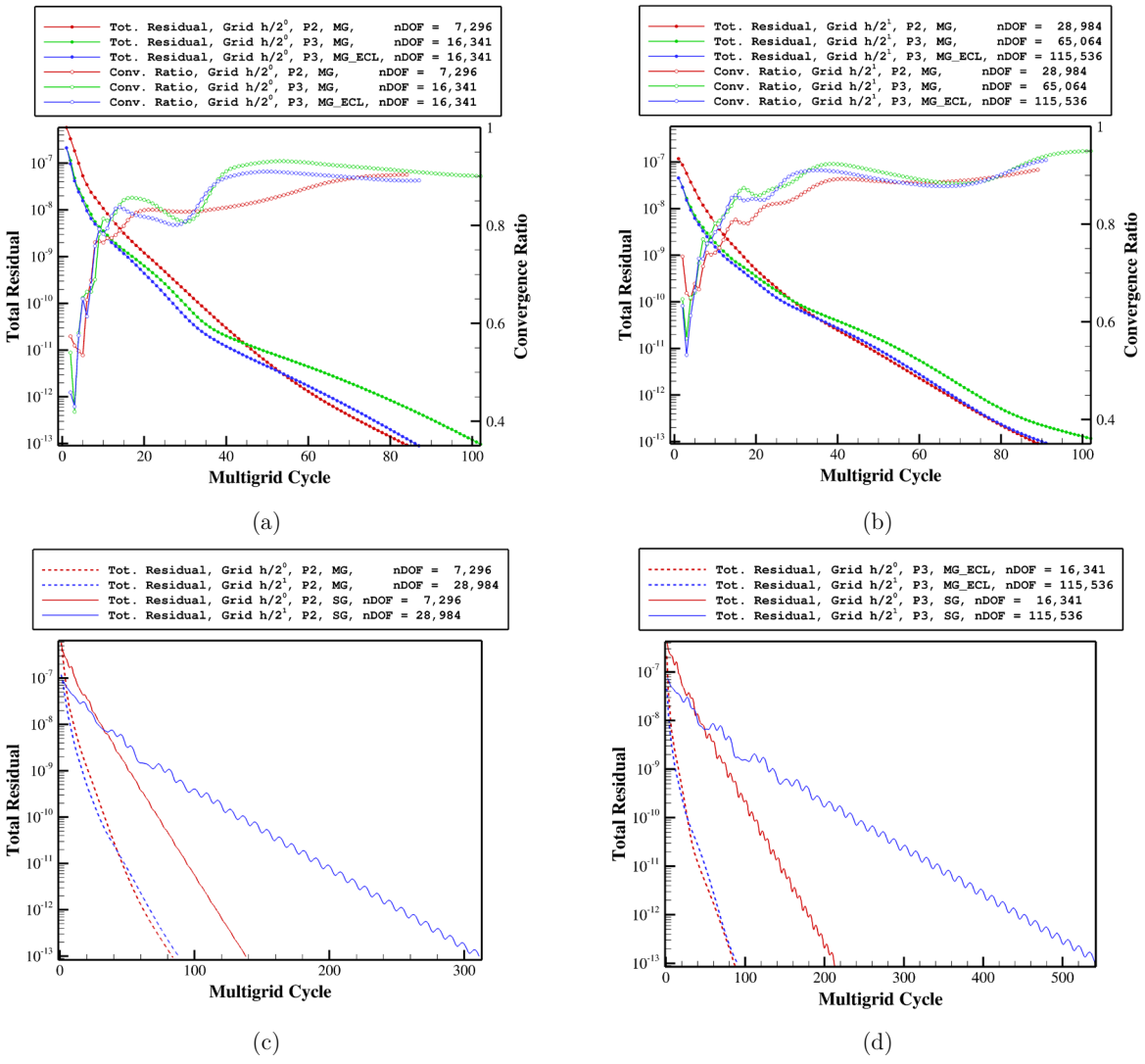


Figure 8. Instance 3, convergence histories of the p-multigrid and single grid algorithms for two meshes with P2 and P3 discretizations for the inviscid flow over NACA0012 airfoil.

D. Instance 4: Turbulent Flow over 30P30N Airfoil

The final test case is presented to assess the performance of the developed implicit-line preconditioner and non-linear p-multigrid algorithm on a high-order discretization of the RANS equations. For this purpose, P2 solutions for a 30P30N airfoil at free stream Mach number of 0.2, Reynolds number of 9.0e+6, and the angle-of-attack of 16.0 degrees are studied. The results of this test case are presented in four subsections. In the first subsection, the PTC algorithm with PILJ and ILU preconditioners is used to obtain a P1 solution. Then, in the subsequent subsections, the P1 solution is used as an initial condition and P2 solutions are sought by employing three non-linear solution methods: (1) PTC with PILJ and ILU preconditioners, (2) FAS-PMG with PILJ smoother, and (3) combination of PTC and FAS-PMG.

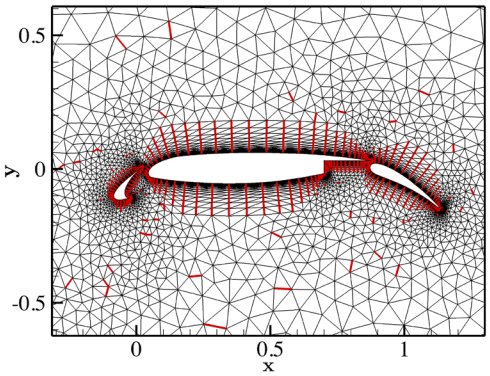
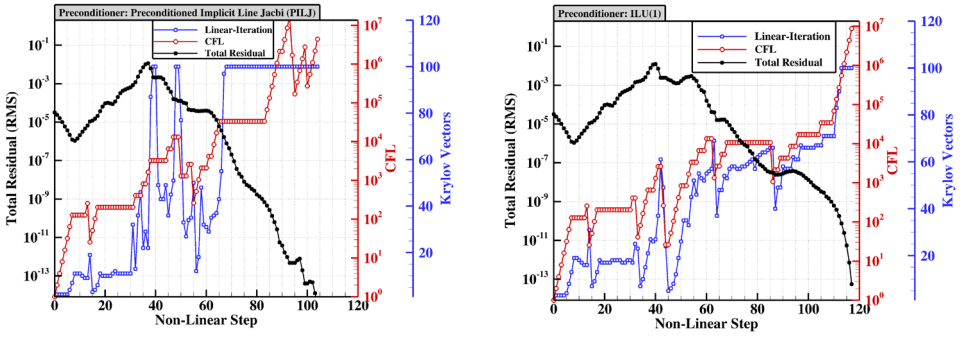


Figure 9. Instance 4, extracted lines from the stiffness matrix of the Laplace operator for the turbulent flow over 30P30N airfoil.

1. P1 Solution using PTC

Figure 9 shows the computational mesh as well as the matrix-based generated lines. For PILJ, the number of outer sweeps, number of inner sweeps, relaxation factor, and capped CFL has been set to 5, 5, 0.4, 500, respectively. This setting is kept for the following subsections. For ILU, one fill level has been used. For both preconditioners, at each non-linear step, the tolerance of the relative residual for the linear system has been set to 10^{-4}, and 100 Krylov vectors have been allocated for residual minimization. Note that in the PTC algorithm, at each non-linear step, the CFL number is modified and a line search algorithm is used to specify the optimum relaxation factor for the updates (see Eq. (32)). Figures 10a and 10b demonstrate the operation of the PTC algorithm during the non-linear convergence for both preconditioners. As seen, preconditioning using PILJ results in slightly fewer non-linear steps.

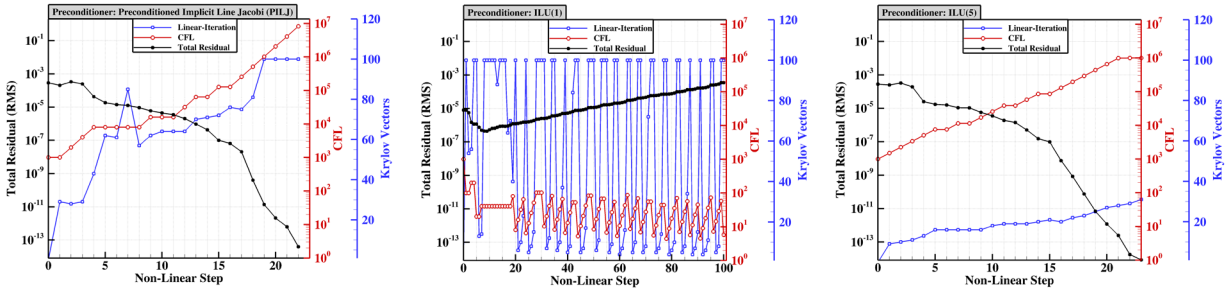


(a) (b)

Figure 10. Instance 4, convergence histories of P1 solutions using PTC algorithm with PILJ and ILU(1) preconditioners for the turbulent flow over 30P30N airfoil.

2. P2 Solution using PTC

Now the P1 solution obtained from the previous subsection is used as an initial condition and the PTC algorithm with both preconditioners is used to obtain a P2 solution. The starting CFL number is set to 1000. Figures 11a to 11c shows the operation of the PTC algorithm during the non-linear convergence for PILJ (with the same setting as the P1 problem), ILU(1), and ILU(5) preconditioners. By comparing to the results of the P1 problems, it is seen that the PILJ preconditioner can solve the P2 problem without any increase in the number of line sweeps, whereas, the ILU preconditioner requires additional fill levels to overcome the increased stiffness.

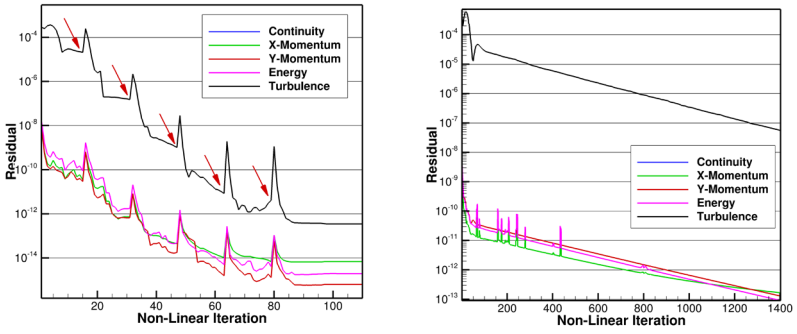


(a) (b) (c)

Figure 11. Instance 4, convergence histories of P1 solutions using PTC algorithm with PILJ, ILU(1), and ILU(5) preconditioners for the turbulent flow over 30P30N airfoil.

3. P2 Solution using FAS-PMG

In this subsection, the FAS-PMG algorithm is used to solve the P2 problem. Here, the GMRES with PILJ preconditioner and 30 Krylov vectors is used as a smoother on the fine level (P2). Figure 12a shows the convergence history of all of the five equations (continuity, momentum, energy, and turbulence) during the multigrid iterations. In these iterations, the starting CFL is set to 1000, and after each coarse grid correction (CGC) (shown by the red arrows), the CFL number is doubled. Between CGCs, the CFL number is kept constant and the updates $d\mathbf{q}$ are fully added to the old values. Note that there is no line search to specify the optimum relaxation factor ($\omega_{opt} = 1.0$). Figure 12b shows the convergence history obtained during the single grid iterations. In contrast to the multigrid solution, the starting CFL is 100 and it is kept constant during the iterations. Here, if a higher CFL number is used, or if the CFL number is increased with the same period as the multigrid case, the solution becomes unstable. Thus, the multigrid algorithm is not only faster but also more stable.



(a) (b)

Figure 12. Instance 4, convergence histories of P2 solutions using FAS-PMG and single grid algorithms for the turbulent flow over 30P30N airfoil.

4. P2 Solution using Combination of PTC and FAS-PMG

In the final subsection, in order to accelerate the non-linear convergence of the P2 problem, the PTC algorithm is augmented by the FAS-PMG algorithm. To this end, the PTC algorithm with GMRES solver and PILJ preconditioner is set up similar to the second subsection, and then the coarse grid corrector (CGC) is called at pre-specified intervals. Figure 13a shows the convergence history resulted from such a combination. By comparing to the single grid solution (shown in Fig. 13b), it is seen that, using only two CGCs, the multigrid case converges noticeable faster. Figure 14 plots the lift coefficient against the computation time for both multigrid and single grid solutions. As seen, CGC calls notably accelerate the convergence of the lift coefficient.

A shortcoming of the PTC algorithm is that a significant portion of the solution time is spent on the initial phase where the Newton convergence has not started yet. This examples shows that FAS multigrid algorithm can shorten the initial phase.

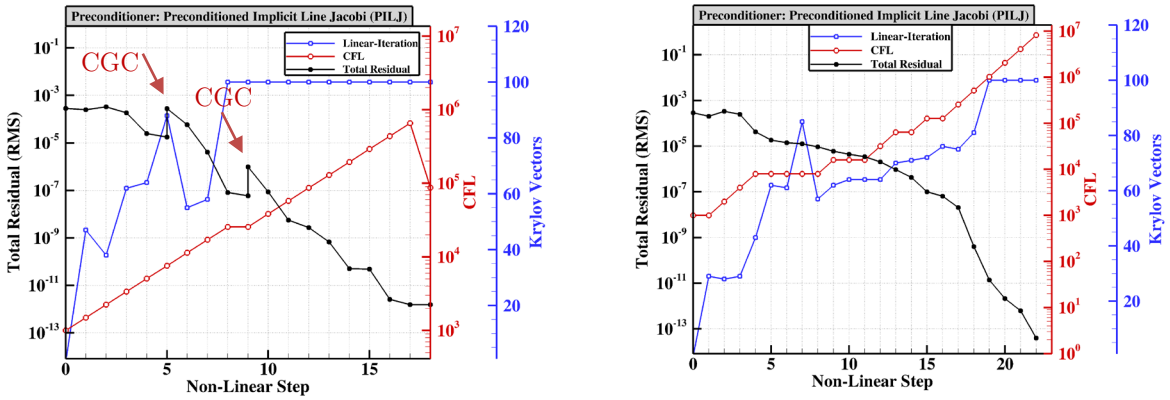


Figure 13. Instance 4, convergence histories of P2 solutions for the turbulent flow over 30P30N airfoil using (a) multigrid (combination of PTC and FAS-PMG) and (b) single grid (PTC) algorithms.

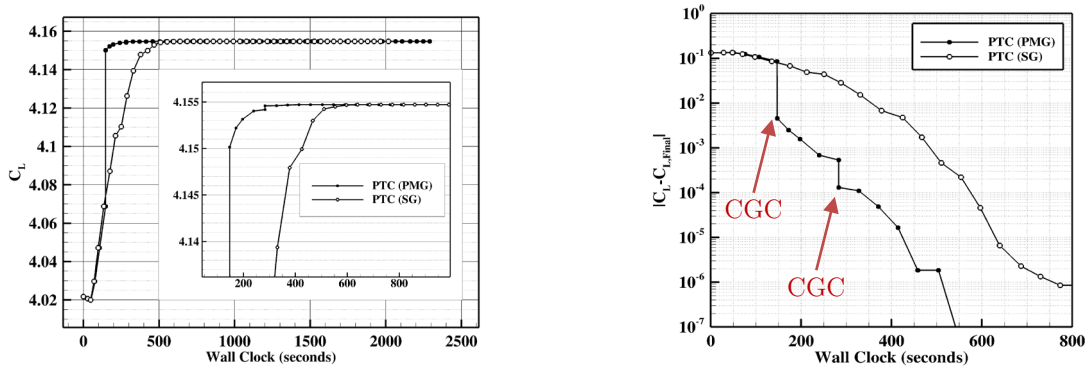


Figure 14. Instance 4, convergence of the lift coefficient in P2 solutions using multigrid (combination of PTC and FAS-PMG) and single grid (PTC) algorithms for the turbulent flow over 30P30N airfoil.

VIII. Conclusion

In this study, a multilevel solution strategy for high-order stabilized finite-element method has been studied. The proposed solution strategy is in essence a spectral multigrid approach in which the solution on the mesh with lowest polynomial degree is solved using a preconditioned Newton-Krylov method based on pseudo-transient continuation (PTC). In this approach, the solver on the coarsest level performs as an exact coarse grid corrector for the finer levels, and thus, the performance and scalability of the whole multilevel strategy greatly depends on the performance and scalability of the solver on the coarsest level.

To enhance the robustness, efficiency, and scalability of the preconditioner on the coarsest level, an implicit line relaxation method with a dual CFL strategy has been developed. The performance of this method has been compared with the incomplete (ILU) lower-upper factorization method. It has been shown that in contrast to the ILU preconditioner, the convergence behavior of the implicit line relaxation preconditioner does not depend on the number the partitions used for the domain decomposition. Also, it has been shown that the implicit-line relaxation preconditioner requires significantly less memory and it is able to solve the problem faster. Thus, this method can be a reliable substitute for the commonly-used ILU preconditioner which does not have a good scaling behavior. However, it should be noted that line solvers alone do not provide global information and will not scale optimally as the problem size increases. Thus, preconditioners that combine local techniques such as line solvers with more global methods such as algebraic multigrid (AMG) should be investigated. In this study, the BoomerAMG preconditioner (once augmented with dual CFL strategy) was found to be effective in terms of numerical convergence, but costly due to slow coarsening rates and high levels of fill obtained on the coarser levels. Therefore, additional research is required to develop more efficient AMG strategies.

To enhance the robustness of the smoothers on the high-order levels, scaling of the high-order modes was studied. Using such scaling, the implicit-line solver can also be used as a preconditioner on the high-order discretizations. However, by scaling the high-order modes, the lines mostly include the low-order modes. Last but not least, it should be noted that the developed multigrid technique has shown an h- and p- independent convergence behavior on the Euler equations.

Acknowledgments

This research was sponsored by NASA's Transformational Tools and Technologies (TTT) Project of the Transformative Aeronautics Concepts Program under the Aeronautics Research Mission Directorate.

References

[1] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, *et al.*, "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," *NASA technical report NASA/CR-2014-218178, NF1676L-18332.*

[2] U. M. Yang, "BoomerAMG: a parallel algebraic multigrid solver and preconditioner," *Applied Numerical Mathematics,* vol. 41, pp. 155-177, 2002.

[3] T. Okusanya, D. Darmofal, and J. Peraire, "Algebraic multigrid for stabilized finite element discretizations of the Navier–Stokes equations," *Computer methods in applied mechanics and engineering,* vol. 193, pp. 3667-3686, 2004.

[4] B. T. Helenbrook, "A two-fluid spectral-element method," *Computer Methods in Applied Mechanics and Engineering,* vol. 191, pp. 273-294, 2001.

[5] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, *et al.*, "PETSc web page, 2001," ed, 2004.

[6] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object-oriented numerical software libraries," in *Modern software tools for scientific computing*, ed: Springer, 1997, pp. 163-202.

[7] B. Smith, "ch. PETSc, the Portable, Extensible Toolkit for Scientific computing," in *Encyclopedia of Parallel Computing*, ed: Springer, 2011.

[8] M. W. Gee, C. M. Siefert, J. J. Hu, R. S. Tuminaro, and M. G. Sala, "ML 5.0 smoothed aggregation user's guide," Technical Report SAND2006-2649, Sandia National Laboratories2006.

[9] W. K. Anderson, L. Wang, S. Kapadia, C. Tanis, and B. Hilbert, "Petrov–Galerkin and discontinuous-Galerkin methods for time-domain and frequency-domain electromagnetic simulations," *Journal of Computational Physics,* vol. 230, pp. 8360-8385, 2011.

[10] L. Wang, W. K. Anderson, J. T. Erwin, and S. Kapadia, "Discontinuous Galerkin and Petrov Galerkin methods for compressible viscous flows," *Computers & Fluids,* vol. 100, pp. 13-29, 2014.

[11] J. C. Newman and W. K. Anderson, "Investigation of Unstructured Higher-Order Methods for Unsteady flow and Moving Domains," presented at the 22nd AIAA Computational Fluid Dynamics Conference, AIAA Paper 2015-2917, Dallas, TX, 2015.

[12] B. R. Ahrabi, W. K. Anderson, and J. C. Newman, "An Adjoint-Based hp-Adaptive Petrov-Galerkin Method for Turbulent Flows," presented at the 22nd AIAA Computational Fluid Dynamics Conference, AIAA Paper 2015-2603, Dallas, TX, June 2015.

[13] W. K. Anderson, B. R. Ahrabi, and J. C. Newman, "Finite Element Solutions for Turbulent Flow over the NACA 0012 Airfoil," *AIAA Journal,* vol. 54, No. 9, pp. 2688-2704, Sept 2016.

[14] B. R. Ahrabi, "An hp-Adaptive Petrov-Galerkin Method for Steady-State and Unsteady Flow Problems," PhD, Department of Computational Engineering, University of Tennesee at Chattanooga, 2015.

[15] D. J. Mavriplis, "Multigrid techniques for unstructured meshes," ICASE-95-27, Institute for Computer Applications in Science and Engineering, Hampton, VA, 1995.

[16] D. J. Mavriplis and S. Pirzadeh, "Large-scale parallel unstructured mesh computations for three-dimensional high-lift analysis," *Journal of aircraft,* vol. 36, pp. 987-998, 1999.

[17] D. Mavriplis, M. Long, T. Lake, and M. Langlois, "NSU3D Results for the Second AIAA High-Lift Prediction Workshop," *Journal of Aircraft,* vol. 52, pp. 1063-1081, 2015.

[18] C. R. Nastase and D. J. Mavriplis, "High-order discontinuous Galerkin methods using an hp-multigrid approach," *Journal of Computational Physics,* vol. 213, pp. 330-357, 3/20/ 2006.

[19] K. Shahbazi, D. J. Mavriplis, and N. K. Burgess, "Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations," *Journal of Computational Physics,* vol. 228, pp. 7917-7940, 11/20/ 2009.

[20] S. R. Allmaras and F. T. Johnson, "Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model," in *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, 2012, pp. 1-11.

[21] Ryan S. Glasby, J. Taylor Erwin, Douglas L. Stefanski, Steve L. Karman, and W. K. Anderson, "Results from HPCMP CREATETM-AV COFFE for Tasks 1-3 for DPW6," in *6th AIAA CFD Drag Prediction Workshop, AIAA Aviation 2016, Washington, D.C., 16-17 June 2016.*

[22] P.-O. Persson and J. Peraire, "Sub-cell shock capturing for discontinuous Galerkin methods," presented at the 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2006-0112, Reno, Nevada, January 2006.

[23] G. E. Barter and D. L. Darmofal, "Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation," *Journal of Computational Physics,* vol. 229, pp. 1810-1827, 2010.

[24] C. H. Whiting and K. E. Jansen, "A stabilized finite element method for the incompressible Navier-Stokes equations using a hierarchical basis," *International Journal for Numerical Methods in Fluids,* vol. 35, pp. 93-116, 2001.

[25] C. H. Whiting, K. E. Jansen, and S. Dey, "Hierarchical basis for stabilized finite element methods for compressible flows," *Computer Methods in Applied Mechanics and Engineering,* vol. 192, pp. 5167-5185, 2003.

[26] A. N. Brooks and T. J. Hughes, "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering,* vol. 32, pp. 199-259, 1982.

[27] T. J. R. Hughes, G. Scovazzi, and T. E. Tezduyar, "Stabilized methods for compressible flows," *Journal of Scientific Computing,* vol. 43, pp. 343-368, 2010.

[28] T. Tezduyar and T. Hughes, "Finite element formulations for convection dominated flows with particular emphasis on the compressible Euler equations," presented at the 21st aerospace sciences meeting, AIAA Paper 1983-0125, 1983.

[29] S. Marras, J. F. Kelly, F. X. Giraldo, and M. Vázquez, "Variational multiscale stabilization of high-order spectral elements for the advection–diffusion equation," *Journal of Computational Physics,* vol. 231, pp. 7187-7213, 2012.

[30] J. T. Erwin, W. K. Anderson, S. Kapadia, and L. Wang, "Three-dimensional stabilized finite elements for compressible Navier–Stokes," *AIAA Journal,* vol. 51, pp. 1404-1419, 2013.

[31] J. Gressier, P. Villedieu, and J.-M. Moschetta, "Positivity of flux vector splitting schemes," *Journal of Computational Physics,* vol. 155, pp. 199-220, 1999.

[32] C. Wang and J. Liu, "Positivity property of second-order flux-splitting schemes for the compressible Euler equations," *Discrete And Continuous Dynamical Systems Series B,* vol. 3, pp. 201-228, 2003.

[33] T. J. Barth, "Numerical methods for gasdynamic systems on unstructured meshes," in *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, ed: Springer, 1999, pp. 195-285.

[34] F. Shakib, T. J. Hughes, and Z. Johan, "A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering,* vol. 89, pp. 141-219, 1991.

[35] C. Rumsey. *Turbulence Modeling Resource Website.* Available: http://turbmodels.larc.nasa.gov

[36] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics,* vol. 43, pp. 357-372, 1981.

[37] Y. Bazilevs and I. Akkerman, "Large eddy simulation of turbulent Taylor–Couette flow using isogeometric analysis and the residual-based variational multiscale method," *Journal of Computational Physics,* vol. 229, pp. 3402-3414, 2010.

[38] Y. Bazilevs and T. J. Hughes, "Weak imposition of Dirichlet boundary conditions in fluid mechanics," *Computers & Fluids,* vol. 36, pp. 12-26, 2007.

[39] Y. Bazilevs, C. Michler, V. Calo, and T. Hughes, "Weak Dirichlet boundary conditions for wall-bounded turbulent flows," *Computer Methods in Applied Mechanics and Engineering,* vol. 196, pp. 4853-4862, 2007.

[40] J. Nitsche, "Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind," *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg,* vol. 36, pp. 9-15, 1971.

[41] M. F. Wheeler, "An elliptic collocation-finite element method with interior penalties," *SIAM Journal on Numerical Analysis,* vol. 15, pp. 152-161, 1978.

[42] R. Hartmann, "Adjoint consistency analysis of discontinuous Galerkin discretizations," *SIAM Journal on Numerical Analysis,* vol. 45, pp. 2671-2696, 2007.

[43] N. K. Burgess and R. S. Glasby, "Advances in numerical methods for CREATETM-AV analysis tools," presented at the 52nd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2014-0417, 2014.

[44] M. Ceze and K. J. Fidkowski, "Constrained pseudo-transient continuation," *International Journal for Numerical Methods in Engineering,* vol. 102, pp. 1683-1703, 2015.

[45] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing,* vol. 7, pp. 856-869, 1986.

[46] B. Philip and T. P. Chartier, "Adaptive algebraic smoothers," *Journal of Computational and Applied Mathematics,* vol. 236, pp. 2277-2297, 3// 2012.

[47] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing,* vol. 20, pp. 359-392, 1998.

[48] B. Hendrickson and R. Leland, "The Chaco user's guide: Version 2.0," Technical Report SAND95-2344, Sandia National Laboratories, Albuquerque, NM, July 1995.

[49] N. L. Mundis, "The development of a robust, efficient solver for spectral and spectral-element time discretizations," PhD, Department of Mechanical Engineering, University of Wyoming, 2014.

[50] P. Solin, K. Segeth, and I. Dolezel, *Higher-Order Finite Element Methods*: CRC Press, 2003.

[51] J. C. Vassberg, M. A. DeHaan, S. M. Rivers, and R. A. Wahls, "Development of a common research model for applied CFD validation studies," presented at the 26th AIAA Applied Aerodynamics Conferencee, AIAA Paper 2008-6919, Honolulu, Hawaii, August 2008.