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

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

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

This paper extends the application of traditional implicit line smoothers to high-order continuous finite-element methods. It is shown that for high-order continuous finite-element discretizations, the interconnections of the degrees of freedom on an implicit line form a banded matrix which is wider than tridiagonal, but still can be factorized completely without generating any fill-ins. This point is then utilized to develop an implicit line preconditioner for Krylov solvers such as GMRES. To improve the robustness of the implicit-line preconditioner, a dual CFL strategy, with a lower CFL number in the preconditioner matrix, is developed. The developed implicit line preconditioner is employed within a high-order multilevel solver framework, in which the compressible 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. In the developed framework, two nonlinear solution strategies, including an inexact Newton method and a p-multigrid scheme, are combined in various ways to devise robust, scalable and efficient solution techniques for reaching steady-state solutions. It is shown that for high-order solutions, the application of the p-multigrid approach before the Newton iterations can drastically reduce the number of nonlinear iterations required to reach full convergence. Two three-dimensional numerical examples including turbulent flows over an M6 wing, and a wing-body-tail configuration from the 4th AIAA Drag Prediction Workshop are presented in which the performance of the implicit line preconditioner and the multilevel solution technique is demonstrated. Results show positive steps toward development of scalable solution techniques.

I. Introduction

During the last few decades, there has been a growing interest in the development and utilization of stabilized finite-element methods for the solution of the compressible Reynolds-Averaged Navier-Stokes (RANS) equations on unstructured grids. The major characteristics of these methods are the continuity of the solution space and minimal cross-wind dissipation. The latter is obtained by a stabilization term which can be interpreted as a perturbation to the classical Galerkin weight functions, such that the up-winding effect is made in the characteristic directions; thus, these methods are also known as Petrov-Galerkin (PG) methods. In the field of computational fluid dynamics (CFD), PG methods are naturally compared with the traditional second-order finite-volume (FV) as well as the developing discontinuous Galerkin (DG) finite-element methods. Some of the advantages of PG methods over the FV methods are: (1) the use of nearest neighbor stencils to reach high-order discretizations, (2) the possibility of the high-order representation of the geometry, (3) more capable mesh adaptivity, and (4) less dissipative solutions, particularly on triangular (in 2D) and tetrahedral (in 3D) elements. Although some of these advantages are shared with DG methods, several studies have shown that for moderate discretization orders (less than 5), and for comparable accuracies, a PG scheme

[1]AIAA Senior Member, Associate Research Scientist, brezaahr@uwyo.edu

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

requires significantly less computational resources than a DG scheme. In view of these advantages, PG schemes have been employed in several solver frameworks such as GGNS, PHASTA [1-4], FUNSAFE [5-9], FUN3D [10, 11], and the COFFE [12-15].

A major incentive for using finite-element methods in CFD is to benefit from high-order discretizations, to reach higher accuracies at lower computational costs. However, numerical experiments have shown that high-order discretizations could result in extremely stiff problems, in both linear and non-linear aspects, which could annihilate the aforementioned incentive. This fact motivates the development of robust and scalable solution algorithms that can be effectively used on current and emerging high-performance computing (HPC) architectures. Seeking robustness, there has been a recent trend towards incomplete factorization methods, such ILU(k), generally used as preconditioners within Newton-Krylov algorithms. However, as shown in our previous work [16], due to their sequential nature these methods are not robustly scalable. Note that in the context of domain decomposition, factorization methods are most often applied locally within each partition; this leads to decreased convergence rates with increasing levels of parallelization. Consequently, the robustness offered by these methods on smaller problems may not be realized on larger problems. As we discussed in Ref. [16], multilevel and multigrid algorithms are powerful candidates to maximize the usage of current and emerging HPC systems, as they have the potential to be numerically optimal, and computationally scalable. These methods are generally designed such that the local errors are reduced by local smoothers on the finer levels of the multilevel sequence, while the global implicitness is achieved through reduced long-range communications on the coarser levels. Following the remarkable success of the multigrid algorithms in FV schemes, application of these methods in finite-element schemes have become an attractive topic for solver technologies.

A major challenge to employing multigrid methods in complex CFD applications is the robustness issue. To alleviate this problem for high-order stabilized finite-element flow solvers, in our previous study [16] we developed a p-multigrid approach in which the solution on the mesh with the lowest polynomial degree (i.e. linear elements) is solved using a preconditioned Newton-Krylov algorithm based on the pseudo-transient continuation (PTC) method. In this approach, the solver on the coarsest level could be used as an exact or approximate coarse grid corrector (CGC) for the finer levels. To enhance the robustness, efficiency and scalability of the preconditioner on the coarsest level, an implicit line smoother with a dual CFL strategy was developed. It was shown that by using the dual CFL strategy, the developed line preconditioner can overcome the stability issues caused by high CFL numbers within the PTC algorithm during the solution of steady-state problems. In that study, the performance of the line preconditioner was compared with an ILU(k) preconditioner and it was shown that the implicit-line preconditioner requires significantly less memory, can solve the problem faster, and its convergence behavior is independent of the number of partitions used in the domain decomposition. In that study, we also utilized the implicit line method on the high-order levels, both as a preconditioner for the GMRES solver within the PTC algorithm, as well as a smoother for finer levels of the multigrid algorithm. Although the implicit line method was found to be effective in both mentioned solution strategies, it was not developed in a comparable manner to the highest order of discretization. Meaning, the implicit lines on high-order levels only included the nodes that are shared with the lowest level, and thus, the smoothing on high-order degrees of freedom was done in a point explicit manner.

In the present work, the previously developed line smoother is improved for high-order discretizations. It is shown that for high-order discretizations, the interconnections of the degrees of freedom on an implicit line form a banded matrix which is wider than tridiagonal, but still can be factorized completely without generating any fill-ins. Also, in this study, the application of the developed p-multigrid solver is extended to three-dimensional problems, and the improved line preconditioner is tested on high-order discretizations.

The reminder of this paper is arranged as the following. First, the features of the developed flow solver for this study is briefly reviewed. Next, the governing equations and the 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

All the development in the present work has been done on the HOMA (High-Order Multilevel Adaptive) solver, which was initially introduced in Ref. [16]. In this flow solver, the compressible Reynolds averaged Navier-Stokes (RANS) equations can be discretized using a Petrov-Galerkin finite-element method. The lowest polynomial space is constructed using the Lagrange linear basis functions, and the higher-order spaces are built using the high-order Lagrange basis functions or a set of hierarchical basis functions. The solver is capable of handling two- and three-dimensional unstructured meshes with mixed element types. The time integration for steady-state and unsteady problems is fully implicit and the linearization is done exactly using the automatic differentiation (AD) technique. 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) [17], and a non-linear p-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 [18-20] has been integrated into the solver, through which the external packages as Hyper/BoomerAMG [21] and Trilinos/ML [22] are accessible for the preconditioning.

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 [23]. 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

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

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

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

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

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

IV. Discretization

A. Spatial Discretization

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

$$\mathcal{L}(\mathbf{q}) = \mathbf{S} \qquad \vec{x} \in \Omega \text{ and } t \in [0, \infty) \tag{14.a}$$

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

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

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

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 the 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. [1, 2])

$$\mathcal{L} \coloneqq [\mathbf{A}^q]\frac{\partial}{\partial t} + [\mathbf{A}_i^E]\frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\mathbf{G}_{\mathrm{ij}}\frac{\partial}{\partial x_j}\right) \tag{15}$$

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 $\big[\mathbf{G}_{ij}\big]$ is the diffusivity matrix which is defined such that $\mathbf{F}_i^v = \big[\mathbf{G}_{ij}\big]\frac{\partial \mathbf{q}}{\partial x_j}$. 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* $\mathcal{T}^h = \{\Omega^1, \Omega^2, \ldots, \Omega^{n_{el}}\}$ 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

$$\boldsymbol{\mathcal{S}}_t^h \coloneqq \{\mathbf{q}|\mathbf{q}(\cdot,t) \in [\mathcal{H}^1(\Omega^h)]^{n_Q}, \mathbf{q}(\cdot,t)|_{\Omega^e} \in [\mathcal{P}_{P^e}(\Omega^e)]^{n_Q}, t \in [0,\infty)\ \forall e \text{ and } \mathbf{q}(\cdot,t) = \mathbf{q}_D^h \text{ on } \Gamma_D^h\} \tag{16}$$

$$\mathcal{W}^h \coloneqq \ \{w|\ w \in \mathcal{H}^1(\Omega^h);\ w|_{\Omega^e} \in \mathcal{P}_{P^e}(\Omega^e)\ \forall e \text{ and } w = 0 \text{ on } \Gamma_D^h\} \tag{17}$$

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 $w^h \in \mathcal{W}^h$,

$$\iint_{\Omega^h}\left(w^h[\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t} - \frac{\partial w^h}{\partial x_i}\mathbf{F}_i - w^h\mathbf{S}\right)d\Omega + \int_{\Gamma_F^h} w^h(\mathbf{F}_i^b n_i)\ d\Gamma = \mathbf{0} \tag{18}$$

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:

$$\mathbf{q}^h = \sum_{i=1}^{n_{DOF}} \mathbf{q}_i N_i \quad \text{on } \Omega^h \tag{19}$$

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) [24-26] scheme has been used for stabilization. In this scheme, a stabilization term is added to the Galerkin discretization as

$$\underbrace{\iint_{\Omega^h}\left(N[\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t}-\frac{\partial N}{\partial x_i}\mathbf{F}_i-N\mathbf{S}\;d\Omega\right)d\Omega+\int_{\Gamma_F^h}N(\mathbf{F}_i^b n_i)\;d\Gamma}_{Galerkin\ Discretization}+\underbrace{\sum_{e=1}^{n_{el}}\iint_{\Omega^e}[\mathbf{P}^e]\left([\mathbf{A}^q]\frac{\partial \mathbf{q}^h}{\partial t}+\frac{\partial \mathbf{F}_i}{\partial x_i}-\mathbf{S}\right)d\Omega}_{Stabilization\ Term}=\mathbf{0} \quad (20)$$

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 [27]. 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 [25]. For the SUPG method,

$$[\mathbf{P}^e]=\left[\frac{\partial \mathbf{F}_i^E}{\partial \mathbf{Q}}\right]\frac{\partial N}{\partial x_i}[\boldsymbol{\tau}^e] \quad (21)$$

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 [28-30]. In the present study, only the stabilization based on eigensystem decomposition has been used [31] with viscous scaling similar to that described in Ref. [32]. This term for element e is given by

$$[\boldsymbol{\tau}^e]^{-1}=\sum_{j=1}^{n_s^e}\left|\frac{\partial N_j^e}{\partial x_i}\left[\frac{\partial \mathbf{F}_i^E}{\partial \mathbf{Q}}\right]\right|+\frac{\partial N_j^e}{\partial x_i}[\mathbf{G}_{ik}]\frac{\partial N_j^e}{\partial x_k} \quad (22)$$

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. [33]). In the above equation,

$$\left|\frac{\partial N_j^e}{\partial x_i}\left[\frac{\partial \mathbf{F}_i^E}{\partial \mathbf{Q}}\right]\right|=[\mathbf{T}][\boldsymbol{\Lambda}][\mathbf{T}]^{-1} \quad (23)$$

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.

B. Initial and Boundary Conditions

In this study, initial conditions are set to free-stream conditions except for 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 [34].

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 [35] based on the free-stream and interior state values. For inviscid walls, the boundary flux vector only takes the pressure from interior and thus $\mathbf{F}_i^b=[0,\delta_{1i}p,\delta_{2i}p,\delta_{3i},0]^T$.

V. Nonlinear Solution Techniques

As mentioned in the introduction, the objective of the present work is development of robust, memory-efficient, and scalable solution strategies and preconditioning techniques for high-order stabilized finite-element methods. The focus is put on the steady-state solution techniques. In this section, the principal nonlinear advancement techniques are described. The first technique is known as pseudo-transient continuation (PTC) [17], which, in essence, is an inexact Newton method with a line search process to find an optimum relaxation factor for the update equation. The second technique is a nonlinear p-multigrid (PMG) correction method which has been developed based on the full approximation scheme (FAS). The PMG algorithm has been developed with a multilevel structure in which the PTC algorithm can be used a nonlinear smoother on the fine levels ($P \geq 2$), or as a solver on the coarsest level ($P = 1$). In fact, the solver on the coarsest level performs as an exact or approximate coarse grid corrector (CGC) for the finer levels. Finally, it should be noted that at each nonlinear step, in both PTC and PMG algorithms, there is a linear solve. In the HOMA solver, the GMRES [38] method is used as the principal linear solver. The preconditioning techniques used for this solver are described in section VI.

A. Pseudo-Transient Continuation (PTC)

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

$$[\mathbf{M}^h]\frac{\partial \mathbf{q}^h}{\partial t} + \mathbf{R}^h(\mathbf{q}^h) = \mathbf{0} \tag{24}$$

where $\mathbf{R}^h$ represents the spatial residual, and $[\mathbf{M}^h]$ denotes the mass matrix. Applying the backward difference formula (BDF) on Eq. (24) results in the following implicit formulation:

$$\mathbf{Res}^{h,n+1}(\mathbf{q}^{h,n+1}) = \frac{[\mathbf{M}^h]}{\Delta t}(\mathbf{q}^{h,n+1} - \mathbf{q}^{h,n}) + \mathbf{R}^h(\mathbf{q}^{h,n+1}) = 0 \tag{25}$$

where $\mathbf{Res}^{h,n+1}$ represents the unsteady flow residual at time step $n+1$. This implicit system is then linearized using an automatic differentiation technique and the vector of the primitive variables is updated in a Newton-type iteration similar to that in Refs. [36, 37]. The linearized system and the update equation are given by

$$[\mathbf{J}^{h,n}(\mathbf{q}^{h,n})]\Delta\mathbf{q}^{h,n} = -\mathbf{R}^{h,n}(\mathbf{q}^{h,n}) \tag{26}$$

$$\mathbf{q}^{h,n+1} = \mathbf{q}^{h,n} + w_{opt}\Delta\mathbf{q}^{h,n} \tag{27}$$

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

$$[\mathbf{J}^h] = \left[\frac{[\mathbf{M}^h]}{\text{CFL } \Delta t} + \frac{\partial \mathbf{R}^h}{\partial \mathbf{Q}^h}\right] \tag{28}$$

At small CFL numbers, the algorithm essentially becomes a simple time stepping 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 shown in the present work: $\theta = 0.2$, $w_{min} = 0.1$, and $\beta_{CFL} = 1.5$.

B. Multigrid Correction

For high-order ($P \geq 2$) problems, the HOMA solver is equipped with a nonlinear p-multigrid (PMG) algorithm, which has been developed based on 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, which was described in the previous subsection. The general concept of this algorithm is described in the following (see also Ref. [39]). Consider the solution of the discrete problem

$$L^l \mathbf{q}^l = \mathbf{g}^l \tag{29}$$

where, for a p-multigrid approach, the superscripts refer to the discretization order of the mesh level. The current estimate of the solution $\mathbf{q}^l$ is denoted as $\tilde{\mathbf{q}}^l_,$ which is obtained by approximate solution of the above equation using an iterative method. The objective of the multigrid correction scheme is to calculate a correction for the approximate solution such that

$$\mathbf{q}^l = \tilde{\mathbf{q}}^l + \mathbf{v}^l \tag{30}$$

Since $\tilde{\mathbf{q}}^l$ does not satisfy the discrete problem exactly, a residual can be defined as

$$\mathbf{r}^l = \mathbf{g}^l - L^l \tilde{\mathbf{q}}^l = L^l \mathbf{q}^l - L^l \tilde{\mathbf{q}}^l \tag{31}$$

If L is a linear operator, the above equation can be written as

$$L^l \mathbf{v}^l = \mathbf{r}^l \tag{32}$$

Now if high-frequency errors in the solution have been eliminated by sufficient fine level smoothing cycles, the sought correction v^l must be smooth and can therefore be computed more efficiently on a coarser level by solving the equation

$$L^{l-1} \mathbf{v}^{l-1} = \hat{I}_l^{l-1} \mathbf{r}^l \tag{33}$$

where the superscript $l - 1$ denotes the coarser level in the multilevel sequence. Also, $\hat{I}_l^{l-1}$ is a restriction operator, which projects the residual vector from the fine level into the coarse level. The exact or approximate solution of v^{l-1} must be employed to correct the original solution on the fine level. This is done as

$$\tilde{\mathbf{q}}^{\,l,new} = \tilde{\mathbf{q}}^{l,new} + I_{l-1}^l \mathbf{v}^{l-1} \tag{34}$$

where I_{l-1}^l is a prolongation operator, which interpolates the coarse level corrections onto the fine level. Once the fine level values have been updated, they may be smoothed again by additional fine level iterations, and the entire process, which constitutes a multigrid cycle, may be repeated until convergence is attained.

If the operator L^l is nonlinear, the difference $L^l\mathbf{q}^l - L^l\tilde{\mathbf{q}}^l$ in Eq. (31), can no longer be replaced by $L^l\mathbf{v}^l$, and thus the above scheme must be modified. This is achieved by introducing a coarse level variable $\mathbf{q}^{l-1}$ defined as

$$\mathbf{q}^{l-1} = \ I_l^{l-1}\tilde{\mathbf{q}}^l + \mathbf{v}^{l-1} \tag{35}$$

where I_l^{l-1} is a restriction operator for the solution variables. The coarse level correction equivalent to Eq. (32) can now be written as

$$L^{l-1}\mathbf{q}^{l-1} - L^{l-1}I_l^{l-1}\tilde{\mathbf{q}}^l = \hat{I}_l^{l-1}\mathbf{r}^l \tag{36}$$

It is useful to rewrite the above equation as

$$L^{l-1}\mathbf{q}^{l-1} = \mathbf{S}^{l-1} \tag{37}$$

where

$$\mathbf{S}^{l-1} = L^{l-1}I_l^{l-1}\tilde{\mathbf{q}}^l + \hat{I}_l^{l-1}\mathbf{r}^l \tag{38}$$

in above form, the coarse level equation is seen to take on a similar structure to the original fine level equation, with a modified source term. This enables the use of similar solution strategies for both coarse and fine levels. After the exact or approximate solution of the coarse level, the fine level variables are updated as

$$\mathbf{q}^{l,new} = \mathbf{q}^{l,old} + I_{l-1}^l(\mathbf{q}^{l-1} - I_l^{l-1}\tilde{\mathbf{q}}^l) \tag{39}$$

which can also be written as

$$\mathbf{q}^{l,new} = \mathbf{q}^{l,old} + I_{l-1}^l\mathbf{v}^{l-1} \tag{40}$$

In the above, the concept of the nonlinear multigrid correction was described for a two-level system. When the multilevel sequence includes more than two levels, the present procedure can be performed recursively on coarser levels. Algorithm 1 describes such an approach for a multilevel system. In this algorithm, l indexes the solution level, n_l is the number of grid 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 2), which is formed based on the linearized system in Eq. (26). $\mathbf{n}_R = \{n_R^l\}$ 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 2, *LSmoother* denotes a linear smoother. Generally, for inviscid flows, this smoother is point Jacobi and for laminar flows, it is preconditioned implicit line Jacobi (PILJ) which will be described in section VI.B. However, for difficult turbulent problems, it is useful to use the implicit line Jacobi as a preconditioner within the GMRES algorithm. By doing this, the linear smoother remains stable, even if the preconditioner becomes unstable (e.g. due to extreme stiffness and/or lack of diagonal dominancy). Note that in contrast to the classical linear smoothers (such as point Jacobi or implicit line smoother), which provide local updates to the solution, the Krylov solvers utilize a combination of global updates to minimize a specific norm and thus, they often do not have good smoothing properties. This problem can be alleviated by using the classical smoothers as the preconditioner for the Krylov methods. In the HOMA solver, the PTC algorithm (equipped with a preconditioned GMRES) is tuned to be used as the nonlinear smoother. A notable difference with the usual use of the PTC algorithm is that for multigrid iterations, the CFL number is either kept constant or grown slowly (e.g. $\beta = 1.1$). By doing this, the PTC works like a smoother rather than a solver.

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 problem #
 9:     $\mathbf{q}^1 = Solve(\mathbf{q}^{1,0}, \mathbf{S}^1)$    # Using PTC #
10: # Prolong the coarse level correction to the fine level and update the fine level solution#
12:     $\mathbf{q}^{l,0} = \mathbf{q}^l + I_{l-1}^l(\mathbf{q}^{l-1} - \mathbf{q}^{l-1,0})$
13: # Post-smoothing #
14:     for $l = 1, n_l$
15:         $\mathbf{q}^l = NSmoother(n_R^l, f_J^l, n_o^l, n_i^l, \omega^l, \mathbf{q}^{l,0}, \mathbf{S}^l)$   # See Algorithm 4 #
17:     end for
18: end for
19: 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:     $d\mathbf{Q} = LSmoother(n_o, n_i, \omega, [\mathbf{J}], [\mathbf{P}], \hat{\mathbf{R}}^k)$  # Linear smoother or linear solver#
6:     Use a constant relaxation factor or perform a line search to find the optimum relaxation factor $\omega_{opt}$
7:     $\mathbf{q}^{k+1} = \mathbf{q}^k + \omega_{opt} d\mathbf{q}$
8: end for
9: return $\mathbf{q}^{n_R}$
```

In our previous work [16], we employed the developed multigrid algorithm to solve the Euler equations on a two-dimensional problem, and obtained an h- and p- independent convergence behavior. This methodology was demonstrated for hierarchical quadratic and cubic elements. In the present work, the algorithm is extended to three-dimensional problems. However, this time, only quadratic elements (with Lagrange basis functions) are studied. Also, it should be noted that the restriction and prolongation operators have been obtained using Galerkin projection and simple interpolation, respectively.

1. *Remarks on Domain Decomposition*

For parallel implementation, first the P2 level is partitioned and then, the P1 partitions are generated by reducing the polynomial order of the P2 partitions. Using this approach the restriction and prolongation processes can be performed locally in each partition. However, note that the load balance during the partitioning process is done based on the P2 level. An alternative approach is to partition the P1 level independent from the P2 level. Although such approach might result in a better load balance for solution on the P1 level, it may cause a notable increase in communication for restriction and prolongation operators. Also, it can drastically complicate the implementation. In this study, by testing several two- and three-dimensional test cases, we realized that the load imbalance (for P1 level) caused by the first approach is negligible.

VI. Linear Preconditioners and Smoothers

Preconditioners and smoothers are some of the building blocks of the linear solvers/smoothers within the PTC and PMG algorithms (see Eq. (26)). This section describes the details of the techniques which have been developed in the present work for this purpose. As mentioned earlier, for difficult turbulent flows, the linear system in Eq. (26) is solved using preconditioned GMRES [38] method.

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]. To reduce the number of fill-ins in this approach, the degrees of freedom can be 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. Thus, 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 our previous work [16], we showed that implicit line smoothers can be reliable substitutes for incomplete factorization methods.

B. Implicit-Line Smoother

The block Jacobi smoother is extensively used in the numerical solution of the linear systems, either as a stand-alone solver, or as a preconditioner for linear solver accelerators such as GMRES. Besides its simplicity, a great advantage of the Jacobi smoother is its parallel scalability. However, it is well known that, due to its pure explicit nature, this method suffers from low convergence rates. This shortcoming can be greatly improved by grouping strongly connected unknowns in separate groups and solving for them in a semi-implicit manner. This notion has been the key idea behind the development of traditional implicit line smoothers, in which the strongly coupled unknowns are grouped in lines, ordered sequentially, and solved for using a tridiagonal solver. This technique can be further explained using the example shown in Fig. 1a. In this figure, a triangular mesh (with linear basis functions) is shown. Two separate lines are colored in blue and red. As seen, the nodes are ordered sequentially based on the lines. The sparsity pattern of the corresponding stiffness matrix is shown in Fig. 1b. In this figure, the interconnections between the blue nodes are shown in blue, the interconnections between red nodes are shown in red, and the connections between blue and red nodes are shown in purple. Note that by ordering based on the lines, the interconnections of the nodes form a tridiagonal structure. Using this structure, Algorithm 3 can be used as a solver or a smoother to solve a typical linear system of equations in a semi-implicit manner. Note that typically, in this algorithm the update equation is solved using the Thomas algorithm [40], which is suitable when the interconnections of the nodes form a tridiagonal structure. This is the case for second-order finite-volume and finite-element schemes. Through numerical experiments, we realized that for third-order continuous finite-element schemes, the tridiagonal solver can result in uncontrollable instabilities on some of the lines. This problem can be explained using an example. Figure 2a shows a sample mesh with P2 elements. In this figure, the blue nodes are those that are shared with the P1 level, and the red nodes belong solely to the P2 level. Also, shown in this figure is a sample line which includes both the blue and red nodes. Figure 2b shows the corresponding sparsity pattern of the stiffness matrix for the numbered nodes. As seen, in this case, the interconnections of the line nodes form a pentadiagonal structure. Thus, a tridiagonal solver cannot solve the corresponding unknowns in an implicit manner. In the present work, we employed a pentadiagonal solver for these lines and retained the stability of the implicit line solver. It should be noted that the lines that include only P2 nodes (i.e. lines with only red nodes) form a tridiagonal structure and thus they can be solved using the Thomas algorithm [40]. By careful inspection of Fig. 2b, one can show that application of the Gauss elimination algorithm on the pentadiagonal portion of the stiffness matrix generates no fill-ins. Therefore, the ILU(0) algorithm returns the complete factorization of this portion of the matrix. Although here this point was demonstrated for a P2 discretization, by using similar schematics, one can easily show that generally for high-order continuous finite-element discretizations, the interconnections between the degrees of freedom on an implicit line form a banded matrix which can be completely factored without generating any fill-ins. This is a key finding of the present work.

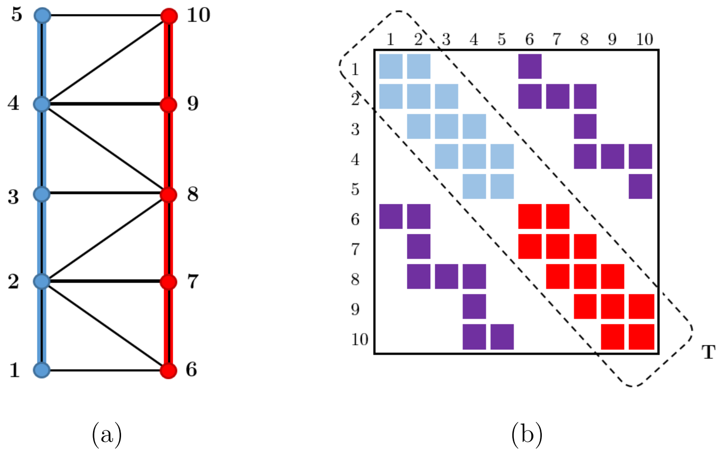


Figure 1. (a) A sample mesh with linear basis functions and two lines, (b) sparsity pattern of the stiffness matrix of the shown mesh.

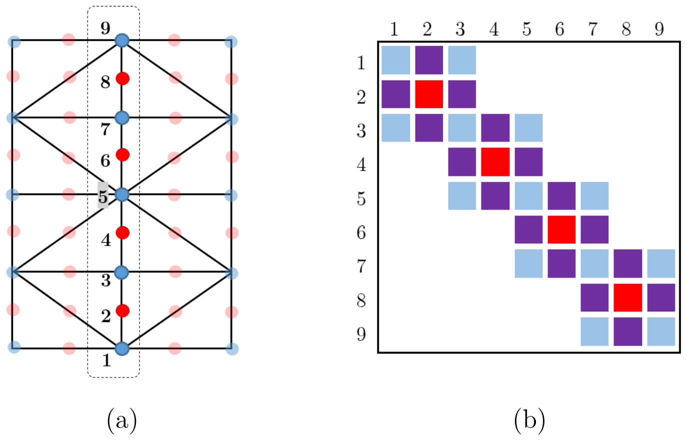


Figure 2. (a) A sample mesh with quadratic basis functions and one line, (b) sparsity pattern of the stiffness matrix for the numbered nodes.

Algorithm 3. 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:     $\mathbf{r}^{k-1} = \mathbf{b} - [\mathbf{A}]\mathbf{x}^{k-1}$     # $[\mathbf{A}] = [\mathbf{T}] + [\mathbf{O}]$, $[\mathbf{T}]$ includes the tridiagonal part of $[\mathbf{A}]$ #
2:     Solve $[\mathbf{T}]d\mathbf{x}^k = \mathbf{r}^{k-1}$     # Using a tridiagonal or a pentadiagonal matrix solver #
3:     $\mathbf{x}^k = \mathbf{x}^{k-1} + \omega \cdot d\mathbf{x}^k$
4: end for
5: return $\mathbf{x}^{n_i}$
```

Algorithm 4. 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:     $\mathbf{r}^{k-1} = \mathbf{b} - [\mathbf{A}]\mathbf{x}^{k-1}$
3:     $d\mathbf{x}^k = ILJ(n_i, \omega, [\mathbf{P}], \mathbf{r}^{k-1}, \mathbf{0})$ # See Algorithm 3 #
4:     $\mathbf{x}^k = \mathbf{x}^{k-1} + d\mathbf{x}^k$
5: end for
6: return $\mathbf{x}^{n_o}$
```

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 may be 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 $[\mathbf{P}]$ 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 ($[\mathbf{A}]\mathbf{x}$=$\mathbf{b}$) is calculated based on the original matrix $[\mathbf{A}]$ and then in an inner loop (with n_i sweeps), the updates $d\mathbf{x}$ are calculated based on the pre-conditioner matrix $[\mathbf{P}]$. The relaxation factor $\boldsymbol{\omega}$ is used only in the inner loop. The pre-conditioner matrix $[\mathbf{P}]$ is calculated as

$$\begin{cases} \text{if CFL} \le \text{CFLCap} & [\mathbf{P}] = [\mathbf{A}] \\ \text{if CFL} > \text{CFLCap} & [\mathbf{P}] = [\mathbf{A}_{\text{CFLCap}}] \;\text{ or }\; [\mathbf{P}] = [\mathbf{A}] + \dfrac{\left[\widehat{\mathbf{M}}\right]}{\text{CFLCap}\ \Delta t} \end{cases} \tag{41}$$

where $[\mathbf{A}_{\text{CFLCap}}]$ is a Jacobian matrix evaluated at the capped CFL number and $\left[\widehat{\mathbf{M}}\right]$ is the 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 [41].

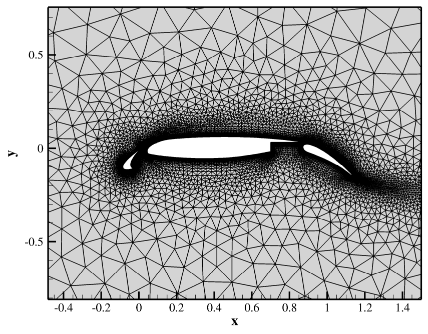
3. Line Generation

As mentioned earlier, the lines should connect strongly coupled unknowns. In diffusion-dominated regions of the flow field, 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 developed in this paper is a generalization of an approach initially developed by Mavriplis [42]. 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. [43] 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. [44]). Figures 3a and 3b show a 30P30N airfoil for which the matrix-based algorithm is used to generate the implicit lines. As seen, the lines are mostly generated in the anisotropic region of the mesh.

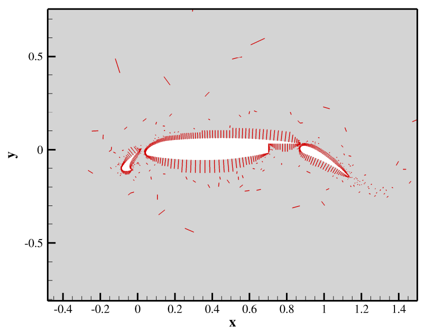
4. Domain Decomposition

The use of implicit-line solvers demands special attention on the grid partitioning. Note that the solution on a single line is an inherently sequential process 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 robustness of the numerical algorithm. To overcome this issue, Mavriplis [42] 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 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 [45]. 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. Figure 4 shows an example of the described mesh partitioning techniques for the 30P30N airfoil.

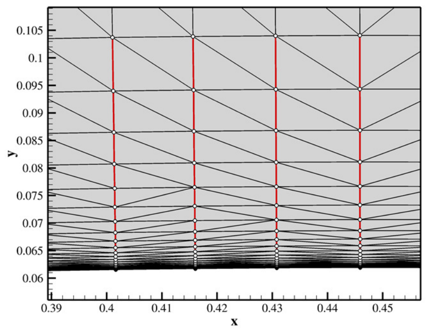
In the present work, for P2 simulations, the implicit line smoother is employed in both P1 and P2 levels. Thus, the partitioning should be done such that none of the lines, form any levels, are cut. As mentioned in Section V.B, in the present work, the P1 partitions are generated by reducing the polynomial order of the P2 partitions. A reasonable choice to obtained the desired lines and partitions for both levels is to generate the P2 lines and then extract the P1 lines from them. This choice is justified using an example. Figures 3c and 3d show the lines that are generated independently when the matrix-based line construction algorithm, with the Laplace operator, is applied on the P1 and P2 levels, respectively. As seen in these figures, the P1 lines can be considered as a subset of the P2 lines. Note that this is mainly because here the Laplace operator has been used to represent the strong connections between nodes.



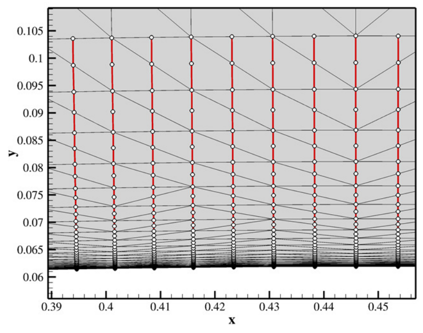
(a) Mesh



(b) Matrix-based lines



(c) P1 lines



(d) P2 Lines

Figure 3. Extracted lines from the stiffness matrix of the Laplace operator for the 30P30N airfoil

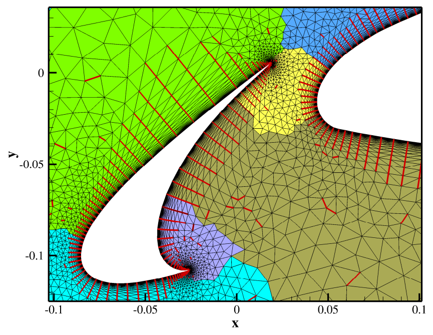
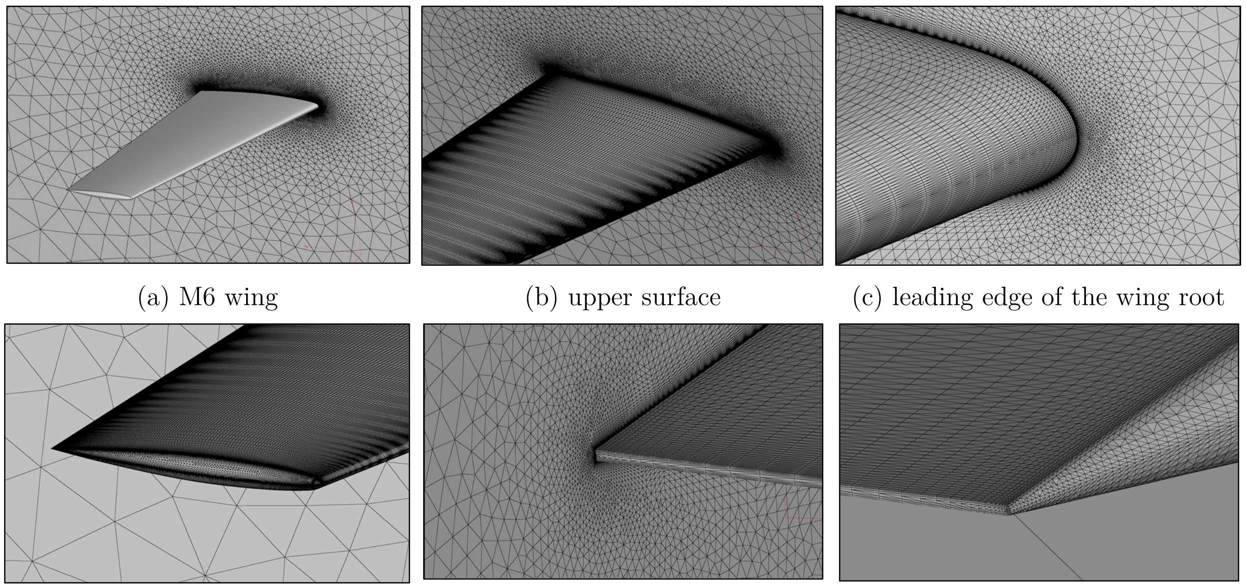


Figure 4. Partitioning the mesh for the 30P30N airfoil in manner that lines are not cut by the partition boundaries.

VII. Numerical Results

A. Instance 1: Turbulent Flow over M6 wing

In the first numerical example, the operation of the developed preconditioned implicit-line Jacobi (PILJ) preconditioner and the non-linear p-multigrid (PMG) algorithm is demonstrated for a three-dimensional test case. For this purpose, a P2 solution for an M6 wing at free stream Mach number of 0.15, Reynolds number of 6.0e+6, and the angle-of-attack of 10.0 degrees is studied. Here, to start the solution process, the PTC algorithm with the PILJ preconditioner is used to solve the problem on the P1 level. Then, this solution is used as an initial guess to start the solution on the P2 level. From this point, two non-linear solution strategies are used to solve the problem on the P2 level. In the first strategy, the PTC algorithm is solely used to iterate on the P2 level until the full convergence is obtained. The second strategy includes two steps: first, the PMG method is used to iterate on the P2 level until a partial convergence is obtained, and then, the nonlinear solution algorithm is switched to PTC to reach the full convergence. In the following plots, the first and second strategies are labeled by PTC and PMG+PTC, respectively.

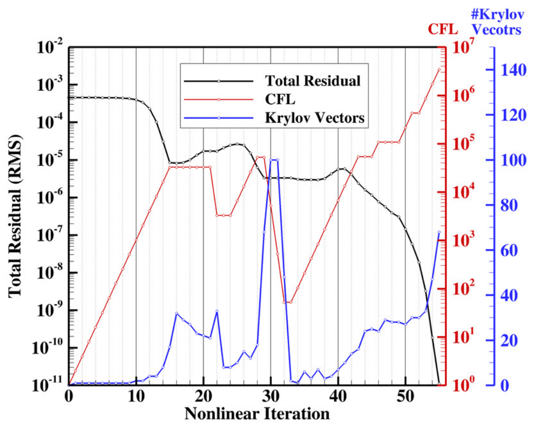

(a) M6 wing (b) upper surface (c) leading edge of the wing root

(d) wing tip (e) trailing edge of the wing root (f) trailing edge of the wing tip

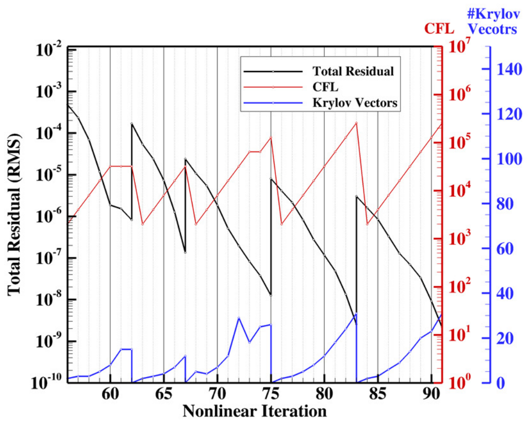
Figure 5. Instance 1, computational mesh for the turbulent flow over the M6 wing.

The computational mesh of this test case, which is shown in Fig. 5, includes 10,283,809 tetrahedral elements and 1,748,112 vertices. For a P2 discretization, this mesh includes 12,100,235 DOFs. In Ref. [16], we showed that a major advantage of the implicit line preconditioner over the ILU(k) preconditioner is the memory efficiency. Meaning that using the implicit line preconditioner the problem can be solved using significantly fewer processors. In this example, however, to show the functioning of this method on large number of processors, the computational mesh is divided into 1440 partitions and each partition is assigned to a processor. It is worth note that although this number of partitions may seem large for 12 million DOFs, the ILU(k) preconditioner, depending on the fill level, may need even more processors to store the factorized Jacobian matrix on the P2 level.

Figure 6a shows the operation of the PTC algorithm during the non-linear iterations to solve the initial problem on the P1 level. This figure plots the total steady-state residual (in black), the number of Krylov vectors (in blue), and the CFL number (in red). As seen, the CFL number starts at one and rises monotonically in the first 15 nonlinear iterations. Then the turbulence model starts to trigger noticeable changes within the flow field and thus the CFL number undergoes few rises and falls. After iteration 33, the CFL number starts to ramp up again and the Newton convergence starts at about iteration 42. The full convergence is obtained at iteration 55, where the CFL number reaches the maximum value of 2 million. In this solution, the tolerance of the linear system has been set to 10^{-3}, and at most 100 Krylov vectors have been allocated for residual minimalization. For the PILJ, the number of outer sweeps, number of inner sweeps, relaxation factor, and CFLCap have been set to 10, 20, 0.2, 500, respectively. These settings are reused for the P2 level except for the CFLCap, which is set to 100.

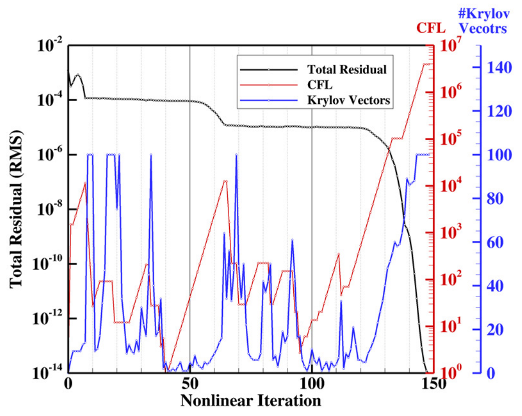


(a)

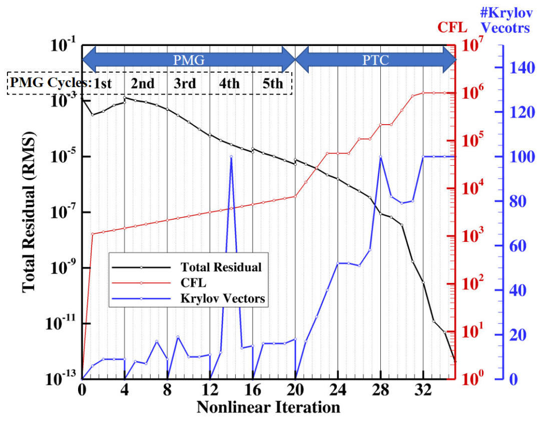


(b)

Figure 6. Instance 1, operation of the PTC algorithm on the P1 level, (a) as the principal nonlinear solver to solve the initial P1 problem, (b) as the coarse grid corrector during the p-multigrid cycles.



(a)

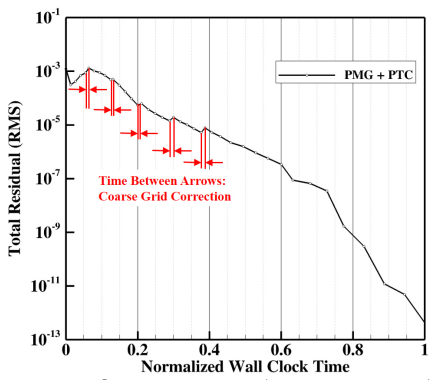


(b)

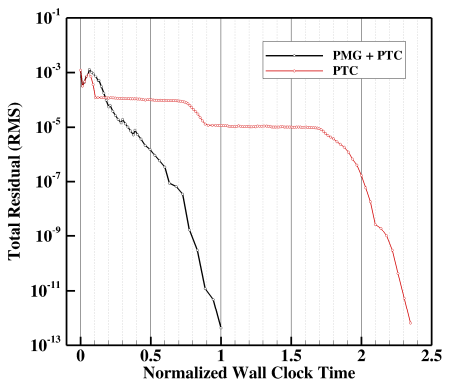
Figure 7. Instance 1, operation of the PTC algorithm on the P2 level, (a) as the principal nonlinear solver, (b) first, as the nonlinear smoother within the p-multigrid cycles, and next as the principal nonlinear solver to reach to the full convergence.

Figures 7a and 7b show the operation of the PTC algorithm on the P2 level. In Fig. 7a, the PTC algorithm is solely used as the principal nonlinear solver to obtain the full convergence, whereas in Fig. 7b, it is first used as the nonlinear smoother within the PMG algorithm, and then as the principle solver to reach the full convergence. During the PMG iterations, $\beta_{CFL} = 1.1$, and during pure PTC iterations, $\beta_{CFL} = 1.5$. In Fig. 7a, it is seen that although the P1 solution has been used as an initial guess, still many nonlinear iterations are required to get to the Newton convergence region. But, Fig. 7b shows that if the pure PTC iterations on the P2 level starts after 5 PMG cycles, the Newton convergence is obtained at significantly fewer nonlinear steps.

Figure 6b shows the operation of the PTC algorithm on the P1 level, when used as the coarse grid corrector (CGC) within the PMG cycles. As seen in this figure, each P1 problem is iterated until its initial nonlinear residual drops by three order of magnitudes. Also, it is seen that all the P1 problems are solved in few number of nonlinear steps. Note that the solutions at different level can be advanced with independent settings. For example, here, the CFL number on the P1 is not the same as the P2 level.



(a)



(b)

Figure 8. Instance 1, convergence of the nonlinear residual on the P2 level versus wall clock, (a) using the PMG and PTC algorithms, (b) comparison of the PMG+PTC and PTC

Figure 8a plots the nonlinear residual of the P2 level versus the normalized wall clock time for the second solution strategy (i.e. PMG+PTC). In this figure, the time intervals shown between the red arrows have been spent on the P1 coarse grid corrections. As seen, at each multigrid cycle, the CGC time is only a fraction of the total time for the cycle. Figure 8b compare the convergence time of both nonlinear solution strategies on the P2 level. Based on this figure, by utilizing the multigrid iterations, the full convergence is obtained more than two times faster.

Based on the above discussion, 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 the FAS multigrid algorithm can shorten the initial phase for high-order ($P \geq 2$) simulations. It should be noted that this concept has been previously studies using second-order finite-volume schemes for both compressible [46] and incompressible flows [47].

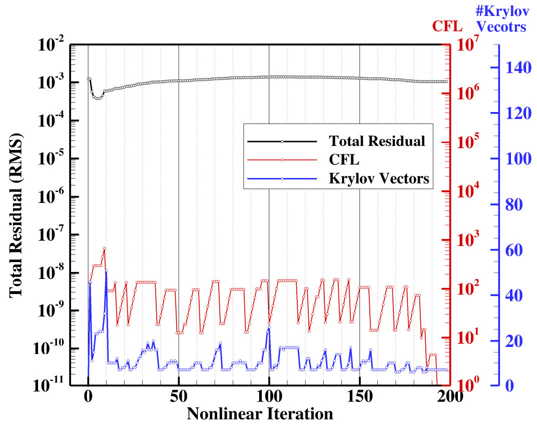


Figure 9. Instance 1, operation of the PTC algorithm with ILU(2) preconditioner on the P2 level. PTC is used as the principal nonlinear solver to solve the problem after starting from a P1 solution.

Finally for this case, the pure PTC algorithm with ILU(2) preconditioner is tested to solve the problem on the P2 level. Like the previous tests, the solution on the P2 level, starts from a converged P1 solution. Figure 9 shows the operation of the PTC algorithm in this test. As seen, the PTC algorithm with ILU(2) preconditioner shows a poor performance and convergence is not obtained. The number of Krylov vectors at each nonlinear iteration is less than the maximum value (i.e. 100). This indicates that the linear convergence has been obtained at all nonlinear steps. Note that the linear tolerance in this example is 10^{-3}. This shows that when the linear system is partially solved, the updates from the ILU preconditioner are less effective than those obtained by the line preconditioner. Clearly, if the linear system is fully solved, the updates from both preconditioners should be identical up to the machine precision.

B. Instance 2: Turbulent Flow over DPW-4 Wing-Body-Tail Geometry

In the second numerical example, the same solution strategies as the first numerical example are employed on a more complex configuration. Here, a P2 solution on a wing-body-tale geometry, taken from 4th AIAA CFD Drag Prediction Workshop (DPW-4) [48], is considered. In this test case, the Mach number is 0.3, the Reynolds number is 5.0e+6, the tail incidence angle is 0.0 degree, and the angle of attack is 4.0 degrees. The computational mesh, which has been generated using the Pointwise software, includes 14,126,323 tetrahedral elements and 2,411,658 vertices. For a P2 discretization, this mesh includes 19,065,261 DOFs. 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 10 shows different parts of the mesh. Compared to the meshes used in the drag prediction workshop, the used mesh here is considered as a coarse mesh. In this example, the mesh is divided into 7200 partitions and each partition is assigned to a processor. As seen in the following, despite the large number of partitions, the developed implicit line preconditioner and nonlinear p-multigrid show a satisfactory convergence behavior, which is not expected when an incomplete factorization preconditioner and a single level nonlinear solution strategy is used. It should be emphasized that this large number of partitions is only used for demonstration of the strong scalability of the developed algorithms.

Like the previous numerical example, to start the solution process, the PTC algorithm with the PILJ preconditioner is used to solve the problem on the P1 level. Then, this solution is used as an initial guess to, and the two previously described nonlinear solution strategies are used to solve the problem on the P2 level. Again, for both strategies on the P2 level, the implicit line method is used as the preconditioner.

Figure 11a shows the operation of the PTC algorithm, when used as the nonlinear solver to solve the initial problem on the P1 level. Figure 10b shows the operation of the PTC algorithm on the P1 level, when used as the coarse grid corrector (CGC) within the PMG cycles. Like the previous example, all the P1 problems are solved in few or reasonable number of nonlinear steps.

Figures 12a and 12b show the operation of the PTC algorithm on the P2 level. In Fig. 12a, the PTC algorithm is solely used as the principal nonlinear solver to obtain the full convergence, and in Fig. 12b, the PTC algorithm is started after 8 PMG cycles. Comparison of these figures results in similar conclusions as the previous numerical example.

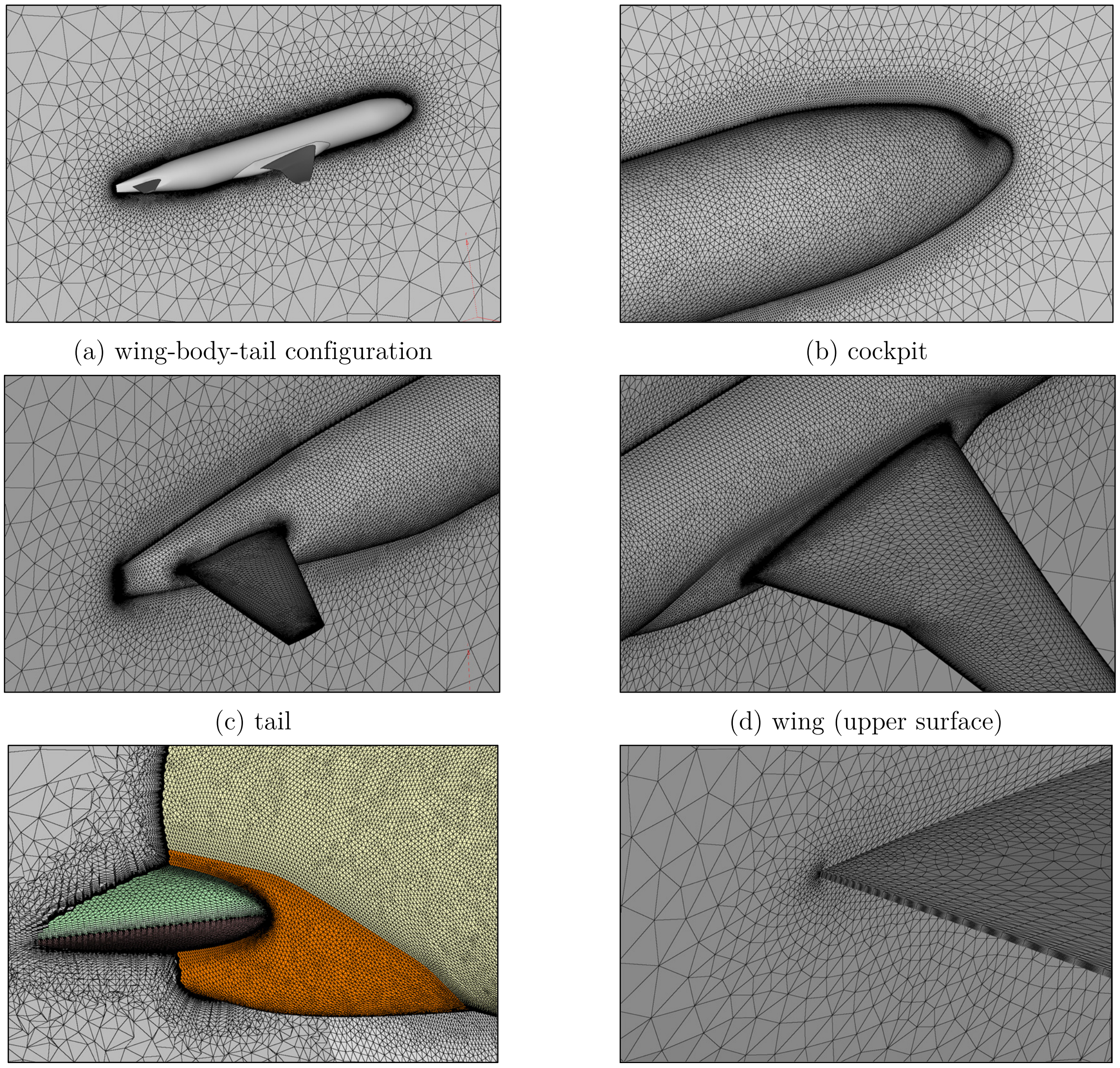
Finally, figures 13a and 13b, compare the convergence of the nonlinear residual and lift coefficient for both nonlinear solution strategies. As seen the PMG+PTC algorithm results in notably faster convergence.

VIII. Conclusion

This paper describes several linear and nonlinear solution techniques that have been developed to build a robust and scalable multilevel solver for stabilized finite-element methods. The solver framework includes two principal nonlinear solvers: an inexact Newton solver based on the pseudo-transient continuation (PTC), and a nonlinear p-multigrid based on the full approximation (FAS) scheme. The solver framework has been developed in an object-oriented manner, and the inexact Newton solver can be used either as principal nonlinear solver, or as a nonlinear smoother for the p-multigrid method. To obtain a robust and scalable preconditioner, a new implicit line smoother has been developed that can be effectively employed on all solution levels. It was shown that for high-order continuous finite-element discretizations, the interconnections of the degrees of freedom on an implicit line form a banded matrix which is wider than tridiagonal, but still can be factorized completely without generating any fill-ins. Also, it has been shown that for high-order solutions, the application of the p-multigrid approach before Newton iterations can drastically reduce the number of nonlinear iterations required to reach a full convergence.

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.

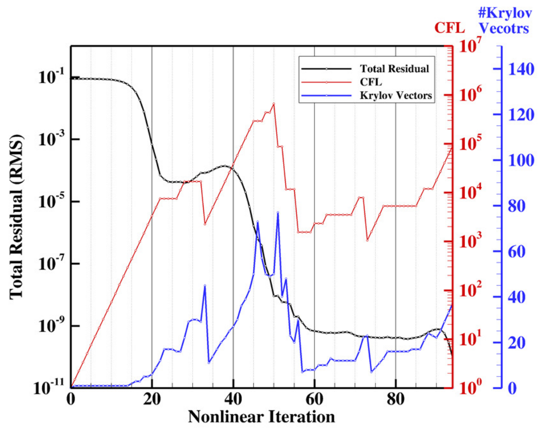


(a) wing-body-tail configuration (b) cockpit

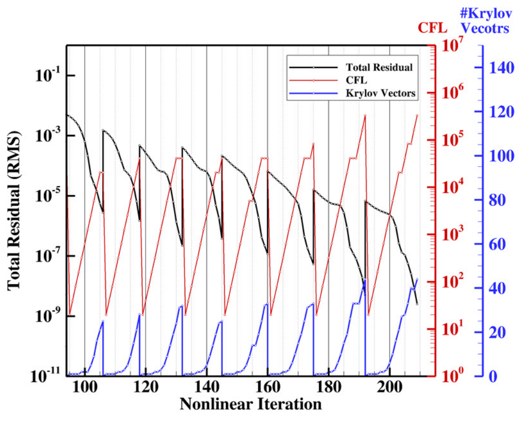
(c) tail (d) wing (upper surface)

(e) boudary layer mesh on the wing (f) trailing edge of the tail

Figure 10. Instance 2, computational mesh for the turbulent flow over the DPW-4 geometry.

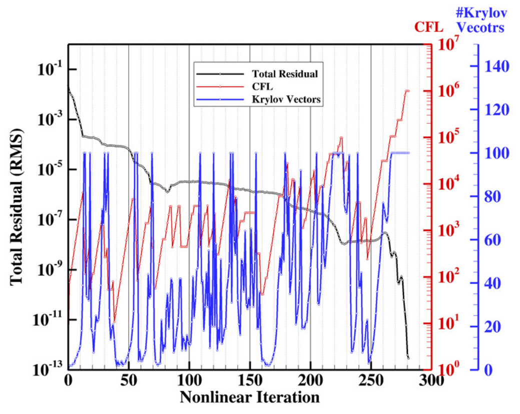


(a)

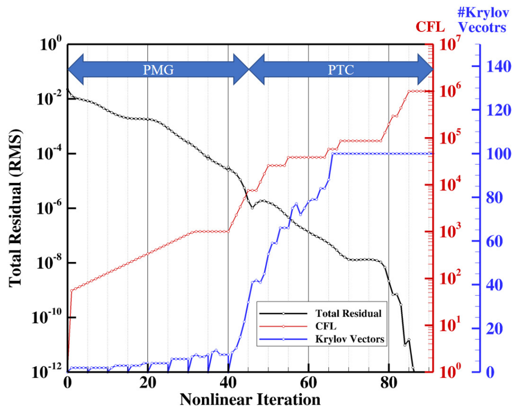


(b)

Figure 11. Instance 2, operation of the PTC on the P1 level, (a) as the nonlinear solver to solve the initial P1 problem, (b) as the coarse grid corrector during the p-multigrid cycles.

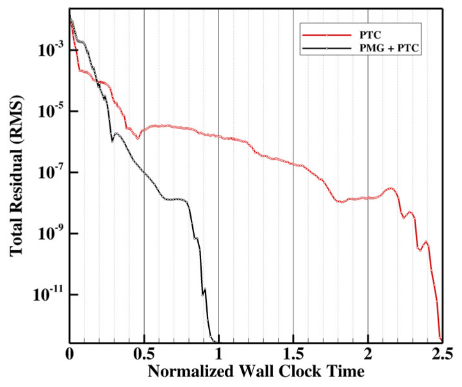


(a)

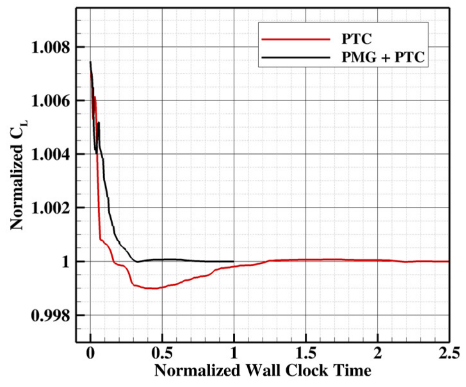


(b)

Figure 12. Instance 2, operation of the PTC on the P2 level, (a) as the principal nonlinear solver, (b) first, as the nonlinear smoother within the p-multigrid cycles, and next as the principal nonlinear solver to reach to the full convergence.



(a)



(b)

Figure 13. Instance 2, Comparison of wall clock times for PTC and PMG+PTC algorithms (a) convergence of the residual, (b) convergence of the lift coefficient.

References

[1] 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.

[2] 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.

[3] K. E. Jansen, C. H. Whiting, and G. M. Hulbert, "A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method," *Computer Methods in Applied Mechanics and Engineering,* vol. 190, pp. 305-319, 2000.

[4] A. K. Karanam, K. E. Jansen, and C. H. Whiting, "Geometry based pre-processor for parallel fluid dynamic simulations using a hierarchical basis," *Engineering with Computers,* vol. 24, pp. 17-26, 2008.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] W. K. Anderson, J. C. Newman, and S. L. Karman, "Stabilized Finite Elements in FUN3D," in *55th AIAA Aerospace Sciences Meeting*, ed: American Institute of Aeronautics and Astronautics, 2017.

[11] M. A. Park and W. K. Anderson, "Spatial Convergence of Three Dimensional Turbulent Flows," in *54th AIAA Aerospace Sciences Meeting*, ed: American Institute of Aeronautics and Astronautics, 2016.

[12] R. S. Glasby and J. T. Erwin, "Introduction to COFFE: The Next-Generation HPCMP CREATE^{TM}-AV CFD Solver," presented at the 54th AIAA Aerospace Sciences Meeting, 2016.

[13] R. S. Glasby and J. T. Erwin, "Comparison Between HPCMP CREATE^{TM}-AV COFFE and Kestrel for Two and Three-Dimensional Turbulent Flow Cases," in *54th AIAA Aerospace Sciences Meeting*, ed: American Institute of Aeronautics and Astronautics, 2016.

[14] J. T. Erwin and R. S. Glasby, "Application of HPCMP CREATE^{TM}-AV COFFE for Three-Dimensional Turbulent Flow Cases," presented at the 54th AIAA Aerospace Sciences Meeting, 2016.

[15] J. T. Erwin, R. S. Glasby, D. L. Stefanski, and S. L. Karman, "Results from HPCMP CREATE^{TM}-AV Kestrel Component COFFE for the 6th Drag Prediction Workshop Cases," presented at the 55th AIAA Aerospace Sciences Meeting, 2017.

[16] B. R. Ahrabi and D. J. Mavriplis, "Scalable Solution Strategies for Stabilized Finite-Element Flow Solvers on Unstructured Meshes," presented at the 55th AIAA Aerospace Sciences Meeting, AIAA Paper 2017-0517, Dallas, TX, January 2017.

[17] C. T. Kelley and D. E. Keyes, "Convergence analysis of pseudo-transient continuation," *SIAM Journal on Numerical Analysis,* vol. 35, pp. 508-523, 1998.

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

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

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

[36] 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.

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

[38] 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.

[39] D. J. Mavriplis, "Multigrid techniques for unstructured meshes," DTIC Document1995.

[40] L. Thomas, "Elliptic problems in linear differential equations over a network: Watson scientific computing laboratory," *Columbia Univ., NY,* 1949.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] D. J. Mavriplis and K. Mani, "Unstructured Mesh Solution Techniques using the NSU3D Solver," presented at the 52nd Aerospace Sciences Meeting, AIAA Paper 2014-0081, 2014.

[47] W. K. Anderson, R. D. Rausch, and D. L. Bonhaus, "Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids," *Journal of Computational Physics,* vol. 128, pp. 391-408, 1996.

[48] 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.