AIAA 2013-2686

Fluid Dynamics and Co-located Conferences
June 24-27, 2013, San Diego, CA
21st AIAA Computational Fluid Dynamics Conference

# Pseudo-transient Continuation, Solution Update Methods, and CFL Strategies for DG Discretizations of the RANS-SA Equations

Marco Ceze* and Krzysztof J. Fidkowski†

*Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA*

**This paper assesses the performance of different time integration strategies for discontinuous Galerkin discretizations of the RANS-SA equations. We consider the pseudo-time continuation method derived from the backward Euler scheme in its constrained and unconstrained versions. Solution update methods based on line-search are proposed and tested in combination with different CFL evolution strategies. We present results for test problems ranging from intermediate to difficult in two and three dimensions.**

## I. Introduction

Turbulent flow problems in external aerodynamics are frequently solved using closure models for the Reynolds-Averaged Navier-Stokes (RANS) equations. This approach allows for the use of coarser meshes than the meshes required for methods that aim for resolving the fine scales of turbulence. Despite the approximations involved in RANS, the resulting model equations are arguably accurate enough for external aerodynamics as discretization errors are the main source of uncertainty in these simulations.

The dominant method for discretizing the RANS models in the aeronautical industry is the finite-volume method (FVM). This method is generally limited to second-order-accurate variants. That is, if the underlying exact solution is smooth the discretization error is expected to decrease quadratically as the mesh is uniformly refined. However, for many problems of practical interest, accuracy requirements are increasingly more stringent and second-order accuracy may not suffice.[1]

In finite-volume and finite-difference schemes, higher orders of accuracy are generally achieved by extending the approximation stencil. This extension does not come free as it interferes with parallelization, hinders the treatment of boundary conditions, and, more importantly, requires time-integration methods with stronger stability properties.

Alternatively, finite-element methods (FEM) can achieve higher orders of accuracy with a fixed, element-wise compact stencil by approximating the flow field using polynomials with local support. The discrete system is coupled by either enforcing solution continuity across element boundaries or by defining unique numerical fluxes between elements. The latter choice yields the discontinuous Galerkin method (DG), which is specifically suited for aerodynamics as it provides stability for convection-dominated problems. Yet, DG methods still present robustness challenges that prevent them from being widely used to solve industry problems. In fact, one of the findings of a recent workshop[a] was that high-order methods are still not as robust as second-order finite-volume methods for problems with turbulence.

An important ingredient for the robustness of second-order finite-volume methods is the advent of limiters. Cockburn *et al.*[2] extended that idea, originally proposed by van Leer,[3] to the discontinuous Galerkin finite-element discretization with Runge-Kutta time stepping. This method is know as RKDG and it preserves monotonicity of mean values. More recently, Kuzmin[4] proposed a form of RKDG that uses a hierarchical derivative limiting approach. This is convenient with Taylor basis functions since the limiter acts directly on the degrees of freedom by a process that is equivalent to $p$-coarsening (lowering the polynomial order) the

*PhD Candidate, AIAA Member
†Assistant Professor, AIAA Member
[a]The 1st International Workshop on High-Order CFD Methods was part of the 50th AIAA Aerospace Sciences Meeting held in January 2012.

cells where the solution is not monotone. Unfortunately, limiting methods are not mature yet in higher-order implicit DG formulations.

Schemes with limiters are robust in time-accurate calculations because they enforce monotonicity restrictions on the discrete solution. However, the steady solution of the discrete residual may not be monotonic.[5] Therefore, enforcing monotonicity can prevent the solver from converging. Conversely, without limiters, numerical oscillations can lead to violations of physicality constraints and also prevent convergence.

Alternatively, artificial dissipation is often used as an attempt to smooth out oscillations. Originally proposed by Von Neumann and Richtmyer[6] for capturing shocks and explored by many others, artificial dissipation methods generally use discontinuity sensors that control even-order derivative terms that damp wave-lengths of the order of the local mesh size. This is achieved by augmenting the residual expression with dissipative terms that are negligible in smooth regions of the flow and are triggered at regions with certain features, such as strong gradients or lack of smoothness. Because of the residual modification, these methods do not prevent Newton-based methods from converging to steady-state. The challenge, however, is to determine the level of artificial dissipation that is adequate for robustness but not too large to destroy solution accuracy. In the finite volume community, this balance was found in a seminal paper by Jameson *et al.*[7] In high-order finite elements discretizations, robust artificial dissipation methods are still being pursued for complex problems.[8, 9]

Full nonlinear convergence of the residual to machine precision levels is not strictly necessary for most flow simulations in the design environment. However, in some practical cases of the aeronautical industry, quantities such as drag and moment vary significantly despite the residual being reduced by several orders of magnitude.[1] Additionally, the theory of error estimation makes use of Galerkin orthogonality which is only theoretically valid if the discrete residual is zero. Therefore, the development of solution advancement methods that robustly drive the residual to *zero* (up to machine precision) is an important step in increasing the prevalence of high-order methods in the aeronautical industry.

This work compares a pseudo-transient continuation method with its physicality-constrained counterpart[10] in a discontinuous Galerkin framework (Section II). The methods are combined with different state update algorithms (Section V) and CFL strategies (Section III). Section IV shows how the constraints are incorporated in the solution path and Section VI describes the scaling of the discretized turbulence model equations. The results for a range of challenging flow problems are presented in Section VII and the concluding remarks are presented in Section VIII.

## II. Spatial Discretization

The Reynolds-Averaged Navier-Stokes (RANS) equations with the Spalart-Allmaras (SA) turbulence model are written in their compact, conservative form as

$$\partial_t \mathrm{u}_s + \partial_i \mathcal{C}_{is}(\mathbf{u}) - \partial_i \mathcal{D}_{is}(\mathbf{u}) = \mathcal{S}_s(\mathbf{u}), \tag{1}$$

where $\mathcal{C}_{is}$ and $\mathcal{D}_{is}$ are the convective and diffusive fluxes respectively, $\mathcal{S}_s$ is the SA source term, $i \in [1, .., \dim]$ indexes the spatial dimensions, and $s$ indexes the equations of conservation of mass, momentum, energy, and turbulent viscosity. Accordingly, the state vector is denoted by $\mathbf{u} = [\rho, \rho v_i, \rho E, \rho\tilde{\nu}]^T$, where $\rho$ is the density, $v_i$ are the spatial components of the velocity, $E$ is the specific total energy, and $\tilde{\nu}$ is the working variable for the SA model.

The discontinuous Galerkin (DG) spatial discretization of the flow equations approximates the solution in a space $\mathcal{V}^{H,p}$ of piecewise polynomials of degree $p$ with local support on each element $\kappa^H \in T^H$, where $T^H$ is the set of elements resulting from a subdivision of the spatial domain. The resulting weak form reads:

$$\partial_t(\mathbf{u}^{H,p}, \mathbf{w}^{H,p}) + \mathbb{R}(\mathbf{u}^{H,p}, \mathbf{w}^{H,p}) = 0 \qquad \mathbf{w}^{H,p} \in \mathcal{V}^{H,p}, \tag{2}$$

where $(\cdot,\cdot)$ denotes an inner product and $\mathbb{R}(\mathbf{u}^{H,p}, \mathbf{w}^{H,p})$ is a weighted residual statement that includes source, convective, and diffusive terms.

Here, we adopt Oliver's[11] modifications to the original SA model.[12] These modifications ensure stability of the model at negative $\tilde{\nu}$ and they are specifically suited for discontinuous Galerkin discretizations.

The Riemann flux involved in the convective term is approximated with Roe's[13] solver in which the SA working variable is transported as a conserved scalar. The diffusion term is discretized using the second form of Bassi & Rebay[14] (BR2) and the SA source term is discretized according to Oliver.[11]

The discrete system is obtained by expanding the components the state $\mathbf{u}^{H,p}$ and the weight functions $\mathbf{w}^{H,p}$ in terms of the basis functions $\phi^{H,p}(\mathbf{x}) \in \mathcal{V}^{H,p}$. The result has the form:

$$\mathbf{M}\mathrm{d}_t\mathbf{U} = -\mathbf{R}(\mathbf{U}), \tag{3}$$

where $\mathbf{U}$ is the discrete state, $\mathbf{R}$ is the discrete residual operator and $\mathbf{M}$ is the block diagonal mass matrix that corresponds to the volume integral of basis function products on each element in the mesh.

## A. Physicality constraints

The flow field is subject to physicality constraints that are not guaranteed to be satisfied as the discretized equations only enforce conservation.

### 1. *Thermodynamic realizability*

The thermodynamic realizability constraints are:

$$\begin{aligned} \frac{p(\mathbf{u}^{H,p}(t,\mathbf{x}))}{p_\infty} &> 0, \\ \frac{\rho(\mathbf{u}^{H,p}(t,\mathbf{x}))}{\rho_\infty} &> 0, \end{aligned} \tag{4}$$

where $p_\infty$ and $\rho_\infty$ refer to free-stream pressure and density, respectively. These quantities are included here only for non-dimensional convenience and they clearly do not alter the positivity constraints. Note that $\rho$ is a conserved variable and, therefore, its extrema match the extrema of the corresponding position in the conserved state $\mathbf{u}^{H,p}$. Pressure, however, does not have this property. In fact, its curvature along a spatial direction $\zeta$ is given by

$$\frac{\partial^2 p}{\partial \zeta^2} = \left(\frac{\partial \mathbf{u}^{H,p}}{\partial \zeta}\right)^T \underbrace{\frac{\partial^2 p}{\partial \mathbf{u}^{H,p} \partial \mathbf{u}^{H,p\,T}}}_{\mathcal{H}_p} \left(\frac{\partial \mathbf{u}^{H,p}}{\partial \zeta}\right) + \left(\frac{\partial p}{\partial \mathbf{u}^{H,p}}\right)^T \frac{\partial^2 \mathbf{u}^{H,p}}{\partial \zeta^2}. \tag{5}$$

The eigenvalues of the Hessian of the pressure with respect to the state are

$$\mathrm{eig}_s(\mathcal{H}_p) = \begin{cases} 0, & \text{for } s = 1,2, \\ -\dfrac{\gamma-1}{\rho}, & \text{for } \dim > 1,\ s = 3 \to \dim + 1, \\ -\dfrac{(\gamma-1)(1+v_i v_i)}{\rho}, & \text{for } s = \dim + 2. \end{cases} \tag{6}$$

Note that for a linear distribution of state quantities along $\zeta$, the only local extremum possible in pressure between two points is a maximum since the eigenvalues of $\mathcal{H}_p$ are non-positive. Therefore, when the state is linearly distributed, we only need to check the pressure constraint at the end points. However, it is difficult to ensure positivity for generic state distributions because the second term in Eqn. 5 involves the sum of positive and negative terms.

### 2. *RANS*

Physical intuition indicates that eddy viscosity should be constrained similarly to pressure and density, *i.e.* $\nu_t > 0$. Oliver's modifications impose this constraint by allowing, we found that enforcing positive total viscosity,

$$\frac{\nu(\mathbf{u}^{H,p}(t,\mathbf{x})) + \nu_t(\mathbf{u}^{H,p}(t,\mathbf{x}))}{\nu(\mathbf{u}^{H,p}(t,\mathbf{x}))} > 0, \tag{7}$$

in the solution path helps convergence in many flow cases. Similarly to the thermodynamic constraints, the physical kinematic viscosity in the denominator of Eqn. 7 makes the constraint non-dimensional.

## III. Pseudo-transient Continuation

Since we are interested in the steady-state solution of the flow equations, high-accuracy is not required for discretizing the unsteady term of Eqn. 3. Instead, stability is the main attribute which makes backward Euler an attractive choice. The fully-discrete form of Eqn. 3 is then:

$$\mathbf{M}\frac{1}{\Delta t}(\mathbf{U}^{n+1} - \mathbf{U}^{n}) + \mathbf{R}(\mathbf{U}^{n+1}) = 0, \tag{8}$$

where $\mathbf{M}$ is the mass matrix and $n$ indexes the time step.

For steady calculations, the residual at the future state in Eqn. 8 is expanded about the current state and the steps in the iterative procedure require linear solves for the update $\Delta\mathbf{U}^k$,

$$\left(\mathbf{M}\frac{1}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}\right)\Delta\mathbf{U}^k = -\mathbf{R}(\mathbf{U}^k), \tag{9}$$

where $k$ is used for the nonlinear iteration number to distinguish the method from the backward Euler case. Note that for $\Delta t \to \infty$ the iterative procedure of Eqn. 9 reduces to Newton's root-finding method.

The linearization of the residual operator involves simplifications due to non-differentiable terms in numerical flux functions and artificial dissipation sensors. Additionally, the sparse structure of the linear system given in Eqn. 9 depends on the type of spatial scheme used for $\mathbf{R}$, and an appropriate choice of iterative solver and preconditioner must be made. In this work, a restarted Generalized Minimal Residual (GMRES) linear solver,[15,16] aided by a line-Jacobi preconditioner,[17] solves the linear system at each step. The DG discretization described in Section II produces a residual Jacobian that is block-sparse, that is, degrees of freedom in a element are coupled only to degrees of freedom in neighbor elements. Within each block, sparsity may exist, for certain choices of basis functions, but we do not take advantage of such sparsity.

In the first stages of calculations initialized by states that do not satisfy all boundary conditions, strong transients are observed due to the propagation of boundary information into the domain. To alleviate those transients and to avoid robustness problems, small time steps are used in an attempt to make the solution follow a physical path. This causes a diagonal dominance in the coefficient matrix in Eqn. 9 and makes the calculation closer to time-accurate if $\Delta t$ does not vary spatially. As an alternative to global time stepping, element-wise time steps can be used by setting a global CFL number defined as:

$$\text{CFL} = \frac{\lambda_{\max}\Delta t_{\kappa^H}}{L_{\kappa^H}}, \tag{10}$$

where $\lambda_{\max}$ is the maximum wave speed and $L_{\kappa^H}$ is a measure of element size, *e.g.* hydraulic diameter.

At each iteration, $k$, the flow state vector $\mathbf{U}^k$ is updated with $\Delta\mathbf{U}^k$. For robustness purposes, an under-relaxation parameter, $\omega^k$, is used to ensure a physical solution at the next iteration (Eqn. 11).

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \omega^k\Delta\mathbf{U}^k \tag{11}$$

### A. CFL evolution strategies

In a pseudo-transient continuation method, the continuation parameter is the CFL number. Hence, a strategy must be chosen to evolve the CFL from its initial value to a large value such that Eqn. 9 becomes Newton's method and the state approaches the steady solution.

#### Switched Evolution Relaxation - SER

Many strategies for evolving the time step are available.[18,19] Amongst them, a widely used strategy is the *Switched Evolution Relaxation* (SER) method proposed by Mulder and van Leer.[20] The general idea of SER is to change the time step or the CFL number based on a measure of convergence which is inferred from the reduction in a residual norm between consecutive iterations. Typically, the $L_2$ norm is used. The algorithm reads as follows:

$$\text{CFL}^k = \min\left(\text{CFL}^{k-1}\frac{|\mathbf{R}^{k-1}|_{L_2}}{|\mathbf{R}^k|_{L_2}}, \text{CFL}_{\max}\right). \tag{12}$$

SER is an effective time step evolution strategy. However, the physicality constraints are verified after the direction $\Delta\mathbf{U}$ is computed and the relaxation parameter $\omega$ in Eqn. 11 must be such that the updated state is physical. In the event of $\omega$ becoming too small and the time step not changing significantly, a contingency plan needs to be designed so that the direction $\Delta\mathbf{U}$ changes. This is discussed in Section V.

Exponential progression with under-relaxation - EXPur

Alternatively, the CFL evolution can be based on the value of the under-relaxation parameter. Specifically, the CFL increases by a factor $\beta > 1$ if a full update ($\omega = 1$) happened in the previous step of the solver. On the other end, if $\omega < \omega_{\min}$ the CFL is reduced by multiplying it by $\kappa < 1$ and the solver step is repeated. The relaxation factor is limited such that the solution stays physical at selected limit points of the interpolated field $\mathbf{u}^{H,p}$. The methods for computing the under-relaxation factor are described in Section V.

This strategy accounts for the physical feasibility constraints for the next update. However, it is an indirect way of avoiding non-physical states in the flow field since the direction $\Delta\mathbf{U}^k$ may still produce states that are closer to becoming non-physical even at the minimum CFL. In particular, this is observed in highly under-resolved meshes.

The CFL strategy is summarized below:

$$\mathrm{CFL}^{k+1} = \begin{cases} \beta \cdot \mathrm{CFL}^k & \text{for } \beta > 1 \quad \text{if} \quad \omega^k = 1 \\ \mathrm{CFL}^k & \qquad\qquad \text{if} \quad \omega_{\min} < \omega^k < 1 \\ \kappa \cdot \mathrm{CFL}^k & \text{for } \kappa < 1 \quad \text{if} \quad \omega^k < \omega_{\min} \end{cases} . \tag{13}$$

Here, the parameters are set to: $\omega_{\min} = 0.01$, $\beta = 1.05 \leftrightarrow 2.0$, and $\kappa = 0.1$.

Residual Difference Method - RDM

This CFL evolution strategy is based on a method described in Ref. [18] and it is a blend of EXP and SER. Similarly to SER, this strategy monitors the solution evolution and increases/decreases the CFL when the residual norm is reduced/increased. However, the maximum change in CFL is limited from above by a factor $\beta$. The algorithm reads as follows:

$$\mathrm{CFL}^k = \min\left(\mathrm{CFL}^{k-1} \cdot \beta^{\frac{|\mathbf{R}^{k-1}|_{L_2} - |\mathbf{R}^k|_{L_2}}{|\mathbf{R}^{k-1}|_{L_2}}}, \mathrm{CFL}_{\max}\right) \quad \text{for } \beta > 1. \tag{14}$$

Note that $\frac{\mathrm{CFL}^k}{\mathrm{CFL}^{k-1}} \leq \beta$, where the equality corresponds to $\mathbf{R}^k$ vanishing completely. A monotonic variant of RDM is obtained by setting the exponent in Eqn. 14 to zero when $|\mathbf{R}^k|_{L_2} > |\mathbf{R}^{k-1}|_{L_2}$. This variant will be referred as mRDM in the remainder of the text.

## B. Optimization aspect of PTC

Assume the coefficient matrix in Eqn. 9 is real and non-singular and the update direction $\Delta\mathbf{U}^k$ is not zero. Multiplying the left-hand side of Eqn. 9 by its transpose gives:

$$\Delta\mathbf{U}^{k^T} \underbrace{\left(\mathbf{M}\frac{1}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}\right)^T}_{A^T} \underbrace{\left(\mathbf{M}\frac{1}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}\right)}_{A} \Delta\mathbf{U}^k = -\Delta\mathbf{U}^{k^T} \underbrace{A^T\mathbf{R}(\mathbf{U^k})}_{\frac{\partial f}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}} > 0. \tag{15}$$

Therefore, $\Delta\mathbf{U}$ is a descent direction for the scalar function $f(\tilde{\mathbf{U}})$ defined by its gradient in the right-hand side of Eqn. 15.

Now, consider the unsteady residual,

$$\mathbf{R}_t(\tilde{\mathbf{U}}) = \mathbf{M}\frac{1}{\Delta t}(\tilde{\mathbf{U}} - \mathbf{U}^k) + \mathbf{R}(\tilde{\mathbf{U}}), \tag{16}$$

where $\tilde{\mathbf{U}}$ is a trial future state. By taking one step of Newton's method for $\mathbf{R}_t(\tilde{\mathbf{U}}) = 0$ with initial guess $\tilde{\mathbf{U}}^0 = \mathbf{U}^k$, we conclude that,

$$f(\tilde{\mathbf{U}}) = \frac{1}{2}|\mathbf{R}_t(\tilde{\mathbf{U}})|^2_{L_2} = \frac{1}{2}\mathbf{R}_t(\tilde{\mathbf{U}})^T\mathbf{R}_t(\tilde{\mathbf{U}}). \tag{17}$$

This is verified by differentiating Eqn. 17 and setting $\tilde{\mathbf{U}} = \mathbf{U}^k$. Consequently, there is a trial state $\tilde{\mathbf{U}}$ along the direction $\Delta\mathbf{U}^k$ such that $f(\tilde{\mathbf{U}}) < f(\mathbf{U}^k)$.

## IV. Incorporating constraints

The minimization character of the PTC method motivates the use of constraint handling techniques from the optimization field. Non-physical states (*e.g.* negative pressure) can lead to instability,[21] therefore we need to keep the iterates within the physical region of the solution space. Interior penalty methods[22] are attractive because of their simplicity and efficiency in acknowledging feasibility constraints in the solution path. These methods augment a scalar objective function with a term that tends to infinity as the solution path approaches a feasibility boundary creating a repelling effect with respect to prohibited regions of the domain.

A different approach for incorporating constraints into pseudo-transient methods is proposed by Kelley *et al.*[23] Their approach involves a step that projects the state into the feasible domain after each non-linear iteration and the fundamental difference between their method and the method we propose here is that we incorporate the constraints when computing the solution update.

### A. Scalar penalization

A simple way of incorporating the realizability constraints in the solution path is to formulate an optimization problem that reads:

$$\boxed{\begin{array}{ll} \text{minimize:} & f(\mathbf{U}) = |\mathbf{R}_t(\mathbf{U})|^2_{L_2} \\ \text{by varying:} & \mathbf{U}, \text{ and } \Delta t \to \infty \\ \text{subject to:} & c_i(\mathbf{u}^{H,p}(t,\mathbf{x})) > 0. \quad \forall \mathbf{x} \in D \end{array}}$$

The constraints, $c_i(\mathbf{u}^{H,p}(t,\mathbf{x})) > 0$, are dependent on the equations being solved. In this work, we consider the RANS-SA constraints presented in Section A.

An interior penalty method handles the constraints by augmenting the objective function, $f(\mathbf{U})$, with an inverse-barrier function of $c_i(\mathbf{u}^{H,p}(t,\mathbf{x}))$. Since the constraints are applied to a functional representation of the state, an integral of the inverse barrier would have to be evaluated in order to enforce the constraints everywhere in the domain. We approximate this integral by using a quadrature rule and the penalty function is written as,

$$\mathbb{P}(\mathbf{U},\mu_{\mathrm{P}}) = \mu_{\mathrm{P}} \sum_{\kappa^H \in T^H} \sum_{i}^{N_c} \sum_{q}^{N_q} \frac{w_q}{c_i(\mathbf{u}^{H,p}(x_q))}, \tag{18}$$

where $N_q$ is the number of quadrature points $x_q$ with weights $w_q$, $N_c$ is the number of constraints indexed by $i$, and $\mu$, in this context, is a scalar penalty factor. Note that $\mathbb{P}$ in Eqn. 18 tends to infinity as the constraints approach zero from the positive side. The augmented function is then given by:

$$g(\mathbf{U},\mu_{\mathrm{P}}) = f(\mathbf{U}) + \mathbb{P}(\mathbf{U},\mu_{\mathrm{P}}). \tag{19}$$

The main idea of the interior penalty method is to solve a sequence of the optimization problems for diminishing penalty factors, $\mu^{j+1} < \mu^j$. For each optimization problem $j$, Newton's approach can be used to compute a search direction for a minimizer of $g(\mathbf{U},\mu^j)$,

$$\mathcal{H}_g(\mathbf{U}^k,\mu^j_{\mathrm{P}})\Delta\mathbf{U}^k = -\frac{\partial g}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k,\mu^j_{\mathrm{P}}}, \tag{20}$$

where $\mathcal{H}_g$ is the Hessian of the augmented objective function,

$$\mathcal{H}_g(\mathbf{U},\mu_{\mathrm{P}}) = \underbrace{\frac{\partial \mathbf{R}_t}{\partial \mathbf{U}}\left(\frac{\partial \mathbf{R}_t}{\partial \mathbf{U}}\right)^T + \frac{\partial^2 \mathbf{R}_t}{\partial \mathbf{U}\partial \mathbf{U}^T}\mathbf{R}_t(\mathbf{U})}_{\mathcal{H}_f(\mathbf{U})} + \mathcal{H}_{\mathbb{P}}(\mathbf{U},\mu_{\mathrm{P}}) \tag{21}$$

and the Hessian of the penalty function, $\mathcal{H}_{\mathbb{P}}$ is a block diagonal matrix due to the local support of $\mathbf{u}^{H,p}$.

It is customary in nonlinear least-squares problems to use the Gauss-Newton[23] approximation to the Hessian:

$$\mathcal{H}_g \approx \tilde{\mathcal{H}}_g(\mathbf{U}, \mu_{\mathrm{P}}) = \frac{\partial \mathbf{R}_t}{\partial \mathbf{U}} \left( \frac{\partial \mathbf{R}_t}{\partial \mathbf{U}} \right)^T + \mathcal{H}_{\mathbb{P}}(\mathbf{U}, \mu_{\mathrm{P}}). \tag{22}$$

In spite of the regularization effect of $\mathcal{H}_{\mathbb{P}}$ and $\mathbf{M}\frac{1}{\Delta t}$ due to their diagonal dominance, in general, the simplified full Hessian in Eqn. 22 is very ill-conditioned due to the squaring of the residual Jacobian, and solving the linear system in Eqn. 20 is very computationally expensive. This is the disadvantage of having a scalar function in Eqn. 17 which permits a simple penalization for enforcing the constraints. Additionally, factorizing $\tilde{\mathcal{H}}_g$ would generally require the explicit construction of that matrix which can be computationally intensive even for small problems. For this reason, the pure optimization approach is inadequate for any realistic problem.

## B. Vector penalization

As an alternative to the scalar penalization, we proposed in Ref. 10 augmenting the residual with a penalty vector to account for the constraints:

$$\mathbf{R}_p(\mathbf{U}) = \mathbf{R}(\mathbf{U}) + \mathbf{P}(\mathbf{U}, \mu_{\mathrm{P}}). \tag{23}$$

In order to have the repelling effect with respect to non-feasible regions of the domain, the penalization vector $\mathbf{P}$ must have a positive projection on the direction of the residual vector $\mathbf{R}$. To satisfy this requirement, we define the penalization vector as:

$$\mathbf{P}(\mathbf{U}, \mu_{\mathrm{P}}) = \Phi(\mathbf{U}, \mu_{\mathrm{P}})\ \mathbf{R}(\mathbf{U}), \tag{24}$$

where $\mu_{\mathrm{P}}$ is a penalty factor and $\Phi$ is a diagonal matrix of the same size as the residual Jacobian with the elemental penalties $\mathbb{P}_{\kappa^H}$ for each row corresponding to an element $\kappa^H$.

$$\Phi_{ij}(\mathbf{U}, \mu_{\mathrm{P}}) = \begin{cases} \mu_{\mathrm{P}}\ \mathbb{P}_{\kappa^H}(\mathbf{U}) & \text{if } i = j \in \mathrm{dof}(\kappa^H) \\ 0 \end{cases} \tag{25}$$

Note that $j \in \mathrm{dof}(\kappa^H)$ denotes the degrees of freedom, in global ordering, pertinent to $\kappa^H$. The elemental penalty is given by:

$$\mathbb{P}_{\kappa^H}(\mathbf{U}) = \sum_i^{N_i} \sum_q^{N_q} \frac{w_q}{c_i(\mathbf{u}^{H,p}(\mathbf{x}_q))}. \tag{26}$$

Equation 26 involves a summation over quadrature points, $\mathbf{x}_q$, that lie inside $\kappa^H$, with weights $w_q$. This summation corresponds to integrating the inverse barrier function in a reference element with unitary volume.

Note that the projection of $\mathbf{P}$ – as defined in Eqn. 24 – onto the residual vector is always positive for non-zero $\mathbf{R}$ since the elemental penalties are strictly positive in the feasible domain, *i.e.*, physical states.

A root of the residual operator corresponds to a root of $\mathbf{R}_p$, so that the steady-state solution is independent of the values of the elemental penalties. We emphasize that the objective of this method is to change the path to the solution, not the solution itself. By applying the pseudo-transient continuation procedure (Eqn. 9) to $\mathbf{R}_p$ we are including physicality constraints in the solution path from the initial condition to steady state. The update direction along that path at step $k$ satisfies

$$\left( \underbrace{(\mathbf{I} + \Phi^k)^{-1} \frac{\mathcal{M}}{\Delta t}}_{a} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k} + \underbrace{(\mathbf{I} + \Phi^k)^{-1} \left( \frac{\partial \Phi}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k} \mathbf{R}(\mathbf{U}^k) \right)}_{b} \right) \Delta \mathbf{U}^k = -\mathbf{R}(\mathbf{U}^k), \tag{27}$$

where $\mathbf{I}$ is the identity matrix and $\Phi^k = \Phi(\mathbf{U}^k, \mu_{\mathrm{P}}^k)$. The equation above is derived by the substituting $\mathbf{R}_p$ into Eqn. 9 and by separating the terms such that the unpenalized residual, $\mathbf{R}$, is on the right hand-side.

This adds the implementation convenience of simply adding entries to the coefficient matrix of the linear systems solved at each step $k$.

The terms "$a$" and "$b$" in Eqn. 27 are block diagonal for the DG method in this work. Additionally, the elemental CFL number gets amplified by $(1 + \mu_{\mathrm{P}}\ \mathbb{P}_{\kappa^H})$ as $\mathbf{I} + \Phi^k$ is a diagonal matrix. In the limit of an infinite time step, the solution path seeks a minimum of $|\mathbf{R}_p|_{L_2}$. Similarly, the time continuation term "$a$" vanishes at elements where the solution approaches a non-physical region while the penalization term "$b$" does not vanish because the function value of inverse barrier penalties (Eqn. 26) tends to infinity at a slower rate than the magnitude of its derivative. In the remainder of the text, we will refer to the method in Eqn. 27 as *Contrained Pseudo-transient Continuation* (CPTC).

The penalty factor is evolved using a form of SER:

$$\mu_{\mathrm{P}}^{k+1} = \mu_{\mathrm{P}}^{k} \frac{1 + \mu_{\mathrm{P}}^{k} \max(\mathbb{P}_{\kappa^H}(\mathbf{U}^k))}{1 + \mu_{\mathrm{P}}^{k-1} \max(\mathbb{P}_{\kappa^H}(\mathbf{U}^{k-1}))}. \tag{28}$$

The purpose of this evolution strategy for $\mu$ is to make the solver acknowledge the presence of a feasibility constraint by increasing its repelling effect as the solution path goes towards a non-physical state anywhere in the domain. This latter point is due to the use of the max function over the elements in Eqn. 28. Conversely, if the solution path is moving away from a feasibility boundary the repelling effect decreases.

The penalty factor is initialized such that $(1 + \mu\ \max(\mathbb{P}_{\kappa^H})) = \mathcal{O}(1)$ but $\mu > 0$. This keeps the pseudo-transient term active and alleviates the initial solution transients while helping the spectral conditioning of initial linear systems. For all the results in this work, $\mu$ is initialized according to:

$$1 + \mu^0 \max(\mathbb{P}_{\kappa^H}(\mathbf{U}^0)) = 10^{0.25}. \tag{29}$$

The CPTC method is summarized in Algorithm 1. The unconstrained PTC follows a similar algorithm, where the steps related to the penalty factor (steps 3 and 10) are ignored and the update direction (step 6) is computed using Eqn. 9. For all the cases presented here, the CFL is reduced by a factor $\kappa = 0.1$ when the under-relaxation factor is below $\omega_{\min} = 0.01$. At that point the state is reverted to a safe state stored when a full update occurs.

**Algorithm 1** Constrained PTC

```
 1: Choose initial CFL and its evolution strategy (Section A)
 2: Set a residual tolerance, $\varepsilon_{\mathrm{res}}$
 3: Initialize $\mu$ according to Eqn. 29
 4: Initialize $\mathbf{U}_{\mathrm{safe}}$ to initial condition
 5: while $|\mathbf{R}(\mathbf{U}^k)| > \varepsilon_{\mathrm{res}}$, $k <$ maximum iterations do
 6:     Compute $\Delta\mathbf{U}^k$ by solving Eqn. 27 using GMRES
 7:     Compute under-relaxation parameter $\omega^k$ (Section V)
 8:     if $\omega^k \geq \omega_{\min}$ then
 9:         $\mathbf{U}^{k+1} \leftarrow \mathbf{U}^k + \omega^k \Delta U^k$.
10:         Evolve $\mu$ using Eqn. 28
11:         Evolve CFL with chosen strategy
12:         if $\omega^k = 1$ then
13:             $\mathbf{U}_{\mathrm{safe}} \leftarrow \mathbf{U}^{k+1}$          ▷ Store a safe state
14:         end if
15:     else
16:         $\mathrm{CFL}^k \leftarrow \kappa\ \mathrm{CFL}^k$ for $\kappa < 1$
17:         $\mathbf{U}^{k+1} \leftarrow \mathbf{U}_{\mathrm{safe}}$          ▷ Revert to last safe state
18:         Return to step 6
19:     end if
20:     $k \leftarrow k + 1$
21: end while
```

## V. Solution update

The solution update methods described here use two main ingredients. First, they require interpolating the state, $\mathbf{u}^{H,p}$, and its update, $\Delta\mathbf{u}^{H,p}$, at certain points, $\mathbf{x}_m$. This involves evaluating the basis functions at

$\mathbf{x}_m$ and use the discrete vectors $\mathbf{U}$ and $\Delta\mathbf{U}$ to yield the field representations, $\mathbf{u}^{H,p}$ and $\Delta\mathbf{u}^{H,p}$. The second ingredient is an update limiter that restricts the changes in primitive variables to a maximum fraction, $\eta_{\max}$, of the current values. This procedure is described for the RANS-SA equations in Algorithm 2.

**Algorithm 2** Limit physical update

```
 1: Given $\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$, $\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$, and a fraction $\eta_{\max} < 1$
 2: $\omega_{\kappa^H} \leftarrow 1$
 3: for all $x_m \in \kappa^H$ do
 4:     $\omega_\rho \leftarrow 1$                                                  ▷ $\omega_\rho$ is the step size for density
 5:     $\rho_m = \rho(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$                                  ▷ Current density at $\mathbf{x}_m$
 6:     $\tilde{\rho}_m = \rho(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H} + \omega_\rho\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$          ▷ Trial density at $\mathbf{x}_m$
 7:     if $\tilde{\rho}_m$ is not within $\eta_{\max}$ of $\rho_m$ then
 8:         Reduce $\omega_\rho$ such that $\tilde{\rho}_m$ is within $\eta_{\max}$ of $\rho_m$
 9:     end if
10:     $\omega_p \leftarrow \omega_\rho$                                                 ▷ $\omega_p$ is the step size for pressure
11:     $p_m = p(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$                                   ▷ Current pressure at $\mathbf{x}_m$
12:     $\tilde{p}_m = p(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H} + \omega_p\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$          ▷ Trial pressure at $\mathbf{x}_m$
13:     while $\tilde{p}_m$ is not within $\eta_{\max}$ of $p_m$ do
14:         $\omega_p \leftarrow \frac{\omega_p}{2}$
15:         $\tilde{p}_m = p(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H} + \omega_p\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$
16:     end while
17:     Limit $\omega_{\tilde{\nu}}$ such that $\chi$ changes by a maximum fraction $\eta_{\max}$
18:     $\omega_{\kappa^H} \leftarrow \min(\omega_\rho, \omega_p, \omega_{\tilde{\nu}}, \omega_{\kappa^H})$
19: end for
20: return $\omega_{\kappa^H}$
```

Some clarifications are in order. First, the maximum fractional change is fixed at $\eta_{\max} = 10\%$ – based on experimentation – for all cases presented in this work. Also, the points $x_m$ can be chosen arbitrarily and we select them to be the quadrature points used for computing the interior and boundary integrals involved in the residual calculation. Finally, the bisection method is used in step 13 of Algorithm 2 because pressure is a nonlinear function of the state.

## A. Maximum Primitive Change

The *Maximum Primitive Change* (MPC) method limits the step size such that the physical update limiter (Algorithm 2) passes for all the elements in the mesh. This procedure is summarized in Algorithm 3.

**Algorithm 3** Maximum Primitive Change

```
1: $\omega^k \leftarrow 1$                                           ▷ Assume full update initially
2: for all $\kappa^H \in T^H$ do                                  ▷ Loop over element in the mesh
3:     Select limit points, $x_m$
4:     Evaluate $\mathbf{u}^{H,p}(x_m)|_{\kappa^H}$ and $\Delta\mathbf{u}^{H,p}(x_m)|_{\kappa^H}$
5:     Call Algorithm 2                                   ▷ Limit change in primitive state
6:     $\omega^k \leftarrow \min(\omega_{\kappa^H}, \omega^k)$
7: end for
8: return $\omega^k$
```

## B. Line-search

In optimization problems, line-searches are used to find a step-size along a decent direction that sufficiently reduces the value of the objective function and its gradient. These conditions are known as the *Wolfe conditions*. When solving systems of nonlinear equations, line-searches improve the global convergence properties of Newton-based methods.[24]

The line-search algorithm developed in this work is based on Modisette's method,[25] and it relies on the optimization character of pseudo-transient continuation (Section B). In short, both algorithms satisfy

Armijo's rule[26] by back-tracking from an initial step-size until an update leads to a reduction in the 2-norm of the unsteady residual.

The difference between Modisette's method and ours is in computing an initial guess for the step size. While Modisette uses the $\omega^k$ computed with MPC as a starting step-size, our method only checks the physicality for elements whose update directions are deemed "unsafe". This is assessed by computing the projection,

$$\theta = \Delta\mathbf{U}^k \cdot \frac{\partial \mathbb{P}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}, \tag{30}$$

where $\mathbb{P}$ is the sum of all elemental penalties $\mathbb{P}_{\kappa^H}$. The elements with positive contributions to $\theta$ have update directions deemed "unsafe" since $\Delta\mathbf{U}^k$ will lead to an increase in $\mathbb{P}_{\kappa^H}$ for those elements.

The line-search algorithm is summarized below.

**Algorithm 4** Line-search

```
 1: $\omega_{\text{phys}} \leftarrow 1$                                   ▷ Initial guess for physical update
 2: for all $\kappa^H$ with positive contribution to $\theta$ do
 3:     Select limit points, $\mathbf{x}_m$
 4:     Evaluate $\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$ and $\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$
 5:     Call Algorithm 2                                  ▷ Limit physical update
 6:     $\omega_{\text{phys}} \leftarrow \min(\omega_{\kappa^H}, \omega_{\text{phys}})$
 7: end for
 8: $\omega^k \leftarrow \omega_{\text{phys}}$
 9: $\tilde{\mathbf{U}} \leftarrow \mathbf{U}^k + \omega^k \Delta\mathbf{U}^k$                       ▷ Trial state vector
10: while $|\mathbf{R}_t(\tilde{\mathbf{U}})|_{L_2} > |\mathbf{R}(\mathbf{U}^k)|_{L_2}$ OR $\tilde{\mathbf{U}}$ is not physical do
11:     $\omega^k \leftarrow \frac{\omega^k}{2}$
12:     $\tilde{\mathbf{U}} \leftarrow \mathbf{U}^k + \omega^k \Delta\mathbf{U}^k$
13: end while
14: return $\omega^k$
```

Note that step 10 in Algorithm 4 checks if the trial state, $\tilde{\mathbf{U}}$, is physical. This check involves verifying if the physicality constraints are satisfied at the limit points. Also, we separate the 2-norm of residual into the individual conservation equations and require a drop in each of those norms. This reduces the effect of badly-scaled discrete systems that cause the residual norm to be dominated by the worst residual component which is also observed by Modisette.[25]

### 1. *Greedy algorithm*

The physical update limiter in Algorithm 2 is heuristic and the line-search algorithm described above can prematurely exit with $\omega^k = \omega_{\text{phys}}$ while $\omega_{\text{phys}} < 1$. This can slow-down the convergence and increase the susceptibility to limit cycles. To address this possibility, a greedy algorithm is introduced. This algorithm amplifies $\omega^k$ while Armijo's rule is satisfied or until a full update is obtained, $\omega^k = 1$.

A safety check based on the projection in Eqn. 30 is performed before amplifying $\omega^k$. Specifically, a negative projection, $\theta$, indicates that is globally "safe" to proceed along $\Delta\mathbf{U}^k$. However, even with $\theta < 0$, amplifying $\omega^k$ may not be locally safe because certain elements may have positive contributions to $\theta$ and $\Delta\mathbf{U}^k$ may lead to non-physical states on those elements. The algorithm is summarized below.

**Algorithm 5** Greedy algorithm

```
 1: if $\omega^k = \omega_{\text{phys}}$ AND $\theta < 0$ then                ▷ $\theta$ is defined in Eqn. 30
 2:     while $\omega^k \le 1$ do
 3:         $\omega^k \leftarrow \beta_\omega \cdot \omega^k$                  ▷ For all cases, we use $\beta_\omega = 1.1$
 4:         $\tilde{\mathbf{U}} \leftarrow \mathbf{U}^k + \omega^k \Delta\mathbf{U}^k$
 5:         if $\tilde{\mathbf{U}}$ is not physical then
 6:             $\omega^k \leftarrow \frac{\omega^k}{2}$
 7:             return $\omega^k$
 8:         end if
 9:         if $|\mathbf{R}_t(\tilde{\mathbf{U}})|_{L_2} > |\mathbf{R}(\mathbf{U}^k)|_{L_2}$ then
10:             $\omega^k \leftarrow \frac{\omega^k}{\beta_\omega}$
11:             return $\omega^k$
12:         end if
13:     end while
14: end if
15: return $\omega^k$
```

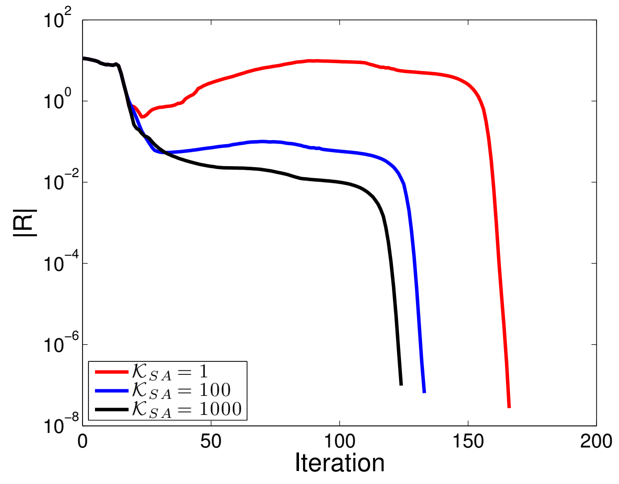
## VI. Scaling of the SA Discrete Equation

Most practical cases in the aeronautical industry are in the Reynolds number regime of $10^6 \rightarrow 10^7$. In this regime, $\tilde{\nu}/\nu_\infty$ typically ranges 4 to 5 orders of magnitude. Therefore, it is also desirable to choose an appropriate scale for $\tilde{\nu}$. The scale used in this work is

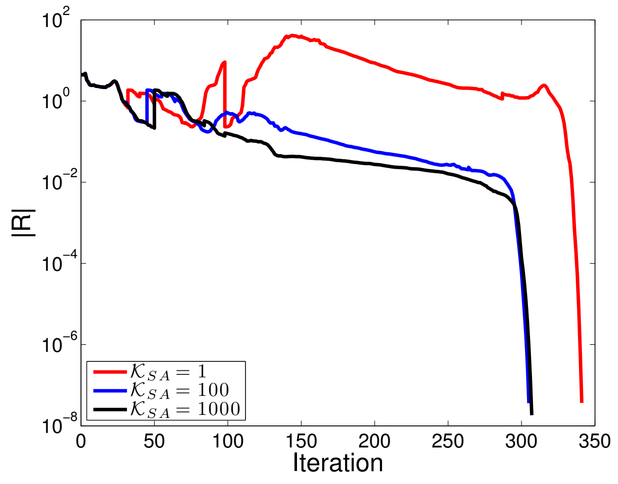
$$(\rho\tilde{\nu})' = \frac{\rho\tilde{\nu}}{\kappa_{\text{SA}}\mu_\infty}, \tag{31}$$

where $(\rho\tilde{\nu})'$ is the scaled conserved variable that is stored and evolved by the solver. $\kappa_{\text{SA}}$ is a scaling factor and $\mu_\infty$ is the freestream dynamic viscosity. Essentially, we are non-dimensionalizing $\rho\tilde{\nu}$ by a factor larger than the physical viscosity.

To exemplify the effect of $\kappa_{\text{SA}}$, we show in Figure 1 the residual history for two flows at $Re = 6.5 \times 10^6$, one subsonic and one transonic. For each case, three scaling factors are used, $\kappa_{\text{SA}} = 1,\ 100,\ 1000$. Note that $\kappa_{\text{SA}}$ significantly affects the convergence history. Specifically, the larger values of $\kappa_{\text{SA}}$ ameliorate the secondary transient observed in RANS computations using DG.[27]



(a) RAE2822 - $M_\infty = 0.3$, $Re = 6.5 \times 10^6$, $\alpha = 2.31°$.



(b) RAE2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$.

**Figure 1. Residual convergence using $p = 1$ for different $\tilde{\nu}$ scaling factors ($\kappa_{\text{SA}}$).**

The drag and lift coefficients ($C_D$ and $C_L$ respectively) for both flow conditions are shown in Table 1 and Table 2. As expected, the scaling factor has virtually no effect on the results. However, it makes the conserved variables closer in magnitude which, in turn, helps implicit time integration methods.

**Table 1. RAE2822 -** $M_\infty = 0.3,\ Re = 6.5 \times 10^6,\ \alpha = 2.31^\circ$ **− Comparison of force coefficients and maximum values of** $x$**-momentum and SA working variable for different scaling factors.**

| Quantity | $\kappa_{\mathrm{SA}} = 1$ | $\kappa_{\mathrm{SA}} = 100$ | $\kappa_{\mathrm{SA}} = 1000$ |
|---|---|---|---|
| $C_D$ | 0.0122 | 0.0122 | 0.0122 |
| $C_L$ | 0.4507 | 0.4507 | 0.4506 |
| $(\rho v_x)_{\max}$ | 1.25182 | 1.25182 | 1.25192 |
| $(\rho\tilde{\nu})'_{\max}$ | $1.03775 \times 10^3$ | $1.03775 \times 10^1$ | 1.03783 |

**Table 2. RAE2822 -** $M_\infty = 0.734,\ Re = 6.5 \times 10^6,\ \alpha = 2.79^\circ$ **− Comparison of force coefficients and maximum values of** $x$**-momentum and SA working variable for different scaling factors.**

| Quantity | $\kappa_{\mathrm{SA}} = 1$ | $\kappa_{\mathrm{SA}} = 100$ | $\kappa_{\mathrm{SA}} = 1000$ |
|---|---|---|---|
| $C_D$ | 0.0198 | 0.0198 | 0.0198 |
| $C_L$ | 0.7334 | 0.7334 | 0.7334 |
| $(\rho v_x)_{\max}$ | 1.11808 | 1.11808 | 1.11811 |
| $(\rho\tilde{\nu})'_{\max}$ | $1.64361 \times 10^3$ | $1.64362 \times 10^1$ | 1.64378 |

## VII. Test-suite results

We now present results for a set of flow cases ranging from intermediate to difficult.[b] For each case, we combine the PTC and CPTC methods with each of the CFL evolution strategies and solution update algorithms. We identify each run with a sequence of 3 digits (Table 3) that respectively correspond to the continuation method, the solution update method, and the CFL evolution strategy.

For all cases, the residual convergence criterion is 9 orders of magnitude reduction compared to its initial norm. In order to compare the methods under equal-footing, the discretization and the GMRES parameters are the same for all runs in Table 3 and they only vary between flow cases. Similarly, a single mesh is generated and used for all the runs in each case. The purpose of the meshes used in this chapter is not to allow accurate solutions. Instead, they are generated so as to have enough spatial resolution to reveal the flow features relevant for each case.

The cases are initialized with uniform flow under free-stream conditions and initial $\mathrm{CFL}^0 = 1$. For each case, we present a color-coded table that assesses the success of all the runs. In these tables, green means the run converged, yellow means that the run reached the total wall time or maximum iterations without convergence and red means the run had either a non-physical error or the CFL is decreased below minimum ($\mathrm{CFL}_{\min} = 10^{-10}$ for all cases). The converged runs in each case are compared with respect to number of nonlinear iterations, number of GMRES iterations and total wall time.

### A. RAE 2822 − $M_\infty = 0.734,\ Re = 6.5 \times 10^6,\ \alpha = 2.79^\circ$

The first case is transonic, turbulent flow over the RAE 2822 airfoil. The scheme's polynomial order for this case is $p = 2$ and the residual operator includes Persson and Peraire's[9] shock-capturing term. The SA equation is scaled by $\kappa_{\mathrm{SA}} = 1000$ and the free-stream turbulence level is 0.1%. The outer boundary of the domain is located 100 chord-lengths away from the airfoil and each edge of the mesh shown in Figure 2(a) is a quartic ($q = 4$) polynomial. The mesh has 990 quadrilaterals and the height of the first layer of elements off the wall is such that $y^+ \approx 5 \times 10^3$, based on a flat-plate correlation for the coefficient of friction.

[b]Level of difficulty was assessed by the 1st International Workshop on High-Order CFD Methods.

**Table 3. Summary of combinations of algorithms; PTC: pseudo-transient continuation; CPTC: constrained pseudo-transient continuation; MPC: maximum primitive change; LS: line-search; LS+G: line-search with greedy algorithm; SER: switched evolution relaxation; EXP: exponential progression; RDM: residual difference method; mRDM: monotonic residual difference method.**

| Run ID | $\Delta\mathbf{U}^k$ method | $\omega^k$ method | CFL$^k$ method |
|---|---|---|---|
| 1.1.1 | PTC | MPC | EXP ($\beta = 1.2$) |
| 1.1.2 | PTC | MPC | SER |
| 1.1.3 | PTC | MPC | RDM ($\beta = 2.0$) |
| 1.1.4 | PTC | MPC | mRDM ($\beta = 2.0$) |
| 1.2.1 | PTC | LS | EXP ($\beta = 1.2$) |
| 1.2.2 | PTC | LS | SER |
| 1.2.3 | PTC | LS | RDM ($\beta = 2.0$) |
| 1.2.4 | PTC | LS | mRDM ($\beta = 2.0$) |
| 1.3.1 | PTC | LS+G | EXP ($\beta = 1.2$) |
| 1.3.2 | PTC | LS+G | SER |
| 1.3.3 | PTC | LS+G | RDM ($\beta = 2.0$) |
| 1.3.4 | PTC | LS+G | mRDM ($\beta = 2.0$) |
| 2.1.1 | CPTC | MPC | EXP ($\beta = 1.2$) |
| 2.1.2 | CPTC | MPC | SER |
| 2.1.3 | CPTC | MPC | RDM ($\beta = 2.0$) |
| 2.1.4 | CPTC | MPC | mRDM ($\beta = 2.0$) |
| 2.2.1 | CPTC | LS | EXP ($\beta = 1.2$) |
| 2.2.2 | CPTC | LS | SER |
| 2.2.3 | CPTC | LS | RDM ($\beta = 2.0$) |
| 2.2.4 | CPTC | LS | mRDM ($\beta = 2.0$) |
| 2.3.1 | CPTC | LS+G | EXP ($\beta = 1.2$) |
| 2.3.2 | CPTC | LS+G | SER |
| 2.3.3 | CPTC | LS+G | RDM ($\beta = 2.0$) |
| 2.3.4 | CPTC | LS+G | mRDM ($\beta = 2.0$) |

Note the coarse resolution in Figure 2(b), this is because accuracy is not the primary goal of these cases, but rather to assess the ability of the solver to get a *zero-residual* solution.

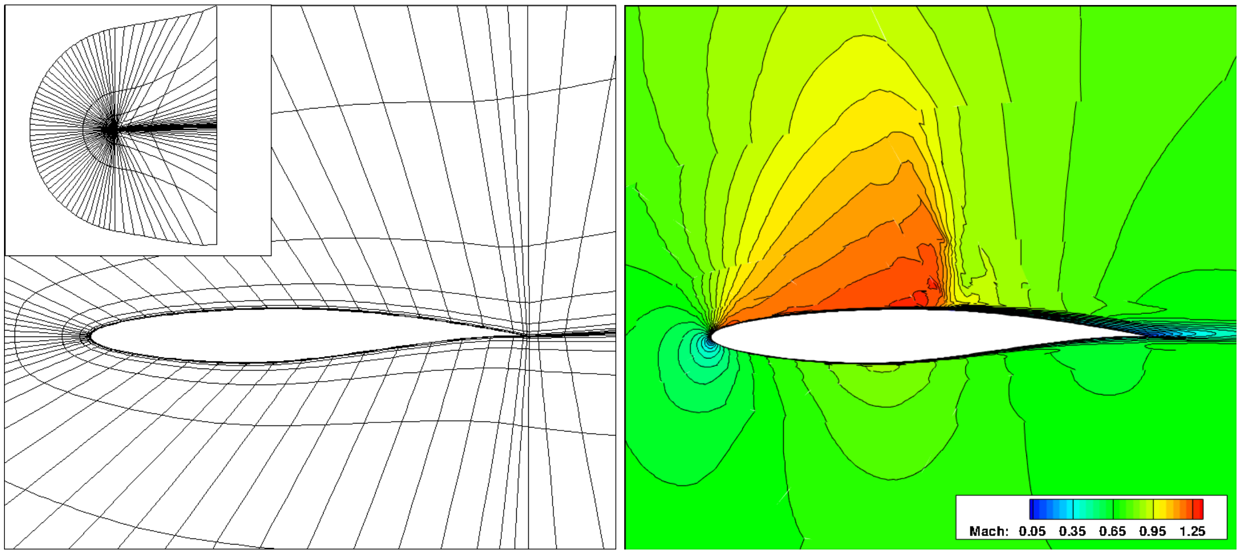
The maximum number of iterations for each run is 10000 and the computational resources are 32 processors and a maximum wall time of 8 hours. Both PTC and CPTC converges with all solution update methods using either EXP or mRDM while none of the runs using SER and RDM converges for this case (Table 4). Those evolution strategies are non-monotonic as they can decrease or increase the CFL at each iteration. In this case, they are distracted by secondary transients and reduce the CFL, making the solver resolve those transients and thereby consuming many iterations.

The constrained version of PTC is not able to converge using MPC and mRDM (run 2.1.4) while the unconstrained version converges under the same conditions (run 1.1.4). As the penalization term increases in run 2.1.4, MPC keeps limiting the update and eventually it becomes small enough that the CFL decreases at a faster rate than the penalty increases.

Table 5 compares the converged runs for this case. The fastest run for this case is 1.2.1 followed by run 2.3.1. Within the runs using the greedy line-search (runs $x.3.x$ in Table 5), CPTC takes fewer nonlinear iterations and less time. This is due to the greedy algorithm being triggered more often since the update direction at each nonlinear step accounts for the physicality constraints.

## B. NACA 0012 – $M_\infty = 0.8,\ Re = 6.5 \times 10^6,\ \alpha = 0°$

The second test case is also transonic, turbulent flow over an airfoil. The scheme's polynomial order is $p = 2$. The SA model is scaled by $\kappa_{\text{SA}} = 100$ and the free-stream turbulence level is 0.01%. Since this is a



(a) Quartic mesh (990 elements).

(b) Mach number contours.

**Figure 2. RAE2822 -** $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$, $p = 2$: **mesh and Mach number contours.**

**Table 4. RAE 2822 -** $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$, $p = 2$: **success assessment of all runs.**

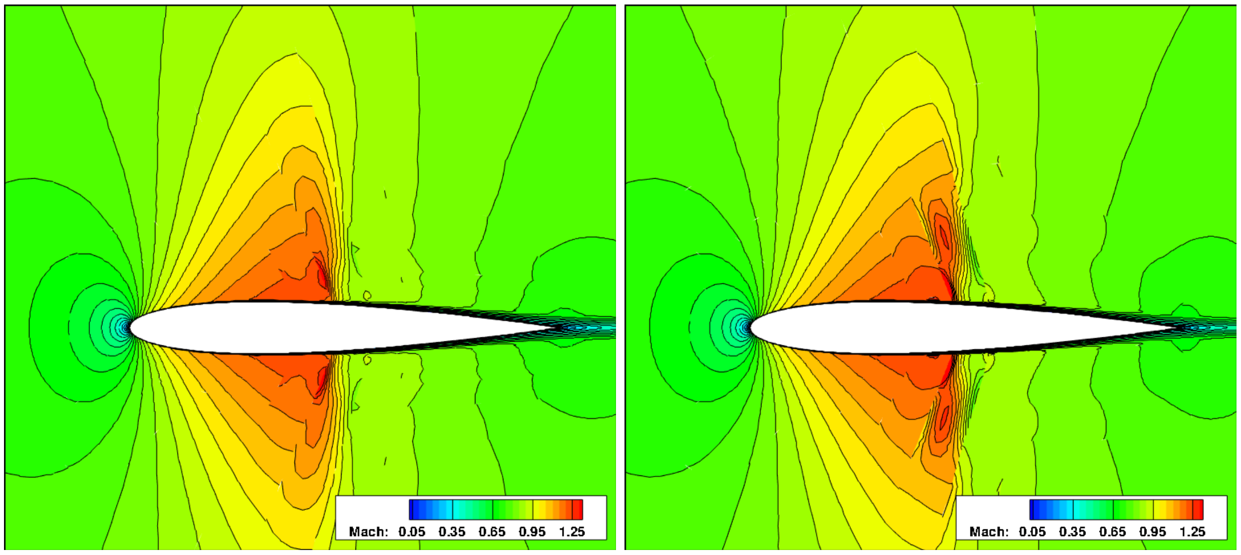
| | PTC | | | CPTC | | |
|---|---|---|---|---|---|---|
| | MPC | LS | LS+G | MPC | LS | LS+G |
| EXP | 🟩 run 1.1.1 | 🟩 run 1.2.1 | 🟩 run 1.3.1 | 🟩 run 2.1.1 | 🟩 run 2.2.1 | 🟩 run 2.3.1 |
| SER | 🟨 run 1.1.2 | 🟨 run 1.2.2 | 🟨 run 1.3.2 | 🟨 run 2.1.2 | 🟥 run 2.2.2 | 🟨 run 2.3.2 |
| RDM | 🟨 run 1.1.3 | 🟨 run 1.2.3 | 🟨 run 1.3.3 | 🟨 run 2.1.3 | 🟨 run 2.2.3 | 🟨 run 2.3.3 |
| mRDM | 🟩 run 1.1.4 | 🟩 run 1.2.4 | 🟩 run 1.3.4 | 🟥 run 2.1.4 | 🟩 run 2.2.4 | 🟩 run 2.3.4 |

non-lifting case, the outer boundary of the domain is located at 30 chord-lengths from the airfoil and the mesh is composed of 1740 quadrilaterals with quartic polynomial edges (Figure 3). The height of the first layer of elements is such that $y^+ \approx 2 \times 10^3$, based one a flat-plate correlation for the friction at the wall. We emphasize that the purpose of meshes in this chapter is not to allow for accurate solutions but to simply reveal the relevant flow features. The computational resources for this case are 40 processors for 8 hours of wall-time and the maximum number of iterations is 10000.

We present two sets of runs for this case. The first set uses Persson and Peraire's[9] shock-capturing term and Figure 4(a) shows the Mach number contours obtained using that term. The second set of runs does not use shock capturing and Figure 4(b) shows the Mach contours for this condition. Note the oscillatory behavior of the solution in the vicinity of the shocks. Clearly, it is not ideal to simulate flows with shocks without a shock-capturing scheme. However, this exercise is useful to assess the robustness of the solution advancement methods.

**Table 5. RAE 2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$, $p = 2$: metrics for converged runs normalized by run 1.1.1 – absolute values in parentheses.**

| Run ID | Nonlinear iterations | GMRES iterations | Wall time |
|---|---|---|---|
| 1.1.1 | 1.000 (2184) | 1.000 (41527) | 1.000 ($8.886 \times 10^3 s$) |
| 1.1.4 | 1.652 | 1.006 | 1.410 |
| 1.2.1 | 0.421 | 0.574 | 0.582 |
| 1.2.4 | 1.841 | 1.307 | 2.140 |
| 1.3.1 | 0.243 | 0.291 | 0.682 |
| 1.3.4 | 1.282 | 0.921 | 1.833 |
| 2.1.1 | 0.595 | 0.598 | 0.685 |
| 2.2.1 | 0.851 | 0.899 | 1.352 |
| 2.2.4 | 1.404 | 1.254 | 2.148 |
| 2.3.1 | 0.150 | 0.280 | 0.673 |
| 2.3.4 | 0.603 | 0.543 | 1.502 |



(a) Mach number contours with shock-capturing. (b) Mach number contours without shock-capturing.

**Figure 4. NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$: Mach number contours.**

Table 6 shows the success of the runs using the shock-capturing term, with which most of the runs converges. Interestingly, the constrained solver with the greedy line-search and SER (run 2.3.2) exceeds the maximum allotted time while the unconstrained solver with the same update method and CFL strategy (run 1.3.2) converges within the time limit.

**Table 6. NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ with shock-capturing: success assessment of all runs**

| | PTC | | | CPTC | | |
|---|---|---|---|---|---|---|
| | MPC | LS | LS+G | MPC | LS | LS+G |
| EXP | 🟩 run 1.1.1 | 🟩 run 1.2.1 | 🟩 run 1.3.1 | 🟩 run 2.1.1 | 🟩 run 2.2.1 | 🟩 run 2.3.1 |
| SER | 🟩 run 1.1.2 | 🟩 run 1.2.2 | 🟩 run 1.3.2 | 🟩 run 2.1.2 | 🟩 run 2.2.2 | 🟨 run 2.3.2 |
| RDM | 🟨 run 1.1.3 | 🟨 run 1.2.3 | 🟨 run 1.3.3 | 🟨 run 2.1.3 | 🟨 run 2.2.3 | 🟨 run 2.3.3 |
| mRDM | 🟩 run 1.1.4 | 🟩 run 1.2.4 | 🟩 run 1.3.4 | 🟩 run 2.1.4 | 🟩 run 2.2.4 | 🟩 run 2.3.4 |

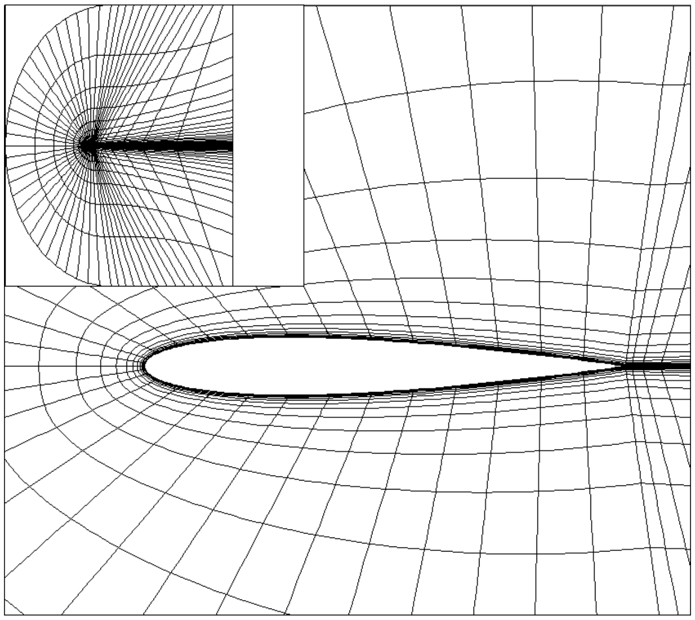


**Figure 3. NACA 0012 -** $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$: **quartic mesh (1740 elements).**

Figure 5 compares the final states for the runs using the greedy line-search and SER (runs $x$.3.2). Note the under-development of the wake and the residual for the turbulence equation for run 2.3.2 (Figure 5(b)). During that run, the CFL is reduced from its initial value after the initial transient and remains below 1 for most of the iterations and, thus, not reaching Newton convergence. The same issue occurs with all runs using RDM for evolving the CFL.

Within the successful runs using shock-capturing (Table 7), the quickest are 2.3.1 and 2.3.4, as they take practically the same time to converge. Similarly to the previous case, the number of iterations for run 2.3.1 is smaller than its unconstrained counterpart, run 1.3.1, due to the greedy algorithm being triggered more often with the constrained solver.

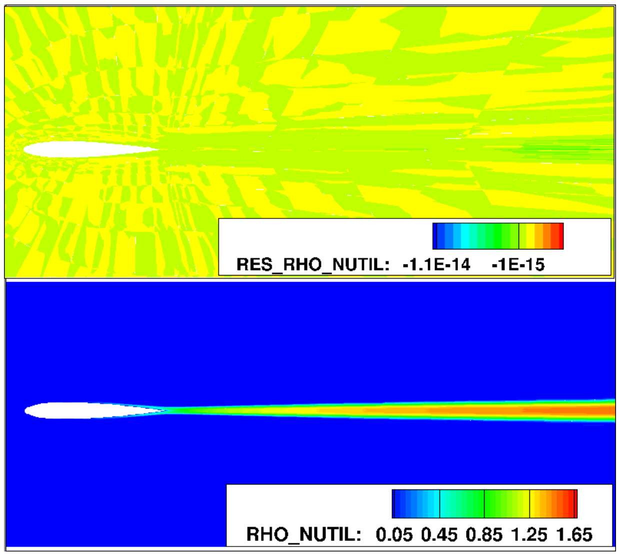
We now discuss the runs without the shock-capturing scheme. Table 8 shows the success of all these runs. Again, the monotonic CFL strategies perform better than the non-monotonic strategies. The constrained solver is successful with all the solution update strategies using both EXP and mRDM, while the unconstrained method using MPC and EXP finishes with the CFL below minimum.

Table 9 compares the runs that converges without the shock-capturing scheme. Note that the runs using the greedy algorithm are the fastest as they save many nonlinear iterations and they also compute fewer matrix-vector products involved in the GMRES iterations. Amongst the converged runs using the greedy algorithm, the runs using the unconstrained solver (runs 1.3.$x$) are slightly faster than their constrained counterpart (runs 2.3.$x$) despite the greedy algorithm being triggered more often. This is because the constrained runs use more GMRES iterations.

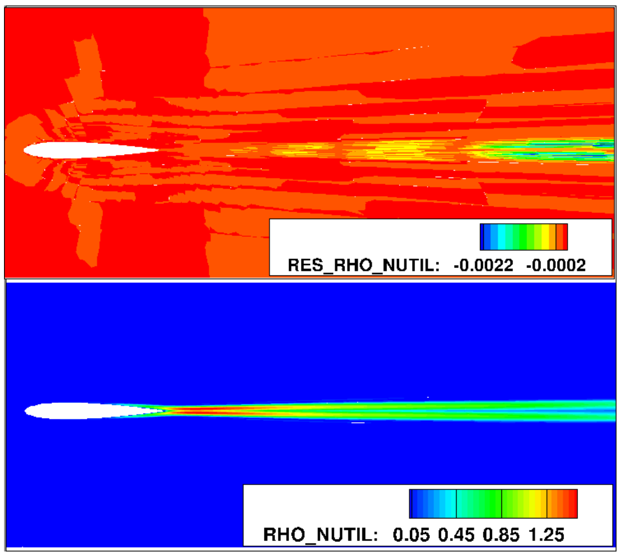
## C. MDA 30p30n – $M_\infty = 0.2,\ Re = 9 \times 10^6,\ \alpha = 16°$

This test case is subsonic, turbulent flow over a high-lift, multi-element airfoil. The scheme's approximation order is $p = 1$. The discrete equation for the turbulence model is scaled by $\kappa_{\text{SA}} = 1000$ and the free-stream level of turbulence is 1%. Figure 6 shows the mesh and Mach number contours for this flow. A linear multi-block mesh is generated with the objective of having a good alignment of the cells with the wake. Patches of elements from the linear mesh are then agglomerated to generate the quartic mesh shown in Figure 6(a). The agglomerated mesh has 4070 elements and the off-wall spacing is such that $y^+ \approx 3 \times 10^3$, based on a flat-plate correlation.

For each run, 8 wall-clock hours of 16 processors are allotted and the maximum number of iterations is 10000. Table 10 shows the success of all the runs for this flow. Note that the rate of success is considerably smaller than the previous cases. The non-monotonic CFL strategies, SER and RDM, reduce the CFL in a



(a) Contours for $\rho\tilde{\nu}$ and its residual for run 1.3.2.



(b) Contours for $\rho\tilde{\nu}$ and its residual for run 2.3.2 (under-converged).

**Figure 5. NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0^\circ$, $p = 2$ with shock-capturing: comparison of final states for runs $x.3.2$.**

attempt to resolve the transients, a process which takes many iterations due to the stiffness of this problem.

Both PTC and CPTC are able to obtain converged solutions with the update methods based on the line-search. However, the constrained solver with the greedy line-search (run 2.3.4) reaches the maximum number of iterations before convergence while the unconstrained counterpart succeeds in converging the residual within the iteration limit.

It is worth emphasizing the challenging nature of this case. The blunt flap cove and the sharp turn behind the slat are geometric features that cause flow separation and hence the flow exhibits strong initial transients when initialized with uniform free-stream conditions. Clearly, such a flow initialization strategy is not ideal. However, we believe it is a useful exercise to test the solver under such demanding conditions.

Amongst the converged runs (Table 11), the constrained solver with line-search and mRDM (run 2.2.4) is the most efficient in this case. Note that the wall-time is strongly related to the number of GMRES iterations. This is expected, as most of the computational time in an implicit solver is spent computing matrix-vector products involved in the linear solves.

## D. DPW 3 Wing 1 – $M_\infty = 0.76$, $Re = 5 \times 10^6$, $\alpha = 0.5^\circ$

This case is transonic, turbulent flow over the baseline wing from the Third Drag Prediction Workshop. A linear multi-block (C-topology) mesh is generated following the workshop's guidelines.[28] The linear elements are agglomerated to generate a cubic ($q = 3$) mesh with 29310 elements (Figure 7(a)). The spacing of the linear mesh is such that the agglomerated mesh presents $y^+ \approx 1$, based on a friction coefficient correlation for a flat plate. Figure 7(b) shows the contours of Mach number and SA working variable at a mid-span slice of the flow domain.

The SA discrete equation is scaled by $\kappa_{\text{SA}} = 1$ and the free-stream level of turbulence is 0.1%. The scheme's approximation order is $p = 1$ and Persson and Peraire's[9] shock-capturing term is included in the residual operator. Each run uses 804 processors for a maximum of 5 hours and the maximum number of iterations is 4000. Under these limits, only the constrained solver converges as shown in Table 12. Note that the greedy line-search fails in most of the runs. This is due to excessively large updates in the beginning of the calculation that eventually lead to non-physical states at the trailing edge of the wing. The greedy algorithm is prematurely triggered because the inner-product in Eqn. 30 is dominated by negative contributions. In the case of the constrained solver, the local penalization is not enough to make the projection, $\theta$, globally positive.

The constrained solver using MPC and mRDM (run 2.1.4) takes the shortest time and fewest number

**Table 7. NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ with shock-capturing: metrics for converged runs normalized by run 1.1.4 – absolute values in parentheses.**

| Run ID | Nonlinear iterations | GMRES iterations | Wall time (seconds) |
|---|---|---|---|
| 1.1.1 | 1.000 (1133) | 1.000 (37374) | 1.000 ($7.301 \times 10^3$) |
| 1.1.2 | 4.185 | 2.501 | 3.130 |
| 1.1.4 | 1.109 | 1.169 | 1.129 |
| 1.2.1 | 0.733 | 0.958 | 0.875 |
| 1.2.2 | 4.162 | 2.488 | 3.137 |
| 1.2.4 | 1.237 | 1.171 | 1.191 |
| 1.3.1 | 0.387 | 0.653 | 0.589 |
| 1.3.2 | 3.996 | 2.420 | 3.062 |
| 1.3.4 | 0.646 | 0.695 | 0.724 |
| 2.1.1 | 0.945 | 0.931 | 0.934 |
| 2.1.2 | 2.178 | 1.662 | 1.843 |
| 2.1.4 | 1.139 | 1.172 | 1.171 |
| 2.2.1 | 0.606 | 0.830 | 0.739 |
| 2.2.2 | 2.143 | 1.646 | 1.846 |
| 2.2.4 | 0.954 | 1.054 | 1.010 |
| 2.3.1 | 0.317 | 0.633 | 0.554 |
| 2.3.4 | 0.400 | 0.585 | 0.554 |

**Table 8. NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ without shock-capturing: success assessment of all runs**

| | PTC | | | CPTC | | |
|---|---|---|---|---|---|---|
| | MPC | LS | LS+G | MPC | LS | LS+G |
| EXP | 🟥 run 1.1.1 | 🟩 run 1.2.1 | 🟩 run 1.3.1 | 🟩 run 2.1.1 | 🟩 run 2.2.1 | 🟩 run 2.3.1 |
| SER | 🟥 run 1.1.2 | 🟥 run 1.2.2 | 🟥 run 1.3.2 | 🟨 run 2.1.2 | 🟨 run 2.2.2 | 🟨 run 2.3.2 |
| RDM | 🟥 run 1.1.3 | 🟥 run 1.2.3 | 🟥 run 1.3.3 | 🟥 run 2.1.3 | 🟥 run 2.2.3 | 🟥 run 2.3.3 |
| mRDM | 🟩 run 1.1.4 | 🟩 run 1.2.4 | 🟩 run 1.3.4 | 🟩 run 2.1.4 | 🟩 run 2.2.4 | 🟩 run 2.3.4 |

of GMRES iterations to converge (Table 13). As discussed in the previous cases, the number of GMRES iterations strongly affects the run time. However, other aspects can significantly affect the total run time. For example, runs 2.1.1 and 2.2.4 have significantly different run times even though they take virtually the same number of GMRES iterations. The extra time in run 2.2.4 is mostly due to the additional residual evaluations involved in the line-search algorithm that compensate for the fewer nonlinear iterations.

Figure 8 shows the residual and penalization histories for the runs listed in Table 13. Note that runs 2.1.1 and 2.2.1 track similar penalization paths with the exception that the run with line-search (2.2.1) takes fewer non-linear iterations. The runs using mRDM also follow similar paths in the first $\sim 40$ iterations but the line-search (2.2.4) returns a few "no-updates" (marked by the sudden drops in CFL Figure 8(b)), that is, when $\omega^k < \omega_{\min}$ and the state is reset to the last safe update.

## VIII. Concluding Remarks

The results presented here are sensitive to various parameters in the discretization, in the linear solver, and, more importantly, in the solution update methods. Specifically, the $\beta$ parameter in the CFL strategies significantly affects the performance of the solver. The value of this parameter is somewhat heuristic and the rationale for deciding on its value is that $\beta$ closer to 1 is less aggressive and it tends to be more robust. However, less aggressive CFL strategies are more likely to trap the solver in transients that may not be

**Table 9. NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ without shock-capturing: metrics for converged runs normalized by run 1.1.4 – absolute values in parentheses.**

| Run ID | Nonlinear iterations | GMRES iterations | Wall time |
|---|---|---|---|
| 1.1.4 | 1.000 (3707) | 1.000 (34671) | 1.000 ($1.347 \times 10^4 s$) |
| 1.2.1 | 0.281 | 0.456 | 0.327 |
| 1.2.4 | 0.454 | 0.557 | 0.477 |
| 1.3.1 | 0.0968 | 0.316 | 0.170 |
| 1.3.4 | 0.179 | 0.301 | 0.229 |
| 2.1.1 | 0.401 | 0.440 | 0.410 |
| 2.1.4 | 0.308 | 0.537 | 0.367 |
| 2.2.1 | 0.250 | 0.474 | 0.309 |
| 2.2.4 | 0.386 | 0.542 | 0.422 |
| 2.3.1 | 0.127 | 0.335 | 0.200 |
| 2.3.4 | 0.136 | 0.529 | 0.275 |

**Table 10. MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$, $p = 1$: success assessment of all runs**

| | PTC | | | CPTC | | |
|---|---|---|---|---|---|---|
| | MPC | LS | LS+G | MPC | LS | LS+G |
| EXP | 🟨 run 1.1.1 | 🟨 run 1.2.1 | 🟩 run 1.3.1 | 🟨 run 2.1.1 | 🟨 run 2.2.1 | 🟩 run 2.3.1 |
| SER | 🟨 run 1.1.2 | 🟨 run 1.2.2 | 🟨 run 1.3.2 | 🟨 run 2.1.2 | 🟨 run 2.2.2 | 🟨 run 2.3.2 |
| RDM | 🟨 run 1.1.3 | 🟨 run 1.2.3 | 🟨 run 1.3.3 | 🟨 run 2.1.3 | 🟨 run 2.2.3 | 🟨 run 2.3.3 |
| mRDM | 🟨 run 1.1.4 | 🟨 run 1.2.4 | 🟩 run 1.3.4 | 🟨 run 2.1.4 | 🟩 run 2.2.4 | 🟨 run 2.3.4 |

physical. An example of such a problem is when the flow is initialized with uniform free-stream flow next to a wall and, if a flow expansion, *e.g.* at a blunt trailing edge, is solved time-accurately, negative pressure may occur. The penalization approach reduces the sensitivity to these non-physical transients but it is not a *bullet-proof* approach and better flow initialization methods are certainly helpful.

The combination of the constrained solver with line-search and the mRDM CFL strategy converges all the cases presented here, however, this combination is not the fastest as EXP outperforms mRDM in certain cases. Also, the line-search algorithms make use of the penalization idea and using them with PTC is very effective in obtaining solutions for most of the cases presented here.

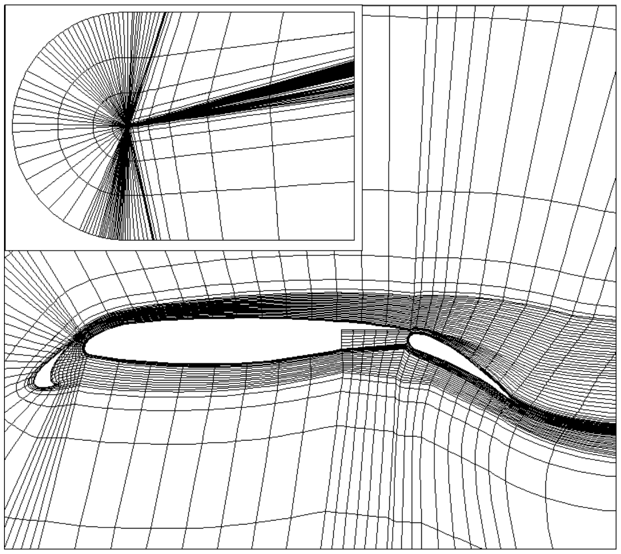
Lastly, other works[29–33] have reported lack of robustness with the version of SA used here. More recent modifications to the SA turbulence model[32,34] alleviate the non-smooth behavior of the turbulence production term at $\tilde{\nu} < 0$. This is expected to further improve the robustness of RANS simulations using high-order DG and testing the methods presented here using those modifications is ongoing work.

## Acknowledgments

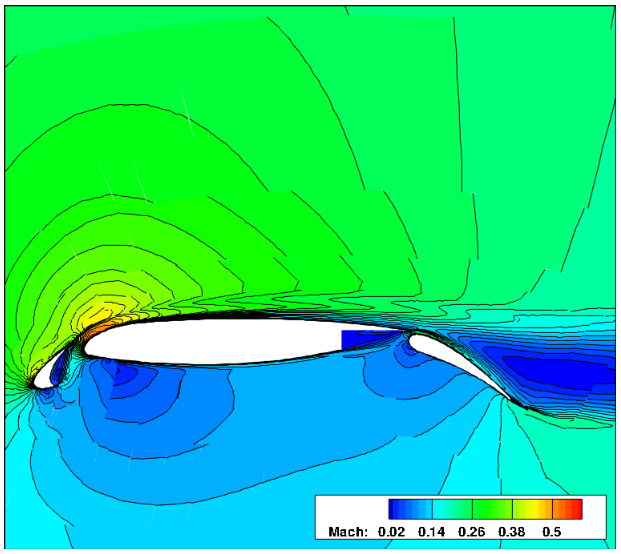
The authors acknowledge the support given by the University of Michigan to the development of this work and appreciate the computational resources provided by NASA which were essential to obtaining the results presented here. Funding for this research is provided by the Air Force Office of Scientific Research under contract number FA9550-10-C-0040.

## References

[1]Buttazzo, G., Frediani, A., Allmaras, S. R., Bussoletti, J. E., Hilmes, C. L., Johnson, F. T., Melvin, R. G., Tinoco, E. N., Venkatakrishnan, V., Wigton, L. B., and Young, D. P., "Algorithm Issues and Challenges Associated with the Development of Robust CFD Codes," *Variational Analysis and Aerospace Engineering*, Vol. 33 of *Springer Optimization and Its Applications*,



(a) Quartic mesh (4070 elements).



(b) Mach number contours.

**Figure 6. MDA 30p30n** $-$ $M_\infty = 0.2,\ Re = 9 \times 10^6,\ \alpha = 16°$**: mesh and Mach number contours.**

**Table 11. MDA 30p30n** $-$ $M_\infty = 0.2,\ Re = 9 \times 10^6,\ \alpha = 16°,\ p = 1$**: metrics for converged runs normalized by run** 1.3.1 $-$ **absolute values in parentheses.**

| Run ID | Nonlinear iterations | GMRES iterations | Wall time (seconds) |
|---|---|---|---|
| 1.3.1 | 1.000 (1412) | 1.000 (608770) | 1.000 ($5.085 \times 10^3$) |
| 1.3.4 | 4.750 | 0.935 | 1.005 |
| 2.2.4 | 5.135 | 1.346 | 1.059 |
| 2.3.1 | 1.161 | 0.881 | 0.920 |

Springer New York, 2009, pp. 1–19.

[2]Cockburn, B., Lin, S.-Y., and Shu, C.-W., "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems," *Journal of Computational Physics*, Vol. 84, No. 90-113, 1989.

[3]van Leer, B., "Towards the ultimate conservative difference scheme. II - Monotonicity and conservation combined in a second order scheme," *Journal of Computational Physics*, , No. 14, 1974, pp. 361–370.

[4]Kuzmin, D., "A Vertex-Based Hierarchical Slope Limiter for P-Adaptive Discontinuous Galerkin Methods," *Journal of Computational and Applied Mathematics*, , No. 233, 2010.

[5]Venkatakrishnan, V., "Convergence to Steady State Solutions of The Euler Equations on Unstructured Grid with Limiters," *Journal of Computational Physics*, , No. 118, 1995, pp. 120–130.

[6]Neumann, J. V. and Richtmyer, R. D., "A method for the numerical calculation of hydrodynamic shocks," *Journal of Applied Physics*, Vol. 21, 1950, pp. 232–237.

[7]Jameson, A., Schmidt, W., and Turkel, E., "Numerical simulation of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes," *AIAA 5th Computational Fluid Dynamics Conference*, 1981.

[8]Barter, G. E., *Shock Capturing with PDE-Based Artificial Viscosity for an Adaptive Higher–Order Discontinuous Galerkin Finite Element Method*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[9]Persson, P.-O. and Peraire., J., "Sub-cell shock capturing for discontinuous Galerkin methods," *44th AIAA Aerospace Sciences Meeting and Exhibit*, No. 2006-112, 2006.

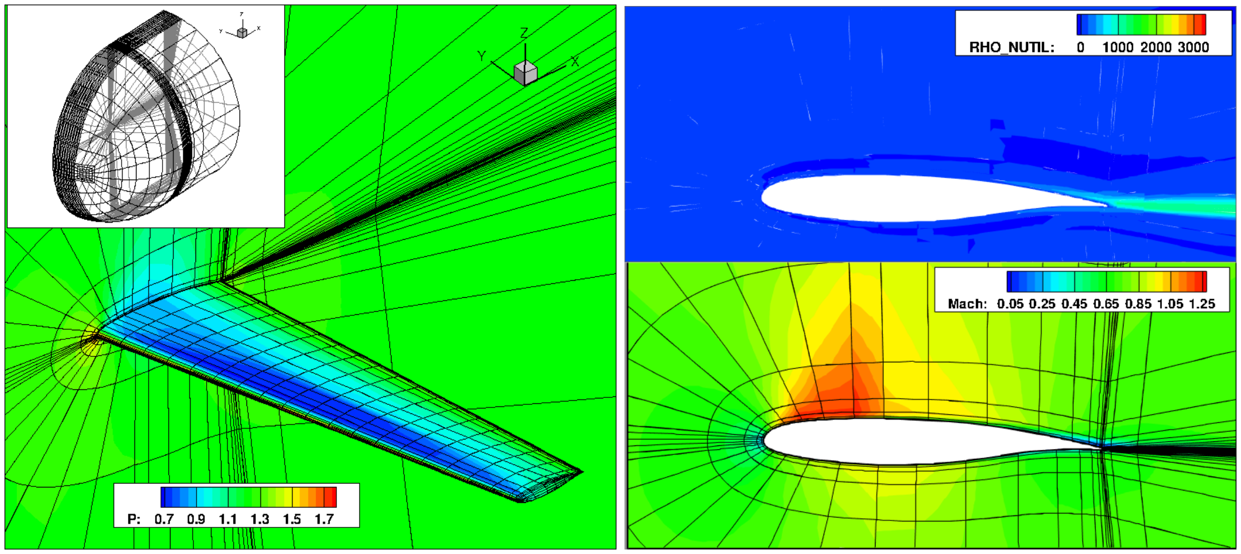
[10]Ceze, M. and Fidkowski, K. J., "A Robust Adaptive Solution Strategy for High-Order Implicit CFD Solvers," *20th AIAA Computaional Fluid Dynamics Conference*, No. AIAA 2011-3696, AIAA, 2011.

[11]Oliver, T. A., *A High–order, Adaptive, Discontinuous Galerkin Finite Elemenet Method for the Reynolds-Averaged Navier-Stokes Equations*, PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[12]Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *30th Aerospace Sciences Meeting and Exhibit*, No. AIAA-92-0439, AIAA, 1992.

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

[14]Bassi, F. and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discon-*



(a) Mesh and static pressure contours. Upper left corner shows a view of the full computational domain (29310 elements).

(b) Mach number and $\rho\tilde{\nu}$ contours.

**Figure 7. DPW 3 Wing 1 – $M_\infty = 0.76,\ Re = 5 \times 10^6,\ \alpha = 0.5°$, $p = 1$: mesh and Mach number contours.**

**Table 12. DPW 3 Wing 1 – $M_\infty = 0.76,\ Re = 5 \times 10^6,\ \alpha = 0.5°$, $p = 1$: success assessment of all runs**

| | PTC | | | CPTC | | |
|---|---|---|---|---|---|---|
| | MPC | LS | LS+G | MPC | LS | LS+G |
| EXP | 🟥 run 1.1.1 | 🟨 run 1.2.1 | 🟥 run 1.3.1 | 🟩 run 2.1.1 | 🟩 run 2.2.1 | 🟥 run 2.3.1 |
| SER | 🟥 run 1.1.2 | 🟥 run 1.2.2 | 🟥 run 1.3.2 | 🟨 run 2.1.2 | 🟨 run 2.2.2 | 🟥 run 2.3.2 |
| RDM | 🟨 run 1.1.3 | 🟨 run 1.2.3 | 🟥 run 1.3.3 | 🟥 run 2.1.3 | 🟥 run 2.2.3 | 🟥 run 2.3.3 |
| mRDM | 🟨 run 1.1.4 | 🟨 run 1.2.4 | 🟨 run 1.3.4 | 🟩 run 2.1.4 | 🟩 run 2.2.4 | 🟥 run 2.3.4 |

*tinuous Galerkin Methods: Theory, Computation and Applications*, edited by K. Cockburn and Shu, Springer, Berlin, 2000, pp. 197–208.

[15]Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[16]Saad, Y., "A Flexible Inner-Outer Preconditioned GMRES Algorithm," *SIAM Journal on Scientific Computing*, Vol. 14, No. 2, 1993, pp. 461–469.

[17]Fidkowski, K. J., *A High–order Discontinuous Galerkin Multigrid Solver for Aerodynamic Applications*, MS thesis, M.I.T., Department of Aeronautics and Astronautics, June 2004.

[18]Bucker, H. M., Pollul, B., and Rasch, A., "On CFL evolution strategies for implicit upwind methods in linearized Euler equations," *International Journal for Numerical Methods in Fluids*, Vol. 59, 2009, pp. 1–18.

[19]Kelley, C. T. and Keyes, D. E., "Convergence analysis of pseudo-transient continuation," *SIAM Journal on Numerical Analysis*, 1998.

[20]Mulder, W. A. and van Leer, B., "Experiments with Implicit Upwind Methods for the Euler equations," *Journal of Computational Physics*, 1985.

[21]Atkins, H. L. and Pampell, A., "Robust and Accurate Shock Capturing Method for High-Order Discontinuos Galerkin Methods," *20th AIAA Computaional Fluid Dynamics Conference*, 2011.

[22]Hartung, J., "A Stable Interior Penalty Method for Convex Extremal Problems," *Numerische Mathematik*, Vol. 29, No. 2, 1978.

[23]Kelley, C. T., Liao, L.-Z., Qi, L., Chu, M. T., Reese, J., and Winton, C., "Projected Pseudo-Transient Continuation," *SIAM Journal on Numerical Analysis*, Vol. 46, No. 6, 2008, pp. 3071–3083.

[24]Kelley, C. T., *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995.

[25]Modisette, J. M., *An Automated Reliable Method for Two-Dimensional Reynolds-averaged Navier-Stokes Simulations*, PhD dissertation, Massachusetts Institute of Technology, 2011.

**Table 13. DPW 3 Wing 1 – $M_\infty = 0.76$, $Re = 5 \times 10^6$, $\alpha = 0.5°$, $p = 1$: metrics for converged runs normalized by run 2.1.1 – absolute values in parentheses.**

| Run ID | Nonlinear iterations | GMRES iterations | Wall time (seconds) |
|---|---|---|---|
| 2.1.1 | 1.000 (1255) | 1.000 (68253) | 1.000 ($5.694 \times 10^3 s$) |
| 2.1.4 | 0.897 | 0.935 | 0.845 |
| 2.2.1 | 0.880 | 0.944 | 1.021 |
| 2.2.4 | 0.751 | 1.002 | 1.127 |

[26]Armijo, L., "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives," *Pacific Journal of Mathematics*, Vol. 16, No. 1, 1966.

[27]Burgess, N. K. and Mavriplis, D. J., "Robust Computation of Turbulent Flows using a Discontinuous Galerkin Method," *50th AIAA Aerospace Sciences Meeting and Exhibit*, No. AIAA 2012-0457, 2012.

[28]Frink, N. T., "3rd AIAA CFD Drag Prediction Workshop Gridding Guidelines," NASA Langley, 2007, http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop3/gridding_guidelines.html.

[29]Persson, P.-O., Nguyen, N. C., and Peraire, J., "RANS Solutions Using High-Order Discontinuous Galerkin Methods," *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.

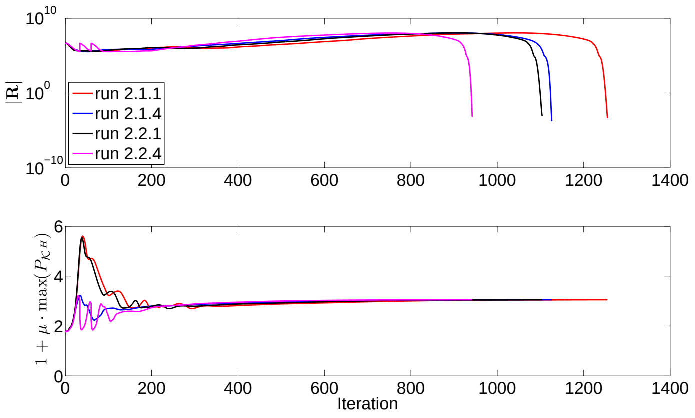
[30]Oliver, T. A. and Darmofal, D. L., "An Unsteady Adaptation Algorithm for Discontinuous Galerkin Discretizations of the RANS Equations," *18th AIAA Computational Fluid Dynamics*, 2007.

[31]Burgess, N. K., Nastase, C. R., and Mavriplis, D. J., "Efficient Solution Techiniques for Discontinous Galerkin Discretizations of Navier-Stokes Equations on Hybrid Anisotropic Meshes," *48th AIAA Aerospace Sciences Meeting and Exhibit*, 2010.

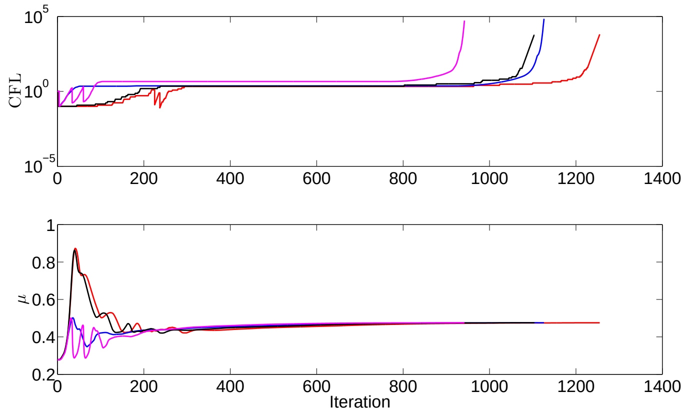
[32]Moro, D., Nguyen, N. C., and Peraire, J., "Navier-Stokes Solutions Using Hybridizable Discontinuous Galerkin Methods," *20th AIAA Computaional Fluid Dynamics Conference*, 2011.

[33]Burgess, N. K., *An Adaptive Discontinuous Galerkin Solver for Aerodynamic Flows*, Ph.D. thesis, University of Wyoming, 2011.

[34]Allmaras, S. R., Johnson, F. T., and Spalart, P. R., "Modifications and Clarifications for the Implementation of the Spalart-AllmarasTurbulence Model," *Seventh Intenational Conference on Computational Fluid Dynamics (ICCFD7)*, 2012.



(a) Residual norm and penalization histories.



(b) CFL and penalty factor histories.

**Figure 8. DPW 3 Wing 1** $-$ $M_\infty = 0.76,\ Re = 5\times10^6,\ \alpha = 0.5°,\ p = 1$**: residual norm and penalization histories for converged runs.**