Performance of a Newton-Krylov-Schur Algorithm for the Numerical Solution of the Steady Reynolds-Averaged Navier-Stokes Equations

David A. Brown[1] and David W. Zingg[2]

University of Toronto Institute for Aerospace Studies, Toronto, Ontario, Canada, M3H 5T6

A methodology is presented for characterizing flow solver performance. The methodology can be applied to assess the efficiency of a given approach, where efficiency is defined in terms of accuracy per unit cost measured in CPU time. The procedure is presented by demonstrating its application to the parallel Newton-Krylov-Schur finite-difference flow solver known as Diablo. The benchmark cases to which the procedure is applied are two-dimensional turbulent flows modeled using the Reynolds-averaged Navier-Stokes equations on three families of NACA 0012 grids with three sets of operating conditions. Performance statistics are presented in a variety of ways that show the relationships between CPU time, grid spacing, and accuracy in ways that are informative for both flow solver users and developers.

I. Introduction

This paper presents a series of external computational aerodynamics flow solver performance studies recently conducted as part of a discussion group session at the 53rd AIAA Aerospace Sciences Meeting [1]. The objective of this discussion group is to develop a methodology for comparing and analyzing flow solver algorithms in terms of robustness and overall efficiency.

The overall efficiency of CFD codes is highly dependent on the hardware, compiler technology, and how efficiently the codes have been programmed. Also, the relative performance of CFD codes can be case-dependent and also grid-dependent. Instead of attempting any sort of direct or conclusive comparison, the methodology in this paper employs a variety of metrics which can be used to compare and assess flow solver performance. The purpose of this procedure is to guide future research and development efforts and also to improve understanding of program performance for both users and developers. The intent of this paper is to present the pertinent flow solver statistics in such a way that the sources of CPU time are evident and that the relative advantages of different algorithms can be assessed in a meaningful way. It is our objective to develop an approach that can be applied to evaluate the efficiency benefits of higher-order and adaptive approaches as well, although such methods are not included here.

The flow solver studied in this paper is known as Diablo. It is based on a parallel implicit Newton-Krylov-Schur algorithm. Diablo uses a finite-difference discretization with summation-by-parts operators and simultaneous approximation terms [2–7], referred to as an SBP-SAT discretization, and is used to solve external aerodynamic flows in two or three spatial dimensions. The equations solved in this paper are the compressible Reynolds-averaged Navier-Stokes equations

---

[1] PhD candidate, Student Member AIAA

[2] Professor and Director, J. Armand Bombardier Foundation Chair in Aerospace Flight, Associate Fellow AIAA

with the Spalart and Allmaras [8] one-equation turbulence model (RANS-SA). It should be stressed that this is a research code written by students, and although the algorithms are written to be efficient, the code is not highly optimized for computational efficiency.

Diablo originated as an Euler (inviscid) solver which was developed by Hicken and Zingg [9]. It was augmented to the full Navier-Stokes equations by Osusky et al. [10] and to the RANS-SA equations by Osusky and Zingg [11, 12]. The Diablo RANS-SA flow solver has been validated at the Fifth AIAA Drag Prediction Workshop [13, 14], where the focus was on the three-dimensional wing-body geometry referred to as the NASA Common Research Model (CRM). In addition, several supplemental two-dimensional cases were investigated for solver accuracy, all from the Turbulence Modeling Resource (TMR) website:* flow over a flat plate, flow over a bump in channel, and flow over the NACA 0012 airfoil [14].

When the grid spacing of a family of grids is small enough such that the flow solution obtained on the grids is in the asymptotic region of grid convergence, the accuracy of the lift and drag functionals can be estimated using Richardson extrapolation as long as at least three grid levels are used [15], where each grid in the grid family, except for the finest, has been coarsened by a common factor from the previous grid level [16, 17]. This method is useful for analysis because it facilitates an estimate of the solution accuracy at a given mesh level. As the grid is refined, the lift and drag functionals of interest can be computed with reduced numerical error but at increased computational cost. The emphasis of the current study is on the relationship between accuracy and computational cost. A methodology for quantifying this relationship will be presented.

The main benchmark cases investigated in the current study are the steady RANS-SA equations solved in two spatial dimensions on the NACA 0012 airfoil at Mach 0.15, Reynolds number 6 million, and several angles of attack. Since Diablo is not written to handle two-dimensional cases very efficiently, nor low Mach number flows, and because the code suffers from programming inefficiencies, it is important to emphasize that the focus of this study is on the methodology and not the final data. To give some examples of the performance of Diablo for three-dimensional transonic cases, performance studies for the transonic ONERA M6 case and transonic CRM wing-body case from Osusky and Zingg [18] are also included.

The NACA 0012 grids used for the main benchmark cases are publicly available from the TMR website. Three grid families are available on this website, the main difference being in the trailing-edge spacing. While the effect of the trailing-edge spacing on flow solver performance is minimal, results obtained by previous researchers using the established flow solvers CFL3D [19] and FUN3D [20] have demonstrated that the different trailing-edge spacings have a noticeable impact on the accuracy of the lift, drag, and moment coefficients and also the minimum grid spacing required to be considered in the asymptotic region of grid convergence. This will affect the relationship between the error and computational cost. Different numerical dissipation models are also investigated for their effect on flow solution accuracy and convergence to steady state.

II. Governing Equations and Spatial Discretization

This section presents the continuous version of the Navier-Stokes equations and the Spalart-Allmaras turbulence model. The spatial discretization is also presented but in minimal detail.

[] * Turbulence Modeling Resource, http://turbmodels.larc.nasa.gov, Langley Research Center, United States

A. Navier-Stokes Equations

The three-dimensional Navier-Stokes equations under the coordinate transformation $(x, y, z) \rightarrow (\xi, \eta, \zeta)$ are given by Pulliam and Zingg [21]:

$$\partial_t \hat{q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} + \partial_\zeta \hat{G} = \frac{1}{\mathrm{Re}} \left(\partial_\xi \hat{E}_v + \partial_\eta \hat{F}_v + \partial_\zeta \hat{G}_v \right), \tag{1}$$

where

$$\hat{q} = J^{-1} q,$$

$$\hat{E} = J^{-1} \left(\xi_x E + \xi_y F + \xi_z G \right), \ \hat{F} = J^{-1} \left(\eta_x E + \eta_y F + \eta_z G \right), \ \hat{G} = J^{-1} \left(\zeta_x E + \zeta_y F + \zeta_z G \right),$$

$$\hat{E}_v = J^{-1} \left(\xi_x E_v + \xi_y F_v + \xi_z G_v \right), \ \hat{F}_v = J^{-1} \left(\eta_x E_v + \eta_y F_v + \eta_z G_v \right), \ \hat{G}_v = J^{-1} \left(\zeta_x E_v + \zeta_y F_v + \zeta_z G_v \right),$$

$$J = \left(x_\xi y_\eta z_\zeta + y_\xi z_\eta x_\zeta + z_\xi x_\eta y_\zeta - x_\xi z_\eta y_\zeta - y_\xi x_\eta z_\zeta - z_\xi y_\eta x_\zeta \right)^{-1}, \tag{2}$$

$$\mathrm{Re} = \frac{\rho_\infty a_\infty l}{\mu_\infty}, \tag{3}$$

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ u \left(e + p \right) \end{bmatrix}, \ F = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v w \\ v \left(e + p \right) \end{bmatrix}, \ G = \begin{bmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ w \left(e + p \right) \end{bmatrix}, \tag{4}$$

$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ E_{v,5} \end{bmatrix}, \ F_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ F_{v,5} \end{bmatrix}, \ G_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ G_{v,5} \end{bmatrix}, \tag{5}$$

$$q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}. \tag{6}$$

In the above equations: ρ is the density, a is the speed of sound, e is the energy, p is the pressure, l is the mean chord length, $\mathbf{u} = (u, v, w)$ are the Cartesian velocity components, $\tau = \tau \left(u, v, w, \mu, \mu_t \right)$ are the viscous stresses given by the Newtonian stress tensor, Re is the Reynolds number, and $\mu = \mu \left(a \right)$ is the viscosity given by Sutherland's law as:

$$\mu = \frac{a^3 \left(1 + S^* / T_\infty \right)}{a^2 + S^* / T_\infty}, \tag{7}$$

where $S^* = 198.6°\mathrm{R}$ is Sutherland's constant. The turbulent viscosity $\mu_t = \mu_t \left(\rho, \mu, \tilde{\nu} \right)$, where $\tilde{\nu}$ is the turbulence variable, is added to μ in the case of turbulent flows. The subscript ∞ indicates the free-stream value of a quantity. Explicit expressions for E_v, F_v, and G_v are omitted here but are

given by Pulliam and Zingg [21] or Osusky and Zingg [12].

Non-dimensional variables are used: the density is non-dimensionalized by ρ_∞, the velocities by a_∞, the viscosity by μ_∞, the temperature by T_∞, and the spatial coordinates by l. Assuming that the flow behaves as an ideal gas, the pressure can be written in terms of energy and velocity:

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right), \tag{8}$$

reducing the effective number of variables to five for the Euler and Navier-Stokes equations.

B. Turbulence Model

The effect of turbulence is included through the Spalart-Allmaras turbulence model. The standard form of the equation is used, given here in Cartesian coordinates:

$$\begin{aligned}\frac{\partial\tilde{\nu}}{\partial t} + u\frac{\partial\tilde{\nu}}{\partial x} + v\frac{\partial\tilde{\nu}}{\partial y} + w\frac{\partial\tilde{\nu}}{\partial z} = \frac{c_{b1}}{\mathrm{Re}}\left(1 - f_{t2}\right)\tilde{S}\tilde{\nu} + \frac{1 + c_{b2}}{\sigma\mathrm{Re}}\nabla\cdot\left[\left(\nu + \tilde{\nu}\right)\nabla\tilde{\nu}\right] - \frac{c_{b2}}{\sigma\mathrm{Re}}\left(\nu + \tilde{\nu}\right)\nabla^2\tilde{\nu} \\ - \frac{1}{\mathrm{Re}}\left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + \mathrm{Re}f_{t1}\Delta U^2,\end{aligned} \tag{9}$$

where ν is the kinematic viscosity, and $\tilde{\nu}$ will be referred to simply as the turbulence variable. Many of the terms above are functions of $\tilde{\nu}$ or other state variables. More details of the SA model and the discretization used in this paper, including boundary conditions, are available from Osusky and Zingg [12].

C. Spatial Discretization

The RANS-SA equations are discretized on single and multi-block structured grids using a finite-difference discretization with summation-by-parts operators and simultaneous approximation terms (SATs) to weakly enforce the boundary conditions on the domain boundaries and to couple the system across block interfaces [2–5, 7]. All of the results presented were computed using spatially second-order accurate SBP operators, except for the advection term of the SA model for which a first-order upwinding scheme is used. Osusky [22] has investigated the use of a third-order dissipation scheme but found that the accuracy benefit was not significant and that the convergence time increased significantly.

The SAT approach minimizes the amount of information that needs to be communicated between processors when the algorithm is parallelized. Additionally, the fact that this discretization does not need to form any derivatives across block interfaces reduces the continuity requirement for meshes at interfaces. In fact, only C^0 continuity is necessary for grid lines at interfaces, allowing for the algorithm to provide accurate solutions even on grids with slope changes at block interfaces.

Numerical dissipation is added to the discrete flow equations for numerical stability. The two types of dissipation commonly applied in the Diablo algorithm are the scalar dissipation model developed by Jameson et al. [23] and later refined by Pulliam [24] and the matrix dissipation model of Swanson and Turkel [25].

The advantage of scalar dissipation is that it tends to result in a more stable algorithm, but it is also more dissipative, resulting in higher error. Since the numerical dissipation vanishes in the limit of the grid spacing vanishing, both dissipation models will give the same grid converged solution. Hence, scalar and matrix dissipation give similar results on fine meshes.

III. Solution Methodology

The flow solver uses a parallel Newton-Krylov-Schur algorithm for steady three-dimensional flows. The distributed Schur complement [26] parallel preconditioner uses ILU(p) [27] internally, and the linear systems are solved iteratively using the Krylov solver FGMRES [27]. The ILU(p) factorization is built from a nearest-neighbour approximation to the Jacobian matrix where the fourth-difference dissipation is approximated with a second-difference dissipation operator, and the cross-derivatives in the viscous shear stresses are neglected.

With the use of the Krylov solver, Newton iterations are solved inexactly, so the method is referred to as an inexact Newton method. Since Newton's method will usually not converge unless a suitable starting guess is given, iterations are commenced using a pseudo-transient iterative method which is used to reduce the residual between 4 and 6 orders of magnitude before switching to the inexact Newton phase. These values are based on parameter studies performed by Osusky [22]. While he has found that a two order of magnitude reduction can be sufficient for some 2D cases, he has advocated the use of more conservative values based on experience. We have found that at least four orders of magnitude are needed for the NACA 0012 cases studied in this paper. For three dimensional cases, Osusky has found that an efficient choice for the switching tolerance is generally around a 4 or 5 order of magnitude residual reduction. Four orders of magnitude was sufficient for the three-dimensional results analyzed in this paper. In general, cases with finer grid spacing or cases involving shocks can require smaller values of this tolerance than subsonic cases on coarser grids.

A. Inexact Newton Method

Consider a nonlinear system of algebraic equations, represented by

$$\mathcal{R}\left(\mathbf{q}\right) = \mathbf{0}, \tag{10}$$

$$\mathcal{R} : \mathbb{R}^M \rightarrow \mathbb{R}^M, \ \mathbf{q} \in \mathbb{R}^M.$$

In the context of CFD, this system of equations represents the discrete residual. Then Newton's method is given by:

$$\mathcal{A}^{(n)} \, \Delta\mathbf{q}^{(n)} = -\mathcal{R}\left(\mathbf{q}^{(n)}\right), \tag{11}$$

$$\Delta\mathbf{q}^{(n)} \equiv \mathbf{q}^{(n+1)} - \mathbf{q}^{(n)},$$

where $\mathcal{A}^{(n)} : \mathbb{R}^M \rightarrow \mathbb{R}^M$ is the Jacobian of $\mathcal{R}\left(\mathbf{q}\right)$. Since the linear system is being solved to some relative tolerance $\eta^{(n)} \in \mathbb{R}$, the actual Newton step is taken inexactly:

$$\left\| \mathcal{R}\left(\mathbf{q}^{(n)}\right) + \mathcal{A}^{(n)} \, \Delta\mathbf{q}^{(n)} \right\| \leq \eta^{(n)} \left\| \mathcal{R}\left(\mathbf{q}^{(n)}\right) \right\|. \tag{12}$$

B. Pseudo-Transient Continuation

Obtaining an initial iterate for Newton's method can be accomplished using a globally convergent algorithm, which will typically have a lower convergence rate than Newton's method. Such algorithms are known as continuation methods.

Pseudo-transient continuation is a pseudo-time-marching method, i.e. it is an imitation of physical time-marching, though time-accuracy is not required. The update formula at the n-th iteration is given by the implicit Euler method with local time linearization [28]:

$$\left(\mathcal{T}^{(n)} + \mathcal{A}^{(n)}\right) \Delta\mathbf{q}^{(n)} = -\mathcal{R}\left(\mathbf{q}^{(n)}\right), \tag{13}$$

where $\mathcal{T}^{(n)} = \frac{1}{\Delta t}\mathcal{I}$, and $\mathcal{I}$ is the identity matrix. Since time-accuracy is not required in the context of globalization, Δt can take large values and vary spatially. In this study, Δt is evolved according to:

$$\Delta t_i^{(n)} = T_i a \left(b\right)^{m\left\lfloor \frac{n}{m} \right\rfloor}, \; T_i = \frac{1}{1 + J_i^{\frac{1}{D}}}, \tag{14}$$

where J is the geometric Jacobian resulting from the coordinate transformation on the mesh, i is the grid node index, D is the number of spatial dimensions (either 2 or 3 in this paper), and $\lfloor \cdot \rfloor$ is the floor operator*. The floor operator is present in the formula in the case where we update the preconditioner every m iterations instead of every iteration. Applying this technique can reduce the overall convergence time of the algorithm. However, the value of m was set to unity for all cases in this paper. The values of a and b used in the current study are based on Osusky and Zingg [12] and the explicit values used in the NACA 0012 studies are reported in Section V. The values taken by these parameters can generally be chosen more conservatively for more difficult cases, such as cases on finer grids or cases where shocks are present.

The pseudo-transient continuation phase is terminated and the inexact-Newton phase initiated when the relative residual

$$\mathcal{R}_{\text{rel}}^{(n)} \equiv \frac{\left\| \mathcal{R}\left(\mathbf{q}^{(n)}\right) \right\|}{\left\| \mathcal{R}\left(\mathbf{q}^{(0)}\right) \right\|} \tag{15}$$

is reduced below some user-specified tolerance μ_{rel}.

C. Matrix-Vector Products

When using a Krylov solver such as FGMRES, it is not necessary to calculate and store $\mathcal{A}^{(n)}$ since at no point in the algorithm is an explicit expression for $\mathcal{A}^{(n)}$ required, and this matrix can be expensive in terms of data storage. The Krylov solver does, however, require an approximation to the matrix-vector product $\mathcal{A}^{(n)}\mathbf{v}$, $\mathbf{v} \in \mathbb{R}^M$. There are several ways in which this matrix-vector product can be approximated without forming the full Jacobian.

One option is to use finite-difference matrix-vector products:

$$\mathcal{A}^{(n)}\mathbf{v} \approx \frac{\mathcal{R}\left(\mathbf{q}^{(n)} + \epsilon\mathbf{v}\right) - \mathcal{R}\left(\mathbf{q}^{(n)}\right)}{\epsilon}, \tag{16}$$

where $\epsilon \in \mathbb{R}$ is the perturbation parameter and is chosen to balance between truncation error and round-off error. Based on Nielsen et al. [20], the following formula is used in this work:

$$\epsilon = \sqrt{\frac{M\delta}{\mathbf{v}^T\mathbf{v}}},$$

where $\delta \in \mathbb{R}$ is a value near to machine precision ($\delta = 10^{-13}$ is used in all cases in this study). A second option is to recycle the nearest-neighbour approximate Jacobian used to form the ILU preconditioner. This approximate Jacobian uses a small-stencil approximation to the third-order dissipation term, ignores the cross-derivative terms in the discretization of the viscous terms, and ignores the differentiation of the pressure sensor used for shock-capturing. Matrix-vector products using the approximate Jacobian will be referred to as approximate matrix-vector products.

Using the approximate Jacobian matrix to compute matrix-vector products is less accurate than using finite-differencing but comes at reduced computational cost. In the inexact Newton

---

[] * $\lfloor x \rfloor = y$, where y is the largest integer less than or equal to x.

phase it is beneficial to use the finite-differencing method. However, either type of matrix-vector product can be used during the globalization phase. We have generally found that using approximate matrix-vector products in the continuation phase gives faster convergence than the finite-differencing method without sacrificing robustness.

IV. Performance Analysis

Performance is analyzed in terms of both the accuracy of the discretization and the efficiency of the flow solver at obtaining the steady solution on various mesh levels and at different operating conditions.

A. The High Performance Computing System

All computations were performed on the SciNet General Purpose Cluster (GPC), hosted in Toronto, Ontario, Canada. The GPC consists of 3,780 nodes (IBM iDataPlex DX360M2) with a total of 30,240 cores (Intel Xeon E5540) at 2.53GHz, with 16GB RAM per node (2GB per core).

B. Estimating Grid-Converged Functionals

The grid-converged functionals are the values of the functionals for the theoretical case of infinite mesh resolution. These values can be obtained by applying Richardson extrapolation to the three finest grid levels on which the functionals have been calculated. This procedure is described in detail by Roache [29].

To apply Richardson extrapolation, the convergence rate $p \in \mathbb{R}$ must first be calculated from the three finest converged functional values:

$$p = \frac{\ln\left(\left|f_3 - f_2\right| / \left|f_2 - f_1\right|\right)}{\ln\left(r\right)}, \tag{17}$$

where f_i is the functional of interest at grid level i (1 being the finest and 3 the coarsest) and $r \in \mathbb{R}$ is the grid ratio, which is equal to 2 for all studies in this paper. The convergence rate is then used to extrapolate the functional value on an infinite resolution mesh, denoted f_0, using the two finest grid levels:

$$f_0 = \frac{r^p f_1 - f_2}{r^p - 1}. \tag{18}$$

The three-grid Richardson extrapolation method is not reliable unless it is applied in the asymptotic region [29]. If the final three points are non-monotonic, or if the estimated convergence rate is less than 1, then the grid-converged functional is estimated from a first-order extrapolation by setting $p = 1$ in equation (18).

C. Benchmarking Flow Solver Performance

The purpose of this study is to quantify the computational cost associated with specific levels of functional accuracy. Osusky and Zingg [18] have previously performed performance analysis on the Diablo RANS-SA flow solver, and a similar approach will be taken here. Osusky and Zingg [18] also discuss the pros and cons of various cost measures for comparing solvers.

Measuring performance from the CPU time required to complete a flow solve can be useful for comparing the relative performance of a flow solver under different conditions, but since performance will depend on the computer system, this is not a useful measure for comparing against the performance of other CFD codes run on a different computer system. One way to

Table 1: Grid details for the NACA 0012 geometry. OW=Off-Wall, TE=Trailing Edge. Spacings measured in chord units (c).

		Family I		Family II		Family III	
Level	Grid Size	OW Spacing	TE Spacing	OW Spacing	TE Spacing	OW Spacing	TE Spacing
1	7169×2049	1.000×10^{-7}	1.25×10^{-4}	1.000×10^{-7}	1.25×10^{-5}	1.000×10^{-7}	3.75×10^{-5}
2	3585×1025	2.009×10^{-7}	2.50×10^{-4}	2.009×10^{-7}	2.50×10^{-5}	2.009×10^{-7}	7.50×10^{-5}
3	1793×513	4.057×10^{-7}	5.00×10^{-4}	4.056×10^{-7}	5.00×10^{-5}	4.056×10^{-7}	1.50×10^{-4}
4	897×257	8.270×10^{-7}	1.00×10^{-3}	8.265×10^{-7}	1.00×10^{-4}	8.267×10^{-7}	3.00×10^{-4}
5	449×129	1.719×10^{-6}	2.00×10^{-3}	1.716×10^{-6}	2.00×10^{-4}	1.717×10^{-6}	6.00×10^{-4}
6	225×65	3.716×10^{-6}	4.00×10^{-3}	3.706×10^{-6}	4.00×10^{-4}	3.711×10^{-6}	1.20×10^{-4}
7	113×33	8.736×10^{-6}	8.00×10^{-3}	8.685×10^{-6}	8.00×10^{-4}	8.708×10^{-6}	2.40×10^{-4}

eliminate this problem is to use a benchmark such as the TauBench codes* to non-dimensionalize cost, as was done by Wang et al. [30]. The TauBench code roughly simulates the CPU cost of running the flow solver Tau for a mesh of a user-specified size, number of processors, and number of iterative steps.

The benchmark that was used for all cases in this paper is 2.50×10^5 nodes on 1 processor with 10 iterative steps. The average CPU time after running the TauBench code four times on the SciNet general purpose cluster was 9.571s, which will be referred to as one *work unit*. The convention is to measure wall time rather than processor time. The total CPU time can be measured by multiplying the wall time by the number of processors used.

V. NACA 0012 Results

Three nested families of grids are provided by the TMR website for the modified NACA 0012 geometry. All grids are structured C-topology grids with the far-field boundary located 500 chords from the airfoil. The main difference between the three grid families is the trailing-edge spacing.

Since all grid families are structured grids consisting of the same number of nodes, refining the grid in the trailing-edge region has the effect of coarsening the grid elsewhere. Aside from these differences, the three grid families are very similar. The finest (level 1) mesh for all three families consists of 7196×2049 nodes, with 4097 points along the airfoil surface. Coarser meshes are generated from the finer ones by removing every node with an even index number in each direction. Grid details can be found in Table 1.

All flow solutions are computed at a Reynolds number of 6 million based on the chord length with freestream Mach number and temperature of 0.15 and 540R respectively. The angles of attack of interest are $\alpha = 0$, 10°, and 15°. A far-field vortex correction is not used for any cases in this paper. The additional error incurred by using scalar dissipation instead of matrix dissipation can also be assessed by comparing the functionals calculated at various levels of grid convergence. However, as discussed previously, the choice of numerical dissipation should not affect the grid converged solution.

A. Grid Convergence

The grid convergence data is plotted in Figures A.1 through A.3 in the Appendix. Tables A.1 through A.3 in the Appendix show the estimated grid-converged values for all three grid families

[] * Taubench Version 1.1, IPACS, http://www.ipacs-benchmark.org, DLR, Germany

and all three sets of operating conditions calculated from mesh levels 2 through 4. The extrapolated functional values from Diablo, FUN3D, and CFL3D are in fairly good agreement where the data is available for all three flow solvers. However, Diablo generally seems to lose more accuracy on the coarser meshes, particularly with scalar dissipation.

As explained previously, Richardson extrapolation was used by applying equation (18) with the predicted convergence rate calculated from equation (17) in all cases where the functional values on these grid levels were monotonic and the convergence rate calculation gave $p > 1$. In cases where the convergence rate p was calculated to be less than or equal to 1, or if the functional values at the three finest grid levels were non-monotonic, a first-order extrapolation was used to predict the grid-converged values by applying equation (18) with convergence rate $p = 1$. The extrapolated values for cases where $p < 1$ are shown in Table A.4.

Due to the presence of the trailing-edge singularity, the design order of convergence is difficult to predict. Therefore it can be unclear when the asymptotic region is reached, and the use of Richardson extrapolation must be assessed carefully. Qualitative analysis of Figures A.1 through A.3 indicates that the extrapolated data fit well with the established pattern of the calculated functional values. Furthermore, the extrapolated values obtained by Diablo, FUN3D, and CFL3D, where available, have generally been found to be in better agreement than the values calculated directly on the finest mesh. Based on these observations, it is our assessment that the extrapolated values are better estimates of the theoretical grid-converged values than those calculated directly on the finest mesh level.

B. Accuracy

Performance studies were carried out with pseudo-time-stepping parameters $a = 0.001$ and $b = 1.35$ for most of the NACA 0012 cases based on the recommendations of Osusky and Zingg [12], which are based on the parameter studies of Osusky [22]. The only exception was the angle of attack $\alpha = 0^\circ$ case at mesh level 3, for which the value of b was reduced to 1.25 for Family I and 1.3 for both Family II and Family III due to stalled convergence for this particular case at the more aggressive value of b. The reason this problem was encountered for this case is because the reference time step t_{ref} became too large late in the approximate Newton phase due to the large number of iterations that were taken in this phase.

The residual drop parameter μ_{rel} used for switching to the inexact Newton phase is set to $\mu_{\text{rel}} = 10^{-4}$ for the $\alpha = 10^\circ$ and $\alpha = 15^\circ$ cases for all grid levels and all grid families. This parameter was set to $\mu_{\text{rel}} = 10^{-5}$ for the $\alpha = 0^\circ$ case at all grid levels and for all grid families. The more conservative tolerance may not have been necessary for all $\alpha = 0^\circ$ cases but was applied based on the convergence failure of a few cases, which stalled after switching to the inexact Newton phase, indicating that a suitable starting guess for Newton's method was not reached in those cases.

The ILU(p) fill level was set to $p = 2$ for the approximate Newton phase and $p = 4$ for the inexact Newton phase for all cases. While the optimal value of p is generally smaller on coarse grids than fine grids, we did not optimize this parameter for each case as such tuning is not performed in practice and it seemed more appropriate for the analysis to use consistent parameter settings as much as possible for all cases. This may result in some modest inefficiency for the cases on the coarser grid levels.

An understanding of how the error relates to the mesh spacing is useful for mesh design and is the first subject investigated. Figures 1 and 2 show the relationship between the numerical error in the functional estimates and the mesh spacing. Since the average mesh spacing is proportional to $\sqrt{1/N}$ for two-dimensional cases under uniform mesh refinement, where N is the number of grid points, the average mesh spacing h is represented as $\sqrt{1/N}$. The numerical errors in the calculated C_d, C_l, and C_m values presented in the figures were estimated based on the extrapolated values obtained from the grid convergence study individually for each grid family for each α. These

extrapolated values are reported in Tables A.1 through A.3.

It is apparent from Figures 1 and 2 that the accuracy depends not only on the mesh spacing but the grid family. The different functionals also exhibit different error profiles as the mesh is refined. The level of mesh refinement required depends on the level of accuracy desired. The lift coefficient is calculated to within 5% accuracy even on the coarsest mesh. The drag coefficient is more sensitive to mesh refinement. At mesh level 5, the numerical error in C_d is in excess of 10% for all grid families at $\alpha = 15°$. At mesh level 4 the numerical error is reduced to around 4%. The C_m is the most sensitive to the grid family. With grid Family I, the error in C_m is estimated to be around 10% even at grid level 3 for $\alpha = 10°$. However, the C_m appears to be estimated much more accurately for grid Family II; the error estimates are already below 2% by grid level 5 for both $\alpha = 10°$ and $\alpha = 15°$.

C. Flow Solver Performance

When the mesh is refined by a factor of two in each direction, the computational cost will clearly increase since there are four times more nodes in the computational domain (in 2D) and therefore roughly four times more floating point operations for each residual evaluation. However, the total cost of the flow solve will increase by more than a factor of four because the linear system conditioning will worsen and the nonlinear problem can also become more difficult to converge. It is of practical interest to quantify how much extra cost is incurred by the flow solver to attain a certain level of accuracy.

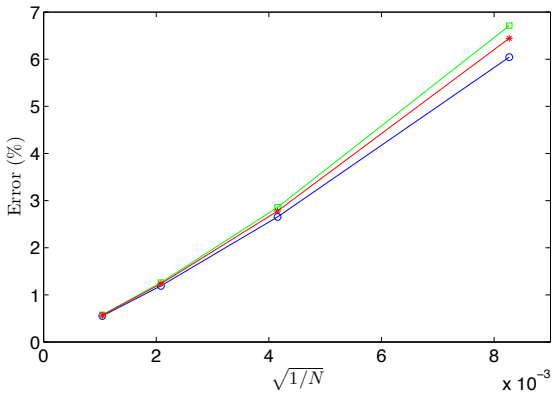
The slower convergence rate of both the linear solver and nonlinear iterations for finer grids can be observed by plotting the residual history against the number of linear iterations taken by FGMRES, as shown in Figure 3. This is a way to investigate the effect of grid refinement on the linear system and nonlinear iterative methods without considering CPU time. It is also noteworthy that the zero angle of attack case takes the most iterations, particularly for the level 3 Family I grid, though the reason for this is unclear.

A plot of the CPU cost of reducing the L^2-norm of the residual by eleven orders of magnitude versus the number of grid nodes is shown in Figure 4. It can be observed from the plots that performance is similar for the three grid families. One observation that stands out when comparing Figure 4 with Figure 3 is that the CPU time is not noticeably higher for the $\alpha = 0°$ case than it is for the other two angles of attack (with the exception of the level 3 Family I grid), despite taking significantly more linear and nonlinear iterations. The reason for this is because the extra iterations occurred mainly in the pseudo-transient phase which makes use of much cheaper matrix-vector products. This emphasizes an important point, which applies to both linear and nonlinear iterations: the cost of an iteration can vary substantially throughout the flow solve and the number of iterations is not necessarily proportional to CPU time.

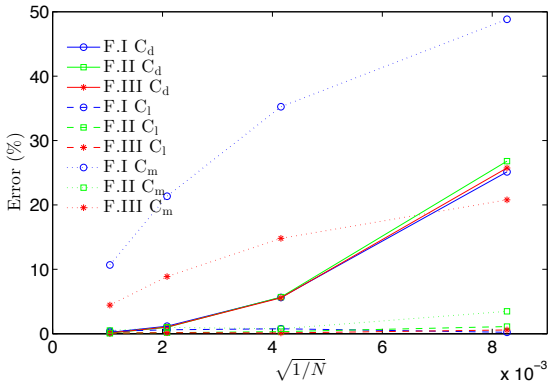
Figure 5 shows the number of equivalent residual evaluations required to reduce the L^2-norm of the residual by eleven orders of magnitude plotted against the number of grid nodes, where an equivalent residual evaluation is the total CPU time divided by the CPU time required to calculate the residual vector once. This can alleviate some of the timing inconsistency incurred by running different flow solvers on different processors. Also, since the cost of the residual vector is expected to be roughly proportional to the number of grid nodes*, any increase in the number of equivalent residual evaluations is mainly due to an increased number of linear and nonlinear iterations.

Figure 6 shows the work units per grid node required to reduce the norm of the residual by eleven orders of magnitude on each mesh level. The figure would look very similar to Figure 5

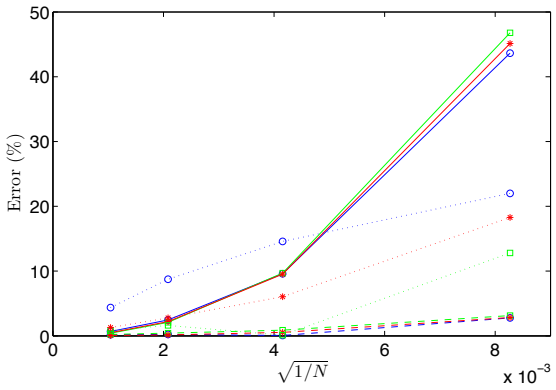
[] * For these NACA 0012 cases the cost of the residual generally increases by a factor of between 4.0 and 4.4 at each grid refinement.



a) $\alpha = 0°$

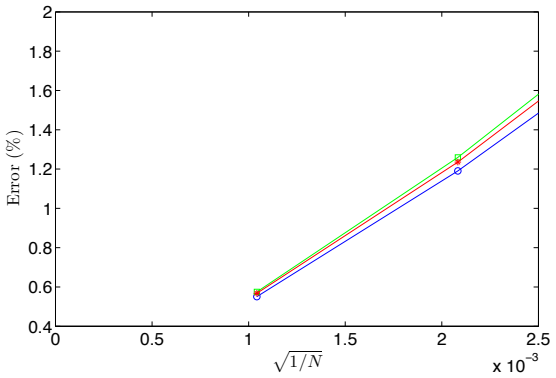


b) $\alpha = 10°$

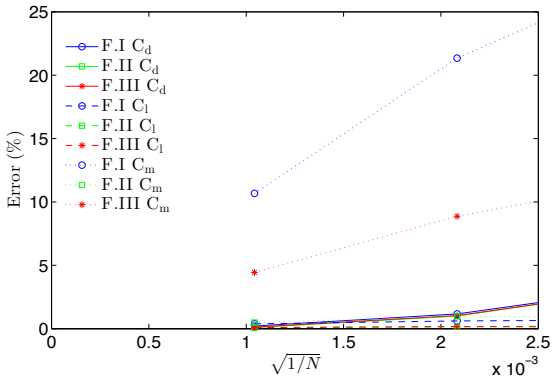


c) $\alpha = 15°$

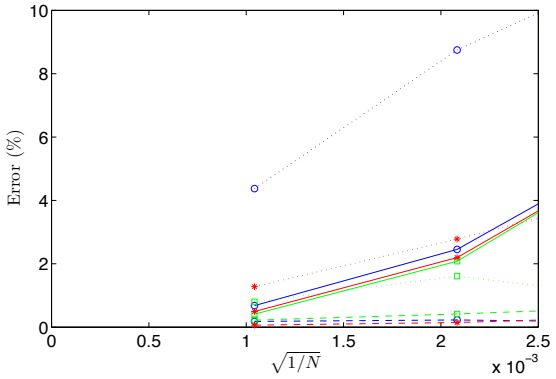
Fig. 1: Error in the C_d and C_l values calculated at the converged flow solution at grid levels 3 to 6 for the NACA 0012 cases. Results are shown for all three grid families (F.I=Family I, for example) using matrix dissipation.



a) $\alpha = 0°$



b) $\alpha = 10°$



c) $\alpha = 15°$

Fig. 2: The same as Figure 1 but with more detail visible.

except that both axes of the plot are logarithmic, and a line of best fit is included instead of simply connecting the data points. This leads to a rather interesting and useful analysis which can be performed. If the CPU time per grid node is represented by t/N and the number of nodes is N, then the following model might be assumed relating the CPU time per grid node to the number of nodes:

$$t/N = \kappa N^{\beta}. \tag{19}$$

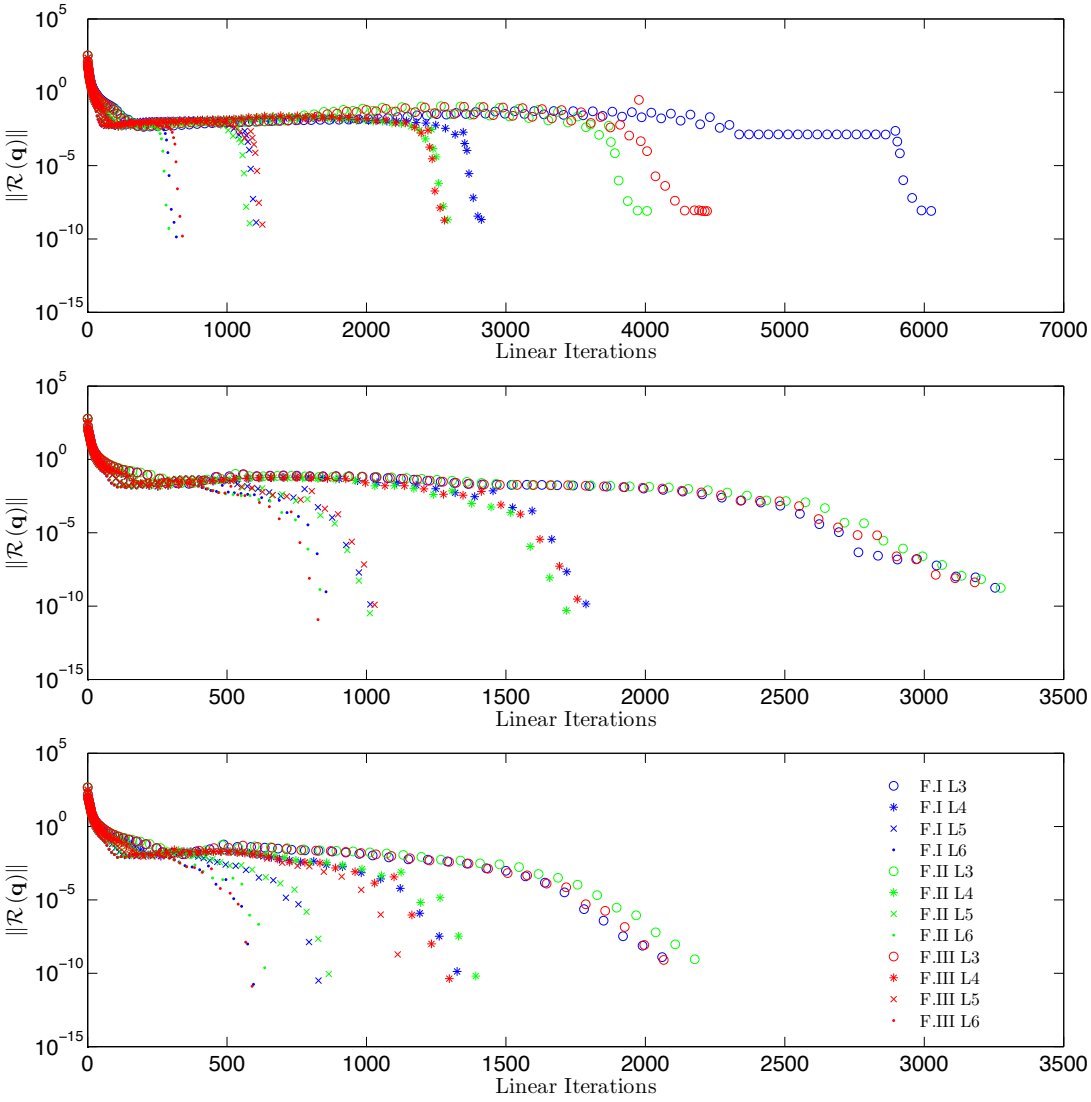


Fig. 3: Residual history plotted against the number of linear iterations for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation. The legend entries describe the family and grid level (L3=level 3, L4=level 4, etc.). From top to bottom, the angles of attack are 0°, 10°, and 15°. Each marker on the plot represents one iteration of either Newton's method or the pseudo-transient method.

In this equation, κ and β are unknowns. The parameter β is a metric relating computational cost to mesh size - if $\beta = 0$ then the code exhibits perfect linear scaling with respect to the number of grid points. This parameter is estimated as the slope of the best fit lines shown in Figure 6. Estimates for β were calculated for each case and each grid family and are shown in Table 2. They range from 0.2295 to 0.4623.

A similar study was performed by Pueyo and Zingg [31] using the flow solver known as Probe. Probe is a two-dimensional flow solver which uses the same finite-difference discretization as ARC2D [32], the fundamental difference being that Probe uses a Newton-Krylov algorithm to solve the discrete flow equations whereas ARC2D uses an approximate factorization algorithm. Both Probe and ARC2D use the Baldwin-Lomax [33] algebraic model. The test case is explicitly-tripped flow over the RAE 2822 airfoil at Mach 0.729, $\alpha = 2.31°$, and Re $= 6.50 \times 10^6$. Pueyo and Zingg [31] calculated $\beta = 0.325$ for Probe and $\beta = 0.73$ [34] for ARC2D for this case. Their

Table 2: Values of the parameter β from equation (19)

	Angle of Attack		
	0°	10°	15°
Family I	0.4623	0.2295	0.4297
Family II	0.2833	0.2335	0.2677
Family III	0.3163	0.2487	0.4289

calculation was based on equivalent residual evaluations instead of CPU time per grid node. This will only result in the same β if their residual calculation scales perfectly with grid size. Since the cost of the residual in Diablo tends to increase slightly with mesh size, calculating β in terms of relative residual evaluations would result in slightly smaller β values than those presented in Table 2.

The final study performed relates the accuracy obtained at each grid level to the cost of performing the flow solve. The results can be found in Figure 7, where the most accurate and most expensive points correspond to the finer grid levels. Such plots enable the assessment of a solver's efficiency, i.e., the cost associated with reducing numerical error below a specified threshold.

The non-monotonic behaviour of the C_l and C_m values as the mesh is refined may occur for two reasons:

1. These functionals take scalar values and are not a true representation of the solution error - the scalar values may match the accurate grid-converged functional estimates as a results of error cancellation even if the actual error in the flow solution is high.

2. The grid-converged functional values are estimated with limited accuracy. The non-monotonic behaviour is evident mainly when the error in the functional values is estimated to be quite low. Lack of precision in estimating the grid-converged functional estimates is most likely significant enough to affect the error estimates in this error regime.

VI. Three-Dimensional Results

To demonstrate the performance of the Diablo flow solver for three-dimensional cases in transonic flows, performance studies are also included for two cases from Osusky and Zingg [18]. The two cases selected are as follows:

1. Flow over the ONERA M6 wing at Reynolds number 1.1×10^7, Mach number 0.8395, and angle of attack 3.06°. The case was run on the SciNet general purpose cluster using 128 processors.

2. Flow over the NASA Common Research Model (denoted CRM-t2 in Osusky and Zingg [18]) at Reynolds number 5×10^6, Mach number 0.85, and angle of attack 2.229°. The case was run on the SciNet general purpose cluster using 832 processors.

The grid details are given in Table 3. The pseudo-time-stepping parameters used for the ONERA M6 case are $a = 0.001$ and $b = 1.3$, and the parameters used for the CRM case are $a = 0.001$ and $b = 1.25$. The switching tolerance for the inexact Newton phase μ_{rel} is set to 10^{-4} for both cases. For the CRM case, the ILU(p) fill level is set to $p = 2$ in the approximate Newton phase and $p = 3$ in the inexact Newton phase. For the ONERA M6 case, $p = 2$ is used for both phases.

Figure 8 shows the relationship between CPU time and error for these two cases, though it should be emphasized that the nominal error values obtained for these cases should be regarded as estimates, so the data in the figure are approximate and only recommended for qualitative analysis. The total CPU time is calculated by multiplying the wall time taken to reduce the flow residual eleven orders of magnitude by the total number of processors used. In addition, the relationship between CPU time and grid spacing is shown in Figure 9.

Table 3: Grid details for the ONERA M6 and NASA CRM grids

Grid	Blocks	Processors	Nodes, N	Average off-wall spacing (c)
ONERA M6 - 1	128	128	$35,152,000$	9.01×10^{-7}
ONERA M6 - 2	128	128	$4,599,936$	1.92×10^{-6}
ONERA M6 - 3	128	128	$628,824$	4.35×10^{-6}
CRM - 1	6656	832	$7,008,768$	5.13×10^{-6}
CRM - 2	6656	832	$3,261,440$	6.84×10^{-6}
CRM - 3	6656	832	$1,164,800$	1.03×10^{-5}

VII. Concluding Remarks

A methodology has been presented for comparing and assessing the performance of different flow solvers. The methodology focuses on characterizing the relationship between the computational cost of completing a flow solve, the grid refinement, and the accuracy of the lift and drag coefficients. Several metrics were used to measure the flow solve completion time, each with their relative merits. By analyzing the flow solver from several perspectives it is possible to develop an understanding of the flow solver which can help to guide future research directions. Specifically this methodology enables the assessment of the efficiency of high-order and adaptive methods.

The methodology was presented by demonstrating its application using the RANS-SA capabilities of the flow solver known as Diablo. The methodology was used to profile the performance of Diablo for three families of 2D NACA 0012 grids under several operating conditions. The performance of Diablo was also characterized for an ONERA M6 wing case as well as a CRM wing-body case, both under transonic flow conditions.

Acknowledgements

The authors gratefully acknowledge financial assistance from the Natural Science and Engineering Research Council (NSERC), the Canada Research Chairs program, and the University of Toronto. Computations were performed on the GPC supercomputer at the SciNet HPC Consortium. SciNet is funded by: the Canada Foundation for Innovation under the auspices of Compute Canada; the Government of Ontario; Ontario Research Fund - Research Excellence; and the University of Toronto.

The authors would also like to acknowledge the contributions of Dr. Michal Osusky who contributed to the quality and content of this paper through several relevant private communications. Also, Mr. Howard Buckley for his assistance with some of the early work investigating the NACA 0012 test cases. In addition, the authors have benefited from private communications with Drs. Boris Diskin and James Thomas.

References

[1] Brown, D. A., Buckley, H. P., Osusky, M., and Zingg, D. W., "Performance of a Newton-Krylov-Schur Algorithm for the Numerical Solution of the Steady Reynolds-Averaged Navier-Stokes Equations," *53rd AIAA Aerospace Sciences Meeting*, Kissimee, Florida, United States, January 2015, AIAA 2015-1744.

[2] Kreiss, H. and Scherer, G., "Finite element and finite difference methods for hyperbolic partial differential equations," *Mathematical Aspects of Finite Elements in Partial Differential Equations: proceedings of a symposium conducted by the Mathematics Research Center, the University of Wisconsin*, edited by C. de Boor, Mathematics Research Centre, the University of Wisconsin, Academic Press, 1974.

[3] Strand, B., "Summation by parts for finite difference approximations for d/dx," *Journal of Computational Physics*, Vol. 110, 1994, pp. 47–67.

[4] Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes," *Journal of Computational Physics*, Vol. 111, 1994, pp. 220–236.

[5] Nordström, J. and Carpenter, M. H., "Boundary and interface conditions for high-order finite-difference methods applied to the Euler and Navier-Stokes equations," *Journal of Computational Physics*, Vol. 148, 1999, pp. 621–645.

[6] Svärd, M. and Nordström, S., "On the order of accuracy for difference approximations of initial-boundary value problems," *Journal of Computational Physics*, Vol. 218, No. 1, 2006, pp. 333–352.

[7] Del Rey Fernández, D. C., Hicken, J. E., and Zingg, D. W., "Review of Summation-By-Parts Operators with Simultaneous Approximation Terms for the Numerical Solution of Partial Differential Equations," *Computers and Fluids*, Vol. 95, 2014, pp. 171–196.

[8] Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," January 1992, AIAA-92-0439.

[9] Hicken, J. E. and Zingg, D. W., "A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms," *AIAA Journal*, Vol. 46, No. 11, 2008, pp. 2773–2786.

[10] Osusky, M., Hicken, J. E., and Zingg, D. W., "A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations using the SBP-SAT approach," January 2010, AIAA-2010-116.

[11] Osusky, M. and Zingg, D. W., "A parallel Newton-Krylov-Schur flow solver for the Reynolds-Averaged Navier-Stokes equations," January 2012, AIAA-2012-0442.

[12] Osusky, M. and Zingg, D. W., "A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations discretized using summation-by-parts operators," *AIAA Journal*, Vol. 51, No. 12, 2013, pp. 2833–2851.

[13] Levy, D. W., Laflin, K. R., Tinoco, E. N., Vassberg, J. C., Mani, M., Rider, B., Rumsey, C., Wahls, R. A., Moorrison, J. H., Brodersen, O. P., Crippa, S., Mavriplis, D. J., and Murayama, M., "Summary of Data from the Fifth AIAA Drag Prediction Workshop," January 2013, AIAA-2013-0046.

[14] Osusky, M., Boom, P. D., and Zingg, D. W., "Results from the Fifth AIAA Drag Prediction Workshop Obtained with a Parallel Newton-Krylov Flow Solver Discretized Using Summation-by-Parts Operators," June 2013, AIAA-2013-2511.

[15] Baker, T. J., "Mesh Generation: Art or Science?" *Progress in Aerospace Sciences*, Vol. 41, No. 1, 2005, pp. 29–63.

[16] Zingg, D. W., "Viscous Airfoil Computations Using Richardson Extrapolation," January 1991, AIAA-91-1559-CP.

[17] Zingg, D. W., "Grid Studies for Thin-Layer Navier-Stokes Computations of Airfoil Flowfields," *AIAA Journal*, Vol. 30, No. 10, 1993, pp. 2561–2564.

[18] Osusky, M. and Zingg, D. W., "Steady three-dimensional turbulent flow computations with a parallel Newton-Krylov-Schur algorithm," January 2014, AIAA-2014-0242.

[19] Krist, S. L., Biedron, R. T., and Rumsey, C. L., "CFL3D Users Manual (Version 5.0)," Tech. rep., National Aeronautics and Space Administration, 1998, NASA TM 1998-208444.

[20] Nielsen, E. J., Anderson, W. K., Walters, R. W., and Keyes, D. E., "Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code," June 1995, AIAA-95-1733.

[21] Pulliam, T. H. and Zingg, D. W., *Fundamental Algorithms in Computational Fluid Dynamics*, Springer, 2014.

[22] Osusky, M., *A Parallel Newton-Krylov-Schur Algorithm for the Reynold-Averaged Navier-Stokes Equations*, Ph.D. thesis, University of Toronto, Toronto, Ontario, Canada, 2013.

[23] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler equations by Finite-Volume Methods using Runge-Kutta Time-Stepping Schemes," June 1981, AIAA-1981-1259.

[24] Pulliam, T. H., "Artificial Dissipation Models for the Euler Equations," *AIAA Journal*, Vol. 24, No. 12, December 1986, pp. 1931–1940.

[25] Swanson, R. C. and Turkel, E., "On central-difference and upwind schemes," *Journal of Computational Physics*, Vol. 101, No. 2, 1992, pp. 292–306.

[26] Saad, Y. and Sosonkina, M., "Distributed Schur complement techniques for general sparse linear systems," *SIAM Journal of Scientific Computing*, Vol. 21, 1999, pp. 1337–1357.

[27] Saad, Y., *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2nd ed., 2003.

[28] Lomax, H., Pulliam, T. H., and Zingg, D. W., *Fundamentals of Computational Fluid Dynamics*, Springer-Verlag, 2001.

[29] Roache, P. J., *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, New Mexico, 1998.

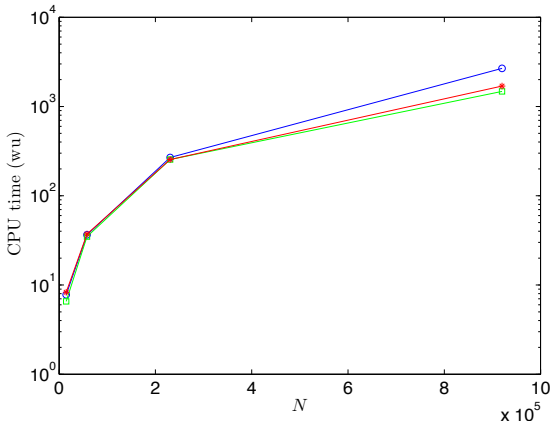
[30] Wang, Z. J., Fidkowski, K., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T., Kroll, N., May, G., Persson, P., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845.

[31] Pueyo, A. and Zingg, D. W., "Improvements to a Newton-Krylov solver for aerodynamic flows," January 1998, AIAA 98-0619.

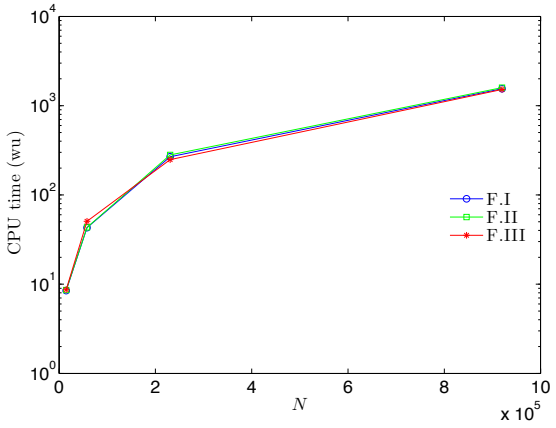
[32] Pulliam, T. H., "Efficient solution methods for the Navier-Stokes equations," Lecture Notes for the von Karman Institute for Fluid Dynamics Lecture Series, January 1986.

[33] Baldwin, B. S. and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," June 1978, AIAA 78-257.

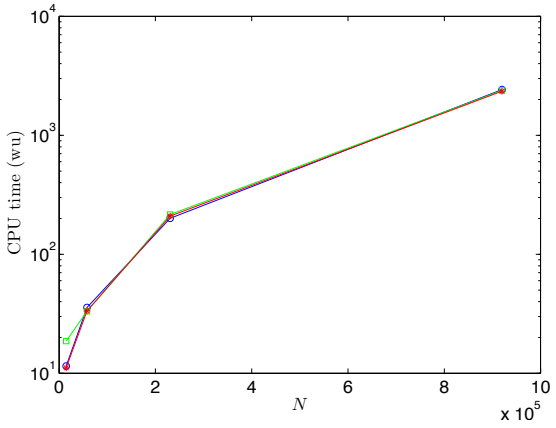
[34] Pueyo, A., *An Efficient Newton-Krylov Method for the Euler and Navier-Stokes Equations*, Ph.D. thesis, University of Toronto, Toronto, Ontario, Canada, 1998.



a) $\alpha = 0°$

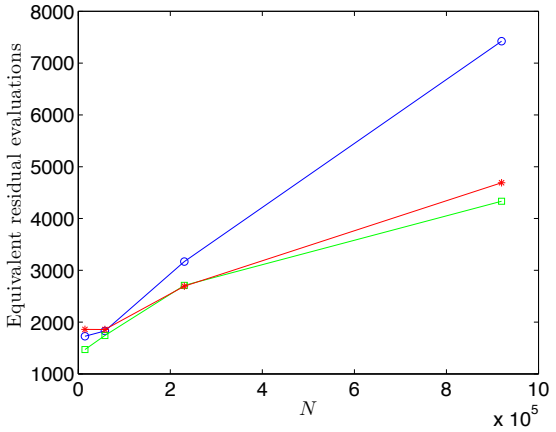


b) $\alpha = 10°$

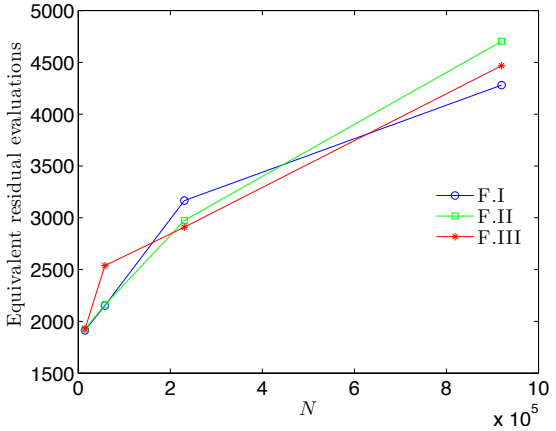


c) $\alpha = 15°$

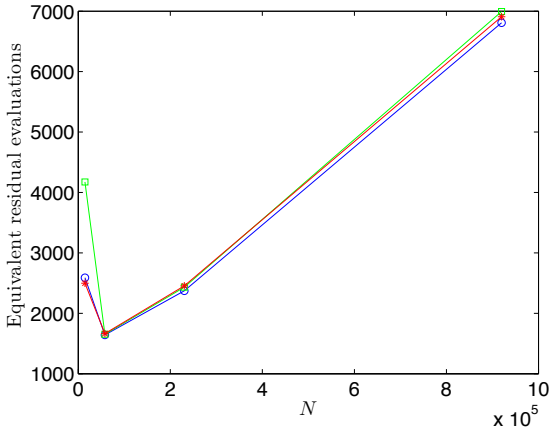
Fig. 4: Work units taken to reduce the L^2-norm of the residual by eleven orders of magnitude for mesh levels 3 to 6 for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation.



a) $\alpha = 0°$

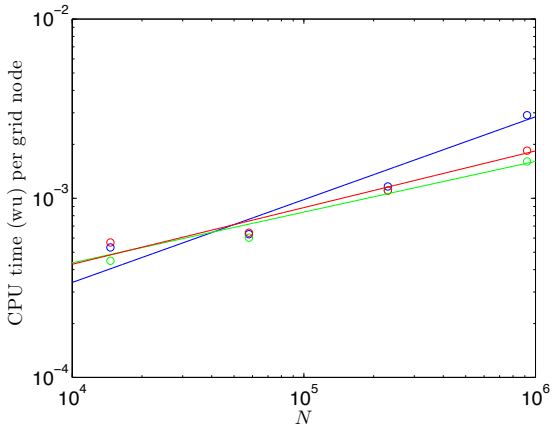


b) $\alpha = 10°$

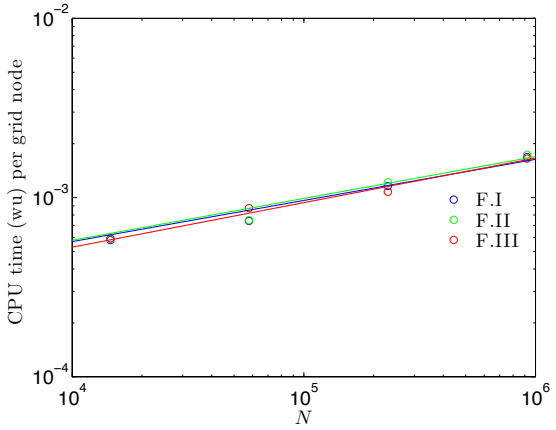


c) $\alpha = 15°$

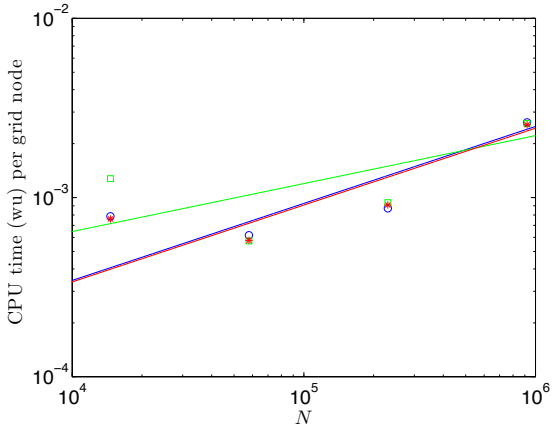
Fig. 5: Equivalent residual evaluations required to reduce the L^2-norm of the residual by eleven orders of magnitude for grid levels 3 to 6 for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation.



a) $\alpha = 0^\circ$

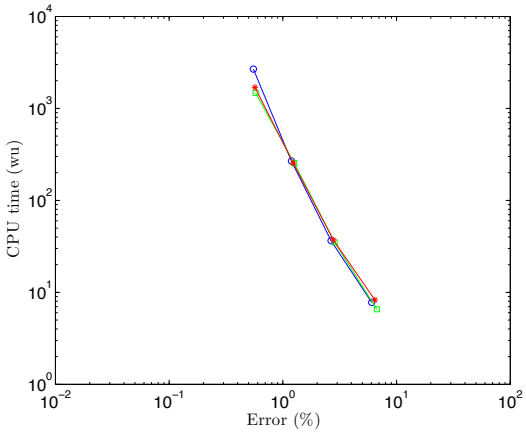


b) $\alpha = 10^\circ$

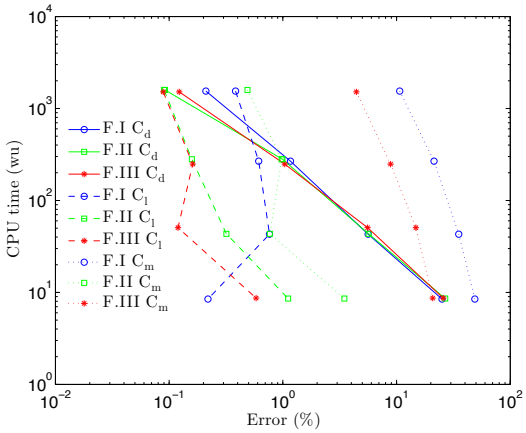


c) $\alpha = 15^\circ$

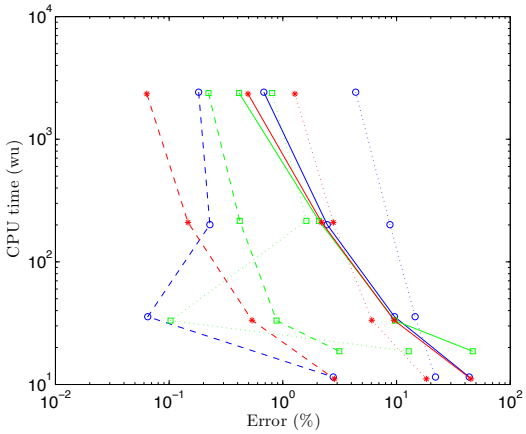
Fig. 6: Work units per grid node taken to reduce the L^2-norm of the residual by eleven orders of magnitude plotted on a logarithmic scale and including a line of best fit.



a) $\alpha = 0^\circ$



b) $\alpha = 10^\circ$



c) $\alpha = 15^\circ$

Fig. 7: Work units required to reduce the L^2-norm of the residual by eleven orders of magnitude plotted versus C_d and C_l error for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation.

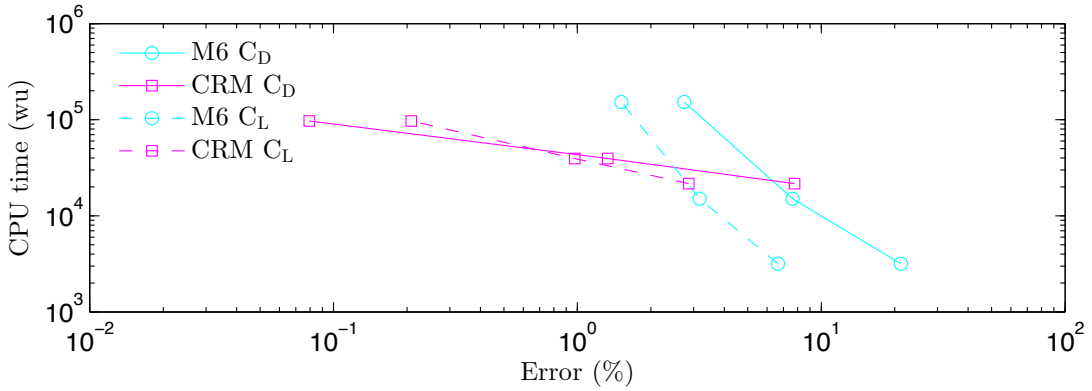


Fig. 8: Work units taken to reduce the L^2-norm of the residual by eleven orders of magnitude plotted versus C_d and C_l error for the ONERA M6 and NASA CRM cases.

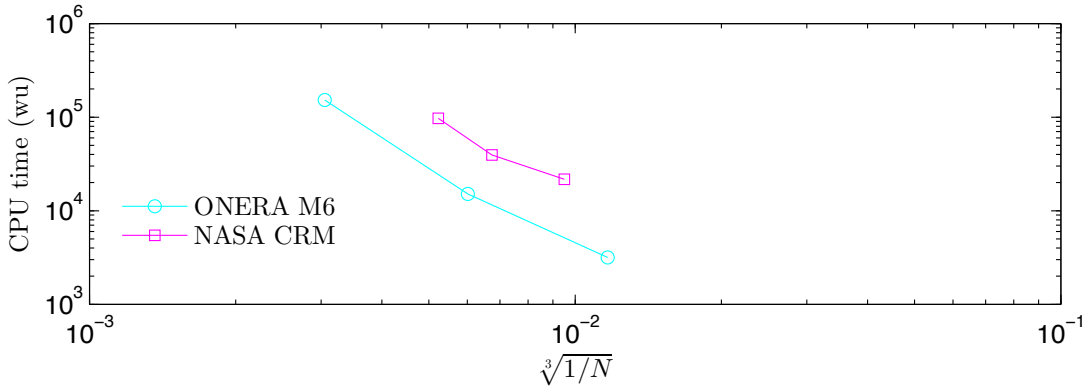


Fig. 9: Work units taken to reduce the L^2-norm of the residual by eleven orders of magnitude plotted versus grid spacing for the ONERA M6 and NASA CRM cases.

APPENDIX A: ADDITIONAL DATA AND FIGURES

Table A.1: Richardson extrapolated functional values for the NACA 0012 case at $\alpha = 0°$, calculated from grid levels 2 to 4

		C_d	C_{dp}	C_{dv}
Fam. I	scalar	8.1201×10^{-3}	1.3050×10^{-3}*	6.8164×10^{-3}
	matrix	8.1284×10^{-3}	1.3038×10^{-3}	6.8260×10^{-3}*
F. II	scalar	8.1191×10^{-3}	1.3036×10^{-3}	6.8168×10^{-3}
	matrix	8.1276×10^{-3}	1.3020×10^{-3}	6.8260×10^{-3}
F. III	scalar	8.1195×10^{-3}	1.3040×10^{-3}	6.8169×10^{-3}
	matrix	8.1280×10^{-3}	1.3023×10^{-3}	6.8270×10^{-3}*

Table A.2: Richardson extrapolated functional values for the NACA 0012 case at $\alpha = 10°$. This table also includes extrapolated grid converged functional values calculated from data obtained from the TMR website for the flow solvers FUN3D and CFL3D. The data was obtained without the use of far-field vortex correction. The functional values for the Diablo flow solver were calculated from grid levels 2 to 4 whereas the FUN3D and CFL3D flow solvers used grid levels 1 to 3.

		C_d	C_{dp}	C_{dv}	C_l	C_m
Fam. I	scalar	1.2262×10^{-2}	6.0692×10^{-3}	6.1901×10^{-3}	$1.0916^{\dagger}$	6.8418×10^{-3}*
	matrix	1.2259×10^{-2}	6.0674×10^{-3}	6.2060×10^{-3}*	1.0908*	6.8396×10^{-3}*
	FUN3D	1.2223×10^{-2}	6.0184×10^{-3}	6.2043×10^{-3}*	1.0905*	6.9422×10^{-3}*
	CFL3D	1.2212×10^{-2}	6.0079×10^{-3}	6.2060×10^{-3}*	1.0888*	7.3407×10^{-3}*
F. II	scalar	1.2254×10^{-2}	6.0605×10^{-3}	6.1930×10^{-3}	1.0910	6.7517×10^{-3}
	matrix	1.2249×10^{-2}	6.0602×10^{-3}	6.2050×10^{-3}	1.0911*	6.7696×10^{-3}*
	FUN3D	1.2225×10^{-2}	6.0208×10^{-3}	6.2038×10^{-3}*	1.0913*	6.7725×10^{-3}*
	CFL3D	1.2216×10^{-2}†	6.0121×10^{-3}	6.2041×10^{-3}*	1.0911	6.8067×10^{-3}
F. III	scalar	1.2253×10^{-2}	6.0609×10^{-3}	6.1920×10^{-3}	1.0904*	6.9299×10^{-3}*
	matrix	1.2250×10^{-2}	6.0599×10^{-3}	6.2013×10^{-3}	1.0903*	6.9102×10^{-3}
	FUN3D	1.2225×10^{-2}	6.0205×10^{-3}	6.2040×10^{-3}†	1.0912	6.7856×10^{-3}
	CFL3D	1.2217×10^{-2}†	6.0126×10^{-3}†	6.2049×10^{-3}*	1.0905*	6.9561×10^{-3}†

[] * These values were calculated using a first-order extrapolation because the convergence rate p calculated using equation (17) was less than 1.

[] † These values were calculated using a first-order extrapolation because the data on the three finest grid levels was non-monotonic.

Table A.3: Richardson extrapolated functional values for the NACA 0012 case at $\alpha = 15°$, calculated from grid levels 2 to 4

		C_d	C_{dp}	C_{dv}	C_l	C_m
Fam. I	scalar	2.1025×10^{-2}	1.5824×10^{-2}	5.1972×10^{-3}	1.5519	$1.6298 \times 10^{-2*}$
	matrix	2.1018×10^{-2}	1.5825×10^{-2}	$5.2150 \times 10^{-3*}$	1.5513^*	$1.6370 \times 10^{-2*}$
F. II	scalar	2.0998×10^{-2}	1.5794×10^{-2}	5.2016×10^{-3}	1.5505	$1.6401 \times 10^{-2*}$
	matrix	2.0996×10^{-2}	1.5805×10^{-2}	$5.2160 \times 10^{-3*}$	1.5510^*	1.6431×10^{-2}
F. III	scalar	2.1000×10^{-2}	1.5793×10^{-2}	5.2013×10^{-3}	1.5501	$1.6543 \times 10^{-2*}$
	matrix	2.0997×10^{-2}	1.5791×10^{-2}	$5.2170 \times 10^{-3*}$	1.5502	1.6526×10^{-2}

Table A.4: Comparison of functional estimates calculated with Richardson and first-order extrapolation for all Family I cases where $p < 1$ was observed.

α	Solver	Functional	p	Richardson	First order
0°	Diablo s.	C_{dp}	0.73697	1.3065×10^{-3}	1.3050×10^{-3}
	Diablo m.	C_{dv}	0.96347	6.8271×10^{-3}	6.8260×10^{-3}
10°	Diablo s.	C_m	0.45322	1.0073×10^{-2}	6.8418×10^{-3}
	Diablo m.	C_{dv}	0.94753	6.2082×10^{-3}	6.2060×10^{-3}
	Diablo m.	C_l	0.25046	1.0818	1.0908
	Diablo m.	C_m	0.45831	7.6879×10^{-3}	6.8396×10^{-3}
	FUN3D	C_{dv}	0.59819	6.2048×10^{-3}	6.2043×10^{-3}
	FUN3D	C_l	0.76012	1.0899	1.0905
	FUN3D	C_m	0.78594	7.0593×10^{-3}	6.9422×10^{-3}
	CFL3D	C_{dv}	0.28369	6.2174×10^{-3}	6.2060×10^{-3}
	CFL3D	C_l	0.39605	1.0875	1.0888
	CFL3D	C_m	0.46506	7.5735×10^{-3}	7.3407×10^{-3}
15°	Diablo s.	C_m	0.10551	2.2495×10^{-2}	1.6298×10^{-2}
	Diablo m.	C_{dv}	0.93289	5.2183×10^{-3}	5.2150×10^{-3}
	Diablo m.	C_l	−0.97961	1.5556	1.5513
	Diablo m.	C_m	0.48168	1.7141×10^{-2}	1.6370×10^{-2}

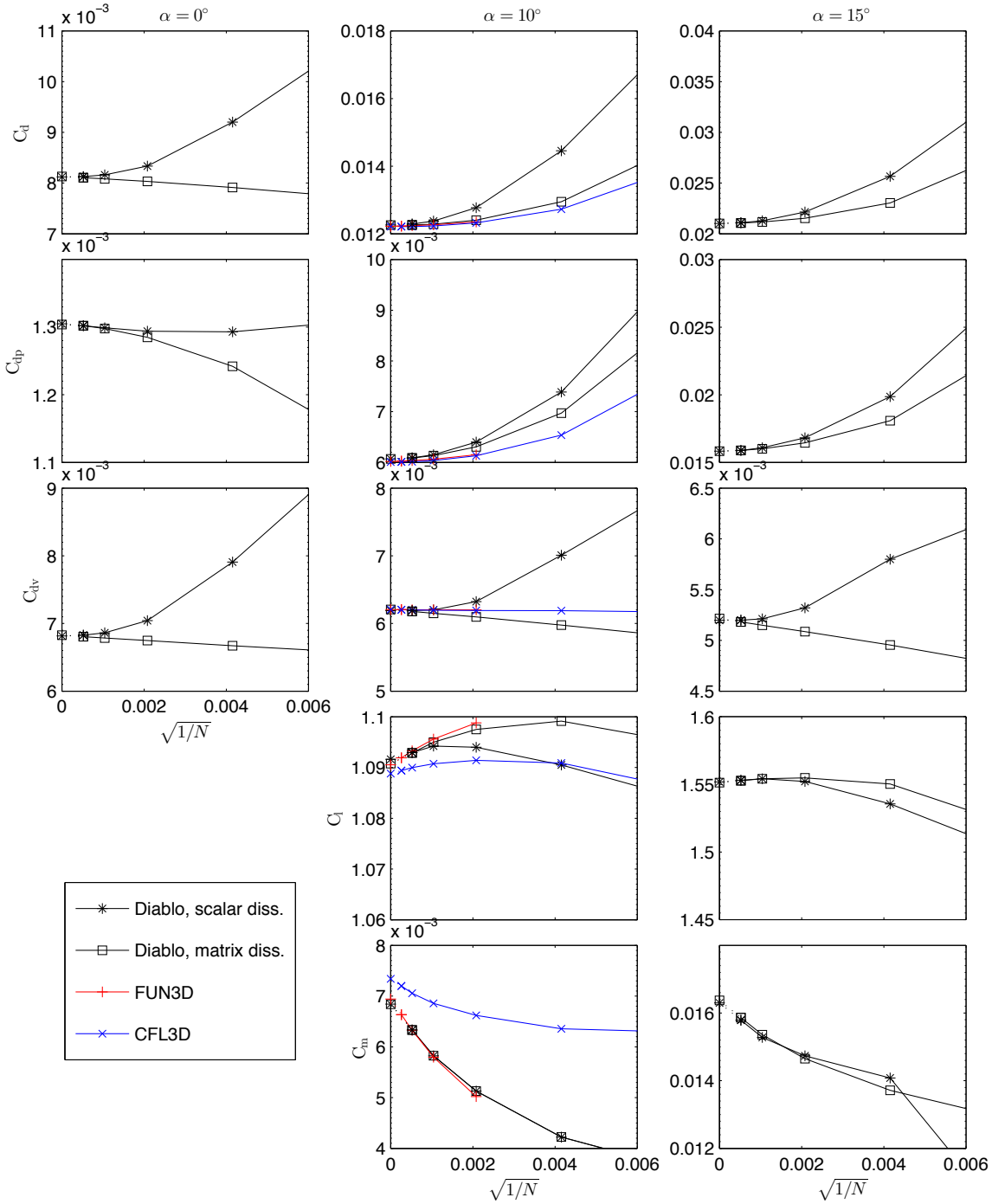


Fig. A.1: Grid convergence data for Family I of the NACA 0012 case. Grid levels 2 through 5 are shown for Diablo, levels 1 through 5 for CFL3D, and levels 1 through 4 for FUN3D. The data shown for Diablo include both scalar and matrix dissipation, and the extrapolated grid-converged values are shown at $\sqrt{1/N} = 0$.

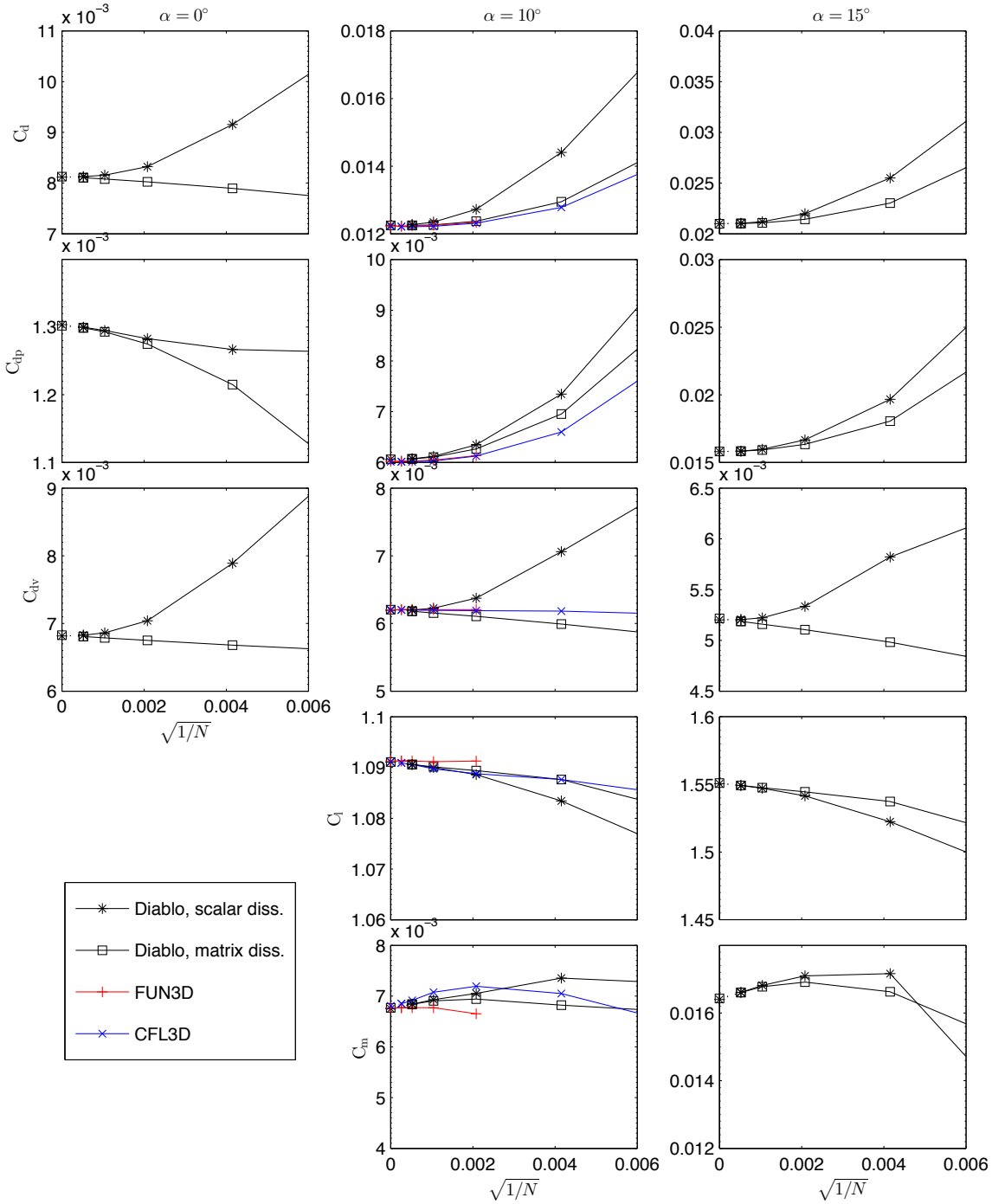


Fig. A.2: Grid convergence data for Family II of the NACA 0012 case. Grid levels 2 through 5 are shown for Diablo, levels 1 through 5 for CFL3D, and levels 1 through 4 for FUN3D. The data shown for Diablo include both scalar and matrix dissipation, and the extrapolated grid-converged values are shown at $\sqrt{1/N} = 0$.

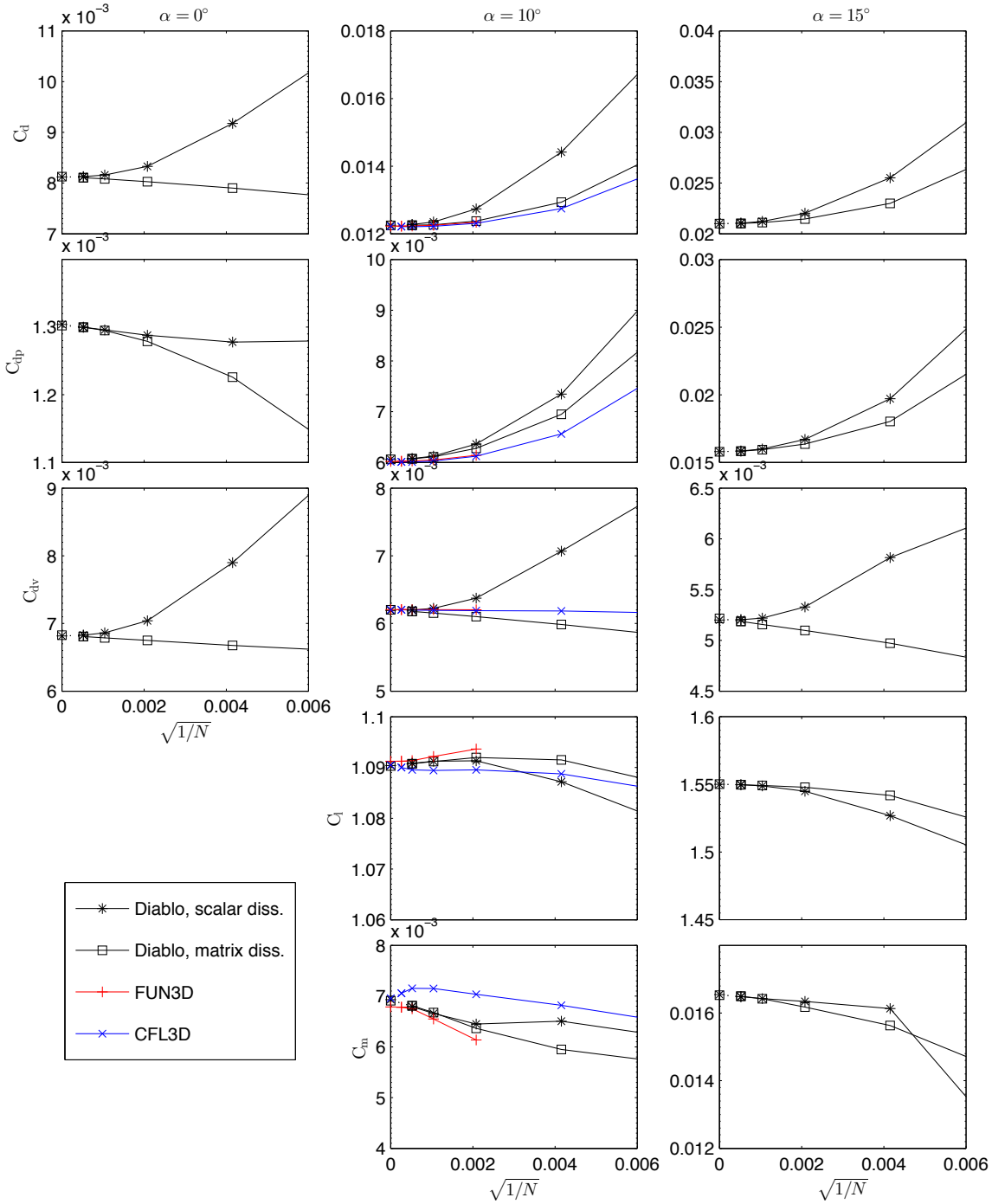


Fig. A.3: Grid convergence data for Family III of the NACA 0012 case. Grid levels 2 through 5 are shown for Diablo, levels 1 through 5 for CFL3D, and levels 1 through 4 for FUN3D. The data shown for Diablo include both scalar and matrix dissipation, and the extrapolated grid-converged values are shown at $\sqrt{1/N} = 0$.