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

David A. Brown* Howard P. Buckley† Michal Osusky‡

David W. Zingg§

University of Toronto Institute for Aerospace Studies, Toronto, Ontario, Canada

Computational results are presented for steady two-dimensional turbulent flows modelled through the Reynolds-averaged Navier-Stokes equations using the parallel Newton-Krylov-Schur finite-difference flow solver known as Diablo. The focus of this paper is to present performance results for the flow solver on three families of NACA 0012 grids for some specified operating conditions as part of a discussion group on turbulent flow computations. The results are presented in terms of computing time in TauBench work units as a function of the numerical error as the mesh is refined. Results for two three-dimensional cases are also presented.

I. Introduction

This paper presents a series of computational fluid dynamics flow solver performance studies which have been conducted as part of a discussion group session at the 53rd AIAA Aerospace Sciences Meeting. The purpose of this session is to analyze and compare the performance of various external aerodynamic flow solvers for solving the Reynolds-averaged Navier-Stokes (RANS) equations on a family of two-dimensional NACA 0012 grids at very low Mach number.

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,[1,2,3,4,5] 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 RANS equations with the Spalart-Allmaras (SA)[6] one-equation turbulence model.

Diablo originated as an Euler (inviscid) solver which was developed by Hicken and Zingg.[7] It was augmented to the full Navier-Stokes equations by Osusky et al.[8] and to the RANS-SA equations by Osusky and Zingg.[9,10] The Diablo RANS-SA flow solver has been validated at the Fifth AIAA Drag Prediction Workshop[11,12] where the focus was on the three-dimensional wing-body geometry referred to as the NASA Common Research Model (CRM). In addition, several supplementary two-dimensional cases were investigated for solver accuracy, all from the TMR website: flow over a flat plate, flow over a bump in channel, and flow over the NACA 0012 airfoil geometry.[12]

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 a Richardson extrapolation as long as at least three grid levels are used,[13] where each successively coarser grid in the grid family is constructed by removing every even numbered grid point from a finer grid level.[14,15] The emphasis of the current study is not solely on the accuracy of the flow solver but rather on the relationship between accuracy and computational cost. As the grid is refined, the lift and

*PhD candidate, Student Member AIAA

†Research Engineer, Member AIAA

‡Postdoctoral Fellow, Member AIAA (currently Mechanical Engineer, GE Global Research

§Professor and Director, Tier 1 Canada Research Chair in Computational Fluid Dynamics and Environmentally Friendly Aircraft Design, J. Armand Bombardier Foundation Chair in Aerospace Flight, Associate Fellow AIAA

drag functionals of interest can be computed with reduced numerical error but at increased computational cost. The paper will attempt to give some quantification of how the computational cost of a flow solution using the Diablo flow solver relates to the numerical error of the lift and drag functionals and how the cost depends on the level of accuracy.

While Osusky and Zingg[16] have previously completed some performance analysis for Diablo on the ONERA M6 wing and the NASA CRM wing-body geometry, the current study will be on two-dimensional flows as specified by the discussion group. Diablo is designed specifically for three-dimensional flows and suffers some significant coding inefficiencies when used for two-dimensional cases. These must be considered when comparing Diablo's efficiency with two-dimensional flow solvers, or even to other three-dimensional flow solvers which may handle two-dimensional cases more efficiently.

The test case investigated in the current study is the NACA 0012 airfoil at different operating conditions. The grids used for the study are publicly available from the Turbulence modelling Resource (TMR) website.[17] 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[18] and FUN3D[19] have demonstrated that the different trailing-edge spacing has 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 and the inclusion of quadratic constitutive relations (QCR)[20] are also investigated for their effects on flow solution accuracy at different mesh spacing.

II. Governing Equations and Spatial Discretization

This section presents the continuous version of the Navier-Stokes and Spalart-Allmaras turbulence model, including any modifications to the model investigated in this study. The spatial discretization is also discussed.

II.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, μ is the viscosity, $\mathbf{u} = (u, v, w)$ are the Cartesian velocity components, $\tau = \tau(u, v, w, \mu)$ are the viscous stresses given by the Newtonian stress tensor, Re is the Reynolds number, and $\mu = \mu(a)$ 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 $S^* = 198.6°\text{R}$ is Sutherland's constant. 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 Osusky.[10]

Non-dimensional variables are used: the density is normalized by ρ_∞, the velocities by a_∞, the viscosity by μ_∞, the temperature by T_∞, and the spatial terms appearing in the derivatives by l. Assuming that the flow behaves as an ideal gas, the pressure variable 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.

II.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:

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

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.[10]

The quadratic constitutive relations (QCR)[20] can be expressed as a modification to the Boussinesq approximation:

$$\tau_{ij} = \bar{\tau}_{ij} - c_{nl1}\left(O_{ik}\bar{\tau}_{jk} + O_{jk}\bar{\tau}_{ik}\right), \tag{10}$$

$$O_{ik} \equiv \frac{\partial_k u_i - \partial_i u_k}{\sqrt{\partial_n u_m \partial_n u_m}},$$

$$\partial_n u_m \partial_n u_m = \left(\partial_x u\right)^2 + \left(\partial_x v\right)^2 + \left(\partial_x w\right)^2 + \left(\partial_y u\right)^2 + \left(\partial_y v\right)^2 + \left(\partial_y w\right)^2 + \left(\partial_z u\right)^2 + \left(\partial_z v\right)^2 + \left(\partial_z w\right)^2,$$

where u_i are the velocity components (u, v, w) and $c_{nl1} = 0.3$. Additionally, $\bar{\tau}_{ij}$ is the Reynolds stress tensor obtained from the Boussinesq approximation.

II.C. Spatial Discretization

The RANS equations are discretized 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.[1,3,2,22,5] 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.

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.

II.D. Artificial Dissipation Models

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 decreases as the grid spacing is reduced, 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 employs a parallel Newton-Krylov-Schur algorithm for steady three-dimensional flows on multi-block meshes. The 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 smaller-stencil approximation to the Jacobian matrix where the fourth-difference dissipation is approximated with a smaller-stencil 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 taken 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.

III.A. Inexact Newton Methods

Consider a nonlinear system of algebraic equations, represented by

$$\mathcal{R}\left(\mathbf{q}\right) = \mathbf{0}, \tag{11}$$
$$\mathcal{R} : \mathbb{R}^N \rightarrow \mathbb{R}^N, \ \mathbf{q} \in \mathbb{R}^N.$$

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

$$\mathcal{A}^{(n)}\,\Delta\mathbf{q}^{(n)} = -\mathcal{R}\left(\mathbf{q}^{(n)}\right), \tag{12}$$
$$\Delta\mathbf{q}^{(n)} \equiv \mathbf{q}^{(n+1)} - \mathbf{q}^{(n)}, \ \mathcal{A}^{(n)} \equiv \nabla\mathcal{R}\left(\mathbf{q}^{(n)}\right),$$

where $\mathcal{A}^{(n)} : \mathbb{R}^N \rightarrow \mathbb{R}^N$ 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:

$$\left\|\mathcal{R}\left(\mathbf{q}^{(n)}\right) + \mathcal{A}^{(n)}\,\Delta\mathbf{q}^{(n)}\right\| \leq \eta^{(n)}\left\|\mathcal{R}\left(\mathbf{q}^{(n)}\right)\right\|. \tag{13}$$

III.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]

$$\left(\mathcal{T}^{(n)} + \mathcal{A}^{(n)}\right) \Delta\mathbf{q}^{(n)} = -\mathcal{R}\left(\mathbf{q}^{(n)}\right), \tag{14}$$

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:

$$\Delta t_i^{(n)} = T_i a \left(b\right)^{m\left\lfloor \frac{n}{m} \right\rfloor}, \; T_i = \frac{1}{1 + J_i^{\frac{1}{D}}}, \tag{15}$$

where J is the geometric Jacobian resulting from the coordinate transformation on the mesh, i is the grid point index, D is the number of spatial dimensions (either 2 or 3 in this paper), and $\lfloor \cdot \rfloor$ is the floor operator[a]. The floor operator is present in the formula in the case where we choose to update the preconditioner every m iterations instead of every iteration. The input variables a and b were set to $a = 0.001$ and $b = 1.35$ for most of the NACA 0012 cases presented in this paper, which are fairly aggressive parameter settings for turbulent flows. The value of b was reduced to 1.25 in a few cases when convergence difficulties were encountered. The value of m was set to 1 for all NACA 0012 cases.

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

$$\mathcal{R}_{\text{rel}}^{(n)} \equiv \frac{\left\|\mathcal{R}\left(\mathbf{q}^{(n)}\right)\right\|}{\left\|\mathcal{R}\left(\mathbf{q}^{(0)}\right)\right\|} \tag{16}$$

is reduced below some user-specified tolerance μ_{rel}. For the NACA 0012 cases presented in this paper, μ_{rel} was usually set to 10^{-4}, though it was reduced to 10^{-5} in some cases when convergence difficulties were encountered.

III.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 $\mathcal{A}^{(n)}\mathbf{v}$, $\mathbf{v} \in \mathbb{R}^N$. 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:

$$\mathcal{A}^{(n)}\mathbf{v} \approx \frac{\mathcal{R}\left(\mathbf{q}^{(n)} + \epsilon\mathbf{v}\right) - \mathcal{R}\left(\mathbf{q}^{(n)}\right)}{\epsilon}, \tag{17}$$

where $\epsilon \in \mathbb{R}$ is the perturbation parameter and is chosen to balance between truncation error and round-off error. In this work, the following formula is used:

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

where δ is a value near to machine precision ($\delta = 10^{-13}$ is used in all cases in this study), and N is the number of elements in the vector $\mathbf{v}$. A second option is to recycle the smaller-stencil 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

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

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. For deep convergence in the inexact Newton 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 solution on various mesh levels and flow conditions.

IV.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).

IV.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:

$$p = \frac{\ln\left(\left|f_3 - f_2\right| / \left|f_2 - f_1\right|\right)}{\ln\left(r\right)}, \tag{18}$$

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:

$$f_0 = \frac{r^p f_1 - f_2}{r^p - 1}. \tag{19}$$

The three-grid Richardson extrapolation method is not reliable unless it is applied in the asymptotic region of a method.[29] If the final three points are non-monotonic, or if the estimated convergence rate is less that 1, then the grid converged functional is estimated from a first-order extrapolation by setting $p = 1$ in equation (19).

IV.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[16] have previously performed some performance analysis on the Diablo RANS-SA flow solver, and a similar approach will be taken here. Osusky and Zingg[16] 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 eliminate this problem is to use a benchmark such as the TauBench codes[30] to nondimensionalize cost as was done by Wang et al.[31] 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 2.50×10^6 nodes on 1 processor with 10 iterative steps. The average CPU time after running the TauBench code four times on the SciNet general

Table 1. Grid details for the NACA 0012 geometry. OW=Off-Wall, TE=Trailing Edge. Spacings measured in chords (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}

purpose cluster was 9.571s, which will be referred to as one *work unit*. When CPU time is measured across n_p processors, the wall time is multiplied by n_p to get the total CPU time. Wall time is measured from the beginning to the end of the flow solve, excluding pre- and post-processing steps.

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, two-dimensional grids with the farfield 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 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 0, 10°, and 15°. The effects of QCR are considered for Family I. 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, the choice of dissipation should not affect the grid converged solution.

V.A. Grid Convergence

Tables A.1 through A.3 show the predicted grid-converged values for all three grid families and all three sets of operating conditions calculated from mesh levels 2 through 4. As explained previously, Richardson extrapolation was used by applying equation (19) with the predicted convergence rate calculated from equation (18) in all cases where the functional values on these grid levels were monotonic and the convergence rate calculation gave $p > 1$. Otherwise, a first-order extrapolation was used to predict the grid-converged values by applying equation (19) with convergence rate $p = 1$.

The grid convergence data is plotted in Figures A.1 through A.3. 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. An additional observation is that many functional values, especially C_l and C_m, do not appear fully grid converged, even on the finest mesh. This effect was observed for all three flow solvers and was least pronounced on the Family II mesh, which had the finest trailing-edge spacing.

V.B. 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 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 can be observed by plotting the residual history against the number of linear iterations taken by FGMRES as shown in Figure 1. This is a way to investigate the effect of grid refinement on the linear system and nonlinear iterative methods without considering CPU time. A plot of the CPU cost of reducing the L^2-norm of the residual by eleven orders of magnitude versus the number of nodes is shown in Figure 2. It can be observed from the plots that performance is similar for the three grid families. It is also noteworthy that the zero angle of attack case converges the slowest.

Figure 3 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 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 attempt to alleviate some of the timing inconsistency incurred by running different flow solves on different processors which may have slightly different operating speeds. Also, since the cost of the residual vector is expected to be roughly proportional to the number of grid nodes,[b] any increase in the number of equivalent residual evaluations is mainly due to an increased number of linear and nonlinear iterations. Similarly, Figure 4 shows the work units per grid point required to reduce the norm of the residual by eleven orders of magnitude on each mesh level, which, as expected, appears quite similar to Figure 3.

The next study is an accuracy study. This is presented in Figure 5, which shows how the error in C_d and C_l decreases with mesh spacing.

The final study performed relates the accuracy obtained at each grid level to the cost of performing the flow solve. The error in the calculated C_d and C_l values was estimated based on the extrapolated value obtained from the grid convergence study. The plot can be found in Figure 6, where the most accurate and most expensive points correspond to the finer grid levels.

VI. Three-Dimensional Results

The CPU expense is also studied as a function of the level of accuracy needed for the three-dimensional cases of Osusky and Zingg.[16] 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[16]) 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 2.

Figure 7 shows how the CPU time depends on the level of accuracy for these two cases. The total CPU time is calculated by multiplying the wall time taken to converge the flow solver by the total number of processors used for that case.

[b] For these NACA 0012 cases the cost of the residual seems to increase by a factor of between 4.0 and 4.4 at each grid refinement.

Table 2. 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

The Diablo RANS flow solver capabilities have been characterized using three families of 2D NACA 0012 grids for fully turbulent flow at a low Mach number, as well as a 3D ONERA M6 wing grid family at transonic Mach number, and a 3D CRM wing-body case, also at a transonic Mach number. The study focused 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.

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.

References

[1]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.

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

[3]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.

[4]Lundquist, T. and Nordström, J., "The SBP-SAT Technique for Initial Value Problems," *Journal of Computational Physics*, Vol. 270, 2014, pp. 86–104.

[5]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.

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

[7]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.

[8]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.

[9]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.

[10]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.

[11]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.

[12]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.

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

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

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

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

[17]"NASA Turbulence Modeling Resource," http://http://turbmodels.larc.nasa.gov, Accessed: 2014-09-10.

[18]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.

[19]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.

[20]Spalart, P. R., "Strategies for Turbulence Modelling and Simulation," *International Journal of Heat and Fluid Flow*, Vol. 21, 2000, pp. 252–263.

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

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

[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]DLR Germany, "TauBench Version 1.1, IPACS," http://www.ipacs-benchmark.org, Accessed: 2014-09-20.

[31]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.-O., 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.

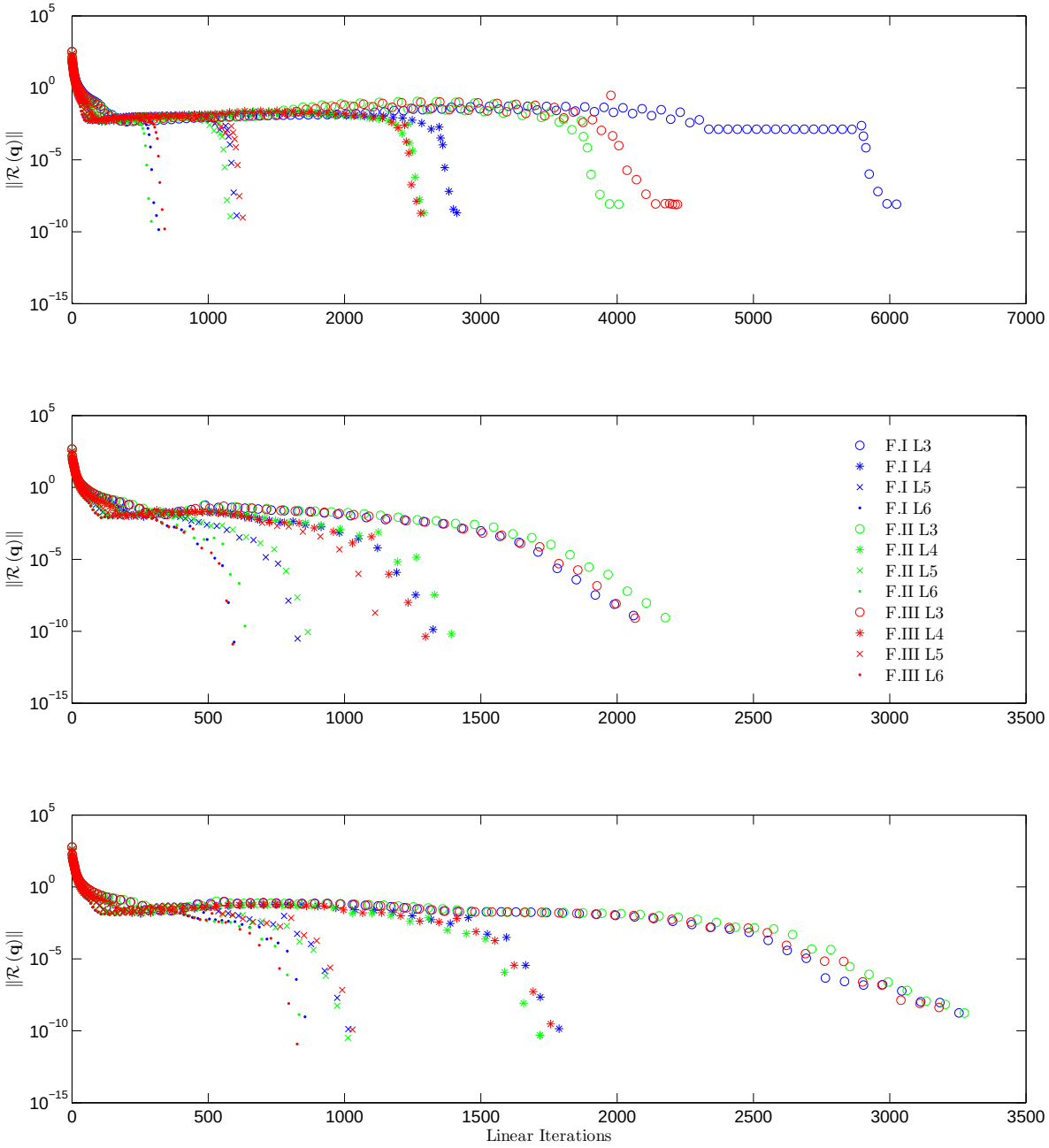

Figure 1. Residual history plotted against the number of linear iterations that were required to reduce the L^2-norm of the residual by eleven orders of magnitude for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation. Angles of attack from top to bottom are 0°, 10°, and 15°. The legend entries describe the family (F.I=Family I, for example) and grid level (L3=level 3, L4=level 4, etc.).

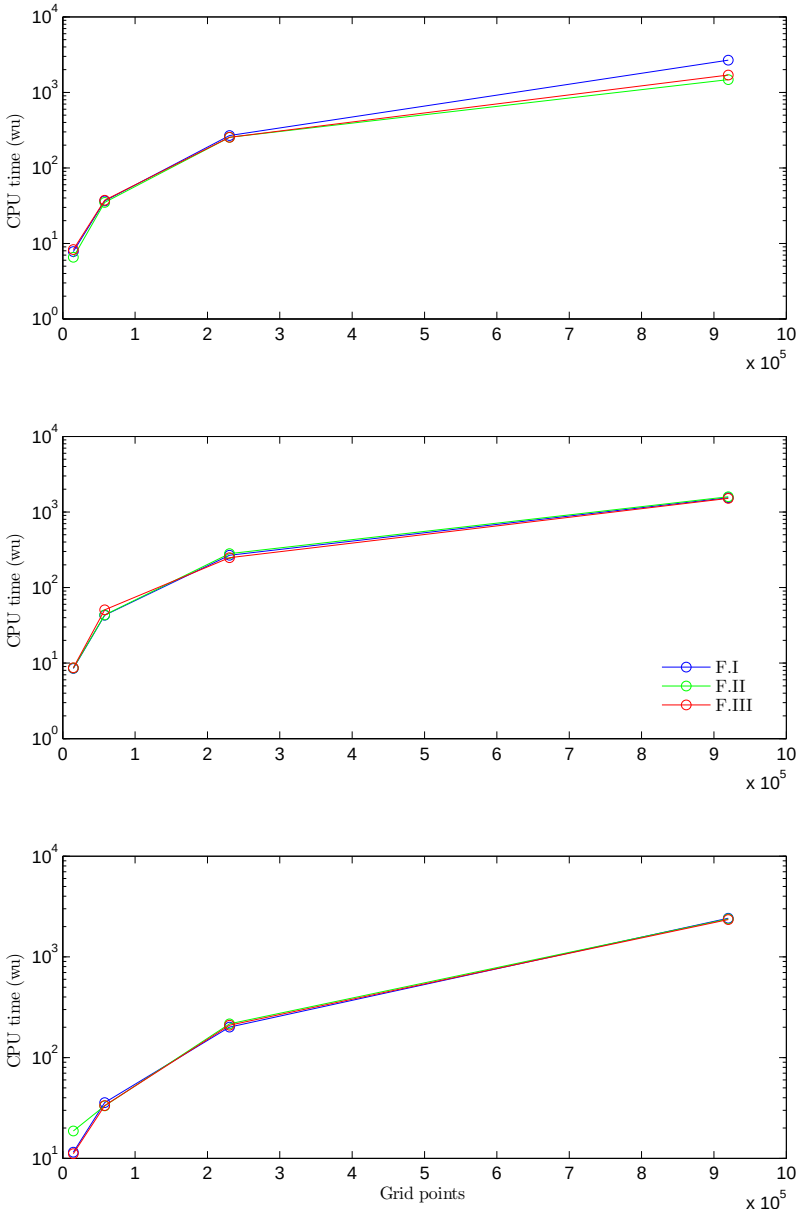

Figure 2. Work units taken to reduce the L^2-norm of the residual by eleven orders of magnitude for different levels of mesh refinement for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation. Angles of attack from top to bottom are 0°, 10°, and 15°.

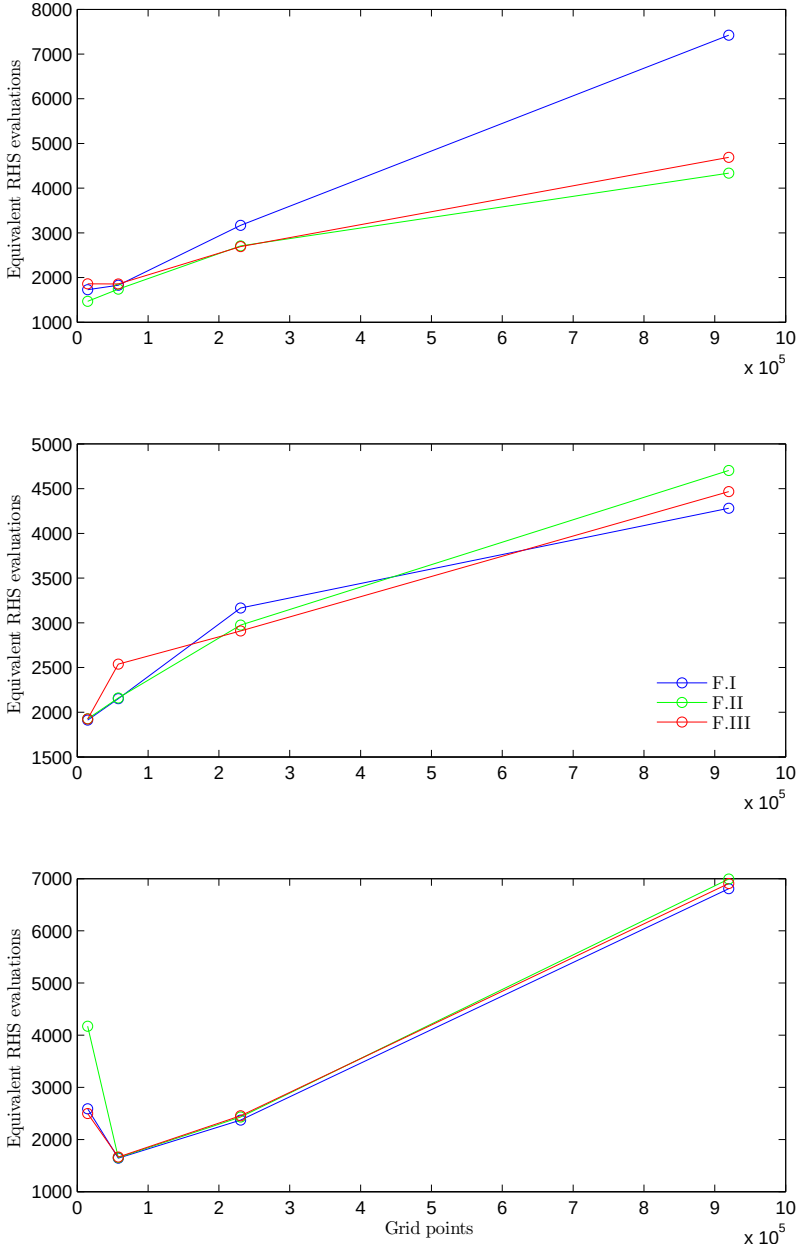

Figure 3. Equivalent residual evaluations required to reduce the L^2-norm of the residual by eleven orders of magnitude for grid levels 3 to 6 for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation. Angles of attack from top to bottom are 0°, 10°, and 15°.

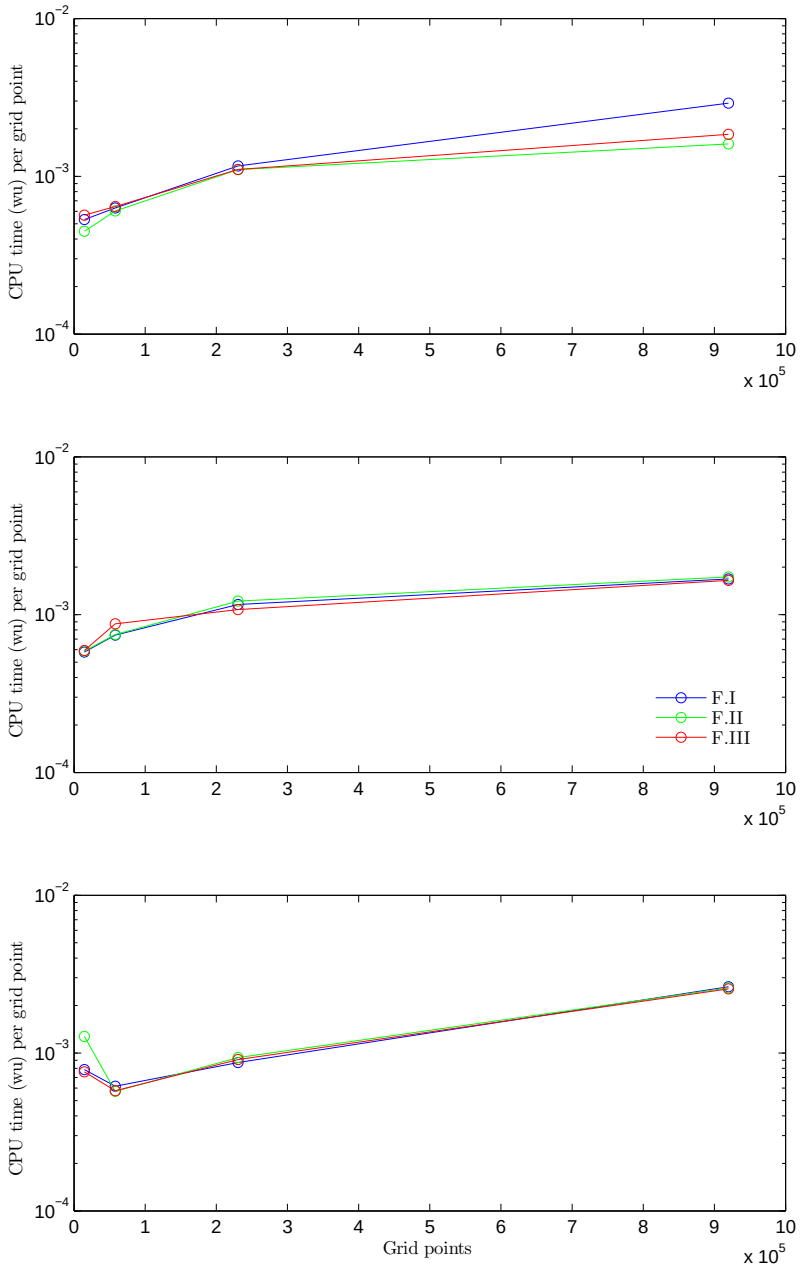

Figure 4. Work units per grid point required to reduce the L^2-norm of the residual by eleven orders of magnitude for grid levels 3 to 6 for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation. Angles of attack from top to bottom are 0°, 10°, and 15°.

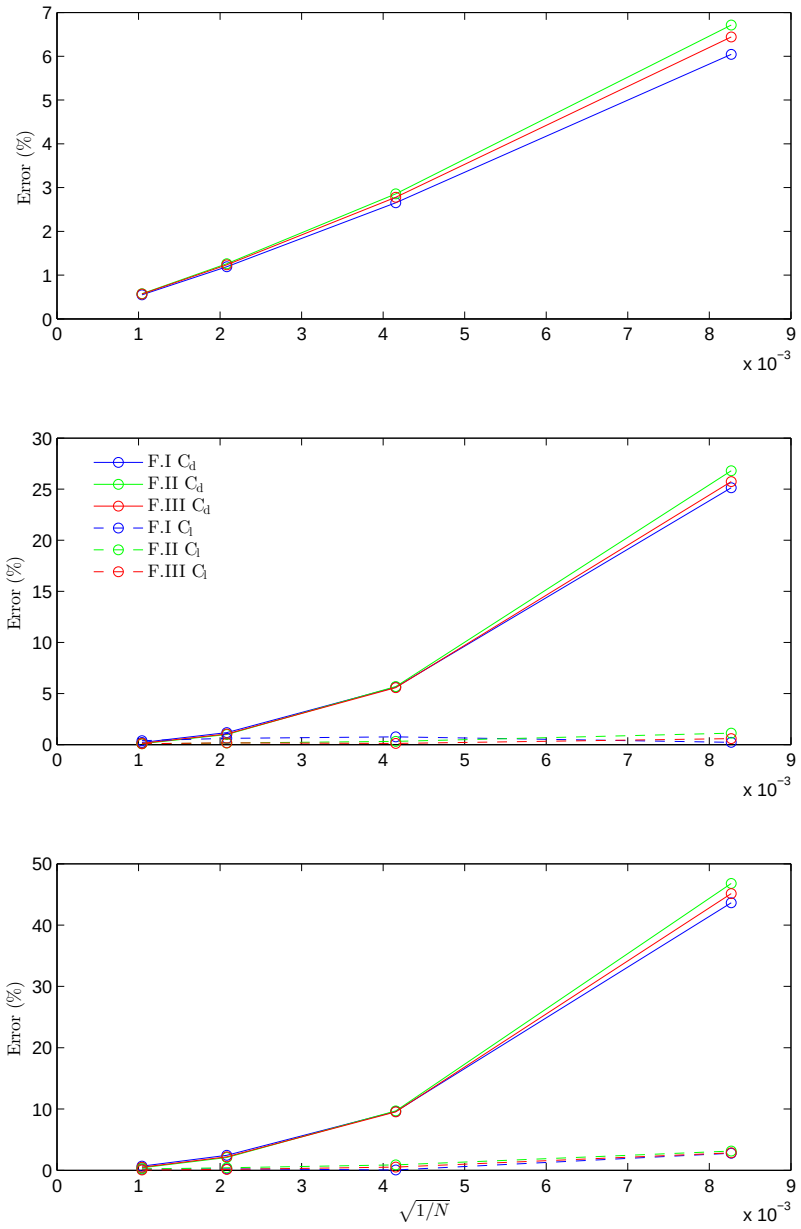

Figure 5. Percentage error in the C_d and C_l values calculated at the converged flow solution at grid levels 3 to 6 for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation. Angles of attack from top to bottom are 0°, 10°, and 15°.

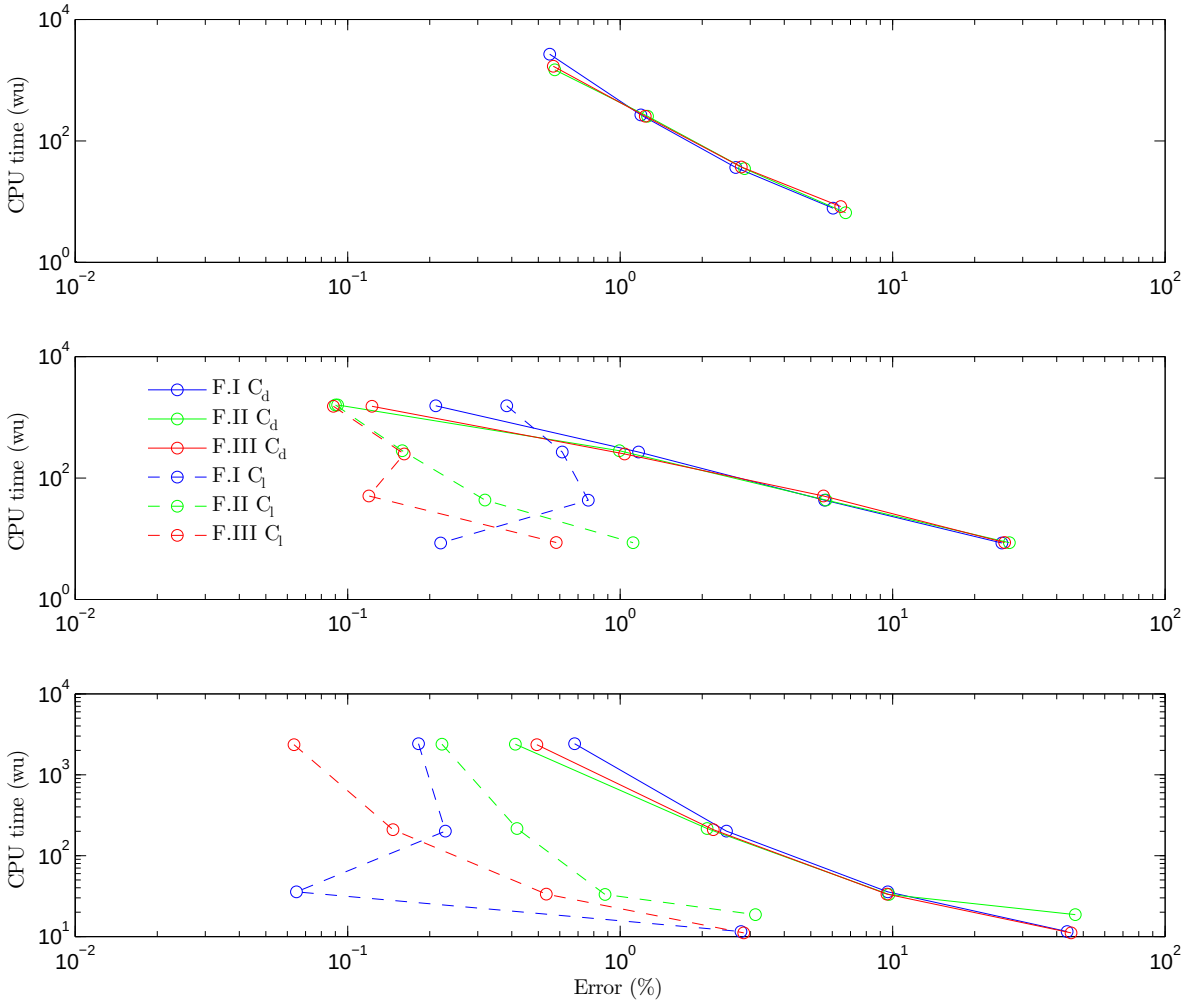

Figure 6. Work units taken to achieve a certain C_d and C_l error threshold for the NACA 0012 cases. Results are shown for all three grid families using matrix dissipation. Angles of attack from top to bottom are 0°, 10°, and 15°.

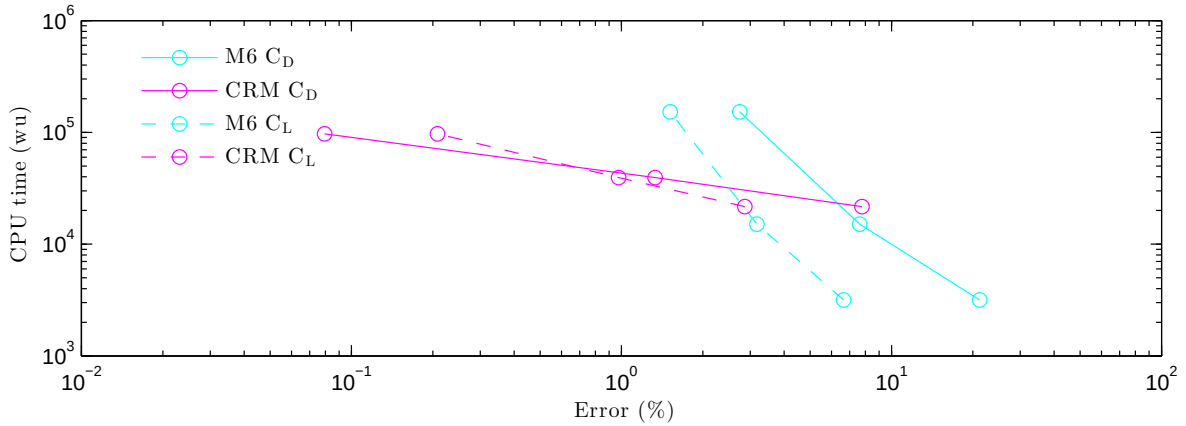

Figure 7. Work units taken to reduce the L^2-norm of the residual by 12 orders of magnitude for different levels of mesh refinement for the ONERA M6 and NASA CRM cases.

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}
Family I	scalar	8.1201×10^{-3}	1.3050×10^{-3}*	6.8164×10^{-3}
	scalar, QCR	8.1203×10^{-3}	1.3030×10^{-3}*	6.8173×10^{-3}
	matrix	8.1284×10^{-3}	1.3038×10^{-3}	6.8260×10^{-3}*
	matrix, QCR	8.1316×10^{-3}	1.3033×10^{-3}	6.8281×10^{-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 farfield point 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
Family I	scalar	1.2262×10^{-2}	6.0692×10^{-3}	6.1901×10^{-3}	1.0916*	6.8418×10^{-3}*
	scalar, QCR	1.2253×10^{-2}	6.0514×10^{-3}	6.1901×10^{-3}	1.0916*	6.7099×10^{-3}
	matrix	1.2259×10^{-2}	6.0674×10^{-3}	6.2060×10^{-3}*	1.0908*	6.8396×10^{-3}*
	matrix, QCR	1.2250×10^{-2}	6.0444×10^{-3}	6.2150×10^{-3}*	1.0913*	6.7731×10^{-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}*

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
Family I	scalar	2.1025×10^{-2}	1.5824×10^{-2}	5.1972×10^{-3}	1.5519*	1.6298×10^{-2}*
	scalar, QCR	2.0773×10^{-2}	1.5450×10^{-2}*	5.2130×10^{-3}*	1.5560*	1.6013×10^{-2}*
	matrix	2.1018×10^{-2}	1.5825×10^{-2}	5.2150×10^{-3}*	1.5513*	1.6370×10^{-2}*
	matrix, QCR	2.0746×10^{-2}*	1.5500×10^{-2}*	5.2420×10^{-3}*	1.5553*	1.6055×10^{-2}
F. II	scalar	2.0998×10^{-2}	1.5794×10^{-2}	5.2016×10^{-3}	1.5505	1.6401×10^{-2}*
	matrix	2.0996×10^{-2}	1.5805×10^{-2}	5.2160×10^{-3}*	1.5510*	1.6431×10^{-2}
F. III	scalar	2.1000×10^{-2}	1.5793×10^{-2}	5.2013×10^{-3}	1.5501	1.6543×10^{-2}*
	matrix	2.0997×10^{-2}	1.5791×10^{-2}	5.2170×10^{-3}*	1.5502	1.6526×10^{-2}

* Values marked with an asterisk were calculated using a first-order extrapolation because either the convergence rate calculated using equation (18) was less than 1 or the functional estimates on the three finest grid levels were non-monotonic.

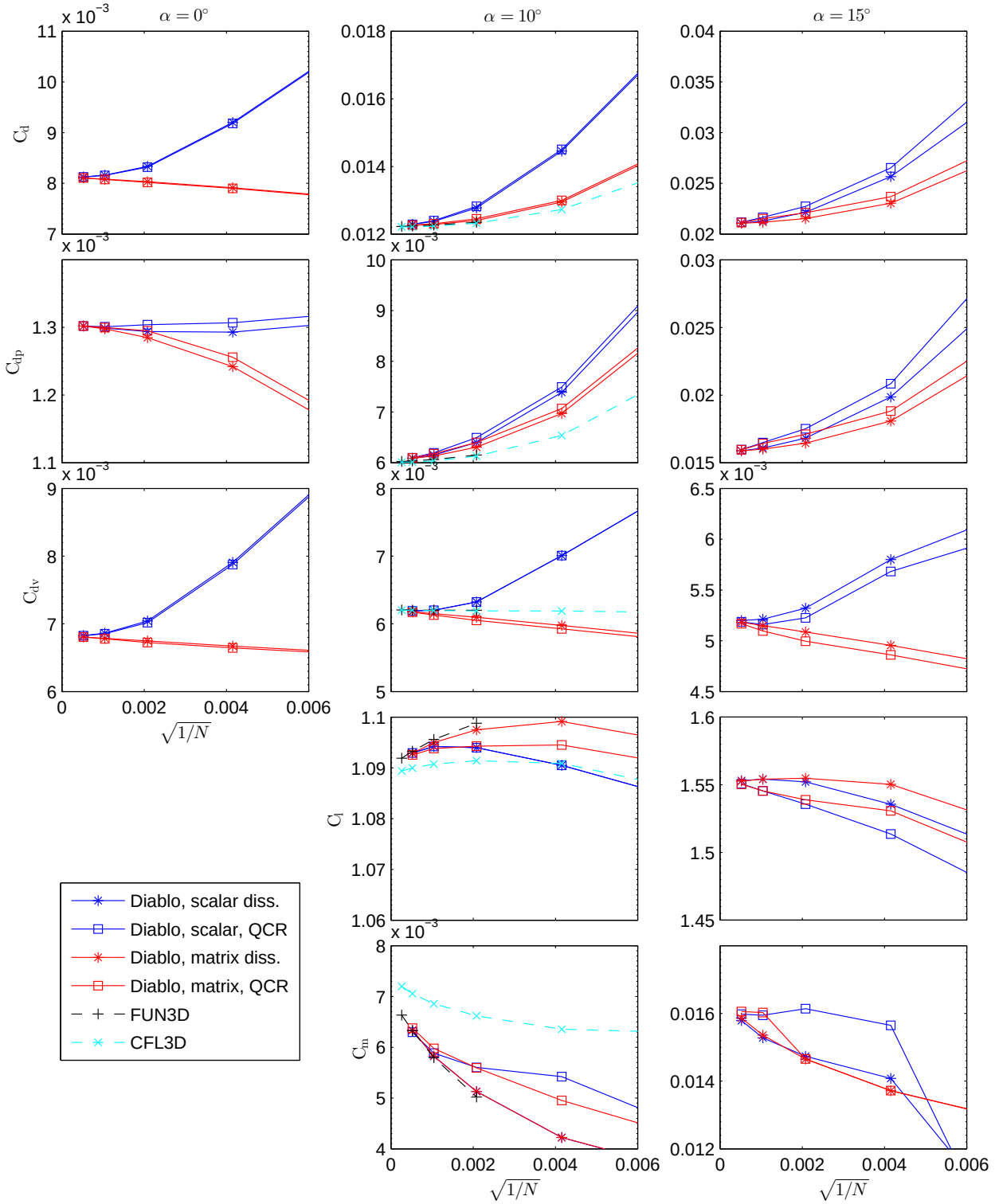

Figure A.1. Grid convergence data for Family I of the NACA 0012 case. Grid levels 2 through 5 are shown for Diablo, levels 1 through 5 for CFL3D, and levels 1 through 4 for FUN3D. The data shown for Diablo include both scalar and matrix dissipation, with and without QCR.

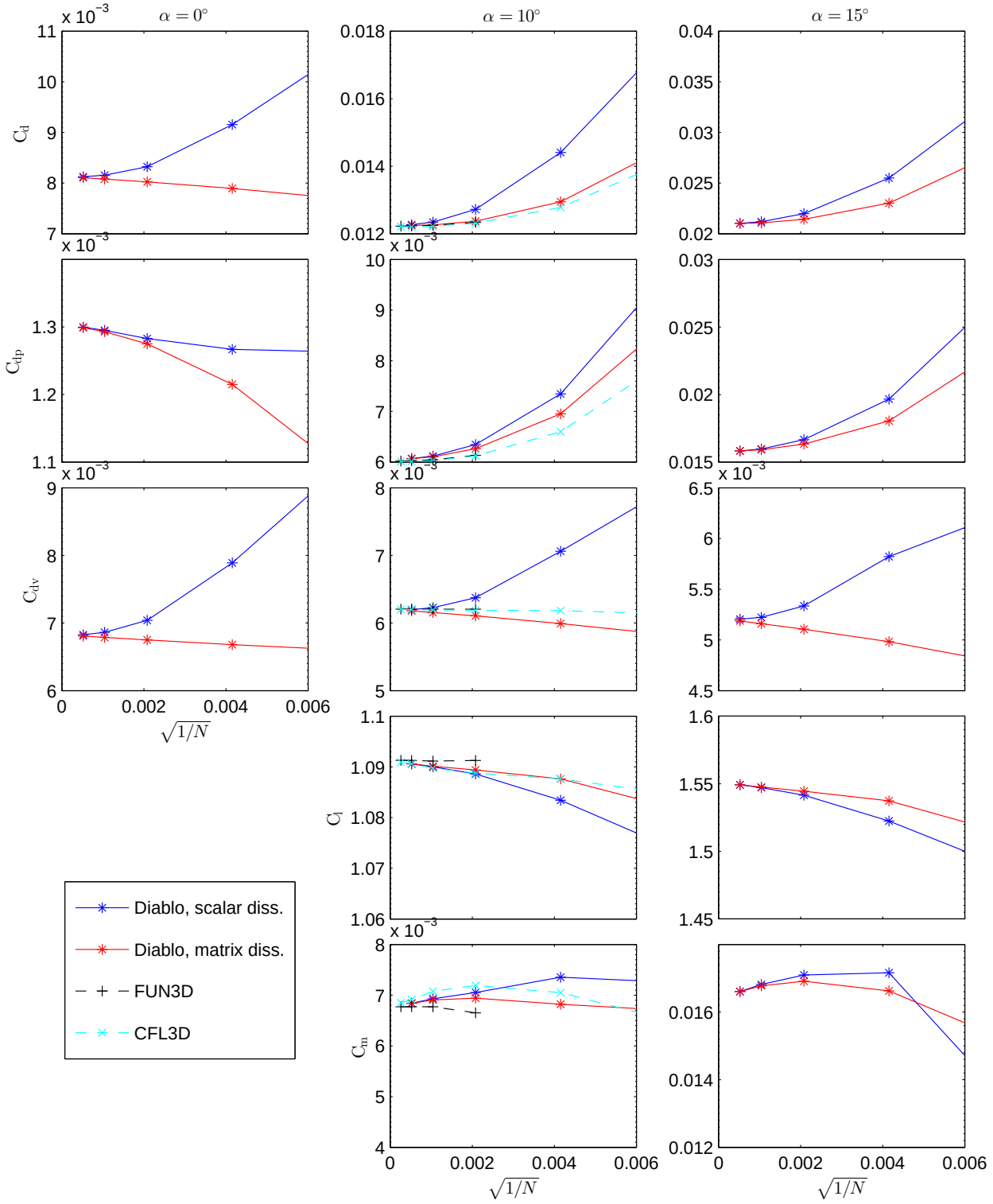

Figure A.2. Grid convergence data for Family II of the NACA 0012 case. Grid levels 2 through 5 are shown for Diablo, levels 1 through 5 for CFL3D, and levels 1 through 4 for FUN3D. The data shown for Diablo include both scalar and matrix dissipation.

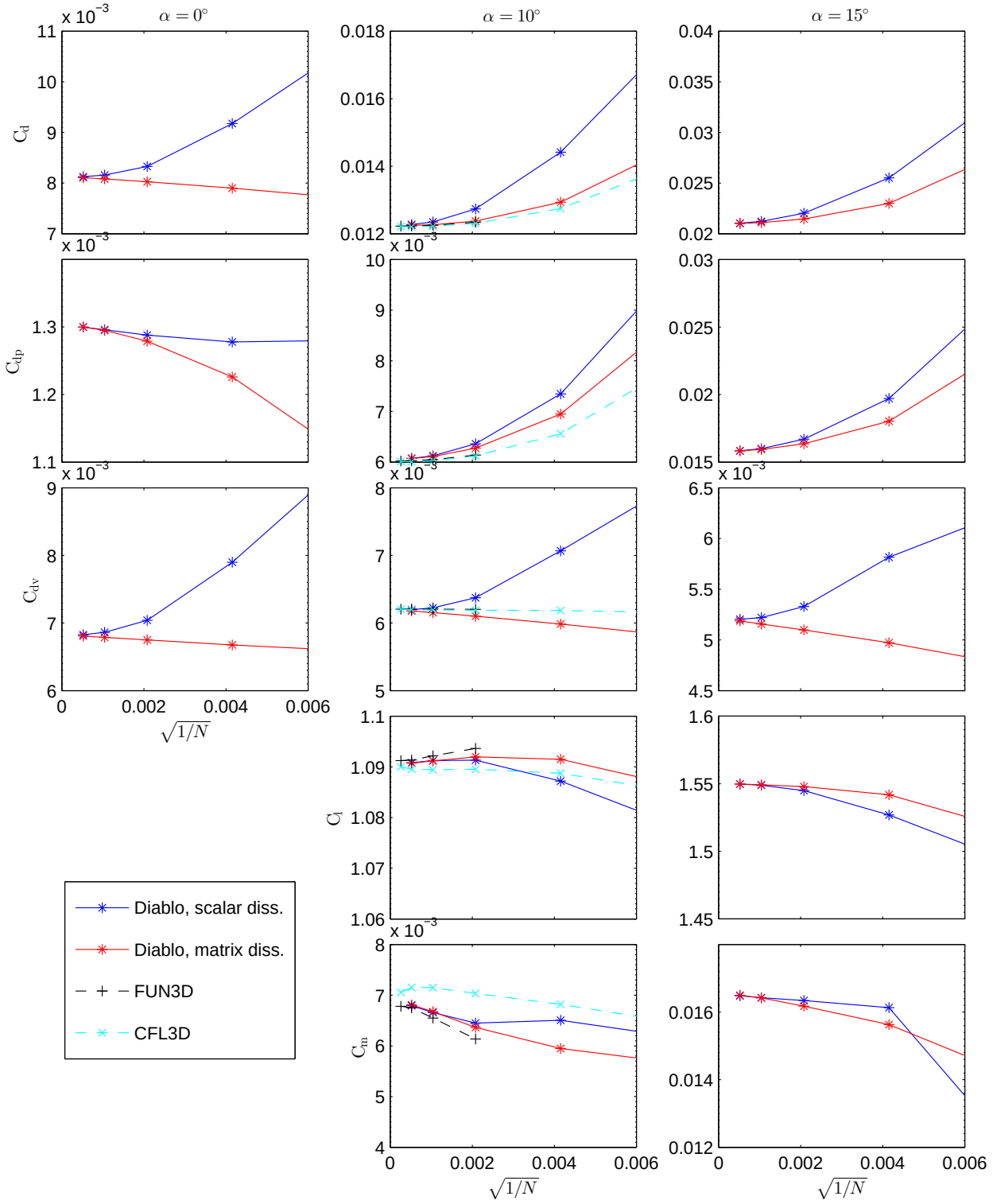

Figure A.3. Grid convergence data for Family III of the NACA 0012 case. Grid levels 2 through 5 are shown for Diablo, levels 1 through 5 for CFL3D, and levels 1 through 4 for FUN3D. The data shown for Diablo include both scalar and matrix dissipation.