42nd AIAA Aerospace Sciences Meeting and Exhibit, Paper Number 2004-0436
Reno, NV
January 5–8, 2004

DEVELOPMENT OF A HIGHER-ORDER SOLVER FOR AERODYNAMIC APPLICATIONS

Krzysztof J. Fidkowski*
David L. Darmofal†

We present the results from the development of a higher-order discontinuous Galerkin finite element solver using p-multigrid with line Jacobi smoothing. The line smoothing algorithm is presented for unstructured meshes, and p-multigrid is outlined for the nonlinear Euler equations of gas dynamics. Analysis of 2-D advection shows the improved performance of line implicit versus point implicit relaxation. Through a mesh refinement study, we determine that the accuracy of the discretization is the optimal $O(h^{p+1})$ for three different smooth problems. The multigrid convergence rate is found to be independent of the polynomial order but does depend weakly on the grid size. Timing studies for each problem indicate that higher order is advantageous over grid refinement when high accuracy is required.

INTRODUCTION

While CFD has achieved significant maturity during the past decades, computational costs are extremely large for aerodynamic simulations of aerospace vehicles. In this applied aerodynamics context, the discretization of the Euler and/or Navier-Stokes equations is almost exclusively performed by finite volume algorithms. The pioneering work of Jameson began this evolution to the status quo.[1–3] During the 1980's, upwinding mechanisms were incorporated into these finite volume algorithms leading to increased robustness for applications with strong shocks, and perhaps more importantly, to better resolution of viscous layers due to decreased numerical dissipation in these regions.[4–8] The 1990's saw major advances in the application of finite volume methods to Navier-Stokes simulations (in particular the Reynolds-Averaged Navier-Stokes equations). Significant gains were made in the use of unstructured meshes and solution techniques for viscous problems.[9–12] While these algorithmic developments have resulted in an ability to simulate aerodynamic flows for very complex problems, the time required to achieve sufficient accuracy in a reliable manner places a severe constraint on the application of CFD to aerospace design.

The accuracies of many of the finite-volume methods currently used in aerodynamics are at best $p = 2$, i.e. the error decreases as $O(h^p)$ where h is a measure of the grid spacing. As a practical matter, however, the accuracy of these methods on more realistic problems appears to be less than this, ranging between $1 \leq p \leq 2$. The development of a practical higher-order solution method could result in a significant decrease in the computational time required to achieve an acceptable error level. To better demonstrate the potential of higher-order methods, we make the following assumptions.

- The error, E, in the solution (or an output of the solution) is $O(h^p)$.
- The number of elements, N_{el}, in the grid is related to the cell size by $N_{el} = O(h^{-d})$, where d is the spatial dimension of the problem.
- Higher-order accuracy is achieved by increasing the number of unknowns per element (in a finite element manner), N_{dof}, which scales as $N_{dof} = O(p^d)$. Thus, the total number of unknowns is $N = N_{el}N_{dof} = O\left((p/h)^d\right)$.
- The number of floating point operations (or the work), W, required to solve the discrete system is $O(N^w)$ where w is the complexity of the solution algorithm.
- The time required to complete a single operation is $1/F$, and hence the total time for solution of the system of equations if $T = W/F$.

Combining these assumptions, we find that the time to achieve a specified error is

$$T = O\left(\left(p/E^{1/p}\right)^{wd}/F\right).$$

Taking the log of this relationship,

$$\log T = wd\left(-\frac{1}{p}\log E + \log p\right) - \log F + \text{constant}.$$

If the accuracy requirements are stringent, i.e. $E << 1$, we expect that the $\log E$ term will dominate the $\log p$ term. Thus, the time required will depend exponentially on p, w, and d. This reasoning demonstrates the significant benefit of improving the order of accuracy and the solution complexity. Furthermore, since the computational time scales only inversely with the computational speed, F, small changes in w/p can be as significant as increasing computational power.

Numerous reasons exist for why current finite-volume algorithms are not practical at higher-order. The root cause of many of these difficulties lies in the extended stencils which these algorithms employ. For finite volume discretizations that explicitly add numerical dissipation, the extended stencils arise from the higher-order stabilization terms. For finite volume algorithms which introduce stabilization through upwinding, the extended

*Research Assistant. Email: kfid@mit.edu
†Associate Professor, Senior Member AIAA. 77 Massachusetts Ave. 37-401, Cambridge, MA 02139. Ph. (617) 258-0743. Email: darmofal@mit.edu

stencils arise through the local interpolants used to increase accuracy. These extended stencils contribute to difficulties in:

- *Stable iterative algorithms.* As is well known, the iterative solution of these discretizations requires either multi-stage methods and/or implicitness beyond a locally implicit scheme. Another common iterative approach employs backwards Euler in which the Jacobian of the higher-order discretization is replaced by a lower-order approximation. Unfortunately, Mavriplis[13] has shown that the use of lower-order approximations severely limits the convergence rate attainable for higher-order finite-volume simulations of complex problems even when the lower-order systems are solved exactly.

- *Memory requirements.* Extended stencils degrade the sparsity of the linearized systems of equations used in implicit solution methods. This increased fill results in very high memory requirements and is the reason that lower-order approximations are often utilized.

- *Parallelization.* Similar to the increased memory requirements, the large support of the extended stencils also increases the communication bandwidth required for parallel computations. However, when the number of cells in each processor is large (as is common in the coarser grain parallelism used today), this effect may be minimal.

By contrast, finite element formulations introduce higher-order effects compactly within the element. Thus, viewed from the element level, the stencils are not extended for higher-order finite element discretizations. For Discontinuous Galerkin (DG) formulations, the element-to-element coupling exists only through the flux at the shared boundaries between elements. This limited coupling for DG discretizations is an enabling feature which permits the development of efficient higher-order solvers and potentially *significant* improvements in the turn-around time for reliably accurate aerodynamic simulations. Recently, Venkatakrishnan et al[14] showed that higher-order finite element schemes have significant advantages for smooth inviscid and viscous flows; however, they also delineate several remaining challenges that must be addressed before higher-order methods will be robust and efficient for practical applications, which include shocks or other under-resolved flow features.

In this paper, we consider multigrid solution algorithms for higher-order DG discretizations. Though the current paper only presents results for inviscid flows, the solution algorithm is designed for high Reynolds number viscous problems, which will be the subject of future work. We begin with a description of the DG discretization for the Euler equations. Then, we present a p-multigrid algorithm in which the coarse discretizations are formed from lower order discretization (using a hierarchical basis) and in which the smoother is line-element block Jacobi. Stability analysis indicates that the eigenvalues of the iterative algorithm are relatively insensitive to p, but dependent on h. An important result from this analysis is that element block Jacobi schemes are stable regardless of the order of the approximation without the need for multi-staging (which is not true for higher-order methods using extended stencils). Finally, numerical results are presented for Ringleb flow, flow through a variable-area duct, and flow over an airfoil, demonstrating that high accuracy solutions are obtained in less computational time using higher-order discretizations combined with the p-multigrid algorithm.

DISCRETIZATION

We next describe the DG discretization of the compressible Euler equations (for additional details, consult the review by Cockburn and Shu[15] and the references). The two-dimensional Euler equations of gas dynamics are given by:

$$\mathbf{u}_t + \nabla \cdot \mathcal{F}(\mathbf{u}) = 0, \tag{1}$$

where $\mathbf{u}$ is the conservative state vector,

$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix},$$

and $\mathcal{F} = (\mathbf{F}^x, \mathbf{F}^y)$ is the inviscid flux,

$$\mathbf{F}^x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}, \quad \mathbf{F}^y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix}.$$

The total enthalpy is given by $H = E + p/\rho$, and the equation of state is

$$p = (\gamma - 1)\left[\rho E - \frac{1}{2}\rho\left(u^2 + v^2\right)\right].$$

Denote $\mathcal{V}_h^p$ to be the space of discontinuous vector-valued polynomials of degree p on a subdivision T_h of the domain Ω into elements such that $\bar{\Omega} = \bigcup_{\kappa \in T_h} \bar{\kappa}$. The DG discretization of the Euler equations is of the following form: find $\mathbf{u}_h \in \mathcal{V}_h^p$ such that $\forall \mathbf{v}_h \in \mathcal{V}_h^p$,

$$\begin{aligned} \sum_{\kappa \in T_h} \Big\{ & \int_\kappa \mathbf{v}_h^T \left(\mathbf{u}_h\right)_t d\mathbf{x} - \int_\kappa \nabla \mathbf{v}_h^T \cdot \mathcal{F}(\mathbf{u}_h) d\mathbf{x} \\ & + \int_{\partial\kappa \setminus \partial\Omega} \mathbf{v}_h^{+T} \mathcal{H}(\mathbf{u}_h^+, \mathbf{u}_h^-, \hat{\mathbf{n}}) ds \\ & + \int_{\partial\kappa \cap \partial\Omega} \mathbf{v}_h^{+T} \mathcal{H}_{\mathrm{bd}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \hat{\mathbf{n}}) ds \Big\} = 0, \end{aligned} \tag{2}$$

where $\mathcal{H}(\mathbf{u}_h^+, \mathbf{u}_h^-, \hat{\mathbf{n}})$ and $\mathcal{H}_{\mathrm{bd}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \hat{\mathbf{n}})$ are inviscid, numerical flux functions for interior and boundary edges, respectively. Also, the $()^+$ and $()^-$ notation indicates the trace value taken from the interior and exterior of the element, respectively, and $\hat{\mathbf{n}}$ is the outward-pointing normal of the element. The numerical flux function used to evaluate the boundary flux on $\partial\Omega$ need not coincide with that used for the interior edges. For the interior flux function, we use the Roe-averaged flux function.[5] The boundary conditions on $\partial\Omega$ are imposed weakly by constructing an exterior boundary state on $\partial\Omega$ that is a

function of the inner state and boundary condition data, $\mathbf{u}_h^-(\mathbf{u}_h^+, \text{BCData})$.

The final discrete form of the DG discretization is constructed by selecting a basis for $\mathcal{V}_h^p$. Specifically, a set of element-wise discontinuous functions $\{\phi_j\}$ is introduced, such that each ϕ_j has local support on only one element. The solution to the DG discretization has the following form,

$$\mathbf{u}_h(t,x) = \sum_j \overline{\mathbf{u}}_j(t)\phi_j(x).$$

Even though our interest lies in the steady-state solution, the presence of the unsteady term is useful for improving the initial transient behavior of the solver. A simple backward Euler discretization in time is used so that the final discrete equations are

$$\mathcal{M}\frac{1}{\Delta t}\left(\overline{\mathbf{u}}^{n+1} - \overline{\mathbf{u}}^{n}\right) + \mathbf{R}(\overline{\mathbf{u}}^{n+1}) = 0, \tag{3}$$

where $\mathcal{M}$ is the mass matrix and $\mathbf{R}$ is the residual vector representing the final three terms of (2). In the following discussion, we will drop the overbar notation for the discrete solution vector.

SOLUTION METHOD

To solve the nonlinear system, $\mathbf{R}(\mathbf{u}) = 0$, we use a p-multigrid scheme with a line Jacobi smoother. A generic iterative scheme can be written as,

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \mathbf{P}^{-1}\mathbf{R}\left(\mathbf{u}^n\right), \tag{4}$$

where the preconditioner, $\mathbf{P}$, is an approximation to $\frac{\partial \mathbf{R}}{\partial \mathbf{u}}$. We have considered two preconditioners: an elemental block-Jacobi smoother, in which the unknowns on each element are solved simultaneously, and an elemental line block-Jacobi smoother, in which the unknowns on each line of elements are solved simultaneously. We will focus on the details of the line smoother, and the multigrid solver based on it.

Line-Implicit Smoother

The motivation for the use of a line smoother is that in strongly convective systems, the transport of information proceeds along characteristic directions. By solving implicitly on lines of elements connected along these directions, we can alleviate the stiffness associated with strong convection. Also, for viscous flows, the line solver is an important ingredient in removing the stiffness associated with regions of high grid anisotropy which are frequently required in viscous layers.[11,16] In such cases, the lines are formed between elements which the strongest coupling, not purely based on convection. To implement such a smoother, we need to be able to construct a set of lines of elements and to solve implicitly on each of these lines.

We first give an overview of the solution process. Assume we have N_l lines, or disjoint sets of adjacent elements, such that every element exists in one line. We wish to solve our system of equations implicitly on each of these lines. To do so, we construct a set of N_l block tridiagonal systems and solve each to obtain the state updates. Consider line $l, 1 \le l \le N_l$, containing n_l elements and let $\mathbf{M}^l$ denote the linear system for that line. Note that $\mathbf{M}^l$ is a block $n_l \times n_l$ matrix. The diagonal blocks $\mathbf{M}^l_{j,j}$ consist of the local Jacobians associated with the elements on the line. The off-diagonal blocks $\mathbf{M}^l_{j,k}$ represent the influence of the states in element k on the residual in element j. We only include the block-tridiagonal entries in $\mathbf{M}^l$ although it is possible for a line of elements to 'wrap back' such that true matrix structure would not be tridiagonal. Since this element would be weakly coupled to any previous element, ignoring these off-diagonal blocks is not likely to cause a substantial loss in performance.

The final form of the preconditioner based on this elemental line smoother is augmented by the addition of the unsteady term

$$\mathbf{P} = \mathbf{M} + \frac{1}{\Delta t}\mathcal{M}, \tag{5}$$

where $\mathbf{M}$ is the entire set of line matrices. The addition of the time term corresponds to solving for a finite time step, Δt, in the unsteady problem. Mathematically, this addition makes the system more diagonally dominant and hence better conditioned for the iterative method. Physically, instead of solving the steady state equations, we are now solving for the evolution of the system at a time increment of Δt. The time term is used to help alleviate transients during the solution process. As the solution begins to converge, $\Delta t \to \infty$. A discussion of how Δt is set is given in the section on Robustness.

Inversion of the $\mathbf{P}$ uses a block-tridiagonal algorithm in which the diagonal block is LU decomposed. As the dominant cost of the line solver (especially for higher-order schemes) is the LU decomposition of the diagonal, the computational cost of the line smoother is only slighter larger than the simpler elemental block-Jacobi. However, the performance of the line smoother is significantly better due to the increased implicitness along strongly coupled directions.

Line Creation

The effectiveness of the line smoother depends on the length of the lines and on their alignment with the convective direction. In general, these criteria are difficult to achieve on irregular triangular meshes unless the flow pattern is known ahead of time and the mesh is constructed accordingly. Nevertheless, a line-creation algorithm was developed that yields a unique set of lines of maximum length on general irregular meshes.

The first step in the line generation is the construction of an element connectivity matrix $C_{j,k}$, a sparse, symmetric matrix that contains information on the strength of the coupling between the elements, or, equivalently, between the blocks of the Jacobian matrix. Since for the inviscid problem we are interested in the direction of convection, the absolute value of the inter-elemental volumetric flux (velocity weighted by face area) was used as the connectivity measure. In practice, the matrix $C_{j,k}$ is formed during residual calculation, when the inter-elemental fluxes are available, which results in marginal additional cost.

Given the connectivities in $C_{j,k}$, the line creation algorithm is employed. In the following pseudocode description, let $N(j,f)$ denote the element adjacent to element j, across face f. In addition, let $F(j)$ denote the set of faces enclosing element k.

Line Creation Algorithm

1. Obtain a seed element i
2. Call MakePath(i) - *Forward Path*
3. Call MakePath(i) - *Backward Path*
4. Return to (1). The algorithm is finished when no more seed elements exist.

MakePath(j)

While path not terminated:

For element j, pick the face $f \in F(j)$ with highest connectivity, such that element $k = N(j, f)$ is not part of the current line. Terminate the path if any of the following conditions hold:

- face f is a boundary face
- element k is already part of a line
- $C(j, f) < C(k, g),\ \forall g \in F(k), g \neq f$

Otherwise, set $j = k$ and continue.

Robustness and Limiting

One of the key goals in designing the solver was robustness - i.e. making it applicable to a wide variety of problems. Since the smoother uses a preconditioner based on a linearized form of the governing nonlinear equations, failure can occur if the initial guess is not close to the final solution. In practice, this failure is manifested through the appearance of non-physical states, such as negative density or pressure. To avoid such occurrences, a limiter was created to act on the state updates $d\mathbf{u}$ returned by the solver. Limiting is done in two different forms, depending on the degree of the nonlinear behavior.

As a first step, the state update is limited through under-relaxation:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \alpha d\mathbf{u} \tag{6}$$

The under-relaxation factor α is calculated as the greatest $\alpha > 0$ that keeps the density and pressure changes under 10% of the current values over all the elements. It is desirable to use the same α for all the elements in order to not hinder the performance of the solver. In practice, however, there can exist one or two elements which require a much lower α than the rest of the domain. Using the same small α globally in this case would unnecessarily slow the progress to the solution. To resolve this problem, a minimum global $\alpha = 0.001$ was used on all elements except locally for those which required a smaller under-relaxation factor. The calculation of this factor results in a minimal computational overhead relative to the smoother and prevents failure during the initial solution transients for many problems.

However, even with under-relaxation limiting, the solver can fail in the initial steps of a difficult problem through the inability to limit changes in the pressure. Since pressure is a nonlinear function of the conservative state variables, an iterative method is used to determine the pressure limit. The limiter fails if an acceptable α is not found after a certain maximum number of steps. In such cases, the second limiting method is employed: on each successive failure of under-relaxation, Δt is lowered by a constant factor, until under-relaxation is applied successfully. This Δt is then used for the next several solver iterations before it is successively increased back to its original maximum value (e.g. 10^{10}).

Storage and Implementation

The greatest storage requirement comes from the line preconditioner, which is essentially equal in size to the full Jacobian. The benefit of storing the full Jacobian is that doing so allows multiple linear iterations per one inversion of the diagonal blocks. We have found that linear iterations benefit overall computational time; however, storage of the full Jacobian leads to excessive memory requirements for problems in which the element count and interpolation order are large. Hence, a memory-lean version of the line solver was written in which the Jacobian is stored only for one line of elements at a time. In addition to the memory savings, the updates $d\mathbf{u}^l$ obtained for each line can be applied to the states as each line is processed, resulting in a Gauss-Seidel type iterative scheme. This method was implemented and showed slightly faster convergence rates for general problems.

p-Multigrid

Motivation

p-multigrid was used in conjunction with the line smoother to increase the performance of the solver. In standard multigrid techniques, solutions on spatially coarser grids are used to correct solutions on the fine grid. This method is motivated by the observation that most smoothers are poor at eliminating low frequency error modes on the fine grid. However, these low frequency error modes can be effectively corrected by smoothing on the coarser grids, in which these modes appear as high frequency. In p-multigrid, the idea is the same, with the exception that lower order interpolants serve as the "coarse" grids.[17,18]

p-multigrid fits naturally within the framework of high-order DG discretizations. There is no need to store additional grid information since the same spatial grid is used by all levels. In addition, a hierarchical basis can be used, eliminating the need to store separate state information at each level. The transfer operators between the grids, prolongation and restriction, are local and only need to be stored for a reference elements. These operators become trivial in the case of a hierarchical basis, and they reduce computational time by simplifying the implementation.

FAS and Two-Level Multigrid

To solve the nonlinear system in question, the Full Approximation Scheme (FAS) was chosen as the multigrid method. Much of the description that follows is adapted from Briggs.[19]

Consider the discretized system of equations given by

$$\begin{aligned} \mathbf{R}^p(\mathbf{u}^p) &= \mathbf{f}^p \\ \mathbf{r}^p &\equiv \mathbf{f}^p - \mathbf{R}^p(\mathbf{u}^p). \end{aligned}$$

In the above, $\mathbf{u}^p$ is the discrete solution vector for p^{th} order interpolation on a given grid, and $\mathbf{R}^p(\mathbf{u}^p)$ is the

associated nonlinear system. $\mathbf{f}^p$ is a source term (zero for fine-level problem), and $\mathbf{r}^p$ is the discrete residual. In a basic two-level multigrid method, the exact solution on a coarse level is used to correct the solution on a fine level. This correction scheme is given as follows:

- Restrict the state and residual to the coarse level: $\mathbf{u}_0^{p-1} = \tilde{I}_p^{p-1}\mathbf{u}^p$, $\mathbf{r}^{p-1} = I_p^{p-1}\mathbf{r}^p$.
- Solve the coarse grid problem: $\mathbf{R}^{p-1}(\mathbf{u}^{p-1}) = \mathbf{R}^{p-1}(\mathbf{u}_0^{p-1}) + \mathbf{r}^{p-1}$.
- Interpolate the coarse grid error and correct the fine level state: $\mathbf{u}^p = \mathbf{u}^p + I_{p-1}^p(\mathbf{u}^{p-1} - \mathbf{u}_0^{p-1})$.

I_p^{p-1} is the residual restriction operator, and I_{p-1}^p is the state prolongation operator. $\tilde{I}_p^{p-1}$ is the state restriction operator and is not necessarily the same as the residual restriction. We note that the FAS coarse level equation can be written as

$$\mathbf{R}^{p-1}(\mathbf{u}^{p-1}) = I_p^{p-1}\mathbf{f}^p + \tau_p^{p-1}$$
$$\tau_p^{p-1} \equiv \mathbf{R}^{p-1}(I_p^{p-1}\mathbf{u}^{p-1}) - I_p^{p-1}\mathbf{R}^{p-1}(\mathbf{u}_0^{p-1}).$$

This equation differs from the original coarse level equation by the presence of the correction term τ_p^{p-1}, which improves the correction property of the coarse level. In particular, if the fine level residual is zero, the coarse level solution is $\mathbf{u}^{p-1} = \mathbf{u}_0^{p-1}$.

V-cycles and FMG

To make multigrid practical, the basic two level correction scheme is extended to a V-cycle and to full multigrid (FMG). In a V-cycle, a sequence of coarse levels (two or more) is used to correct the solution on the fine level. Descending from the finest level to the coarsest, a certain number of pre-smoothing steps, ν_1, is performed on each level before the problem is restricted to the next coarser level. On the coarsest level, the problem is either solved directly or smoothed a relatively large number of times, ν_c. Ascending back to the finest level, a certain number of post-smoothing steps, ν_2, is performed on each level before the prolongation. Each such described V-cycle constitutes a multigrid iteration.

Using plain V-cycles to obtain a high-order solution requires starting the smoothing iterations on the highest order approximation. As this level contains the largest number of DOFs, smoothing on it is the most expensive. It would make sense to first obtain an approximation to the solution using the coarser levels before smoothing on the finest level. This is the premise behind FMG in which V-cycles on successively finer levels are used to approximate the solution on the finest level. By the time the solution is prolongated to the finest level, it is usually a close approximation to the final solution, with the exception of certain high frequency errors that can be smoothed efficiently on that level.

In an effective multigrid scheme - that is, one in which the smoother, transfer operators, and coarse level approximation spaces are well matched - FMG should require only a few V-cycles on each level before prolongating to the next finer level.[20] In practice, this behavior can be tested by using a known output to track the error at each multigrid iteration.

A decision that has to be made in the FMG cycle is when to start iterating on the next finer level. Converging the solution fully on each level is not practical because the discretization error on the coarser levels is usually well above machine zero. One can perform a constant number of V-cycles on each level; however, this adds an additional parameter in the solution process. An alternative is to prolongate when a residual-based criterion is met. The criterion used is that prolongation takes place when the residual on the current level drops below 1/2 of the residual on the next finer level. The residual on the next finer level is computed at the end of each V-cycle, at a slight additional computational cost.

Operator Definition

We now define the transfer operators: I_{p-1}^p, I_p^{p-1}, and $\tilde{I}_p^{p-1}$ used in the multigrid scheme. Let Ω denote the entire domain, and let ϕ_i^p denote the i^{th} basis function of order p in an ordering over all the basis functions in Ω. Since the approximation spaces are nested, we can write ϕ_i^{p-1} in terms of ϕ_j^p,

$$\phi_i^{p-1} = \sum_j \alpha_{i,j}^{p-1}\phi_j^p. \tag{7}$$

The prolongation operator, I_{p-1}^p, transfers $\mathbf{u}^{p-1}$ to the next finer level. Thus, we seek a representation of the coarse level solution on the finer level. That is, we wish to calculate $\mathbf{u}^p$ such that

$$\begin{aligned} \mathbf{u}_j^p &= I_{p-1}^p\mathbf{u}_i^{p-1}, \\ \sum_j \mathbf{u}_j^p\phi_j^p &= \sum_i \mathbf{u}_i^{p-1}\phi_i^{p-1}. \end{aligned}$$

Using (7) we have,

$$\begin{aligned} \sum_j \mathbf{u}_j^p\phi_j^p &= \sum_i \mathbf{u}_i^{p-1}\sum_j \alpha_{j,i}^{p-1}\phi_j^p, \\ \sum_j I_{p-1}^p\mathbf{u}_i^{p-1}\phi_j^p &= \sum_j\sum_i \alpha_{j,i}^{p-1}\mathbf{u}_i^{p-1}\phi_j^p. \end{aligned}$$

Since a state representation is unique in the basis ϕ_j^p, it must be the case that,

$$I_{p-1}^p = (\alpha^{p-1})^T. \tag{8}$$

To form the residual restriction operator, we return to the definition of the residual vector,

$$\mathbf{R}_j^p = \int_\Omega (L)\phi_j^p d\Omega.$$

L represents the original system of partial differential equations. Given $\mathbf{R}^p$ we would like to determine $\mathbf{R}^{p-1} = I_p^{p-1}\mathbf{R}^p$. Writing out $\mathbf{R}_i^{p-1}$ and using (7) yields,

$$
\begin{aligned}
\mathbf{R}_i^{p-1} &= \int_\Omega (L)\phi_i^{p-1} d\Omega \\
&= \sum_j \alpha_{i,j} \int_\Omega (L)\phi_j^p d\Omega \\
&= \sum_j \alpha_{i,j} R_j^p.
\end{aligned}
$$

Thus, the residual restriction operator is,

$$
I_p^{p-1} = \alpha^{p-1}. \tag{9}
$$

Finally, the state restriction operator, with which we can write $\mathbf{u}^{p-1} = \tilde{I}_p^{p-1}\mathbf{u}^p$, can be found by enforcing state equality between the coarse and fine levels in a weak form,

$$
\begin{aligned}
\int_\Omega \phi_k^{p-1} \sum_i \mathbf{u}_i^{p-1}\phi_i^{p-1} d\Omega &= \int_\Omega \phi_k^{p-1} \sum_j \mathbf{u}_j^p \phi_j^p d\Omega \\
\sum_i \mathcal{M}_{k,i}^{p-1}\mathbf{u}_i^{p-1} &= \sum_j \mathcal{N}_{k,j}^{p-1}\mathbf{u}_j^p \\
\mathbf{u}_i^{p-1} &= (\mathcal{M}^{p-1})^{-1}\mathcal{N}^{p-1}\mathbf{u}_j^p \\
\tilde{I}_p^{p-1} &= (\mathcal{M}^{p-1})^{-1}\mathcal{N}^{p-1}. \qquad (10)
\end{aligned}
$$

$$
\mathcal{M}_{k,i}^{p-1} = \int_\Omega \phi_k^{p-1}\phi_i^{p-1} d\Omega \qquad \mathcal{N}_{k,j}^{p-1} = \int_\Omega \phi_k^{p-1}\phi_j^p d\Omega
$$

The weak form is necessary because we cannot expect to be able to represent a general p^{th} order solution using basis functions of order $p-1$. The form presented recovers all of the components of the coarse level basis functions present in the fine level solution.

Although the operators have been defined in a global sense, the local compact support for the basis functions makes it sufficient to calculate these operators on a reference element and to apply them element-wise throughout the domain. For example, it can be shown that for a hierarchical basis, I_{p-1}^p is the identity matrix with zero rows appended, and I_p^{p-1} is the identity matrix with zero columns appended. In general, $I_p^{p-1} \neq \tilde{I}_p^{p-1}$.

Implementation

One consideration in the implementation of the multigrid scheme is storage. For each level we have defined state, residual, and source vectors. However, since the residual data can be overwritten, it is not necessary to allocate three separate vectors for each level. Figure 1 shows the implementation used for a general basis.

As shown, only one source vector of adequate size is allocated. During restriction, the state vector is transferred directly via $\mathbf{u}^{p-1} = \tilde{I}_p^{p-1}\mathbf{u}^p$. Three transfers then take place regarding the residual and source terms. First, $\mathbf{R}^p$ is restricted via $\mathbf{R}^{p-1} = I_p^{p-1}\mathbf{R}^p$. Second, the source term used by level p is stored in the residual vector. Finally, the coarse level residual is transferred to the source term: $\mathbf{f}^{p-1} = \mathbf{R}^{p-1}$. Analogous steps are taken

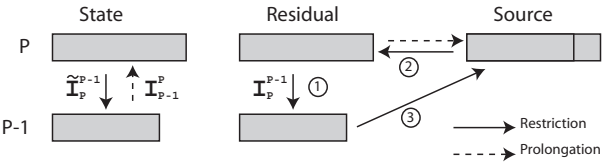

Fig. 1 Diagram of storage in multigrid algorithm

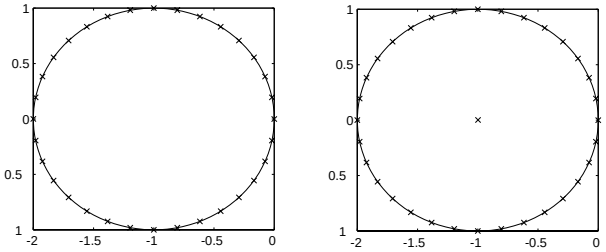

Fig. 2 Smoothing footprint for $p = 0$ (a) and $p > 0$ (b)

when transferring to the next coarser level. During prolongation, the state $\mathbf{u}^p$ is corrected by interpolating the difference between the coarse level solution and the restriction of the fine level solution. In addition, the source term is restored from the residual vector.

Since prolongation introduces a $d\mathbf{u}$ correction to the solution, the update goes through the standard limiting process to make sure it does not drive the solution unstable. In the interest of time and stability, FMG was used as the standard multigrid solver.

ANALYSIS

We first present the results of a stability analysis of DG applied to the advection problem,

$$
\begin{aligned}
\vec{V}\cdot\nabla u &= f(\vec{x}), \\
a u_x &= f(x) \qquad (1D), \\
a u_x + b u_y &= f(x,y) \quad (2D).
\end{aligned} \tag{11}
$$

In this problem, the velocity $\vec{V}$ is constant, u is the unknown concentration variable, and f is the source function. The problem is defined on the interval $[-1,1]$ ($[-1,1]\times[-1,1]$ in 2D) with periodic boundary conditions, which allow for a spectral stability analysis.

1D

In 1D, we consider the use of the elemental block Jacobi scheme as a smoother for higher order discretization. Fourier (Von Neumann) analysis is used to determine the footprint, which consists of the eigenvalues of $-\mathbf{M}^{-1}\mathbf{A}$, where $\mathbf{A}$ is the resulting linear operator from the DG discretization of the scalar convection problem. For this 1D problem, the eigenvalues and eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$ can be computed analytically with the result that the elemental block Jacobi scheme is stable independent of order. The footprints of the $p = 0$ and $p > 0$ preconditioned operators are shown in Figure 2. For $p = 0$, the eigenvalues are identical to those obtained from the traditional upwind finite-difference scheme, as the schemes

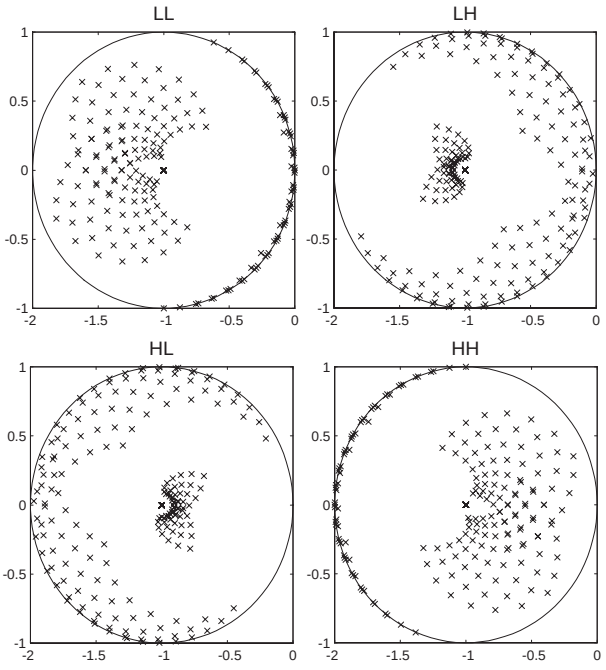

Fig. 3 Block Jacobi footprint for $\alpha = 25^o$

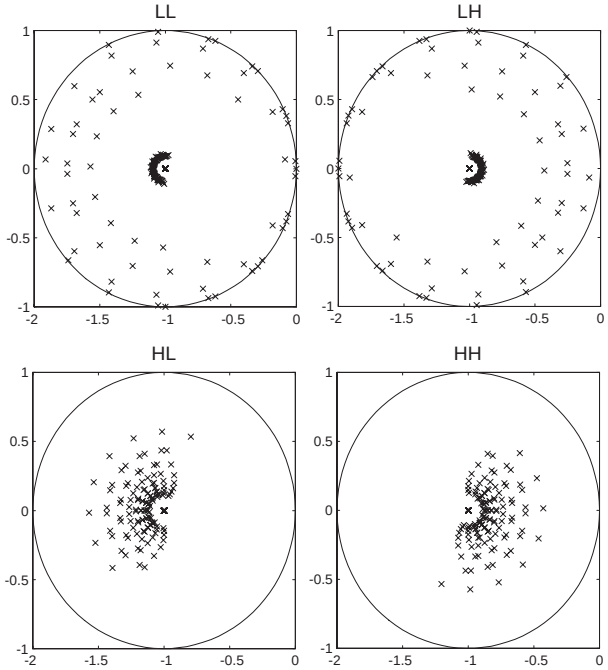

Fig. 4 Line Jacobi footprint for $\alpha = 25^o$

are identical. For higher order, the additional eigenvalues all lie at -1, implying that the smoother will converge at a rate that is independent of the order and guaranteeing p-independent multigrid as well.

2D

In 2D, we analyze the stability and smoothing of both the elemental block Jacobi and the line block Jacobi algorithm. While the eigenvalues cannot be calculated analytically, numerical results show that $|\rho(-\mathbf{M}^{-1}\mathbf{A} - I)| \leq 1$ for all flow angles $\alpha = tan^{-1}(b/a)$ for both preconditioners. For multigrid studies, it is useful to differentiate between low (L) and high (H) frequency modes. We do this by defining the L modes to be those with $-\pi/2 < \theta \leq \pi/2$. All other values of θ correspond to H modes. This separation is ideal for h-multigrid in which the grid size is halved on each finer grid. For p-multigrid, the ideal separation is not clear, but the stated separation is used for simplicity. With this distinction, we can separate the eigenvalues based on the mode pair (θ_j, θ_k) of the eigenvectors: LL, LH, HL, or HH. Figure 3 shows this separation for the case of $\alpha = 25^o$.

The modes least affected by under-relaxed-Jacobi smoothing are those with footprint eigenvalues closest to 0 in the complex unit disk. As expected, the HH modes are effectively reduced by the iterative method. The LL modes are densely clustered near 0, and, as in 1D, we cannot expect to reduce these errors without some form of multigrid. The HL modes are effectively reduced by the smoother, but the LH modes are not reduced, a consequence of the flow being aligned more in the x-direction for $\alpha = 25^o$. Since the LH modes contain high frequency components, they cannot be represented on uniformly-coarsened grids to be affected by multigrid. The presence of these errors stalls the convergence and degrades the performance of multigrid with block-Jacobi smoothing.

A line smoother is capable of reducing the "LH" modes - i.e. the modes that are low frequency in the flow direction and high frequency in the transverse direction. The line updates are performed in Jacobi fashion in that the unknown values for all elements not on the line are held constant when calculating the update. Line smoothing is most effective in cases where the lines are aligned with the flow direction. Figure 4 shows the footprint of line smoothing for the case of $\alpha = 25^o$ and horizontal (x-aligned) lines. In comparison to Block Jacobi (Figure 3), Line Jacobi is more effective at reducing the HH, HL, and LH modes. Smoothing of the LH modes still results in some eigenvalues close to 0, a consequence of the 25^o difference between the flow angle and the line angle.

RESULTS

We now present accuracy and solver performance results for three smooth problems: Ringleb flow, flow over a Gaussian bump, and flow over a Joukowski airfoil. For each problem, uniform grid refinement was performed to study the accuracy of the discretization. Performance was determined from timing and convergence rate of the FMG scheme. The following parameters were used in all the cases:

- - Hierarchical basis, as given in Šolín et. al.[21]
- - $\nu_1 = 4$ pre-smoothing sweeps and $\nu_2 = 4$ post-smoothing sweeps
- - $\nu_c = 100$ sweeps on the coarsest level, $p = 0$
- - Memory-lean line solver with one linear iteration per non-linear iteration
- - Residual-based level switching criterion for FMG

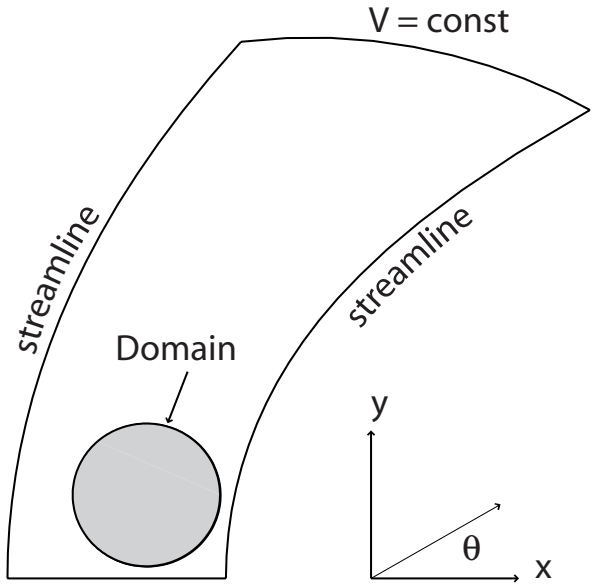

Fig. 5 Ringleb Flow

- - Initialization with a converged solution on $p = 0$

All timing runs were performed on an Intel Pentium 4 2.53 GHz system with 512 MB RAM.

Ringleb Flow

Ringleb Flow is an exact solution of the Euler equations obtained using the hodograph method. The streamlines and iso-Mach lines for a typical Ringleb solution domain are shown in Figure 5.

The relevant transformation equations between the Cartesian variables (x, y) and the hodograph variables (V, θ) are,

$$\begin{aligned}
\Psi &= \frac{1}{V} sin(\theta) \\
c^2 &= 1 - \frac{\gamma - 1}{2} V^2 \\
x &= \frac{1}{2\rho}\left[\frac{1}{V^2} - 2\Psi^2\right] + \frac{J}{2} \\
y &= \pm \frac{\Psi}{\rho V} cos(\theta) \\
J &= \frac{1}{c} + \frac{1}{3c^3} + \frac{1}{5c^5} - \frac{1}{2} log \frac{1+c}{1-c} \\
\rho &= c^{2/(\gamma-1)}
\end{aligned}$$

Taking advantage of the fact that the exact solution is known, we take as our domain the circle shown inside the regular Ringleb domain. The boundary condition is imposed by setting the exact state just outside the domain, and using this state in the flux function. An accuracy study was performed using a set of three hierarchical grids (88, 352, and 1408 elements). Orders of interpolation ranging from $p = 0$ to $p = 3$ were used, and the output of interest was the $L2$ norm of the error. Each case was converged to machine zero residual.

Figure 6 shows the solution accuracy versus grid size and order. Optimal accuracy convergence of $p + 1$ is

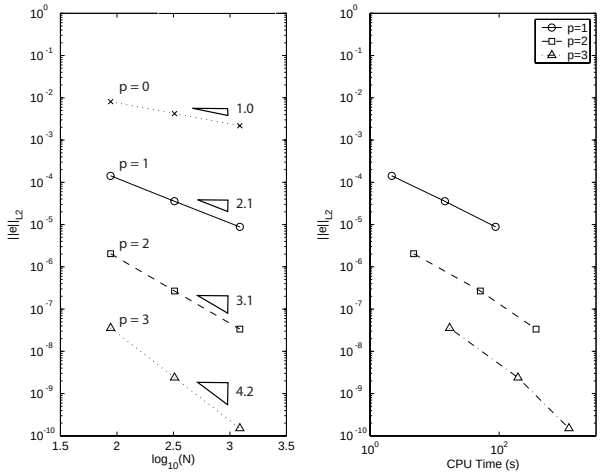

Fig. 6 Ringleb Flow: accuracy vs. CPU time

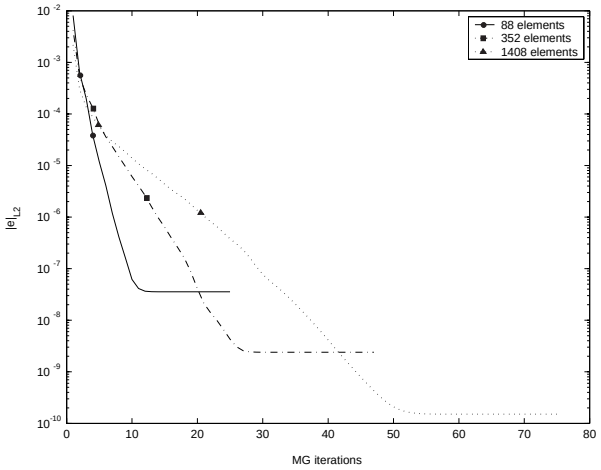

Fig. 7 Ringleb Flow: Error convergence history

attained, in that $||e||_{L2} = Ch^{p+1}$. Figure 6 also shows the error plotted versus CPU time to solution. A solution was taken to be converged when the error norm came within 1 percent of its final value, determined by converging the solution to machine zero residual beforehand. The advantage of high order interpolation is clear: a $p = 3$ solution on the coarsest grid yields the same accuracy as a $p = 2$ solution on a grid 16 times the size in a time of 17 seconds as compared to 352 seconds.

The accuracy convergence histories for $p = 3$ FMG solutions on each grid are shown in Figure 7. Grid dependence is evident, showing one of the drawbacks of the solution algorithm. However, also evident is the advantage of p-refinement. The symbols on each plot denote switches to the next higher order. In stepping up the order level from $p = 1$ to $p = 3$, the convergence rate per iteration does not degrade. Thus, the convergence rate of the V-cycle appears order-independent for a particular grid.

The corresponding residual histories for each grid are shown in Figure 8. We see that for each grid it is not necessary to converge the residual to machine zero to

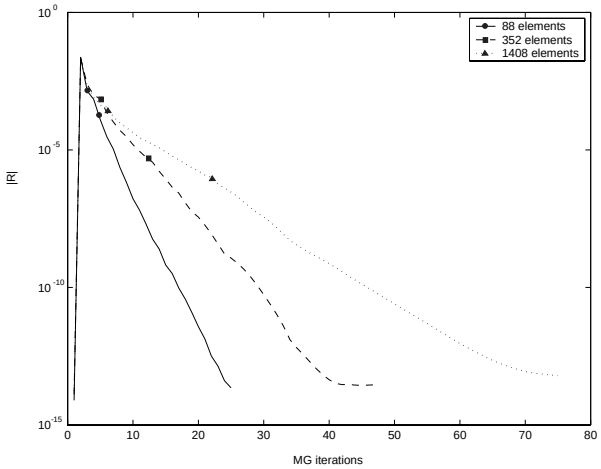

Fig. 8 Ringleb Flow: Residual convergence history

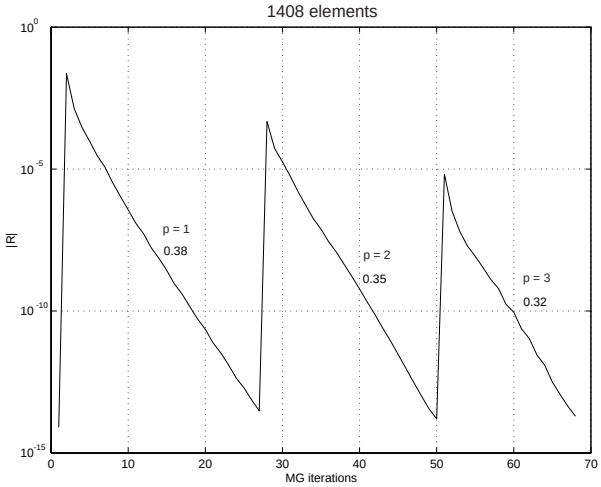

Fig. 9 Ringleb Flow: FMG residual history with full convergence on each level

obtain accuracy to the discretization level. Finally, to illustrate the asymptotic multigrid rates, Figure 9 shows the residual history for FMG with full convergence on each level. We note that the rate does not degrade with increasing order.

Flow over a Gaussian Bump

The second test problem is that of channel flow over a Gaussian bump. The problem setup is shown in Figure 10. The channel height is 12σ, the channel length is 24σ, and the bump height is 0.4σ, where σ is the standard deviation of the Gaussian. Wall boundary conditions were enforced on the top and bottom channel boundaries. At the outflow, the static pressure was set constant and at the inflow, the total temperature, the total pressure, and the flow angle (0^o) were prescribed, resulting in a free-stream flow of $M = 0.2$. The output of interest in this case was the L_2 norm of the entropy error, $||S - S_{fs}||_{L2}$, where S_{fs} is the free-stream entropy.

Again, three hierarchical grids (587, 2348, 9392 elements) were used in a hierarchical study. The results are shown in Figure 11. As in the Ringleb case, optimal

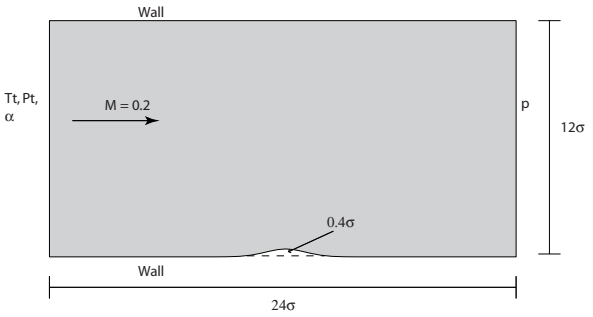

Fig. 10 Domain for flow over a Gaussian bump

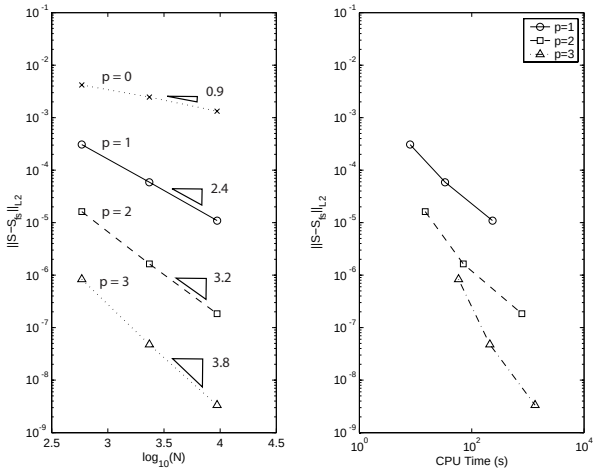

Fig. 11 Gaussian bump: accuracy vs. CPU time

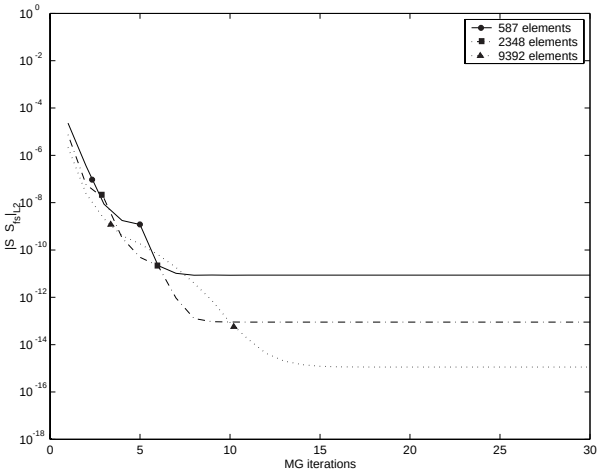

Fig. 12 Gaussian bump: Entropy error convergence history

error convergence of $p + 1$ is attained. Figure 11 also shows the accuracy versus CPU time for each run. The advantage of higher order for obtaining accurate solutions is again evident.

The accuracy convergence histories are shown in Figure 12. Grid dependence is apparent but not significant. The step-like behavior of the error is due to the rapid er-

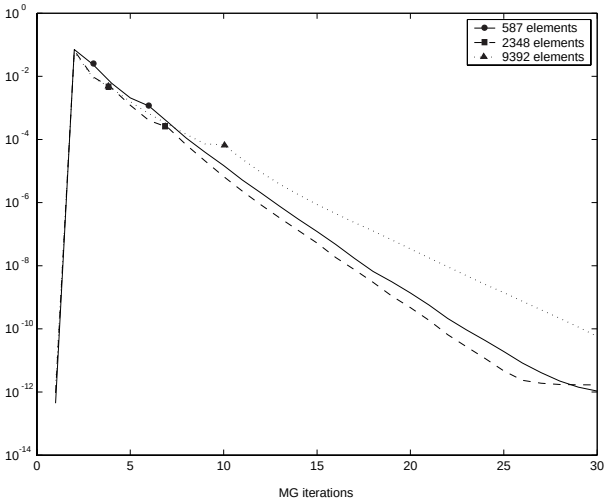

Fig. 13 Gaussian bump: Residual convergence history

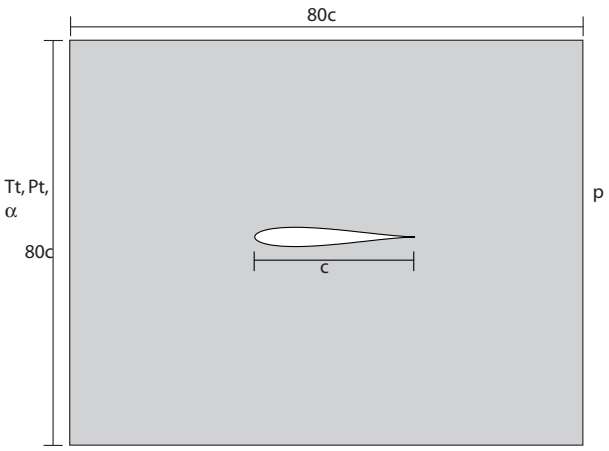

Fig. 14 Domain for flow over a Joukowski airfoil (not to scale)

ror reduction following a transfer to a finer level, which is a characteristic of an effective multigrid scheme.

Figure 13 shows the residual histories for each run. We note the relatively constant convergence rate with respect to order, and the fact that the entropy error output bottoms out well before the residual is converged to machine zero.

Joukowski Airfoil

The third test problem is a 12 percent thick Joukowski airfoil at $M = 0.2$ and $\alpha = 0^o$. The airfoil was created using a standard Joukowski transformation. The computational domain is shown in Figure 14. Total temperature, total pressure, and flow angle were specified at the inlet, static pressure was specified at the outlet, and the free-stream state was prescribed at the top and bottom boundaries. For this case, the output of interest was the absolute value of the drag on the airfoil, which should approach zero.

The results of an accuracy study (grid sizes of 974, 3896, and 15584 elements) are shown in Figure 15. Optimal convergence of $p + 1$ is roughly attained, although the error on the finest grid $p = 3$ solution appears to bottom out. This effect is likely a consequence of a singularity caused by the finite trailing edge angle and the inviscid flow assumption. We expect this effect to disappear with the introduction of viscous modeling.

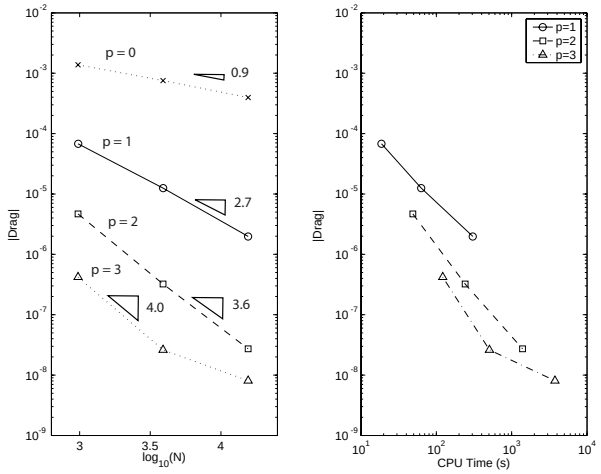

Fig. 15 Joukowski airfoil: accuracy vs. CPU time

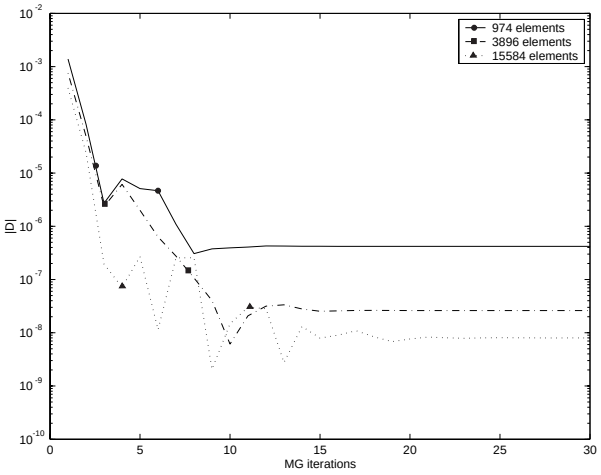

Fig. 16 Joukowski airfoil: Drag convergence history

The accuracy and residual histories are shown in Figures 16 and 17, respectively. The drag convergence appears more oscillatory than the previous cases, but still proceeds faster than the residual convergence. Grid dependence exists, but does not appear to be strong.

CONCLUSIONS

We have presented the details of a higher-order solution method for DG applied to the Euler equations. The p-multigrid algorithm fits naturally into the local high-order finite element discretization. Grid transfer operators are local and become trivial to implement in the case of a hierarchical basis. The line smoother is effective at removing the error modes associated with convection, as predicted by the 2D analysis of an advection problem. The results for three smooth problems

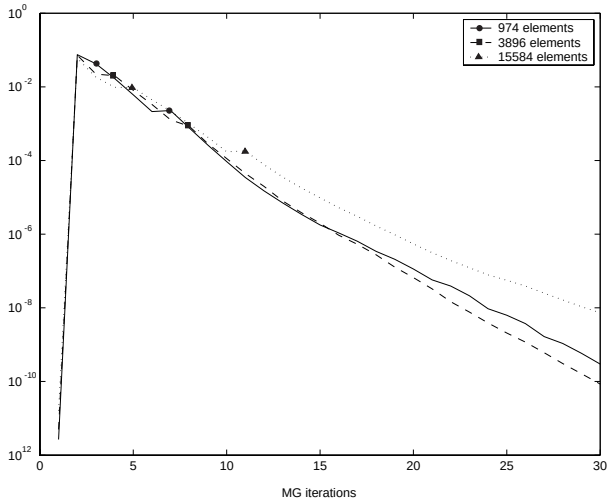

Fig. 17 Joukowski airfoil: Residual convergence history

illustrate how the coupling of the line smoother with an FMG scheme yields an effective and efficient solver.

Optimal accuracy convergence is obtained for the three different outputs of interest: the L_2 error norm in the Ringleb case, the L_2 entropy norm in the bump case, and the drag in the Joukowski airfoil case. Residual history plots show grid dependence, which is not strong in the bump and airfoil cases. Most importantly, the timing results demonstrate the benefit and practicality of using higher order for attaining high accuracy. Using the solution algorithm presented, the lower-order discretizations require highly refined grids and more computational time to attain the levels of accuracy of higher order discretizations on coarser grids.

FUTURE DIRECTIONS

The cases considered in this paper have been smooth and hence suitable for showing the benefit of high order. For problems with shocks, limiting is required to stabilize the oscillatory behavior of high-order approximations near discontinuities. A limiter has not yet been implemented, and its effects on accuracy and convergence rate are unknown, although decreased performance is expected. In addition to the formulation of a limiter, we will also introduce viscous modeling, with appropriate modifications to the solver. Following these additions, the next steps are parallelization and turbulence modeling, and performance comparisons for practical cases of interest.

References

[1]Jameson, A., Schmidt, W., and Turkel, E., "Numerical simulation of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes," AIAA-81-1259, 1981.

[2]Jameson, A., "Solution of the Euler equations for two-dimensional transonic flow by a multigrid method," *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 327–356.

[3]Jameson, A., Baker, T. J., and Weatherhill, N. P., "Calculation of Inviscid Transonic Flow over a Complete Aircraft," 1986.

[4]Van Leer, B., "Flux-vector splitting for the Euler equations," Tech. Rep. 81-11, ICASE, 1981.

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

[6]Van Leer, B., "Upwind-difference methods for aerodynamic problems governed by the Euler equations," *Lectures in Applied Mathematics*, Vol. 22, 1985.

[7]Roe, P., "Characteristic-based schemes for the Euler equations," *Ann. Rev. Fluid Mech.*, Vol. 18, 1986, pp. 337–65.

[8]Van Leer, B., Thomas, J., Roe, P., and Newsome, R., "A Comparison of numerical flux formulas for the Euler and Navier-Stokes equations," AIAA Paper 87-1104, 1987.

[9]Anderson, W. K., Rausch, R. D., and Bonhaus, D. L., "Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids," *J. Comput. Phys.*, Vol. 128, 1996, pp. 391–408.

[10]Pierce, N. and Giles, M., "Preconditioned multigrid methods for compressible flow calculations on stretched meshes," *Journal of Computational Physics*, Vol. 136, 1997, pp. 425–445.

[11]Mavriplis, D., "Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes," *Journal of Computational Physics*, Vol. 145, 1998, pp. 141–165.

[12]Mavriplis, D. and Pirzadeh, S., "Large-scale parallel unstructured mesh computations for 3D high-lift analysis," *AIAA Journal of Aircraft*, Vol. 36, 1999, pp. 987–998.

[13]Mavriplis, D., "An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers," *Journal of Computational Physics*, Vol. 175, 2001, pp. 302–325.

[14]Venkatakrishnan, V., Allmaras, S., Kamenetskii, D., and Johnson, F., "Higher order schemes for the compressible Navier-Stokes equations," AIAA Paper 2003-3987, 2003.

[15]Cockburn, B. and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *Journal of Scientific Computing*, 2001, pp. 173–261.

[16]Allmaras, S., "Analysis of semi-implicit preconditioners for multigrid solution of the 2-D compressible Navier-Stokes equations," 1995.

[17]Rønquist, E. M. and Patera, A. T., "Spectral element multigrid. I. Formulation and numerical results," *J. Sci. Comput.*, Vol. 2(4), 1987, pp. 389–406.

[18]Helenbrook, B., Mavriplis, D., and Atkins, H., "Analysis of p-multigrid for continuous and discontinuous finite element discretizations," AIAA Paper 2003-3989, 2003.

[19]Briggs, W., Henson, V. E., and McCormick, S. F., *A Multigrid Tutorial, 2nd Ed.*, SIAM, 2000.

[20]Brandt, A., *Guide to Multigrid Development*, Springer-Verlag, 1982.

[21]Solín, P., Segeth, K., and Zel, I. D., *Higher-Order Finite Element Methods*, Chapman and Hall, 2003.