Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 207 (2005) 92–113

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations

Krzysztof J. Fidkowski, Todd A. Oliver, James Lu, David L. Darmofal *

Aerospace Computational Design Laboratory, Massachusetts Institute of Technology, Building 37, Room 401, USA

Received 5 October 2004; received in revised form 10 January 2005; accepted 10 January 2005
Available online 25 February 2005

Abstract

We present a p-multigrid solution algorithm for a high-order discontinuous Galerkin finite element discretization of the compressible Navier–Stokes equations. The algorithm employs an element line Jacobi smoother in which lines of elements are formed using coupling based on a $p = 0$ discretization of the scalar convection–diffusion equation. Fourier analysis of the two-level p-multigrid algorithm for convection–diffusion shows that element line Jacobi presents a significant improvement over element Jacobi especially for high Reynolds number flows and stretched grids. Results from inviscid and viscous test cases demonstrate optimal h^{p+1} order of accuracy as well as p-independent multigrid convergence rates, at least up to $p = 3$. In addition, for the smooth problems considered, p-refinement outperforms h-refinement in terms of the time required to reach a desired high accuracy level.
© 2005 Published by Elsevier Inc.

1. Introduction

While computational fluid dynamics has achieved significant maturity, computational costs can be extremely large when high accuracy simulations are required. The development of a practical high-order solution method could diminish this problem by significantly decreasing the computational time required to achieve an acceptable error level. An attractive approach for achieving high-order accuracy is the discontinuous Galerkin (DG) formulation, in which element-to-element coupling exists only through fluxes at the shared boundaries between elements. This limited coupling is an enabling feature which permits the development of an efficient high-order solver. Furthermore, the DG formulation is well-suited for general unstructured meshes, and, due to its elementwise discontinuous representation, it is a natural method for h and p adaptation.

* Corresponding author. Tel.: +1 617 258 0743x5548; fax: +1 617 258 5143.
E-mail address: darmofal@mit.edu (D.L. Darmofal).

0021-9991/$ - see front matter © 2005 Published by Elsevier Inc.
doi:10.1016/j.jcp.2005.01.005

DG methods have been developed extensively for hyperbolic conservation laws, and, to a lesser extent, for diffusive operators. Bassi and Rebay [1–3] have performed numerous studies for the Euler and Navier–Stokes equations, showing the potential benefits of high-order accuracy with DG in these applications. Further details on DG methods are given by Cockburn and Shu [4], who outline the state-of-the-art in this field and provide implementation details.

As noted by Cockburn and Shu, an area which has received less attention is solution algorithms for high-order DG discretizations. This paper considers the application of multigrid for the solution of high-order DG discretizations of the compressible Navier–Stokes equations. Multigrid for the solution of the Euler equations was pioneered by Jameson in 1983 [5], and since then there have been numerous advances in the field. In particular, Mavriplis extended multigrid to unstructured meshes [6], and Allmaras [7] presented a thorough 2D preconditioner analysis investigating the requirements for the proper elimination of all error modes. In addition, Pierce and Giles [8] studied efficient preconditioners for the Euler and Navier–Stokes equations on structured meshes, preconditioners which were then applied to anisotropic, unstructured meshes by Mavriplis [9].

In the context of DG approximations, Hemker et al. [10] analyzed block Jacobi smoothing strategies with *h*-multigrid for 1D diffusion problems. This work shows that for diffusion, better iterative performance can be attained by forming blocks from unknowns on the interfaces between elements, as opposed to the more apparent choice of element-wise partitioning. In addition, *p*-multigrid, or multi-order, solution strategies have been studied for high-order DG [11,12], showing several advantages such as ease of implementation and order-independent convergence rates.

In this paper, a *p*-multigrid algorithm with an element line Jacobi smoother is presented for the solution of high-order DG discretizations of the compressible Navier–Stokes equations [13–15]. Section 2 gives a description of a DG discretization for the compressible Navier–Stokes equations developed by Bassi and Rebay [3] and used throughout this paper. Section 3 then presents the *p*-multigrid and element line Jacobi algorithms. The element lines are formed from the strongest element-to-element coupling as determined by a low-order ($p = 0$) discretization of the convection–diffusion equation. Fourier analysis of the two-level *p*-multigrid algorithm for convection–diffusion shows that element line Jacobi is significantly better than element Jacobi especially for high Reynolds number flows and stretched grids. Finally, computational results for some sample aerodynamic applications confirm the benefits of the *p*-multigrid and element line smoothing and show that high-order solutions obtained with this algorithm can provide significant reductions in the computational time required to achieve desired accuracy levels.

2. Discontinuous Galerkin discretization

The steady, 2D, compressible Navier–Stokes equations in strong, conservation form are given by

$$\nabla \cdot \mathscr{F}_i(\mathbf{u}) - \nabla \cdot \mathscr{F}_v(\mathbf{u}, \nabla\mathbf{u}) = 0, \tag{1}$$

where $\mathbf{u} = (\rho, \rho u, \rho v, \rho E)^{\mathrm{T}}$ is the conservative state vector, $\mathscr{F}_i = (\mathbf{F}_i^x, \mathbf{F}_i^y)$ is the inviscid flux, and $\mathscr{F}_v = \mathscr{A}_v \nabla\mathbf{u} = (\mathbf{F}_v^x, \mathbf{F}_v^y)$ is the viscous flux.

Denote by $\mathscr{V}_h^p$ the space of discontinuous vector-valued polynomials of degree p on a subdivision T_h of the domain Ω into elements such that $\Omega = \bigcup_{\kappa \in T_h} \kappa$. The DG discretization of (1) is given by the following form: find $\mathbf{u}_h \in \mathscr{V}_h^p$ such that $R(\mathbf{w}_h, \mathbf{u}_h) = 0$, $\forall \mathbf{w}_h \in \mathscr{V}_h^p$, where

$$R(\mathbf{w}_h, \mathbf{v}_h) \equiv -\sum_{\kappa \in T_h} \int_\kappa \nabla\mathbf{w}_h^{\mathrm{T}} \cdot \mathscr{F}_i(\mathbf{v}_h) \mathrm{d}\mathbf{x} + \int_{\Gamma_i} \left(\mathbf{w}_h^+ - \mathbf{w}_h^-\right)^{\mathrm{T}} \mathscr{H}_i(\mathbf{v}_h^+, \mathbf{v}_h^-, \hat{\mathbf{n}}) \mathrm{d}s + \int_{\partial\Omega} \mathbf{w}_h^{+\mathrm{T}} \mathscr{H}_i^b(\mathbf{v}_h^+, \mathbf{v}_h^b, \hat{\mathbf{n}}) \mathrm{d}s$$
$$+ \mathbb{V}(\mathbf{w}_h, \mathbf{v}_h), \tag{2}$$

and $\mathbb{V}(\mathbf{w}_h, \mathbf{v}_h)$ denotes the discretization of the viscous terms. In (2), Γ_i is the union of interior faces, and $\partial\Omega$ is the domain boundary. The $()^+$ and $()^-$ notation indicates the trace value taken from opposite sides of a face; $\hat{\mathbf{n}}$ is the unit normal pointing outward from the $()^+$ elements, and $\mathscr{H}_i(\mathbf{v}_h^+, \mathbf{v}_h^-, \hat{\mathbf{n}})$ and $\mathscr{H}_i^b(\mathbf{v}_h^+, \mathbf{v}_h^b, \hat{\mathbf{n}})$ are numerical flux functions approximating $\mathscr{F}_i$ on interior and boundary faces, respectively. In this work, the Roe flux was used for $\mathscr{H}_i(\mathbf{v}_h^+, \mathbf{v}_h^-, \hat{\mathbf{n}})$ [16].

The viscous terms are discretized using the second form of Bassi and Rebay (BR2) [3]. This discretization was chosen because it achieves optimal order of accuracy while maintaining a compact stencil [17]. To write the discretization in a concise form, jump, $[\![\cdot]\!]$, and average, $\{\cdot\}$, operators are defined on interior faces as follows:

$$[\![s]\!] = s^+\hat{\mathbf{n}}^+ + s^-\hat{\mathbf{n}}^-, \qquad \{\boldsymbol{\varphi}\} = \frac{1}{2}(\boldsymbol{\varphi}^+ + \boldsymbol{\varphi}^-),$$

where s is a spatial scalar quantity and $\boldsymbol{\varphi}$ is a spatial vector quantity. Both operators produce spatial vectors. Using this notation, $\mathbb{V}(\mathbf{w}_h, \mathbf{v}_h)$ is given by

$$\begin{aligned}\mathbb{V}(\mathbf{w}_h, \mathbf{v}_h) = {} & \sum_{\kappa\in T_h} \int_\kappa \nabla\mathbf{w}_h^{\mathrm{T}} \cdot (\mathscr{A}_v \nabla\mathbf{v}_h)\mathrm{d}\mathbf{x} - \int_{\Gamma_i} \left([\![\mathbf{v}_h]\!]^{\mathrm{T}} \cdot \{\mathscr{A}_v^{\mathrm{T}}\nabla\mathbf{w}_h\} + [\![\mathbf{w}_h]\!]^{\mathrm{T}} \cdot \{\mathscr{A}_v\nabla\mathbf{v}_h\}\right)\mathrm{d}s \\ & + \int_{\Gamma_i} \eta_f [\![\mathbf{w}_h]\!]^{\mathrm{T}} \cdot \{\boldsymbol{\delta}_f\}\,\mathrm{d}s + \int_{\partial\Omega} (\mathbf{v}_h^b - \mathbf{v}_h^+)^{\mathrm{T}} (\mathscr{A}_v^{\mathrm{T}}\nabla\mathbf{w}_h)^+ \cdot \hat{\mathbf{n}}\,\mathrm{d}s - \int_{\partial\Omega} \mathbf{w}_h^{+\mathrm{T}} (\mathscr{A}_v\nabla\mathbf{v}_h)^b \cdot \hat{\mathbf{n}}\,\mathrm{d}s \\ & + \int_{\partial\Omega} \eta_f \mathbf{w}_h^{+\mathrm{T}} \boldsymbol{\delta}_f^b \cdot \hat{\mathbf{n}}\,\mathrm{d}s, \end{aligned} \tag{3}$$

where $\boldsymbol{\delta}_f$ and $\boldsymbol{\delta}_f^b$ are auxiliary variables associated with faces and defined by the following weak statement: find $\boldsymbol{\delta}_f \in [\mathscr{V}_h^p]^2$ such that $\forall \boldsymbol{\tau}_h \in [\mathscr{V}_h^p]^2$,

$$\int_\Omega \boldsymbol{\tau}_h^{\mathrm{T}} \cdot \boldsymbol{\delta}_f\,\mathrm{d}\mathbf{x} = \int_{\sigma_f} [\![\mathbf{v}_h]\!]^{\mathrm{T}} \cdot \{\mathscr{A}_v^{\mathrm{T}}\boldsymbol{\tau}_h\}\,\mathrm{d}s \tag{4}$$

for interior faces, and

$$\int_\Omega \boldsymbol{\tau}_h^{\mathrm{T}} \cdot \boldsymbol{\delta}_f^b\,\mathrm{d}\mathbf{x} = \int_{\sigma_f} (\mathbf{v}_h^+ - \mathbf{v}_h^b)^{\mathrm{T}} [(\mathscr{A}_v^{\mathrm{T}}\boldsymbol{\tau}_h) \cdot \hat{\mathbf{n}}]^+\,\mathrm{d}s, \tag{5}$$

for boundary faces, where σ_f denotes the face indexed by f. The auxiliary variables are locally eliminated from (2) by solving (4) or (5) at each face in the domain. See [3,14] for a more detailed derivation of the BR2 discretization.

The quantity η_f appearing in (3) is a stabilization parameter. The discretization originally proposed by Bassi and Rebay [3] does not include this additional parameter and is therefore equivalent to $\eta_f = 1$. However, the analysis by Brezzi et al. [18] shows that a sufficient condition for stability is $\eta_f \geqslant 3$ for discretization of the 2D Poisson equation on triangular grids. For $p \geqslant 1$, optimal accuracy can be proven using this stability result [18,17]. However, for $p = 0$, the BR2 discretization does not produce an optimally accurate solution for $\eta_f \geqslant 3$ [14]. To recover optimal accuracy for $p = 0$, the following definition of η_f is used at interior edges of triangular elements:

$$\eta_f = \frac{3}{1 + \frac{1}{2}\left(\frac{h_\sigma^+}{h_\sigma^-} + \frac{h_\sigma^-}{h_\sigma^+}\right)},$$

where h_σ^+ and h_σ^- are the heights of the elements adjoining edge σ. For boundary edges, $\eta_f = 3/2$.

To obtain high-order convergence rates, curved boundaries must be approximated to high order. For DG discretizations of inviscid flows, Bassi and Rebay [1] have shown that superparametric representations are required for $p = 1$ discretizations. For the results reported in this paper, piecewise cubic interpolations of the geometry were used regardless of the order of solution approximation ($p \leqslant 3$). Even in these curved boundary elements, the solution basis is polynomial in the physical space – that is, the basis functions are defined in a reference element different from the one used in numerical integration. Appendix A gives a description of the implementation of such a basis.

The boundary conditions on $\partial\Omega$ are imposed weakly by constructing an exterior boundary state, $\mathbf{v}_h^b$, and a normal derivative of the state, $(\nabla\mathbf{v}\cdot\hat{\mathbf{n}})^b$, that are functions of the interior quantities and boundary condition data. These quantities are then used to construct the inviscid and viscous fluxes at the domain boundaries, as described in Appendix B.

The final discrete form is obtained by choosing a basis for the polynomial approximation space. In this work, two types of basis functions are used: a nodal Lagrange basis with equispaced nodes, and a hierarchical basis composed of vertex, edge, and bubble functions, as described in [19]. The hierarchical basis offers better conditioning and, as described in Section 3.3, simplifies the multigrid prolongation and restriction operators. Regardless of the particular basis, the DG discretization produces a system of nonlinear, algebraic equations, denoted as

$$\mathbf{r}(\mathbf{u}) = 0, \tag{6}$$

where $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$ is the vector of degrees of freedom for the chosen basis such that

$$\mathbf{u}_h(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{u}_i \phi_i(\mathbf{x}),$$

and $\phi_i(\mathbf{x})$ are the piecewise discontinuous, polynomial basis functions.

3. Solution method

3.1. Preconditioners

To smooth the errors on a given discretization level, a general preconditioned iterative scheme is used,

$$\mathbf{v}^{n+1} = \mathbf{v}^n - \omega\mathbf{P}^{-1}\mathbf{r}(\mathbf{v}^n), \tag{7}$$

where the preconditioner matrix, $\mathbf{P}$, is an approximation to the Jacobian, $\partial\mathbf{r}/\partial\mathbf{v}$, and ω is the under-relaxation factor. Two types of preconditioners are examined: element Jacobi and element line Jacobi. In element Jacobi, the unknowns on a single element are solved for simultaneously. Thus, the initial preconditioner matrix, $\mathbf{M}^{\text{elem}}$, is the block diagonal of the Jacobian.

The element line Jacobi scheme improves upon element Jacobi by solving implicitly on lines of elements connected along directions of strong convection or grid anisotropy. For example, for viscous flows, line preconditioning is an important ingredient in removing the stiffness associated with regions of high grid anisotropy frequently used in viscous layers [7,9,20].

The measure of coupling used in constructing the lines is based on a $p = 0$ discretization of the scalar transport operator,

$$\nabla\cdot(\rho\mathbf{u}s) - \nabla\cdot(\mu\nabla s),$$

where $\rho\mathbf{u}$ and μ are taken from the solution at the current iteration. Specifically, the coupling between two elements j and k that share a face is given by, $C_{j,k} \equiv \max(|\partial r_j/\partial s_k|, |\partial r_k/\partial s_j|)$, where r is the discrete scalar transport operator for $p = 0$. The initial preconditioner $\mathbf{M}^{\text{line}}$ consists of block-tridiagonal subsystems formed from $\partial\mathbf{r}/\partial\mathbf{v}$ for each line of elements.

To improve the robustness during the initial iterations, the preconditioners for the element and element line smoothers are augmented by an unsteady term, resulting in $\mathbf{P} = \mathbf{M} + \mathscr{M}/\Delta t$, where $\mathbf{M}$ is $\mathbf{M}^{\text{elem}}$ or $\mathbf{M}^{\text{line}}$ and $\mathscr{M}$ is the mass matrix

$$\mathscr{M}_{i,j} = \int_{\Omega} \phi_i \phi_j \, d\mathbf{x}.$$

Note that for the DG approximation, $\mathscr{M}$ is a block diagonal matrix since each basis function is nonzero inside only a single element. As the solution converges, $\Delta t \to \infty$. Specifically, Δt is first calculated using the initialized flow and an initial *CFL* number, CFL_0. Following each successful multigrid iteration, the *CFL* number is augmented according to $CFL = \eta_{CFL} * CFL$, and Δt is recalculated. A successful iteration is defined as one which does not require *extra* under-relaxation (that is, aside from ω) in the state update or the multigrid prolongation. Such extra under-relaxation is performed to ensure physical values for variables such as density and pressure. In particular, changes in pressure and density are limited to β percent of their current values. If extra under-relaxation is required, the *CFL* number is not increased. As the solution converges, the requested updates decrease in magnitude so that no extra under-relaxation is required, leading to $CFL \to \infty \Rightarrow \Delta t \to \infty$. In practice, $CFL_0 = 1$, $\eta_{CFL} = 10$, and $\beta = 10\%$ are used.

Inversion of $\mathbf{P}$ in the element line smoother uses a block-tridiagonal algorithm, which involves the LU decomposition of the block diagonal. As the dominant cost of the element line Jacobi solver (especially for high-order schemes) is this LU decomposition, the computational cost of the element line Jacobi smoother scales as that of the simpler element Jacobi. However, the performance of the element line Jacobi smoother is significantly better due to the increased implicitness along strongly coupled directions.

3.2. Line creation

The effectiveness of the element line Jacobi smoother depends on the length and quality of lines used. After computing the elemental coupling, lines of elements are formed using the line creation algorithm developed by Fidkowski [13]. The line creation process is divided into two stages: line creation and line connection. In describing line creation, let $N(j;f)$ denote the element adjacent to element j across face f, and let $F(j)$ denote the set of faces enclosing element j. Then, the line creation algorithm proceeds as follows:

```
Stage I: Line creation
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 face $f$ is a boundary face or if $C_{j,k}$ is not one of the top two distinct connectivities in element $k$ (terminate if top two distinct connectivities do not exist). Otherwise, assign element $j$ to the current line, set $j = k$, and continue.
```

After completing Stage I, it is possible that the endpoints of two lines are adjacent to each other [13]. However, for best solver performance, it is desirable to use lines of maximum length. Thus, Stage II, or line connection, is employed:

Stage II: Line connection

1. Loop through endpoint elements, j, of all lines. Denote by $H_j \subset F(j)$ the set of faces h of j that are boundary faces or that have $N(j;h)$ as an endpoint element of another line.
2. Choose $h \in H_j$ of maximum connectivity. If h is not a boundary face, let $k = N(j;h)$.
3. If k has no other neighboring line endpoints or boundary faces of equal or higher connectivity, then connect the two lines to which j and k belong.

Both stages of the line creation algorithm result in a unique set of lines for a given flow condition, independent of seed element. A detailed proof of this statement is given in [13]. The sketch of the proof is as follows. First, following Stage I, a given pair of neighboring elements, j, k, are connected if and only if $C_{j,k}$ is one of the top two connectivities in j and in k. This observation can be verified by examining the MakePath algorithm. It implies uniqueness after Stage I because it provides a rule for connecting a given pair of neighboring elements regardless of the seed elements used for the algorithm.

Next, after Stage II, two neighboring endpoint elements, j and k, of distinct lines will be joined to form one line if and only if $C_{j,k}$ is maximum over the set of faces in H_j and H_k. This observation follows directly from the Stage II line connection algorithm. It implies uniqueness after Stage II because it dictates whether two line endpoints will be connected regardless of the order in which the endpoints are visited. Thus, the set of lines is unique after both stages of the line creation algorithm.

The lines formed for a 2D NACA 0012 test case are shown in Fig. 1. In the outer, convection-dominated flow, the lines follow the streamline direction. In the boundary layer and wake, the effects of high aspect ratio and diffusion couple elements in the direction normal to convection, resulting in lines normal to the streamlines.

3.3. *p*-Multigrid

The use of multigrid techniques is motivated by the observation that the smoothers developed in Section 3.1 are ineffective at eliminating low-frequency error modes on the fine grid. In standard h-multigrid, spatially coarser grids are used to correct the error on the fine grid. On the coarse grids, low-frequency error modes from the fine grid appear as high-frequency modes, and are therefore effectively eliminated by the smoothers. p-Multigrid is motivated by the same principle except that lower-order approximations serve as the coarse levels [21]. p-Multigrid fits naturally into the high-order DG, unstructured grid framework,

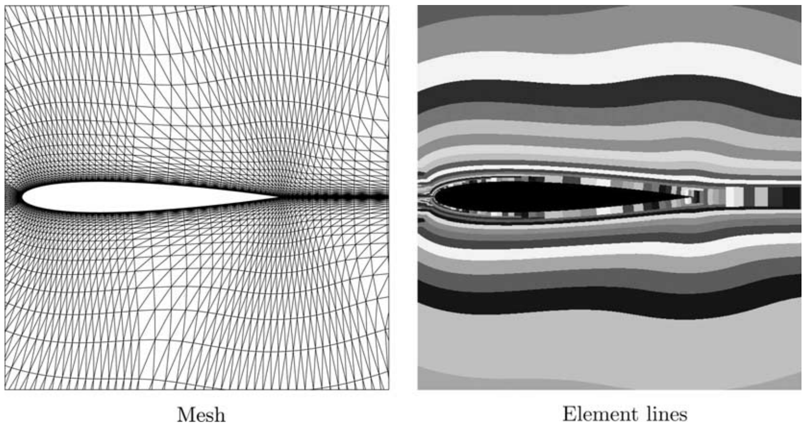


Fig. 1. Lines formed around NACA 0012 at $M = 0.5$, $Re = 5000$, $\alpha = 0°$.

since spatially coarser meshes are not used. Furthermore, the prolongation and restriction operators between orders are local to the elements.

To accommodate the nonlinearity in the system of equations, multigrid is implemented using the full approximation scheme (FAS), introduced by Brandt [22]. For a two-level FAS method, the exact solution on a coarse level is used to correct the solution on the fine level. Specifically, the procedure is as follows:

- Perform ν_1 smoothing iterations on the fine level: $\mathbf{v}^p \leftarrow \mathbf{v}^p - \omega\mathbf{P}^{-1}\mathbf{r}^p(\mathbf{v}^p)$.
- Restrict the state and residual to the coarse level: $\mathbf{v}^{p-1} \leftarrow \tilde{I}_p^{p-1}\mathbf{v}^p$, $\mathbf{f}^{p-1} \leftarrow I_p^{p-1}\mathbf{r}^p(v^p) + \mathbf{r}^{p-1}(\mathbf{v}^{p-1})$.
- Solve the coarse level problem: $\mathbf{r}^{p-1}(\mathbf{u}^{p-1}) = \mathbf{f}^{p-1}$.
- Prolongate the coarse level error and correct the fine level state: $\mathbf{v}^p \leftarrow \mathbf{v}^p + I_{p-1}^p(\mathbf{u}^{p-1} - \mathbf{v}^{p-1})$.
- Perform ν_2 smoothing iterations on the fine level: $\mathbf{v}^p \leftarrow \mathbf{v}^p - \omega\mathbf{P}^{-1}\mathbf{r}^p(\mathbf{v}^p)$.

When the fine level residual is zero, the coarse level correction is zero since $\mathbf{u}^{p-1} = \mathbf{v}^{p-1}$. In this work, the $p-1$ discretization is chosen as the coarse approximation. However, further efficiencies might be possible with greater reductions in the order. For example, Rønquist and Patera have used $p/2$ reductions. For the relatively low p values considered in the current work, reducing the order to $p-1$ is nearly equivalent to $p/2$, and, thus, $p-1$ was chosen for simplicity.

The state prolongation and residual restriction operators can be defined in a standard manner using the bases of the approximation spaces [23,24]. Specifically, assuming the approximation spaces are nested (i.e. $\mathscr{V}_h^{p-1} \subset \mathscr{V}_h^p$), a unique set of coefficients, α_{ij}^{p-1} exists relating the $p-1$ and p basis functions

$$\phi_i^{p-1} = \sum_j \alpha_{ij}^{p-1}\phi_j^p.$$

Thus, the state prolongation operator can be defined as $I_{p-1}^p = (\alpha^{p-1})^{\mathrm{T}}$. Similarly, the weighted residual for a $p-1$ basis function can be written as a linear combination of weighted residuals from the p basis functions

$$R(\phi_i^{p-1}, \mathbf{v}_h^p) = R\left(\sum_j \alpha_{ij}^{p-1}\phi_j^p, \mathbf{v}_h^p\right) = \sum_j \alpha_{ij}^{p-1} R(\phi_j^p, \mathbf{v}_h^p).$$

Thus, the residual restriction operator can be defined as, $I_p^{p-1} = \alpha^{p-1}$, i.e. the transpose of the prolongation operator. $\tilde{I}_p^{p-1}$ is the state restriction operator and does not have to be the same as the residual restriction, since it represents the restriction of a physical state as opposed to a weighted residual. In this work, an L_2 orthogonal projection was chosen, giving the following definition:

$$\tilde{I}_p^{p-1} = (\mathscr{M}^{p-1})^{-1}\mathscr{N}^{p-1}, \qquad \text{where } \mathscr{M}_{k,i}^{p-1} = \int_\Omega \phi_k^{p-1}\phi_i^{p-1}\,\mathrm{d}\mathbf{x}, \quad \mathscr{N}_{k,j}^{p-1} = \int_\Omega \phi_k^{p-1}\phi_j^p\,\mathrm{d}\mathbf{x}.$$

The operators are sparse globally, but take the form of possibly dense matrices locally on each element. With a Lagrange basis, the local operators are dense. However, with 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. Further details on the operator definitions can be found in [13].

To improve the efficiency of the multigrid method, the two-level correction scheme is extended to V-cycles and full multigrid (FMG). In a V-cycle, one or more levels are used in a recursive fashion to correct the fine level solution. FMG solves the coarse-level problems (though usually only in an approximate manner) prior to iterating on the fine-level problem, thereby potentially reducing the overall solution time as well as increasing the robustness by improving the starting solutions on the finer level approximations. To determine when to switch approximation orders in the FMG process, a residual-based approach is used. At the end of each V-cycle, the order p weighted residuals are compared to the order $p+1$ weighted residuals for the current p level approximation [13]. When the L_1 norm of the order p weighted residuals is smaller than η_{switch} times the order $p+1$ weighted residuals, the approximation level is raised to $p+1$. The evaluation of the $p+1$ weighted residuals adds only marginally to the solution cost because the calculation is done only once per multigrid V-cycle and only for the V-cycles that occur before the finest level is reached (i.e. only on the first few V-cycles). Unless otherwise noted, this residual-based switching approach is used with $\eta_{\text{switch}} = 0.5$ for all FMG results.

4. Fourier analysis

To determine appropriate values of the under-relaxation parameter, ω, and to quantify the performance of the smoothers and p-multigrid, Fourier analysis is performed for convection–diffusion in one and two dimensions with periodic boundary conditions. The convection–diffusion equations with constant velocity, $\mathbf{V} = (a, b)$, are given by

$$\begin{aligned} &au_x - vu_{xx} = f(x) \quad \text{on } [-1,1], \\ &Rau_x + bu_y - v(u_{xx} + u_{yy}) = f(x,y) \quad \text{on } [-1,1] \times [-1,1]. \end{aligned}$$

Full-upwinding is used for the inter-element inviscid flux, and the BR2 scheme is used for the viscous flux.

In one-dimension, only the element Jacobi method for pure convection ($v = 0$) is analyzed. In particular, the algorithm's asymptotic convergence rate is shown to be independent of order for $p \geqslant 0$. For 2D convection–diffusion, element and element line Jacobi, with and without p-multigrid, are analyzed. In two dimensions, the algorithms do exhibit p-dependence, although for element line Jacobi with p-multigrid, this dependence is weak for the moderate p considered.

4.1. Element jacobi for 1D convection

For this analysis, a Lagrange basis is used on equi-spaced nodes. However, the eigenvalues are unchanged for any non-degenerate basis of the same polynomial approximation since changing the basis results in a similarity transformation of the DG system of equations. In addition, although $a > 0$ is assumed, analogous results hold for general values of a.

Defining the error at iteration n as $\mathbf{e}^n \equiv \mathbf{v}^n - \mathbf{u}$, the preconditioned iterative scheme in (7) becomes, $\mathbf{e}^{n+1} = \mathbf{S}\mathbf{e}^n$ where $\mathbf{S} = \mathbf{I} - \omega\mathbf{P}^{-1}\mathbf{A}$ and $\mathbf{A}$ is the matrix arising from the DG discretization of the convection equation. The action of $\mathbf{A}$ on $\mathbf{e}^n$ can be expressed in stencil form. Specifically, for any element j,

$$\hat{\mathbf{A}}_U\hat{\mathbf{e}}^n_{j-1} + \hat{\mathbf{A}}_0\hat{\mathbf{e}}^n_j + \hat{\mathbf{A}}_D\hat{\mathbf{e}}^n_{j+1} = \hat{\mathbf{r}}^n_j, \tag{8}$$

where $\hat{\mathbf{e}}^n_j$ and $\hat{\mathbf{r}}^n_j$ are the length $p + 1$ error and residual vectors, respectively, for element j at iteration n, and $\hat{\mathbf{A}}_U$, $\hat{\mathbf{A}}_0$, and $\hat{\mathbf{A}}_D$ are $(p + 1) \times (p + 1)$ matrices obtained from the DG discretization. For example, for $p = 2$ using the equidistant Lagrange basis

$$\hat{\mathbf{A}}_U = -a\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad \hat{\mathbf{A}}_0 = \frac{1}{2}a\begin{pmatrix} 1 & \frac{4}{3} & -\frac{1}{3} \\ -\frac{4}{3} & 0 & \frac{4}{3} \\ \frac{1}{3} & -\frac{4}{3} & 1 \end{pmatrix}, \qquad \hat{\mathbf{A}}_D = 0.$$

Thus, the only non-zero contribution from the upwind element (i.e. $\hat{\mathbf{A}}_U$) is due to the influence of its downwind interface value, while the downwind element contribution (i.e. $\hat{\mathbf{A}}_D$) is identically zero. For the element Jacobi preconditioner, $\mathbf{P}$ is a block-diagonal matrix with diagonal blocks, $\hat{\mathbf{P}} = \hat{\mathbf{A}}_0$.

Due to the assumed periodicity, the eigenmodes of $\mathbf{S}$ will be of the form, $\hat{\mathbf{e}}_j = \exp(\mathrm{i}j\theta_x)\hat{\mathbf{e}}_0$, with $\theta_x \in \{m\pi h : \forall m \in -M/2 + 1, \ldots, M/2\}$, and $h = 2/M$, where M is the number of elements. For an error of this form, the evolution equation simplifies to

$$\hat{\mathbf{e}}_j^{n+1} = \hat{\mathbf{S}}(\theta_x)\hat{\mathbf{e}}_j^n, \quad \text{where } \hat{\mathbf{S}}(\theta_x) = \mathbf{I} - \omega\hat{\mathbf{A}}_0^{-1}(\hat{\mathbf{A}}_U \mathrm{e}^{-\mathrm{i}\theta_x} + \hat{\mathbf{A}}_0 + \hat{\mathbf{A}}_D \mathrm{e}^{\mathrm{i}\theta_x}).$$

For the equi-distant Lagrange basis with full-upwinding

$$\hat{\mathbf{S}}(\theta_x) = \mathbf{I} - \omega(\mathbf{I} - \hat{\mathbf{U}}\mathrm{e}^{-\mathrm{i}\theta_x}), \quad \text{where } \hat{\mathbf{U}} = -\hat{\mathbf{A}}_0^{-1}\hat{\mathbf{A}}_U = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}.$$

The eigenvalues of $\hat{\mathbf{S}}$ are $1 - \omega(1 - \exp \mathrm{i}\theta_x)$ (corresponding to the eigenvalue of the standard $p = 0$ upwind discretization) and p-repeated values of $1 - \omega$ (corresponding to the p higher-order degrees of freedom). Thus, for the element Jacobi preconditioner, the iterative method has the same eigenvalues for all $p \geqslant 0$ implying that the asymptotic convergence rate will be independent of p for $p \geqslant 0$.

4.2. *p-multigrid for 2D convection–diffusion*

The error evolution for a two-level multigrid scheme is given by

$$\mathbf{e}^{n+1} = \mathbf{M}^p\mathbf{e}^n, \quad \text{where } \mathbf{M}^p = (\mathbf{S}^p)^{\nu_2}\mathbf{K}^p(\mathbf{S}^p)^{\nu_1}.$$

$\mathbf{S}^p = \mathbf{I} - \omega\mathbf{P}^{-1}\mathbf{A}^p$ is the smoothing operator, with ω the under-relaxation factor, and $\mathbf{K}^p = \mathbf{I} - I_{p-1}^p(\mathbf{A}^{p-1})^{-1}I_p^{p-1}\mathbf{A}^p$ is the two-grid correction operator. ν_1 and ν_2 are the number of pre- and post-smoothing iterations; $\nu_1 = 1$ and $\nu_2 = 1$ were used for the analysis.

Fourier analysis of the two-level multigrid algorithm proceeds in a manner similar to the analysis in Section 4.1. In two dimensions, however, the eigenvalues depend on wave numbers in both directions, θ_x and θ_y. In general, the eigenvalues of the iterative algorithm depend on the following factors: smoother choice; fine and coarse orders; element Reynolds number, $Re \equiv a\Delta x/\nu$; element aspect ratio, $\mathrm{AR} = \Delta x/\Delta y$; flow angle, $\alpha = \tan^{-1}(b/a)$; ω; θ_x and θ_y. The details of two-level multigrid Fourier analysis can be found in various texts on multigrid, e.g. [23,25,26].

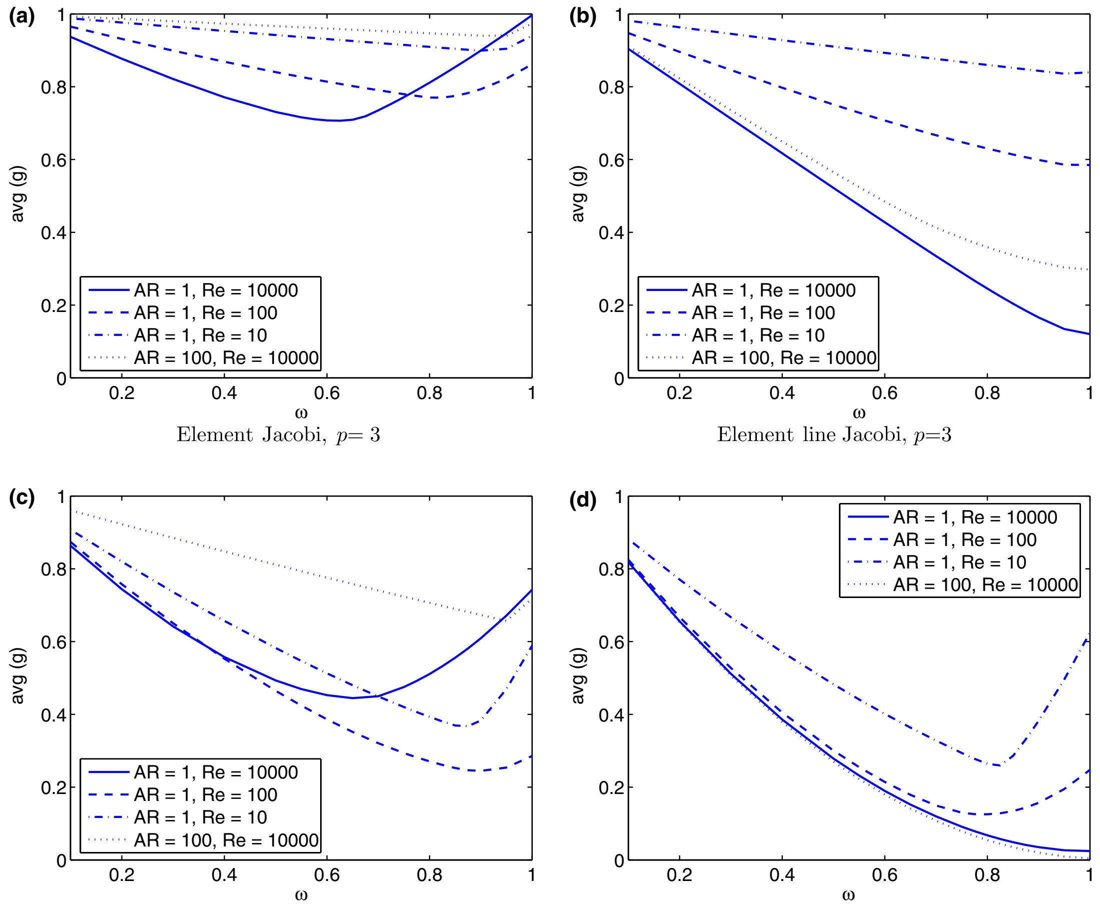
For element line smoothing, the lines are formed based on the coupling between elements for the $p = 0$ discretization of convection–diffusion. Since the problem considered has constant coefficients (i.e. constant velocity, viscosity, and grid), the line direction is constant over the grid. In particular, the direction of the lines is horizontal when

$$\Delta y\left(|a| + \frac{\nu}{\Delta x}\right) \geqslant \Delta x\left(|b| + \frac{\nu}{\Delta y}\right)$$

and vertical otherwise [14].

The eigenvalues of $\mathbf{M}^p$ are calculated as a function of ω for four specific cases: (a) nearly inviscid, $AR = 1, Re = 10{,}000$, (b) moderate Reynolds number, $AR = 1, Re = 100$, (c) low Reynolds number, $AR = 1, Re = 10$, and (d) boundary layer, $AR = 100, Re = 10{,}000$. The asymptotic convergence rate is bounded by the spectral radius of $\mathbf{M}^p$, $\rho(\mathbf{M}^p)$. As will be shown, for some flow conditions, isolated regions exist in wavenumber space (θ_x, θ_y), where $\rho(\mathbf{M}^p) \to 1$. This indicates that some grid dependence will exist for the algorithms (observed in the results of Section 5). For the purpose of choosing appropriate values of ω, the performance of the iterative algorithms is quantified using the *average* eigenvalue (i.e. amplification factor) over all the modes.

The average amplification factors for $p = 3$ are shown in Fig. 2. A comprehensive study of the parameter space $0 \leqslant p \leqslant 3$, $1 \leqslant \mathrm{AR} \leqslant 500$, $0 \leqslant Re \leqslant 10{,}000$, $1° \leqslant \alpha \leqslant 25°$, and $0 < \omega < 1$ revealed that all of the algorithms are stable in this range. In terms of choosing the under-relaxation factor to optimize the average amplification, a value of $\omega \approx 0.8$ is reasonable for all algorithms. In practice, although this value may not be optimal for all conditions, it has been shown to perform well for the majority of



(a) Element Jacobi, $p = 3$

(b) Element line Jacobi, $p=3$

(c) $p = 3/p = 2$ Multigrid with element Jacobi smoothing

(d) $p = 3/p = 2$ Multigrid with element line Jacobi smoothing

Fig. 2. Average amplification factor of $\mathbf{M}^p$ versus ω for various AR and Re, at $\alpha = 1°$.

the cases tested. The average amplification factors for $p = 1$ to $p = 3$ are tabulated in Tables 1–4 for $\omega = 0.8$. The results show clearly that the element Jacobi algorithm is the weakest algorithm while the p-multigrid algorithm with element line Jacobi smoothing is the strongest. In particular, the line smoother is significantly better than the element smoother for the boundary layer conditions ($AR = 100$, $Re = 10{,}000$). As shown in the tables, the performance of p-multigrid with element line smoothing is nearly independent of p in this range, and even the other algorithms generally have a weak dependence on p.

The profiles of the $p = 3/p = 2$ multigrid amplification factors for the choice of $\omega = 0.8$ are shown in Fig. 3. In these profiles, the multigrid amplification factor, g, of a mode refers to the maximum eigenvalue for that mode (as each mode is associated with several eigenvectors for $p > 0$). While in all cases g is low over most of the mode space, there are isolated regions where g tends to 1. In particular, for the $AR = 1$, $Re = 10{,}000$ case, modes that are low frequency in the streamwise direction ($\theta_x \approx 0$) are poorly damped. This effect suggests an h, or grid resolution, dependence which is observed in Section 5.

Table 1
Element Jacobi average amplification factors for $\omega = 0.8$

Order	$AR = 1$ $Re = 10,000$	$AR = 1$ $Re = 100$	$AR = 1$ $Re = 10$	$AR = 100$ $Re = 10,000$
$p = 3$	8.11E−1	7.70E−1	9.10E−1	9.47E−1
$p = 2$	8.09E−1	7.16E−1	8.52E−1	9.08E−1
$p = 1$	8.05E−1	7.32E−1	7.23E−1	8.09E−1

Table 2
Element-line Jacobi average amplification factors for $\omega = 0.8$

Order	$AR = 1$ $Re = 10,000$	$AR = 1$ $Re = 100$	$AR = 1$ $Re = 10$	$AR = 100$ $Re = 10,000$
$p = 3$	2.45E−1	6.30E−1	8.59E−1	3.59E−1
$p = 2$	2.32E−1	4.34E−1	7.61E−1	3.60E−1
$p = 1$	2.26E−1	2.74E−1	5.27E−1	3.66E−1

Table 3
p-Multigrid with element smoothing: average amplification factors for $\omega = 0.8$

Fine/coarse orders	$AR = 1$ $Re = 10,000$	$AR = 1$ $Re = 100$	$AR = 1$ $Re = 10$	$AR = 100$ $Re = 10,000$
$p = 3/p = 2$	5.11E−1	2.71E−1	3.93E−1	7.07E−1
$p = 2/p = 1$	5.58E−1	2.53E−1	3.51E−1	5.98E−1
$p = 1/p = 0$	6.05E−1	3.94E−1	3.62E−1	4.96E−1

Table 4
p-Multigrid with element line smoothing: average amplification factors for $\omega = 0.8$

Fine/coarse orders	$AR = 1$ $Re = 10,000$	$AR = 1$ $Re = 100$	$AR = 1$ $Re = 10$	$AR = 100$ $Re = 10,000$
$p = 3/p = 2$	6.78E−2	1.26E−1	2.64E−1	5.47E−2
$p = 2/p = 1$	5.85E−2	9.45E−2	2.22E−1	5.79E−2
$p = 1/p = 0$	4.97E−2	7.17E−2	2.20E−1	1.21E−1

5. Results

This section presents accuracy and solver performance results for an inviscid duct flow and a viscous airfoil flow. These results are representative of the algorithm's performance on a wider variety of smooth flow test cases, which are documented in the theses of Fidkowski [13] and Oliver [14] for a preliminary implementation of the current algorithm. For timing, full multigrid was used with a hierarchical basis, 8 pre- and post-smoothing sweeps for $p > 0$, and 150 sweeps on the coarsest level ($p = 0$). The large number of $p = 0$ smoothing sweeps was meant to ensure adequate solution of the coarse-level problem. An exact solve or h-multigrid at $p = 0$ may be more efficient alternatives, although these have not yet been investigated. All cases were started from a converged $p = 0$ solution and were run on an Intel Pentium 4 3.0 GHz system with 896 MB RAM.

5.1. Inviscid flow in a 2D duct

A representative, smooth inviscid case is that of duct flow over a Gaussian bump perturbation. The domain and an intermediate mesh are depicted in Fig. 4. The bump geometry was represented using piecewise cubic polynomials for all solution approximation orders. Wall boundary conditions were enforced on the top and bottom of the channel. At the outflow, the static pressure was set, and at the inflow, the total

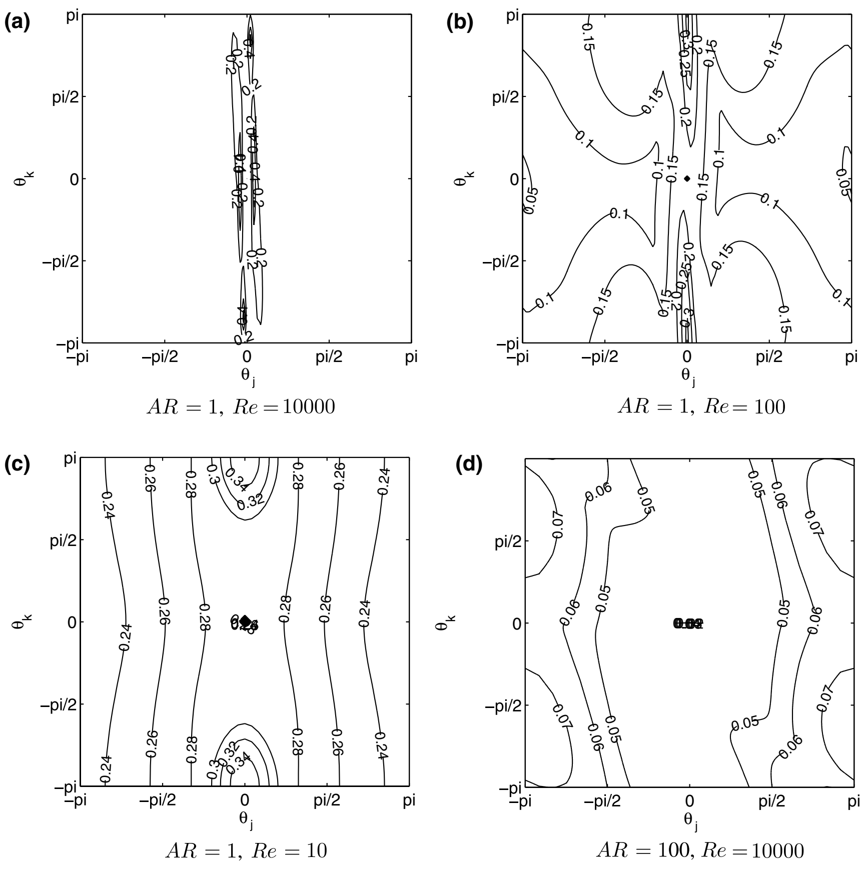


Fig. 3. Amplification factor contours for p-multigrid ($p = 3/p = 2$) with element line Jacobi for various AR and Re at $\omega = 0.8$. Shown as contours over (θ_x,θ_y), the modes in the x and y directions.

temperature, total pressure, and flow angle (0°) were prescribed, resulting in a free-stream Mach number of $M = 0.5$. For this mesh and flow, the element lines formed by the solver spanned the horizontal length of the bump domain.

Four meshes (324, 1296, 5184, and 20,736 elements) were used in a convergence study. The output of interest was the L_2 norm of the entropy error, $\|e_S\|_{L_2}$, where $e_S = S - S_{fs}$ and S_{fs} is the free-stream entropy. The results in Fig. 5(a) show that optimal error convergence, $\|e_S\|_{L_2} \sim h^{p+1}$, is approximately attained. Fig. 5(b) shows the accuracy versus CPU time for each run. Timings for $p = 0$ were not obtained because the converged $p = 0$ solutions served as the starting points for FMG at higher orders. A solution was deemed converged in post-processing at the iteration when the output error was first within one percent of the zero-residual error value. The advantage of higher-order for obtaining accurate solutions is clearly evident.

To study the asymptotic behavior of residual and output errors, an additional FMG run was performed in which the residual was fully converged on each level (as opposed to the residual-based switching normally used). The resulting residual history is shown in Fig. 6(a). As shown, the p-multigrid convergence rate does not degrade, and even improves slightly, with increasing order, at least up to order 3. In addition, the accompanying step-like error plot in Fig. 6(b) illustrates that truncation-level error is attained with only a few p-multigrid iterations after transfer to a finer level. Thus, in practice, it is not necessary to converge to machine zero residual to obtain outputs of interest to truncation-level accuracy [22].

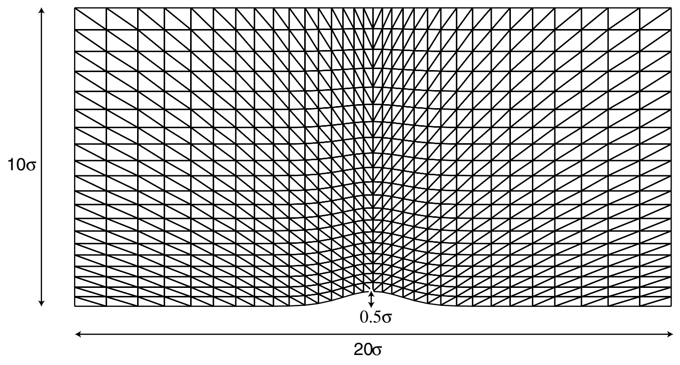


Fig. 4. 1296 element mesh for flow over a Gaussian bump.

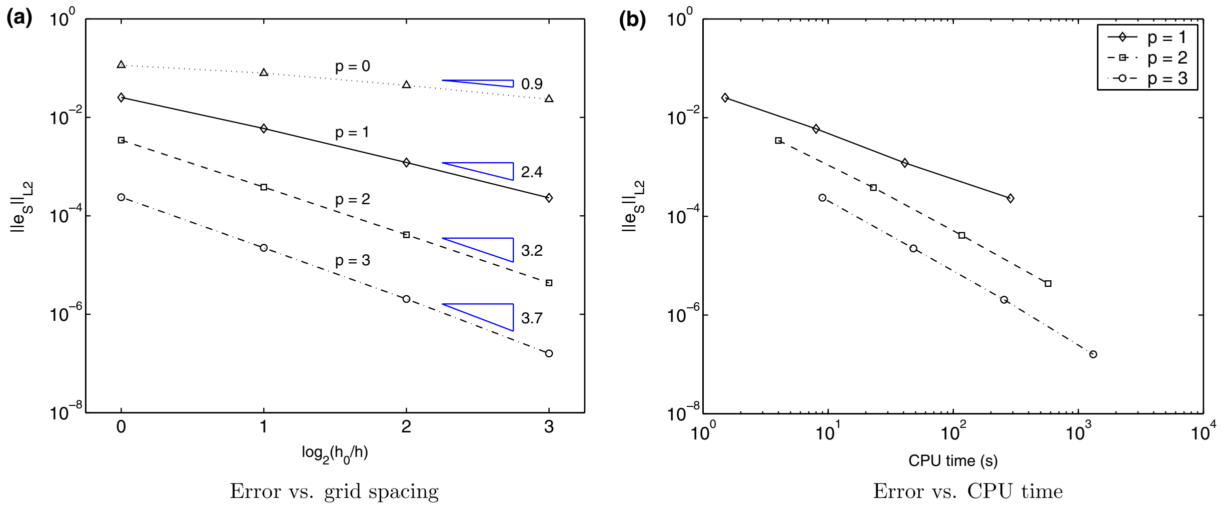


Fig. 5. Entropy error results for $0 \leqslant p \leqslant 3$ for flow in a 2D duct.

While the performance of p-multigrid with element line Jacobi smoothing is generally independent of p (for the orders tested), h dependence has been observed. Fig. 7 shows the residual convergence for $p = 1$ and $p = 3$ on the first three meshes. In both cases, the number of iterations required to converge the residual increases with the number of elements. In particular, for $p = 1$, on the coarsest mesh, the residual converges to 10^{-10} in approximately 13 iterations, while on the fine mesh, 19 iterations are required. For these same cases, the element line smoother was run without p-multigrid, resulting in the convergence histories in Fig. 8. As shown, the smoother alone exhibits much stronger h-dependence than when combined with p-multigrid, although the smoother convergence remains p-independent (for the orders tested).

Fig. 9 compares four different solution algorithms: element Jacobi, element line Jacobi, p-multigrid with element Jacobi smoothing, and p-multigrid with element line Jacobi smoothing in term of CPU time. As shown, incorporation of p-multigrid significantly improves the solver performance. In all cases, the combination of p-multigrid with element line smoothing is the best solver, and as p increases, it becomes relatively more efficient.

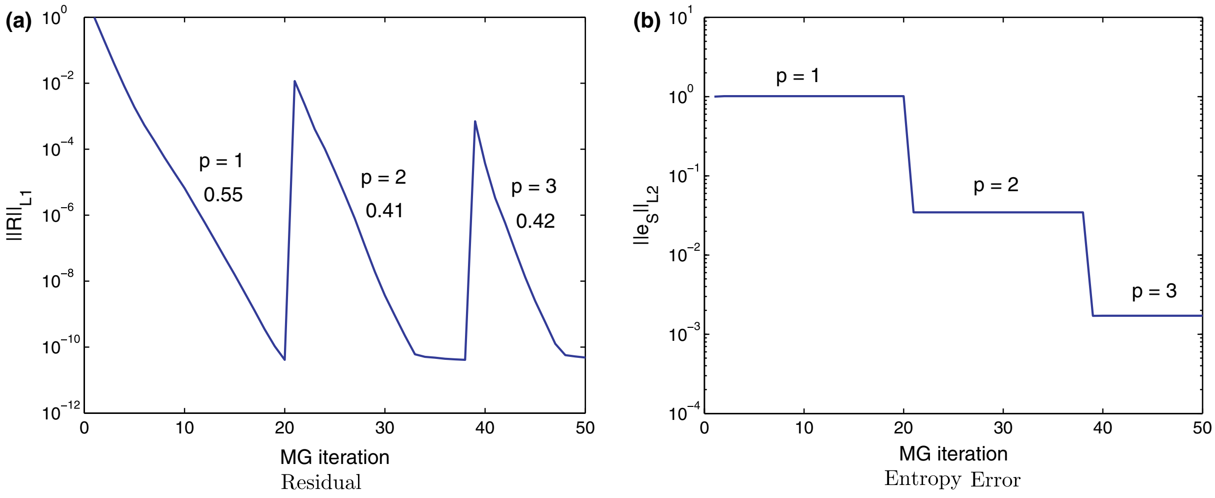


Fig. 6. 2D duct (1296 elements): FMG history with full convergence on each level.

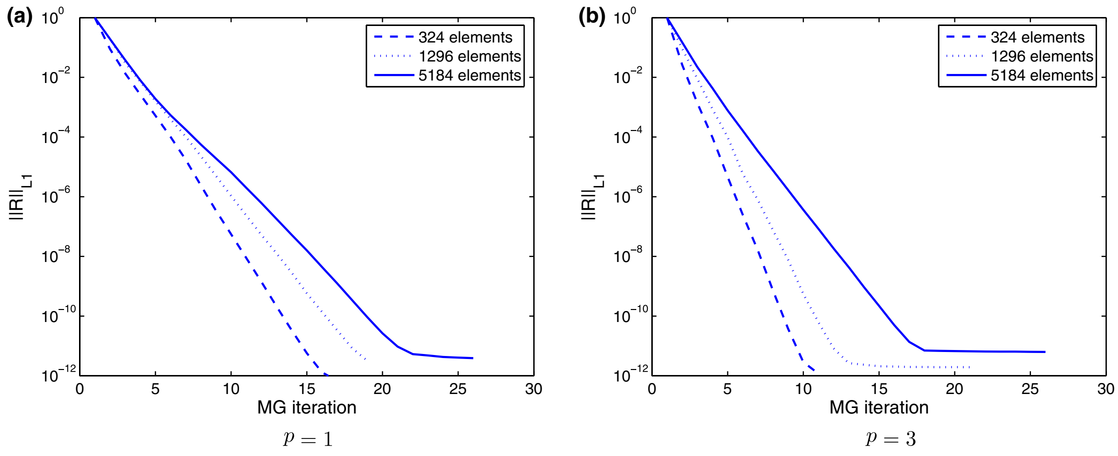


Fig. 7. p-Multigrid residual convergence versus grid size, for inviscid duct flow at $M = 0.5$.

5.2. Viscous flow over a NACA 0012 airfoil

A representative viscous case is that of $M = 0.5$, $Re = 5000$, laminar flow over a NACA 0012 airfoil at $\alpha = 0°$. A set of four structured grids was used to generate the data. The meshes were created by modifying a baseline grid provided by Swanson [27] and coarsening three times, resulting in four nested meshes containing 672, 2688, 10,752, and 43,008 elements. The coarsest mesh is shown in Fig. 10. On the airfoil surface, a no-slip, adiabatic boundary condition was imposed. As mentioned earlier, the boundary was represented with piecewise cubic polynomials for all orders of solution approximation. The boundary nodes were placed using a slightly modified version of the analytic definition of the NACA 0012:

$$y = \pm 0.6(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1036x^4). \tag{9}$$

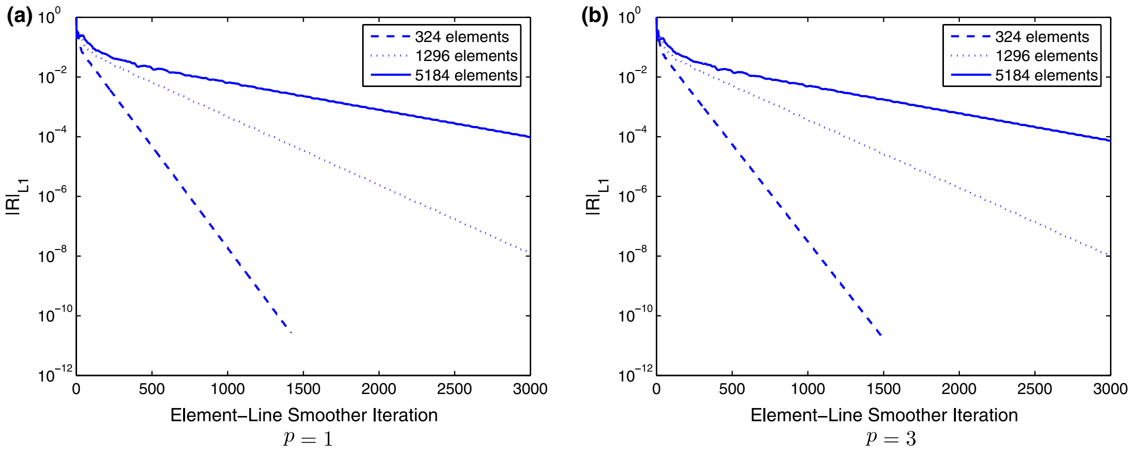


Fig. 8. Element line smoother residual convergence versus grid size for inviscid duct flow at $M = 0.5$.

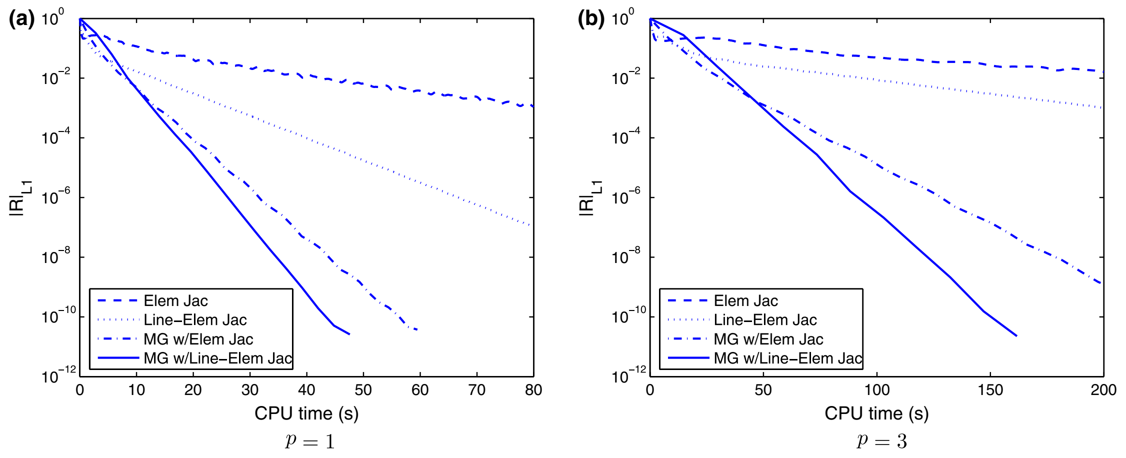


Fig. 9. Solver comparison on 1296 element mesh for inviscid bump at $M = 0.5$.

In (9), the x^4 coefficient of the traditional definition of the NACA 0012 has been modified such that the trailing edge ($x = 1$) has zero thickness.

The drag error for each grid and interpolation order is shown in Fig. 11. The drag error is defined as the difference between the drag for a given case and the drag computed from the $p = 3$ solution on the 43,008 element mesh. Fig. 11(a) shows that the drag error converges at approximately $O(h^{2p})$ for all $p \geqslant 1$ cases not including the second $p = 3$ grid refinement. While it is not clear why the $p = 3$ rate degrades for the second refinement, it is possible that superconvergent results should be expected for the drag error in this case [28].

For comparison, results computed by FUN2D, which uses a node-centered finite volume algorithm, on the same meshes are shown in Fig. 11(b) [29,30]. The convergence of the FUN2D and $p = 1$ methods in terms of DOF is very similar. However, the $p = 2$ and $p = 3$ results are significantly better. For example, the drag for the 2688 element $p = 2$ solution (6.5×10^4 DOF) is within 0.1 drag counts of the finest grid, $p = 3$ result, while the drag for the finest grid, $p = 1$ solution (43,008 elements, 5.2×10^5 DOF) has an error of approximately 0.3 counts. Thus, at error levels of practical interest, higher-order solutions provide significantly better accuracy per DOF than $p = 1$ solutions on more refined meshes. However, this DOF comparison does

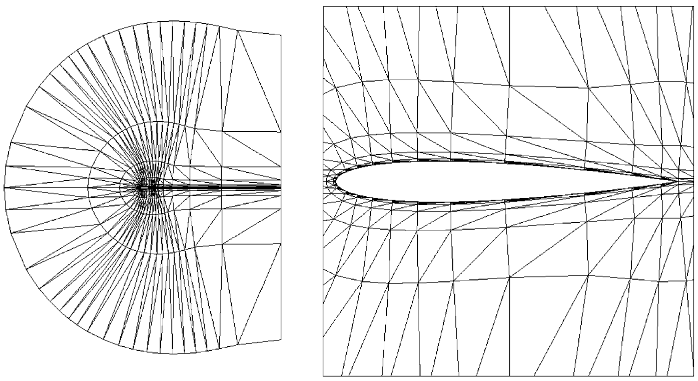


Fig. 10. Coarse NACA 0012 mesh, 672 elements.

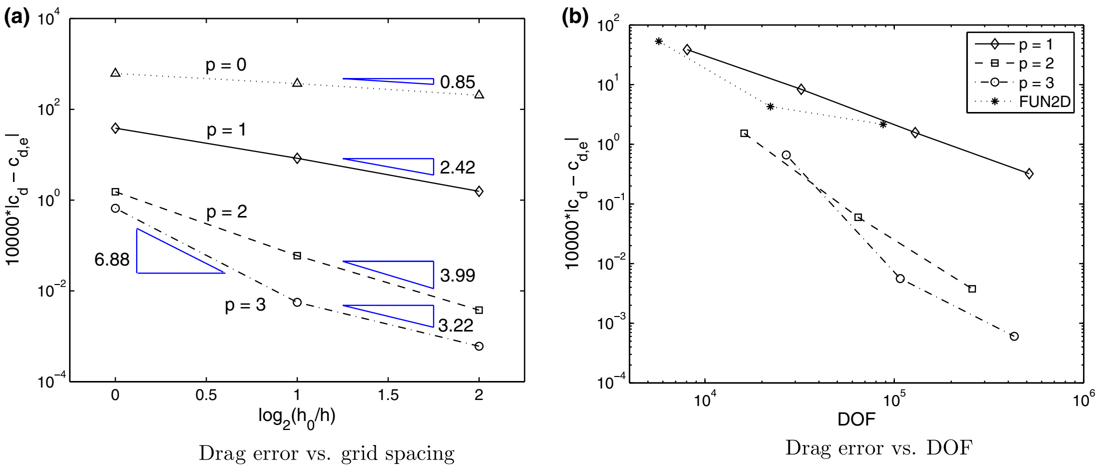


Fig. 11. Drag error results for $0 \leqslant p \leqslant 3$ for NACA 0012 at $M = 0.5$, $Re = 5000$, $\alpha = 0°$.

not imply that the high-order solver outperforms FUN2D in terms of CPU time. While a CPU time comparison is presented in Fig. 12, the FUN2D results from Fig. 11(b) are not included because no attempt was made to optimize the performance of FUN2D or the current solver for the problem considered.

Fig. 12 shows the drag error versus CPU time. The values plotted were obtained using a drag-based stopping criterion. Specifically, the calculation was stopped when the change in the drag per iteration was less that 0.01 drag counts, $10,000|c_d^{n+1} - c_d^n| < 0.01$, for three consecutive iterations. This stopping criterion was used to minimize the computational time required to compute the drag given the knowledge that the drag converges to engineering tolerance faster than the residual converges to machine zero. Clearly, the higher-order solutions produce smaller error in less time than the $p = 1$ solution. For example, to achieve a drag error of approximately 0.1 counts, the $p = 1$ solution would take approximately 100 times longer than either the $p = 2$ or $p = 3$ solutions.

As in the inviscid case, while the multigrid performance was seen to be p-independent for the orders tested, h dependence was observed. Fig. 13 shows this effect in the residual convergence for $p = 1$ and $p = 3$ on three meshes. Compared to the inviscid case, the h dependence is stronger. For $p = 1$, on the 672 element mesh, the residual converges to 10^{-10} in approximately 15 multigrid iterations, while on the fine mesh, 36 iterations are required.

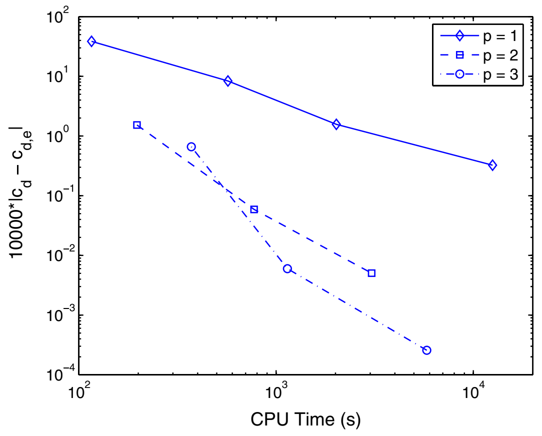


Fig. 12. Drag error versus CPU time for NACA 0012 at $M = 0.5$, $Re = 5000$, $\alpha = 0°$.

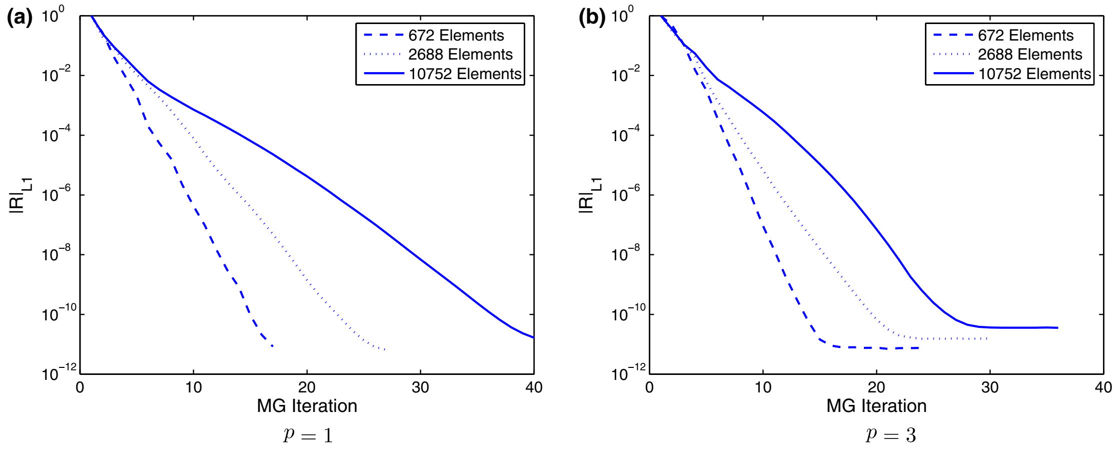


Fig. 13. p-Multigrid residual convergence versus grid size for NACA 0012 at $M = 0.5$, $Re = 5000$, $\alpha = 0°$.

Fig. 14 shows the performance of the element line Jacobi smoother with varying h and p. Note that these results were obtained using an under-relaxation factor of $\omega = 0.6$ because a weak unsteadiness was observed when using only the element line smoother with $\omega = 0.8$. This unsteadiness exhibited itself as a Kelvin–Helmholtz-like instability in the wake downstream of the airfoil which was damped when ω was lowered.

As in the inviscid case, the convergence of the residual using only the element line smoother is strongly h dependent. However, unlike the inviscid results, a slight p dependence is also present. For example, on the 2688 element mesh, for $p = 1$, the residual converges to 10^{-10} in approximately 3/4 the number of element line Jacobi iterations required to converge the $p = 3$ residual to the same level. Thus, in addition to reducing the h dependence of the element line smoother, p-multigrid also decreases the p dependence.

Fig. 15 shows the performance of the four different solution algorithms compared in the inviscid case. Again, the incorporation of the element line Jacobi smoother significantly improves performance, and p-multigrid with element line Jacobi smoothing is the most efficient solver.

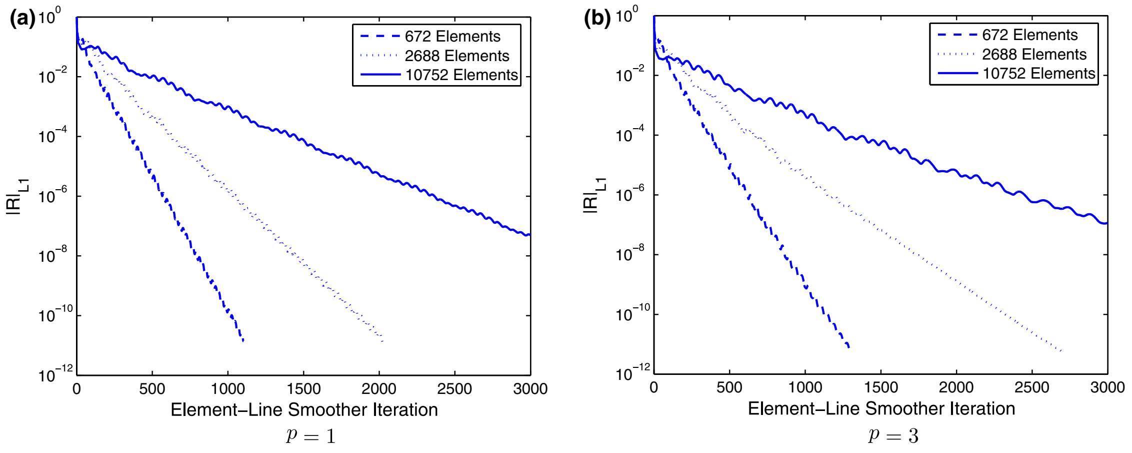


Fig. 14. Element line smoother residual convergence versus grid size for NACA 0012 at $M = 0.5$, $Re = 5000$, $\alpha = 0°$.

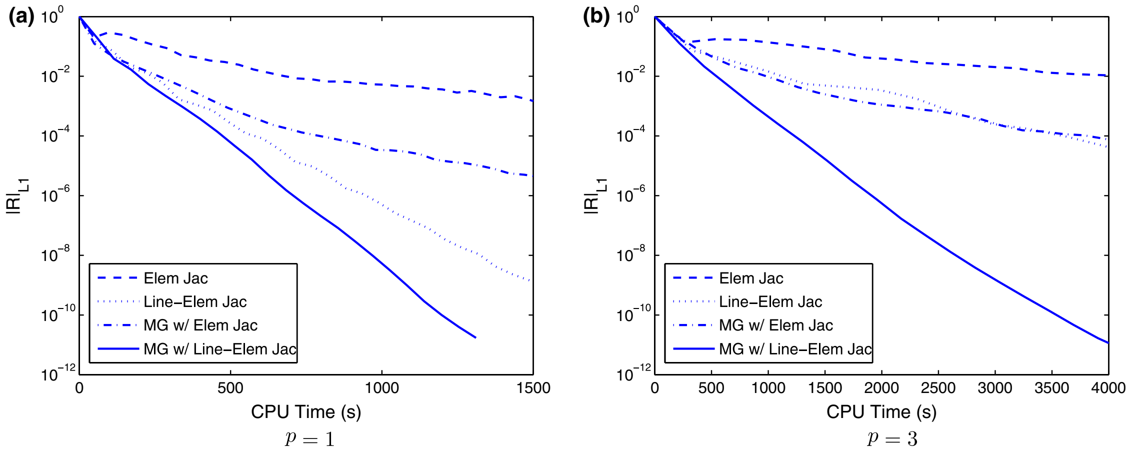


Fig. 15. Solver comparison on 2688 element mesh for NACA 0012 at $M = 0.5$, $Re = 5000$, $\alpha = 0°$.

6. Conclusions

This paper presented a solution algorithm for the compressible Navier–Stokes equations discretized using a discontinuous Galerkin method. The algorithm combines p-multigrid with an element line smoother. Fourier analysis is done to quantify the performance of the proposed algorithm and to compare it with simpler variations (such as element Jacobi with/without p-multigrid or element line Jacobi without p- multigrid). Simulations of an inviscid bump flow and a high Reynolds number airfoil flow are also performed. The analysis and results show that for the combination of element line smoothing and p-multigrid, the residual converges at nearly p-independent rates but has some asymptotic h dependence. Furthermore, the efficiency in terms of CPU time of the proposed algorithm compared to other solution approaches improves as the order of accuracy is increased. Finally, the simulations show that high-order solutions obtained with this algorithm can provide significant reductions in the computational time required to achieve desired accuracy levels.

Acknowledgments

The authors thank the members of the Project X team for numerous contributions during the course of this work, specifically: Mike Brasher, Bob Haimes, Paul Nicholson, Jaime Peraire, and Matthieu Serrano. The assistance of David Venditti in performing the FUN2D simulations is gratefully acknowledged. K. Fidkowski's work was supported by the Department of Energy Computational Science Graduate Fellowship, and T. Oliver's work was supported by the Department of Defense National Defense Science and Engineering Graduate Fellowship. A portion of this work was supported by NASA contract NAG-1-02037.

Appendix A. Representation on curved elements

In Section 2, the comment was made that the solution basis used for curved elements is polynomial in the physical space. The importance of this fact is evident in the evaluation of a typical integral over a curved element:

$$\int_{\kappa} f(\phi)\mathrm{d}\mathbf{x} = \sum_{g} w_g f(\phi)|\mathbf{J}|\mathrm{d}\xi, \tag{A.1}$$

where numerical quadrature is used in the reference element corresponding to the curved element. In (A.1), w_g is the weight corresponding to quadrature point g; ϕ represents the basis functions; $\mathbf{x}$ represents the coordinates in the physical space, and ξ represents the coordinates in the reference space, as shown in Fig. 16. **A** is the nonlinear mapping from the reference element to the curved element, and $\mathbf{J} = \partial\mathbf{x}/\partial\xi$ is the Jacobian of the mapping. If the basis functions, ϕ, are defined in the standard fashion on the reference element, (A.1) becomes

$$\int_{\kappa} f(\phi)\mathrm{d}\mathbf{x} = \sum_{g} w_g f(\phi(\xi_g))|\mathbf{J}|\mathrm{d}\xi, \tag{A.2}$$

where ξ_g are the quadrature point coordinates. However, with this definition, the basis functions transformed to the physical space ($\mathbf{x}$) do not necessarily form a complete basis of the desired order, since the mapping **A** is nonlinear for curved elements. An alternative used in this work is to define the basis in a reference triangle corresponding to the shadow of the curved element, where the shadow is defined as the triangle formed by connecting the curved element vertices with straight lines (see Fig. 16, dashed lines). Since the resulting reference-to-global shadow transformation, **B**, is linear, the basis functions thus defined form a complete basis of the desired degree in the physical space. With this definition, (A.1) becomes

$$\int_{\kappa} f(\phi)\mathrm{d}\mathbf{x} = \sum_{g} w_g f(\phi(\mathbf{B}^{-1}\mathbf{A}\xi_g))|\mathbf{J}|\mathrm{d}\xi. \tag{A.3}$$

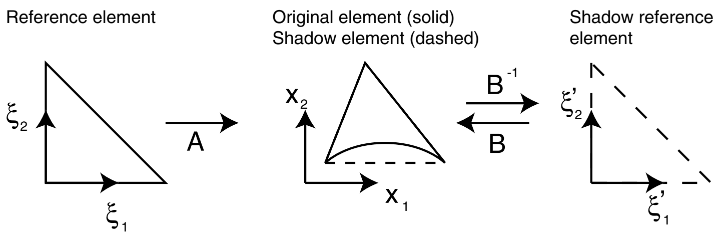


Fig. 16. Reference and shadow triangles corresponding to a curved element.

That is, the quadrature points in the original reference element are transformed to the shadow reference element via the mapping $\mathbf{B}^{-1}\mathbf{A}$, and the basis functions are evaluated using these transformed points. Note that in the case of a linear element, $\mathbf{B} = \mathbf{A}$ and $\mathbf{B}^{-1}\mathbf{A}$ is the identity, so that the standard representation, (A.2) is recovered. Finally, if the element in the physical space is severely curved, such that the vertices are colinear, the shadow triangle definition has to be modified (for example by using a bounding box of the curved element).

Appendix B. Boundary conditions

Boundary conditions are enforced weakly via the domain boundary integrals appearing in (2). To evaluate these integrals, the two terms $\mathbf{v}_h^b$ and $(\mathscr{A}_v\nabla\mathbf{v}_h)^b$ must be defined.

B.1. Full-state condition

In cases where the entire state vector at the boundary, $\mathbf{u}_h^b$, is known, the inviscid flux is computed using the Riemann solver exactly as if the face were an interior face:

$$\mathscr{H}_i^b(\mathbf{v}_h^+, \mathbf{v}_h^b, \hat{\mathbf{n}}) = \mathscr{H}_i(\mathbf{v}_h^+, \mathbf{u}_h^b, \hat{\mathbf{n}}).$$

No conditions are set on the viscous flux. Thus, $(\mathscr{A}_v\nabla\mathbf{v}_h)^b$ is set by interpolating $(\mathscr{A}_v\nabla\mathbf{v}_h)^+$ to the boundary. $\boldsymbol{\delta}_f^b$ is computed using $\mathbf{v}_h^b$ via (5).

B.2. Inflow/outflow conditions

At an inflow/outflow boundary, the boundary state, $\mathbf{v}_h^b$, is defined using the outgoing Riemann invariants and given boundary data. Table 5 details the inflow/outflow conditions used in this work. Note that, while the Euler equations are well-posed with just the boundary conditions listed in Table 5, the Navier–Stokes equations are not. Conditions – numeric and/or physical – are needed to set $(\mathscr{A}_v\nabla\mathbf{v}_h)^b$ and $\boldsymbol{\delta}_f^b$. For this work, $(\mathscr{A}_v\nabla\mathbf{v}_h)^b$ is set by extrapolating from the interior of the domain, and $\boldsymbol{\delta}_f^b$ is computed via (5).

B.3. Adiabatic, no slip wall conditions

For a no slip, adiabatic wall, the velocity components and normal temperature gradient at the wall are prescribed: $u^b = 0$, $v^b = 0$, $\partial \mathrm{T}/\partial \mathrm{n}|_b = 0$. To compute the boundary state, the static pressure, p, and stagnation enthalpy per unit mass, H, are set using interior data:

$$p = (\gamma - 1)\rho E^+,\ H = \frac{\rho E^+ + p}{\rho^+} \quad \Rightarrow \quad \mathbf{v}_h^b = (\rho^+, 0, 0, \rho E^+)^{\mathrm{T}}.$$

Table 5
Types of inflow/outflow boundary conditions

Condition	Number of BCs	Value specified	Outgoing invariant
Subsonic inflow	3	T_T, p_T, α	J_+
Supersonic inflow	4	$\rho, \rho u, \rho v, \rho E$	None
Subsonic outflow	1	p	J_+, v, s
Supersonic outflow	0	None	J_+, J_-, v, s

The adiabatic condition, combined with the no slip condition, requires the viscous flux associated with the energy equation to be zero. The interior viscous flux is used for the other components. Furthermore, since the energy equation viscous flux is specified, the corresponding component of the auxiliary variable, $\boldsymbol{\delta}_f^b$, is set to zero. All other auxiliary variable components are computed as stated in (5).

References

[1] F. Bassi, S. Rebay, High-order accurate discountinuous finite element solution of the 2-D Euler equations, J. Comput. Phys. 138 (1997) 251–285.

[2] F. Bassi, S. Rebay, A high-order accurate discountinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, J. Comput. Phys. 131 (1997) 267–279.

[3] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier–Stokes equations, in: K. Cockburn, Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Springer, Berlin, 2000, pp. 197–208.

[4] B. Cockburn, C.W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (3) (2001) 173–261.

[5] A. Jameson, Solution of the Euler equations for two-dimensional transonic flow by a multigrid method, Appl. Math. Comput. 13 (1983) 327–356.

[6] D.J. Mavriplis, Multigrid solution of the 2-D Euler equations on unstructured triangular meshes, AIAA J. 26 (1988) 824–831.

[7] S.R. Allmaras, Analysis of semi-implicit preconditioners for multigrid solution of the 2-D compressible Navier–Stokes equations, AIAA Paper Number 95–1651-CP, 1995.

[8] N.A. Pierce, M.B. Giles, Preconditioned multigrid methods for compressible flow calculations on stretched meshes, J. Comput. Phys. 136 (1997) 425–445.

[9] D.J. Mavriplis, Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes, J. Comput. Phys. 145 (1998) 141–165.

[10] P. Hemker, W. Hoffmann, M. van Raalte, Two-level fourier analysis of a multigrid approach for discontinuous Galerkin discretisation, SIAM J. Sci. Comput. 25 (2004) 1018–1041.

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

[12] F. Bassi, S. Rebay, Numerical solution of the Euler equations with a multiorder discontinuous finite element method, in: Second International Conference on Computational Fluid Dynamics, Sydney, Australia, 2002.

[13] K. Fidkowski, A high-order discontinuous Galerkin multigrid solver for aerodynamic applications, Master's Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2004.

[14] T.A. Oliver, Multigrid solution for high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, Master's Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, August 2004.

[15] T.A. Oliver, K.J. Fidkowski, D.L. Darmofal, Multigrid solution for high-order discontinuous Galerkin discretization of the compressible Navier–Stokes equations, in: Proceedings of Third International Conference on Computational Fluid Dynamics, Toronto, Canada, 2004 (in press).

[16] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.

[17] D. Arnold, F. Brezzi, B. Cockburn, L. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2002) 1749–1779.

[18] F. Brezzi, G. Manzini, D. Marini, P. Pietra, A. Russo, Discontinuous Galerkin approximations for elliptic problems, Numer. Methods Partial Differential Eqns. 16 (2000) 365–378.

[19] P. Solín, K. Segeth, I.D. Zel, Higher-Order Finite Element Methods, Chapman & Hall, London, 2003.

[20] T. Okusanya, D. Darmofal, J. Peraire, Algebraic multigrid for stabilized finite element discretizations of the Navier–Stokes equations, Comput. Mech. Appl. Math. Eng. 193 (2004) 3667–3686.

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

[22] A. Brandt, Guide to Multigrid Development, Springer, Berlin, 1982.

[23] W. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, second ed., SIAM, Philadelphia, PA, 2000.

[24] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, Springer, Berlin, 1994.

[25] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, London, 2000.

[26] P. Wesseling, An Introduction to Multigrid Methods, Wiley, New York, 1992.

[27] R. Radespiel, R. Swanson, An investigation of cell centered and cell vertex multigrid schemes for the Navier–Stokes equations, AIAA Paper Number 89-0453, 1989.

[28] M.B. Giles, E. Süli, Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality, Acta Numer. 11 (2002) 145–236.
[29] W.K. Anderson, D.L. Bonhaus, An implicit upwind algorithm for computing turbulent flows on unstructured grids, Comput. Fluids 23 (1994) 1–21.
[30] E. Nielsen, FUN2D/3D Fully Unstructured Navier–Stokes user manual. NASA Langley Research Center, Computational Modeling and Simulation Branch, Virginia. Available from: <http://fun3d.larc.nasa.gov>.