45th AIAA Aerospace Sciences Meeting and Exhibit
8 - 11 January 2007, Reno, Nevada

AIAA 2007-512

45th AIAA Aerospace Sciences Meeting and Exhibit, January 8–11, 2007, Reno, NV

A Parallel *hp*-Multigrid Solver for Three-Dimensional Discontinuous Galerkin Discretizations of the Euler Equations

Cristian R. Nastase * and Dimitri J. Mavriplis †

Department of Mechanical Engineering, University of Wyoming, Laramie, Wyoming 82071-3295

A combined h and p multigrid solution strategy is developed for high-order Discontinuous Galerkin discretizations of the three-dimensional Euler equations. This solver is used to compute inviscid compressible flow over realistic three-dimensional aerodynamic configurations, and the performance of the solver in terms of convergence efficiency and parallel scalability is investigated. The *hp* multigrid solver is found to deliver nearly optimal convergence rates, which are insensitive to the discretization order p, and to the mesh resolution h. The solver is also shown to scale well on massively parallel computer architectures, demonstrating good scalability up to 2008 processors of the NASA Columbia Supercomputer.

I. Introduction

Interest in the use of higher-order discretizations (higher than second order) for industrial computational fluid dynamic problems, including aerodynamics, has become more widespread over the last several years. This is partly due to the difficulties encountered with traditional second-order accurate methods at delivering consistently grid converged results and in quantifying the spatial discretization errors.[1,2] The asymptotic properties of higher-order methods makes them suitable for problems where high spatial accuracy is required, since for smooth solutions, spatial error is reduced ever more rapidly with increasing grid resolution at higher "p" orders of accuracy.

While much research has been devoted to investigating and developing higher-order discretizations for a variety of applications, these methods are still seldom used in applications such as aerodynamics, due to the relatively large computational overheads associated with current solvers for these discretizations. The development of a competitive high-order simulation capability rests on the capability of efficiently solving the discrete equations arising from these sophisticated discretizations, as well as the ability of these methods to scale well on current-day massively parallel computer architectures. Discretizations such as spectral element methods[3] and discontinuous Galerkin methods[4] have long been known to have the potential to scale well on massively parallel computer architectures, due to their use of a compact stencil, and the presence of extensive element-based local operations. While these properties are naturally preserved for an explicit scheme, the challenge in the development of an implicit or steady-state solution scheme is to obtain a rapidly converging method which maintains the favorable locality and scalability properties of these discretizations. Thus, for example, an exact Newton scheme using sparse matrix inversion with fill-in, can be expected to perform poorly for such applications.

The approach taken in this work is to develop an efficient *hp*-multigrid scheme for Discontinuous Galerkin (DG) discretizations, which on the one hand delivers near optimal convergence rates independent of p (order of accuracy) and h (degree of mesh resolution), while on the other hand, maintaining good scalability by resorting to locally implicit techniques (at the element level) on each level of the multigrid sequence. The idea of a multigrid method is to accelerate the solution of a fine grid problem by computing corrections on coarser grid levels, where the low-frequency errors are treated more effectively, and the smoothing iterations are less costly. For high-order DG discretizations, a natural approach is to use lower-order discretizations on the same physical grid as the coarser multigrid levels. This *p*-multigrid approach was originally proposed by Ronquist and Patera,[5] and has been pursued for the steady-state Euler equations in previous work by the second author [6] and by Fidkowski et al.[7,8] for the Navier-Stokes equations with encouraging results. More recently, $p-$multigrid has been extended to time-dependent implicit flows, where it has been used to solve the non-linear system arising at each implicit time step, using low and high-order implicit time discretizations in conjunction with high-order spatial DG discretizations.[9]

*Postdoctoral Research Associate, AIAA Member; email: nastase@uwyo.edu.
†Professor, AIAA Associate Fellow; email: mavripl@uwyo.edu.

In order to deliver favorable convergence rates, the coarsest level ($p = 0$) problem in the p-multigrid scheme must be sufficiently converged at each multigrid cycle. While the computational effort required to converge the $p = 0$ level problem for two-dimensional and small three-dimensional problems is often insignificant using an explicit or locally implicit scheme, this work grows quadratically with the number of mesh cells, and will become problematic for large scale three-dimensional problems. A related approach which overcomes these deficiencies can be found in the implicit multi-level solution techniques for high-order discretizations developed by Lottes and Fisher.[10]

The approach taken in this work is to use a more traditional h-multigrid method for solving the $p = 0$ problem within the p-multigrid context. Thus, the so called hp-multigrid approach involves the use of multiple p-levels, supplemented by a sequence of physically coarser agglomerated unstructured meshes, all of which are used within a single multigrid cycle. In previous work, the hp-multigrid approach has been demonstrated for two-dimensional and small three dimensional problems, and was found to be capable of delivering p- and h-independent convergence rates.[11,12]

In the current work, the hp-multigrid approach in three dimensions is extended to run on massively parallel computer architectures, and the performance of this solver is investigated, both in terms of convergence efficiency and parallel scalability, on highly resolved meshes, for realistic aerodynamic configurations, for various orders of accuracy, using large numbers of processors.

II. Governing Equations

The conservative form of the compressible Euler equations describing the conservation of mass, momentum and total energy are given in vectorial form

$$\frac{\partial \mathbf{U}(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0 \tag{1}$$

subject to appropriate boundary and initial conditions within a three-dimensional domain Ω. Explicitly, the state vector $\mathbf{U}$ of the conservative variables and the Cartesian components of the inviscid flux $\mathbf{F} = (\mathbf{F}^x, \mathbf{F}^y, \mathbf{F}^z)$ are:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E_t \end{pmatrix}, \quad \mathbf{F}^x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E_t + p) \end{pmatrix}, \quad \mathbf{F}^y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E_t + p) \end{pmatrix}, \quad \mathbf{F}^z = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E_t + p) \end{pmatrix}, \tag{2}$$

where ρ is the fluid density, (u,v,w) are the fluid velocity Cartesian components, p is the pressure and E_t is the total energy. For an ideal gas, the equation of state relates the pressure to total energy by:

$$p = (\gamma - 1)\left[E_t - \frac{1}{2}\rho(u^2 + v^2 + w^2)\right] \tag{3}$$

where $\gamma = 1.4$ is the ratio of specific heats.

III. Spatial Discretization

The computational domain Ω is partitioned into an ensemble of non-overlapping elements and within each element the solution is approximated by a truncated polynomial expansion

$$\mathbf{U}(\mathbf{x},t) \approx \mathbf{U}_p(\mathbf{x},t) = \sum_{j=1}^{M} \mathbf{u}_j(t)\phi_j(\mathbf{x}) \tag{4}$$

where M is the number of modes defining the truncation level. The semi-discrete formulation (*i.e.* continuous in time) employs a local discontinuous Galerkin formulation[13–16] in spatial variables within each element Ω_k. The weak formulation for Eq. (1) is obtained by minimizing the residual with respect to the expansion function in an integral sense:

$$\int_{\Omega_k} \phi_i \left[\frac{\partial \mathbf{U}_p(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}_p)\right]_k d\Omega_k = 0 \tag{5}$$

After integrating by parts the weak statement of the problem becomes:

$$\int_{\Omega_k} \phi_i \frac{\partial \mathbf{U}_p}{\partial t} d\Omega_k - \int_{\Omega_k} \nabla \phi_i \cdot \mathbf{F}(\mathbf{U}_p) d\Omega_k + \int_{\partial\Omega_k} \phi_i \mathbf{F}^*(\mathbf{U}_p) \cdot \mathbf{n} d(\partial\Omega_k) = 0 \tag{6}$$

The local discontinuous Galerkin approach makes use of element-based basis functions, which results in solution approximations which are local, discontinuous, and doubled valued on each elemental interface. Monotone numerical fluxes are used to resolve the discontinuity, providing the means of communication between adjacent elements and specification of the boundary conditions. This local character of the discontinuous Galerkin method simplifies tremendously the parallel implementation, as will be shown later. The numerical flux, $\mathbf{F}^*(\mathbf{U}_p) \cdot \mathbf{n}$, is obtained as a solution of a local one-dimensional Riemann problem and depends on the internal interface state, $\mathbf{U}_p^-$, the adjacent element interface state, $\mathbf{U}_p^+$ and the orientation as defined by the normal vector, $\mathbf{n}$, of the interface. An approximate Riemann solver is used to compute the flux at inter-element boundaries. Current implementations include the flux difference splitting schemes of Rusanov,[17] Roe,[18] HLL[19] and HLLC.[20–22]

The discrete form of the local discontinuous Galerkin formulation is defined by the particular choice of the set of basis functions, $\{\phi_i, i = 1 \ldots M\}$. The basis set is defined on the master element $\widehat{\Omega}(\xi_j, j = 1 \ldots 3)$ spanning between $\{-1 < \xi_j < 1\}$. We seek a set of hierarchical basis functions in order to simplify our subsequent spectral multigrid implementation. The basis set contains *vertex*, *edge* and *bubble* functions[23,24] based on Jacobi polynomials of variable weights. Since the basis set is defined in the master element, a coordinate transformation, $\mathbf{x}_p = \mathbf{x}_p(\xi_1, \xi_2, \xi_3)$, is required to compute the derivatives and the integrals in physical space $\Omega_k(x, y, z)$. For iso-parametric elements, the basis functions are expressed as functions of ξ_1, ξ_2 and ξ_3, and the coordinate transformation, and its Jacobian are given by:

$$\mathbf{x}_p = \sum_{j=1}^{M} \widehat{\mathbf{x}}_j \phi_j(\xi_1, \xi_2, \xi_3), \qquad J_k(\xi_1, \xi_2, \xi_3) = \left| \frac{\partial(x, y, z)}{\partial(\xi_1, \xi_2, \xi_3)} \right| \tag{7}$$

In the simple case of straight-sided or -faced elements the mapping is linear and its Jacobian, J_k, and its metrics are constant within each element, and can be evaluated just by using the element vertex coordinates. In the case of elements with curved faces, the Jacobian J_k varies within the element, and must be evaluated explicitly at the individual quadrature points in the numerical integration process. While straight-sided/faced elements are used in the interior of the domain, curved boundary elements which conform to the original description of the boundary geometry must be used in order to retain the $p+1$ accuracy property of the discretization scheme. Since most mesh generation packages produce a list of elements, with coordinates given only for the corner points of the elements, additional information is required in order to create curved boundary elements. This is achieved by projecting additionally created surface points onto the original boundary geometry description, as depicted in Figure 1 for the two-dimensional case.

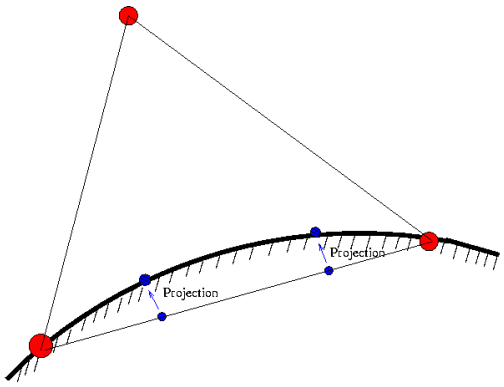


Figure 1. Two-dimensional illustration of projection of additional surface points for creation of curved surface element.

The number of additional surface points to be created on each face is a function of the p-order of the discretization. These points are initially created on the flat face of the non-curved geometry using linear interpolation, and then projected onto the surface of the original defining geometry used in the grid generation process. In this work, the VGRID grid generation program[25] is used to generate three-dimensional unstructured tetrahedral meshes. A projection utility which snaps an arbitrary list of points to the closest surface geometry used in VGRID was developed and provided for this purpose.[26] The correspondence between the coordinates of the new projected surface points in physical space and in isoparametric space is then used to compute the exact Jacobian for the curved elements.

While elements with a face on the geometry boundary must be treated as curved elements, additional neighboring elements are also required to be treated as curved elements for complete consistency. A simple algorithm for determin-

ing the extent of the set of curved elements begins by identifying all elements with one or more faces on the boundary. This constitutes the set of elements where all faces must be considered curved. The second step is to propagate the cell coefficients back to all elemental faces, which might have only one curved edge on the curved surface. The third step is to transfer all face coefficients to all adjacent elements, even if they are not physically on the curved surface. This will guarantee that all the curvature information is exact and matched for all elements close to the curved surface.

For the general case (*i.e.*curved elements), using Eq. (7), the solution expansion and the weak statement within each element, $\widehat{\Omega}_k$, becomes:

$$\mathbf{U}_p(\xi,\eta,t) = \sum_{j=1}^{M} \widehat{\mathbf{u}}_j(t)\phi_j(\xi_1,\xi_2,\xi_3) \tag{8}$$

$$\int_{\widehat{\Omega}_k} \phi_i \frac{\partial \mathbf{U}_p}{\partial t} |J_k| d\widehat{\Omega}_k - \int_{\widehat{\Omega}_k} \nabla\phi_i J_k^{-1} \cdot \mathbf{F}(\mathbf{U}_p) |J_k| d\widehat{\Omega}_k + \int_{\partial\widehat{\Omega}_k} \phi_i \mathbf{F}^*(\mathbf{U}_p) \cdot \mathbf{n} |J_k| (\partial\widehat{\Omega}_k) = 0 \tag{9}$$

The resulting semi-discrete form, Eq. (9), can be further simplified as:

$$\mathbf{M}\frac{d\mathbf{U}_p}{dt} + \mathbf{R}(\mathbf{U}_p) = 0 \tag{10}$$

where $\mathbf{M}$ and $\mathbf{R}(\mathbf{U}_p)$ represent the mass matrix and the non-linear residual associated with the master element $\widehat{\Omega}_k$, respectively. This system of ordinary equations, Eq. (10), is solved in the modal space and the integrals are evaluated by economical Gaussian quadrature rules,[23,27,28] which requires a projection of the solution values to the quadrature points used in the numerical integration. In order to preserve the $p+1$ accuracy order of the numerical approximation, the element integral uses quadrature rules which are exact for polynomial degree $2p$ within the master element, while the boundary integral uses quadrature rules which are exact for polynomial degree $2p+1$.[29] For boundary elements with curved edges or faces, the Jacobians must be evaluated at the integration quadrature points, whereas for interior elements with straight edges or faces, these are constant and need only be evaluated once for each element.

IV. The Implicit Solver

For steady-state cases, the temporal derivative term may be omitted and the system of equations (Eq. (10)) associated with each element becomes:

$$\mathbf{R}(\mathbf{U}_p) = 0 \tag{11}$$

where $\mathbf{R}(\mathbf{U}_p)$ now represents the non-linear steady-state residual. We use variants of an element-Jacobi scheme to solve this system of equations. The Newton iteration associated with Eq. (11) yields at each "$n+1$" step:

$$\begin{aligned} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}_p}\right]^n \Delta\mathbf{U}_p^{n+1} &= -\mathbf{R}(\mathbf{U}_p^n) \\ \mathbf{U}_p^{n+1} &= \mathbf{U}_p^n + \alpha\Delta\mathbf{U}_p^{n+1} \end{aligned} \tag{12}$$

where α is a parameter used for robustness to keep $\|\alpha\Delta\mathbf{U}_p^{n+1}/\mathbf{U}_p^{n+1}\|_{L_\infty} \leq 10\%$. An element-Jacobi scheme can be defined as an approximate Newton scheme where the full Jacobian matrix is replaced by the block diagonal entries representing the coupling between all modes within each element, $[\partial\mathbf{R}/\partial\mathbf{U}_p] \approx [D]_i$, thus neglecting the coupling between neighboring element modes, which arises through the inter-element flux evaluations. The $[D]$ blocks represent small dense matrices associated with each grid element. These element matrices are inverted using Gaussian elimination to produce a lower-upper (LU) factorization of each element matrix. In the case of the three-dimensional Euler equations (Eq. (1)) the number of entries in the block diagonal matrix ($[D]$) for each tetrahedral element is given in Table (1). Clearly, the growth in size of $[D]$ is a non-linear function of the discretization order, and dictates the memory requirement. Therefore, in our simulations we keep the discretization order to $p \leq 6$, which should be reasonable for aerodynamic applications. The non-linear iteration Eq. (12) becomes:

$$\Delta\mathbf{U}_p^{n+1} = [D^n]^{-1}(-\mathbf{R}(\mathbf{U}_p^n)) \tag{13}$$

This solver is denoted as the non-linear element Jacobi (NEJ). A second variant of this solver it the *quasi* non-linear element Jacobi (qNJ). This variant employs "k" quasi non-linear iterations, where only the residual, $\mathbf{R}(\mathbf{U}_p^k)$, is updated, and the block diagonal matrices, $[D^n]$, are kept constant throughout the outer-iteration "n". Therefore, the $(k+1)^{th}$ step is:

$$\Delta\mathbf{U}_p^{k+1} = [D^n]^{-1}(-\mathbf{R}(\mathbf{U}_p^k)) \tag{14}$$

where the $[D^n]^{-1}$ matrix is actually stored in LU factorized form. This approach is expected to yield similar converge rates per cycle as in the NEJ variant, with improved performance in terms of CPU time, since the expensive $[D]$ matrix assembly and LU factorization procedure are performed less frequently. In order to speed up the convergence, a non-linear element Gauss-Seidel approach is also possible, based on the *quasi* element-Jacobi solver. Hence, a third variant of this solver is the *quasi* non-linear element Gauss-Seidel (qNGS).

p	Size of $[D]$
0	5×5
1	20×20
2	50×50
3	100×100
4	175×175
5	280×280
6	420×420

Table 1. The size of the diagonal matrix $[D]$ as a function of expansion order (p) for tetrahedral element.

The next variants will consider a linearized type of integration strategy. The fourth variant of this solver is denoted as the linearized element Jacobi (LEJ) method. In this approach, the full Jacobian matrix is retained, but is decomposed into block diagonal $[D]$ and off-diagonal $[O]$ components:

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}_p}\right]^n = [D^n] + [O^n] \tag{15}$$

An iterative procedure can now be written by taking the $[O]$ components, which contain terms arising from the inter-element flux evaluations, to the right-hand-side of Eq. (12). In matrix form the $(k+1)^{th}$ step of the linearized element Jacobi step is written as:

$$\Delta \mathbf{U}_p^{k+1} = [D^n]^{-1} \left(-\mathbf{R}(\mathbf{U}_p^n) - [O^n] \Delta \mathbf{U}_p^k \right) \tag{16}$$

Note that the linearized element Jacobi scheme also involves a dual iteration strategy, where each n^{th} outer non-linear iteration entails "k" inner linear iterations. The advantage of this formulation is that the non-linear residual $\mathbf{R}(\mathbf{U}_p^n)$ and the Jacobian entries $[D^n]$ and $[O^n]$ are held constant during the linear iterations. This can significantly reduce the required computational time per cycle for expensive non-linear residual constructions. Because this scheme represents an exact linearization of the element-Jacobi scheme (Eq. (13)), both approaches can be expected to converge at the same rates per cycle (asymptotically).[30] On the other hand, the linearized element Jacobi scheme requires extra storage for the $[O]$ Jacobian blocks, which may not be feasible for large three-dimensional problems.

The convergence of Eq. (16) can be further accelerated by using a Gauss-Seidel strategy where the off-diagonal matrices are split into lower, $[L]$, and upper, $[U]$ contributions (*i.e.* $[O] = [L] + [U]$). This last solver variant (LGS) becomes:

$$\Delta \mathbf{U}_p^{k+1} = [(D+L)^n]^{-1} \left(-\mathbf{R}(\mathbf{U}_p^n) - [U^n] \Delta \mathbf{U}_p^k \right) \tag{17}$$

which again involves a dual iteration strategy, but follows an ordered sweep across the elements using latest available neighboring information in the Gauss-Seidel sense.

V. The *hp*-Multigrid Approach

Multigrid methods are known as efficient techniques for accelerating convergence to steady state for both linear and non-linear problems,[30,31] and can be applied with a suitable existing relaxation technique. The rapid convergence property relies on an efficient reduction of the solution error on a nested sequence of coarse grids.

The spectral multigrid approach is based on the same concepts as a traditional h-multigrid method, but makes use of "coarser" levels which are constructed by reducing the order of accuracy of the discretization, rather than using physically coarser grids with fewer elements. Thus, all grid levels contain the same number of elements, which alleviates the need to perform complex interpolation between grid levels and/or to implement agglomeration-type procedures.[30] Furthermore, the formulation of the interpolation operators, between fine and coarse grid levels, is

greatly simplified when a hierarchical basis set is employed for the solution approximation. The main advantage is due to the fact that the lower order basis functions are a subset of the higher order basis (*i.e.* hierarchical) and the *restriction* and *prolongation* operators become simple projection operators into a lower and higher order space, respectively.[7] Therefore their formulation is obtained by a simple deletion or augmentation of the basis set. The *restriction* from fine to coarse level is obtained by disregarding the higher order modal coefficients and transferring the values of the low order modal coefficients exactly. Similarly, the *prolongation* from coarse to fine levels is obtained by setting the high order modes to zero and injecting the values of the low order coefficients exactly.

Multigrid strategies are based on a recursive application of a two-level solution mechanism, where the second (coarser) grid is solved exactly, and used to accelerate the solution on the finer grid.[31] Because the exact solution of the coarse grid problem at each multigrid cycle is most often prohibitively expensive, the recursive application of multigrid to solve the coarse grid problem offers the preferred approach for minimizing the computational cost of the multigrid cycle, thus resulting in a complete sequence of coarser grids. For spectral (p)-multigrid methods, the recursive application of lower order discretizations ends with the $p = 0$ discretization on the same grid as the fine level problem. For relatively fine meshes, the (exact) solution of this $p = 0$ problem at each multigrid cycle can become expensive, and may impede the h-independence property of the multigrid strategy. The $p = 0$ problem can either be solved approximately by employing the same number of smoothing cycles on this level as on the finer p levels, or the $p = 0$ problem can be solved more accurately by performing a larger number of smoothing cycles at each visit to this coarsest level. In either case, the convergence efficiency will be compromised, either due to inadequate coarse level convergence, or to excessive coarse level solution cost. An alternative is to employ an h-multigrid procedure to solve the coarse level problem at each multigrid cycle. In this scenario, the p-multigrid scheme reverts to an agglomeration multigrid scheme once the $p = 0$ level has been reached, making use of a complete sequence of physically coarser agglomerated grids, thus the designation hp-multigrid. Agglomeration multigrid methods make use of an automatically generated sequence of coarser level meshes, formed by merging together neighboring fine grid elements, using a graph algorithm. First-order accurate ($p = 0$) agglomeration multigrid methods for unstructured meshes are well established and deliver near optimal grid independent convergence rates.[32] This procedure has the potential of resulting in a truly h- and p-independent solution strategy for high-order accurate discontinuous Galerkin discretizations. Figure (2) illustrates a three dimensional view of a typical two level h-multigrid agglomerated (AMG) configuration.

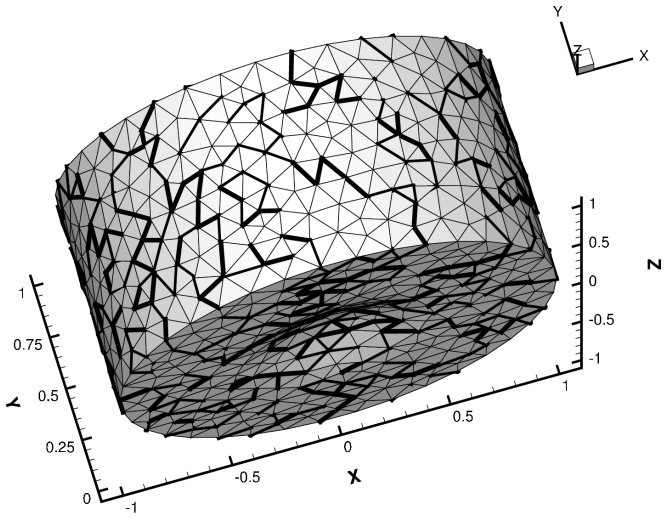


Figure 2. A typical two level h-multigrid mesh configuration.

V.A. Parallel Implementation

In the current-day scientific computing environment, realistic three-dimensional simulations will almost always require the use of massively parallel distributed memory computer architectures. The discontinuous Galerkin hp-multigrid solver described in this work achieves parallelism through domain decomposition, and uses the standard MPI message-passing library for inter-processor communication.[33] The mesh is partitioned using the METIS graph partitioner[34] operating on the dual graph of the mesh, where the cell centroids represent the graph vertices, and faces delimiting

neighboring cells represent the graph edges. The partitioning operation assigns groups of cells to individual partitions, and faces which border on two cells in different partitions are said to be intersected, and are assigned uniquely to one of these partitions, while a ghost cell is created in the partition containing the intersected face. This ghost cell corresponds to the neighboring cell in the remote partition for the intersected face, as depicted in Figure 3, and is used as a buffer to temporarily store the flux values computed on the face, which are then sent and accumulated to the real image of the ghost cell in the remote partition using the MPI communication library. This approach entails no duplication of computations in adjacent partitions, and results in the same number of operations and exact same residual values at each iteration as the corresponding sequential algorithm. Standard techniques for masking communication latency are employed, including packing all messages destined for the same remote partition into a single large message, and using non-blocking send and receive calls through the MPI library.

Since the mesh topology is unchanged for the various p levels of the *hp*-multigrid algorithm, the same communication patterns are used for all p levels. Furthermore, no communication is required in the restriction and prolongation operations between the various p-multigrid levels.

On the other hand, the agglomerated coarse levels used in the *h*-portion of the *hp*-multigrid algorithm are all partitioned independently with METIS, and then sorted in order to maximize the overlap between coarse and fine h-level partitions. Thus, on each agglomerated coarse h-level, a different communication pattern is used, and additional communication patterns are required for performing the restriction and prolongation between the coarse and fine levels of the h-multigrid sequence. The partitioning operations and determination of the communication patterns and buffers are performed sequentially and are precomputed and stored prior to the initiation of the flow solution phase.

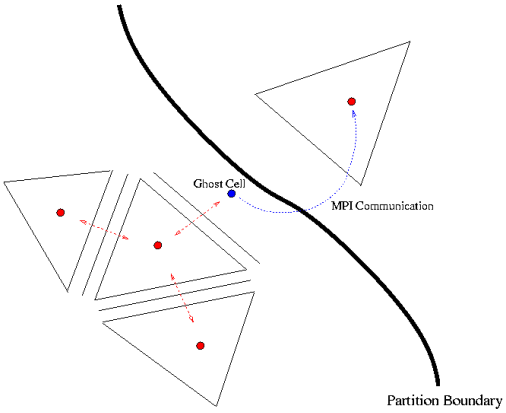


Figure 3. Illustration of intersection of mesh elements at inter-processor partition boundary, creation of ghost cell, and communication between ghost cell and real image in remote partition.

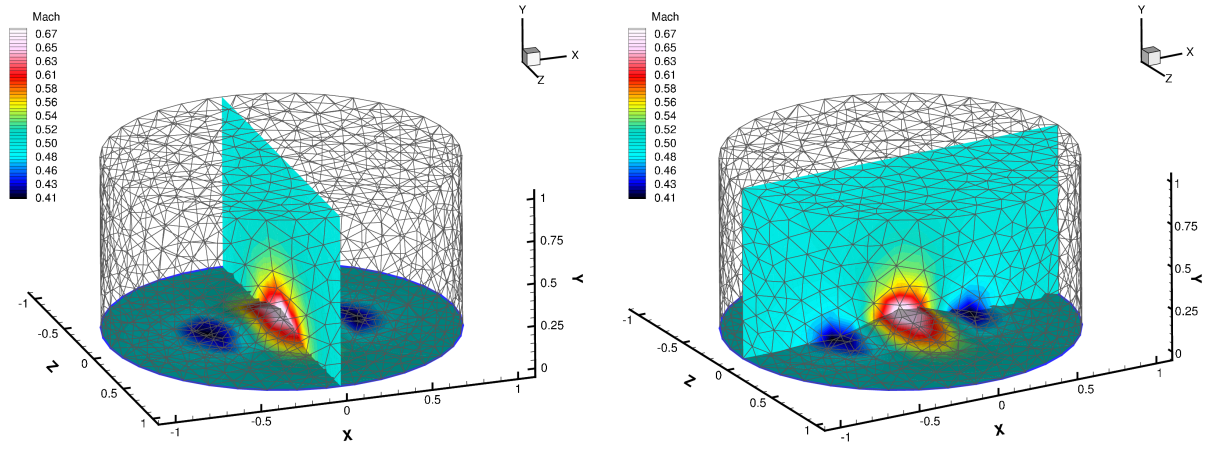
VI. Simple flow configuration

The accuracy of the spatial discretizations and the efficiency of the solution schemes described above are evaluated for the Euler equations using a test problem consisting of the compressible flow over a three-dimensional bump. A series of four grids on this configuration have been generated, consisting of $N = 1220$, 5041, and 10349 tetrahedral elements, respectively, in order to study the grid convergence of the discontinuous Galerkin discretizations of various orders. For each case the solution was converged to machine zero in the discretization error studies. Unless otherwise stated, all the simulated results are initiated with a $p = 0$ solution obtained a priori. The full domain extends from $-1 \leq x \leq 1$ in the stream-wise direction, from $0 \leq y \leq 1$ in the vertical direction and from $-1 \leq z \leq 1$ in the cross-stream direction, with wall boundaries at $y = 0$. All the simulation results for this configuration are performed with the HLLC flux using a single processor machine.

VI.A. Single-Grid Results

The aforementioned flow configuration is used to assess the accuracy of the DG discretizations. Figure (4) shows the Mach contour lines for a free-stream Mach number of $M_\infty = 0.5$, for the $p = 4$ (fifth-order) accurate discretization on

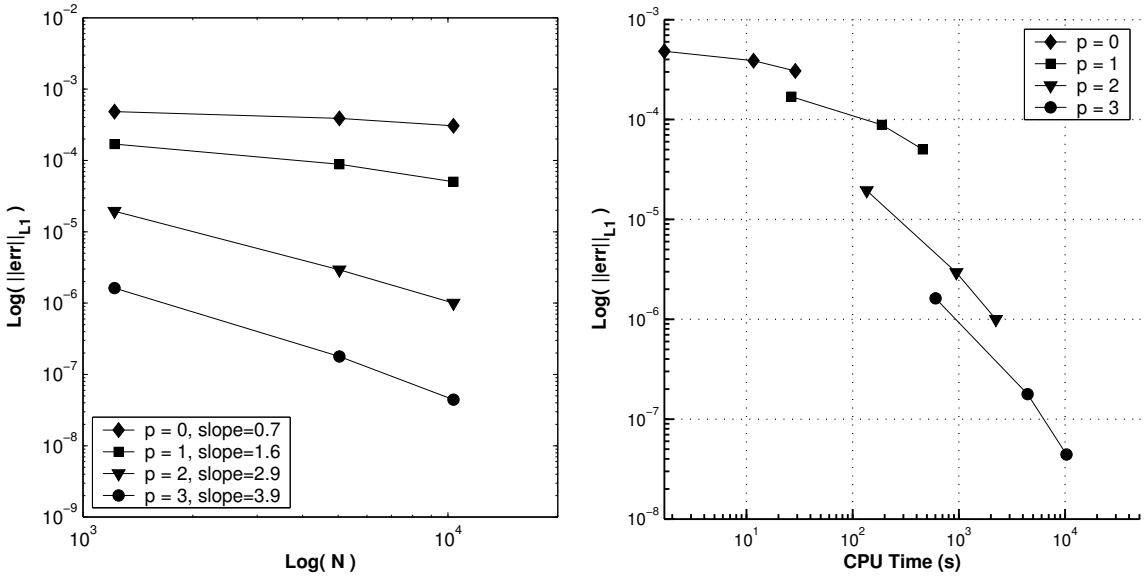
the finest mesh of 10,349 cells.



(a) Cross-stream section (b) Stream-wise section

Figure 4. The Mach number contours for the three-dimensional bump case on 10,349 element mesh using fifth-order accurate (p=4) discretization.

Figure (5(a)) shows the accuracy (*i.e.* the L_1 entropy error norm) of the steady-state solution for $1st$, $2nd$, $3rd$ and $4th$ order accurate discretizations as a function of the number of elements. For three-dimensional configurations the number of elements, N, is proportional to $1/h^3$, where h represents an approximation of the cell size. The asymptotic slope of these curves indicates that the optimal error convergence rate ($\approx h^{p+1}$) is obtained. A comparison of the computed accuracy versus CPU time is given in Figure (5(b)), where the various p-discretizations have been converged to machine zero on the various grid configurations using the *quasi* linearized element Jacobi driven multigrid scheme. In general, for a given level of accuracy, the CPU time decreases when the approximation order is increased, with the benefit increasing for smaller accuracy tolerances.

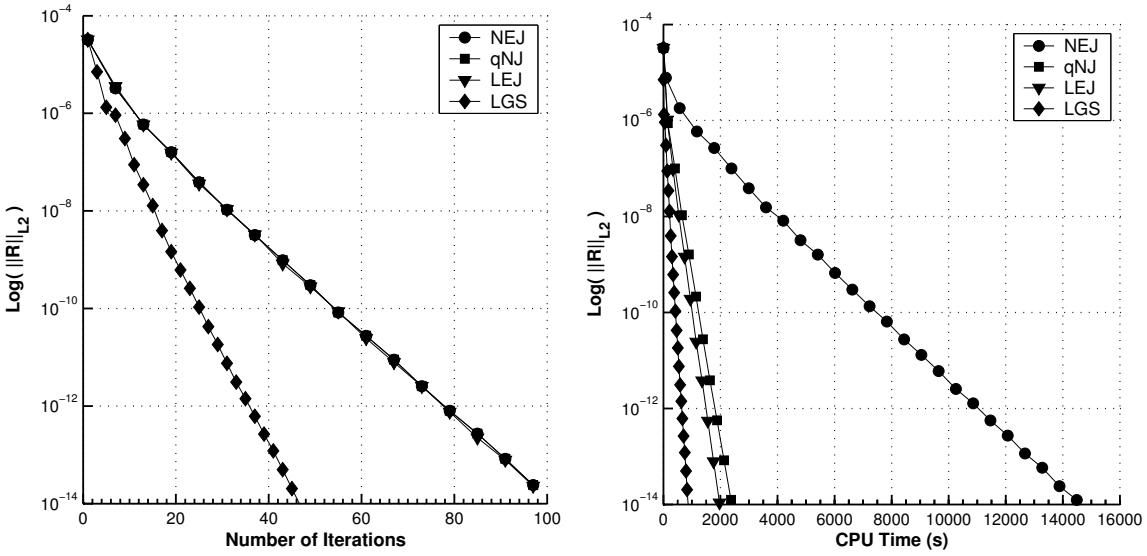


(a) (b)

Figure 5. The L_1 norm of the entropy error as a function of: (a) h/p-refinement; (b) CPU time.

Figure (6(a)) depicts the convergence of the non-linear element Jacobi, *quasi* non-linear element Jacobi, linearized element Jacobi, and linearized element Gauss-Seidel schemes on the mesh of $N = 10349$ elements, for the $p = 4$ discretization on a single mesh level, in the absence of the multigrid scheme. The convergence is measured in terms of overall number of cycles, linear cycles for the linear schemes, and non-linear cycles for the element-Jacobi scheme. As expected, the non-linear element-Jacobi, quasi non-linear element Jacobi and linearized element-Jacobi schemes

converge at similar rates in terms of numbers of cycles, while the linearized element Gauss-Seidel scheme converges substantially faster. When compared in terms of CPU time, Figure (6(b)), the linearized element-Jacobi and element Gauss-Seidel schemes are seen to be substantially more efficient than the non-linear element Jacobi scheme. The linearized schemes utilize 10 linear iterations between each non-linear update, and thus result in 10 times fewer non-linear residual and Jacobian evaluations and LU factorizations than the element-Jacobi scheme. The savings are substantial due to the large cost of the dense Jacobian factorization procedure. The quasi non-linear element Jacobi proves to be an appropriate compromise for three-dimensional cases where memory limitations are important (*i.e* only storage of the diagonal blocks, $[D]$, is required).



(a) (b)

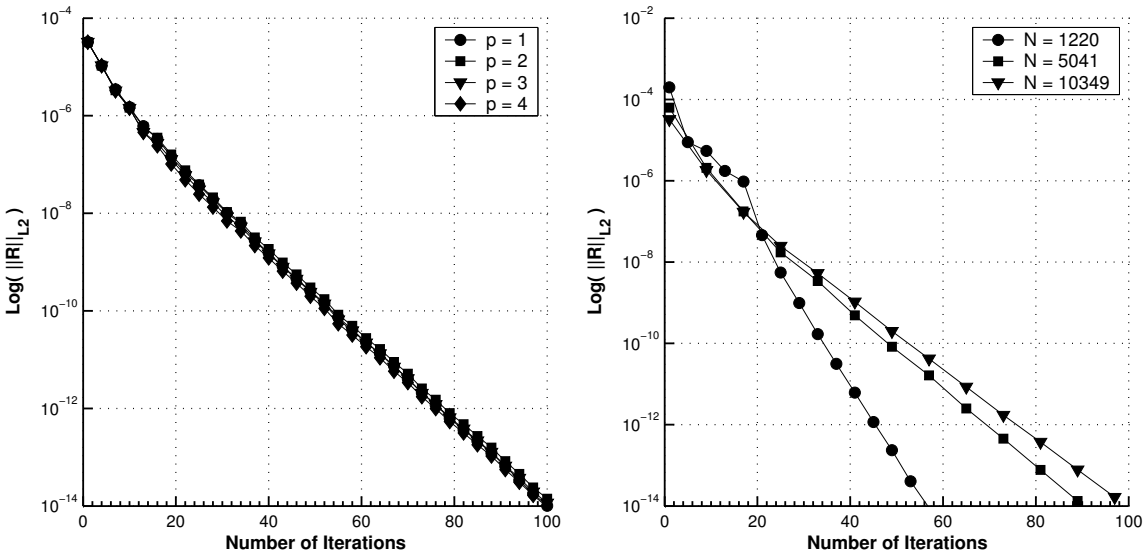
Figure 6. Comparison of convergence of non-linear element-Jacobi (NEJ), quasi non-linear element-Jacobi (qNJ), linear element-Jacobi (LEJ), and linear element Gauss-Seidel (LGS), on a mesh size of $N = 10349$ elements and order $p = 4$, in terms of: (a) Number of iterations; (b) CPU time.

Due to memory limitations, the remaining results will make exclusive use of the quasi non-linear element Jacobi (qNJ) scheme. Figure (7(a)) illustrates the convergence of the qNJ solver as measured by the rate of the residual reduction versus the number of iterations, for approximation orders varying from $p = 1$ to $p = 4$, on the mesh of $N = 10349$ elements. Clearly, the method yields a convergence rate which is independent of the order of accuracy of the discretization for a fixed size grid. However, increasing the number of elements, N, has an adverse effect on the convergence rate. In Figure (7(b)), the convergence rate for $p = 4$ is seen to degrade as the number of mesh elements is increased. This h-dependence of the element-Jacobi solver is addressed through the use of the hp-multigrid scheme.

VI.B. Multigrid Results

In the context of the hp-multigrid methodology, the same flow configuration is considered with geometrical parameters, boundary and initial conditions as defined in Section (VI). In light of the multigrid method described in the previous section, one can devise multiple techniques depending on the number of p- and h- levels consider as well as the number of iterations per level. For all simulations performed, the qNJ solver is employed via a full-multigrid (FMG) strategy. Herein we consider four cases:

- RUN1, where only $4p$-levels are considered from $p = 4$ to $p = 1$, using 10 iterations per level.
- RUN2, where $4p$-levels and $1h$-level ($p = 0$) are considered, using 10 iterations per level.
- RUN3, where $4p$-levels and $1h$-level ($p = 0$) are considered, using 10 iterations per $p-$level and the h-level is converged to machine precision.
- RUN4, where $4p$-levels together with 3,4 and 5 h-levels (AMG) are considered for mesh sizes of $N = 1220$, $N = 5041$, $N = 10349$ elements, respectively, using 10 iterations per level.



(a) (b)

Figure 7. The L_2 norm of the residual *vs.* number of linear element-Jacobi (LEJ) cycles for: (a) Fixed mesh size of $N = 2015$ elements and various orders (p); (b) Fixed order $p = 4$

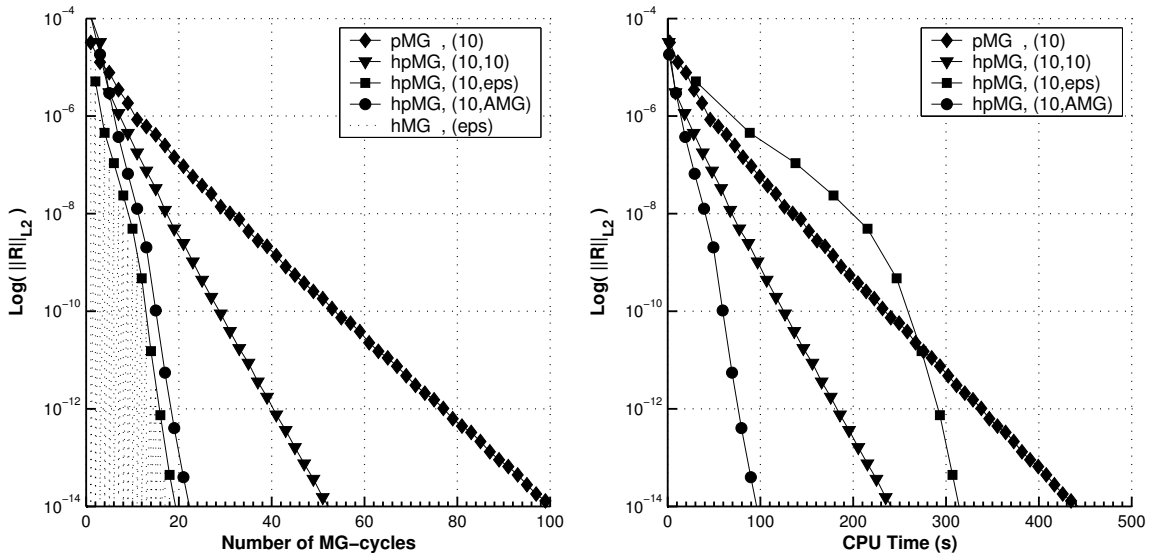
Figure (8(a)) depicts the convergence rate for a fixed mesh size of $N = 10349$ elements and $p = 1$ order, for all four cases. The dotted line represents the $p = 0$ residual convergence for the RUN3 case. Clearly, the RUN3 case displays the fastest convergence in term of number of multigrid cycles. This is actually the maximum convergence rate since the coarsest level is solved exactly. Therefore, the RUN3 case is considered a reference case. It is important to observe that the RUN4, which contains 3 agglomerated h-levels, displays the same convergence rate (*i.e.* slope) as RUN3, while performing the same number of smoothing steps (10) all levels. As expected, in terms of CPU-time, the RUN4 case outperforms all other cases (Figure (8(b))). This behavior is maintained as well for higher orders (not shown). Therefore, the RUN4 is the multigrid strategy of choice.

Figure (9)a illustrates the convergence rate for a fixed mesh size of $N = 10349$ elements and various orders ($p = 1,2,3,4$), for the RUN4 case, while Figure (9)b illustrates the convergence rate for a fixed order of $p = 4$ over the various coarse and fine meshes. These figures illustrate the $p-$ and $h-$ independence of the hp-multigrid algorithm, with only slight increase in the total number of iterations from 23 to 25 occurring when going from the 1220 to the 10,349 element mesh. In spite of the good performance of the hp-multigrid on this test case, the small mesh sizes and simple geometry make for a relatively simple problem, and the evaluation of the solver on more complex configurations is required.

VII. Complex flow configuration

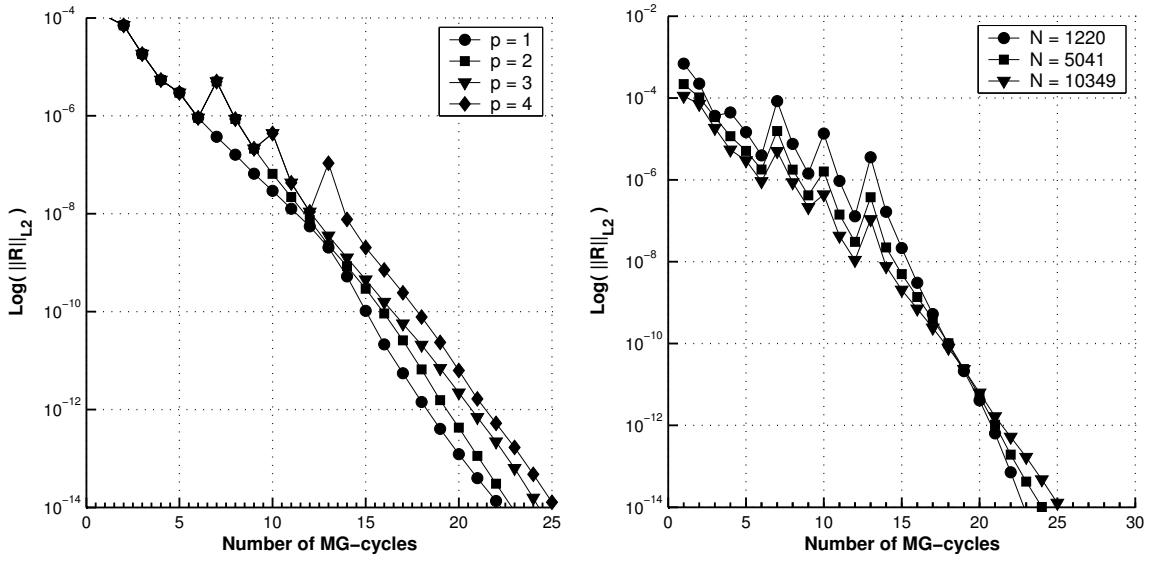
In order to fully assess the performance of the hp-multigrid DG solver both in terms of speed of convergence and parallel scalability, a more complex test problem consisting of compressible flow over a three-dimensional DLR-F6 wing-body configuration is considered. A series of three coarse to fine grids was generated on this configuration using the VGRID grid generation program.[25] The coarse, medium and fine meshes contain approximately 185,000, 450,000 and 2.6 million tetrahedral cells, respectively and are depicted in Figure 10. The agglomeration procedure was used to construct 4 coarse levels for the $N = 185,000$ mesh, 5 coarse levels for the $N = 450,000$ mesh, and 6 coarse levels for the $N = 2,600,000$ mesh, for use in the h-multigrid portion of the solver. These meshes were then partitioned and the computations were performed on an in-house Linux cluster, as well as on the NASA Columbia Supercomputer.

A full multigrid variant of the hp-multigrid solver was used exclusively. This is initiated by initializing a p=0 solution using freestream values, and approximately solving this low-order problem using a small number of h-multigrid cycles. The resulting flowfield is then used as the the initial condition for the solution of a p=1 problem, which is solved using the hp-multigrid scheme, and the process is repeated recursively until the desired p-level solution is reached. Ten multigrid cycles are used at each p-level in the full multigrid procedure, prior to reaching the last level, after which of the order of 100 multigrid cycles are used to fully converge the final highest p-level problem. Each multigrid cycle,



(a) (b)

Figure 8. The L_2 norm of the residual for a fixed mesh size of $N = 10349$ elements and a fixed $p = 1$ order: (a) *vs.* number of qNJ cycles; (b) *vs.* CPU time



(a) Mesh size of $N = 10349$ elements and various orders (p) (b) Fixed $p = 4$ order and various mesh sizes

Figure 9. The L_2 norm of the residual *vs.* the number of multigrid cycles, for the RUN4 case.

in turn, consists of a V-cycle with 10 quasi-non-linear element Jacobi cycles performed on the coarsening phase of the cycle. The freestream Mach number is taken as Mach=0.5, and the incidence is zero degrees, in order to avoid the formation of any shock waves for this case.

For large three-dimensional problems of this type, the use of the agglomerated h-grid levels is crucial for maintaining rapid convergence rates , since the low-order p=0 problem itself will be slowly converging in the absence of multigrid. This is demonstrated in Figure 11, where the convergence history of the p=0 problem on the coarse, medium and fine grid is depicted with and without the multigrid algorithm. In the single level (non-multigrid) case, the solution of the p=0 problem is seen to require a large number of iterations, which grows substantially as the mesh resolution is increased, reaching over 3000 cycles for the finest 2.6M cell mesh. On the other hand, the h-multigrid scheme produces consistent grid independent convergence rates for this problem in approximately 50 multigrid cycles. The convergence properties of the p=0 problem in terms of number of cycles and cpu time are summarized in Table (2).

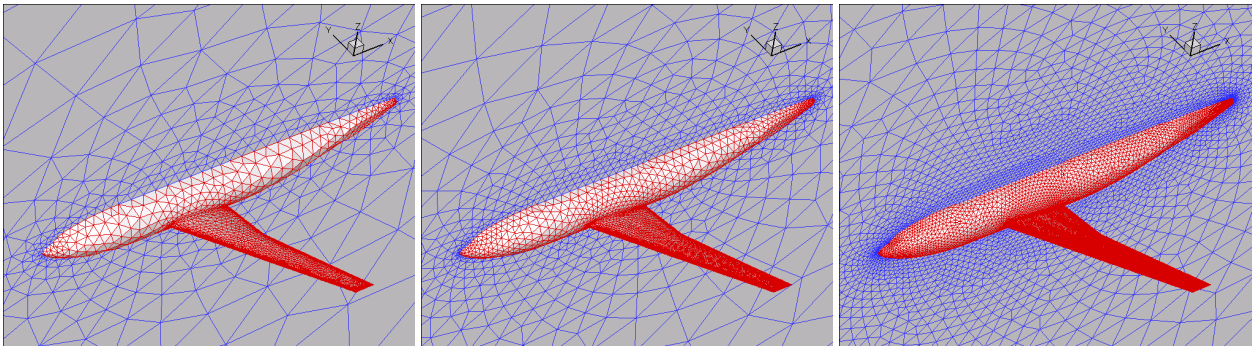
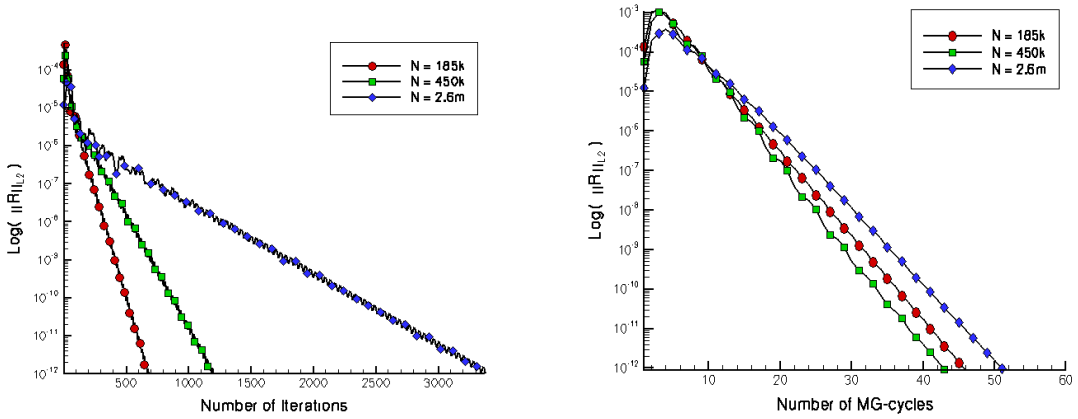


Figure 10. Coarse (185,000 cells), medium (450,000 cells) and fine (2.6M cells) mesh for wing body configuration.



(a) Single level (b) Multigrid

Figure 11. Comparison of convergence of $p = 0$ discretization (a) without and (b) with h-multigrid for coarse, medium and fine meshes for wing-body configuration.

N	Single grid	AMG	No. of AMG Levels
$185k$	679	46	4
$450k$	1200	43	5
$2.6m$	3375	51	6

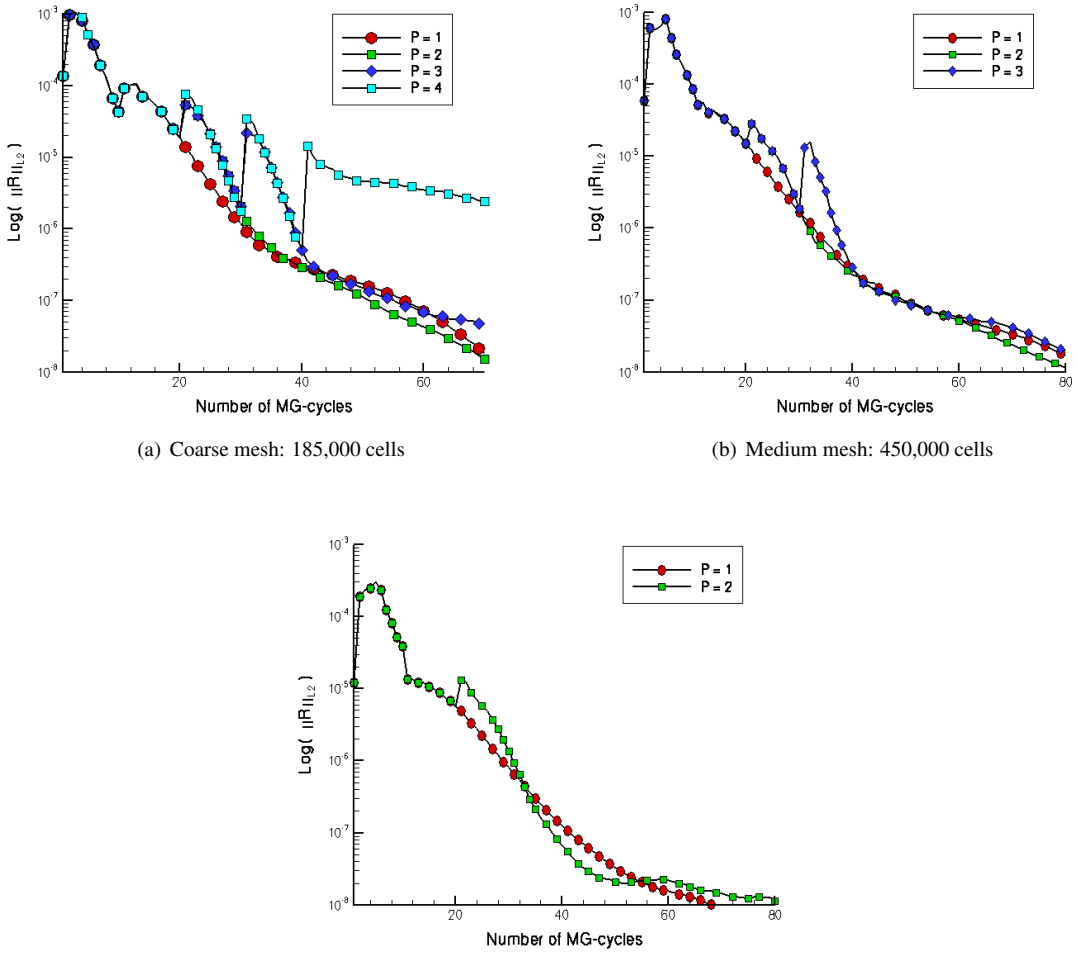
Table 2. Convergence in terms of number of iterations without and with multigrid for for fixed order $p = 0$ on coarse, medium and fine meshes.

VII.A. Multigrid Results

Figures 12 and 13 illustrate the convergence histories achieved on the three DLR-F6 wing-body meshes for various p orders of accuracy using the full *hp*-multigrid scheme. The peaks in the convergence plots correspond to the transition from low to higher-order discretizations in the initial phase of the full multigrid scheme. At each transition, the residual increases suddenly as a new higher-order discretization is initialized, but quickly returns to the level observed by continuing the multigrid iterations on the original p-order discretization. Thereafter, the higher-order discretization resumes convergence at a similar rate to the lower order discretizations.

The plots in Figure 12 reveal essentially p-independent convergence rates for all three grids, except for the p=4 discretization which is slower to converge on the coarse 185,000 cell grid. On the finer grids, this discretization exhibited poor robustness and could not be made to converge adequately, while the p=3 discretization failed on the finest grid and was only run successfully on the coarse and medium grids.

Figure 13 compares the convergence rates for p=1 and p=2 on all three meshes (since the higher p discretizations failed to run on the finer meshes), showing nearly h-independent convergence rates going from 180,000 to 2.6 million cell meshes. In all cases, the residuals are reduced 5 to 7 orders of magnitude in under 100 *hp*-multigrid cycles, by which time the lift coefficient has attained its final converged value.



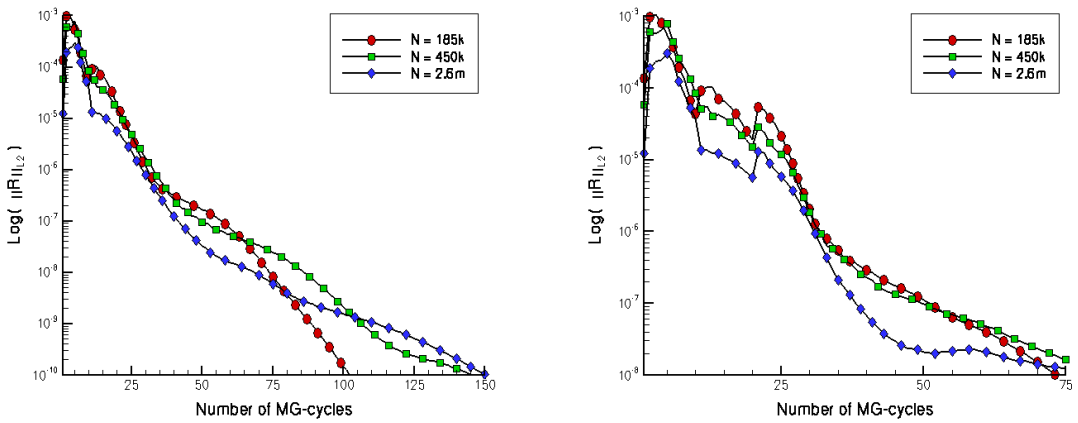
(a) Coarse mesh: 185,000 cells

(b) Medium mesh: 450,000 cells

(c) Fine mesh: 2.6 million cells

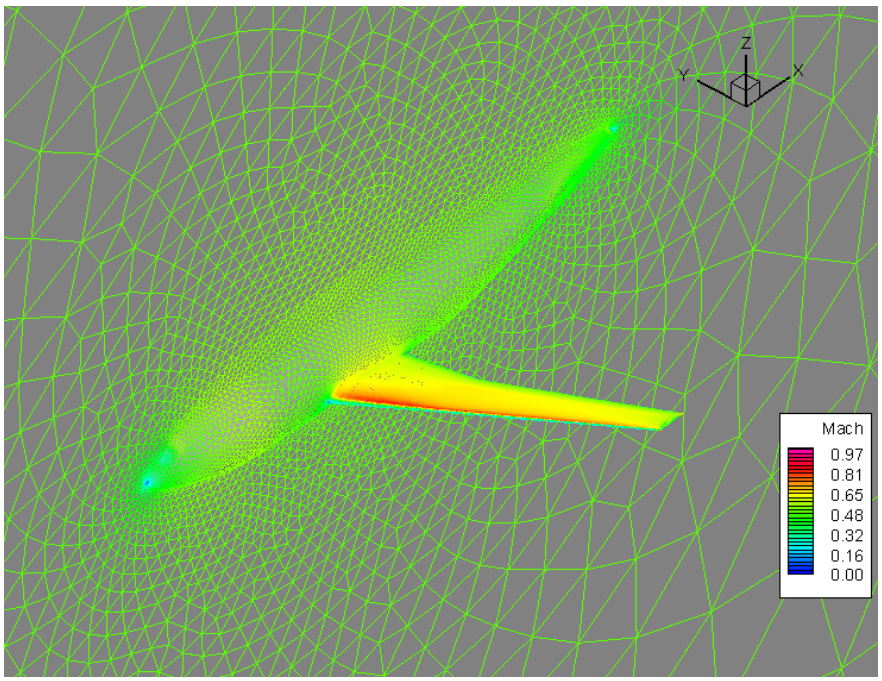
Figure 12. The L_2 norm of the residual *vs.* number of *hp*-multigrid cycles for compressible flow solution on wing-body configuration using various discretization orders and on various size meshes.

The solution in terms of computed Mach contours is displayed in Figure 14 for $p = 1$ order of accuracy on the fine mesh, illustrating the high accuracy achievable on relatively coarse meshes using high-order DG discretizations.



(a) $p = 1$ (b) $p = 2$

Figure 13. Variation of convergence histories for $p = 1$ and $p = 2$ discretizations with grid resolution as measured by the L_2 norm of the residual *vs.* number of hp-multigrid cycles on the various meshes.



(a) $p = 1$

Figure 14. Computed solution in terms of Mach contours on DLR-F6 wing body configuration.

VII.B. Computational Performance

The parallel performance of the *hp*-multigrid algorithm has been assessed by examining the scalability of this solver for various orders of accuracy (from p=0 to p=4) on the coarse (185,000 cells) and fine (2.6M cells) meshes, running on the NASA Columbia machine, using from 32 to 2008 cpus. The NASA Columbia Supercluster consists of a collection of loosely coupled SGI Altix nodes, each containing 512 tightly connected cpus. Four of these nodes are tightly coupled through a shared NUMALink4 interconnect, providing a global shared memory image and high bandwidth over 2048 cpus.[35] While the global shared memory image is not required in our case, due to the MPI implementation of the parallel *hp*-multigrid solver, the high bandwidth interconnect is helpful for achieving good scalability on more than 512 processors.

Figure 15a depicts the parallel speedup achieved for the *hp*-multigrid algorithm running on the coarse mesh of 185,000 cells, going from 32 to 1004 processors, where the 32 processor runs are assumed to exhibit perfect speedup, in view of the fact that runs on smaller number of processors were not performed for lack of memory at the higher orders of accuracy (p). In all cases, three h-multigrid levels are employed, with the appropriate number of p levels depending on the overall order of accuracy of the finest level. For p=0 (i.e. first-order accurate), which corresponds to the traditional agglomeration h-multigrid applied to a cell-centered finite volume scheme, scalability is seen to fall off dramatically as the number of processors is increased. This is to be expected, since the number of elements is relatively small in this case. For example, on 1004 cpus, the number of mesh cells per processor is less than 185 for this case. Thus, the ratio of computation to communication is extremely small in such cases. However, as the order of accuracy is increased, the scalability improves dramatically. For example, the second-order accurate p=1 discretization achieves a speedup of 750 on 1004 cpus, while the higher order discretizations achieve speedups close to 950 on 1004 cpus. Results are depicted in Figure 15b for the fine 2.6 million cell mesh using up to 2008 cpus. Similar trends are observed, with the p=0 results showing a significant drop-off in scalability, while the highest order results show near perfect scalability up to 2008 cpus, due to the increased computation to communication ratio for this finer mesh.

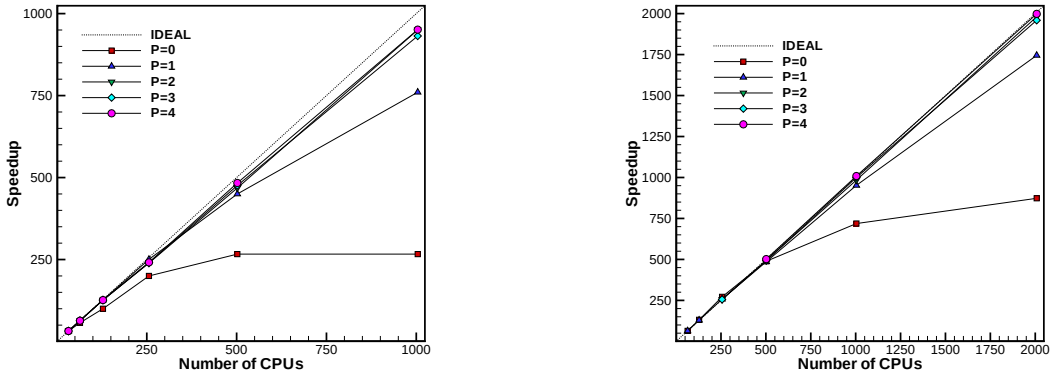
In spite of the fact that the higher-order p multigrid cycles exhibit superior scalability, these are significantly more costly per cycle than the lower order p solvers. In Figure 16 the work ratio, defined as the cpu time required for an *hp*-multigrid cycle ($p \geq 0$) divided by the cpu time required for a $p = 0$ multigrid cycle, is plotted as a function of p, and compared with the ratio of the square of the number of degrees of freedom in an element, as previously given in Table (1). These two ratios are seen to be nearly identical over the considered range of p, indicating that the cost of the *hp*-multigrid iteration is governed by the size (number of elements) of the diagonal element matrix $[D]$. The superior scalability of the higher order schemes is due to the large amount of computational effort required on each element for assembling, factorizing and multiplying the block diagonal element matrix, compared to the inter-element flux communication.

On 2008 cpus, a single multigrid iteration at p=4 required 83 seconds of wallclock time, while the same multigrid iteration at p=1 required only 1.2 seconds. This translates into an overall solution time of under 2 minutes for the p=1 case on the fine mesh, using approximately of the order of 100 multigrid cycles and potentially 2 hours for the $p = 4$ case, although the $p = 4$ (and $p = 3$) cases were not run to completion on the fine mesh due to robustness issues. Note that the $p = 4$ case on the finest grid corresponds to a problem with over 455 million degrees of freedom.

VIII. Concluding Remarks and Work in Progress

A high-order discontinuous Galerkin discretization using hierarchical basis functions on tetrahedra has been developed and implemented using an *hp*-multigrid approach. A non-linear element-Jacobi scheme is used as a smoother on each level of the multigrid sequence. The quasi-linear variant of the element Jacobi smoother, where the Jacobian is frozen for a number of iterations was found to provide the best compromise between computational efficiency and storage requirements for three-dimensional cases. The *hp*-multigrid scheme demonstrates approximately p-independent and h-independent convergence rates. The coupling of p- and h-multigrid procedures, through the use of agglomerated coarse levels for unstructured meshes, increases the overall solution efficiency compared to a p-alone multigrid procedure, and is essential for maintaining fast convergence for large three-dimensional cases. The combined *hp* multigrid solver preserves the locality and high computation to communication ratio inherent in DG discretizations and thus scales well on large numbers of processors, particularly for higher-order discretizations,

However, robustness issues remain for the higher-order discretizations on the finer grids and these are currently under investigation. Future work will concentrate on extending these techniques to the Navier-Stokes equations using hybrid element topologies, and on applying the multigrid solver to time-implicit problems in three-dimensions using higher-order spatial and temporal discretizations.



(a) (b)

Figure 15. The speedup *vs.* the number of processors for: (a) coarse mesh; (b) fine mesh

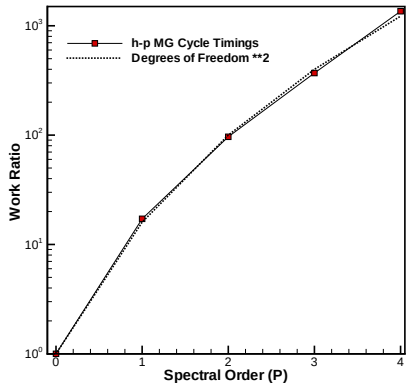


Figure 16. The work ratio and the square of the degrees of freedom *vs.* the spectral order (p).

Acknowledgments

The authors are grateful for the assistance of Shahyar Pirzadeh, NASA Langley Research Center, for providing the point projection facility used for creating curved element surfaces from VGRID geometry files. The computer time provided by NASA's Advanced Supercomputing Division on the NASA Columbia machine is also greatly acknowledged. This work was supported by a grant from the Office of Naval Research ONR grant number N00014-04-1-0602.

References

[1]Laflin, K., Brodersen, O., Rakowitz, M., Vassberg, J., Wahls, R., and Morrison, J., "Summary of Data from the Second AIAA CFD Drag Prediction Workshop," AIAA Paper 2004-0555.

[2]Vassberg, J. C., Tinoco, E. N., Mani, M., Brodersen, O. P., Eisfeld, B., Wahls, R. A., Morrison, J. H., Zickuhr, T., Laflin, K. R., and Mavriplis, D. J., "Summary of the Third AIAA CFD Drag Prediction Workshop," AIAA Paper 2007-0260.

[3]Karniadakis, G. E. and Sherwin, S. J., *Spectral/hp Element Methods for CFD*, Oxford University Press, 1999.

[4]Cockburn, B., Karniadakis, G. E., and Shu, C. W., *Discontinuous Galerkin Methods: Theory, Computation and Applications*, Springer, 2000.

[5]Ronquist, E. M. and Patera, A. T., "Spectral Element Multigrid I: Formulation and Numerical Results," *SIAM J. Sci. Comput.*, Vol. 2, No. 4, 1987, pp. 389–406.

[6]Helenbrook, B., Mavriplis, D. J., and Atkins, H., "Analysis of "p"-Multigrid for Continuous and Discontinuous Finite Element Discretizations," *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, 2003, AIAA Paper 2003–3989.

[7]Fidkowski, K. J. and Darmofal, D. L., "Development of a Higher-Order Solver for Aerodynamic Applications," *Proceedings of the 42nd Aerospace Sciences Meeting and Exihibit, Reno NV*, 2004, AIAA Paper 2004–0436.

[8]Fidkowski, K. J., Oliver, T. A., J.Lu, and Darmofal, D. L., "p-Multigrid Solution of High-Order Discontinuous Galerkin Discretizations of the Compressible Navier-Stokes Equations," submitted to J. Comput. Phys.

[9]Wang, L. and Mavriplis, D. J., "Implicit Solution of the Unsteady Euler Equations for High-Order Accurate Discontinuous Galerkin Discretizations," AIAA-Paper 2006-0109.

[10]Lottes, J. W. and Fischer, P. F., "Hybrid Multigrid/Schwarz Algorithms for the Spectral Element Method," *SIAM J. Sci. Comput.*, 2004, To appear.

[11]Nastase, C. R. and Mavriplis, D. J., "High-Order Discontinuous Galerkin Methods using an *hp*-Multigrid Approach," *J. Comput. Phys.*, Vol. 213, No. 1, 2006, pp. 330–357.

[12]Nastase, C. R. and Mavriplis, D. J., "Discontinuous Galerkin Methods Using an hp-Multigrid Solver for Inviscid Compressible Flows on Three-dimensional Unstructured Meshes," *Proceedings of the 44rd Aerospace Sciences Meeting and Exihibit, Reno NV*, 2006, AIAA Paper 2006–0107.

[13]Bassi, F. and Rebay, S., "High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations," *J. Comput. Phys.*, Vol. 138, 1997, pp. 251–285.

[14]Cockburn, B. and Shu, C.-W., "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems," *SIAM J. Numer. Anal.*, Vol. 35, No. 6, 1998, pp. 2440–2463.

[15]Warburton, T. C., Lomtev, I., Du, Y., Sherwin, S. J., and Karniadakis, G. E., "Galerkin and Discontinuous Galerkin Spectral/*hp* Methods," *Comput. Methods Appl. Mech. Engrg.*, Vol. 175, 1999, pp. 343–359.

[16]Cockburn, B. and Shu, C.-W., "Runge-Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems," *SIAM J. Sci. Comput.*, Vol. 16, No. 3, 2001, pp. 173–261.

[17]Davis, S. F., "Simplified Second-Order Godunov-Type Methods," *SIAM J. Sci. Statist. Comput.*, Vol. 9, No. 3, 1988, pp. 445–473.

[18]Roe, P. L., "Approximate Riemann Solvers, Parameter vectors, and Difference Schemes," *J. Comput. Phys.*, Vol. 43, 1981, pp. 357–372.

[19]Harten, A., Lax, P. D., and Van Leer, B., "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws," *SIAM Review*, Vol. 25, No. 1, 1983, pp. 35–61.

[20]Toro, F. E., *Riemann Solvers and Numerical Methods for Fluid Dynamics*, Applied Mechanics, Springer-Verlag, New York, NY, 1999.

[21]Batten, P., Clarke, N., Lambert, C., and Causon, D. M., "On the Choice of Wavespeeds for the HLLC Riemann Solver," *SIAM J. Sci. Comput.*, Vol. 18, No. 2, 1997, pp. 1553–1570.

[22]Batten, P., Leschiner, M. A., and Goldberg, U. C., "Average-State Jacobians and Implicit Methods for Compressible Viscous and Turbulent Flows," *J. Comput. Phys.*, Vol. 137, 1997, pp. 38–78.

[23]Solin, P., Segeth, P., and Zel, I., *High-Order Finite Element Methods*, Studies in Advanced Mathematics, Chapman and Hall, 2003.

[24]Szabo, B. and Babuska, I., *Finite Element Analysis*, John Wiley & Sons, Inc., New York, NY, 1991.

[25]Pirzadeh, S., "Three-Dimensional Unstructured Viscous Grids by the Advancing-Layers Method," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 43–49.

[26]Pirzadeh, S.

[27]Dunavant, D. A., "Highr Degree Efficient Symmetrical Gaussian Guadrature Rules for the Triangle," *Int. J. Numer. Meth. Engng.*, Vol. 21, 1985, pp. 1129–1148.

[28]Dunavant, D. A., "Economical Symmetrical Quadrature Rules for Complete Polynomials Over a Square Domain," *Int. J. Numer. Meth. Engng.*, Vol. 21, 1985, pp. 1777–1784.

[29]Cockburn, B., Hou, S., and Shu, C.-W., "The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case," *Math. Comput.*, Vol. 54, No. 545, 1990.

[30]Mavriplis, D. J., "An Assessment of Linear versus Non-Linear Multigrid Methods for Unstructured Mesh Solvers," *J. Comput. Phys.*, Vol. 175, 2002, pp. 302–325.

[31]Trottenberg, U., Schuller, A., and Oosterlee, C., *Multigrid*, Academic Press, London, UK, 2000.

[32]Mavriplis, D. J., "Multigrid Techniques for Unstructured Meshes," *VKI Lecture Series VKI-LS 1995-02*, mar 1995.

[33]Gropp, W., Lusk, E., and Skjellum, A., *Using MPI: Portable Parallel Programming with the Message Passing Interface*, MIT Press, Cambridge, MA, 1994.

[34]Karypis, G., "METIS, University of Minnesota, Department of Computer Science," http://www-users.cs.umn.edu/ karypis/metis, 2003.

[35]Biswas, R., Djomehri, M. J., Hood, R., Jin, H., Kiris, C., and Saini, S., "An Application-Based Performance Characterization of the Columbia Supercluster," Paper presented at the 2005 Supercomputing Conference, Seattle, WA.