Discontinuous Galerkin Methods Using an \(hp\)-Multigrid Solver for Inviscid Compressible Flows on Three-dimensional Unstructured Meshes

Cristian R. Nastase * and Dimitri J. Mavriplis †

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

The development of optimal, or near optimal solution strategies for higher-order discretizations, including steady-state solutions methodologies, and implicit time integration strategies, remains one of the key determining factors in devising higher-order methods which are not just competitive but superior to lower-order methods in overall accuracy and efficiency. The goal of this work is to investigate and develop a fast and robust algorithm for the solution of high-order accurate discontinuous Galerkin discretizations of non-linear systems of conservation laws on unstructured grids. Herein we extend our previous work to the three-dimensional steady-state Euler equations, by coupling the spectral \(p\)-multigrid approach with a more traditional agglomeration \(h\)-multigrid method for hybrid meshes, in a full-multigrid iteration strategy. In this \(hp\)-multigrid approach the coarse “grid” levels are constructed by reducing the order \((p)\) of approximation of the discretization using hierarchical basis functions \((p\text{-multigrid})\), together with the traditional \((h\text{-multigrid})\) approach of constructing coarser grids with fewer elements. The overall goal is the development of a solution algorithm which delivers convergence rates which are independent of “\(p\)” (the order of accuracy of the discretization) and independent of “\(h\)” (the degree of mesh resolution), while minimizing the cost of each iteration. The investigation of efficient smearers to be used at each level of the multigrid algorithm is also pursued, and comparisons between different integration strategies are made as well. Current three-dimensional results demonstrate convergence rates which are independent of both order of accuracy \((p)\) of the discretization and level of mesh resolution \((h)\).

I. Introduction

While most currently employed CFD algorithms are asymptotically second-order accurate in time and in space, the use of higher-order discretizations in both space and time offers a possible avenue for improving the predictive simulation capability for many applications. This is due to the fact that higher-order methods exhibit a faster asymptotic convergence rate in the discretization error than lower (second)-order methods. For example, with a fourth-order accurate spatial discretization, the error is reduced by a factor of \(2^4 = 16\) each time the mesh resolution is doubled, while a second-order accurate method only achieves a \(2^2 = 4\) reduction in error with each doubling of the mesh resolution. Since a doubling of mesh resolution in three dimensions entails an increase of overall work by a factor of \(2^3 = 8\), achieving an arbitrarily prescribed error tolerance with second-order accurate methods in three dimensions can quickly become unfeasible.

Thus, for increasingly high accuracy levels, higher-order methods ultimately become the method of choice. Therefore, the expectation is that an efficient higher-order discretization may provide an alternate path for achieving high accuracy in a flow with a wide disparity of length scales at reduced cost, by avoiding the use of excessive grid resolution.

On the other hand, for levels of accuracy often associated with mean-flow engineering calculations, higher-order methods have proved to be excessively costly compared to simpler second-order accurate methods. Clearly, because of the different asymptotic nature of these methods, the cost comparison between methods is a strong function of the required levels of accuracy. Nevertheless, for many engineering type calculations, higher-order methods have been found to be non-competitive compared to the simpler second-order accurate methods.

*Postdoctoral Research Associate, AIAA Member; email: nastase@uwyo.edu.
†Professor, AIAA Associate Fellow; email: mavripl@uwyo.edu.
Copyright © 2006 by Cristian R. Nastase and Dimitri J. Mavriplis. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.
While the formulation of discretization strategies for higher-order methods such as Discontinuous Galerkin [1–6] and Streamwise Upwind Petrov-Galerkin [7] methods are now fairly well understood, the development of techniques for efficiently solving the discrete equations arising from these methods has generally been lagging. This is partly due to the complex structure of the discrete equations originating from fairly sophisticated discretization strategies, as well as the current application of higher-order methods to problems where simple explicit time-stepping schemes are thought to be adequate solution mechanisms, due to the close matching of spatial and temporal scales, such as acoustic phenomena.

The development of optimal, or near optimal solution strategies for higher-order discretizations, including steady-state solutions methodologies, and implicit time integration strategies, remains one of the key determining factors in devising higher-order methods which are not just competitive but superior to lower-order methods in overall accuracy and efficiency.

Previous work by the second author [8] has examined the use of spectral multigrid methods, where convergence acceleration is achieved through the use of coarse levels constructed by reducing the order \( p \) of approximation of the discretization (as opposed to coarsening the mesh) for Discontinuous Galerkin discretizations. The idea of spectral multigrid was originally proposed by Ronquist and Patera [9], and has been pursued for the Euler and Navier-Stokes equations by Fidkowski et al., [10, 11] with encouraging results. Implicit multi-level solution techniques for high-order discretizations have also been developed by Lottes and Fisher [12].

In this work, we extend our recent two-dimensional \( hp \)-multigrid approach described in Ref. [13] to the three-dimensional steady-state Euler equations, by coupling the spectral \( p \)-multigrid approach with a more traditional agglomeration \( h \)-multigrid method for hybrid meshes, in a full-multigrid iteration strategy. The investigation of efficient smoothers to be used at each level of the multigrid algorithm is also pursued, and comparisons between different integration strategies are made as well. The overall goal is the development of a solution algorithm which delivers convergence rates which are independent of “\( p \)” (the order of accuracy of the discretization) and independent of “\( h \)” (the degree of mesh resolution), while minimizing the cost of each iteration.

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

\[
\frac{\partial U(x,t)}{\partial t} + \nabla \cdot F(U) = 0
\]  

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

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

where \( \rho \) 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:

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

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

III. Spatial Discretization

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

\[
U(x,t) \approx U_p(x,t) = \sum_{j=1}^{M} u_j(t) \phi_j(x)
\]
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 [2–4, 6] in spatial variables within each element $\Omega_k$. The weak formulation for Eq. (1) is obtained by minimizing the residual with respect to the expansion function in an integral sense:

$$
\int_{\Omega_k} \phi_i \left[ \frac{\partial U_p(x,t)}{\partial t} + \nabla \cdot F(U_p) \right] \, d\Omega_k = 0
$$

(5)

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

$$
\int_{\Omega_k} \phi_i \frac{\partial U_p}{\partial t} \, d\Omega_k - \int_{\Omega_k} \nabla \phi_i \cdot F(U_p) \, d\Omega_k + \int_{\partial\Omega_k} \phi_i F^e(U_p) \cdot n \, d(\partial\Omega_k) = 0
$$

(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. The numerical flux, $F^e(U_p) \cdot n$, is obtained as a solution of a local one-dimensional Riemann problem and depends on the internal interface state, $U_p^-$, the adjacent element interface state, $U_p^+$, and the orientation as defined by the normal vector, $n$, 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 [14], Roe [15], HLL [16] and HLLC [17–19].

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 $\hat{\Omega}_j(\xi_j, j = 1 \ldots 3)$ 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 [20, 21] based on Jacobi polynomials of variable weights. Since the basis set is defined in the master element, a coordinate transformation, $x_p = x_p(\xi_1, \xi_2, \xi_3)$, 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 $\xi_1$, $\xi_2$ and $\xi_3$, and the coordinate transformation, and its Jacobian are given by:

$$
x_p = \sum_{j=1}^{M} \hat{x}_j \phi_j(\xi_1, \xi_2, \xi_3), \quad J_k(\xi_1, \xi_2, \xi_3) = \left| \frac{\partial(x,y,z)}{\partial(\xi_1, \xi_2, \xi_3)} \right|
$$

(7)

Complicated geometries require the use of curved boundaries, especially in the case of high order methods where elements span a larger portion of boundaries than the case of low order methods. Therefore, in order to maintain the desired $p + 1$ global accuracy order, we make use of curved-sided elements for (wall) boundary elements only, and straight-sided elements for all other (interior) elements. 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, additional information must be provided in order to determine the coordinate expansion coefficients, $\hat{x}_j$. Herein, we outline a simple procedure to obtain the curved element list and additional face points for a three-dimensional surface, given that the surface has an analytical expression. The first step is to find the elements with one face on the surface. This will be elements with all faces curved. In this case, the additional points on the face obtained, via the surface function, will determine all the expansion coefficients within the element. The second step is to propagate the cell coefficients back to all elemental faces curved. In this case, the additional points on the face obtained, via the surface function, will determine all the expansion coefficients within the element. 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, $\hat{\Omega}_k$, becomes:

$$
U_p(\xi, \eta, t) = \sum_{j=1}^{M} \hat{U}_j(t) \phi_j(\xi_1, \xi_2, \xi_3)
$$

(8)

$$
\int_{\hat{\Omega}_k} \phi_i \frac{\partial U_p}{\partial t} |J_k| \, d\hat{\Omega}_k - \int_{\hat{\Omega}_k} \nabla \phi_i J_k^{-1} \cdot F(U_p) |J_k| \, d\hat{\Omega}_k + \int_{\partial\hat{\Omega}_k} \phi_i F^e(U_p) \cdot n |J_k| |\partial\hat{\Omega}_k| = 0
$$

(9)

This set of equations is solved in the modal space and the integrals are evaluated by economical Gaussian quadrature rules [20, 22, 23], which requires a projection of the solution values to the quadrature points used in the numerical integration. In order to preserve $p + 1$ 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 \(2p + 1\) [24]. 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. Figure (1) depicts the quadrature points for \(p = 3\) within the tetrahedron element and its face.

![Tetrahedral master element](image)

![Triangular master element](image)

**Figure 1. Interior and face quadrature points on master element for \(p = 3\) (○ edge, ● interior).**

**IV. The Implicit Steady State Solver**

Neglecting the temporal derivative term, the system of equations (Eq. (9)) associated with each element becomes:

\[
R(U_p) = S_p
\]

where \(R(U_p)\) is the non-linear residual and \(S_p\) is the source term. Although in the case of the Euler equations \(S_p = 0\), the use of a source term will facilitate the introduction of the multigrid algorithm in the next section. We use variants of an element-Jacobi scheme to solve this system of equations. The Newton iteration associated with Eq. (10) yields at each “\(n + 1\)” step:

\[
\begin{align*}
\left[ \frac{\partial R}{\partial U_p} \right]^n U_p^{n+1} & = S_p - R(U_p^n) \\
U_p^{n+1} & = U_p^n + \alpha \Delta U_p^{n+1}
\end{align*}
\]

where \(\alpha\) is a parameter used for robustness to keep \(\|\alpha \Delta U_p^{n+1} / U_p^{n+1}\|_{L_{\infty}} \leq 10\%\). The element-Jacobi scheme can be viewed 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, \(\left[ \frac{\partial R}{\partial U_p} \right] = [D]\), 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 is reasonable for aerodynamic applications. The non-linear iteration Eq. (11) becomes:

\[
\Delta U_p^{n+1} = [D^n]^{-1}(S_p - R(U_p^n))
\]

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, \(R(U_p^{(k)}),\) is updated,
and the block diagonal matrices, $[D^p]$, are kept constant from the outer-iteration “$n$”. Therefore, the $(k+1)^{th}$ step is:

$$
\Delta U^{k+1}_p = [D^p]^{-1}(S_p - R(U^k_p))
$$

(13)

This approach is expected to yield similar converge rates per cycle as in the NEJ variant, with improved performance in terms of CPU time. 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).

<table>
<thead>
<tr>
<th>$p$</th>
<th>Size of $[D]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$5 \times 5$</td>
</tr>
<tr>
<td>1</td>
<td>$20 \times 20$</td>
</tr>
<tr>
<td>2</td>
<td>$50 \times 50$</td>
</tr>
<tr>
<td>3</td>
<td>$100 \times 100$</td>
</tr>
<tr>
<td>4</td>
<td>$175 \times 175$</td>
</tr>
<tr>
<td>5</td>
<td>$280 \times 280$</td>
</tr>
<tr>
<td>6</td>
<td>$420 \times 420$</td>
</tr>
</tbody>
</table>

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:

$$
\left[ \frac{\partial R}{\partial U_p} \right]^n = [D^p] + [O^p]
$$

(14)

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. (11). In matrix form the $(k+1)^{th}$ step of the linearized element Jacobi step is written as:

$$
\Delta U^{k+1}_p = [D^p]^{-1}\left(S_p - R(U^n_p) - [O^p]\Delta U^k_p\right)
$$

(15)

Note that the linearized element Jacobi scheme 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 $R(U^n_p)$ and the Jacobian entries $[D^p]$ and $[O^p]$ 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. (12)), both approaches can be expected to converge at the same rates per cycle (asymptotically) [25]. 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. (15) 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:

$$
\Delta U^{k+1}_p = [(D + L)^n]^{-1}\left(S_p - R(U^n_p) - [U^n]\Delta U^k_p\right)
$$

(16)

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. In this work, we employ a frontal sweep along the elements which begins near the inner boundary and proceeds toward the outer boundary, using the numbering assigned to the grid elements from an advancing front mesh generation technique [26]. All the simulation results shown here are performed using the HLLC flux only.

V. Single Grid Results

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. Figure (2) shows the mesh configuration and the Mach contour lines for a free-stream Mach number of \( M_{\infty} = 0.5 \).

![Mach number contours for the three-dimensional bump](image)

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

**Figure 2.** The Mach number contours for the three-dimensional bump.

The initial conditions are set to free-stream values. 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 \( y = 0 \). For this particular case, the grid consists of \( N = 10349 \) tetrahedral elements, and the discretization order is \( p = 4 \) (i.e. fifth-order accurate). The discretization error can be assessed by measuring the norm of the entropy error for this isentropic flow. The entropy error is defined as \( ds = s - s_\infty \), where \( s_\infty \) is the free stream entropy.

Figure (3(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 (3(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 described in the next section. 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.

Figure (4(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. 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 (4(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 than the element-Jacobi scheme. The savings are substantial due to the fact that these non-linear evaluations include the expensive quadrature integration procedures. The quasi non-linear element Jacobi proves to be an appropriate compromise for three-dimensional cases where memory limitations are dominant (i.e only storage of the diagonal blocks, \( [D] \), is required).

Due to memory limitations the remaining results will make exclusive use of the quasi non-linear element Jacobi (qNJ) scheme. Figure (5(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
Figure 3. The $L_1$ norm of the entropy error as a function of: (a) $h/p$-refinement; (b) CPU time.

(a)

(b)

Figure 4. 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.

(a)

(b)
the discretization for a fixed size grid. However, increasing the number of elements, \( N \), has an adverse effect on the convergence rate. In Figure (5(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 an \( hp \)-multigrid scheme.

![Graphs showing convergence rate deterioration](image)

**Figure 5.** 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 \( p = 4 \)

### VI. The \( hp \)-Multigrid Approach

#### A. General Description

Multigrid methods are known as efficient techniques for accelerating convergence to steady state for both linear and non-linear problems [25, 27], 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 [25]. 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 [10]. 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 [27]. 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 convergence rates [28]. This procedure has the potential of resulting in a truly $h$- and $p$-independent solution strategy for high-order accurate discontinuous Galerkin discretizations. Figure (6) illustrates a typical $h$-multigrid level configuration on structured grids.

![Figure 6. Typical $h$-multigrid level configuration.](image)

Based on our experience with the linearized element-Jacobi solver, we also consider two ways of applying multigrid to the non-linear Euler equations. The first is to apply multigrid directly to the non-linear problem (Eq. (10)) via the Full Approximation Storage (FAS) scheme. In a two-level $p$-multigrid method this scheme is given as:

- Iterate the fine-grid level problem and its residual, $r_p$, using any of the element-Jacobi variants aforementioned:
  \[
  R_p(U^n_p) = S_p, \quad r^n_p = S_p - R_p(U^n_p)
  \] (17)

- Obtain the source term for the coarse (i.e. $p - 1$) level by restricting both the solution and the residual:
  \[
  S_{p-1} = I^{-1}_{p} r^n_p, \quad U^n_{p-1} = \tilde{I}_{p}^{-1} U^n_p
  \] (18)

- Solve the coarse grid level problem
  \[
  R_{p-1}(U^n_{p-1}) = S_{p-1}
  \] (19)

- Calculate the coarse grid error, $e^n_{p-1}$:
  \[
  e^n_{p-1} = U^n_{p-1} - \tilde{I}^{-1}_{p} U^n_p
  \] (20)

- Prolongate the coarse grid error and correct the fine-grid level approximation:
  \[
  U^{n+1}_p = U^n_p + I^{p-1}_{p} e^n_{p-1}
  \] (21)
In the case of $p$-multigrid, $\tilde{I}_p^{-1}$ and $I_p^{p-1}$ denote the state and residual restriction (i.e., from $p$ to $p-1$) operators, respectively. In the case of a hierarchical basis, $I_p^{p-1}$ is the identity matrix with zero columns appended. Moreover, $I_p^{-1} = \tilde{I}_p^{-1}$ for $p$-multigrid but note that this is not true in the case of $h$-multigrid. Similarly, the prolongation (i.e., from $p-1$ to $p$) operator, $I_{p-1}^{p}$, is obtained as the transpose of the restriction operator, $I_p^{p-1} = (I_p^{p-1})^T$.

The second way of applying multigrid to the non-linear set of governing equations is to use the Coarse Grid Correction (CGC) multigrid technique on the linearized problem obtained at each Newton iteration (Eq. (27)). This methodology, sometimes referred as “Newton-multigrid”, is given (using the dual iteration strategy) as follows:

- Outer non-linear ($n^{th}$) iteration. Iterate the discrete linear problem using any of the linearized element-Jacobi variants (LEJ or LGS) aforementioned:

$$\left[ \frac{\partial R_p}{\partial U_p} \right]^n \Delta U_p^{n+1} = S_p - R_p(U_p^n)$$  \hspace{1cm} (22)

- Inner linear ($k^{th}$) iteration. Solve for the fine-grid level correction $w^k_p = \Delta U_p^k$ with initial guess $w^0_p = 0$:

$$\left[ J_p^k \right] w^k_p = f^k_p$$  \hspace{1cm} (23)

where

$$\left[ J_p^k \right] = \left[ \frac{\partial R_p}{\partial U_p} \right]^n, \hspace{0.5cm} f^k_p = S_p - R_p(U_p^n)$$  \hspace{1cm} (24)

- Obtain the source term for the coarse level by restricting the linear residual $r^k_p$:

$$f_{p-1}^k = I_{p-1}^{p} r^k_p, \hspace{0.5cm} r^k_p = f^k_p - [J_p^k] w^k_p$$  \hspace{1cm} (25)

- Solve the coarse grid correction problem with initial guess $\Delta w_{p-1}^k = 0$:

$$\left[ J_{p-1}^k \right] \Delta w_{p-1}^k = f_{p-1}^k, \hspace{0.5cm} \{J_{p-1}^k\} \text{ is a subset of } \{J_p^k\}.$$  \hspace{1cm} (26)

- Prolongate the coarse grid correction and update the fine-grid correction:

$$\Delta U_p^{k+1} = \Delta U_p^k + I_{p-1}^{p} \Delta w_{p-1}^k$$  \hspace{1cm} (27)

- Fine-grid non-linear update:

$$U_p^{n+1} = U_p^n + \Delta U_p^{k+1}$$  \hspace{1cm} (28)

In this implementation the basis set is hierarchical beginning at $p = 1$. Therefore, the Jacobian, $\left[ J_{p-1}^k \right] = \left[ \frac{\partial R_p(U_p)}{\partial U_p} \right]^n$, (and its inverse) represents a subset of $\left[ J_p^k \right]$ and require no additional operator for its construction. Once the $p = 1$ level is reached the state variable, $U_p$, and the residual, $R_p(U_p)$, are restricted to $p = 0$, via two different operators defined as follows:

$$U_h = \frac{1}{3} \sum_{i=1}^{3} U_{p=1}^{i}; \hspace{0.5cm} R_h(U_h) = \sum_{i=1}^{3} R_p(U_{p=1}^{i})$$  \hspace{1cm} (29)

where $\{i = 1 \ldots 3\}$ is the modal index corresponding to $p = 1$. Therefore, in the case of $h$-multigrid (i.e., $p = 0$), the fine grid problem becomes $R_h(U_h) = S_h$, and both FAS and CGC algorithms are obtained in a similar fashion, with the exception of the $\left[ J_h^k \right] = \left[ J_{p=0}^k \right]$ term, in the case of CGC algorithm, which needs to be evaluated once at every non-linear $n^{th}$ step for all $h$-levels as $\left[ J_h^k \right] = \left[ \frac{\partial R_h(U_h)}{\partial U_h} \right]^n$, where the restriction of the state variable and its residual to a coarse level, $H$, is obtained as:

$$U_H = \frac{1}{A_H} \sum_{k=1}^{N_h} (U_h^k A_h^k), \hspace{0.5cm} R_H(U_H) = \sum_{k=1}^{N_h} (R_h(U_h^k))$$  \hspace{1cm} (30)

where $N_h$ is the number of elements used in the agglomeration, $A_h$ is the fine level elemental area, and $A_H = \sum_{k=1}^{N_h} (A_h^k)$ is the coarse level area. This two-level multigrid can be easily extended to a multi-level scheme.
For robustness it is important to augment the resulting multi-level $hp$-multigrid with a full multigrid (FMG) technique, in order to provide a good initial guess for the fine level problem. Moreover, the use of FMG is critically important in the case of the CGC scheme for it is known that the Newton iteration will diverge if the initial guess is not close enough to the final solution. In our $hp$-multigrid approach, the solution process begins at the coarsest grid level ($p = 0$), using all the $h$-levels available, and ends at the fine level where all the $p$- and $h$-levels are used to advance to solution to the desired accuracy, as depicted in Figure (7). Alternatively, the FMG strategy can be initiated at the coarsest $h$-level, but no advantage over the latter approach was found, at least for the inviscid grids/problem considered. This will be further investigated in a future work pertaining viscous flows.

B. Flow over a three-dimensional bump

In the context of multigrid methodology, the compressible flow over a three dimensional bump is considered, with geometrical parameters, boundary and initial conditions as defined in Section (V). Results are presented for the FAS multigrid algorithm only. Unless otherwise stated, all the simulated results are obtained via FMG using three V-cycles per level, starting at $p = 0$ level. Figure (8) illustrates a three dimensional view of a typical two level $h$-multigrid agglomerated (AMG) configuration.

Multigrid strategy approach

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 4$p$-levels are considered from $p = 4$ to $p = 1$, using 10 iterations per level.
- **RUN2**, where 4$p$-levels and 1$h$-level ($p = 0$) are considered, using 10 iterations per level.
- **RUN3**, where 4$p$-levels and 1$h$-level ($p = 0$) are considered, using 10 iterations per $p$-level and the $h$-level is converged to machine precision.
- **RUN4**, where 4$p$-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.

Figure (9(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 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 only 3 agglomerated $h$-levels, displays the same convergence rate (i.e. slope) as RUN3, but requires only 10 iterations per level. The shift in RUN3 vs. RUN4 is due to the fact that, in the former, the first FMG-level (i.e. $p = 0$) is solved exactly while in the latter only 3 cycles are performed. As expected, in terms of CPU-time, the RUN4 case outperforms all other cases (Figure (9(b))). This behavior is maintained as well for higher orders (not shown). Therefore, the RUN3 and RUN4 are the multigrid strategy of choice.
Figure 8. A typical two level $h$-multigrid mesh configuration.

Figure 9. 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.
The $h$- and $p$-dependence

Figure (10) illustrates the convergence rate for a fixed mesh size of $N = 10349$ elements and various orders ($p$), for all multigrid strategies considered. The first 12 iterations actually correspond to the initial FMG process. While the RUN1 case, Figure (10(a)), and RUN2, Figure (10(b)), do not scale with the order $p$, the RUN3 case, Figure (10(c)), and RUN4 case, Figure (10(d)), display a $p$-independent convergence rate. This is due to the fact that the coarsest level is solved exactly. Also, this behavior is observed for all other discretization orders (not shown). It is interesting to observe that for $p = 4$ all four strategies converge in 25–30 cycles. This might not be the general case but rather particular to the mesh size considered. Therefore, the $h$-dependence is considered in Figure (11) for a fixed order $p = 4$, at all four cases. Clearly, a simple $p$-multigrid, Figure (11(a)), strategy renders the convergence $h$-dependent. The RUN2 case, Figure (11(a)), RUN3 case, Figure (11(c)), and RUN4 case, Figure (11(c)), show almost $h$-independent convergence rates. This behavior is to be expected for RUN3 and RUN4 and not for RUN2, where only one $p = 0$ level is used. It turns out that this is a fortunate case but for all other orders, as expected, the $h$-independence is not obtained, as illustrated in Figure (12). In conclusion, the fastest convergence rate is obtained in terms of both number of multigrid cycles and CPU time by using $p$-multigrid together with $h$-multigrid (RUN4). Figure (13) shows the residual convergence vs. CPU time for a mesh size of $N = 10349$ elements and $p = 4$ order.

VII. 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. Non-linear element-Jacobi, as well as linearized element-Jacobi and Gauss-Seidel schemes are used as smoothers on each level of the multigrid sequence. The linearized smoothers require additional storage, but are generally more efficient than their non-linear counterparts. The $hp$-multigrid scheme demonstrates both $p$-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 the benefits of the $hp$-multigrid approach can be expected to increase for finer meshes. Future work will concentrate on extending these techniques to the Navier-Stokes equations using hybrid element topologies. In addition, the efficiency of these solvers will be addressed in the context of MPI parallel implementation.

Acknowledgments

This work was supported by a grant from the Office of Naval Research ONR grant number N00014-04-1-0602.

References


Figure 10. The $L_2$ norm of the residual vs. the number of qNJ cycles, for a fixed mesh size of $N = 10349$ elements and various orders ($p$): (a) RUN1; (b) RUN2; (c) RUN3; (d) RUN4;
Figure 11. The $L_2$ norm of the residual vs. the number of qNJ cycles, for a fixed $p = 4$ order and various fine grid mesh sizes: (a) RUN1; (b) RUN2; (c) RUN3; (d) RUN4;
Figure 12. The $L_2$ norm of the residual vs. the number of qNJ cycles, for a fixed mesh size of $N = 10349$ elements and a fixed $p = 3$ order.

Figure 13. The $L_2$ norm of the residual vs. CPU time, for a fixed mesh size of $N = 10349$ elements and a fixed $p = 4$ order.


