10th World Congress on Computational Mechanics
8-13 July 2012 • São Paulo • Brazil

OUTPUT-BASED *HP*-ADAPTATION APPLIED TO AERODYNAMIC FLOWS

Marco Ceze[1], Krzysztof Fidkowski[1]

[1] Department of Aerospace Engineering, University of Michigan
(mceze@umich.edu, kfid@umich.edu)

Abstract. *We present a method for concurrent mesh and polynomial-order adaptation with the objective of direct minimization of output error using a selection process for choosing the optimal refinement option from a discrete set of choices that includes directional spatial resolution and approximation order increment. The scheme is geared towards compressible viscous aerodynamic flows, in which solution features make certain refinement options more efficient compared to others. No attempt is made, however, to measure the solution anisotropy or smoothness directly or to incorporate it into the scheme. Rather, mesh anisotropy and approximation order distribution arise naturally from the optimization of a merit function that incorporates both an output sensitivity and a measure of the computational cost of solving on the new mesh. An adjoint state is used to translate the residual perturbation resulting from each refinement option into an output sensitivity with respect to each mesh modification option. Two measures of computational cost are explored: a generic measure that accounts for the number of degrees of freedom of the discrete state, and one that accounts for the number of floating-point operations involved in solving the discrete problem. We restrict the mesh refinement mechanics to quadrilateral and hexahedral meshes. Many such meshes and associated meshing programs exist from the structured CFD community, and these can be leveraged to produce the starting meshes for the proposed adaptation. Additionally, we discuss implementation challenges of hp-adaptive methods for aerodynamic problems, such as load balancing on distributed-memory systems. The method is applied to output-based adaptive simulations of laminar and Reynolds-averaged compressible Navier-Stokes equations on body-fitted meshes in two and three dimensions. Two-dimensional results show significant reduction in the degrees of freedom and computational time to achieve output convergence when the discrete choice optimization is used compared to uniform h or p adaptation. Three-dimensional results show that the presented method is an affordable way of achieving output convergence on notoriously difficult cases such as the third Drag Prediction Workshop W1 configuration.*

Keywords: *hp-adaptation, computational fluid dynamics, discontinuous Galerkin.*

1. INTRODUCTION

Aerodynamic flows exhibit features in a wide range of length scales and singular features whose distributions are not known *a priori*. Hence, it is key for an efficient computation that the adaptive algorithm is capable of generating stretched elements in areas where the solution exhibits anisotropy and choosing a local approximation order appropriate to the smoothness of the solution. Anisotropic features include boundary layers, wakes, and shocks, where the disparity of length scales is such that stretching ratios in the hundreds or thousands are common. Singular or near-singular features, at least for the primal solution, include shocks, trailing edges, edges of boundary layers and wakes, and trailing vortices,

The choice between subdividing an element or locally changing the scheme's discretization order is not trivial and has been the subject of much previous research [3, 22, 34, 24, 16, 4]. Bey [3] uses the error equidistribution principle to first subdivide elements and then increase the polynomial order where the solution is deemed smooth. Conversely, Heuveline and Rannacher [22] propose a process that prioritizes p-refinement and only subdivides an element when the previous step leads to an increase in the elemental error indicator. Houston and Süli [24] introduced two methods for assessing the local smoothness of the solution using Legendre series expansions and estimates of the local Sobolev index. In that same article, they also provide an overview of different strategies for the decision of the refinement type. Burgess and Mavriplis [4] use a solution-jump indicator to decide between h and p refinements. Following a different approach, Rachowicz *et al* [34] choose h or p refinement based on an estimated lowest interpolation error.

Anisotropic mesh adaptation in aerospace applications is also a prolific research topic. The dominant method for detecting anisotropy has relied on estimates of the directional interpolation error of a representative scalar, such as the Mach number [32, 28]. When used alone, this technique reduces to equidistributing the interpolation error of the chosen scalar over the computational domain, with the absolute level of interpolation error prescribed by the user [5, 19]. Alternately, this technique can be combined with output-based error estimation by using the output adaptive indicator to set the element size and the directional interpolation error to set the element stretching [38, 40]. The same idea can be extended to high-order discretizations [11, 9], although the measurement of directional interpolation error becomes more tedious. A more fundamental problem with this approach in the context of output-based adaptation is the assumption that mesh anisotropy should be governed by the directional interpolation error of one scalar quantity. This assumption is heuristic because it does not take into account the process by which interpolation errors create residuals that affect the output of interest. As a result, recent research has turned to adaptation algorithms that directly target the output error.

Formaggia *et al* [15, 14, 13, 29] combine Hessian-based interpolation error estimates with output-based *a posteriori* error analysis to arrive at an output-based error indicator that explicitly includes the anisotropy of each element. Schneider and Jimack [36] calculate the sensitivities of the output error estimate with respect to node positions and formulate an optimization problem to reduce the output error estimate by redistributing the nodes. They then combine this node repositioning with isotropic local mesh refinement sequentially in a hybrid optimization/adaptation algorithm. Park [31] introduces an algorithm that directly targets the output error through local mesh operators of element swapping, node movement, element collapse, and element splitting. Using the output error indicator to rank elements and nodes, these operations are performed in sequence and automatically result in mesh anisotropy.

Following a similar approach presented by Houston *et al* [23], we proposed in Ref. [6] a direct mesh optimization technique in which a particular mesh refinement is chosen from a discrete number of possible choices in a manner that directly targets reduction of the output error. That strategy is specifically suited for hanging-node meshes, in which a handful of refinement options is typically available for each element and in which the adaptation mechanics are relatively simple. We extended our previous work to hp-adaptation of quadrilateral and hexahedral meshes in Ref. [8].

In order to fully use the potential of hp-adaptive methods in practical aerodynamic flows, the computational resources must be used efficiently. We present in this paper a method for balancing the computational work in distributed-memory systems that aims to equalize the number of floating point operations of each processor and improve the effectivity of the preconditioner.

2. PROBLEM STATEMENT

Let $\mathbf{U}$ denote a discrete state vector and consider the semi-discrete system,

$$\mathbf{U}_t = -\mathbf{R}(\mathbf{U}), \tag{1}$$

where $\mathbf{R}(\mathbf{U})$ is a discrete residual operator derived from a weighted-residual statement of a set of partial differential equations. In this work, we consider laminar and Reynolds-averaged compressible Navier-Stokes equations [30].

The field representation of the state is given by an expansion in terms of basis functions $\phi_j^{H,p}(x) \in \mathcal{V}^{H,p}$,

$$\mathbf{u}^{H,p}(t,x) = \sum_j \mathbf{U}_j(t)\phi_j^{H,p}(x), \tag{2}$$

where $\mathcal{V}^{H,p}$ is the space of polynomials of order p with local support on each of the elements κ^H in a non-overlapping discretization T^H of the domain D.

We are interested in the steady-state solution of the flow equations, therefore high-accuracy is not required for discretizing the unsteady term in Eqn. 1. Instead, stability is the main attribute which makes backward Euler an attractive choice. The fully discrete form of Eqn. 1 is then:

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

where $\mathcal{M}$ is a block-diagonal mass matrix for discontinuous Galerkin discretizations.

In time-accurate calculations, Eqn. 3 is solved for the future state using a nonlinear solver such as Newton-Raphson. For steady calculations, the residual at the future state in Eqn. 3 is expanded about the current state and the steps in the iterative procedure require linear solves for the update $\Delta\mathbf{U}^k = \mathbf{U}^{k+1} - \mathbf{U}^k$,

$$\left(\mathcal{M}\frac{1}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\bigg|_{\mathbf{U}^k}\right)\Delta\mathbf{U}^k = -\mathbf{R}(\mathbf{U}^k), \tag{4}$$

where k is used for the linear iteration number to distinguish the method from the time-accurate backward Euler case.

In principle, Newton's root-finding method could be used to solve $\mathbf{R}(\mathbf{U}) = 0$ directly. However, the unsteady term in Eqn. 1 is kept to alleviate the spectral conditioning of the linear systems in the initial stages of the calculation and to improve the global convergence property of the solver.

The linearization of the residual operator involves simplifications due to non-differentiable terms in numerical flux functions and artificial dissipation sensors. Additionally, the sparse structure of the linear system given in Eqn. 4 depends on the type of spatial scheme used for $\mathbf{R}$, and an appropriate choice of iterative solver and preconditioner must be made. In this work, we use the Generalized Minimal Residual (GMRES) algorithm with a line-Jacobi preconditioner [10]. Note that for $\Delta t \to \infty$ the iterative procedure of Eqn. 4 reduces to Newton's root-finding method.

3. OUTPUT ERROR ESTIMATION

Output-based error estimation techniques identify all areas of the domain that are important for the accurate prediction of an engineering output. The resulting estimates properly account for error propagation effects that are inherent to hyperbolic problems, and they can be used to ascribe confidence levels to outputs or to drive adaptation. A key component of output error estimation is the solution of an adjoint equation for the output of interest. In a continuous setting, an adjoint, $\boldsymbol{\psi} \in \mathcal{V}$, is a Green's function that relates residual source perturbations to a scalar output of interest, $J(\mathbf{u})$, where $\mathbf{u} \in \mathcal{V}$ denotes the state, and where $\mathcal{V}$ is an appropriate function space. Specifically, given a variational formulation of a partial differential equation: determine $\mathbf{u} \in \mathcal{V}$ such that

$$\mathbb{R}(\mathbf{u}, \mathbf{w}) = 0, \qquad \forall \mathbf{w} \in \mathcal{V}, \tag{5}$$

the adjoint $\boldsymbol{\psi} \in \mathcal{V}$ is the sensitivity of J to an infinitesimal source term $\delta \mathbf{r} \in \mathcal{V}$ added to the left-hand side of the original PDE. $\boldsymbol{\psi}$ satisfies a linear equation,

$$\mathbb{R}'[\mathbf{u}](\mathbf{w}, \boldsymbol{\psi}) + J'[\mathbf{u}](\mathbf{w}) = 0, \qquad \forall \mathbf{w} \in \mathcal{V}, \tag{6}$$

where the primes denote Fréchét linearization with respect to the arguments in square brackets. Details on the derivation of the adjoint equation can be found in many sources, including the review in Ref. [12]. Specifically, in the present work we employ the discrete adjoint method, in which the system is derived systematically from the primal system [18, 27].

An adjoint solution can be used to estimate the numerical error in the corresponding output of interest. The resulting adjoint-weighted residual method is based on the observation that a solution $\mathbf{u}^{H,p}$ in a finite-dimensional approximation space $\mathcal{V}^{H,p}$ will generally not satisfy the original PDE. The adjoint $\boldsymbol{\psi} \in \mathcal{V}$ translates the residual perturbation to an output perturbation via,

$$\delta J = J(\mathbf{u}^{H,p}) - J(\mathbf{u}) \approx -\mathbb{R}(\mathbf{u}^{H,p}, \boldsymbol{\psi}). \tag{7}$$

This expression is based on a linear analysis, and hence for nonlinear problems and finite-size perturbations, the result is approximate.

Although the continuous solution $\mathbf{u}$ is not required directly, the continuous adjoint $\boldsymbol{\psi}$ must be approximated to make the error estimate in Eqn. 7 computable. In practice, $\boldsymbol{\psi}^{h,p^+}$ is solved approximately or exactly on a finer finite-dimensional space $\mathcal{V}^{h,p^+} \supset \mathcal{V}^{H,p}$ [35, 1, 37]. This finer space can be obtained either through mesh subdivision or approximation order increase [26, 20, 30] – denoted here by changes in the superscript H and p, respectively.

The adjoint-weighted residual evaluation in Eqn. 7 can be localized to yield an adaptive indicator consisting of the relative contribution of each element to the total output error. In this work, the finer space is obtained by approximation order increment, $\mathcal{V}^{H,p+1} \supset \mathcal{V}^{H,p}$, and $\boldsymbol{\psi}^{H,p+1}$ is approximated by injecting $\boldsymbol{\psi}^{H,p}$ into $\mathcal{V}^{H,p+1}$ and applying 5 element block-Jacobi smoothing iterations. The output perturbation in Eqn. 7 is approximated as

$$\delta J \approx -\sum_{\kappa^H \in T^H} \mathbb{R}_{\kappa^H}(\mathbf{I}_{H,p}^{H,p+1}(\mathbf{u}^{H,p}), \boldsymbol{\psi}^{H,p+1} - \mathbf{I}_{H,p}^{H,p+1}(\boldsymbol{\psi}^{H,p})), \tag{8}$$

where $\mathbf{I}_{H,p}^{H,p+1}(\cdot)$ is an injection operator from p to $p+1$ in the coarse mesh T^H, and $\mathbb{R}_{\kappa^H}$ corresponds to the residual of element κ^H. Note, the difference between the coarse-space and fine-space adjoints is not strictly necessary due to Galerkin orthogonality [12]. However, when the primal residual is not fully-converged to machine precision levels the use of the adjoint perturbation gives better error estimates. Equation 8 expresses the output error in terms of contributions from each coarse element. A common approach for obtaining an adaptive indicator is to take the absolute value of the elemental contribution in Eqn. 8 [39, 2, 21, 17, 1, 6],

$$\eta_{\kappa^H} = \left| \mathbb{R}_{\kappa^H}(\mathbf{I}_{H,p}^{H,p+1}(\mathbf{u}^{H,p}), \boldsymbol{\psi}^{H,p+1} - \mathbf{I}_{H,p}^{H,p+1}(\boldsymbol{\psi}^{H,p})) \right|. \tag{9}$$

With systems of equations, indicators are computed separately for each equation and summed together. Due to the absolute values, the sum of the indicators, $\sum_{\kappa^H} \eta_{\kappa^H}$, is greater or equal to the original output error estimate. However, it is not a bound on the actual error because of the approximations made in the derivation.

4. MESH ADAPTATION MECHANICS

The elemental adaptive indicator, η_{κ^H}, drives a fixed-fraction hanging-node adaptation strategy. In this strategy, which was chosen for simplicity and predictability of the adaptive algorithm, a certain fraction, f^{adapt}, of the elements with the largest values of η_{κ^H} is marked for refinement. Marked elements are refined according to discrete options which correspond to subdividing the element in different directions or increasing the approximation order. For quadrilaterals, the discrete options are: x-refinement, y-refinement, xy-refinement and p-increment, as depicted in Figure 1. Note, x and y refer to reference-space coordinates of elements that can be arbitrarily oriented and curved in physical space. Also, the subelements created through refinement inherit the approximation order from the original element. In three dimensions a hexahedron can be refined in eight ways: three single-plane cuts, three double-plane cuts, isotropic refinement, and p increment.

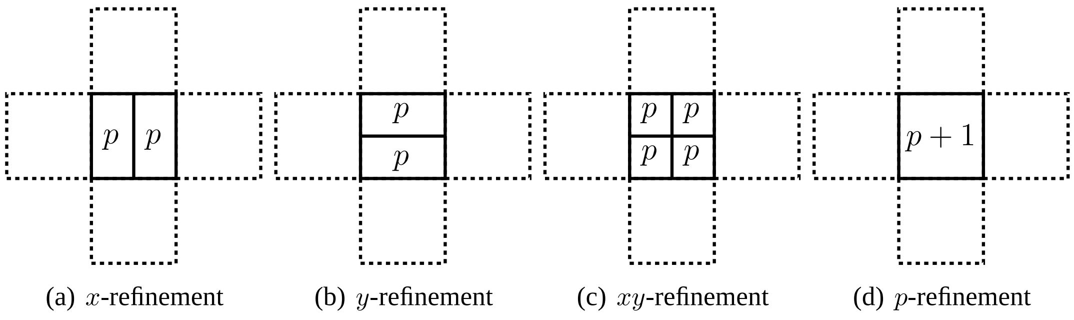


Figure 1. Quadrilateral refinement options. The dashed lines indicate the neighbors of the refined element.

h-refinement is performed in an element's reference space by employing the coarse element's reference-to-global coordinate mapping in calculating the refined element's geometry node coordinates. The refined elements inherit the same geometry approximation order and quadrature rules as the parent coarse element. As a result, there is no loss of element quality when a nonlinear mapping is used to fit the element to a curved geometry. Therefore, curved elements near a boundary can be efficiently refined to capture boundary layers in viscous flow. For simplicity of implementation, the initial mesh is assumed to capture the geometry sufficiently well, through a high enough order of geometry interpolation on curved boundaries, such that no additional geometry information is used throughout the refinements. That is, refinement of elements on the geometry boundary does not change the geometry. We note that for highly-anisotropic meshes, curved elements may be required away from the boundary, and for simplicity we use meshes with curved elements throughout the domain.

Note that elements created in a hanging-node refinement can be marked for h-refinement again in subsequent adaptation iterations. In this case, neighbors will be cut to keep one level of refinement difference between adjacent cells. This is illustrated in Figure 2.

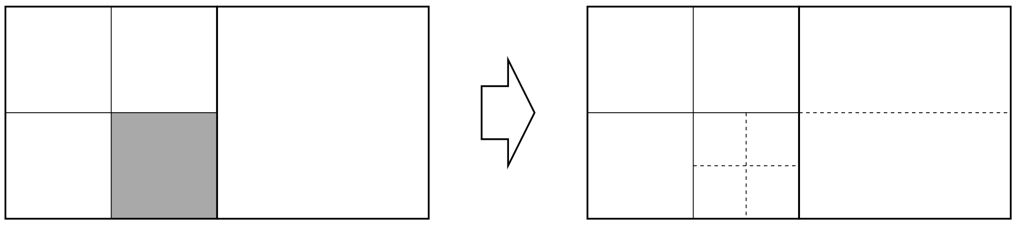


Figure 2. Hanging-node adaptation for a quadrilateral mesh, with a maximum of one level of refinement separating two elements. The shaded element on the left is marked for refinement, and the dashed lines on the right indicate the additional new edges formed.

5. MERIT FUNCTION

The choice of a particular refinement option is made locally in each element flagged for refinement. This choice is made by defining a merit function $m(i)$ that ranks each available refinement option i. This function is defined as

$$m(i) = \frac{b(i)}{c(i)} \,, \tag{10}$$

where b and c respectively correspond to measures of the benefit and the computational cost of the refinement option indexed by i. These measures depend on the method used for solving the flow equations and they should be tailored for each specific solver. We define them further in this section in the context of the applications presented in this paper.

During calculation of the merit function, local mesh and data structures are created that include the flagged element and its first-level neighbors along with the corresponding primal and adjoint states. In these local structures, the central element is refined in turn according to each of the discrete options. On the refined local mesh, the merit function is computed and the refinement option with the largest value of $m(i)$ is chosen.

5.1. Cost

We consider two measures of computational cost. The first measure is solution storage that is proportional to the number of degrees of freedom in the discrete state vector. For tractability, we consider only the degrees of freedom pertinent to the flagged element κ^H,

$$c_{\mathrm{DOF}}(i) = \sum_{\kappa^h \in \kappa^H} (p_{\kappa^h}(i) + 1)^{\mathrm{dim}}, \tag{11}$$

where $\kappa^h \in \kappa^H$ denotes the subelements embedded in the original element selected for refinement and $p_{\kappa^h}(i)$ is the element's approximation order after the refinement as depicted in Figure 1. Note that $p_{\kappa^h} = p_{\kappa^H}$ for *h*-refinement while the number of embedded elements changes. Conversely, $p_{\kappa^h} = p_{\kappa^H} + 1$ for *p*-refinement and there is only one embedded element, *i.e.* the original element. Also, we are not considering the rank of the conserved state vector N_s because it is a constant throughout the mesh. It is worth emphasizing that this measure of cost is insensitive to the type of time integration used to solve Eqn. 1, and therefore it is a generic measure of cost.

The second measure of computational cost incorporates information about the time integration method. In this work, most of the computational time is spent solving the linear system in Eqn. 4 using the GMRES algorithm. In a sparse structure such as in Eqn. 4, we approximate the number of floating point operations in applying GMRES by the number of non-zero entries in the Jacobian matrix. Based on this observation, we define the second measure of cost as:

$$c_{\mathrm{NZ}}(i) = \sum_{\kappa^h \in \kappa^H} \left\{ (p_{\kappa^h}(i) + 1)^{2\cdot \mathrm{dim}} + \sum_{\partial\kappa^h \setminus \partial D} \left[(p_{\kappa^h}(i) + 1) \cdot (p^-_{\kappa^h}(i) + 1) \right]^{\mathrm{dim}} \right\}, \tag{12}$$

where $p^-_{\kappa^h}$ denotes the approximation order of the neighboring element across face $\partial\kappa^h$, which must not be part of the boundary of the domain, ∂D. The first term in Eqn. 12 accounts for the self-blocks of the residual Jacobian matrix corresponding to each of the subelements. The second term corresponds to the dependence of the subelements' residual on the neighboring states. The cost function does not take into account possible sparsity within the blocks of the Jacobian matrix, as such sparsity is not taken into account by the solver. Note that c_{NZ} is more sensitive to the number of spatial dimensions than c_{DOF}.

5.2. Benefit

The benefit $b(i)$ is a measure of how much improvement in the prediction of an output results from refinement option i. Evidently, the definition of benefit is not unique and it may be tailored for different applications and solution methods. However, it is desirable that such a definition is tractable and computationally inexpensive.

In an output-based mesh adaptation cycle, the steady-state residual is driven to zero at each step. Therefore, mesh modification on the element level can be interpreted as a local residual perturbation. Since an adjoint solution represents the sensitivity of an output with respect to a residual perturbation, we define our benefit function as:

$$b(i) = \sum_{\kappa^h \in \kappa^H} |\mathrm{R}_{\kappa^h}(\mathrm{U}_k \mathrm{T}_{kl}(i))_j| \cdot |\Psi_k \mathrm{T}_{kl}(i)|, \tag{13}$$

where $R_{\kappa^h}(\cdot)_j$ is a discrete residual component in the embedded element, $\mathrm{T}_{kl}(i)$ is a matrix that transfers the discrete primal and adjoint solutions, U_k and Ψ_k, to the local meshes for each refinement i. Note that the adjoint variables act as positive weights for each of the perturbations.

The definition in Eqn. 13 relies on the following observations:

- At each step of the adaptation cycle, a discrete primal solution is found so that the residual vector is machine-zero. Therefore, the benefit as defined in Eqn. 13 is also machine-zero if computed before refining the central element.
- In the limit of the discrete solution representing the exact solution to machine precision, the result of Eqn. 13 will be of the order of machine precision for any refinement option.
- The refinement option with the largest $b(i)$ is expected to be the option that produces the largest change in the output of interest.

Note that Eqn. 13 is inexpensive to compute since only a residual calculation in the local mesh and data structures is required for each refinement option. Also, this framework is different than a residual-based decision because the values of the discrete adjoint provide information on the distribution of output sensitivity.

6. MESH PARTITIONING

At higher p-orders, the time taken to solve the primal and dual problems increases and the non-homogeneity of p affects the distribution of computational work amongst the processors. Therefore, dynamic load-balancing for hp-adaptive methods is important for efficient use of computational resources. However, such balance is not trivial and, in fact, is a topic of research rarely explored. The difficulty is that the computational effort for the residual operator, and its Jacobian, is not constant between elements in the mesh and the performance of the line-Jacobi preconditioner deteriorates [10] when cells with strong coupling reside in different processors.

Typically in CFD, the mesh is represented as an irregular graph where each element κ^H is a node in the graph and the interior faces $\partial\kappa^H \setminus \partial D$ are edges in the graph (Figure 3). This graph is then partitioned using a multilevel algorithm in which sequences of smaller graphs are systematically generated and partitioned until the partitions are as close to equal size as possible.

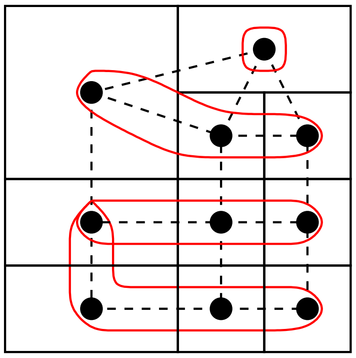


Figure 3. Example of mesh (continuous lines) and corresponding graph (dashed lines); the sets of elements circled in red represent lines of the preconditioner.

In our work, we use the multilevel k-way graph partitioning algorithm implemented in the ParMETIS library [25] which permits the attribution of weights to nodes and edges of the graph. The node weights are used to represent the computational effort for each element due to non-uniformity of p-orders. The edge weights are used to make the partitioning algorithm avoid separating elements with strong coupling improving the effectivity of the preconditioner.

The inter-domain communication stores the data in one layer of fictitious elements neighboring each inter-domain boundary. This information is enough for the residual calculation and the assembly of its Jacobian due the compact stencil of DG discretizations.

The node weights are computed based on the number of non-zero entries in the self-blocks of the Jacobian matrix,

$$\omega_{\kappa^H} = (p_{\kappa^H} + 1)^{2\cdot\mathrm{dim}}, \tag{14}$$

where p_{κ^H} is the polynomial order of element κ^H.

The edge weights are computed in the following sequence.

1. Loop through edges of the graph and compute:

$$\omega_{\partial\kappa^H\setminus\partial D} = (p^+_{\kappa^H} + 1)^{\mathrm{dim}} + (p^-_{\kappa^H} + 1)^{\mathrm{dim}}, \tag{15}$$

   where $p^+_{\kappa^H}$ and $p^-_{\kappa^H}$ are the polynomial orders of the elements on both sides of the interior face.

2. Loop trough edges of the graph that are part of lines of the preconditioner (red node groups in Figure 3) and augment $\omega_{\partial\kappa^H\setminus\partial D}$ with

$$\omega_{\partial\kappa^H\setminus\partial D} \Leftarrow \omega_{\partial\kappa^H\setminus\partial D} \cdot \max(v^+_{\kappa^H}, v^-_{\kappa^H}), \tag{16}$$

where $v^+_{\kappa^H}$ and $v^-_{\kappa^H}$ are the number of edges that are connected to the nodes on each end of edge $\partial\kappa^H \setminus \partial D$.

Equation 15 gives weights to the edges that are proportional to the amount of data transferred in each exchange of information between processors. The second step (Eqn. 16) makes the partitioner prefer to separate elements that are not strongly coupled.

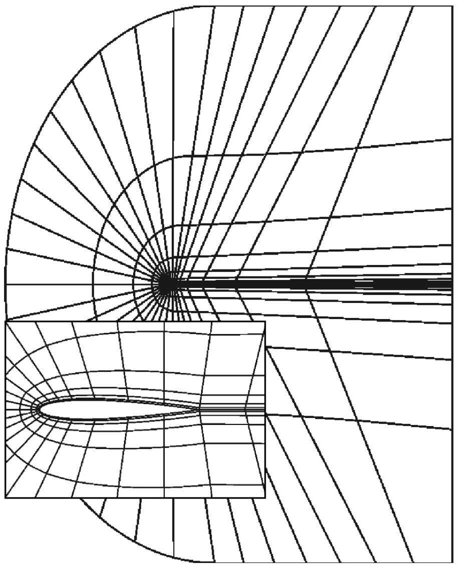
7. RESULTS

In this section, we assess the performance of our hp-adaptation framework using the cost measures c_{DOF} and c_{NZ}. The performance is measured in terms of number of degrees of freedom and wall-clock time. In the output-based adaptation methods, the time stamps include the solution of both the primal and adjoint solves and the time taken to estimate the output error, while for the uniform refinements only the primal solve time is included.

At each step of the adaptive process, $f^{\text{adapt}} = 10\%$ of the elements with largest η_{κ^H} are selected for refinement. The output of interest is drag and we start from a $p = 1$ solution for all the results presented.

7.1. NACA 0012 - $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$

The first case is two-dimensional flow at $M_\infty = 0.5, \alpha = 1.0^o$ and $Re = 5 \times 10^3$ over the NACA 0012 airfoil. We compare the hp-adaptation framework using both cost measures against uniform h and p refinements. Figure 4 shows the initial mesh and Mach contours. This case used 1 Nehalem 8-core node from the Nyx cluster at the University of Michigan.


(a) Initial mesh for the NACA 0012.

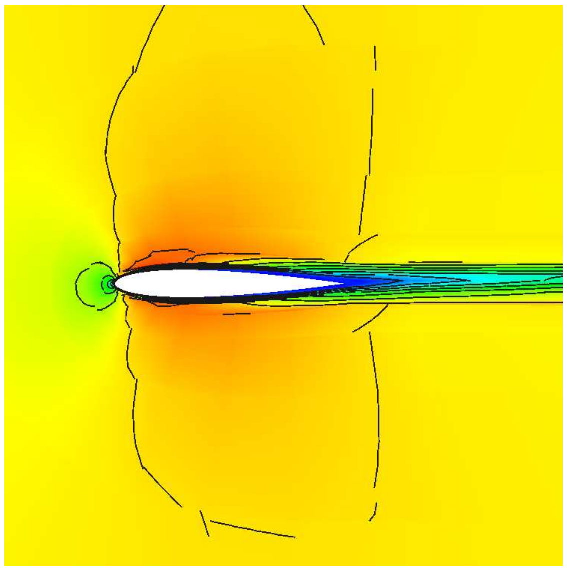
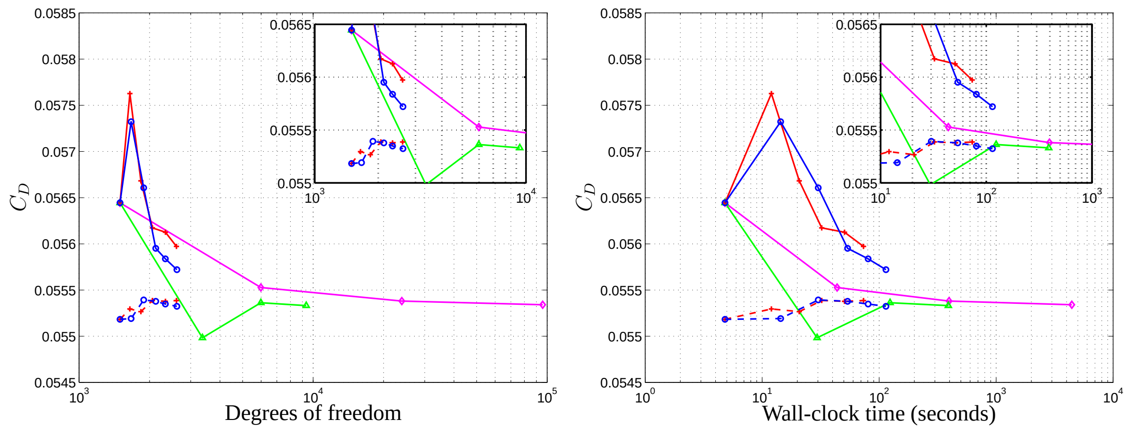

(b) Initial Mach contours obtained on the initial mesh with $p = 1$.

Figure 4. NACA 0012 - $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$: Initial quartic ($q = 4$) mesh and Mach contours.

Figure 5(a) shows the drag coefficient convergence in terms of number of degrees of freedom. While both hp-adaptation runs present similar convergence histories for the corrected output (dashed lines), the uncorrected drag values (solid lines) converge faster with c_{DOF} than when c_{NZ} is employed. Additionally, the hp-adaptation runs converge the corrected output with significantly fewer degrees of freedom than the uniform refinements. Figure 5(b) shows the performance in terms of wall-clock time of our hp- adaptation method without the adaptive repartitioning described in Section 6. This comparison favors the uniform refinement strategies.



(a) Drag coefficient evolution with respect to degrees of freedom

(b) Drag coefficient evolution with respect to wall-clock time

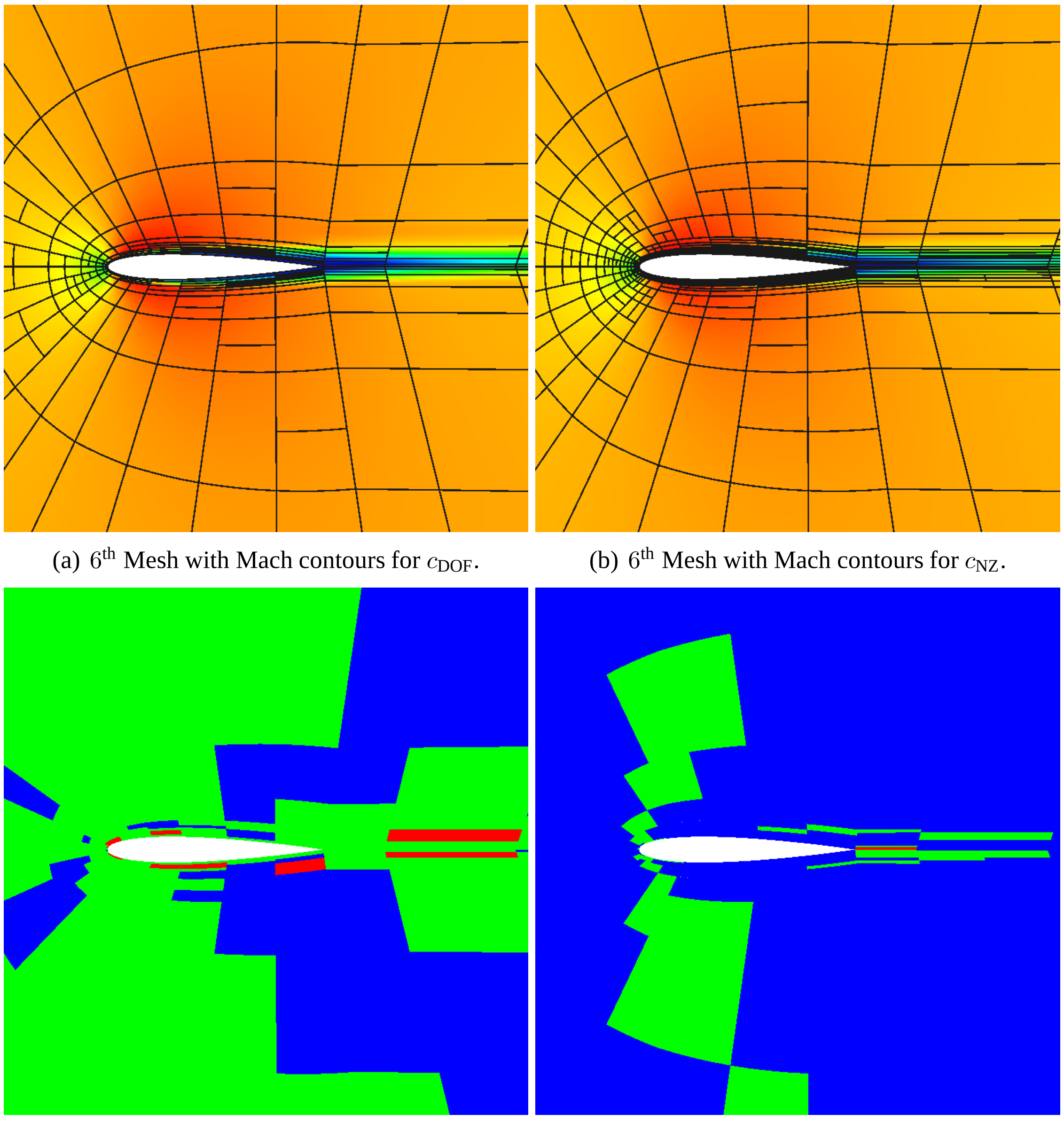
Figure 5. NACA 0012, $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$: drag coefficient convergence; $\diamond$: uniform h-refinement; $\triangle$: uniform p-refinement; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}. The dashed lines correspond to the drag values corrected with the error estimate.

Table 1 shows the percentage of choice of the different refinement options for each of the cost measures. Note that p-increment is chosen significantly more often when c_{DOF} is used, while c_{NZ} prefers single-cut h-refinement. Additionally, c_{NZ} performs slightly better in terms of time (Figure 5(b)), while c_{DOF} converges faster in terms of degrees of freedom (Figure 5(a)).

Table 1. NACA 0012, $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$, drag-driven adaptation: percentage of choice for each refinement option; iso-h: isotropic h-refinement; sc-h: single-cut h-refinements; iso-p: isotropic p-refinement.

	c_{DOF}			c_{NZ}		
Adaptation step	iso-h	sc-h	iso-p	iso-h	sc-h	iso-p
1	0.0	35.1	64.9	0.0	86.5	13.5
2	0.0	31.6	68.4	0.0	85.0	15.0
3	0.0	27.5	72.5	0.0	70.4	29.5
4	0.0	19.5	80.5	0.0	91.7	8.3
5	0.0	31.0	69.0	0.0	70.4	29.6
6	0.0	20.0	80.0	0.0	82.7	17.2

Figure 6 shows the final meshes and p-order distributions. The adaptive scheme produces a larger area of the domain with higher order cells when c_{DOF} is used to measure cost (Figure 6(c)). In contrast, the adaptive algorithm, when using c_{NZ}, chooses p-increment mostly in the wake region combined with anisotropic h-refinement as seen in Figures 6(d) and 6(b) respectively.


(a) 6th Mesh with Mach contours for c_{DOF}.

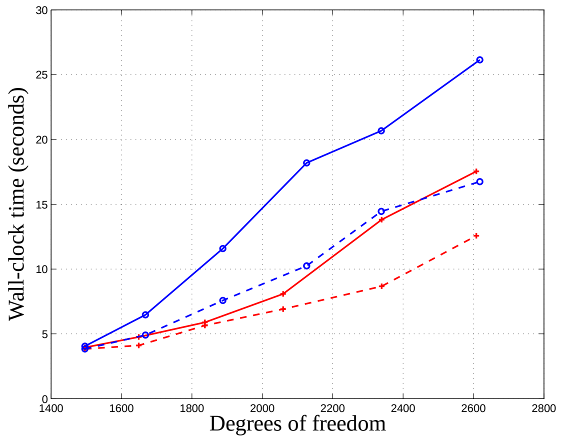
(b) 6th Mesh with Mach contours for c_{NZ}.

(c) 6th p-order distribution for c_{DOF}; blue indicates $p = 1$; red indicates $p = 3$.

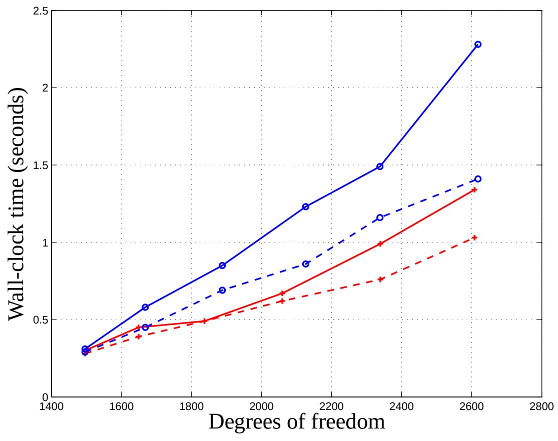
(d) 6th p-order distribution for c_{NZ}; blue indicates $p = 1$; red indicates $p = 3$.

Figure 6. NACA 0012, $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$: hp-adapted meshes for drag.

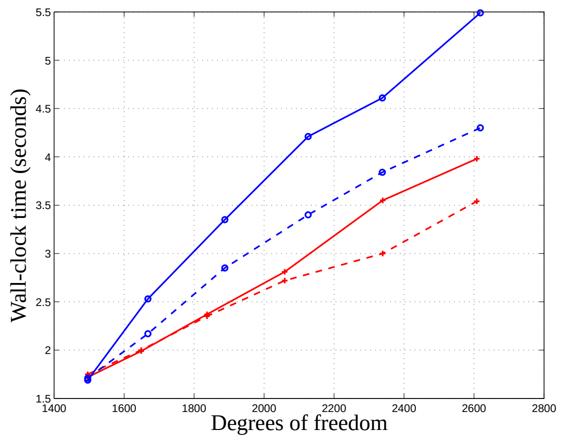
We now analyze the effect of the mesh partitioning algorithm described in Section 6. Figure 7(d) shows that the weighted partitioning saves close to 1000 GMRES iterations in the last 2 adaptation steps. This is mostly due to using the element lines of the preconditioner to assign weights to the edges of the graph. We use block-Jacobi smoothing for the fine-space solves involved in error estimation, therefore, the savings shown in Figure 7(c) are due to a better distribution of computational work amongst the processors. Figures 7(a) and 7(b) show the timings for the primal and adjoint solves. The weighted partitioning runs are significantly faster then the baseline unweighted partitioning method.



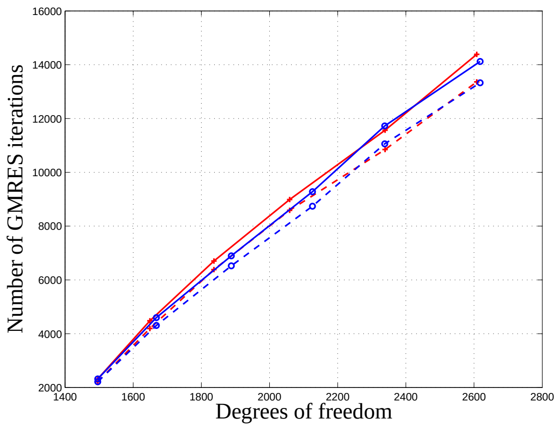
(a) Wall-clock time taken for primal solves.



(b) Wall-clock time taken for adjoint solves.



(c) Wall-clock time taken for error estimation and adaptation.



(d) Number of GMRES iterations taken for primal and adjoint solves.

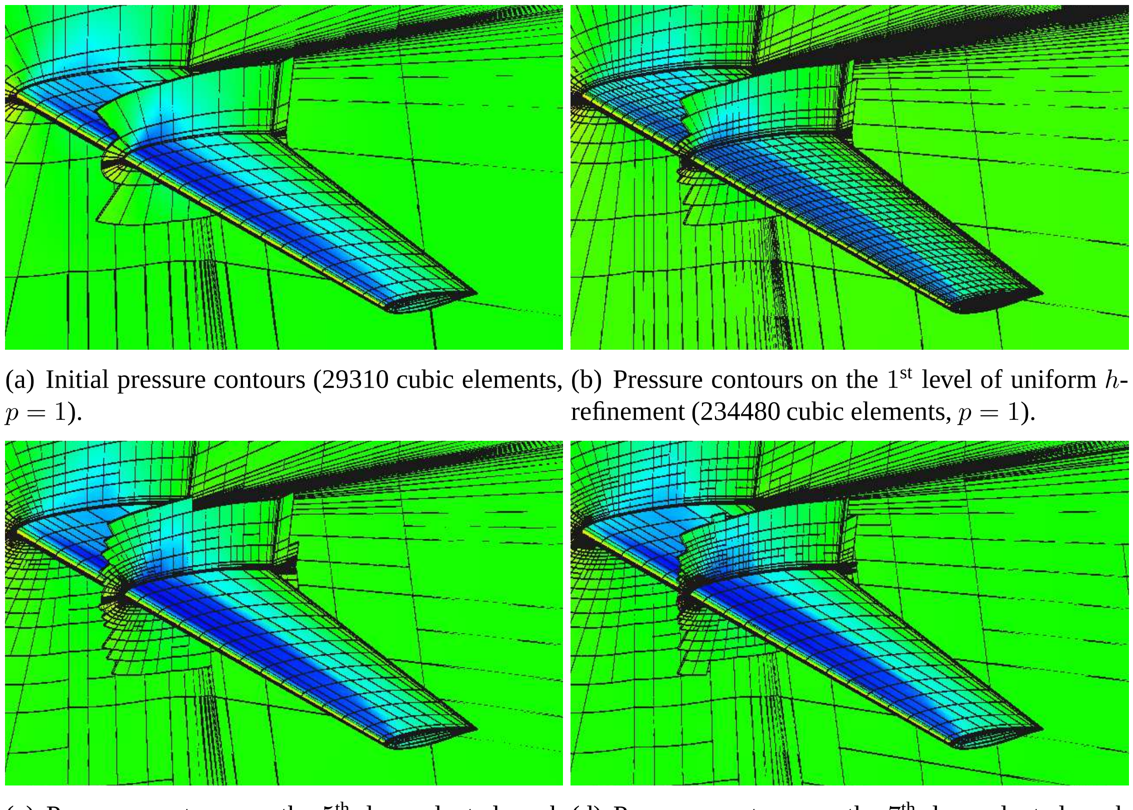
Figure 7. NACA 0012, $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$: effect of weighted partitioning - the points correspond to each of the adaptation steps; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}; solid lines: unweighted partitioning; dashed lines: weighted partitioning.

7.2. Third Drag Prediction Workshop W1 geometry - $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$

In this case study, we consider the baseline wing geometry (DPW-W1) from the third AIAA Drag Prediction Workshop. This case consists of turbulent, transonic flow over a tapered wing and the mesh adaptation routine is driven by the drag output. The initial curved mesh, shown in Figure 8(a), was obtained through agglomeration of cells from a finer structured linear C-grid generated specifically for this purpose. In the agglomeration, each curved hexahedral element was obtained by merging twenty seven linear elements using a distance-based Lagrange interpolation of the nodal coordinates, resulting in cubic ($q = 3$) geometry interpolation. Also, the spacing of the linear mesh is such that the agglomerated mesh presents $y^+ \approx 1$ for the first element off the wall as recommended in the workshop and the outer boundary is located at 100 mean-aerodynamic-chord-lengths away from the wing.

We used the Spalart-Allmaras turbulence model without trip terms and we assumed the flow as fully turbulent. Also, Persson and Peraire's [33] shock-capturing method is used to improve stability. The flow was initialized with free-stream values and, due to the difficulty of this case, the constraint handling technique presented in Ref. [7] was used to compute the baseline flow with linear ($p = 1$) approximation order. As a basis of comparison, all the adaptive schemes started from the same initial solution so that all methods are compared against the same initial time-stamp. For the adjoint-based adaptation methods, the wall-clock time taken for the initial adjoint solve is also included in the initial starting time. Similarly to the previous case, the adaptive runs used unweighted partitioning and this favors uniform refinement. We assess the effect of weighted partitioning later in this section.

All of the calculations for this case were executed on 180 Harpertown 8-core nodes from NASA's Pleiades supercomputer. Due to the computational expense of these runs, we did not perform a statistical study to account for machine performance variability in the CPU-time measurements.


(a) Initial pressure contours (29310 cubic elements, $p = 1$).

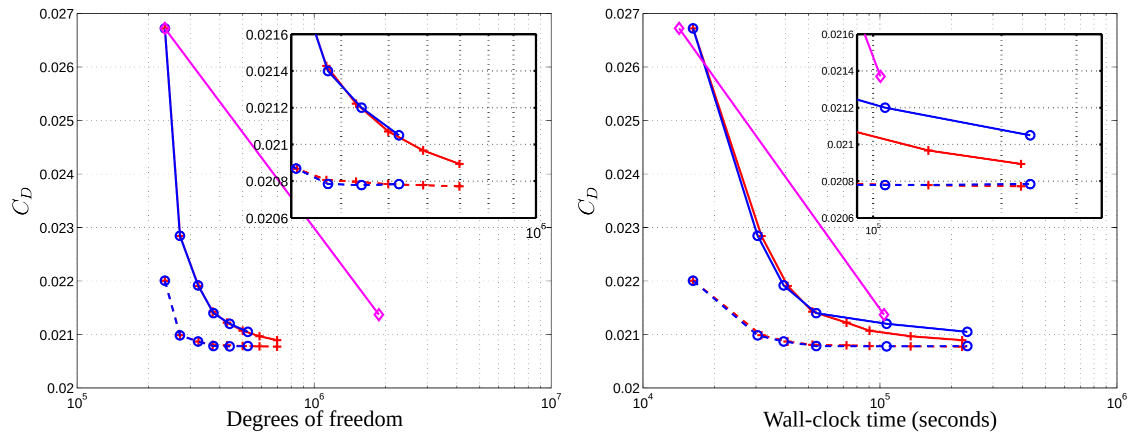
(b) Pressure contours on the 1st level of uniform h-refinement (234480 cubic elements, $p = 1$).

(c) Pressure contours on the 5th drag-adapted mesh using c_{DOF} (59503 cubic elements).

(d) Pressure contours on the 7th drag-adapted mesh using c_{NZ} (85377 cubic elements).

Figure 8. DPW Wing 1 - $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: Initial and drag-adapted meshes with pressure contours.

We compare three mesh improvement strategies starting from the initial $p = 1$ solution shown in Figure 8(a). As reference, one of the strategies is uniform h-refinement (Figure 8(b)) in which all hexahedra are divided into 8 elements. The two cost measures described earlier are compared for hp-adaptation in which $f^{\text{adapt}} = 10\%$ of the elements is selected for refinement at each adaptation step. Additionally, we fixed the overall budget of CPU wall-time for each of the three runs and the last converged solution obtained within that budget are shown in Figure 8.



(a) Drag coefficient evolution with respect to number of degrees of freedom

(b) Drag coefficient evolution with respect to CPU time

Figure 9. DPW Wing 1 - $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: drag coefficient convergence; $\diamond$: uniform h-refinement; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}. The dashed lines correspond to the drag values corrected with the error estimate.

Figure 9 shows the drag coefficient convergence for the mesh refinement strategies. The solid lines in Figures 9(a) and 9(b) are the computed drag values and the dashed lines correspond to the output corrected with the error estimate. The difference between these corrected values for the last two adaptation steps of the output-based strategies is within 0.15 counts of drag. Note that the performance in terms of degrees of freedom of the output-based strategies is very similar. However, in terms of CPU time, the use of c_{NZ} leads to faster output convergence. This difference is due to the more representative measure of solution cost by c_{NZ}. This effect is illustrated in Table 2 where we show the frequency at which the refinement options were chosen for each cost measure at each adaptation step. Note that for c_{NZ}, p-refinement is chosen significantly less often than for c_{DOF} and both methods have a propensity to choose p-refinement more often in the later stages of adaptation. Moreover, the large increase in CPU time between the third and fourth adaptation steps for c_{DOF} (Figure 9(b)) is an effect of p-increment being chosen more often for c_{DOF} (Table 2) which makes the primal and adjoint solves more expensive.

Figure 10 shows the effect of the weighted partitioning method. Note in Figure 10(a) that the first primal solve was the longest, this is due to RANS-SA being very stiff. In the first primal solve, the weighted partition runs took about two thirds of the time taken by the un-weighted runs, even though the preconditioner lines used for the edge weights were computed with the free-stream initial condition. Additionally, the savings with c_{DOF} increases as the adaptation progresses and the number of higher-order cells increases. The complete adaptive sequence is not presented in Figure 10 because the disparity of node weights increases with higher p-orders in 3 dimensions and the partitioner creates subdomains without elements. The solution to this problem is part of our ongoing work.

Table 2. DPW Wing 1 - $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: percentage of choice for each refinement option; iso-h: isotropic h-refinement; sc-h: single-cut h-refinements; dc-h: double-cut h-refinements; iso-p: isotropic p-refinement.

	c_{DOF}				c_{NZ}			
Adaptation step	iso-h	sc-h	dc-h	iso-p	iso-h	sc-h	dc-h	iso-p
1	0.0	99.3	0.0	0.7	0.0	100.0	0.0	0.0
2	0.0	97.3	0.0	2.7	0.0	99.9	0.0	0.1
3	0.0	94.9	0.0	5.1	0.0	99.8	0.0	0.2
4	0.0	91.8	0.4	7.8	0.0	99.1	0.3	0.6
5	0.0	90.6	0.3	9.1	0.0	98.7	0.5	0.8
6	–	–	–	–	0.0	98.6	0.5	0.9
7	–	–	–	–	0.0	98.6	0.4	1.0

8. CONCLUSIONS

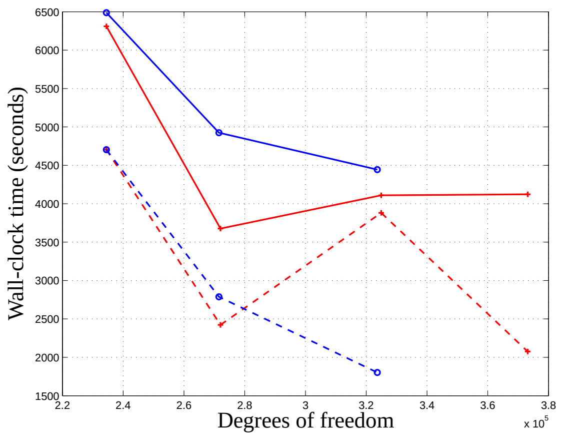
We demonstrated the use of an optimization-based hp-adaptation method in aerodynamic problems. The objective function used for ranking the refinement options uses a measure of output sensitivity and a measure of computational cost. We compared two measures of cost: number of degrees of freedom and number of floating point operations. The latter is correlated to the number of non-zero entries in the residual Jacobian.

The cost measures presented here are not perfect as they do not account for stiffness effects of different refinements in the iterative solution method. These effects have a direct impact in the CPU time, but they are difficult and computationally intensive to estimate because they are global measures and their effect depends on the type of solver and preconditioner used. Since the merit function is computed multiple times in each refinement cycle, the local property of cost and benefit measures is attractive.

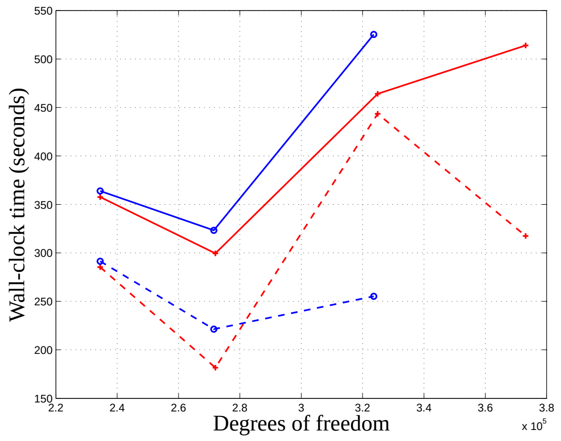
We proposed a method for assigning weights to the nodes and edges of the graph used for partitioning the mesh. The method accounts for the non-homogeneity of computational cost of the elements in the mesh and uses preconditioner information to improve parallel efficiency. The results show speedup of up to 2 with the weighted partitioning in 3 dimensions. Although not yet fully robust due to the possibility of empty partitions, the weighted-partitioning algorithm is an attractive option for the element-line preconditioner. Extension to other preconditioners, such as the incomplete lower-upper (ILU) factorization, is a subject of future work.

Acknowledgements

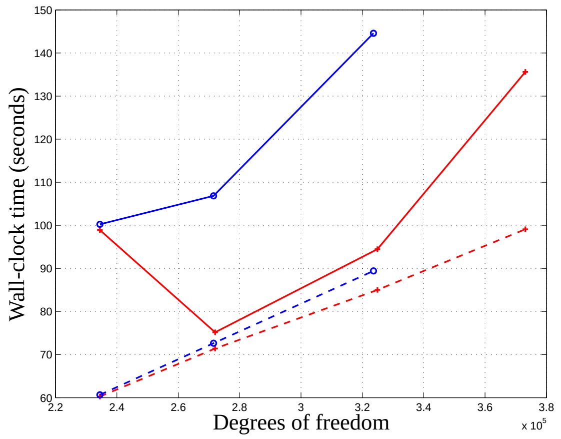
The authors acknowledge the support given by the University of Michigan to the development of this work and appreciate the computational resources provided by NASA which were essential to obtaining the results presented here. Funding for this research is provided by the Air Force Office of Scientific Research under contract number FA9550-10-C-0040.



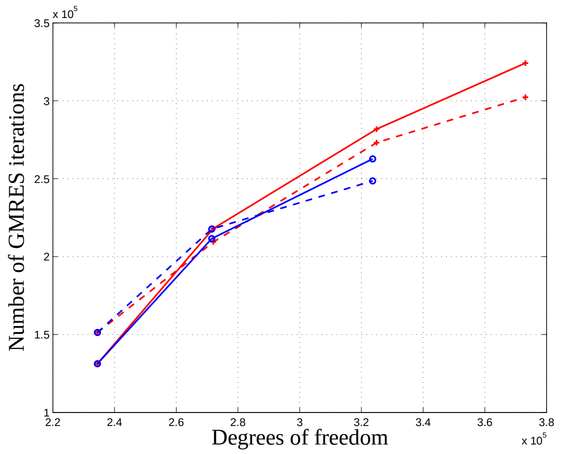
(a) Wall-clock time taken for primal solves.



(b) Wall-clock time taken for adjoint solves.



(c) Wall-clock time taken for error estimation and adaptation.



(d) Number of GMRES iterations taken for primal and adjoint solves.

Figure 10. DPW Wing 1 - $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: effect of weighted partitioning - the points correspond to each of the adaptation steps; $\circ$: hp-adaptation with $c_{\rm DOF}$; $+$: hp-adaptation with $c_{\rm NZ}$; solid lines: unweighted partitioning; dashed lines: weighted partitioning.

References

[1] Timothy Barth and Mats Larson, *A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes*, Finite Volumes for Complex Applications III (London) (R. Herban and D. Kröner, eds.), Hermes Penton, 2002, pp. 41–63.

[2] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numerica (A. Iserles, ed.), Cambridge University Press, 2001, pp. 1–102.

[3] Kim S. Bey, *An hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws*, Ph.D. thesis, University of Texas at Austin, 1994.

[4] Nicholas K. Burgess and Dimitri J. Mavriplis, *An hp-adaptive discontinuous Galerkin solver for aerodynamic flows on mixed-element meshes*, 49th AIAA Aerospace Sciences

Meeting and Exhibit, 2011.

[5] M. J. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau, *Anisotropic unstructured mesh adaptation for flow simulations*, International Journal for Numerical Methods in Fluids **25** (1997), 475–491.

[6] Marco Ceze and Krzysztof J. Fidkowski, *Output-driven anisotropic mesh adaptation for viscous flows using discrete choice optimization*, 48th AIAA Aerospace Sciences Meeting and Exhibit, 2010.

[7] ______, *A robust adaptive solution strategy for high-order implicit CFD solvers*, 20th AIAA Computaional Fluid Dynamics Conference, AIAA, 2011.

[8] ______, *An anisotropic hp-adaptation framework for functional prediction*, AIAA Journal (2012 Accepted).

[9] K. J. Fidkowski and D. L. Darmofal, *A triangular cut–cell adaptive method for high–order discretizations of the compressible Navier–Stokes equations*, Journal of Computational Physics **225** (2007), 1653–1672.

[10] Krzysztof J. Fidkowski, *A high–order discontinuous Galerkin multigrid solver for aerodynamic applications*, MS thesis, M.I.T., Department of Aeronautics and Astronautics, June 2004.

[11] ______, *A simplex cut-cell adaptive method for high–order discretizations of the compressible Navier-Stokes equations*, PhD dissertation, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2007.

[12] Krzysztof J. Fidkowski and David L. Darmofal, *Review of output-based error estimation and mesh adaptation in computational fluid dynamics*, AIAA Journal **49** (2011), no. 4, 673–694.

[13] L. Formaggia, S. Micheletti, and S. Perotto, *Anisotropic mesh adaptation with applications to CFD problems*, Fifth World Congress on Computational Mechanics (Vienna, Austria) (H. A. Mang, F. G. Rammerstorfer, and J. Eberhardsteiner, eds.), July 7-12 2002.

[14] Luca Formaggia and Simona Perotto, *New anisotropic a priori error estimates*, Numerische Mathematik **89** (2001), 641–667.

[15] Luca Formaggia, Simona Perotto, and Paolo Zunino, *An anisotropic a posteriori error estimate for a convection-diffusion problem*, Computing and Visualization in Science **4** (2001), 99–104.

[16] Stefano Giani and Paul Houston, *High–order hp–adaptive discontinuous Galerkin finite element methods for compressible fluid flows*, ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications (Norbert Kroll, Heribert Bieler, Herman Deconinck, Vincent Couaillier, Harmen

van der Ven, and Kaare Sørensen, eds.), Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 113, Springer Berlin / Heidelberg, 2010, pp. 399–411.

[17] M. B. Giles and E. Süli, *Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality*, Acta Numerica, vol. 11, 2002, pp. 145–236.

[18] Michael B. Giles, Mihai C. Duta, Jens-Dominik Müller, and Niles A. Pierce, *Algorithm developments for discrete adjoint methods*, AIAA Journal **41** (2003), no. 2, 198–205.

[19] Wagdi G. Habashi, Julien Dompierre, Yves Bourgault, Djaffar Ait-Ali-Yahia, Michel Fortin, and Marie-Gabrielle Vallet, *Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles*, International Journal for Numerical Methods in Fluids **32** (2000), 725–744.

[20] Ralf Hartmann, *Adjoint consistency analysis of discontinuous Galerkin discretizations*, SIAM Journal on Numerical Analysis **45** (2007), no. 6, 2671–2696.

[21] Ralf Hartmann and Paul Houston, *Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations*, Journal of Computational Physics **183** (2002), no. 2, 508–532.

[22] V. Heuveline and R. Rannacher, *Duality-based adaptivity in the hp-finite element method*, Journal of Numerical Mathematics **11** (2003), no. 2, 95–113.

[23] Paul Houston, Emmanuil H. Georgoulis, and Edward Hall, *Adaptivity and a posteriori error estimation for DG methods on anisotropic meshes*, International Conference on Boundary and Interior Layers, 2006.

[24] Paul Houston and Endre Süli, *A note on the design of hp-adaptive finite element methods for elliptic partial differential equations*, Computer Methods in Applied Mechanics and Engineering **194** (2005), 229–243.

[25] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal for Scientific Computing **20** (1998), no. 1, 359–392.

[26] James Lu, *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2005.

[27] Charles A. Mader, Joaquim R.R.A. Martins, Juan J. Alonso, and Edwin van der Weide, *ADjoint: An approach for the rapid development of discrete adjoint solvers*, AIAA Journal **46** (2008), no. 4, 863–873.

[28] D. J. Mavriplis, *Adaptive mesh generation for viscous flows using Delaunay triangulation*, Journal of Computational Physics **90** (1990), 271–291.

[29] D. J. Mavriplis and A. Jameson, *Hermite-based mesh adaptation for functional outputs improvement in fluid flow simulation*, AIAA Journal **47** (2009), no. 8, 1965–1976.

[30] Todd A. Oliver, *A high–order, adaptive, discontinuous Galerkin finite elemenet method for the Reynolds-Averaged Navier-Stokes equations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[31] Michael A. Park, *Anisotropic output-based adaptation with tetrahedral cut cells for compressible flows*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[32] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, *Adaptive remeshing for compressible flow computations*, Journal of Computational Physics **72** (1987), 449–466.

[33] P.-O. Persson and J. Peraire., *Sub-cell shock capturing for discontinuous Galerkin methods*, AIAA Paper 2006-112, 2006.

[34] W. Rachowicz, D. Pardo, and L. Demkowicz, *Fully automatic hp-adaptivity in three dimensions*, Tech. Report 04-22, ICES, 2004.

[35] R. Rannacher, *Adaptive Galerkin finite element methods for partial differential equations*, Journal of Computational and Applied Mathematics **128** (2001), 205–233.

[36] R. Schneider and P. K. Jimack, *Toward anisotropic mesh adaptation based upon sensitivity of a posteriori estimates*, Tech. Report 2005.03, University of Leeds, School of Computing, 2005.

[37] P. Solín and L. Demkowicz, *Goal-oriented hp-adaptivity for elliptic problems*, Computer Methods in Applied Mechanics and Engineering **193** (2004), 449–468.

[38] D. A. Venditti, *Grid adaptation for functional outputs of compressible flow simulations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2002.

[39] D. A. Venditti and D. L. Darmofal, *Grid adaptation for functional outputs: application to two-dimensional inviscid flows*, Journal of Computational Physics **176** (2002), no. 1, 40–39.

[40] ______, *Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows*, Journal of Computational Physics **187** (2003), no. 1, 22–46.