Verification of Unstructured Grid Adaptation Components

Michael A. Park,* Aravind Balan,† and W. Kyle Anderson‡
NASA Langley Research Center, Hampton, VA 23681, USA

Marshall C. Galbraith,§ Philip C. Caplin,¶ and Hugh A. Carson∥
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Todd Michal∗∗ and Joshua A. Krakos††
Boeing Research & Technology, St. Louis, MO, USA

Dmitry S. Kamenetskiy‡‡
Boeing Research & Technology, Seattle, WA, USA

Adrien Loseille§§ and Frédéric Alauzet¶¶
INRIA Paris-Saclay, Alan Turing Building, 91120 Palaiseau, France

Loïc Frazza∗∗∗
Sorbonne Universités, UPMC Paris 06, 4 place Jussieu 75252 Paris cedex 05, France

Nicolas Barral†††
Imperial College London, South Kensington Campus, London SW7 2AZ, UK

Adaptive grid techniques have made limited impact on production analysis workflows where the control of discretization error is critical to obtaining reliable simulation results. Recent progress has matured a number of independent implementations of flow solvers, error estimation methods, and anisotropic grid adaptation mechanics. However, subtle known and unknown differences in implementation details produce inconsistent results between integrated grid adaptation methods. The error estimation and anisotropic grid adaptation procedures of unstructured grid adaptation can be treated like any other model in the simulation process. These procedures can be verified in a manner similar to turbulence models as documented by the Turbulence Modeling Resource (TMR) website. Once verified, unstructured grid adaptation can be integrated into production fluid simulation capabilities with confidence that the implementation is consistent with published descriptions and best practices.

I. Introduction

The use of Reynolds-averaged Navier–Stokes (RANS) equations with a turbulence model has become a critical tool for the design of aerospace vehicles. However, the issues that affect the grid convergence of three dimensional (3D) configurations are not completely understood, as documented in the AIAA Drag Prediction Workshop series[4–6]. This led to an effort to verify the turbulence models with the Turbulence Modeling Resource (TMR) website[4]. Morrison,
Kleb, and Vassberg [5] identified that “the DPW series does not have the systematic build up and definition on both the computational and experimental side that is required for detailed verification and validation.” This has led to a focus on benchmark problems of increasing difficulty by Diskin et al. [6–9]. These benchmark problems provide ideal examples to evaluate unstructured grid adaptation methods because well resolved solutions are available from a number of independent flow solvers on a series of carefully constructed uniformly-refined grids. The International Workshop on High-Order CFD Methods [10] also provides a range of cases that are suitable to computing highly accurate solutions.

Alauzet and Loseille [11] documented the dramatic progress made in the last decade for solution-adaptive methods that includes the anisotropy to resolve simulations with shocks and boundary layers, and identify where solution-adaptive improvements are needed via complex simulations. Park et al. [12] documented the current state of solution-based anisotropic grid adaptation and motivated further development with the impacts that improved capability would have on aerospace analysis and design in the broader context of the CFD Vision 2030 Study by Slotnick et al. [13].

The verification of error estimation and metric construction is a continuation of the efforts of Park et al. [14] to separate the solution-adaptive process into a number of components that can be independently verified, evaluated, and improved. Developing and documenting the evaluation methods is equally important as the test case descriptions. An informal Unstructured Grid Adaptation Working Group (UGAWG) has been formed to continue this process as described in their first benchmark [15], which focused on evaluating adaptive grid mechanics for analytic metric fields on planar and simple curved domains. The first benchmark contains a list of future directions, which includes the focus of this paper: separating the solution-adaptive process into components (e.g., error estimate, flow solver, mesh adaptation mechanics) that can be examined by modifying or replacing one component with the remaining components fixed.

The UGAWG evaluated the Three Dimensional Benchmark Turbulent Flows cases [9] as documented by Park et al. [16]. Michal et al. [17] also made an application of multiple anisotropic error estimation techniques to the ONERA M6 Wing. Both of these references compared the results of integrated adaptation processes composed of different flow solvers, error estimation techniques, and adaptive grid mechanics. The convergence of forces, moment, and grid properties were compared. Grids resulting from one integrated adaptation process were examined with different flow solvers. This comparison has yielded an understanding of the integrated adaptation process properties and has allowed some best practices to be identified. However, to gain a deeper understanding of implementation choices and details, individual components of the process must be verified.

The verification and validation process is described in detail by Oberkampf and Roy [18]. Verification and validation is part of the AIAA Engineering Standards for CFD and Complex Aerospace Systems [19]. Oberkampf and Trucano [20] specialize the discussion to CFD. The primary focus of this article is verification, where the question is asked, “has the model been implemented correctly?” The other question of validation, “is this the right model to use in this prediction?” has less emphasis in the current investigation. Trucano, Pilch, and Oberkampf [21] caution the use of the code-to-code comparisons in the context of rigorous verification exercises, but also indicate there are benefits when the Code Comparison Principle (CCP) is used appropriately. “Even given the philosophical limitations that we have stressed, benefits achieved from the use of the CCP for verification of complex problem numerical accuracy would likely increase if a rational methodology was consistently applied.” [21] The current work targets a rational application of code-to-code comparisons guided by these principles.

Documenting the outputs of a model for a specified set of inputs is advocated during publication to support the reproducibility aspect of the scientific method [22,23]. By verifying published methods by comparison to known sets of inputs and outputs or independent implementations, we set the stage for the further development of existing error estimation techniques and the creation of new methods. Oberkampf and Trucano [24] detail the desired properties of verification and validation benchmarks. The TMR website provides a successful presentation model for these verification and validation benchmarks. A repository has been initiated for unstructured grid adaptation verification (https://ugawg.github.io/) that can be further refined and improved by the results of this effort.

For example, the ideal norm exponent of the multiscale metric is debated between 2 and 4 [16,17,25]. It is difficult to determine the relative advantages of the selection of this norm order if there are known and potentially unknown differences in implementation, i.e., Hessian recovery. If the properties of the 2-norm and 4-norm can be replicated between implementations by verifying these multiscale metric implementations, this replicated evidence can increase the certainty in the behavior of a particular norm exponent for a particular case.

The method description begins with the nomenclature used to describe the components of the integrated grid adaptation procedure and an introduction to the multiscale metric. The tools used in code-to-code comparison process are described. The estimation and grid mechanics components are evaluated first on analytic functions and scalar convection to verify the convergence order properties of interpolation and output error. Two wing examples are examined with laminar and RANS viscous flow models. The force and moment trajectories of fixed grid refinement and the
solution adaptive processes illustrate the convergence and efficiency of the adaptive processes.

II. Method

The components of unstructured grid adaptation are shown on Fig. 1. Starting with an initial grid, a flow solution (and optionally, an adjoint solution) is computed. The information from the flow solution are used to estimate error and specify a new grid resolution request via an anisotropic metric. If the estimated errors are larger than limits specified by the practitioner, the current grid system is modified by grid mechanics to adhere to the anisotropic metric. Once the adapted grid is available, the previous flow solution is optionally interpolated to the new grid to provide an initial condition for the flow solver that approximates the converged solution. This improved initial condition can decrease the execution time and improve the robustness of the flow solution calculation, but initialization to freestream conditions is also possible. The process is repeated until an exit criteria is met (e.g., accuracy requirement or resource limit). There are potential interactions between each of these elements that impact the overall convergence and efficiency of the adaptation process. Details of the specific implementations of these components is detailed in the following subsections.

![Fig. 1 Solution-based grid adaptation process.](image)

The error estimation and metric construction block of Fig. 1 is responsible for forming the metric to specify the anisotropic density of the adapted grid. Loseille and Alauzet [26] provide a thorough introduction of the metric tensor field. A number of metric construction methods and will be demonstrated. Some metric formulations include the adjoint solution, which allows the targeting of a goal or functional output (e.g., lift, drag). The multiscale metric controls the $L_p$ norm of the interpolation error of a solution scalar field and forms the foundation of some of these goal or output metrics. The multiscale metric is computed as follows [26, 27]. The complexity $C$ of a metric $M$ is defined as an integral of the continuous metric field,

$$C(M) = \int_{\Omega} \sqrt{\det(M(x))} \, dx,$$

which is evaluated on the discrete grid and metric. The relationship between $C$ and the number of vertices and elements in the adapted grid is shown theoretically by [26] experimentally by [14, 28]. The (reconstructed) Hessian of a scalar field $H$ is locally scaled by the Hessian determinate and globally scaled to a specified target,

$$M_{L^p} = D_{L^p} \det(H)^{-1/p} |H|,$$

where the global scaling $D_{L^p}$,

$$D_{L^p} = \frac{C_t}{C \left( \det(H)^{-1/p} |H| \right)^{2/d}},$$

corrects the the complexity of the locally scaled Hessian to produce $M_{L^p}$ with specified target complexity $C_t$. Both scaling operations depend on dimensionality of the domain, which is $d = 3$ in this case. $M_{L^p}$ provides optimal control of the scalar field $p$-norm interpolation error. A lower $p$-norm targets weaker variations of the scalar field and a larger $p$-norm targets rapid variations of the scalar field. The metric calculation methods described below have subtle differences in how the multiscale metric is formed.
An adaptive series is created by optimizing a grid at a fixed complexity and then increasing the target complexity in a series of steps where the complexity is held fixed and grid is optimized at each step. The progression of forces and moments during these steps is referred to as a trajectory in the result sections. Typically, 5 to 10 fixed complexity adaptations are performed before increasing the complexity to the next target. These fixed complexity adaptations are referred to as subiterations in Michal et al. [17], where only the final force or moment value is plotted for the fixed complexity target. Alternatively, an average of a number of fixed complexity adaptations can be used to reduce the jitter in trajectories.

A. FUN3D-FV

FUN3D-FV (Fully-Unstructured Navier-Stokes 3D) [29,30] is a finite-volume Navier-Stokes solver in which the flow variables are stored at the vertices or nodes of the grid. FUN3D-FV solves the equations on mixed-element grids, including tetrahedra, pyramids, prisms and hexahedra. The adaptive grids in this study contain only tetrahedra. At interfaces between neighboring control volumes, the inviscid fluxes are computed using the Roe [31] approximate Riemann solver based on the values on either side of the interface. For second-order accuracy, interface values are extrapolated from the vertices with gradients computed at the grid vertices. These gradients are reconstructed with an unweighted least-squares technique [29].

The full viscous fluxes are discretized using a finite-volume formulation in which the required velocity gradients on the dual faces are computed using the Green-Gauss theorem. On tetrahedral grids this is equivalent to a Galerkin type approximation. The solution at each time step is updated with a backward Euler time-integration scheme. At each time step, the linear system of equations is approximately solved with a multicolor point-implicit procedure [32]. Local time-step scaling is employed to accelerate convergence to steady state. The negative Spalart-Allmaras (SA-neg) turbulence model [33] is loosely-coupled to the meanflow equations, where the meanflow and turbulence model equations are relaxed in an alternating sequence.

The SA-neg turbulence model requires the distance from every node to the nearest noslip boundary condition. The standard wall distance calculation in FUN3D-FV finds the nearest surface node and then searches adjacent triangles to see if they are closer than the closest surface node. The standard wall distance method over estimates the distance to the noslip boundary if the closest triangle is not adjacent to the closest surface node. To provide an accurate wall distance, which is critical to the SA-neg model, an alternative method is used on adapted grids. The alternative method encloses each surface triangle in a bounding box. These bounding boxes are stored in an Alternating Digital Tree (ADT) [34] for fast searches. The alternative wall distance method finds the closest surface triangle for adapted unstructured grids.

B. FUN3D-SFE Flow Solver

FUN3D-SFE is a continuous Stabilized Finite-Element discretization within FUN3D [35]. The discretization is based on a stabilized finite-element approach that includes the Streamlined Upwind Petrov-Galerkin (SUPG) scheme [36,37], Galerkin least squares [38], and variational multiscale methods [39]. In the results shown here, only the SUPG scheme is considered.

In the current implementation, the SA-neg turbulence model is tightly coupled with the flow equations, yielding a nonlinear algebraic system of equations with six variables at each vertex. The alternate wall distance calculation method described in the previous FUN3D-FV section is also used with FUN3D-SFE. A linear nodal basis is used in this study, which is designed to be second-order accurate in space. The current implementation includes the capability for computing on tetrahedra, hexahedra, pyramids, and prisms, although all the results shown in the present paper are for pure tetrahedral grid construction.

To advance the solution toward a steady state, the density, velocities, temperature, and the turbulence working variable are updated in a tightly-coupled Newton-type solver described by Anderson, Newman, and Karman [35]. Here, an initial update to the flow variables is computed using a locally varying time-step parameter that is later multiplied by the current CFL number, which is adjusted during the iterative process. At each iteration, a linearized residual matrix is formed and solved using the generalized minimal residual (GMRES) [40] algorithm with a preconditioner based on an incomplete lower upper (LU) decomposition with two levels of fill [41] and a Krylov subspace dimension of 300.

Using the full update of the variables, the $L_2$ norm of the unsteady residual is compared to its value at the beginning of the iteration. If the $L_2$ norm after the update is less than one half of the original value, the CFL number is doubled and the iterative process continues to the next iterative cycle. If the $L_2$ reduction target for the residual is not met, a line search is conducted to determine an appropriate relaxation factor. Here, the $L_2$ norm of the residual is determined at four locations along the search direction and the optimal relaxation factor is determined by locating the minimum of a
cubic polynomial curve fit through the samples. After the line search, the solution is updated using the relaxation factor and the CFL number is neither increased nor decreased.

The FUN3D-SFE solution variables include density, velocity components, and temperature. These variables differ from those normally stored in the perfect gas compressible FUN3D-FV discretization, which uses nondimensional conserved variables. The choice in solving for temperature directly in the finite-element discretization has been made to facilitate future computations of real-gas flows where the equation of state is invariably given directly in terms of density and temperature. At the completion of the solution process, the finite-element solution variables are converted to conserved variables and passed back into the FUN3D-FV solver, which subsequently computes the forces using previously developed routines. The forces are computed with FUN3D-FV for the ONERA M6 with strong, noslip boundary conditions. The Laminar Delta Wing uses weak boundary conditions, so the forces are computed directly from the residual routines within FUN3D-SFE.

C. WOLF Flow Solver

WOLF is a vertex-centered (flow variables are stored at vertices of the mesh) mixed finite-volume and finite-element Navier-Stokes solver on unstructured meshes composed of triangles in 2D and tetrahedra in 3D. The convective terms are solved by the finite-volume method on the dual mesh composed of median cells. It uses the HLLC approximate Riemann solver to compute the flux at the cell interface. Second order space accuracy is achieved through a piecewise linear interpolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure, which uses a particular edge-based formulation with upwind elements. A specific low dissipation scheme is considered using combination of centered (edge gradient) and upwind gradients (element gradient). A dedicated slope limiter is employed to damp or eliminate spurious oscillations that may occur in the vicinity of discontinuities. The viscous terms are solved by the $P^1$ Galerkin finite element method (FEM) which provides second order accuracy.

The implicit temporal discretization considers the backward Euler time-integration scheme. At each time step, the linear system of equations is approximately solved using a Symmetric Gauss-Seidel (SGS) implicit solver and local time stepping to accelerate the convergence to steady state. A Newton method based on the SGS relaxation is very attractive, because it uses an edge-based data structure which can be efficiently parallelized. From our experience, we have made the following crucial choices to solve the compressible Navier-Stokes equations:

- The residual of the linear system is reduced by two orders of magnitude by SGS relaxation.
- Breadth-first search renumbering improves the convergence rate of the implicit method and increases overall efficiency.
- Fully differentiating the HLLC approximate Riemann solver and the FEM viscous terms is very advantageous.
- A clever strategy to specify the time step is required to achieve high efficiency, automation, and robustness in the solution of the non-linear system of algebraic equations to steady-state.

For the turbulence model, the negative Spalart-Allmaras (SA-neg) is loosely-coupled to the mean-flow equations, where the mean-flow and turbulence model equations are relaxed in an alternating sequence. The flow solver WOLF is thoroughly detailed in [27, 42] with all the associated bibliography.

As regards the adjoint state computation, needed for goal-oriented error estimates, the matrix of the linear system is simply the implicit matrix transposed and the right hand-side of the system is the chosen functional (e.g., drag, lift) exactly differentiated. In particular, for viscous flows, $\mu$ and the stress tensor $\tau$ are exactly differentiated. To solve the adjoint system, we use a restarted GMRES preconditioned with LUSGS relaxation. It is important to solve the adjoint linear system to machine precision to obtain an accurate adjoint state for mesh adaptation.

D. GGNS Flow Solver

GGNS (General Geometry Navier-Stokes) is a Boeing-developed flow solver built upon the Streamline Upwind/Petrov Galerkin (SUPG) stabilized finite element (FE) discretization. The code uses piecewise linear finite elements resulting in a second order accurate discretization. Additional first-order artificial viscosity built upon the nodal DG(0) discretization is added for shock capturing. The indicator triggering this additional stabilization is based on the oscillation of the Mach number across a cell. The solver can work with unstructured grids of mixed-element type (tetrahedrons, prisms, and pyramids) as well as pure tetrahedral grids. The number of degrees of freedom for the second-order SUPG scheme is equal to the number of nodes in the computational grid. The discretization is “node-based” in the sense that it is conservative over the dual volumes of an unstructured grid. More details on discretization used in the GGNS solver, including the particular choices of discretization variables and special treatment of the essential boundary conditions via the Lagrange-multiplier based technique [43], can be found in Kamenetskiy et al. [44].
The discrete nonlinear solver in the GGNS code implements a variant of the Newton-Krylov-Schwartz algorithm. On the code level, this is accomplished using the Portable, Extensible Toolkit for Scientific Computation (PETSc) [45–47] solver framework. Time stepping is employed to drive to the steady state solution. On each time step, an exact Jacobian matrix for the discretization is formed by an automatic differentiation technique. The linear system arising from the Newton’s method is approximately solved using GMRES with a drop-tolerance-based BILUT preconditioner (locally on subdomains) implemented in the context of the additive Schwartz method with minimal overlap [41]. Right preconditioning is employed to maintain consistency between the nonlinear and linear residuals. The compact stencil property of the SUPG scheme helps to reduce the fill-in levels in the approximate factorization, thereby reducing the memory footprint.

A line search is applied along the direction provided by the approximate solution of the linear system. Residual decrease and physical realizability of the updated state are tracked during the line search. A heuristic feedback algorithm is implemented to communicate failure of the line search back to the time-stepping algorithm, so that the Courant-Friedrichs-Lewy (CFL) number can be increased or decreased as necessary. There is no upper preset limit for the CFL number in the time-marching algorithm; so Newton-type quadratic convergence (or, at least, superlinear, due to inexact linear solves) is routinely achieved at steady state.

E. SANS Flow Solver

Approximate solutions to the RANS equations are computed here with the Solution Adaptive Numerical Simulator (SANS) [48] software framework currently under development at the Massachusetts Institute of Technology. SANS a general framework for solving discrete finite element approximations to partial differential equations. A range of finite element methods are currently implemented in SANS, including high-order discontinuous (DG) and continuous (CG) Galerkin finite element methods.

The nonlinear system of FEM equations is solved using pseudo time continuation (PTC) damped Newton’s method with a line search to ensure residuals decreased. The complete linearization of the residual is computed via operator overloaded automatic differentiation [49]. The PTC algorithm computes an element local time step based on the characteristic speed, element size, and a CFL number. The inverse CFL is driven towards zero such that Newton like convergence rate are recovered. PETSc is used to solve the linear system for each PTC iteration restarted GMRES preconditioned with ILU factorization. For parallel solutions, we utilize the restricted additive Schwarz preconditioner with a single layer of overlap. The ILU preconditioner is applied to each subdomain, and restarted GMRES is applied to the global system. Adjoint systems are solved utilizing the same linear solver as the primal. All discrete solutions are converged to near machine-zero residuals.

F. MOESS Output Metric

The Metric Optimization via Error Sampling and Synthesis (MOESS) [50,51] adaptation algorithm is based on the continuous mesh framework developed by Loseille and Alauzet [26]. Galbraith, Allmaras, and Darmofal [48] detail the implementation of MOESS in SANS. The output error estimation method used within MOESS is the Dual Weighted Residual (DWR) method, as originally devised by Becker and Rannacher [52]. DWR consists of computing an adjoint solution for an output functional in an enriched solution space, and weighting it against the residual to give an estimate of $J(u) - J(u_{h,p})$ for some output functional $J(u)$. Originally devised for the Continuous Galerkin (CG) discretization, this approach was then extended by Carson et al. [53] for the DGBR2 discretization wherein an additional adjoint for the BR2 lifting operator was developed to account for errors in the lifting operator residual. The CG local error estimation used in SANS is based on the work of Richter and Wick [54] who modified the DWR estimate to make use of a nodal partition of unity to localize the error estimate using the weak form residual.

The MOESS algorithm [51] constructs a set of elemental local models that approximate the change in a localized error estimate for output of interest as a function of a step matrix change to the implied metric of the grid. These local models are fit from local solves where an element $\kappa$ is locally refined, either isotropically or by splitting an edge, and an approximate solve is performed. A local solve consists of fixing the DOF outside of a patch, $\omega_\kappa \supseteq \kappa$ whilst allowing those in the patch to vary. For a DG local solve $\omega_\kappa \equiv \kappa$ but for a CG local solve $\omega_\kappa$ consists of the elements attached to the vertices of $\kappa$. The localized error estimate is then reevaluated using the result of this local solve, which produces a change in the error estimate as a function of the step matrix change to the element. The local solve is repeated for multiple different refinements of the element and the results for all the local refinements are synthesized into a local model. Using the local models, an optimal step matrix change is found which minimizes the estimated error subject to a maximum DOF constraint as well as constraints on the magnitude of the entries of the step matrices. The optimized
step matrices are applied to the implied metric to create the new metric request that is supplied to a mesh generator.

**G. refine Multiscale Metric**

To form the metric, a Hessian of the scalar field is reconstructed by a k-exact quadratic reconstruction \[55\] or recursive application of \(L^2\)-projection \[27\]. The k-exact reconstruction is formed over vertices that are neighbors of neighbors of the vertex having its Hessian reconstructed. The least-squares system is solved with QR factorization \[29\]. If the least-squares system is under determined or poorly conditioned, the reconstruction stencil is iteratively grown one additional layer until a well-conditioned least-squares system is formed. No special boundary treatment is employed, but the one-sided stencil created on boundaries can contribute to poor conditioning that is mitigated with reconstruction stencil growth.

The Hessian is also reconstructed by recursive application of \(L^2\)-projection gradient reconstruction scheme. The gradient is computed in each element and a volume-weighted average is collected at each vertex \[27\]. The second-derivative Hessian terms are formed by computing the reconstructed gradients of these gradients formed in the first pass. The mixed derivative terms of the Hessian are averaged. A special boundary treatment is employed. The reconstructed Hessian on the boundary is replaced with an extrapolation from neighboring interior vertex, which have a well-formed stencil.

The reconstructed Hessian is then diagonalized into eigenvalues and eigenvectors. The absolute value of the Hessian is formed by recombing the absolute value of the eigenvalues with eigenvectors to ensure the Hessian is symmetric positive definite. The metric at each vertex is scaled to control the \(L_p\) norm \[27\] with Eq. (2). The gradation of the metric field is limited isotropically in the metric space with the “metric-space-gradation” of \[56\]. The complexity is computed, and the metric is globally scaled to set its complexity to a specified value. The complexity Eq. (1) is evaluated discretely by assuming it is piece-wise constant in each median dual.

**H. WOLF+FEFLO.A Multiscale Metric**

Three methods are available to compute the Hessian: double \(L^2\)-projection \[27\], double weighted least square, and \(k\)-exact weighted least square methods. A similar implementation as section II.G is done. The double \(L^2\)-projection method is preferred for improved robustness, efficiency, and accuracy over other reconstruction methods.

In the presented results, we have considered the local Mach number as sensor and the \(L^2\) norm of the interpolation error to compute the multiscale metric \[27\]. A metric gradation process is applied to smooth the metric field following the “mixed-space-gradation” approach of \[56\].

**I. Firedrake+PRAgMaTic Multiscale Metric**

The PRAgMaTic remeshing library was integrated with the Firedrake solver suite \[57,58\]. In this study, PRAgMaTic is called via its Firedrake interface, and the metric computation is done with Firedrake.

A weak Finite-Element formulation for the Hessian is written:

\[
H = \nabla^2 u,
\]

hence

\[
\int_{\Omega} \left( \sigma \cdot H - \sigma \cdot \nabla^2 u \right) dV = 0, \forall \sigma \in \Sigma,
\]

where \(\sigma \in \Sigma\) are test functions. This is then integrated by parts into:

\[
\int_{\Omega} \left( \sigma \cdot H + \text{div}(\sigma) \cdot \nabla u \right) dV = \int_{\partial \Omega} n \cdot (\sigma \cdot \nabla u) dS,
\]

where \(n\) is the outward normal. The problem is specified using Firedrake’s high level language UFL, and discretized automatically. The non-linear numerical problem is then solved using the PETSc library. There is currently no additional treatment of the Hessian on the boundary.

The Hessian is then symmetrized by taking the average value of opposite non diagonal elements, and made positive by taking the absolute value of the eigen values. Then the multiscale metric described in \[26\] is computed, and the eigenvalues of the metric are truncated to a specified minimal size. Gradation of the metric sizes can be limited following the isotropic method from \[56\]; however, no gradation was performed for analytic scalar field cases.
J. GGNS+EPIC Multiscale Metric

The Mach Hessian for each element is evaluated from the flow solution by GGNS using a least-squares approach on an extended stencil. GGNS then passes the Hessian at each element to EPIC, which converts it to adaptation metrics via an element-centered modification of Alauzet and Loseille [27], which minimizes the $L_p$ norm of interpolation error of the scalar field for a given grid complexity. In this modification, each elemental Hessian is scaled to control the $L_p$ norm with Eq. (2). The global scaling factor, $D_{L_p}$, is initialized as Eq. (3). Limits are then applied to the resulting elemental metric as detailed in the EPIC description. When enabled, the metric gradation is limited. The complexity, Eq. (4), of the resulting elemental adaptive metric is computed and the global scale factor, $D_{L_p}$, is modified to better match the requested value. The metric is then iteratively recomputed until computed complexity is within a specified tolerance of the requested value. A continuous metric field is generated by Log-Euclidean [59] interpolation of the elemental metrics to the grid nodes.

K. GGNS+EPIC Output Metric

The GGNS output metric was introduced by Michal et al. [17]. For the steady-state system of Navier–Stokes equations in the conservative form and written in conservation variables [10]

$$(F_i(u))_{x^i} - (K_{ji}(u)u_{x^i})_{x^i} = 0,$$

let $u^{(h)} = u^{(h)}(x)$ denote numerical primal solution obtained from the stabilized FE discretization on a given grid and $z^{(h)} = z^{(h)}(x)$ — discrete adjoint solution corresponding to an output functional of interest. Here $x$ stands for a vector of spatial coordinates $x \in \mathbb{R}^d$, $x = (x^1, \ldots, x^d)$, $i = 1, d$, $d = 2$ or $3$; $F_i(u)$ and $K_{ji}(u)u_{x^i}$ are inviscid and viscous fluxes respectively.

The same system in the quasi-linear form looks as follows:

$$A_i(u)u_{x^i} - (K_{ji}(u)u_{x^i})_{x^i} = 0, \quad A_i(u) = (F_i(u))_{u}.$$  

Introducing the entropy set of variables $v = v(u), u = u(v), u(v(u)) \equiv u$ the quasi-linear system can be cast into the symmetric form:

$$\tilde{A}_i(v)v_{x^i} - (\tilde{K}_{ji}(v)v_{x^i})_{x^i} = 0, \quad$$  

where $\tilde{A}_i(v) = A_i(u(v))u_{v}$, $\tilde{K}_{ij}(v) = K_{ij}(u(v))u_{v},$

where the following symmetry properties are known to hold for the coefficient matrices: $\tilde{A}_i(v) = \tilde{A}_i^T(v)$, $\tilde{K}_{ij}(v) = \tilde{K}_{ji}^T(v)$. GGNS goal-oriented cell-based error indicator has the following form:

$$\varepsilon_K = \int_K \left| \tilde{A}_i(v^{(h)})z^{(h)}_{x^i} + \tilde{K}_{ji}(v^{(h)})z^{(h)}_{x^i} \right| \left| v^{(h)} - R_h v^{(h)} \right| dx,$$

with $R_h v^{(h)}$ standing for a high-order reconstruction of the primal solution $v^{(h)}$.

The factor involving adjoint solution in this local error estimate is interpreted as the "strong adjoint quasi-linear PDE" applied to a numerical adjoint solution. In practical implementation piecewise linear reconstruction is used to represent $z^{(h)}$ and to obtain its gradient. For the second order derivatives Zienkiewicz-Zhu type "patch recovery" reconstruction for the gradients [60] is first obtained at the nodes of the grid and then these are linearly interpolated inside the elements. The interpolation error for the primal solution $(v^{(h)} - R_h v^{(h)})$ is not evaluated directly, instead, we rely on a semi-heuristic approach when the Hessian of the Mach number field is provided to the adaptation module. Note, that this is the only source of metric anisotropy in the present approach; in particular, any information about the adjoint solution enters the error indicator through the isotropic weight. To evaluate Hessian of the primal solution the least-square approach based on an extended stencil reconstruction is involved resulting in the quadratic reconstruction $R_h u^{(h)}$. The error indicator (6) along with the described implementation has strong similarities to the well known Venditti–Darmofal [61] approach but is not exactly equivalent to the latter.

L. WOLF+FEFLO.A Goal-Oriented Metric

We have proposed a goal-oriented error estimate for the laminar Navier-Stokes equation in [62] which is the extension of the inviscid Euler equation goal-oriented error estimate [63]. The main advantage of these estimates in comparison

\[\text{Einstein summation notation adapted}\]
to other goal-oriented error estimates is that the anisotropy of the mesh appears naturally using the continuous mesh framework [26, 28]. From the comparison [17] of the above error estimate and the one of section II.K, we came up with the following new error estimate using integration by part and linearization:

\[
\| J(W) - J(W_h) \|_{L^1(\Omega_h)} \approx \int_{\Omega} |W^*| \left( |\nabla \cdot (F^E(W) - F^E(\Pi_h W)) - \nabla \cdot (F^V(W) - F^V(\Pi_h W))| \right) \, d\Omega \\
\leq \int_{\Omega} \left( |\nabla W^*| \left| F^E(W) - \Pi_h F^E(W) \right| + |W^*| \left| \sum_i \frac{\partial S_i}{\partial x_i}(W) - \frac{\partial S_i}{\partial x_i}(\Pi_h W) \right| \right) \, d\Omega \\
\approx \int_{\Omega} \sum_i \frac{\partial F_i}{\partial W} \cdot \nabla x_i W^* + K_{ij} H(W^*) \right| \cdot |W - \Pi_h W| \, d\Omega
\]

where \( J \) is the considered output functional, \( W \) is the conservative variables vector, \( W^* \) the associated adjoint state, \( F^E = (F_i) \) the convective fluxes, \( F^V = (S_i) \) the viscous fluxes, \( K_{ij} \) the viscous terms under matrix form and \( H(W^*) \) the Hessian of the adjoint state. The error estimate is a weighted sum of \( L^1 \) interpolation error on the conservative variables where the weights depend on the gradient and the Hessian of the adjoint state and on the convective and viscous fluxes.

**M. avro**

avro is a dimension-independent mesh adaptation package under development at the Massachusetts Institute of Technology and has been demonstrated on up to 4D mesh generation problems [64]. It is based on local cavity operators inspired by the work of Coupez [65] and Loseille [66] and uses a combination of edge split, edge collapse, edge swaps, facet ((d − 1)-simplex) swaps and vertex smoothing to conform to a prescribed metric. The emphasis in avro is to conform to a metric field requesting edge lengths that are within reasonable bounds as measured on the input mesh, which is often the case in an adaptive framework. In particular, it is designed to output a metric-conforming mesh when the input edge lengths are within \([1/\alpha, \alpha]\) for \( \alpha = 2 \) but also works well for \( \alpha = 4 \). avro associates each mesh vertex with an EGADS geometry entity in order to (1) check the topological validity of each mesh operator and (2) for projecting vertices to the geometry. Operators are sequenced by (1) collapsing short edges followed by (2) splitting long edges such that no short edges are further created. Swaps are interleaved within splits and collapses to (1) improve the quality of the mesh and (2) weave out of topological configuration that restrict the ability to split or collapse. These operators are also scheduled such that the target metric complexity is matched as best as possible.

**N. refine**

The refine open source grid adaptation mechanics package was developed by NASA. It is available via [GitHub](https://github.com/NASA/refine) under the Apache License, Version 2.0. It is designed to output a unit grid [26] in a provided metric field. The current version under development uses the combination of edge split and collapse operations proposed by Michal and Krakos [67]. Node relocation is performed to improve adjacent element shape. A new ideal node location of the node is created for each adjacent element. A convex combination of these ideal node locations is chosen to yield a new node location update that improves the element shape measure in the anisotropic metric [68]. Geometry is accessed through the EGADS [69] application program interface. Parallel execution is facilitated by EGADSLite [70].

**O. PRAGMaTic**

PRAGMaTic (Parallel anisotropic adaptive mesh toolKit) is an open source 2D and 3D anisotropic adaptation code developed as a C++ library at Imperial College London, [https://meshadaptation.github.io](https://meshadaptation.github.io). PRAGMaTic modifies the input grid through a series of local edge-based grid manipulations [15]. First, iterative applications of coarsening (edge collapse), edge/face swapping, and refinement (edge splitting) is used to optimize the resolution and the quality of the grid. Then, an element-shape-constrained Laplacian smoothing step fine-tunes the grid element shape measure. PRAGMaTic aims at generating quality grids for a wide range of numerical simulations, notably for geophysics applications, and it has been integrated with the PETSc library [60] and the Firedrake solver suite [57, 58].

---

9
The EPIC anisotropic grid adaptation process provides a modular framework for anisotropic unstructured grid adaptation that can be linked with external flow solvers. EPIC relies on repeated application of edge break, edge collapse, element reconnection and node movement operations to modify a grid such that element edge lengths match a given anisotropic metric tensor field. EPIC-ICS includes only edge insertion, edge collapse, and element swaps. The EPIC-ICSM variant, adds node movement. The metric field on the adapted grid is continuously interpolated from the initial metric field. Several methods are available to preprocess the metric so as to limit minimum and maximum local grid sizes, control stretching rates of grid size and/or anisotropy, and ensure smoothness of the resulting distribution. In addition, the metric distribution can be limited relative to the initial grid and/or to the local geometry surface curvature. The surface grid is maintained on an IGES geometry definition with geometric projections and a local regridding procedure. The adaptive grid mechanics are applied to the tetrahedral grid with the option to insert right angle prismatic or tetrahedral elements into the adapted grid near wall boundaries.

Q. FEFLO.A

FEFLO.A is a generic purpose adaptive mesh generator dealing with 2D, 3D, and surface mesh generation. It belongs to the class of metric-based mesh generator which aims at generating a unit mesh with respect to a prescribed metric field $M$. A mesh is said to be unit when composed of almost unit-length edges and unit-volume element.

The adaptive remesher is based a combination of generalized standard operators (insertion, collapse, swap of edges and faces). The generalized operators are based on recasting the standard operators in a cavity framework [66, 71]. Additional modifications on the cavity allow to either favor a modification, that would have been rejected with the standard operator, or to improve the final quality by combining automatically many standard operators at once. In addition, the CPU time is also improved and becomes independent of the current modification. The unit speed is around 20,000 points inserted or removed per second on Intel i7 architecture at 2.7 GHz. For robustness purpose, both the surface and the volume mesh are adapted simultaneously, and each local modification is checked to verify that a valid mesh is obtained. For the volume, the validity consists in checking that each newly created element has a strictly positive volume. For the surface, the validity is checked by ensuring that the deviation of the geometric approximation with respect to a reference surface mesh remains within a given tolerance.

The generation of a unit mesh is decomposed into two steps:

1) Generate a unit-mesh: the mesh modification operators are used in the goal to optimize the length of the edges in $M$.

2) Optimization: the mesh modification operators are used to improve the quality $Q_M$.

During surface remeshing, either a P3 background surface is used or a direct CAD queries is used. The CAD kernel is based on EGADS and OPENCASCADE [69].

III. Scalar Field Verification

Three analytic functions are provided to verify the implementation of the multiscale metric and the MOESS algorithm and to stress the mesh generator to match both isotropic and anisotropic metric fields. The three functions $\text{sinfun3}$, $\text{tanh3}$, and $\text{sinatan3}$, progress in anisotropy to stress the algorithms of each mesh generator. For each analytic scalar field, $s$, a discrete approximation, $s_{h,p}$, is computed via an $L^2$ projection,

$$0 = \int_{\Omega} \phi (s - s_{h,p}) \quad \forall \phi \in V_{h,p},$$

where $p$ signifies the polynomial degree of the discrete approximation. Sequences of successively refined grids are used to verify that the $L^2$-error

$$\eta = \sqrt{\int_{\Omega} (s - s_{h,p})^2},$$

asymptotically decays at a rate of $p + 1$. All functions are evaluated on a unit cube domain $\Omega \equiv [0, 1] \times [0, 1] \times [0, 1]$.

As the multiscale metric approximates linear interpolation error via a Hessian reconstruction, all results in this section use a continuous linear, $p = 1$, polynomial approximation of the scalar functions to compute multiscale metric fields. The refine multiscale metrics are limited to a gradation of 5.

Rather than targeting interpolation error, the MOESS algorithm seeks to minimize an error functional. Typically the DWR method is used as the error functional, but this only provides an approximate error estimate. In order to eliminate
the approximation in the error functional for this verification exercise, the MOESS algorithm is modified to minimize the square $L^2$-error directly. The local solves requires the error functional to be localized to each element, and the square $L^2$-error localized to the element $\kappa$ is simply

$$\eta_\kappa = \int_\kappa (s - s_{h,p})^2.$$  

(7)

The MOESS algorithm is exercised for both continuous and discontinuous finite element solution spaces. For a discontinuous solution space, the $L^2$ projection is decoupled between elements and thus, the local solves are exact (up to quadrature error). However, for a continuous solution space, $L^2$ projection is a global operation. In this case there remains some approximation in the local solve. As the MOESS algorithm does not make any assumption as polynomial degree, all sequences of grids for each scalar functions are produced with polynomial degrees $p = 1$, $p = 2$, and $p = 3$.

A series of target complexities is used to demonstrate the asymptotic $p + 1$ convergence rate for both the multiscale and MOESS generated metric fields. The there is no specific requirement on how this series of complexities is constructed, but the meshes must be fine enough to capture the asymptotic convergence rate. Here, the sequence element counts are derived using a discontinuous space from a sequence of DOF counts. As shown in Table 1, the number of elements for a given DOF count decreases with increasing $p$. The target complexity is set by multiplying an element count by the volume of a unit-length tetrahedron, i.e. $\sqrt{2}/12$. Ideally, if the generated meshes have perfect unit-length tetrahedron as measured under the metric, the resulting grid should match the target number of elements. However, because unit-length tetrahedron do not tile or fill space, the observed ratio of complexity to elements in practice is typically somewhere around $1/12$ or $1.1/12$ as demonstrated in Refs. [28] and [14]. Thus, the element count of the generated meshes is not expected to perfectly match the target element count, but it should be close.

Nearly all the results in the following sections use SANS to compute the $L^2$ projections. The exception are the meshes generated with PRAgMaTic, where the $L^2$ projection is computed with Firedrake. Thus, some differences between PRAgMaTic and other mesh generators can be attributed to differences between SANS and Firedrake. The multiscale metrics are all generated using the algorithm implemented with refine or Firedrake, and MOESS metrics are generated using the algorithm implemented in SANS. All adaptation sequences for a given complexity start with a uniform mesh with 750 elements (uniform $5 \times 5 \times 5$ hexes divided into tetrahedron). The adaptation process is repeated 30 times with the given complexity.

<table>
<thead>
<tr>
<th>DG Elements</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DOF</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
</tr>
<tr>
<td>4,000</td>
<td>1,000</td>
<td>400</td>
<td>200</td>
</tr>
<tr>
<td>8,000</td>
<td>2,000</td>
<td>800</td>
<td>400</td>
</tr>
<tr>
<td>16,000</td>
<td>4,000</td>
<td>1,600</td>
<td>800</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2,048,000</td>
<td>512,000</td>
<td>204,800</td>
<td>102,400</td>
</tr>
</tbody>
</table>

Table 1 Discontinuous Galerkin element counts for given DOF counts
Two reasonable choices as the representative length scale to measure the error convergence rate are $1/\sqrt{\text{DOF}}$ and $1/\sqrt{\kappa}$. For a finite element discretization, the DOF count represents the cost to solve the linear system of an implicit discretization, and the element count is proportional to the residual/Jacobian evaluation cost. Since the purpose of this section is verification of the metric calculation and mesh generation, the convergence rate is illustrated using the number of element. Using the element count also more tightly clusters the data when comparing continuous and discontinuous solution spaces as illustrated in Fig. 2. In addition, the target element count can be exactly represented with the dashed vertical lines in Fig. 2(a) where as the target DOF count for the continuous space can only be approximated.

The figures in this section that include multiple mesh generators show the averages of last five $L^2$-error values in the adaptation sequence. The figures that only include one mesh generator show the last five $L^2$-error values in the adaptation sequence as demonstrated in Fig. 2. The lines in these figures also connects the average errors values between target complexities. All figures with $L^2$-errors include reference lines illustrate the expected convergence rates.

A. sinfun3

The sinfun3 function, given in Eq. (8) and shown in Fig. 3, has smooth variation that is mostly isotropic.

$$xyz = (x - 0.4)(y - 0.4)(z - 0.4)$$

$$s = \begin{cases} 
0.1 \sin(50.0 \ xyz) & \text{if } xyz \leq -1.0\pi/50.0 \\
\sin(50.0 \ xyz) & \text{if } xyz \leq 2.0\pi/50.0 \\
0.1 \sin(50.0 \ xyz) & \text{else}
\end{cases}$$  \hspace{1cm} (8)

The convergence of the $L^2$-error using all the mesh generators using the multiscale algorithms are shown in Fig. 4. The $L^2$-errors are all converging at the expected $2^{nd}$-order rate. Little differences can be observed between the different mesh generators or the two multiscale algorithms implemented with refine and the multiscale metric computed with Firedrake, as is expect on this smooth problem. Similar results are observed using the MOESS algorithm as shown in Fig. 5. All $L^2$-errors are converging at the expected rate of $p + 1$. Separate convergence rates and grids generated with the respective mesh generators are shown in Appendix A.

While the expect convergence rates were achieved, the grids adapted to the sinfun3 created neighbor of neighbor k-exact Hessian reconstruction stencils that were ill-conditioned. This prompted a reformulation of the k-exact reconstruction algorithm to improve the conditioning by growing the stencil to include additional layers in the refine multiscale metric implementation.
\[ xyz = (x - 0.4) \times (y - 0.4) \times (z - 0.4); \]
\[
\text{if } (xyz \leq (-1.0 \times \pi / 50.0)) \{ \\
\quad \text{scalar} = 0.1 \times \sin(50.0 \times xyz); \\
\} \text{ else if } (xyz \leq (2.0 \times \pi / 50.0)) \{ \\
\quad \text{scalar} = \sin(50.0 \times xyz); \\
\} \text{ else } \\
\quad \text{scalar} = 0.1 \times \sin(50.0 \times xyz); \\
\}

Fig. 3 sinh3 scalar function

(a) Convergence: reference line slope of 2.
(b) \( L^2 \) target 128k elements.
(c) k-exact target 128k elements.

Fig. 4 sinh3 \( L^2 \)-error multiscale metric convergence. Example meshes generated with refine.

(a) Convergence: reference line slopes of \( p + 1 \).
(b) P1 CG target 128k elements.
(c) P1 DG target 128k elements.

Fig. 5 sinh3 \( L^2 \)-error MOESS convergence. Example meshes generated with FEFLO.A.

B. tanh3

The \( \text{tanh}3 \) function, given in Eq. 9 and shown in Fig. 6 has strong anisotropic regions including a boundary layer feature on \( z = 0 \) face.
The convergence of the $L^2$-error using all the mesh generators using the multiscale algorithms are shown in Fig. 7. Due to the anisotropy in the tanh3 function, larger grid sizes are required to reach the 2$^{nd}$-order asymptotic rate. The outlier is the meshes generated with PRAgMaTIc using the Firedrake multiscale algorithm. From inspection of the grids show in Fig. 8, the Firedrake multiscale algorithm is deficient when extrapolating the metric to boundaries where higher levels of anisotropy are present. Since the Hessian is not recovered on the boundary, this was expected. Metric gradation helps recovering a nicer boundary mesh, but at the cost of an increased DOF count. Despite this deficiency, the $L^2$-error using the Firedrake multiscale metric is converging at a 2$^{nd}$-order rate, but at a significantly higher value than the errors using the refine multiscale metrics. This demonstrate the value of code-to-code comparisons to aid in identifying deficiencies.

The $L^2$-errors using the MOESS algorithm are shown in Fig. 9. The $L^2$-errors cluster reasonably well around the 2$^{nd}$-order reference line, but the clustering is not as clear around the 3$^{rd}$- and 4$^{th}$-order lines, but the rates are generally correct. The convergence rates for each mesh generator using both multiscale and MOESS metrics are shown in Appendix B.

The errors from refine using the CG space are significantly higher relative to the other mesh generator. As an example, the convergence rates obtained with refine compared with those obtained with EPIC-ICSM are shown in Fig. 10. Aside from the last point, the error rates computed with EPIC-ICSM agree well with the expected $p + 1$ rates. The refine results using the CG solution space are the only ones that exhibit a degraded rate and increased errors.

These higher error values appear to be due to isotropic elements in grids generated with refine near the $z = 0$ boundary as shown in Fig. 10. Notably, the elements in this region are anisotropic using refine with the multiscale metric as shown in Fig. 7. The other mesh generators do give the expected anisotropy using the MOESS metrics, e.g. the grid generated by EPIC-ICSM shown in Fig. 10. Thus, the issue here is likely due to the mesh generation algorithm in refine rather than an issue with the MOESS algorithm.

One possible cause for this differing behavior with refine might be contributed to how the metric fields are generated with the multiscale algorithm relative to the MOESS algorithm. The multiscale metric is roughly a fixed target metric, as it multiscale algorithm uses Hessians from the given scalar field which is mostly influenced by the accuracy of the scalar approximation. This means that that background grid that supports multicsale metric can have edge lengths as measured under the metric can have edge lengths of $O(10^3)$, and refine is tailored to process very long edges well. The MOESS algorithm however always computes a metric that is a deviation away from the current implied of the background grid. This means that the edge lengths of the background grid are typically in the less then 10. While refine works well for very long edges, it may have difficulties with initial grids with near unit-length edges.

$$s = \tanh((x + 1.3)^{20}(y - 0.3)^9z)$$

Fig. 6  tanh3 scalar function

```plaintext
scalar = tanh(pow(x + 1.3, 20.0) * 
pow(y - 0.3, 9.0) * z);
```
Fig. 7  \textit{tanh3} $L^2$-error multiscale metric convergence. Example meshes generated with \textit{refine}.

Fig. 8  Pragmatic generated meshes using multiscale metrics with target 128k elements.

Fig. 9  \textit{tanh3} $L^2$-error MOESS convergence. Example meshes generated with \textit{FEFLO.A}.
C. \text{sinatan3}

The \text{sinatan3} function, given in Eq. \eqref{eq:sinatan3} and shown in Fig. \ref{fig:6}, has a strong curved anisotropic region with a low amplitude smooth background variation.

\begin{equation}
    s = 0.1 \sin(50xz) + \frac{\text{atan}(0.1/(\sin(5y) - 2xz))}{\sin(5y) - 2xz)}
\end{equation}

The $L^2$-error convergence using the multiscale metric is shown in Fig. \ref{fig:12}. Note that the scale of the vertical axis differs from the previous figure as the $L^2$-error is larger. The strong anisotropy in the \text{sinatan3} function again requires larger grid sizes to nearly observe the asymptotic $2^{nd}$-order rate. For the grids shown in Fig. \ref{fig:12} the k-exact multiscale metric does not appear to detect the smooth background variation. However, previous calculations with approximately 5 million elements do show that the k-exact metric does eventually capture the background variation. The $L^2$ multiscale metric does detect the background variation, which is likely why the $L^2$-error values using $L^2$ metric is generally slightly smaller relative to the k-exact metrics.

As shown in Fig. \ref{fig:13}, this function is likely not regular enough for the higher order solution spaces to converge at the expected $p + 1$ rate. As a result, the $L^2$-error converges at approximately $2^n d$-order rate independent of $p$. Like the \text{tanh3} function, the results from \text{refine} are outliers. As shown in Fig. \ref{fig:14}, \text{refine} is not creating meshes with the expected anisotropy with the MOESS metric. This again appears to be an issue with combining MOESS with \text{refine} as the grids generated with EPIC-ICSM using MOESS metrics exhibit the expected anisotropy as shown in Fig. \ref{fig:14} and the grids generated with \text{refine} using multiscale metrics have the expected anisotropy as shown in Fig. \ref{fig:12}.
The convergence rates for each mesh generator using both multiscale and MOESS metrics are shown in Appendix B.

\[
\begin{align*}
\text{eps} &= 0.1; \\
\text{scalar} &= 0.1 \times \sin(50.0 \times x \times z) \\
&\quad + \\text{atan}(\text{eps} / (\sin(5.0 \times y) - 2.0 \times x \times z));
\end{align*}
\]

**Fig. 11** sinatan3 scalar function

**Fig. 12** $L^2$-error multiscale metric convergence. Example meshes generated with refine.

**Fig. 13** sinatan3 $L^2$-error MOESS convergence. Example meshes generated with FEFLO.A.
IV. Scalar PDE Verification: Triple Boundary Layer (TripleBL)

The triple boundary layer (TripleBL) function, $u$, represents a simplified version of the type of anisotropic boundary layer solutions that often occur in fluid flow simulations. Such anisotropic features are an ideal candidate for anisotropic metric based adaptation and as such a correctly implemented metric based mesher ought to be able to resolve $u$.

The (TripleBL) function (shown in Fig. 15) is given by the expression

$$u(x, y, z) = 1 - \frac{1 - \exp\left(-\frac{a(x-1)}{\nu}\right)}{1 - \exp\left(-\frac{b(x-1)}{\nu}\right)} \frac{1 - \exp\left(-\frac{c(x-1)}{\nu}\right)}{1 - \exp\left(-\frac{c}{\nu}\right)}$$

where $\nu = \frac{1}{30}$ and $a = b = c = 1$. Unlike the previous scalar functions, the TripleBL function is an analytic solution to the linear 3D Advection-Diffusion (AD) PDE,

$$\nabla \cdot \left( \bar{V} u - \nu \nabla u \right) = 0, \quad \bar{V} = \begin{pmatrix} a \\ b \\ c \end{pmatrix},$$

with Dirichlet boundary conditions on all boundaries of the unit cube. This corresponds to $u(1, y, z) = u(x, 1, z) =$
which for the analytic solution is

$$J_a(w) = \int_\Omega w,$$

which for the analytic solution is

$$J_a(u) = 1 - \frac{1}{abc} \frac{\left(e^{a/\nu} (a - \nu) + v\right) \left(e^{b/\nu} (b - \nu) + v\right) \left(e^{c/\nu} (c - \nu) + v\right)}{(e^{a/\nu} - 1) (e^{b/\nu} - 1) (e^{c/\nu} - 1)}.$$ (14)

The second output functional, $J_b$, is an integral of the flux on the $x = 1$ boundary of the domain, i.e.

$$J_b(w) = \int_{\partial \Omega} \left( \mathbf{v} \cdot \mathbf{n} - \mathbf{v} \nabla w(1,y,z) \right) \cdot \mathbf{n} \, dydz,$$

which for the analytic solution is

$$J_b(u) = a \left(1 - \frac{e^{a/\nu} \left(e^{b/\nu} (b - \nu) + v\right) \left(e^{c/\nu} (c - \nu) + v\right)}{bc \left(e^{a/\nu} - 1\right) \left(e^{b/\nu} - 1\right) \left(e^{c/\nu} - 1\right)} \right).$$ (16)

For adjoint consistent discretizations, a convergence rate of $2p$ is expected for output functionals. \[72\]

To verify convergence rates, multiscale metrics are computed from the discrete $p = 1$ unstabilized CG solutions with the increasing target complexities. Both $L^2$-errors as well as errors in $J_a$, i.e. $|J_a(u_{h,p}) - J_a(u)|$, are expected to converge at a $2^{nd}$-order rate. The convergence rates of the errors computed from grids generated with avro, EPIC-ICSM, feflo.a, and refine using the refine $L^2$ and k-exact algorithms are shown in Fig. [16]. The $L^2$-error converges for all mesh generators at a $2^{nd}$-order rate. The convergence rate for the $J_a$-error is a bit more noisy, but the trend is generally $2^{nd}$-order. The meshes generated with refine also shown in Fig. [16] are representative of all the meshes created with the mesh generators. Convergence rates for the individual mesh generators as well as figures of the generated meshes are shown in Appendix [D].

Convergence rates of $L^2$- and $J_a$-error computed using $J_a$ as the functional for the DWR error estimate in the MOESS algorithm for each mesh generator is shown in Fig. [17]. The reference lines for the $L^2$-error have slopes of $p + 1$, and the reference lines for $J_a$-error have slopes of $2p$. Unfortunately the errors are too noisy to decipher the trends of all mesh generators at once. Focusing on the errors computed with the EPIC-ICSM mesh generator shown in Fig. [18] the slopes in the $J_a$-error roughly follow the expected rates of $2p$. Because the $J_a$ is a volume functional, and MOESS is minimizing the error in this functional, the $L^2$ also converges at rate the expected rate of $p + 1$. The errors for each mesh generators are shown in Appendix [D].

Errors in $L^2$ and $J_b$ using $J_b$ as the functional for the DWR error estimate in the MOESS algorithm for each mesh generator is shown in Fig. [19]. Again focusing on results with the EPIC-ICSM shown in Fig. [20] the $J_b$-error is converging approximately at expected rate of $2p$. However, unlike for the functional $J_a$, minimizing the error the boundary functional $J_b$ results in grids clustered to the $x = 1$ domain boundary as shown in Figs. [19] and [20]. As a result, the $L^2$-error is not expected to converge.
nu = 1.0/30.0;
\[ a = b = c = 1.0; \]
\[ f = \frac{1 - \exp(-a^*(1 - x)/nu)}{1 - \exp(-a/nu)}; \]
\[ g = \frac{1 - \exp(-b^*(1 - y)/nu)}{1 - \exp(-b/nu)}; \]
\[ h = \frac{1 - \exp(-c^*(1 - z)/nu)}{1 - \exp(-c/nu)}; \]
\[ \text{scalar} = 1 - f*g*h; \]

**Fig. 15** TripleBL scalar function

**Fig. 16** \(L^2\)-error and \(J_n\)-error multiscale metric convergence. Example meshes generated with `refine`.

**Fig. 17** TripleBL \(L^2\)-error and \(J_n\)-error MOESS convergence. Example meshes generated with FEFLO.A.
(a) Convergence: reference line slopes of $p + 1$ and $2p$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 18 TripleBL $L^2$-error and $J_u$-error MOESS convergence and grids using EPIC-ICSM.

(a) Convergence: reference line slopes of $p + 1$ and $2p$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 19 TripleBL $L^2$-error and $J_b$-error MOESS convergence. Example meshes generated with FEFLO.A.

(a) Convergence: reference line slopes of $p + 1$ and $2p$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 20 TripleBL $L^2$-error and $J_b$-error MOESS convergence and grids using EPIC-ICSM.
V. Laminar Delta Wing

The 3D Laminar Delta Wing case is described by Wang et al. [10] as part of its inclusion in the first three International Workshops on High-Order CFD Methods (HIOCFD). This case was denoted BTC3 in the ADIGMA (Adaptive Higher-order Variational Methods for Aerodynamic Applications in Industry) project [73, 74] where adaptive results are detailed in Hartmann et al. [75]. Adaptive results were originally presented by Leicht and Hartmann [76]. Yano [50] published grid adapted results with the MOESS metric. The geometry is a modified from the original description in Klaij, van der Vegt, and van der Ven [77] based on the experiment of Riley and Lowson [78] to match the HIOCFD and ADIGMA case. The dimensions of the Delta Wing are provided in Table 2. The domain is constructed from planar facets, which eliminates the need for curved boundary recovery from the adaptive grid mechanics and curved elements for higher-order solution representations.

<table>
<thead>
<tr>
<th>Reference geometry for the Delta Wing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Chord</td>
</tr>
<tr>
<td>Span</td>
</tr>
<tr>
<td>Thickness</td>
</tr>
<tr>
<td>Reference Area</td>
</tr>
</tbody>
</table>

The freestream conditions are 0.3 Mach, 4,000 Reynolds number based on root chord length, and 12.5° angle of attack. Isothermal noslip boundary conditions are set to freestream temperature. The Prandtl number is 0.72. The viscosity is assumed to be constant to match the workshop case. An example adapted solution is shown in Fig. 21 computed by FUN3D-SFE. The symmetry plane is colored with Mach number, the delta wing is colored with coefficient of pressure. The footprint of the leading edge vortex can be seen in upper surface pressure coefficient. A transparent Mach=0.2 isosurface shows the vortex roll up downstream of the configuration. Adapted results by Park et al. [14] use FUN3D-FV with a slightly different noslip wall temperature and a variable viscosity based on Sutherland’s law. FUN3D-FV could not obtain steady-state solutions in the current study because the Hierarchical Adaptive Nonlinear Method (HANIM) [79] solution scheme required for this case has been removed from FUN3D-FV since the previous study. All the flow solvers in the current study have a Newton solution update scheme. A 50,000 complexity metric field was extracted from the Laminar Delta Wing for a parallel execution study by Chrisochoides et al. [80], where the 50,000 complexity metric field is globally scaled to finer complexities to provide a target grid sizes to establish strong and weak scaling trends.

The drag and lift coefficient trajectories are shown in Fig. 22 with a vertical range that shows all adapted-grid trajectories. The spacing $h$ is estimated $h = N^{-1/3}$, where $N$ is the degrees of freedom of the solution. The SANS

![Image of Laminar Delta Wing with Mach and pressure coefficient coloring]
adaptive trajectories show an average force coefficient value for a number of fixed-complexity adaptations, but the other methods show the last grid at a given complexity. The trajectories that extend to $h < 0.01$ appear to be converging toward the same fine grid values. The trends of some of the trajectories that end with $h > 0.01$ are less consistent. The reference values of the drag and lift coefficients computed for the ADIGMA project are 0.16608 and 0.34865 in [75] and 0.1658 and 0.347 in [76]. The trajectories here indicate a slightly lower drag and a lift within the range provided by the two reference values. Larger grids are used for the adaptive results here than the previously published results from HIOCFD or ADIGMA.

![Fig. 22](image1.png)  
**Fig. 22** Lift and drag coefficient, wide scale.

A zoom of the finer adapted grid solutions is shown in Fig. 23. The addition of node movement to GGNS+EPIC drag adaptation reaches fine-grid earlier. There are no consistent trends when varying the norm exponent of the multiscale metric shown by GGNS+EPIC and FUN3D-SFE+refine. At $h = 0.0048$ (about 5,000,000 complexity or about 11,000,000 vertex), the drag coefficient is in the range 0.16561–0.16566 (half a drag count variation) and the lift coefficient is in the range 0.34722–0.34745 (0.07% variation).

![Fig. 23](image2.png)  
**Fig. 23** Lift and drag coefficient, narrow scale.

All the subiterations for the finer adapted grid solutions is shown in Fig. 24. The multiscale metric trajectories have more jitter at a fixed complexity and a steeper slope than the lift and drag adaptation trajectories. The addition of node movement to GGNS+EPIC drag adaptation reduces the jitter on coarser grids. The relatively larger jitter shown by FUN3D-SFE+refine may be due to inadequate metric conformity at these low error levels. SANS-GLS with refine grid
mechanics and refine multiscale metric has the largest jitter at a fixed-complexity iteration.

### Fig. 24  Lift and drag coefficient, narrow scale, with subiterations.

The convergence of Mach interpolation error is shown in Fig. 25 for the FUN3D-SFE Laminar Delta Wing cases. The “truth” solution used to measure interpolation error is the final mesh in the series. To compute the error, the Mach field on a candidate grid is linearly interpolated to the truth grid. The 2-norm and 4-norm (as shown in parenthesis of the figure legend) of the difference is computed for the series of multiscale metric adapted grids with 2 and 4 exponents. All trajectories show the expected second-order convergence.

### Fig. 25  Mach interpolation error, with subiterations.
VI. ONERA M6

The ONERA M6 wing was originally described in an AGARD report \cite{81}. The geometry has been modified from this original description to have a sharp trailing edge and a well-defined wing tip shape by Mayeur et al. \cite{82, 83}. Test Case 2308 is simulated at 0.84 freestream Mach number, 14.6 million Reynolds number based on root chord, 3.06° angle of attack, and 540 °R freestream static temperature. The root chord is unity, the reference area is 1.15315084119231, and the reference pitching moment length is 0.801672958512342 with pitching moment computed about the leading edge of the root chord.

Uniformly-refined, fixed-grid results obtained from the TMR indicate the expected force values for well-resolved grids as solid lines in the following figures, beginning with Fig. 26. These grids are described by Nishikawa and Diskin \cite{84}, which includes images of the surface grids of the uniformly-refined series. These ONERA M6 grids have a well-resolved leading edge, trailing edge, wing tip, and boundary layer with an O-grid construction. The finest fixed grids have 61 million vertices, $h = 0.00254$. The chordwise resolution on the adapted grids is higher than the chordwise resolution of the fixed grids.

![Fig. 26 ONERA M6 force and moment convergence, last at complexity, all trajectories.](image)

(a) Pressure component of drag coefficient.  
(b) Viscous component of drag coefficient.  
(c) Total lift coefficient. 
(d) Total pitching moment coefficient.

All the adapted and fixed grids are shown in Fig. 26. This figure is complex due to the large number of different adaptive procedures presented, and is intended to provide a general impression of how adaptive methods are performing versus hand-tailored uniformly-refined grids. Subsequent figures will focus on subsets of these results to present the relative performance of the adaptive methods. The last grid at each fixed-complexity step is shown. All the grids in each of these fixed-complexity steps are shown in appendix \cite{F}. Second-order convergence in $h$ is not observed by the fixed-grid series \cite{9}. Diskin et al. \cite{9} tabulates differences in the force and moment coefficients on the finest fixed grid to
be less than one percent. The pressure and viscous drag coefficient components shown in Fig. 26(a and b) are in good agreement between the fixed-grid series and the adapted-grid trajectories. The adapted-grid lift and pitching moment trajectories are clustered near the FUN3D-SFE fixed grid series. The finite-volume methods are also approaching the FUN3D-SFE fixed grid series but at a slower rate than the adapted grids. The vertical scales of the adaptive-grid force and moment plots are narrower than Park et al. \[16\] or Michal et al. \[17\], which indices the consistency and accuracy of the group has improved in a year’s time.

The impact of multiscale norm exponent is shown in Fig. 27 for FUN3D-FV+, FUN3D-SFE+, GGNS+EPIC, and WOLF+FEFLO. The last grid at each fixed-complexity step is shown. All trajectories are trending to the fine fixed-grid values. The viscous drag coefficient converges faster for the 4-norm than the 2-norm. The finite-volume solvers with 2-norm multiscale metrics are the slowest to converge for lift and pitching moment coefficients, and the 4-norm multiscale metrics approach fine-grid values on coarser grids. It is possible that discretization error is too large at these grid sizes and lower discretization error is required to see clear trends between 2-norm and 4-norm multiscale metrics for the pressure drag coefficient. Otherwise, known and unknown differences in implementations may prevent the indication of clearer trends.

Metric formulations that target estimated error in an output functional (lift or drag) are shown in Fig. 28. The last grid at each fixed-complexity step is shown. In this case, total drag and lift coefficients are shown because they are the targets of the adaptive procedure. The pressure and viscous components of these forces are not targeted independently, only the total. A norm-oriented \[85\] metric could be formulated to target these individual components simultaneously. The lift and drag coefficients converge faster for these output metrics than for the multiscale metric. The multiscale
metric controls interpolation error of the solution, not error in forces directly. The output methods agree within 1 drag count and 0.0005 lift coefficient, which is a much smaller range than the fixed grid at a comparable size.

![Graph](image)

(a) Total drag coefficient.

(b) Total lift coefficient.

**Fig. 28** ONERA M6 force and moment convergence, last at complexity, output error.

**VII. Conclusions**

Some fields showed expected convergence rates.
Some fields showed convergence degradation for some metric construction methods
Scalar fields revealed subtle interactions of metric construction and grid adaptive mechanics
Laminar Delta Wing and turbulent RANS ONERA M6 showed very good agreement on finest adapted grids
ONERA M6 benchmarked with carefully hand-crafted fixed grids.
Marked improvement in consistency and accuracy beyond previous reports.

---

27
Appendix

A. *sinfun3* convergence rates and grids for each mesh generator

The Figs. 29 through 37 show the $L^2$-error converge rates using the *sinfun3* function for each mesh generator.

(a) Convergence: reference line with slope of 2.

(b) $L^2$ target 128k elements.

(c) k-exact target 128k elements.

**Fig. 29** *sinfun3* $L^2$-error multiscale metric convergence and grids using *refine*.

(a) Convergence: reference line with slope of 2.

(b) $L^2$ target 128k elements.

(c) k-exact target 128k elements.

**Fig. 30** *sinfun3* $L^2$-error multiscale metric convergence and grids using *FEFLO.A*.
(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 31  $\text{sinfun3}$ $L^2$-error multiscale metric convergence and grids using EPIC-ICSM.

(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 32  $\text{sinfun3}$ $L^2$-error multiscale metric convergence and grids using avro.

(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 33  $\text{sinfun3}$ $L^2$-error multiscale metric convergence and grids using PRAgMaTic.
(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 34  $\sinfun_3$ $L^2$-error MOESS convergence and grids using refine.

---

(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 35  $\sinfun_3$ $L^2$-error MOESS convergence and grids using FEFLO.A.

---

(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 36  $\sinfun_3$ $L^2$-error MOESS convergence and grids using EPIC-ICSM.
(a) Convergence: reference line slopes of $p + 1$.
(b) P1 CG target 128k elements.
(c) P1 DG target 128k elements.

Fig. 37 sinfun3 $L^2$-error MOESS convergence and grids using avro.
B. tanh3 convergence rates and grids for each mesh generator

The Figs. 38 through 46 show the $L^2$-error converge rates using the tanh3 function for each mesh generator.

Fig. 38  tanh3 $L^2$-error multiscale metric convergence and grids using refine.

Fig. 39  tanh3 $L^2$-error multiscale metric convergence and grids using FEFLO.A.
(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 40 \textit{tanh3} $L^2$-error multiscale metric convergence and grids using EPIC-ICSM.

(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 41 \textit{tanh3} $L^2$-error multiscale metric convergence and grids using avro.

(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 42 \textit{tanh3} $L^2$-error multiscale metric convergence and grids using PRAgMaTic.
(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

**Fig. 43** $L^2$-error MOESS convergence and grids using *refine*.

---

(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

**Fig. 44** $tanh3$ $L^2$-error MOESS convergence and grids using *FEFLO.A*.

---

(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

**Fig. 45** $tanh3$ $L^2$-error MOESS convergence and grids using *EPIC-ICSM*.
(a) Convergence: reference line slopes of $p+1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 46  \( \tanh^3 L^2 \)-error MOESS convergence and grids using \texttt{avro}. 
C. \text{sinatan3} convergence rates and grids for each mesh generator

The Figs. 47 through 55 show the $L^2$-error converge rates using the \text{sinatan3} function for each mesh generator.

Fig. 47 \text{sinatan3} $L^2$-error multiscale metric convergence and grids using \text{refine}.

Fig. 48 \text{sinatan3} $L^2$-error multiscale metric convergence and grids using \text{FEFLO.A}.
(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 49  $\sinat3$ $L^2$-error multiscale metric convergence and grids using EPIC-ICSM.

(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 50  $\sinat3$ $L^2$-error multiscale metric convergence and grids using avro.

(a) Convergence: reference line with slope of 2.
(b) $L^2$ target 128k elements.
(c) k-exact target 128k elements.

Fig. 51  $\sinat3$ $L^2$-error multiscale metric convergence and grids using PRAgMaTIc.
(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

**Fig. 52** $\sin \alpha + 3$ $L^2$-error MOESS convergence and grids using *refine*.

(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

**Fig. 53** $\sin \alpha + 3$ $L^2$-error MOESS convergence and grids using FEFLO.A.

(a) Convergence: reference line slopes of $p + 1$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

**Fig. 54** $\sin \alpha + 3$ $L^2$-error MOESS convergence and grids using EPIC-ICSM.
Fig. 55 $\sin \tan 3$ $L^2$-error MOESS convergence and grids using avro.
D. TripleBL convergence rates and grids for each mesh generator

The Figs. 56 through 63 show the $L^2$- and $J_u$-error converge rates using the TripleBL function for each mesh generator. Figures 64 through 67 show the $L^2$- and $J_b$-error converge rates.

Fig. 56  TripleBL $L^2$-error multiscale metric convergence and grids using *refine*.

Fig. 57  TripleBL $L^2$-error multiscale metric convergence and grids using FEFLO.A.
(a) Convergence: reference line with slope of 2.

(b) $L^2$ target 128k elements.

(c) k-exact target 128k elements.

Fig. 58  TripleBL $L^2$-error multiscale metric convergence and grids using EPIC-ICSM.

(a) Convergence: reference line with slope of 2.

(b) $L^2$ target 128k elements.

(c) k-exact target 128k elements.

Fig. 59  TripleBL $L^2$-error multiscale metric convergence and grids using avro.

(a) Convergence: reference line slopes of $p+1$ and $2p$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 60  TripleBL $L^2$- and $J_a$-error MOESS convergence and grids using refine.
Fig. 61 **TripleBL** $L^2$- and $J_u$-error MOESS convergence and grids using FEFLO.A.

Fig. 62 **TripleBL** $L^2$- and $J_u$-error MOESS convergence and grids using EPIC-ICSM.

Fig. 63 **TripleBL** $L^2$- and $J_u$-error MOESS convergence and grids using avro.
(a) Convergence: reference line slopes of $p + 1$ and $2p$.
(b) P1 CG target 128k elements.
(c) P1 DG target 128k elements.

Fig. 64 TripleBL $L^2$- and $J_h$-error MOESS convergence and grids using refine.

(a) Convergence: reference line slopes of $p + 1$ and $2p$.
(b) P1 CG target 128k elements.
(c) P1 DG target 128k elements.

Fig. 65 TripleBL $L^2$- and $J_h$-error MOESS convergence and grids using FEFLO.A.

(a) Convergence: reference line slopes of $p + 1$ and $2p$.
(b) P1 CG target 128k elements.
(c) P1 DG target 128k elements.

Fig. 66 TripleBL $L^2$- and $J_h$-error MOESS convergence and grids using EPIC-ICSM.
(a) Convergence: reference line slopes of $p + 1$ and $2p$.

(b) P1 CG target 128k elements.

(c) P1 DG target 128k elements.

Fig. 67 TripleBL $L^2$- and $J_b$-error MOESS convergence and grids using avro.
E. ONERA M6 Integrated Adaptation Trajectories

The force and moment trajectories summarized in section VI are shown here for each integrated adaptation scheme. All grids that constitute a fixed-complexity step are shown. This examination of integrated adaptation schemes begins with GGNS+EPIC in Fig. 68. Mach multiscale, lift output, and drag output metrics are shown with (EPIC-ICSM) and without (EPIC-ICS) node location optimization. The output-based metrics converged closer to the uniformly refined fixed-grid force and moment coefficients on coarser grids with less jitter at constant complexity than the multiscale metric. The addition of node optimization (EPIC-ICSM) resulted in slightly faster convergence to fine-grid coefficient, particularly for the viscous component of drag. The 2-norm exponent multiscale metric converged dramatically slower than the 4-norm exponent multiscale metric, which was very close to the output-based methods.

![Graphs](image)

(a) Pressure component of drag coefficient.
(b) Viscous component of drag coefficient.
(c) Total lift coefficient.
(d) Total pitching moment coefficient.

**Fig. 68 ONERA M6 force and moment convergence, GGNS+EPIC.**

The WOLF+FEFLO.A cases are shown in Fig. 69. The intermediate pitching moment coefficients are not available, only the final pitching moment at each fixed-complexity step is shown. The drag goal-based metric converges faster and more consistently to the final force and moment values than the multiscale metric. The pressure drag component approaches from above and the viscous drag component approaches from below, which provides cancellation of errors. The pressure and viscous components of these forces are not targeted independently by the drag goal-based metric, only the total. A norm-oriented [85] metric could target these individual drag components simultaneously.

The FUN3D-FV cases shown in Fig. 70 are constructed to explore the impact of multiscale norm exponent, hessian reconstruction method, and metric gradation limit. All grids that constitute a fixed-complexity step are shown. Perturbations are shown from a baseline case of 4-norm multiscale metric, K-exact Hessian reconstruction, and a metric gradation limit of 10. The combination of FUN3D-FV, refine grid adaptation mechanics, and the refine implementation
of the multiscale metric shows a saw-tooth force and moment coefficient trajectory, where the trend at constant complexity is opposite of increasing complexity. With the exception of the drag pressure component, The 2-norm exponent required a larger grid (smaller $h$) to reach the same values than the 4-norm exponent trajectories. All of the trajectories appear to be converging toward the fine-grid, fixed-grid refinement series values.

An improvement over previously reported FUN3D-FV results \[16\] is shown here, where the largest implementation difference is the metric construction. Larger grids and more trajectories could be attempted due to parallel geometry queries enabled by EGADS\textit{Lite}. In this study, identical error estimate and grid mechanics are used for FUN3D-FV and FUN3D-SFE solvers to isolate the impact of the solver. Previous comparisons \[16, 17\] varied the entire integrated grid adaptation system of flow solver, metric construction, and grid adaptation mechanics simultaneously.

The FUN3D-SFE cases shown in Fig. 71 are performed in the same perturbations as the FUN3D-FV cases to explore the impact of multiscale norm exponent, Hessian reconstruction method, and metric gradation limit. Again, all grids that constitute a fixed-complexity step are shown. Both FUN3D-FV and FUN3D-SFE share strong no-slip boundary and normal derivative based force calculation. FUN3D-SFE, \textit{refine} grid adaptation mechanics, and the \textit{refine} implementation of the multiscale metric is less sensitive to the multiscale options and approach the uniformly refined fixed-grid force and moment coefficients faster than FUN3D-FV.

The multiscale metric construction and gradation limit trends are consistent between FUN3D-FV and FUN3D-SFE, where the 2-norm exponent required a larger grid (smaller $h$) to reach the same values than the 4-norm exponent trajectories with the exception of the drag pressure component. The lower gradation limit of 1.5 seems to slow convergence at lower complexity levels than a gradation limit of 10.
Images of the symmetry plane and wing upper surface grids from the FUN3D-FV (Fig. 70) and FUN3D-SFE (Fig. 71) multiscale parameter studies are shown in Fig. 72. The upper row is FUN3D-FV and the lower row is FUN3D-SFE. Gradation and Hessian reconstruction method changes the wake, and varies the isotropic grid density at a distance greater than 5 chord lengths from the wing. An isotropic gradation limiting procedure is used. The isotropic nature of this “metric-space-gradation” limiting procedure extends the wake as a side effect of limiting the gradation normal to the wake sheet. The “mixed-space-gradation” approach of [56] is formulated to mitigate this unintended refinement but not currently implemented in refine. At similar complexity levels, more grid resolution is available to refine near the wing if it is not being used to resolve the wake multiple chord lengths downstream of the wing.

The differences in the grid of the upper surface shock on the symmetry plane is shown for FUN3D-FV (Fig. 70) and FUN3D-SFE (Fig. 71) in Fig. 73. There is a tighter clustering of the grid in the shock for FUN3D-FV than FUN3D-SFE. The FUN3D-FV reconstructed Hessian may have stronger second derivatives near the shock than FUN3D-SFE, which is increasing resolution in the streamwise direction.

**Fig. 70**  ONERA M6 force and moment convergence, FUN3D-FV+refine.
Fig. 71  ONERA M6 force and moment convergence, FUN3D-SFE+refine.
Fig. 72  FUN3D-FV and FUN3D-SFE ONERA M6 wing and symmetry plane grids, approx 750K vertex.
Fig. 73  FUN3D-SFE ONERA M6 wing and symmetry plane grids, approx 750K vertex.
Acknowledgments

Yi Liu provided the FUN3D-FV uniformly refined grid forces. This work was partially supported by the Transformational Tools and Technologies (TTT) Project of the NASA Transformative Aeronautics Concepts Program (TACP).

References


