Enhancements to Overset Methods for Achieving Higher Accuracy and Solution Convergence

Preprint

Dylan Jude,[1] Jayanarayanan Sitaraman,[2] and Michael Brazell[3]

1 NASA Ames Research Center
2 Parallel Geometric Algorithms LLC
3 National Renewable Energy Laboratory

Presented at the AIAA SciTech 2020 Forum
Orlando, Florida
January 6-10, 2020

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

Conference Paper
NREL/CP-2C00-78162
September 2022

Enhancements to Overset Methods for Achieving Higher Accuracy and Solution Convergence

Preprint

Dylan Jude,[1] Jayanarayanan Sitaraman,[2] and Michael Brazell[3]

1 NASA Ames Research Center
2 Parallel Geometric Algorithms LLC
3 National Renewable Energy Laboratory

Suggested Citation
Jude, Dylan, Jayanarayanan Sitaraman, and Michael Brazell. 2022. *Enhancements to Overset Methods for Achieving Higher Accuracy and Solution Convergence: Preprint*. Golden, CO: National Renewable Energy Laboratory. NREL/CP-2C00-78162. https://www.nrel.gov/docs/fy22osti/78162.pdf.

Conference Paper
NREL/CP-2C00-78162
September 2022

National Renewable Energy Laboratory
15013 Denver West Parkway
Golden, CO 80401
303-275-3000 • www.nrel.gov

NOTICE

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Material presented in this paper is a product of the CREATE-AV Element of the Computational Research and Engineering for Acquisition Tools and Environments (CREATE) Program sponsored by the U.S. Department of Defense HPC Modernization Program Office. M. Brazell was partially funded by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two DOE organizations (Office of Science and the National Nuclear Security Administration). The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

Enhancements to Overset Methods for Achieving Higher Accuracy and Solution Convergence

Dylan Jude[1], Jayanarayanan Sitaraman[2], and Michael Brazell[3]

[1]*Science & Technology Corporation, NASA Ames Research Center, Moffett Field, CA*
[2]*Parallel Geometric Algorithms LLC, Sunnyvale, CA*
[3]*National Renewable Energy Laboratory, Golden, CO*

Overset methods, when utilized within a multi-solver framework, provide a very flexible and powerful toolset for analyzing moving body problem. However, many overset framework suffer from several deficiencies such as (1) convergence degradation of the non-linear iterative schemes (2) difficulty in achieving higher-order accuracy and (3) high computational costs over long simulation times. In this work, we explore approaches that mitigate these deficiencies and provide a pathway for improved accuracy and fidelity of overset methods. A high-order, discontinuous Galerkin solver is integrated into an existing overset framework and applied to an unsteady ROBIN fuselage case. For convergence acceleration, an Overset-GMRES linear solver with a timestep controller is applied to a DLR-F6 aircraft steady simulation and an unsteady, notional rotor-fuselage. Finally the same rotor-fuselage case as well as the JVX rotor/wing case is simulated using a reduced-order aerodynamic model.

I. Introduction

The Helios and Kestrel software[1–5] are products of the HPCMP CREATE^{TM}-AV (air vehicles) program[6] that provides high-fidelity analysis capability to the Department of Defense (DoD) for the acquisition of new rotary-wing and fixed wing aircraft. A multi-mesh paradigm[7] is the basis of the CFD aerodynamics solution procedure, where a near-body solver is used to capture viscous flow around complex geometry, and block structured Cartesian solver[8] in the off-body is used to resolve the wake through a combination of high-order algorithms and adaptive mesh refinement (AMR). The PUNDIT[9,10] domain connectivity software facilitates parallel data exchange between the two mesh types as well as enables relative motion between the mesh systems. Coordination of the different codes is managed through a lightweight and flexible Python-based infrastructure.[11]

Helios is now in routine use at several DoD sites and US Helicopter companies. The usual mode of operation is to use a near-body mesh system composed of both structured and unstructured meshes – structured curvilinear meshes are used for simple geometries such as blades and unstructured or strand meshes are used for complex geometry such as fuselages and hubs. Any combination of structured grid based flow solvers (OVERFLOW,[12] HAMSTR[13]), unstructured grid based flow solvers (NSU3D,[14] FUN3D[15] or KCFD[16]) or strand grid based flow solvers[17] can be used to model the flow near the body. The flow away from a body is resolved using a high-order off-body Cartesian grid solver, SAMCart.[18] Helios is highly capable and is in production uses for various acquisition programs including the Joint Multi Role helicopter (JMR), CH-47 block-II upgrade and CH-53K risk analysis and design.

Helios has consistently shown ability to resolve vortex wakes and their interaction with blades and airframes, largely due to the use of the high-order background Cartesian solver SAMCart.[18] All near-body solvers commonly used with Helios, however, are second-order accurate and require fine meshes to resolve complex fluid structures. The development of high-order accurate algorithms has been a core area of CFD research in the last decade. To achieve greater than second-order accuracy on curved, unstructured meshes many research groups have used Discontinuous Galerkin (DG) methods[19,20] in an overset framework. Flux-reconstruction methods have also gained popularity in both sliding-interface[21] and overset[22,23] applications.

Accuracy limitations of the near-body region in Helios are supplemented by challenges in achieving high-order accurate interpolation between meshes. Overset interpolation, especially between moving meshes,

is a likely cause of vortex destabilization in the flow solution of rotor wakes.[24] High-order interpolation is significantly more complex than traditional second order interpolation since it requires additional connectivity data between computational cells. To avoid passing additional large, complex datasets between the flow solver and connectivity routines, a popular approach for high-order interpolation is with the use of callback functions.[20] In this work, PUNDIT is augmented with flow-solver-provided call-back function to perform high-order interpolation between near and off-body grids.

Interpolation between meshes also poses temporal accuracy and convergence challenges for overset systems. In the current framework, the PUNDIT connectivity module within Helios performs exchanges between grids every timestep. Temporal accuracy could be improved by integrating communication between meshes at the non-linear iteration level however this can result in degraded convergence trends. Recent work by Jude et al[25,26] has demonstrated good convergence in an overset system coupled at the linear-solver level in a Python framework similar to Helios. The Overset Generalized Minimum Residual Method (O-GMRES) from Jude et al. has been integrated into the Helios framework for use with SAMCart and mStrand.

Lastly, this work also addresses a new functionality in Helios using a reduced-order aerodynamic model (ROAM) in place of the high fidelity model in the interest of reducing simulation time. The ROAM Helios module represents rotor blades as actuator lines and solid bodies as immersed boundaries. The influence of the blades or solid bodies are accounted for in the background Cartesian grids as momentum sources and internal boundary conditions. One of the goals of using ROAM is to accelerate simulations for generating surrogate or machine-learning models which may involve running hundreds or thousands of cases.

II. Technical Approach

II.A. Governing Equations

SAMCart, a finite-difference solver, and mStrand, a finite-volume solver, share the same governing equations based on the strong conservation form of the Navier–Stokes equations:[27]

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x} + \frac{\partial \mathbf{G}_i}{\partial y} + \frac{\partial \mathbf{H}_i}{\partial z} = \frac{\partial \mathbf{F}_v}{\partial x} + \frac{\partial \mathbf{G}_v}{\partial y} + \frac{\partial \mathbf{H}_v}{\partial z} + \mathbf{S} \tag{1}$$

where $\mathbf{Q}$ is the vector of conserved variables $\{\rho, \rho u, \rho v, \rho w, e\}^T$ with ρ, u, v, w and e denoting the local flow density, three components of velocity and total energy, respectively. $\mathbf{F}, \mathbf{G}$, and $\mathbf{H}$ are the fluxes in the $x-$, $y-$, and $z-$directions respectively for both inviscid and viscous quantities. The Reynolds-Averaged Navier–Stokes equations are closing using the Spalart-Allmaras[28] turbulence model, which adds a sixth variable, $\tilde{\nu}_t$, to the vector $\mathbf{Q}$. The spatial residual, $\mathbf{R}$, is used for convenience to lump together all spatial operations such that

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{R}(\mathbf{Q}) = 0 \tag{2}$$

A second order backwards Euler implicit time marching scheme is imposed to discretize the system in time.

$$\frac{3\mathbf{Q}^{n+1} - 4\mathbf{Q}^n + \mathbf{Q}^{n-1}}{2\Delta t} + \mathbf{R}^{n+1}(\mathbf{Q}^{n+1}) = 0 \tag{3}$$

The term $\mathbf{R}^{n+1}$ is non-linear and can be linearized using the first few terms of a Taylor series. A dual-time marching term, denoted with index p, is also required to remove linearization error. The dual-time marching term corresponds to a subiteration, or non-linear iteration, and uses symbol τ. The non-linear iterations are used to march the solution in pseudo-time to a steady state, at which point the original equations are recovered. The solution at subiteration index $p+1$ serves as an approximation to the solution at physical temporal index $n+1$. This dual-timestepping procedure is sometimes referred to as "pseudo transient continuation." The resulting discretization is

$$\frac{\mathbf{Q}^{p+1} - \mathbf{Q}^p}{\Delta\tau} + \frac{3\mathbf{Q}^{p+1} - 4\mathbf{Q}^n + \mathbf{Q}^{n-1}}{2\Delta t} + \frac{\partial \mathbf{R}^p}{\partial \mathbf{Q}}\Delta\mathbf{Q} = -\mathbf{R}^p(\mathbf{Q}^n) \tag{4}$$

where

$$\Delta\mathbf{Q} = \mathbf{Q}^{p+1} - \mathbf{Q}^p \tag{5}$$

which can be further modified into the linear system

$$\left[\frac{\mathbf{I}}{\Delta\tau}+\frac{\mathbf{I}}{2/3\Delta t}+\frac{\partial\mathbf{R}^p}{\partial\mathbf{Q}}\right]\Delta\mathbf{Q}=-\mathbf{R}^p-\frac{3\mathbf{Q}^p-4\mathbf{Q}^n+\mathbf{Q}^{n-1}}{2\Delta t}=\mathbf{R}_t. \tag{6}$$

The entire right-hand-side (RHS), including the temporal terms, is referred to as the residual for time-accurate computation. The L2-norm of the residual is a measure of subiteration convergence, since as the solution at p approaches $p+1=n+1$, Eq. (3) is recovered. The inversion of the LHS matrix is rarely performed exactly, since this process is typically very computationally expensive. Instead there are three general procedures are used in SAMCart and mStrand to approximate the solution $\Delta\mathbf{Q}$:

1. SAMCart: Lower-Upper Symmetric Gauss-Seidel (LU-SGS) preconditioner (mean-flow) and Alternaning Direction Implicit (turbulence model).
2. mStrand: Multi-colored Gauss-Seidel Strand-line preconditioner.
3. mStrand: Incomplete Lower-Upper factorization.

Only the Gauss-Seidel method is used as the preconditioner in mStrand and the Jacobians in the approximation of $\partial\mathbf{R}/\partial\mathbf{Q}$ are fully coupled between the mean-flow and turbulence variables. In SAMCart, the Jacobians are decoupled between the mean-flow and turbulence equations. A detailed formulation of the residual is not described here because it varies between SAMCart and mStrand solvers.

In this work a Discontinuous Galerkin code DG3D is also integrated within the Helios framework. To derive the weak form of the Navier–Stokes equations, Eq. (1) is first multiplied by a test function ϕ and integrated over the domain Ω. For clarity, Einstein notation is used where the subscripts of i and j represent spatial dimensions and have a range of 1 to 3 and the index k varies over the number of variables.

$$\int_\Omega \phi_m\left(\frac{\partial Q_k}{\partial t}+\frac{\partial F_{ki}}{\partial x_i}\right)\mathrm{d}\Omega=0.$$

Next, integration by parts is performed and the residual R_{km} is defined as:

$$R_{km}=\int_\Omega\left(\phi_m\frac{\partial Q_k}{\partial t}-\frac{\partial\phi_m}{\partial x_i}F_{ki}\right)\mathrm{d}\Omega+\int_\Gamma\phi_m F^*_{ki}\mathrm{n}_i\mathrm{d}\Gamma=0$$

where ϕ are the basis functions and the solution is approximated as $Q_k=\phi_m a_{km}$, where the repeated index m is summed over the number of basis functions. The basis is formed by using piecewise discontinuous shape functions. Each element's shape function is represented as continuous polynomials within the cell and zero outside the cell. The basis in this work is hierarchical and is formed by combining a $p=1$ Lagrangian basis, using the vertex of each element, and a Jacobi polynomial as a kernel for creating higher degree terms. An example of a shape function for a segment is:

$$\phi_m(\xi)=\begin{cases}1 & m=0\\ \frac{1-\xi}{2},\frac{1+\xi}{2} & m=1\\ \left(\frac{1-\xi}{2}\right)\left(\frac{1+\xi}{2}\right)P^{\alpha,\beta}_{m-2}(\xi) & m>1\end{cases}$$

where α and β are parameters that give Jacobi polynomials their orthogonality properties with respect to the weight $(1-\xi)^\alpha(1+\xi)^\beta$. Similar to the segment a basis is created for tetrahedral, pyramidical, prismatic, and hexahedral element types. Also, the basis is used to represent both the solution and geometrical mapping. The polynomial degree can be chosen independently for each, for example the solution can be represented by a $p=1$ basis and the geometrical mapping can be represented by a $p=2$ basis.

The residual consists of volume and face integrals which are approximated using numerical quadrature. For hexahedral elements and quadrilateral elements tensor products of Gaussian quadrature points are used. For the other element types a set of efficient numerical quadrature rules are used.[29] The quadrature rules are chosen to exactly integrate polynomials within the volume integrals to $2p$ accuracy and the face integrals to $2p+1$ accuracy. The integrals over faces Γ require special treatment for the fluxes in these terms. The advective fluxes are calculated using a Riemann solver. Implemented Riemann solvers include: Lax-Friedrichs,[30] Roe,[31] and artificially upstream flux vector splitting scheme (AUFS).[32] In this work, only the Lax-Friedrichs flux is used. The diffusive fluxes are handled using a symmetric interior penalty (SIP) method.[33–35]

II.B. Linear Solver and CFL Controller

II.B.1. Overset GMRES

The Generalized Minimal Residual (GMRES) algorithm is one of many different methods that can be used to solve a linear system of equations $Ax = b$. Compared to more traditional, iterative methods like Gauss–Seidel or approximate factorization, GMRES is a minimization algorithm on the quantity $||b - Ax||$. This minimization is accomplished with the use of m Krylov-subspace vectors V_m, where each vector is the size of the solution vector x. A more detailed overview of GMRES is presented in detail in the work of Yousef Saad.[36] For brevity this section will focus on applications of GMRES to CFD.

GMRES with m vectors applied to the Navier–Stokes equations to approximate the change in the solution $\mathbf{\Delta Q}$ will require $m \times sizeof(\mathbf{\Delta Q})$ in memory. High memory usage is one of the disadvantages of GMRES compared to Gauss–Seidel or approximate factorization methods. For a framework-level implementation, however, the ultimate goal would be to use only a few Krylov vectors (~ 5) and rely upon each flow solver to provide a strong preconditioner.

The GMRES algorithm with right-preconditioning is shown in Alg. 1. Preconditioning is the process of altering the system of equations to a different system that has the same solution but is less stiff and hence easier to solve. Generally right preconditioning is preferred over left because it preserves the magnitude of the residual within linear iterations. In Alg. 1, semantically r is the result of preconditioning the vector v_j. Mathematically the preconditioner is often expressed as a matrix $\mathbf{M}$, which approximates the matrix $\mathbf{A}$ but is easily invertible such that for the system $\mathbf{M}r = v_j$, $r = \mathbf{M}^{-1}v_j$.

Algorithm 1 Right-Preconditioned O-GMRES Algorithm.

1: Compute $r_o = b - Ax_o$, $\beta := ||r_o||_2$, and $v_1 := r_o/\beta$
2: **for** $j = 1, 2, ..., m$ **do**
3: Compute $r_j :=$ PRECONDITION(v_j) ▷ Preconditioner
4: OVERSET_INTERPOLATE(r_j)
5: Compute $w_j := Ar_j$ ▷ Matrix-Vector Product
6: **for** $i = 1, ..., j$ **do**
7: $h_{ij} := (w_j, v_i)$ ▷ Vector Dot Product
8: OVERSET_SUM(h_{ij})
9: $w_j := w_j - h_{ij}v_i$
10: **end for**
11: $h_{j+1,j} = ||w_j||_2$
12: **if** $h_{j+1,j} == 0$ **then**
13: $m := j$
14: **goto** 18
15: **end if**
16: $v_{j+1} = w_j/h_{j+1,j}$
17: **end for**
18: Define the $(m + 1) \times (m)$ Hessenberg matrix $\tilde{H} = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$
19: Compute y_m, the minimum of $||\beta e_1 - \tilde{H}_m y||_2$ and $r_m = V_m y_m$
20: Compute $x_m = x_o +$ PRECONDITION(r_m)
21: OVERSET_INTERPOLATE(x_m)

From the authors' previous work,[37] the highlighted lines in Alg. 1 are required to ensure the GMRES algorithm applies to the full overset system. Without the exchange of data between meshes, the GMRES algorithm is decoupled between solvers. Convergence within linear (Krylov) iterations of a de-coupled system will not be representative of the full system.

The overset interpolation from line 4 in Alg. 1 can be thought of as an extension of the precondition operation from the previous line. In locations where a node receives an interpolated value from another mesh (ie. where the *iblank* value is -1), performing the interpolation at that point is equivalent to a Jacobi preconditioning operation.

The Overset GMRES (O-GMRES) algorithm is written in Python as a partitioned algorithm that does not build the full global solution vector on any processor. Nor does O-GMRES require additional memory over the

traditional GMRES algorithm. The only requirement is that each solver expose routines for preconditioning and matrix-vector product. Python handles pointers to large buffers of memory but all numerical operations are handled with function calls to fast, compiled languages. Pseudo-code with the traditional de-coupled GMRES algorithm is shown in Alg. 2. The improved implementation is shown in Alg. 3, where now the GMRES algorithm is written as part of the Helios framework and the connectivity information is built in to the Krylov iterations.

Algorithm 2 Baseline Timestepping Algorithm in Helios

```
for iter = 1, 2, 3, ... do
    for s ∈ {SAMCart, mStrand, ...} do
        DO_OVERSET_EXCHANGE(Q)
        for subiter = 1, 2, 3, ... do
            s.LINEAR_SOLVER(iter, subiter)        ▷ Linear iterations at the flow-solver-level
        end for
    end for
end for
```

Algorithm 3 Overset GMRES Timestepping Algorithm in Helios

```
for iter = 1, 2, 3, ... do
    for subiter = 1, 2, 3, ... do
        ...                                       ▷ Start of GMRES Algorithm
        for liniter = 1, 2, 3, ... do
            ...                                   ▷ GMRES Operations (precondition, mvp, etc.)
            DO_OVERSET_EXCHANGE(r_j)
            ...
        end for
    end for
end for
```

In all codes where GMRES is used, the matrix-vector product is computed using the approximation

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]\mathbf{X} \approx \frac{\mathbf{R}(\mathbf{Q}+\epsilon\mathbf{X})-\mathbf{R}(\mathbf{Q})}{\epsilon} \tag{7}$$

where ϵ is a small scalar chosen based on the magnitude of $\mathbf{Q}$. Eq. (7) uses the property of Fréchet derivatives and has been shown to work well for CFD applications.[38,39] Using the finite-difference approximation introduces error in the calculation of the matrix-vector product and may prevent the GMRES method from achieving formal quadratic convergence.

II.B.2. Convergence Controller

The same routines exposed at the Python level for O-GMRES also allow for a simple controller to improve convergence. Both SAMCart and mStrand have been modified to accept a scalar multiplication factor of the CFL/pseudo-timestep. The approach for increasing or decreasing the CFL is based on the line-search method from the work of Ceze and Fidkowski.[40] Once the GMRES algorithm produces a candidate solution $\Delta\mathbf{Q}$, the line-search method finds the scalar parameter α which reduces the transient residual $\mathbf{R}_t$:

$$\mathbf{R}_t(\mathbf{Q}+\alpha\Delta\mathbf{Q}) \tag{8}$$

Based on the value of α, the CFL multiplier is then increased, decreased, or kept the same. Pseudo code of the complete line-search algorithm is shown in Alg. 4. Constant β_1 is used to reduce the CFL and β_2 is used to increase the CFL. Constant κ_{LS} specifies a maximum amount by which the residual is allowed to rise between iterations and SRCH_MAX sets a lower bound for α. The constants used in this work are summarized in Table 1.

Algorithm 4 Line-search convergence controller

$CFL_{factor} := min(max(CFL_{factor}, \text{CF\_MIN}), \text{CF\_MAX})$
$\Delta\mathbf{Q} := \text{OGMRES}(\mathbf{Q})$
$\alpha := 1, \quad i_{try} := 0$
$\tilde{\mathbf{Q}} := \mathbf{Q} + \alpha \cdot \Delta\mathbf{Q}$
while $\|\mathbf{R}_t(\tilde{\mathbf{Q}})\| > \kappa_{LS}\|\mathbf{R}_t(\mathbf{Q})\|$ and $i_{try} <$ SRCH_MAX **do**
    $\alpha := \alpha \cdot \alpha_{factor}$
    $\tilde{\mathbf{Q}} := \mathbf{Q} + \alpha \cdot \Delta\mathbf{Q}$
    $i_{try} := i_{try} + 1$
end while
if $\alpha >= 0.7$ **then**
    $CFL_{factor} := \beta_2 \cdot CFL_{factor}$
    for $\mathbf{s} \in$ {SAMCart, mStrand, ...} **do**
        $\mathbf{s}$.SET_CFL_FACTOR(CFL_{factor})
    end for
else if $i_{search} >=$ SRCH_MAX **then**
    $CFL_{factor} := \beta_1 \cdot CFL_{factor}$
    go to 1
end if

Table 1. Constants used in line-search algorithm

	Steady Cases	**Time-accurate Cases**
κ_{LS}	1.01	1.01
β_1	0.1	0.1
β_2	1.2	$\sqrt{10}$
α_{factor}	0.7	0.7
SRCH_MAX	5	5
CF_MIN	1	1
CF_MAX	10^2	10^4

II.C. High-order domain connectivity and interpolation

High-order methods that use finite element type basis are becoming increasingly common and popular for development of accurate flow solvers. There are several high-order discretization approaches possible, such as flux reconstruction[41] or Galerkin type formulations such as Stream-wise Upwind Petrov-Galerkin(SUPG)[42,43] or Discontinuous Galerkin[44,45] formulations. The common theme of all these approaches are high-order elements that are often bounded by curved faces with internal degrees freedoms distributed inside of them to provide the high-order support. The basis functions used to achieve high-order accuracy are often disparate and can range from simple Lagrangian basis with Gauss-Legendre of Gauss-Lobotto quadrature points or more complex hierarchical sets. Development of an overset grid interpolation framework that maintains all of the different types of basis used by different kinds of solvers is not scalable and modular. Instead a better approach is to inherit the the basis functions directly from participating flow solvers as call back functions. In general, the donor receptor relationship will be as shown in Figure 1 where multiple degrees of freedom per receptor element is shown and the degrees of freedom within the same element may multiple donors in another grid. Furthermore, the off-body solver SAMCart[8] also uses a formally 5th order accurate finite difference discretization scheme. It is imperative that the domain connectivity approach consistently couples and sets interpolation patterns with disparate high order discretization schemes, i.e. higher order finite elements and high-order finite difference.

II.D. Unstructured near-body DG solver: DG3D

The high-order near-body solver DG3D is a parallel 3D Discontinuous Galerkin (DG) finite element method. The solver can handle hybrid mixed element meshes (tetrahedra, pyramids, prisms, and hexahedra), curved

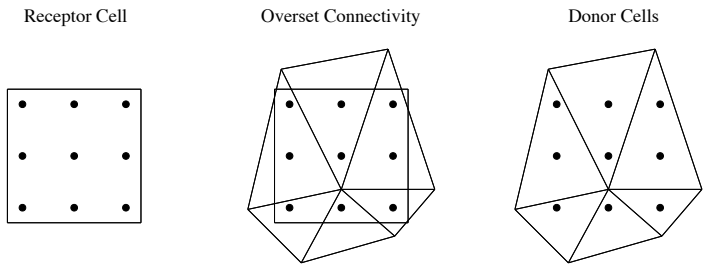


Figure 1. Receptor and donor cell overset connectivity.

elements, and incorporates both p-enrichment and h-refinement capabilities using non-conforming elements (hanging nodes). It is used to discretize the compressible Navier-Stokes equations as well a PDE-based artificial viscosity equation[46–48] and the Spalart-Allmaras turbulence model (negative-SA variant).[49] The advective fluxes are calculated using a Riemann solver. Implemented Riemann solvers include: Lax-Friedrichs,[30] Roe,[31] and artificially upstream flux vector splitting scheme (AUFS).[32] The diffusive fluxes are handled using a symmetric interior penalty (SIP) method.[33,34] The time derivative can be approximated using the explicit scheme RK4 or an implicit BDF2 scheme. The implicit solver uses a Newton-Raphson method to solve the non-linear set of equations. These equations are linearized to obtain the full Jacobian. The linear system is solved using a flexible-GMRES[50] (fGMRES) method. To further improve convergence of fGMRES a right preconditioner can be applied to the system of equations. Preconditioners that have been implemented include Jacobi relaxation, Gauss-Seidel relaxation, line implicit Jacobi, and ILU(0). This solver has been used to successfully to solve hypersonic flows,[51] turbulent flow over wings[52] and aircraft,[53,54] Direct Numerical Simulations (DNS) of Taylor Green Vortex, and overset simulations.[55]

II.E. High order Overset Mesh Assembly: PUNDIT

There are four callback functions that are needed interface the domain connectivity module PUNDIT (Domain connectivity module in Helios) with high-order finite element codes. While these call back functions are currently inherited from the DG3D solver, they do maintain a generic and modular structure promoting easy extension to other finite element codes.

The first callback function is a function which generates a list of receptor points. Typically these are the nodes or cell centers in a mesh, but for a high-order discretization additional points are needed within each cell. The second function is a high-order donor inclusion test which is necessary for curved elements. Once donor cells are located the third function returns the weights for the high-order interpolation on the donor cells. The last callback function converts the interpolated solution at the receptor points back into solution coefficients or modal coefficients on the receptor cell.

There are two approaches for transferring solution data: the first uses a mass matrix approach and the second a Vandermonde matrix approach. One of these two methods is necessary because it is the transfer of solution coefficients rather than the actual solution values that are needed in a modal finite-element method. Both the mass matrix and the Vandermonde matrix methods require solution values from the donor cells at specific points on the receptor cell. Figure 2 shows the locations of the receptor nodes for each method. For the mass matrix method the interpolation points correspond to the quadrature points used in the volume integral evaluation of the residual, while for the Vandermonde matrix method the interpolation points consist of equidistant points. The interpolation points become the receptor points in the first callback function and then the receptor nodes are used by PUNDIT to determine the donor cells.

To find the donor cells a callback function for a high-order curved cell inclusion test is used. PUNDIT uses its native point-location algorithm to search for donor cells and the callback function is used to determine if the receptor point is located inside a donor cell. For straight sided elements PUNDIT can handle this process using straight forward geometric inclusion tests. However for a curved element, a specialized callback function is needed. To test if a point is inside an element, the physical coordinates (x, y, z) are converted to natural coordinates (r, s, t) in the mapped space of the standard isoparametric element, as shown in Figure 3.

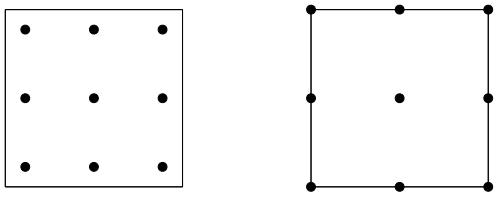


Figure 2. Locations of receptor nodes on a quadrilateral element for mass matrix method (left) and Vandermonde matrix method (right).

The natural coordinates are found using the following equations:

$$\sum_{m=1}^{n_{map}} \psi_m(r,s,t) b_{1m} = x, \quad \sum_{m=1}^{n_{map}} \psi_m(r,s,t) b_{2m} = y, \quad \sum_{m=1}^{n_{map}} \psi_m(r,s,t) b_{3m} = z,$$

where b are the physical coordinate mapping coefficients, ψ are the mapping basis functions, and n_{map} are the number of mapping coefficients for the element. This gives three non-linear equations with three unknowns. To solve this problem a Newton-Raphson method is used which can be written as:

$$\begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial r} & \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t} \end{bmatrix} \begin{bmatrix} \delta r \\ \delta s \\ \delta t \end{bmatrix} = - \begin{bmatrix} \sum_{m=1}^{n_{map}} \psi_m(r,s,t) b_{1m} - x \\ \sum_{m=1}^{n_{map}} \psi_m(r,s,t) b_{2m} - y \\ \sum_{m=1}^{n_{map}} \psi_m(r,s,t) b_{3m} - z \end{bmatrix}$$

Once the natural coordinates are found it is trivial to test if the point lies inside the element. For example, a tetrahedra has four faces that are each associated with a plane that can be tested for inclusion. If any of the following inequalities in Table 2 are true then the point does not lie inside the element. In this table, the parameter ϵ is a small number, if this number approaches zero then strict inclusion is required. However, if this number is larger, the inclusion test will be less restrictive and extrapolation (instead of interpolation) is allowed.

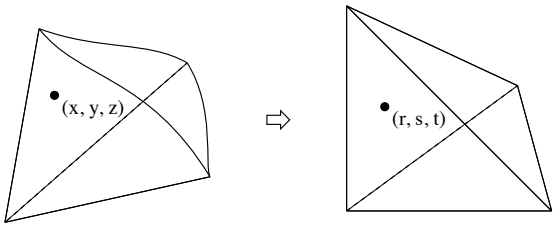


Figure 3. Curved tetrahedron in physical coordinates (left) and natural coordinates (right)

Once a donor cell is found the third callback function creates the interpolation weights. Evaluating the solution basis functions at the natural coordinates found from the inclusion test creates a high-order interpolation for the donor cell. The solution q is passed back to the receptor cell by:

$$q(r,s,t) = \sum_{m=1}^{n_{mode}} \phi_m^d(r,s,t)\, a_m$$

tetrahedron	pyramid	prism	hexahedron
$r+1<-\epsilon$	$r+1<-\epsilon$	$r+1<-\epsilon$	$\lvert r\rvert-1>\epsilon$
$s+1<-\epsilon$	$s+1<-\epsilon$	$\lvert s\rvert-1>\epsilon$	$\lvert s\rvert-1>\epsilon$
$t+1<-\epsilon$	$t+1<-\epsilon$	$t+1<-\epsilon$	$\lvert t\rvert-1>\epsilon$
$r+s+t+1>\epsilon$	$r+t>\epsilon$	$r+t<-\epsilon$	
	$s+t>\epsilon$		

Table 2. Inequalities for each element type for inclusion.

where $\phi_m^d(r,s,t)$ are the donor cell basis functions (where superscript d represents donor) evaluated at the receptor node locations, n_{mode} are the number of solution basis functions, and a_m are the solution coefficients on the donor cell. PUNDIT transfers, in parallel, all of the interpolated solutions from the donor cells to the receptor cells. The fourth callback function converts these solution values back into coefficients on the receptor cells. As discussed previously two approaches have been implemented; the first is the mass matrix method. This approach starts off with a Galerkin projection:

$$\sum_{n=1}^{n_{mode}} \phi_n^r(\xi_{1k},\xi_{2k},\xi_{3k})a_n = q(\xi_{1k},\xi_{2k},\xi_{3k})$$

where ϕ_n^r are the solution receptor basis functions (where superscript r represents receptor), n_{mode} is the number of solution receptor basis functions, ξ_{ik} are the quadrature points (where the indices $i=1,2,3$ and k runs from one to the number of quadrature points n_{qp}), q are the solution values, and a_n are the receptor solution coefficients to be solved for. To solve for the coefficients both sides are multiplied by the receptor cells basis function and integrated over the cell:

$$\sum_{n=1}^{n_{mode}} \int_\Omega \phi_m^r \phi_n^r a_n \,\mathrm{d}\Omega = \int_\Omega \phi_m^r q \mathrm{d}\Omega.$$

This creates a mass matrix defined as:

$$M_{mn} = \int_\Omega \phi_m^r \phi_n^r \mathrm{d}\Omega$$

and the right hand side becomes:

$$f_m = \int_\Omega \phi_m^r q \mathrm{d}\Omega = \sum_{k=1}^{n_{qp}} \phi_m^r(\xi_{1k},\xi_{2k},\xi_{3k}) q(\xi_{1k},\xi_{2k},\xi_{3k}) w_k$$

where w_k are the quadrature weights. To solve for the coefficients the right hand side is integrated and the mass matrix is inverted to give:

$$a = \mathbf{M}^{-1} f.$$

The mass matrix is LU factorized ahead of time and only a forward/backward solve is needed to solve for a.

The second approach is the Vandermonde method. Again, this starts with a projection operator:

$$\sum_{n=1}^{n_{mode}} \phi_n(\zeta_{1m},\zeta_{2m},\zeta_{3m})a_n = q(\zeta_{1m},\zeta_{2m},\zeta_{3m})$$

except now the Vandermonde matrix:

$$V_{mn} = \phi_n(\zeta_{1m},\zeta_{2m},\zeta_{3m})$$

is constructed by evaluating the basis at the points ζ_{im}, where the number of points are equal to the number of basis functions, making the Vandermonde matrix square. The solution coefficients are solved for by inverting the Vandermonde matrix:

$$a = \mathbf{V}^{-1} q$$

for the receptor solution coefficients. For efficiency the Vandermonde matrix is constructed for every element type and polynomial degree and factorized ahead of time so that only forward/backward solves are needed.

There are two downsides to the mass matrix method. The first is that even if the mass matrix is diagonal the right hand side still requires integration with respect to the basis. The second is that, for tetrahedra, pyramids, and prisms, the number of quadrature points is greater than the number of basis modes. Also, for curved elements more quadrature points are needed making the mass matrix approach even more costly. Due to this the mass matrix method requires more computational work than the Vandermonde matrix method. However, a benefit of the mass matrix method is that it gives a more optimal approximation to the interpolated values compared to the Vandermonde approach at equidistant points. The Vandermonde matrix at equidistant points can suffer from interpolation errors at high polynomial degrees.

The interpolation process for the off-body solver (SAMCart) data is as follows: Since the mesh is uniform and isotropic, a tensor product basis composed of Lagrange polynomials is constructed to form a tricubic interpolant that is 4th order accurate. Every receptor node is interpolated using the 64 closest Cartesian points. Locally, these 64 Cartesian points form the nodes of $4 \times 4 \times 4$ reference element that extents from -2 to 1 in each dimension. The Lagrange basis functions within this reference elements are:

$$\phi_1(x) = \frac{-x(x-1)(x-2)}{6} \qquad x \in [-2, 1] \tag{9}$$

$$\phi_2(x) = \frac{(x-1)(x-2)(x+1)}{2} \qquad x \in [-2, 1] \tag{10}$$

$$\phi_3(x) = \frac{-x(x+1)(x-2)}{2} \qquad x \in [-2, 1] \tag{11}$$

$$\phi_4(x) = \frac{(x-1)x(x+1)}{6} \qquad x \in [-2, 1] \tag{12}$$

$$\tag{13}$$

The interpolation to receptor nodes can then be expressed as:

$$q_{interp} = \sum_{i=1}^{4} \sum_{j=1}^{4} \sum_{k=1}^{4} q_{ijk} \phi_i(r) \phi_j(s) \phi_k(t) \tag{14}$$

Where (r, s, t) are $\in [0, 1]$ and found relative to the Cartesian cell that contains a given point. Note that the weights $w_{ijk} = \phi_i(r)\phi_j(s)\phi_k(t)$ depend only on geometry and can be applied to any field. Furthermore, the number of operations in computing the weights can be reduced by expressing the above equation as a tensor product of form $w = (\phi(r) \otimes \phi(s)) \otimes \phi(t)$.

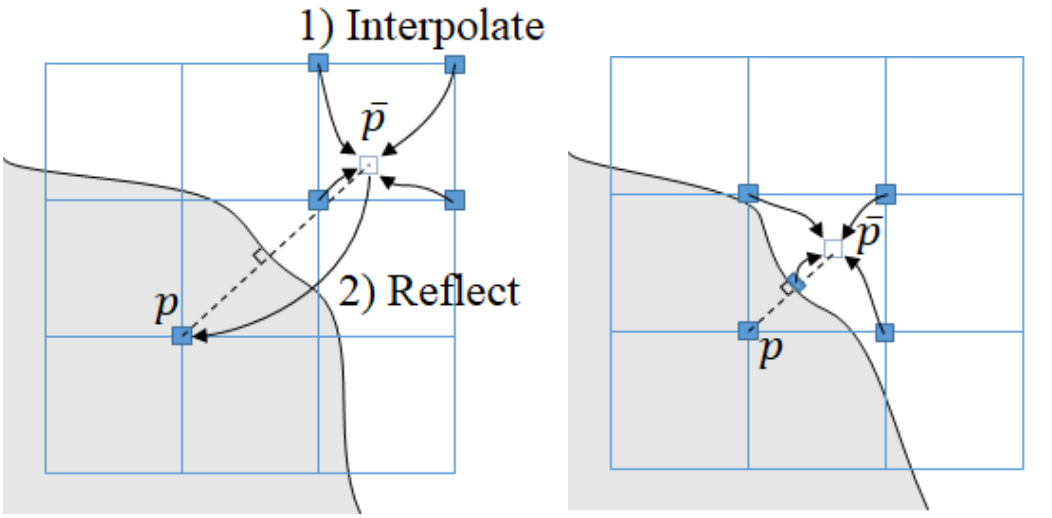
II.F. Immersed Boundary and Actuator Line Methods

This section describes the Reduced-Order Aerodynamic Model (ROAM) module, an addition to the Helios framework that allows bluff bodies represented as immersed boundaries and rotor blade represented as actuator line.

The immersed boundary method (IBM) used in this work is a simple procedure very similar to the work of Ghias et. al[56] from 2007. IBM is used to prescribe a no-penetration boundary on the surface of a stationary body. The boundary condition is applied to points in the Cartesian mesh and appropriate iblanking is assigned to indicate to the Cartesian flow solver which points are within a solid boundary. For simplicity, the body surface nodes are broadcasted to all MPI tasks and each processor is responsible for blanking the Cartesian blocks resident on that processor. Each Cartesian block performs an efficient ray-tracing algorithm on a projection of the body surface to determine whether a vertex lies inside or outside of the body.

Once iblanks are applied, Cartesian vertices within a 3-fringe distance from the wall are assigned "image" locations from which to reflect velocities and hence prescribe the boundary condition. Figure 4 shows the two simplest cases for interpolation and reflection. A given node p inside the boundary finds image point $\bar{p}$ across the boundary by extending the vector (dotted line in Fig. 4) to the closest point on the surface. If the closest point on the surface is on a face, the vector will be perpendicular to that face. If the closest point on the surface is a node, then the vector represents an average normal of all faces sharing that vertex. Figure 4(a) illustrates the case where $\bar{p}$ can be simply interpolated from known field points (ie. where the iblank value is 1). Figure 4(b) illustrates the case were the image point $\bar{p}$ does not have a full cell from which to

interpolate because one or more corners is blanked. To avoid using the blanked value, a projected point on the surface is approximated using the average pressures and densities of nearby field points and the normal velocity component set to zero. Using the approximated surface point ensures a convex interpolation stencil can be used to find $\bar{p}$. Once the values at $\bar{p}$ are found, the pressure and density are copied to p and the velocities are reflected.



(a) Basic Interpolation/Reflection

(b) Interpolation/Reflection with 1 projected point.

Figure 4. Assignment of immersed boundary point using interpolated, reflected values from interior points.

Some geometries may result in the image point $\bar{p}$ also residing within the body, as illustrated in Fig. 5. This scenario occurs when the solid body has sharp, concave corners. In this case our implementation simply interpolates from the surrounding blanked points in the body. This is referred to as implicit interpolation, since the points in the interpolation stencil are themselves interpolated points.

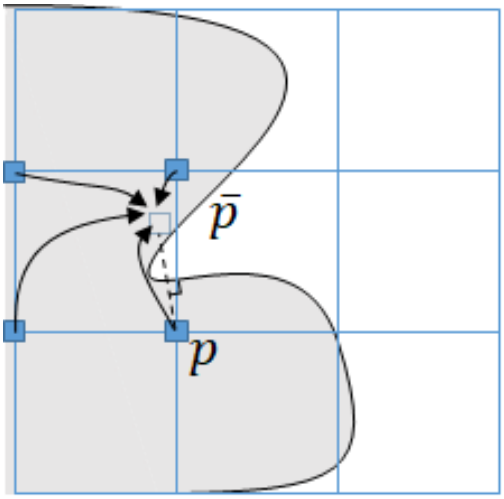


Figure 5. Reflected immersed boundary point may also be within the body, in which case all interpolation points are also blanked points.

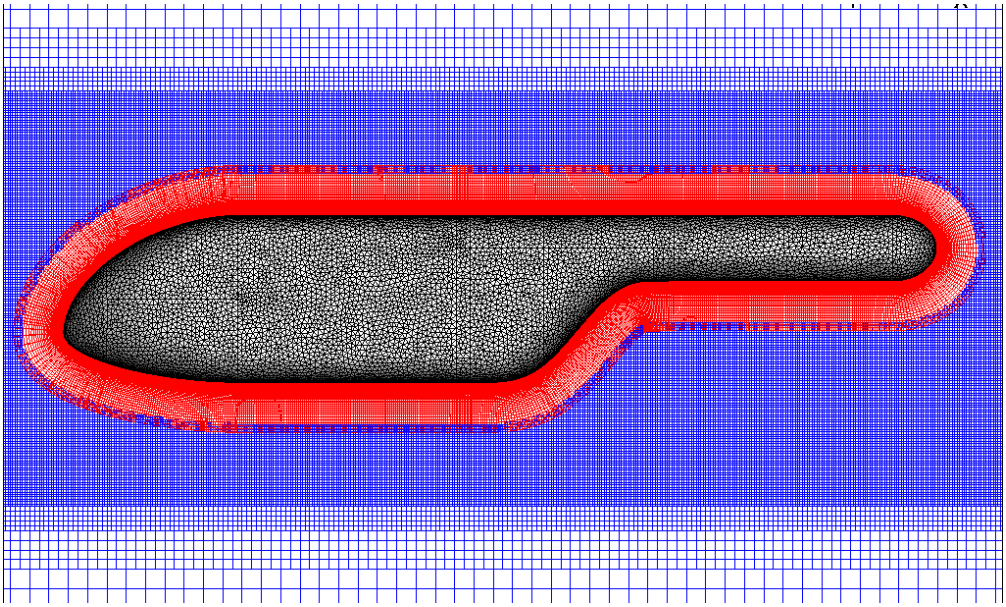
The immersed boundary condition imposes values on ghost points within the SAMCart mesh as a function of field points. When an operation like a matrix-vector-product is performed using Fréchet derivatives, the $\mathbf{Q} + \epsilon\mathbf{X}$ operation (from Eq. (7)) in SAMCart needs to properly apply an immersed boundary response to approximate residuals. The immersed boundary routine is therefore passed from ROAM to SAMCart as a callback function.

The actuator line method is used to model blades in the reduced order aerodynamic model. Lifting lines are discretized into a set of control points and the flow induced velocities at these control points are interpolated from the mesh system modeling the flow field. To exclude the effects of bound circulation, a circle of sensors is created around each control point and the velocity at the control point is found as an average of all sensors. Once the velocities at control points are obtained, linear aerodynamic theory augmented with airfoil tables are used to determine the force coefficients and compute the force distribution. The force distribution is then embedded into the flow field equations as source terms. The transfer of velocities using interpolation and resultant forces using source terms is the basis of the coupling structure between the actuator line model and the Navier–Stokes equations. Keeping with the theme of developing fully coupled linear solver in a modular framework the actuator line method is implemented as a separate module and the O-GMRES algorithm proceeds in a very similar way with actuator line sources computed once per timestep.

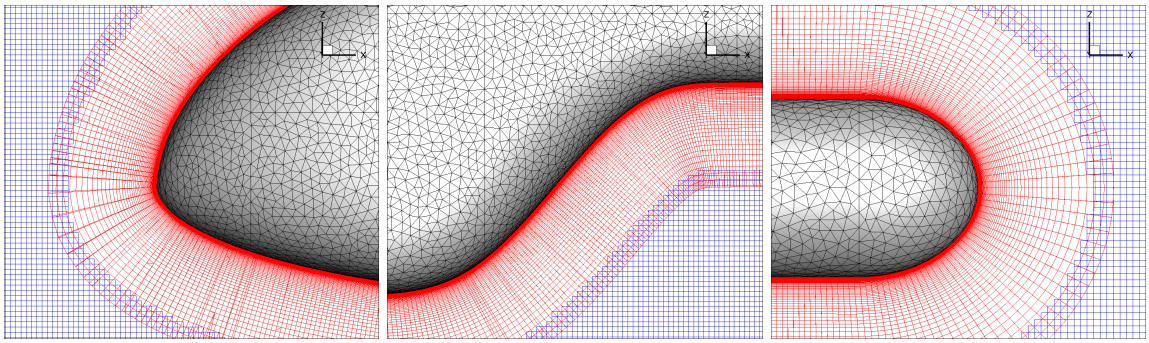
III. Results

III.A. High-order solutions for the ROBIN fuselage

The ROtor-Body INterference fuselage is a well known test case that has been utilized to study smooth separation problems typical to rotorcraft fuselages at low speed conditions. The main feature is the separation of the flow under the tail-boom of the fuselage. A high-order curved prismatic mesh generation scheme was recently developed[57] and utilized to create high-order (p=2) strand type mesh for the fuselage that used 18 node prisms with quadratic basis. The solution process utilized DG3D[44] as the near-body and SAMCart as the off-body solver. The final mesh system has 14.4 million degrees of freedom in the near-body and 57 million degrees of freedom in the off-body. Unsteady RANS simulations are performed in time-accurate mode (BDF2 in time), with a non-dimensional time step of $dt = 0.05$ and a Mach number of 0.1. Time stepping at this rate will make the acoustic waves travel approximately 10 body lengths in 5000 steps. The near-body and off-body solvers are coupled at the physical time step level and are able to converge their respective non-linear residuals 3-4 orders of magnitude every time step. The Overset-GMRES method has not yet been implemented for coupling between DG3D and SAMCart and is one of the items for future work. The connectivity was performed using PUNDIT and the methodology described in previous section was utilized to create the masking patterns (iblanking) and perform solution interpolation. The interpolations from nearbody-to-offbody and offbody-to-nearbody are 3rd order and 4th order accurate respectively. The high-order mesh and the overset strand/Cartesian mesh system are shown in Figure 6. Figure 7 shows the contours of velocity magnitude and surface pressure coefficients obtained using the high-order overset simulations. Figure 8 compares the predicted pressure coefficients with experimental data and peer simulations that used linear meshes and computed using the mStrand solver for the near-body. All simulations show good agreement with each other and fair agreement with the experimental data with considerable discrepancies at and after separation in case of the coarse/medium linear meshes. Both the fine linear mesh using mStrand and medium quadratic mesh (p=2) mesh using DG3D show good agreement in the prediction of the location of separation. Further inspection of details near the separation zone (Figure 8(c)) shows that the quadratic mesh improves the pressure coefficient prediction post separation compared to the fine linear mesh.

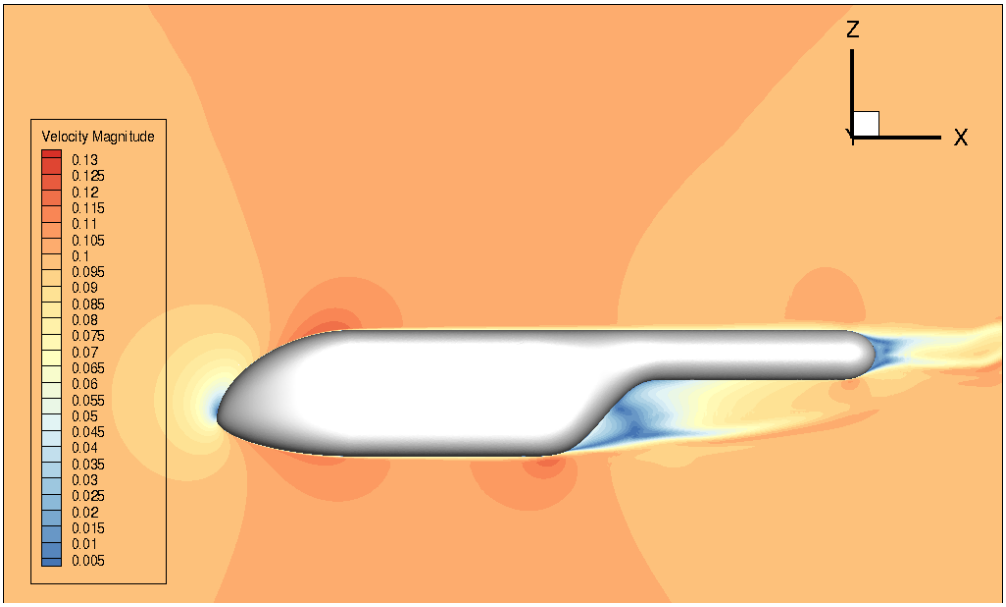


(a) Overall mesh system

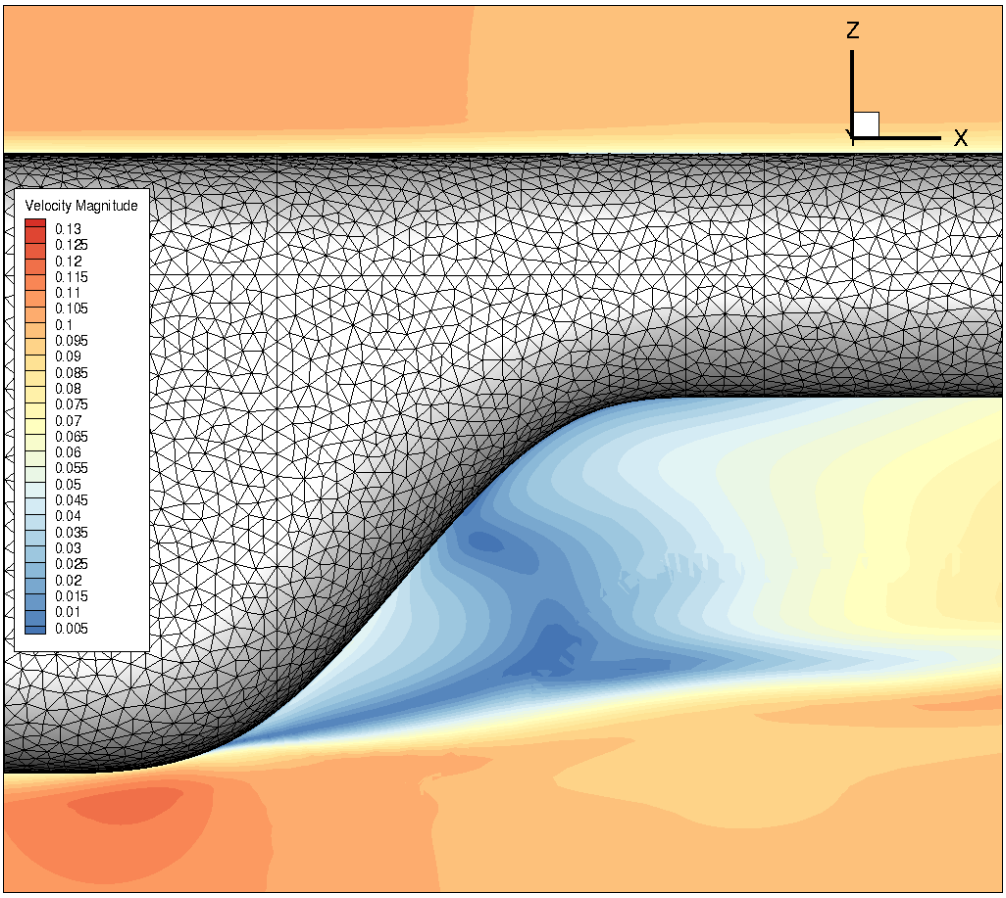


(b) Near-wall mesh system (c) Tail boom (d) Strand/Cartesian/overlap

Figure 6. High-order strand/Cartesian mesh system and connectivity.

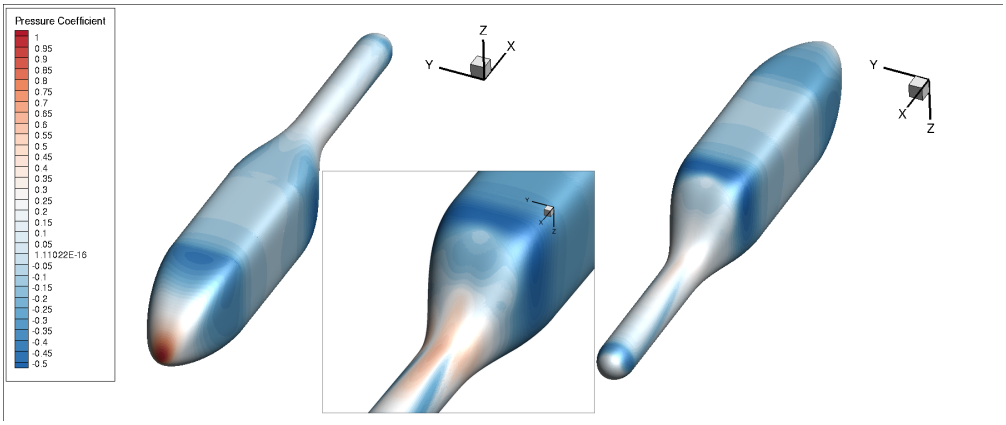


(a) Velocity magnitude overall

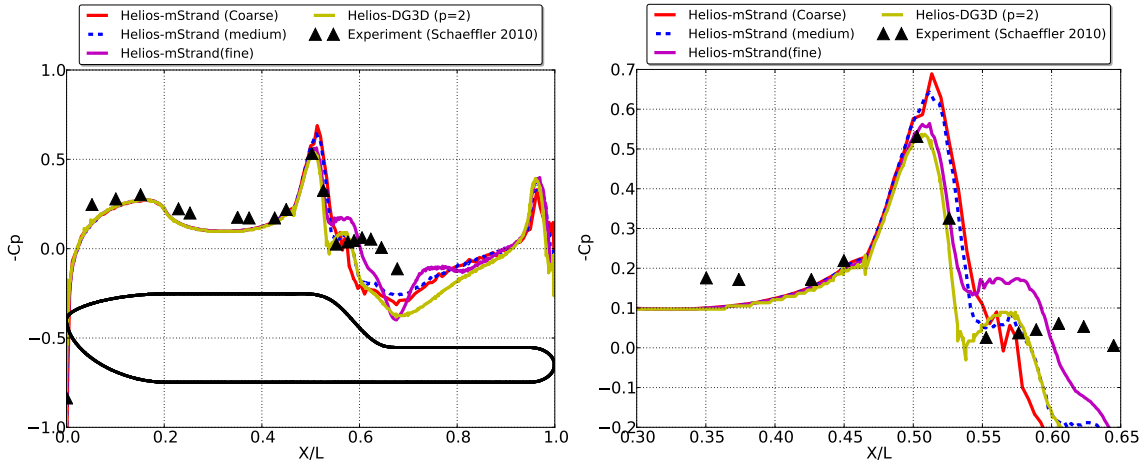


(b) Below tail boom

Figure 7. Velocity magnitude contours showing flow separation under the tail boom and aft of the fuselage.



(a) Surface Cp



(b) lower surface Cp

(c) lower surface Cp (zoomed at separation zone)

Figure 8. Surface pressure coefficient contours and comparison with experimental data/peer computations.

III.B. Overset GMRES Applied to DLR-F6 Aircraft

The DLR-F6 aircraft geometry has been used in drag-prediction workshops[58] as a challenging fixed-wing case. The pressure coefficient solution for the aircraft is shown in Fig. 9. The same geometry is used here, however with the goal of analyzing convergence using three different implicit methods. The first method is the existing Helios approach of staggering and de-coupling the convergence so that communication between meshes occurs every iteration. While this causes temporal accuracy issues for unsteady cases, steady problems do not have this problem. The staggered, decoupled approach allows for different linear-solver methods to be used in different solvers. In this case, mStrand uses its own internal GMRES while SAMCart uses a LU-SGS Gauss-Seidel method. The second implicit method used in this section is referred to a "preconditioner-only" approach, where the Gauss-Seidel preconditioners from each solver are used to march the solution towards a steady state. Overset-GMRES is the third implicit method by which a global implicit system is solved at the Python level.

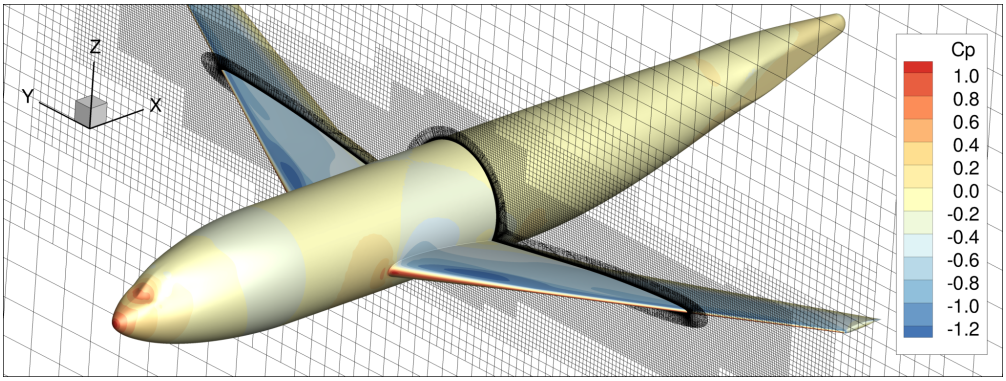


Figure 9. Pressure coefficient solution on the DLR-F6 aircraft.

Convergence of the L2 norm of the residual is shown in Fig. 10. Immediately evident is that the two cases that use only the Gauss-Seidel preconditioners diverge even with a low CFL number of 5. Both methods that use some form of GMRES, however, converge a few orders sufficient for engineering analysis. The O-GMRES method converges faster, achieving almost an extra 1-order drop in the same execution time. The O-GMRES case is augmented with a CFL controller to scale the timestep using Alg. 4. As a result, the CFL for the O-GMRES case varies between 10 and 100.

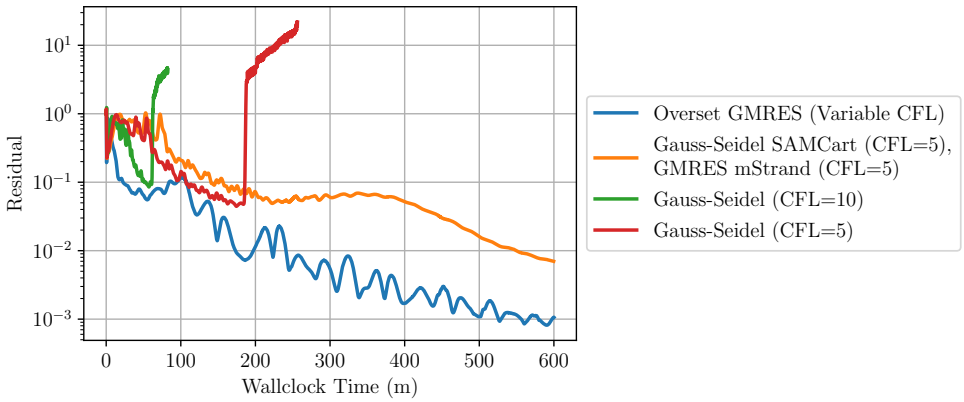
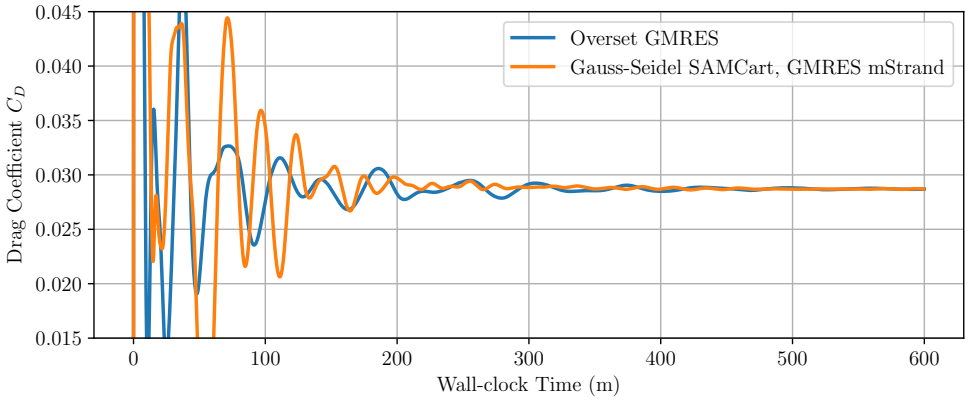
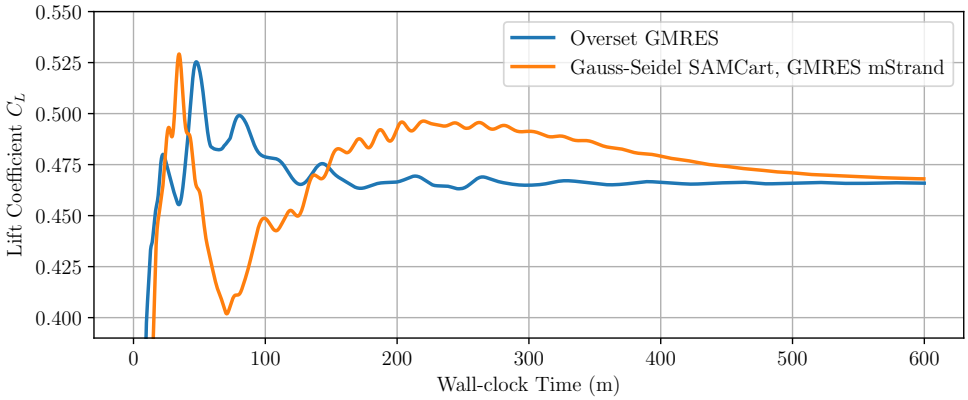


Figure 10. Convergence of residual L2 norm using different linear solvers and timesteps.

Figure 11 shows the convergence of the lift and drag forces for the aircraft between the two methods that did not diverge. Interestingly, although the residual convergence showed the O-GMRES method converging fastest, the drag plot in Fig. 11(a) show roughly the same rate of drag convergence. The lift force convergence from Fig. 11(b) is more consistent with the residual convergence showing O-GMRES converging fastest. For the staggered, decoupled approach, mStrand is using GMRES in the near-body which likely accounts for why the drag converges so quickly. The aircraft lift is more dependent on the evolution of the pressure solution in the off-body; since SAMCart is limited to a small CFL and without GMRES, the lift convergence is slower. In contrast, the O-GMRES method applies GMRES in both the near and off-body, resulting in consistent convergence globally.



(a) Drag Convergence



(b) Lift Convergence

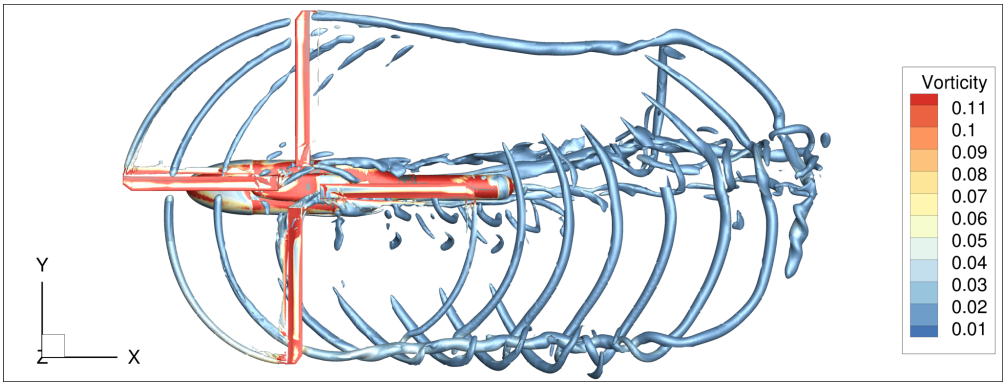
Figure 11. Convergence of (a) drag and (b) lift for two different linear-solver approaches.

III.C. Overset GMRES Applied to PSP Rotor and ROBIN Fuselage

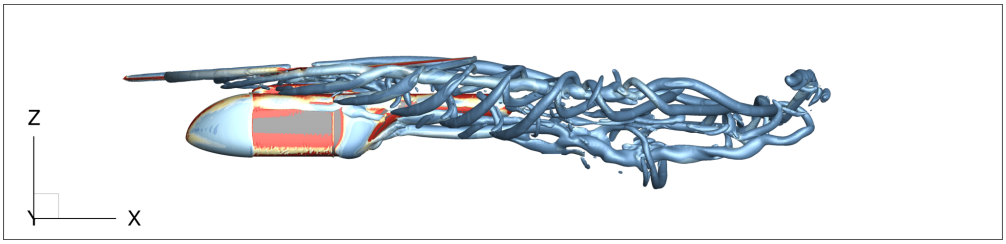
A modified version of the ROBIN fuselage with a pylon was used in experimental tests with a rotor coated with Pressure-Sensitive Paint (PSP) for analysis of laminar-turbulent transition.[59] The same grid geometries as the experiments are used with an added free-stream velocity of $M = 0.1$ to create a notional, forward-flight case for simulation with mStrand and SAMCart. The shaft-tilt of the rotor with respect to the fuselage is $-3.5°$ (forward) and fuselage angle of attack is also $-3°$ with respect to the freestream. The tip Mach number is $M_{tip} = 0.58$ for a rotor of radius $R = 66.5in$. The Reynolds number, based on blade chord ($c = 5.45in$) and tip Mach number, is 1.05×10^6.

Automatic strand meshing is used at runtime to create a mesh with 202 points in the airfoil wrap-around direction, 253 points in the span-wise direction, and 51 points in the strand-direction to a distance of $5in$. Additional triangular elements are used at the root and tip of the blade to close the geometry. Reasonable estimates of pitch and flap angles are used to generate prescribed, rigid deflections. The deflections assume a hinge location $e = 0.08R$, coning angle $\beta_0 = 1.3°$, and pitch angles $\theta_0 = 6.2°$, $\theta_{1c} = 2.2°$, and $\theta_{1s} = -2.0°$. The control angles are not intended to exactly trim the aircraft and are merely used as representative values to obtain a notional forward-flight scenario for convergence analysis.

The simulation was advanced to 2 revolutions using an azimuthal timestep of $\Delta\psi = 0.5°$. The solution is then restarted from the 2-revolution solution and the convergence was analyzed using two methods. The first applies only the preconditioner in each solver to approximate the solution of the linear system, $\mathbf{\Delta Q}$. SAMCart uses the LU-SGS preconditioner for the mean-flow equations, ADI for the turbulence equations, and mStrand uses the multi-colored Gauss-Seidel method for all equations. The second method uses Overset GMRES and variable CFL controller to vary the CFL in mStrand between 10^2 and 10^5. For unsteady cases SAMCart always uses an infinite dual-timestep.



(a) Top view



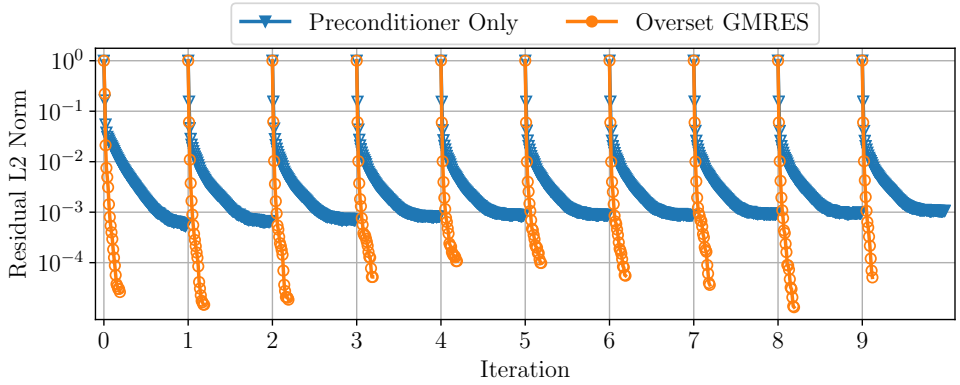
(b) Side view

Figure 12. Top and side view of the solution with iso-surfaces of Q-criterion colored by vorticity to visualize vortex structures. Both blade-vortex and fuselage-vortex interactions are observed.

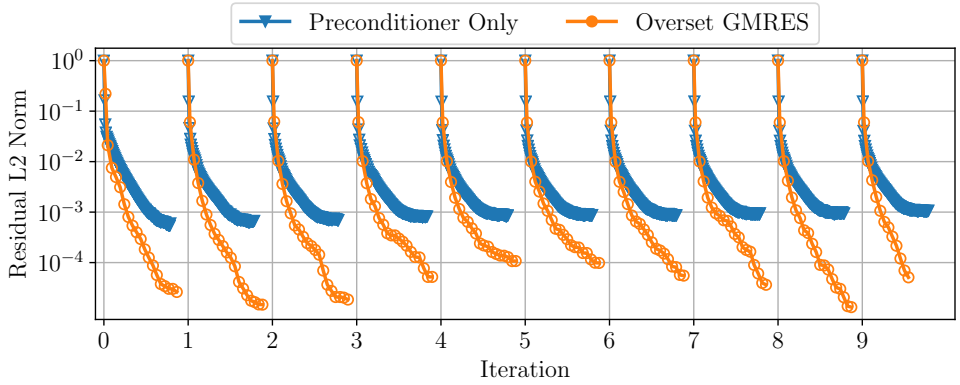
With a relatively large azimuthal timestep of $\Delta\psi = 1°$, the traditional method, shown in blue in Fig. 13(a), requires ~ 100 sub-iterations to converge 3 orders. After ~ 80 sub-iterations, convergence with the approximate linear solver stalls. With the O-GMRES method, however, convergence easily reaches 3 and even 4 orders drop in fewer sub-iterations. Each sub-iteration for the O-GMRES method applies up to 8 linear-solver (Krylov) iterations in this case, meaning each sub-iteration is approximately 8-times more computationally costly than the preconditioner-only approach. Even with the added cost, however, the O-GMRES method still converges faster over wall-clock time, as shown in Fig. 13(b).

Focusing on one timestep helps give a better understanding of the convergence behavior of the O-GMRES method. Figure 14 shows iteration 6 in detail. First, Fig. 14(a) more clearly shows the time-savings with Overset GMRES compared to the baseline approximate linear solver method. To achieve the same ~ 3 order drop, using the O-GMRES algorithm required roughly 1/3 of the time. Second, Fig. 14(b) includes the linear solver convergence as a line with smaller markers alongside the non-linear iterations. In the initial sub-iterations, both the linear and non-linear iterations converge quickly. Around sub-iteration 12, the linear-solver convergence begins to struggle to converge even half an order and would likely benefit from additional Krylov vectors. Still, the non-linear residual continues to drop through all 20 sub-iterations.

Converging 3 or 4 orders may be unnecessary for cases with small timesteps ($\Delta\psi \approx 0.25$) but is much more important for time-accurate cases with larger timesteps $\Delta\psi \approx 1°$. As the timestep increases for rotating problems there is a larger difference in flow solutions between timesteps which translates to a large linearization error in the discretized equations. Decreasing the non-linear error is done primarily in the first few sub-iterations, which is why the convergence initially drops very quickly. The convergence slop after the first few sub-iterations indicates the strength of the linear-solver. As expected, O-GMRES is a much stronger linear solver than Gauss-Seidel and the slope of the orange line in Fig. 14 is therefore significantly steeper.

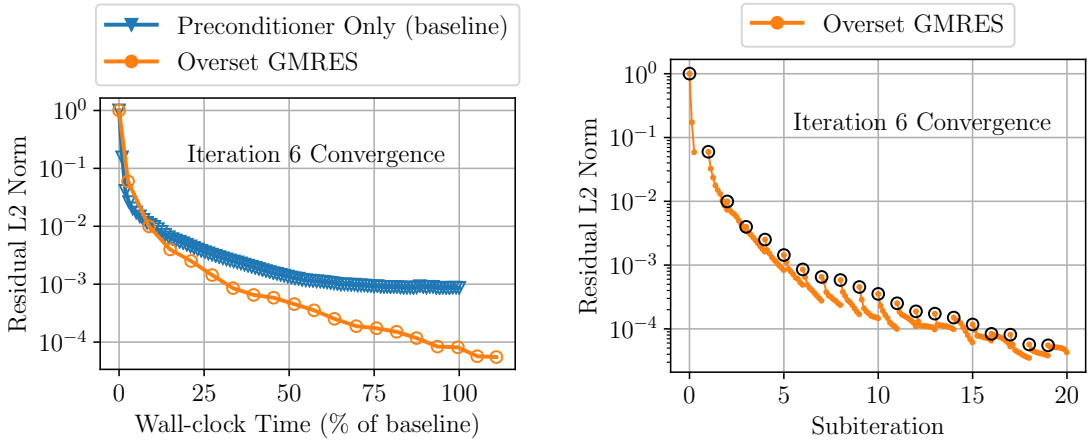


(a) Convergence vs non-linear iterations



(b) Convergence vs wall-clock time

Figure 13. Convergence in the 10 iterations after restarting (t^*) the simulation. Convergence in each timestep is shown in (a) (a) per sub-iteration and in (b) proportional to wall-clock time.



(a) Savings in wall-clock time

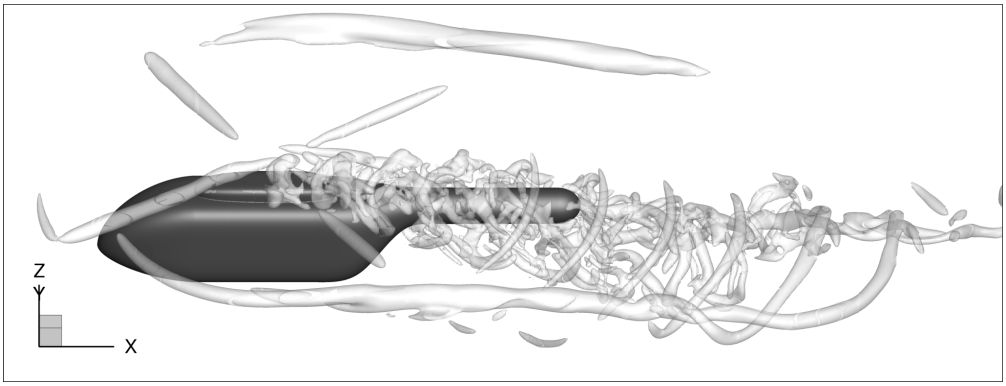
(b) Linear solver convergence

Figure 14. For one timestep, (a) shows a percentage of wall-clock time savings and (b) shows the linear convergence for each non-linear sub-iteration

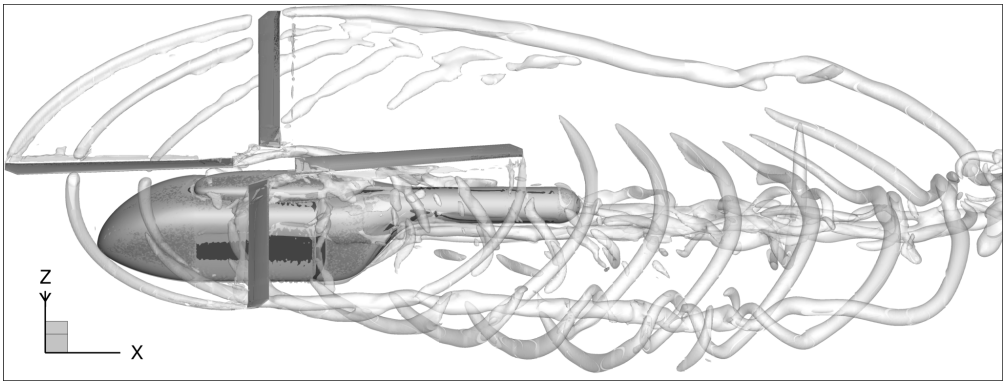
III.D. Reduced Order Aerodynamic Model for the PSP Rotor and ROBIN Fuselage

The same PSP rotor with ROBIN fuselage case from the previous section can be simulated using reduced-order models with the blades as actuator lines and the fuselage as an immersed boundary. An analytic blade model is used based on thin airfoil theory with a lift-curve slope of $C_{l_\alpha} = 6.0$ multiplied by the Glauert Mach correction. The same off-body refinement levels are used in this case with ROAM as the previous case with mStrand and the immersed boundary method applies an inviscid boundary condition at the fuselage wall. For this case ROAM uses an azimuthal timestep of $\Delta\psi = 1^\circ$ whereas the full-fidelity mStrand case uses $\Delta\psi = 0.5^\circ$.

A comparison between the Q-Criterion solution of ROAM and full-fidelity mStrand is shown in figure 15. The Q-Criterion value is based on non-dimensional units of velocity (scaled by the speed of sound) and dimensional grid in inches. The ROAM solution in Fig. 15(a) shows less defined vortex structures compared to the full-fidelity solution in (b). In both cases the wake can be seen blowing down and back onto the tail of the fuselage. ROAM shows complex vortex structures coming from the root of the blade that mStrand does not predict. This is a result of using linear airfoil theory with an analytical lift-curve slope on the root-sections of the blade which are not airfoils. Another noticeable difference in the Q-Criterion solution is around the fuselage. The ROAM immersed boundary model applies an inviscid wall boundary at the fuselage and therefore does not generate the same vorticity in the boundary layer near the fuselage.



(a) ROAM Q-Criterion Solution ($Q = 4 \times 10^{-5}$)



(b) mStrand Q-Criterion Solution ($Q = 4 \times 10^{-5}$)

Figure 15. Comparison between the wake solution from (a) the reduced-order model and (b) the mStrand model.

The evolution of thrust coefficient C_T over two rotor revolutions is shown in Fig. 16. Although mStrand predicts a slightly higher value of thrust, both results are in good agreement. The results could be improved, especially during initial transients, by using airfoil tables instead of linear airfoil theory in ROAM.

III.E. Reduced Order Aerodynamic Model for the JVX Rotor and Wing

The JVX rotor/wing configuration is a 0.68-scale V-22 model test performed to investigate wing/body interactional effects. Helios has previously been used with various near-body solvers to asses rotor thrust

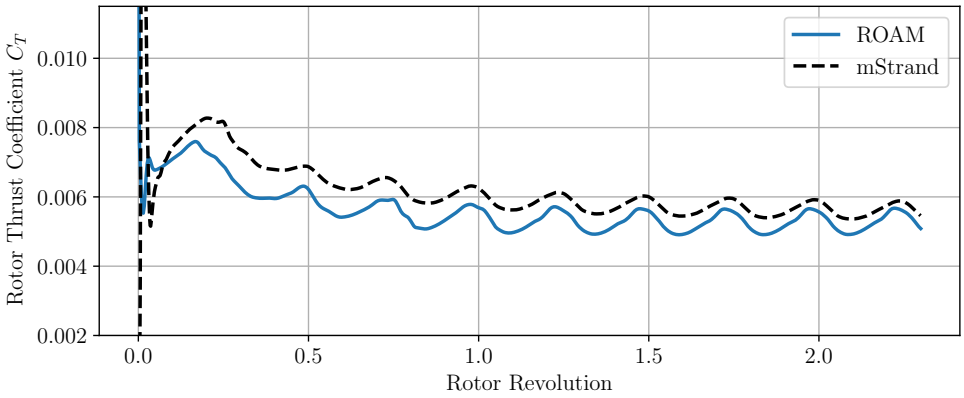


Figure 16. Comparison of thrust coefficient values between ROAM and mStrand

and wing download forces.[60] Using ROAM, the 3-bladed rotor is replaced with actuator lines and the wing is modeled as an inviscid immersed boundary. Rotor and wing loads are compared to results using mStrand and SAMCart meant to represent the high-fidelity solution. The actuator line model is applied using linear airfoil theory with a lift-curve slope of $C_{l\alpha} = 5.73$ and without the Glauert Mach correction factor.

A distinguishing feature of the JVX rotor/wing case is the separation of the flow field around the multi-element airfoil of the wing. Figure 17 shows the immersed boundary location and vorticity solution in the SAMCart meshes near the wing. The vorticity values are computed based on non-dimensional values of velocity (scaled by the speed of sound) and grid units in meters. The flow over the wing can be seen separating, resulting in large vortex structures under the wing.

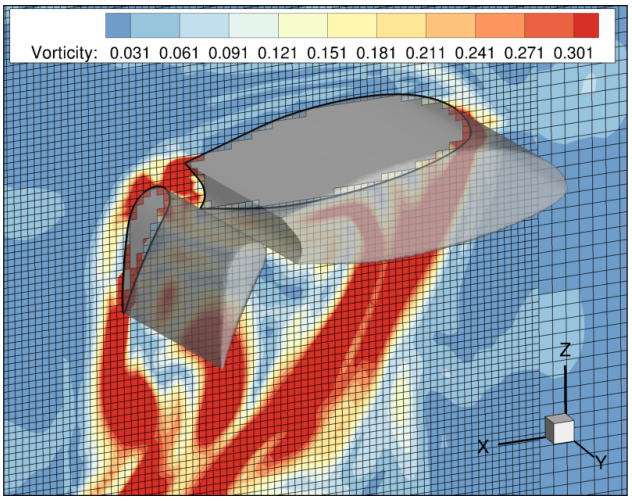
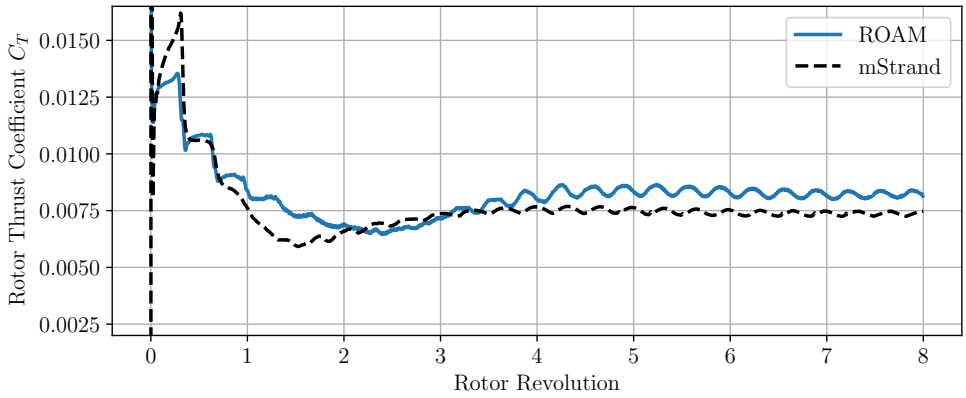
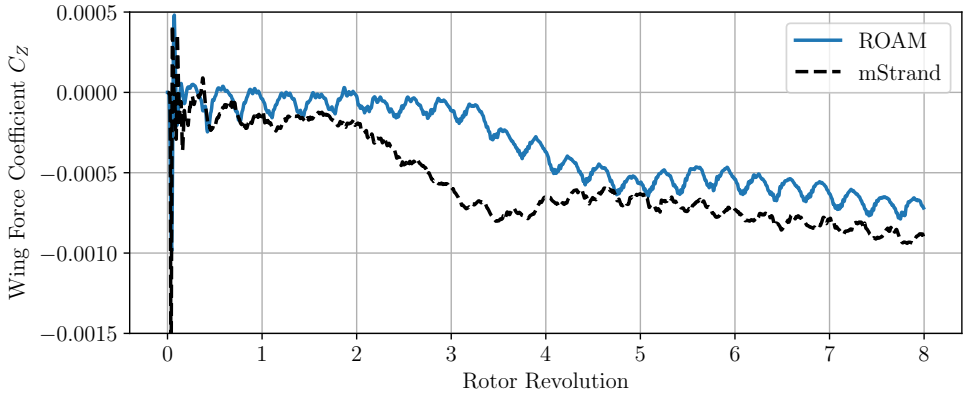


Figure 17. Vorticity solution and shedding shown below the JVX wing

The force history for the rotor using ROAM is compared to mStrand in Fig. 18(a). ROAM over-predicts the rotor thrust and shows a stronger 3/rev signal compared to the mStrand solution. In the final two revolutions, the thrust predicted by ROAM is 11% higher than the mStrand solution. The wing download coefficient, shown in Fig. 18(b) indicates ROAM is under predicting the down-ward force on the wing. This is likely due to ROAM applying an inviscid immersed boundary on the wing which would not accurately predict skin friction. The absence of a boundary layer may also affect the magnitude of shed structures under the wing.



(a) Rotor Thrust Coefficient



(b) Wing Download Coefficient

Figure 18. JVX Rotor thrust and wing download coefficients with ROAM compared to mStrand.

IV. Conclusions

Three facets of overset/embedded boundary methods were explored in this work. They are (1) convergence acceleration of multi-mesh/multi-solver paradigms through a fully coupled linear solver, the Overset-GMRES approach (2) High order near-body flow solution coupled with high-order interpolation and (3) Immersed boundary and actuator line methods for fast reduced order solution. Following are the conclusions drawn in each of these areas respectively. The Overset-GMRES method is found to be beneficial in accelerating the non-linear convergence both in terms of the number of iterations requires as well as the wall-clock time required to achieve a certain level of residual drop. Furthermore, the non-linear residual shows monotonic decrease in the case of Overset GMRES whereas the traditional method shows convergence stall. A high-order solution process was developed by interfacing a finite element based discretization scheme with a high-order Cartesian finite-difference method using high-order overset interpolation. This approach is found to produce improved agreement with measurements for prediction of separation location and post-separation pressure increase. Finally, a reduced order aerodynamic model constructed by combining an immersed boundary method for bluff bodies and actuator line for blades is found to provide reliable and fast solutions to the rotor-fuselage interaction problem.

Further enhancements are necessary in all of the areas pursued, e.g. controller design for convergence acceleration needs to be explored further, Overset GMRES approach needs to be extended to high-order systems, and the accuracy of immersed boundary/actuator line methods needs to be improved. All of these items will be pursued as future work.

Acknowledgments

Material presented in this paper is a product of the CREATE-AV Element of the Computational Research and Engineering for Acquisition Tools and Environments (CREATE) Program sponsored by the U.S. Department of Defense HPC Modernization Program Office.

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. M. Brazell was partially funded by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two DOE organizations (Office of Science and the National Nuclear Security Administration).

References

[1]Venkateswaran Sankaran, Andrew Wissink, Anubhav Datta, Jayanarayanan Sitaraman, Mark Potsdam, Buvana Jayaraman, Aaron Katz, Sean Kamkar, Beatrice Roget, Dimitri Mavriplis, et al. Overview of the helios version 2.0 computational platform for rotorcraft simulations. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 1105, 2011.

[2]Andrew M Wissink, Jayanarayanan Sitaraman, Buvaneswari Jayaraman, Beatrice Roget, Vinod K Lakshminarayan, Mark A Potsdam, Rohit Jain, Andrew Bauer, and Roger Strawn. Recent advancements in the helios rotorcraft simulation code. In *54th AIAA Aerospace Sciences Meeting*, page 0563, 2016.

[3]Jayanarayanan Sitaraman, Andrew Wissink, Venketeswaran Sankaran, Buvana Jayaraman, Anubhav Datta, Zhi Yang, Dimitri Mavriplis, Hossein Saberi, Mark Potsdam, David O'Brien, et al. Application of the helios computational platform to rotorcraft flowfields. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 1230, 2010.

[4]Scott A Morton, Brett Tillman, David R McDaniel, David R Sears, and Todd R Tuckey. Kestrel–a fixed wing virtual aircraft product of the create program. In *DoD High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2009*, pages 148–152. IEEE, 2009.

[5]David R McDaniel, Robert H Nichols, Timothy A Eymann, Robert E Starr, and Scott A Morton. Accuracy and performance improvements to kestrel's near-body flow solver. In *54th AIAA Aerospace Sciences Meeting*, page 1051, 2016.

[6]S Arevalo, C Atwood, P Bell, TD Blacker, S Dey, D Fisher, DA Fisher, P Genalis, J Gorski, A Harris, et al. A new dod initiative: The computational research and engineering acquisition tools and environments (create) program. In *Journal of Physics: Conference Series*, volume 125, page 012090. IOP Publishing, 2008.

[7]Andrew Wissink. An overset dual-mesh solver for computational fluid dynamics. In *7th International Conference on Computational Fluid Dynamics, Paper ICCFD7-1206, Hawaii*, 2012.

[8]Andrew Wissink, Sean Kamkar, Thomas Pulliam, Jayanarayanan Sitaraman, and Venkateswaran Sankaran. Cartesian adaptive mesh refinement for rotorcraft wake resolution. In *28th AIAA Applied Aerodynamics Conference*, page 4554, 2010.

[9]Jayanarayanan Sitaraman, Matthew Floros, Andrew Wissink, and Mark Potsdam. Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids. *Journal of Computational Physics*, 229(12):4703–4723, 2010.

[10]Beatrice Roget and Jayanarayanan Sitaraman. Robust and efficient overset grid assembly for partitioned unstructured meshes. *Journal of Computational Physics*, 260:1–24, 2014.

[11]Andrew Wissink, Jayanarayanan Sitaraman, Venkateswaran Sankaran, Dimitri Mavriplis, and Thomas Pulliam. A multi-code python-based infrastructure for overset cfd with adaptive cartesian grids. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, page 927, 2008.

[12]Pieter G Buning, Dennis C Jespersen, Thomas H Pulliam, WM Chan, Jeffrey P Slotnick, SE Krist, and Kevin J Renze. Overflow user's manual. *NASA Langley Research Center, Hampton, VA*, 2002.

[13]Yong Su Jung, Bharath Govindarajan, and James Baeder. Turbulent and unsteady flows on unstructured line-based hamiltonian paths and strands grids. *AIAA Journal*, pages 1–16, 2017.

[14]Dimitri J Mavriplis and Karthik Mani. Unstructured mesh solution techniques using the nsu3d solver. In *52nd Aerospace Sciences Meeting*, page 0081, 2014.

[15]Robert T Biedron, Jan-Reneé Carlson, Joseph M Derlaga, Peter A Gnoffo, Dana P Hammond, William T Jones, William L Kleb, Elizabeth M Lee-Rausch, Eric J Nielsen, Michael A Park, et al. Fun3d manual: 13.2. 2017.

[16]David Hine, Ryan S Glasby, Jon T Erwin, and Douglas L Stefanski. Results from hpcmp createtm-av kestrel components kcfd and coffe for the open source fighter. In *55th AIAA Aerospace Sciences Meeting*, page 1192, 2017.

[17]Vinod K Lakshminarayan, Jayanarayanan Sitaraman, Beatrice Roget, and Andrew M Wissink. Development and validation of a multi-strand solver for complex aerodynamic flows. *Computers & Fluids*, 147:41–62, 2017.

[18]Andrew M Wissink, Buvaneswari Jayaraman, and Jayanarayanan Sitaraman. An assessment of the dual mesh paradigm using different near-body solvers in helios. In *55th AIAA Aerospace Sciences Meeting*, page 0287, 2017.

[19]Marshall C. Galbraith, John A. Benek, Paul D. Orkwis, and Mark G. Turner. A discontinuous galerkin chimera scheme. *Computers & Fluids*, 98(0):27 – 53, 2014.

[20]Michael J Brazell, Jay Sitaraman, and Dimitri Mavriplis. An overset mesh approach for 3d mixed element high-order discretizations. *Journal of Computational Physics*, 1(322):33–51, 2016.

[21]Bin Zhang and Chunlei Liang. *A Simple, Efficient, High-Order Accurate Sliding-mesh Interface Approach to FR/CPR Method on Coupled Rotating and Stationary Domains*. American Institute of Aeronautics and Astronautics, 2015/06/01 2015.

[22]Jacob A Crabill, Jayanarayanan Sitaraman, and Antony Jameson. A high-order overset method on moving and deforming grids. In *AIAA Modeling and Simulation Technologies Conference*, page 3225, 2016.

[23]Jacob A Crabill, Freddie D Witherden, and Antony Jameson. A parallel direct cut algorithm for high-order overset methods with application to a spinning golf ball. *arXiv preprint arXiv:1711.07663*, 2017.

[24]Jennifer Abras, Nathan S Hariharan, and Robert P Narducci. Wake breakdown of high-fidelity simulations of a rotor in hover. In *AIAA Scitech 2019 Forum*, page 0593, 2019.

[25]Dylan Jude, Jay Sitaraman, Vinod K Lakshminarayan, and James D Baeder. An overset generalized minimal residual method for cfd on heterogeneous compute architectures. In *AIAA Scitech 2019 Forum*, page 0099, 2019.

[26]Dylan Jude, Jayanarayanan Sitaraman, Vinod K Lakshminarayan, and James D Baeder. An overset generalized minimal residual method for the multi-solver paradigm in helios. In *2018 Fluid Dynamics Conference*, page 3247, 2018.

[27]Thomas H. Pulliam and Joseph L. Steger. Implicit finite-difference simulations of three-dimensional compressible flow. *AIAA Journal*, 18(2):159–167, Feb 1980.

[28]PHILLIPE R Spalart and Steven R Allmaras. A one-equation turbulence model for aerodynamic flows. *La Recherche Aerospatiale*, Vol. 1, 1994, pp. 5–21.(5-21), 1994.

[29]Ronald Cools. An encyclopaedia of cubature formulas. *Journal of complexity*, 19(3):445–453, 2003.

[30]Peter D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Communications on Pure and Applied Mathematics*, 7(1):159–193, 1954.

[31]P.L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys. (USA)*, 43(2):357 – 72, 1981/10/.

[32]M. Sun and K. Takayama. An artificially upstream flux vector splitting scheme for the euler equations. *J. Comput. Phys. (USA)*, 189(1):305 – 29, 2003/07/20.

[33]R. Hartmann and P. Houston. An optimal order interior penalty discontinuous galerkin discretization of the compressible navier-stokes equations. *J. Comput. Phys. (USA)*, 227(22):9670 – 85, 2008/11/20.

[34]K. Shahbazi, D.J. Mavriplis, and N.K. Burgess. Multigrid algorithms for high-order discontinuous galerkin discretizations of the compressible navier-stokes equations. *J. Comput. Phys. (USA)*, 228(21):7917 – 40, 2009/11/20.

[35]M. Drosson and K. Hillewaert. On the stability of the symmetric interior penalty method for the Spalart–Allmaras turbulence model. *Journal of Computational and Applied Mathematics*, 246:122 – 135, 2013.

[36]Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[37]Dylan Jude, Jayanarayanan Sitaraman, Vinod K. Lakshminarayan, and James Baeder. An overset generalized minimal residual method for the multi-solver paradigm in Helios. 2018 AIAA Aviation and Aeronautics Forum and Exposition. American Institute of Aeronautics and Astronautics, June 2018.

[38]Max Blanco and David W Zingg. Fast Newton-Krylov Method for Unstructured Grids. *AIAA Journal*, 36(4):607–612, April 1998.

[39]W.Kyle Anderson, Russ D. Rausch, and Daryl L. Bonhaus. Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids. *Journal of Computational Physics*, 128(2):391 – 408, 1996.

[40]Marco Ceze and Krzysztof J. Fidkowski. Constrained pseudo-transient continuation. *International Journal for Numerical Methods in Engineering*, 102(11):1683–1703, March 2015.

[41]J Romero, Kartikey Asthana, and Antony Jameson. A simplified formulation of the flux reconstruction method. *Journal of Scientific Computing*, 67(1):351–374, 2016.

[42]Ryan S Glasby and Jon T Erwin. Introduction to coffe: The next-generation hpcmp createtm-av cfd solver. In *54th AIAA Aerospace Sciences Meeting*, page 0567, 2016.

[43]Thomas JR Hughes, Michel Mallet, and Mizukami Akira. A new finite element formulation for computational fluid dynamics: Ii. beyond supg. *Computer methods in applied mechanics and engineering*, 54(3):341–355, 1986.

[44]Brazell MJ and Mavriplis DJ. 3d mixed element discontinous galerkin with shock capturing. In *21st AIAA Computational Fluid Dynamics Conference*, page 3064, 2013.

[45]Peraire Jaime and Per-Olof Persson. High-order discontinuous galerkin methods for cfd. *Adaptive high-order methods in computational fluid dynamics*, pages 119–152, 2011.

[46]Per-Olof Persson and Jaime Peraire. Sub-cell shock capturing for discontinuous galerkin methods. *Collection of Technical Papers - 44th AIAA Aerospace Sciences Meeting*, 2:1408 – 1420, 2006.

[47]N Burgess. *An Adaptive Discontinuous Galerkin Solver for Aerodynamic Flows*. PhD thesis, University of Wyoming, 2011.

[48]G.E. Barter and D.L. Darmofal. Shock capturing with pde-based artificial viscosity for dgfem: Part i. formulation. *J. Comput. Phys. (USA)*, 229(5):1810 – 27, 2010/03/01.

[49]S.R. Allmaras, F.T. Johnson, and P.R. Spalart. Modifications and clarifications for the implementation of the spalart-allmaras turbulence model. In *7th International Conference on Computational Fluid Dynamics*, 2012.

[50]Youcef Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, March 1993.

[51]Michael J. Brazell and Dimitri J. Mavriplis. *3D Mixed Element Discontinuous Galerkin with Shock Capturing*. American Institute of Aeronautics and Astronautics, 2014/12/03 2013.

[52]Michael J Brazell, Dimitri J Mavriplis, and Zhi Yang. Mesh-resolved airfoil simulations using finite volume and discontinuous galerkin solvers. *AIAA Journal*, pages 2659–2670, 2016.

[53]Michael J Brazell, Behzad R Ahrabi, and Dimitri J Mavriplis. Discontinuous galerkin turbulent flow simulations of nasa turbulence model validation cases and high lift prediction workshop test case dlr-f11. In *54th AIAA Aerospace Sciences Meeting*, page 0861, 2016.

[54]Behzad Reza Ahrabi, Michael J Brazell, and Dimitri J Mavriplis. An investigation of continuous and discontinuous finite-element discretizations on benchmark 3d turbulent flows. In *2018 AIAA Aerospace Sciences Meeting*, page 1569, 2018.

[55]Michael J. Brazell, Dimitri J. Mavriplis, and Jayanarayanan Sitaraman. *An Overset Mesh Approach for 3D Mixed Element High Order Discretizations*. American Institute of Aeronautics and Astronautics, 2015/05/27 2015.

[56]R. Ghias, R. Mittal, and H. Dong. A sharp interface immersed boundary method for compressible viscous flows. *Journal of Computational Physics*, 225(1):528–553, July 2007.

[57]Jay Sitaraman, Beatrice Roget, Vinod Lakshminarayan, and Michael Brazell. High-order curved prismatic mesh generation using minimum distance field. In *27th International Meshing Round Table, Albuquerque, New Mexico*, 2018.

[58]Anthony Sclafani, John Vassberg, Neal Harrison, Mark DeHaan, Christopher Rumsey, S. Rivers, and Joseph Morrison. Drag prediction for the DLR-f6 wing/body and DPW wing using CFL3d and OVERFLOW on an overset mesh. In *45th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, January 2007.

[59]Austin D. Overmeyer and Preston B. Martin. Measured boundary layer transition and rotor hover performance at model scale. In *55th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, January 2017.

[60]Andrew M. Wissink, Buvaneswari Jayaraman, Steven A. Tran, Rohit Jain, Mark A. Potsdam, Jayanarayanan Sitaraman, Beatrice Roget, and Vinod K. Lakshminarayan. Assessment of rotorcraft download using helios v8. In *2018 AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, January 2018.