48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition
4 - 7 January 2010, Orlando, Florida

AIAA 2010-1230

Application of the Helios Computational Platform to Rotorcraft Flowfields

Venkateswaran Sankaran[1,*], Jayanarayanan Sitaraman[2], Andrew Wissink[1], Anubhav Datta[3], Buvana Jayaraman[3], Mark Potsdam[1], Dimitri Mavriplis[2], Zhi Yang[2], David O'Brien[4], Hossein Saberi[5], Rui Cheng[5], Nathan Hariharan[6], and Roger Strawn[1]

[1] *US Army/AFDD, Ames Research Center, Moffett Field, CA*
[2] *University of Wyoming, Laramie, WY*
[3] *Eloret Inc., Ames Research Center, Moffett Field, CA*
[4] *US Army/AED, Huntsville, AL*
[5] *Advanced Rotorcraft Technology, Inc., Mountain View, CA*
[6] *CREATE-AV, Patuxent River, MD*

This article describes the architecture, components, capabilities, and validation of the first version of the Helios platform, targeted towards rotorcraft aerodynamics. Capabilities delivered in the first version include fuselage aerodynamics with and without momentum-disk rotor models, and isolated rotor dynamics for ideal hover and forward flight coupled with aeroelasticity and trim. Helios is based on an overset framework that employs unstructured mixed-element meshes in the near-body domain combined with high-order Cartesian meshes in the off-body domain. In addition, the aerodynamics solution is coupled with structural dynamics and trim using a delta-coupling algorithm. The near-body CFD, off-body CFD, CSD and trim modules are coupled using a Python infrastructure that controls the execution sequence of the solution procedure. Specific validation studies presented include the Slowed Rotor Compound fuselage, Georgia Tech rotor body, TRAM rotor in hover and UH-60A rotor in forward flight. In all cases, Helios predictions are compared with experimental data and other state-of-the-art codes to demonstrate the accuracy, efficiency and scalability of the code.

I. Introduction

Helios which stands for Helicopter Overset Simulations is a new computational platform targeted towards high-fidelity rotorcraft aeromechanics simulations. It is a product of the High Performance Computing Institute for Advanced Rotorcraft Modeling and Simulation (HI-ARMS) and CREATE-AV[1] (Air Vehicles) programs sponsored by the DoD HPC Modernization Program Office. The eventual goal of the Helios code is to transform the analysis-test paradigm that currently exists within the US rotorcraft design community into one built around high-performance computing (HPC). Such a capability would enable engineers to anticipate and correct rotorcraft aeromechanics problems early in the design cycle, well before a prototype vehicle undergoes its first flight tests. The need for this mission is clearly apparent in view of the cost overruns and engineering development problems that have plagued historical DoD rotorcraft acquisition programs.

Rotorcraft computations are challenging because they are inherently multidisciplinary, requiring the solution of moving-body aerodynamics coupled with structural dynamics for rotor blade deformations, and vehicle flight dynamics and controls. Moreover, rotorcraft flowfields need to resolve multiple spatial and temporal scales in the unsteady problem, including 3D unsteady transonics in the advancing side, multiple dynamic stall cycles in the retreating side, wake roll-up and blade vortex interaction in the near-field, and wake inter-twinning and propagation in the far-field. Unlike their fixed-wing counterparts, rotorcraft fly in their own wake, demanding accurate prediction of these flow physics over relatively long distances into

*To whom correspondence should be addressed. Email: vsankaran@merlin.arc.nasa.gov

the field. Additionally, the trim controls of the rotor must be included to determine the right balance of the unsteady aerodynamic and inertial forces that propels, controls and holds the aircraft in equilibrium. A successful rotorcraft simulation model must therefore accurately represent the physical phenomena listed above by coupling computational fluid dynamics (CFD) with computational structural dynamics (CSD) and vehicle flight controls. The present paper describes the overall architecture, components, capabilities, and validation of the first version of the Helios code, which embodies all of these elements. The first version of Helios focuses on certain key capabilities, namely, fuselage aerodynamics, momentum-disk rotor models, and isolated rotors in hover and forward flight. Importantly, these capabilities are implemented within a flexible Python-based computational platform that facilitates ease of extensions in the future.

Helios handles the aerodynamics solutions in an innovative manner using a dual-mesh paradigm: unstructured meshes in the near-body region and Cartesian meshes in the off-body region[2,3] as shown in Figure 1. This approach has been previously used with notable success for rotorcraft computations by the OVERFLOW code,[4,5] albeit with structured curvilinear meshes in the near-body region. Helios uses unstructured near-body meshes to enable greater ease of mesh generation around complex configurations while, at the same time, preserving the accuracy of the viscous scales in the boundary layer region. On the other hand, the Cartesian off-body meshes allow the use of efficient data structures, high-order accurate discretizations and ease of mesh adaptation, all of which are crucial for accurately resolving the tip vortices in the rotor far-wake. Moreover, the near-body mesh deforms with the rotor blades and vehicle, while the off-body meshes are typically stationary. Data transfer between the two mesh systems are performed through the use of overset domain connectivity transfer technology that dynamically cuts holes and fringes in the meshes to accommodate the relative motion between the mesh systems. The near-body solution in Helios is handled by the NSU3D code developed at the University of Wyoming,[6–9] while the off-body solution is performed by the SAMARC component, which in turn is comprised of the SAMRAI Cartesian mesh infrastructure from Lawrence Livermore National Laboratory[10–13] and the Cartesian version of the ARC3D code from NASA Ames Research Center.[14] The domain connectivity framework is provided by the newly developed PUNDIT component,[15] which is fully parallel and automated in operation.

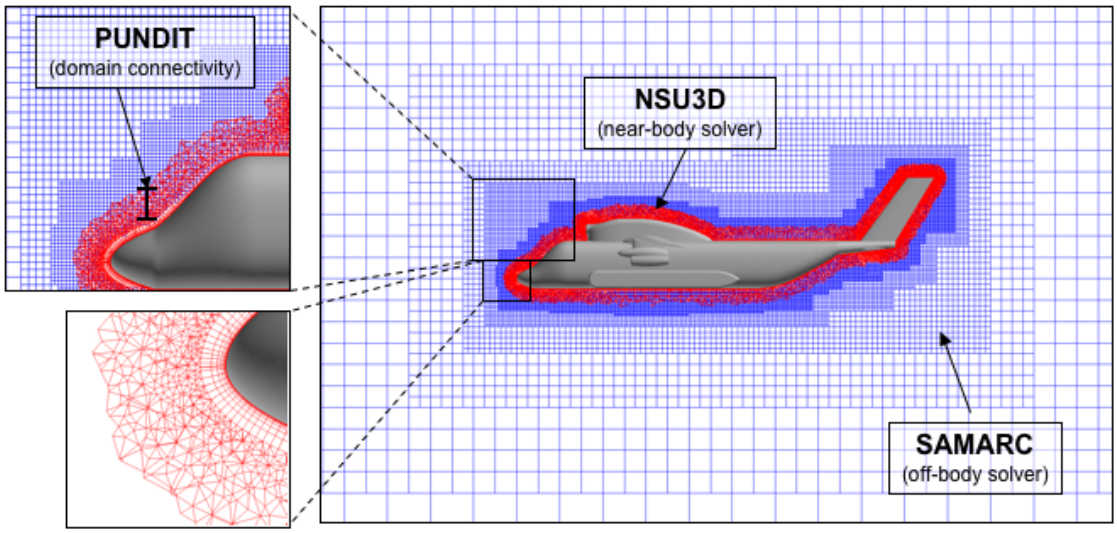

Figure 1. Dual-mesh paradigm used in the Helios code with unstructured near-body grids to capture geometric features and boundary layer near the body surface, and block-structured Cartesian grids to capture far-field flow features.

Rotor structural dynamics and trim modeling, jointly referred to as comprehensive analysis (CA), utilizes the delta coupling or loose-coupling procedure originally developed by Johnson and his co-researchers.[16] The success of this procedure has driven a renewed interest in rotorcraft-CFD in the U.S.[17–24] and Europe.[25–28] In Helios, the coupling between aerodynamics and comprehensive analysis occurs through rotor fluid-structure interface (RFSI) module, which takes the three-dimensional surface patch forces from the near-body CFD solver and imposes them on a one-dimensional beam model for the blade structural dynamics. The fluid-structure interface ensures that these imposed forces conserve energy for the non-linear beam structural dynamics while also preserving the total airloads. The blade deformations and motions obtained from the

comprehensive analysis are in turn communicated back to the aerodynamics through a mesh deformation module (MDM) that appropriately transforms the near-body mesh definition to conform to the new blade shape and position. The CSD and trim analysis capabilities are handled by the Rotorcraft Comprehensive Analysis System, RCAS, developed by the U.S. Army Aeroflightdynamics Directorate (AFDD) and Advanced Rotocraft Technology, Inc. (ART).[29] Finally, the near-body mesh deformation is handled by an internally developed module called "rdeform".

Helios utilizes a Python-based Software Integration Framework (SIF) to couple the diverse software modules that are responsible for the aerodynamics, domain connectivity, structural dynamics, vehicle flight dynamics, fluid-structure interface and mesh deformation.[2,3] As noted above, many of the these modules are derived from legacy single-discipline computer codes written in a variety of languages: FORTRAN, C and C++. The use of Python enables the wrapping of these codes into modules or components and makes them available as shared objects for the computations.[30] Moreover, the transfer of data between the different components is facilitated by the Python layer with minimal data replication through the use of pointers for all data-intensive arrays. We emphasize that SIF is a light and flexible infrastructure which accesses the components at a very high-level and requires little customization or rewriting of the underlying legacy (or new) codes. It is comprised of a small number of run-mode scripts which control the execution sequence of the underlying components within a time-stepping or iterative framework. The SIF code also specifies a set of well-defined component interfaces that would potentially allow the underlying codes or components to be re-engineered or even changed entirely without interfering with the other codes in the infrastructure.

Helios is nearing the beta release of the first version, called "Whitney", which provides capabilities for: (1) stand-alone fuselage aerodynamics with and without momentum-disk model for the rotor, (2) isolated rotor analysis of ideal hover (i.e., steady hover) in rotating and inertial frames, and (3) isolated rotor with aeroelastic coupling effects. Additional functionalities will be introduced in future releases, planned to follow an approximately annual release cycle. The software product is accompanied by a front-end graphical user-interface, which is based upon a common graphical engine developed by the Kestrel team,[31] that is responsible for the fixed-wing simulations product of CREATE-AV. Like all CREATE-AV products, Helios has undergone a series of product acceptance tests by the CREATE-AV ShadowOps team prior to the actual beta release. Several validation studies have been performed to test the different capabilities of the Helios-Whitney code. These include: fundamental aerodynamics studies such as the NACA 0015 wing, DLR-F6 wing-body, the Slowed Rotor Compound fuselage; momentum-disk validation for the Georgia Tech rotor body and the V22 configuration; isolated rotor in ideal hover in rotational and inertial frames for the TRAM and model-scale UH-60A; and isolated rotor in forward flight for the full-scale UH-60A. The present article summarizes a select sample of these validation studies, namely the slowed rotor compound fuselage, the Georgia Tech rotor body, the TRAM rotor in ideal hover and the full-scale UH-60A in forward flight. In addition to experimental data comparisons, Helios predictions are also compared with other computer codes such as OVERFLOW[5] and FUN3D[32] for some of the cases.

The remainder of the paper is organized as follows. In the next section, we present details of the Helios infrastructure, including a description of the Python-based SIF, as well as brief overviews of the Helios components, namely NSU3D, SAMARC, PUNDIT, RFSI, RCAS and *rdeform*. In Section III, we discuss the run-mode scripts that are employed in Whitney in concert with the targeted capabilities. In the following section, we present the results of the validation studies for each capability. Finally, we provide a summary as well as plans for the second version of the Helios code.

II. Infrastructure and Components

Helios is comprised of a number of software components coupled together by a Python-based Software Integration Infrastructure or SIF. In this section, we describe SIF and the various components that are responsible for the aerodynamics, structural dynamics and trim controls applied to rotorcraft simulations. In addition, we also discuss software elements outside of the core Helios development such as the graphical user interface and the run-time environment.

II.A. Software Integration Framework (SIF)

A schematic of the Helios software infrastructure identifying the different codes, their interfaces and execution pattern is shown in Figure 2. The infrastructure discussed here couples the following codes through a Python

infrastructure: (a) the unstructured near-body solver, NSU3D, (b) the high-order Cartesian off-body solver, SAMARC, (c) the domain-connectivity module, PUNDIT, (d) the rotor fluid-structure interface module, RFSI, (e) the rotorcraft comprehensive analysis system, RCAS, and (f) the mesh deformation module, *rdeform*. Many of these components such as NSU3D and RCAS are existing standalone legacy codes; others like PUNDIT, RFSI and *rdeform* are newly developed components; finally, SAMARC is a newly developed component that is built using legacy modules such as SAMRAI and ARC3D.

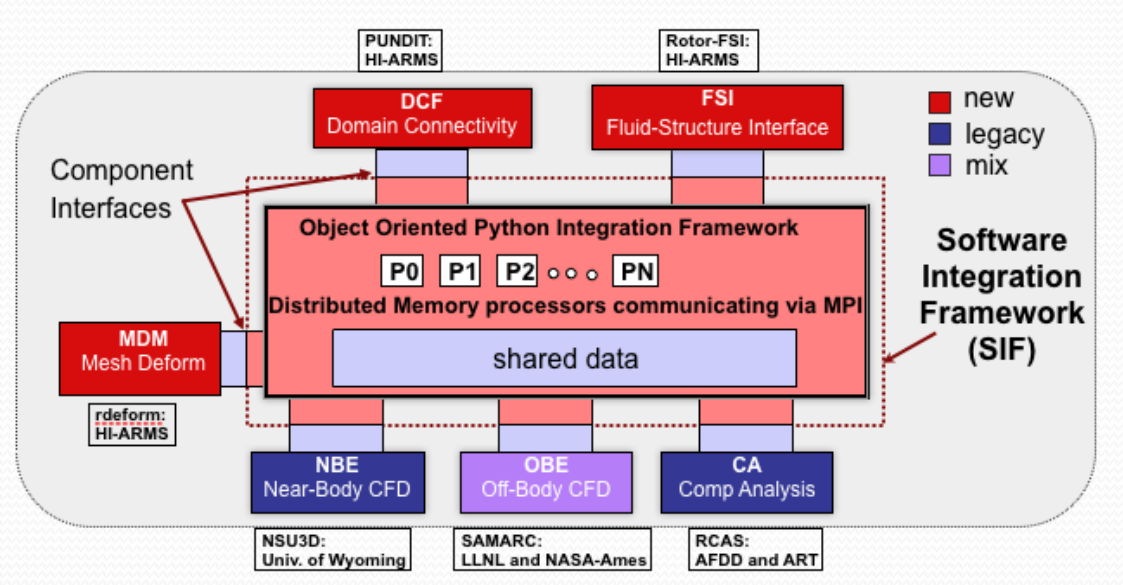

Figure 2. Software infrastructure utilized in Helios

In the figure, SIF is shown encased by the dotted line, while the components are shown linked to SIF. Each component is comprised of the "standalone" module, shown as a colored box, and an interface section, which links the module to SIF. The interface itself is in two parts: the SIF part of the interface (shown in light red) that contains the generic method calls which are not dependent upon the particular identity of the component, and the component part of the interface (shown in light blue) which takes care of the specific data translations (such as appropriate non-dimensionalizations and coordinate transformations) that are specific to the particular component code. For instance, the NSU3D and SAMARC codes utilize different non-dimensionalizations of the fluid dynamics variables, and the conversion between these two forms needed for data exchange occurs in the interface code.

Python supports object orientation and the infrastructure uses this paradigm, treating each module as an object in the main Python script. Each module allocates the required memory for its grid and solution variables and the module interfaces are designed to expose only the variables that need to be known to other modules. Thus, the amount of data that must actually be exchanged between modules is small, and therefore most of the internal data structures for the modules are protected and are not accessible at the Python level. This strategy ensures that the Python infrastructure incurs negligible cost overheads.[2] Parallel execution is achieved using pyMPI, which allows the use of the same parallel HPC computer systems traditionally used for large-scale CFD calculations. At present, we are using the native load balancing capabilities of NSU3D and SAMARC to distribute the solvers data across processors, and then executing each module in a sequential fashion. PUNDIT manages any inter-processor data communication that takes place when exchanging overset data between the two solvers. This has proven to be an effective strategy for the problems we have run to date, although in future versions we anticipate that assigning different numbers of processors for different solver modules would be necessary to maintain optimal load balancing for large problem sizes. RCAS and RFSI are presently executed on a single processor because there is negligible associated latency for the applications of interest here.

Brief descriptions of the different components used in the Helios-Whitney code are given below. More details of SIF and the components are provided in previous work.[2,3,15]

II.B. Near-Body Flow Solver

Helios can operate both in the fully-unstructured mode, wherein the unstructured solution extends from the body surface all the way into the farfield, and in the so-called dual-mesh mode, wherein the unstructured solution is restricted to the near-body region and Cartesian meshes are used in the off-body region. In both cases, we refer to the unstructured solver as the near-body solver. The unstructured near-body solver used in Helios is the NSU3D code developed by Mavriplis at the University of Wyoming.[6–9] The NSU3D code also has provisions for source terms arising from a momentum-disk model for representing rotor effects. The momentum-disk model is contained within a separate module called SMEMRD, developed by O'Brien at U.S. Army AED.[33] We note that the SMEMRD module does not directly communicate with SIF; rather, it plugs into the NSU3D when the momentum-disk option is invoked. In the following, we briefly describe the capabilities and algorithms in the NSU3D and SMEMRD modules.

II.B.1. Unstructured Near-Body Navier-Stokes Solver (NSU3D)

The near-body solver, NSU3D, is an unstructured mesh Unsteady Reynolds-Averaged Navier-Stokes (URANS) solver developed for high-Reynolds number external aerodynamics applications.[6–9] The NSU3D discretization employs a second-order accurate node-centered approach, where the unknown fluid and turbulence variables are stored at the vertices of the mesh, and fluxes are computed on faces of a dual control volumes, with each dual face being associated with a mesh edge. This discretization operates on hybrid mixed-element meshes, generally employing prismatic elements in highly stretched boundary layer regions, and tetrahedral elements in isotropic regions of the mesh. A single edge-based data structure is used to compute flux balances across all types of elements. The single-equation Spalart-Allmaras turbulence model is used for representing the Reynolds stress terms within the NSU3D solver.

The NSU3D solution scheme was originally developed for optimizing convergence of steady-state problems. The basic approach relies on an explicit multistage scheme which is preconditioned by a local block-Jacobi preconditioner in regions of isotropic grid cells. In boundary layer regions, where the grid is highly stretched, a line preconditioner is employed to relieve the stiffness associated with the mesh anisotropy. An agglomeration multigrid algorithm is used to further enhance convergence to the steady-state. The Jacobi and line preconditioners are used to drive the various levels of the multigrid sequence, resulting in a rapidly converging solution technique. For time-dependent problems, first- and second-order implicit backwards difference time discretizations are implemented, and the line-implicit multigrid scheme is used to solve the non-linear problem arising at each implicit time step. NSU3D has been extensively validated in stand-alone mode, both for steady-state fixed-wing cases as a regular participant in the AIAA Drag Prediction workshop series,[34,35] as well as for unsteady aerodynamic and aeroelastic problems,[36] and has been extensively benchmarked on large parallel computer systems.[37]

For operation within an overset environment, "iblanking" capability has been added. The iblank variable specify which nodes the solution variables are to be updated in the near-body solver (iblank = 1) and which nodes are not updated by the solver, i.e., fringes and holes (iblank = 0). The fringes correspond to the overset regions of the near-body grid, which are updated by the off-body code, while holes correspond to locations where the grid intersects solid objects and therefore flow solutions are not required.

II.B.2. Momentum-Disk Module (SMEMRD)

The Software Module for Engineering Methods of Rotor Dynamics (SMEMRD) contains the momentum-disk model for efficiently representing rotor effects. This module is an extension of the momentum-disk model developed at Georgia Tech,[38] which has been enhanced to include a generalized solver interface, a generalized blade property description, airfoil-table-based airloads, and multi-rotor trim capability. The model is based on the source term approach of Rajagopalan,[39] which simplfies the rotor by representing it as an infinitesimally thin disk that imparts both momentum and energy to the surrounding flow. At each iteration, SMEMRD uses the updated velocity field computed by the CFD solver to determine a new rotor loading, which is then passed back to the solver via momentum and energy source terms. This process continues until the rotor loading and flow field converge. At the present time, the module plugs directly into the near-body solver (NSU3D) and does not directly interact with any of the other components. For this reason, the momentum-disk capability is currently available only when Helios is executing in single-mesh or fully unstructured mode and cannot be invoked when operating in the dual-mesh mode. Further details are available in the SMEMRD documentation.[33]

II.C. Cartesian Off-Body Solver (SAMARC)

In Helios' dual-mesh mode, the off-body solution is handled through the automated generation of Cartesian meshes and high-order solution of the fluid equations. The off-body solution process is carried out by the the SAMARC component which is a combination of the Structured Adaptive Mesh Refinement framework called SAMRAI, developed at Lawrence Livermore National Laboratory,[10–13] and the Cartesian version of the ARC3D (called ARC3DC) solver developed by Pulliam.[14] SAMRAI manages the automatic generation of meshes around geometries as well as the mesh partitioning, domain decomposition and parallel MPI-based communication ensuring efficiency on parallel computer systems. ARC3DC performs the fluid dynamic solution in each Cartesian grid block. Both of these modules are briefly described below.

II.C.1. Cartesian Mesh Infrastructure (SAMRAI)

SAMRAI stands for the Structured Adaptive Mesh Refinement Applications Infrastructure.[10–13] It represents a Cartesian grid system that consists of a hierarchy of nested refinement levels with each level composed of a union of rectangular grid regions. All grid cells on a particular level have the same spacing, and the ratio of spacing between levels is generally a factor of two, although it is possible to use other refinement ratios as well. Grid generation, load balancing, and parallel adaptive data exchanges between blocks in the off-body solver are all performed by SAMRAI. The Cartesian grid system offers a number of computational advantages over traditional unstructured meshes. Grid storage costs are minimal because it is only necessary to store the boundaries of each grid block. Cartesian grids also require no metric terms so the number of operations in the spatial discretization are minimal. Additionally, it is straightforward to implement high-order schemes on Cartesian grids as discussed in concert with ARC3D. Finally, the multi-level representation makes grid adaption relatively straightforward. Although the ability to do Cartesian mesh adaption is available within SAMRAI, it has not been activated in the Whitney version of the Helios code. Future versions of the software will enable this feature with further anticipated gains in accuracy and efficiency.

The off-body grids are constructed automatically. In the Helios-Whitney version, the procedure follows the fixed refinement option in SAMRAI, wherein the end-user specifies the extents of the coarsest and finest levels as well as the number of refinement levels used. Typically, the finest grid region is in the form of a bounding box that encompasses the near-body mesh and the number of refinement levels are chosen so that the Cartesian fine-mesh spacing is comparable to the unstructured cell size at the outer fringe of the near-body mesh. At the beginning of the time-step, data on fine patch boundaries are updated either by copying data from a neighboring patch on the same level (if one exists), or through data interpolation from a coarser level. All levels then execute the time-integration scheme by invoking the ARC3DC solver on a block-by-block basis. After updating the solution for a single time-step (or a single sub-step when multi-stage schemes are used) on all patches and on all levels, data from the fine mesh are injected into coarse levels wherever they overlap. Further details are given by Wissink *et al.*[2]

II.C.2. Cartesian Euler Solution Module (ARC3DC)

The flow solution on each Cartesian grid block is performed by ARC3DC, a version of the well-known ARC3D code with high-order algorithms optimized for Cartesian grids.[14] The code solves the Euler equations since it is active only in the off-body regions of the flow, wherein the cell-Reynolds numbers are large and viscous effects are negligibly small. Spatial central differencing schemes of second-, fourth-, and sixth-order spatial accuracy are available. In addition, artificial dissipation schemes of first-, third-, and fifth-order have been implemented. Explicit 3rd-order Runge-Kutta (RK) time-stepping is used for time integration. The solver has provision for "iblanking" to account for overset meshes. Further, each Cartesian block is provided with the appropriate number "fringe" layers to account for the stencil size at block boundaries. A variety of far-field boundary conditions are available including free-stream Dirichlet, inflow/outflow conditions, and Riemann conditions.

II.D. Domain Connectivity Module (PUNDIT)

Data transfer between the near-body and off-body overset meshes are handled by a newly developed domain connectivity module called PUNDIT, which is an acronym for Parallel Unsteady Domain Information Transfer.[15] The formulation is completely parallel and automated without any need for user-intervention. For static mesh operation, PUNDIT performs the near-body/near-body and near-body/off-body donor search

and hole/fringe identification at the start of the computation. Each iteration involves one solution step of the near-body and off-body domains, following which PUNDIT is invoked to perform the data interpolation between the overset mesh systems. For dynamic mesh operation, the donor and hole identification steps have to be carried out at every time-step, which underscores the need for parallel and automated operation.

PUNDIT utilizes an implicit-hole-cutting methodology for identifying donors and receivers and for marking holes and fringes.[40] The core idea of this approach is to retain the grids with the finest resolution at any location in space as part of the computational domain and interpolate data at all coarser grids in this region from the solution on the fine grid. This results in the automatic generation of optimal holes without any user specification as in the case of explicit hole-cutting. PUNDIT uses the cell volume as the resolution capacity for a grid cell and the average of cell volumes of all associated grid cells as the resolution capacity for a grid node. In addition to near-body/off-body connectivity, PUNDIT can also perform near-body/near-body connectivity which is required when there are multiple near-body meshes (as is the case when a multi-bladed rotor mesh is generated by replicating the single-blade mesh multiple times). Additional details of the PUNDIT formulation are given by Sitaraman *et al.*[15]

II.E. Rotor Comprehensive Analysis System (RCAS)

Rotor structural dynamics and trim coupling are handled by the Rotorcraft Comprehensive Analysis System (RCAS).[29] RCAS is a rotorcraft comprehensive analysis (CA) platform developed and maintained by U. S. Army Aeroflightdynamics Directorate (AFDD) in partnership with Advanced Rotorcraft Inc. over the past ten years. In general, RCAS contains state-of-the-art multibody, large deformation, composite nonlinear beam finite-element modeling (FEM), full aircraft trim and vehicle dynamics, and lower-order unsteady lifting-line aerodynamic models. It also contains a suite of solution procedures for level flight trim solution, flutter and aeroelastic stability, ground/air resonance capabilities, unsteady maneuvers, and external slung load/missile release capabilities. In Helios-Whitney, the RCAS component supplies the CSD and trim components. Importantly, it provides the delta-coupling capability that is central to multi-disciplinary CFD-CA coupled analysis.[16] Additional capabilities such as tight-coupling, multiple rotors and rotor-fuselage configurations are not supported in Whitney, but will be included in future versions of Helios.

II.F. Rotor Fluid Structure Interface (RFSI)

The rotor fluid structure interface model or RFSI is responsible for transferring the aerodynamic forces from the near-body CFD solver to the structural dynamics model of the blade. Specifically, it takes 3-D surface patch forces from CFD near-body solver (NSU3D), calculates the corresponding section airloads and imposes them on the 1-D beam models in the CSD solver, RCAS, in this instance. The imposition is conservative (energy) in limit and preservative (total airloads). The method implemented in RFSI is completely generic and is applicable to both structured or unstructured grids i.e., there is no requirement that sectional airloads be calculated in the CFD code itself. Each individual surface patch force is transferred to the structure, consistent with the high-fidelity CFD in Helios. RFSI also ensures consistency (of geometry and frames) between the CFD and CSD solvers. In Helios-Whitney, the module works for an isolated single rotor. Multi-element blades (flaps, slats, morphing etc) and multiple rotors (tandem, coax, tiltrotors) are extensions planned for subsequent releases. Additional details are given by Choi and Datta.[41]

II.G. Mesh Deformation Module

The final component that plugs into Helios are the mesh motion and deformation modules. There are two specific sub-modules: *pydeform* which is used for prescribed motion cases, and *rdeform* which is used for full aero-elastic coupling. In both instances, the module uses the original mesh coordinates and moves and deforms the mesh to account for rotor rotation and blade surface deformations.

II.H. Helios User Interface

The Helios graphical user interface (GUI) is a graphical front-end that allows the end-user to set up run-time inputs prior to execution. The Helios-GUI is based upon the graphical engine developed by the Kestrel team[31] and is customized by adding appropriate Helios templates and configuration files to the code base. In fact, there are two specific Helios-GUI's: (1) the pre-processor-GUI that enables preprocessing of the unstructured meshes used in Helios, and (2) the run-time GUI that sets up the input parameters for the

different components in Helios. The pre-processor GUI aids the user in a number of grid-processing tasks. Basically, it can convert the native cell-based grid format output by various grid generators to the edge-based format that is required by NSU3D. Secondly, it can help perform the mesh sub-setting needed to obtain a near-body mesh from a fully unstructured mesh. This is generally done by specifying a cutting distance from the surface beyond which the mesh nodes of the full mesh are eliminated. Further, the pre-processor-GUI can also do various mesh manipulations such as mesh duplication (eg., to replicate a single blade mesh into a multi-bladed mesh) and mesh rotations and translations to enable fixed collective changes. Finally, it also carries out mesh partitioning to execute the problem on multiple processors. We emphasize that all mesh preprocessing is restricted to the near-body mesh—the off body mesh generation and domain connectivity operations are carried out automatically at the time of execution.

The second GUI is the run-time GUI, which sets up the input parameters for the different codes or components within Helios. The GUI is comprised of different panes for each of the relevant components. Common inputs are specified on the front SIF pane, while component-specific inputs are entered through the appropriate component panes. Since many of the components within Helios-Whitney are legacy codes, the run-time GUI supports the legacy input decks that the stand-alone codes or components typically utilize. Importantly, common inputs are read in by SIF and then distributed to all the components through a *sifInitialize* call. This step ensures that all components are solving the same problem with the correct input parameters. The run-time inputs include the parameters necessary for generating the Cartesian-based off-body mesh. Figure 3 shows the SAMARC inputs' pane. As described earlier, Helios-Whitney supports

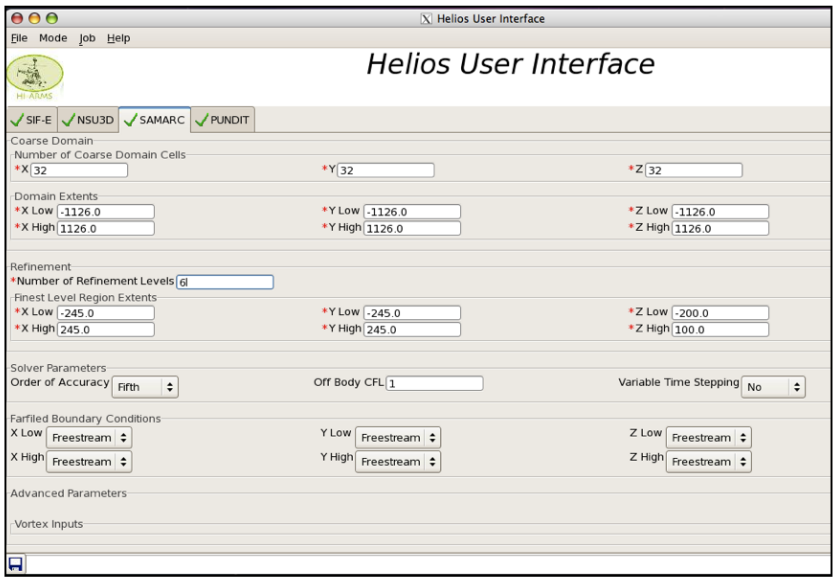

Figure 3. Helios-run-time-GUI showing the SAMARC input pane which contains the inputs required for generating the Cartesian off-body mesh system.

the fixed refinement option in SAMARC which requires the user to specify the coarsest- and finest-level bounding boxes and the number of refinement levels. The rest of the mesh generation process is performed automatically during execution.

II.I. Run-Time Environment

The Helios platform utilizes a variety of codes and languages. For instance, NSU3D is written in FORTRAN-77 and FORTRAN-90, SAMARC is written in C++, SIF is written in Python, the GUI is implemented using wxPython and so on. To minimize compiler and library variations between different commodity clusters, Helios is built within a customized run-time environment called *ptoolsrte* developed by Shende of ParaTools, Inc.[42] This run-time environment provides the infrastructure for supporting multi-language Python-based software and includes packages such as pyMPI, SWIG, f2py, Numpy, SciPy, matplotlib, wxWidgets, wx-Python, and so on. The overall installation procedure then involves first installing *ptoolsrte* at the system level on the target cluster and then installing the Helios code within this environment.

III. Helios Execution

Helios-Whitney delivers three main solution capabilities, namely, (1) isolated fuselage aerodynamics with and without a momentum-disk rotor model, (2) isolated rotor in ideal hover (i.e., steady hover) in rotational (i.e., non-inertial) and inertial frames, and (3) isolated rotor in forward flight with aeroelastic and trim coupling. Some of the capabilities can be exercised either in single-mesh (fully unstructured) or dual-mesh (unstructured-Cartesian) mode, some involve static meshes and others involve moving meshes, and only the last of the above capabilities involves coupling with comprehensive analysis. This latter case represents the highest fidelity analysis mode in the current version of Helios and involves a prescribed shaft thrust and hub moment trim procedure.

According to the particular choice of capability and the mode of execution, we can list seven specific use-cases which are given in Table 1. As is evident from the table, each use-case is associated with a particular

Table 1. List of Use-Cases in Helios-Whitney

NBE - Near-Body Engine, OBE - Off-Body Engine, DCF - Domain Connectivity Framework, CA - Comprehensive Analysis, FSI - Fluid Structure Interface, MDM - Mesh Deformation Module

Use-Case	NBE	OBE	DCF	CA	FSI	MDM	Run-Mode
Single-Mesh (Unstructured) Aero	Yes*	—	—	—	—	—	Static-CFD
Dual-Mesh Aero	Yes	Yes	Yes	—	—	—	Static-CFD
Single-Mesh Rotating Frame (hover)	Yes	—	—	—	—	—	Static-CFD
Dual-Mesh Rotating Frame (hover)	Yes	Yes	Yes	—	—	—	Static-CFD
Dual-Mesh Inertial Rigid Rotor	Yes	Yes	Yes	—	—	Yes	Dynamic-CFD
Dual-Mesh Inertial Presc. Motion	Yes	Yes	Yes	—	Yes	Yes	Dynamic-CFD
Dual-Mesh Inertial with CA	Yes	Yes	Yes	Yes	Yes	Yes	CFD-CA Delta-Coup

* Momentum-disk (SMEMRD) is available only for the single-mesh (unstructured) aerodynamics use-case.

combination of components that are executed by SIF. The execution itself is controlled by a run-mode Python script that essentially acts as the main program. Overall, there are three run-mode scripts: (1) Static-CFD that is used for static-mesh problems, (2) Dynamic-CFD used for dynamic-mesh problems, and (3) Delta-Coupling used for CFD-CA coupling. It should be noted that the Static-CFD run-mode covers problems with stationary meshes, i.e., use-cases 1 and 2, as well as problems with rotating blades that are solved in a rotational frame, i.e., use-cases 3 and 4. The Dynamic-CFD run-mode is used for rotor problems using an inertial frame of reference and includes both hover (use-case 5) and forward flight with prescribed blade motions (use-case 6). The Delta-Coupling run-mode is used for forward flight with aeroelastic and trim coupling (use-case 7). We note that, at the time of Helios execution, the end-users need only to select the appropriate use-case through the Helios-GUI and the required components and run-mode scripts are automatically invoked. The three run-mode scripts are further discussed below.

III.A. Static CFD Analysis

The static-CFD run-mode script is used for the first four use-cases in Table 1 and involves cases where both near-body and off-body meshes remain fixed relative to one another and the domain connectivity donor search and hole-cutting operations are performed only at the start of the computation. Figure 4 shows the flow chart with the execution sequence used in the static-CFD run-mode script. At the start of the computations, during the *initialize* call,the components read their respective input files. The SIF input file specifies the identities of all components needed for a particular use-case as well as global data parameters. As part of the *initialize* step, SIF also initiates the *sifInitialize* call, which sends the common input data to all the components so that they can over-write their default input values. Following this, the CFD solver components read in the mesh and initialize their solution data (*initMeshSolution*). We note that the near-body mesh is read from a file, while the off-body mesh is generated internally. For restart cases, both sets of meshes as well as the solution data are read in from the saved restart files (*readMeshSolution*). The next step, *dcfStages1&2*, calls the DCF component (PUNDIT) to perform the donor search and hole/fringe-cutting operations. The stages 1 and 2 refer to near-body/near-body connectivity and near-body/off-body connectivity steps respectively. As noted earlier, these steps are needed only once in the computations and occur prior to commencing the iterative loop.

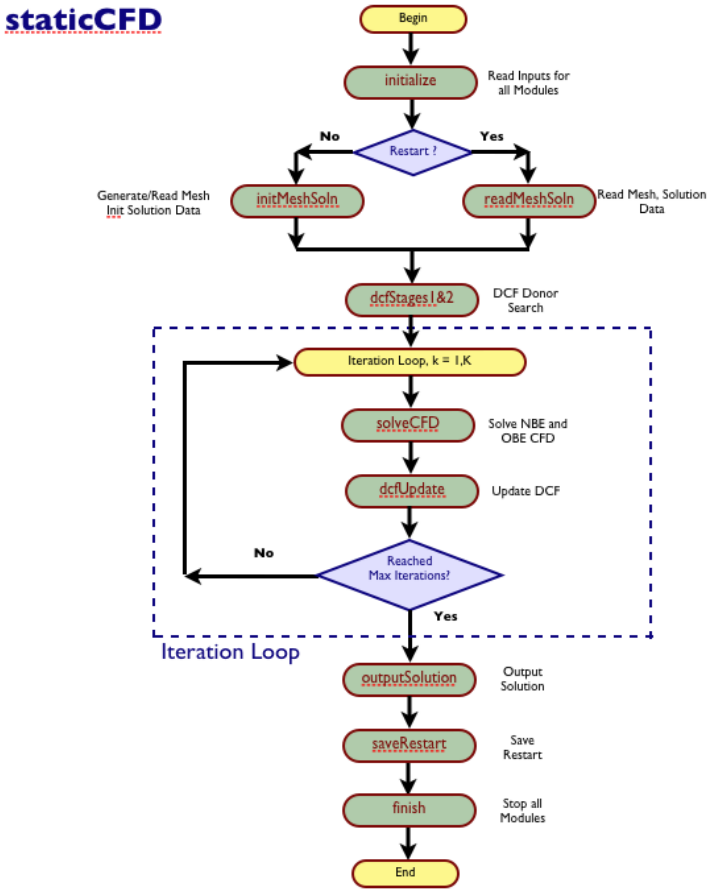

Figure 4. Flow chart showing static-CFD run-mode execution sequence.

The iterative loop starts with the *solveCFD* call which integrates one step of the fluid-dynamic solution. In the dual-mesh case, this involves the sequential solution of the near- and off-body solvers. Following this, the *dcfUpdate* step performs the data interpolation in the "iblanked" fringe points of the near- and off-body meshes. In single-mesh mode, the *solveCFD* step involves only the near-body solver and there is no DCF update step needed. The iterations are continued until the maximum number of iterations specified is reached, at which point, the execution exits the loop. In addition, output solutions for visualizations and restart data are also periodically saved to disk. Once the iterations are completed, solution and restart data are saved and the *finish* step directs all modules to deallocate their data and exit smoothly.

Finally, we note that the above static-CFD run-mode script is simplified for ease of demonstration. For instance, the script shows only the steady-state scenario. Time-accurate computations can also be enabled within this operation mode. Furthermore, actual operation of the script involves checks for consistency of the input variables as well as periodic outputting of solutions and restart data during the iterative process. Finally, we note that the momentum-disk operation is entirely contained within the near-body solver and is transparent to the Python execution.

III.B. Dynamic CFD Analysis

Use-cases involving moving meshes with relative motion between the near-body and off-body meshes utilize the dynamic-CFD run-mode script. Figure 5 shows the execution loop for this run-mode. Helios supports only the dual-mesh mode for moving meshes and therefore the off-body engine and DCF steps are always active in this operation mode. Two kinds of dynamic mesh problems are supported: rigid rotors and flexible rotors with prescribed motion/deformation. In both instances, we note that only the near-body mesh moves and deforms (where applicable) while the off-body mesh remains stationary. The near-body mesh changes

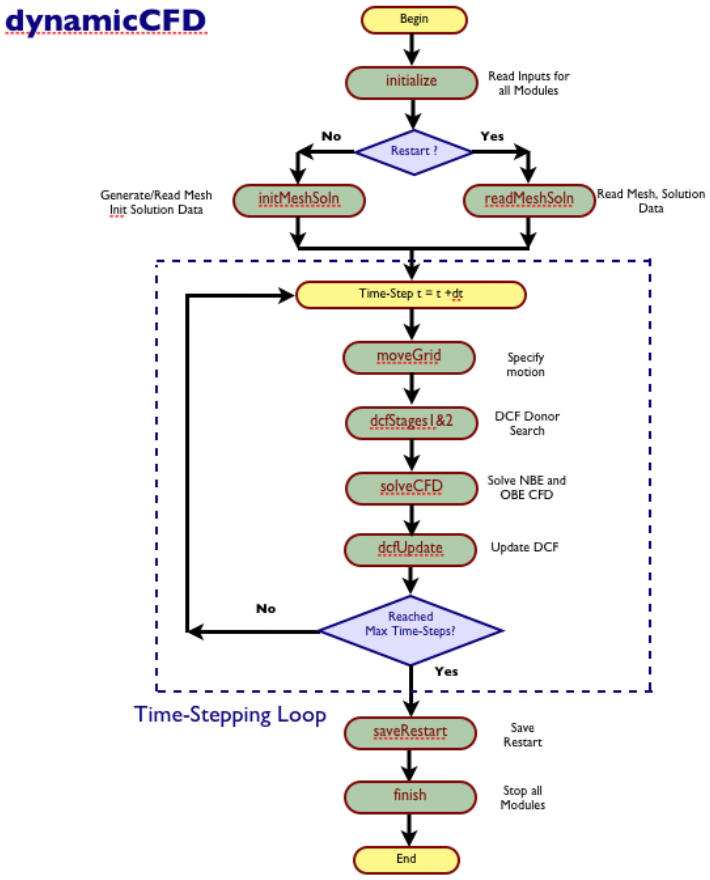

Figure 5. Flow chart showing dynamic-CFD run-mode execution sequence.

are handled by the Mesh Deformation Module called *rdeform*. In addition, moving mesh metrics and the geometric conservation law formulation are handled by the near-body solver (NSU3D). In Helios-Whitney, we support only fixed off-body mesh generation and, as a result, the off-body mesh stays unchanged and is not refined or adapted to conform to the moving geometry.

The dynamic-CFD execution loop resembles the static-CFD execution and in fact utilizes many of the same method calls as building blocks to build the execution sequence. However, it differs from the static run-mode in two crucial ways. Firstly, there is now a *moveGrid* call that occurs within the time-stepping loop. This step is responsible for the motion of near-body rotor mesh to handle rotation as well as near-body mesh deformation when a prescribed motion file is used. The second major distinction is that the *dcfStage1&2* step is now within the time-stepping loop since the donor search and hole identification need to occur every time the near-body mesh moves or deforms. Pundit therefore recalculates the interpolation weights every physical time-step. Following this are the usual *solveCFD* and *dcfUpdate* steps which are unchanged from their static-CFD counterparts. At the end of the time-stepping loop, output and restart data are saved and the modules deallocate their data and exit smoothly. Again, we note that we have not shown all of the steps to keep the description brief and, specifically, solution and restart data can be output at any desired frequency during the time-stepping loop.

III.C. CFD-CA Delta Coupling Analysis

The third run-mode script, shown in Figure 6, is referred to as delta-Coupling and is used for the coupled CFD-CA computations. The delta-coupling or loose-coupling formulation takes advantage of the periodic nature of the rotor dynamics. In this scenario, the aerodynamic forces on the blade are obtained over a period and input into the comprehensive analysis code. In turn, the periodic deflections are used to deform

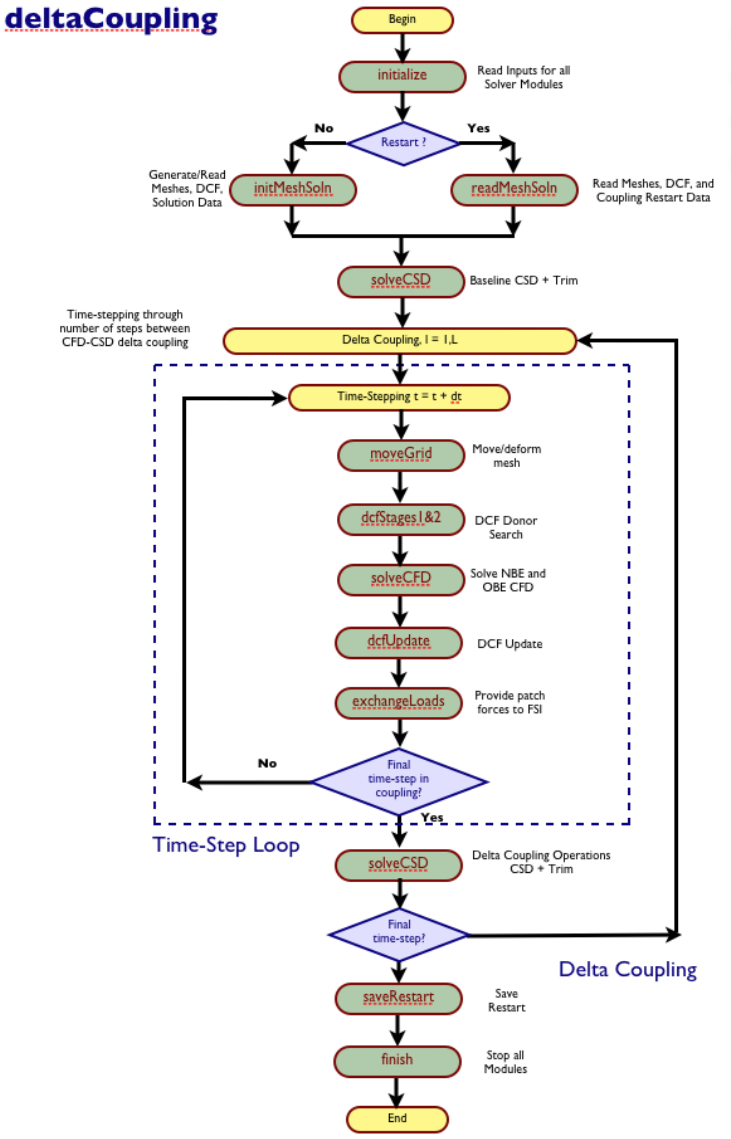

Figure 6. Flow chart showing delta Coupling CFD-CA run-mode execution sequence.

and move the mesh for the aerodynamic simulations over the next period. This sequence of CFD and CA simulations is repeated until the load and deflections converge to a periodic steady-state, thereby ensuring time-accuracy of the response harmonics and, simultaneously, trim. In practice, the coupling is performed every N time steps of the CFD time-marching solution, where N is sufficient to construct airloads over one entire rotor revolution using the solution from all N_b blades. In other words, N must cover a minimum of $2\pi/N_b$ in rotor azimuth for identical blades. Due to transients, usually two or three times this amount is desired so that the constructed solution is periodic. In situations where the blades are dissimilar, N must cover the entire 2π rotor azimuth.

The flow-chart in Fig. 6 utilizes many of the same building blocks as in the previous cases with some important additions: namely, the *solveCSD* and *exchangeLoads* calls and the inclusion of the delta-coupling loop around the inner time-stepping loop. As discussed earlier, the delta-coupling or loose-coupling formulation calculates the structural dynamics and trim for a periodic state of the rotor dynamics. Within the time-stepping loop, the sequence of operations resembles the dynamic-CFD run-mode: the *moveGrid* call moves and deforms the mesh to the new position of the blade surface using the periodic motion file from the comprehensive analysis module; the *dcfStage1&2*, *solveCFD* and *dcfUpdate* steps are the same as before; in

addition, there is now an *exchangeLoads* step wherein the patch forces from the near-body CFD solution are provided to the RFSI module to convert to beam airloads.

Outside of the time-stepping loop are two *solveCSD* calls which are responsible for obtaining the structural dynamics and trim solution from RCAS. The first *solveCSD* call occurs at the outset prior to the delta coupling iterative loop and provides the baseline motions using aerodynamic loads generated internally by RCAS. These baseline (iteration 0) deflections are used to calculate the CFD solution over the first iteration. The second *solveCSD* is called repeatedly within the delta coupling loop, uses the CFD-based airloads data generated by RFSI as the input and calculates the resultant blade motions and deformations. These deflections are used to move the grid over the next period of rotor revolution and so on until the loads and deflections converge.

IV. Results and Validation

Helios-Whitney provides three main capabilities: (1) fuselage aerodynamics with and without momentum-disk, (2) isolated rotor hover analysis in rotational and inertial frames, and (3) isolated rotor with aeroelastic and comprehensive analysis coupling effects. Several validation studies have been performed such as: NACA 0015 wing, DLR-F6 wing-body, Slowed Rotor Compound fuselage, Georgia Tech rotor body, V22 configuration, TRAM and model-scale UH-60A in hover, and full-scale UH-60A in forward flight. In this section, we present results and validation of the Slowed Rotor Compound fuselage, the Georgia Tech rotor body, the TRAM rotor and the full-scale UH-60A.

IV.A. Slowed Rotor Compound Fuselage

The first test case that we consider is the heavy lift, Slowed Rotor Compound configuration fuselage, tested at the NASA Langley 14-×22-Foot Subsonic Tunnel.[43] Additionally, the aerodynamics of the fuselage has been studied using CFD by Allan *et al.* with the OVERFLOW code.[44] The test was conducted with the tunnel ceiling in position and the side-walls removed. A circular post was used to support the fuselage body in the experiment. In our computational study, we do not include the ground, ceiling, side-walls or support. The farfield boundary conditions were maintained at free-stream conditions either through a Dirichlet boundary condition or through characteristics-based Riemann conditions. Test conditions were for a free-stream Mach number of 0.212 over a range of angles of attack.

Helios was executed using both fully-unstructured-mesh as well as dual-mesh modes. The former is equivalent to running stand-alone NSU3D, but the present runs are carried out through the Helios Python infrastructure. Figure 7 shows typical computational meshes used in this study for both cases. The fully unstructured mesh contained about 3-million nodes and was comprised of prisms, tetrahedra and pyramids. The dual-mesh was generated by trimming the same near-body mesh, resulting in about 2.8M nodes. Two off-body meshes were considered: a relatively coarse 5-level off-body grid that contained 3.5-million points and a fine 6-level off-body grid that contained 17-million points. The test runs were typically carried out on 32 or 64 processors.

Convergence results of the fully-unstructured (NSU3D) run are shown in Fig. 8. On the left, the overall residual convergence is shown with the density residual converging steadily for five to six orders of magnitude over about 4000 iterations before leveling out. On the right, the force convergence results are shown also as an L2 norm of the change in the forces from iteration to iteration. Again, very regular and steady convergence is observed over several orders of magnitude, indicating that the flowfield is converging well even in the near-wall boundary layer region. Similar convergence results are shown for the Helios dual-mesh (NSU3D-SAMARC) run in Fig. 9 for the coarse off-body grid.

Results for the fine-off-body mesh are not shown here, but are similar. The initial convergence rates of both the density residual and the force residuals are comparable to the previous case, indicating that the solution development is controlled largely by the near-body solution. Overall convergence subsequently slows down as the convergence of the explicit off-body solver starts to dominate. Nevertheless, the overall convergence of the dual-mesh calculation still results in five to six orders drop in the residual over about 4000 iterations, which is similar to the single-mesh case. It is notable that the near-body density residual again flattens out after the five to six order drop. The reason for this is not yet fully understood, although it is likely that small levels of unsteadiness in the fuselage wake may be a contributing factor. Further studies are necessary to confirm these observations.

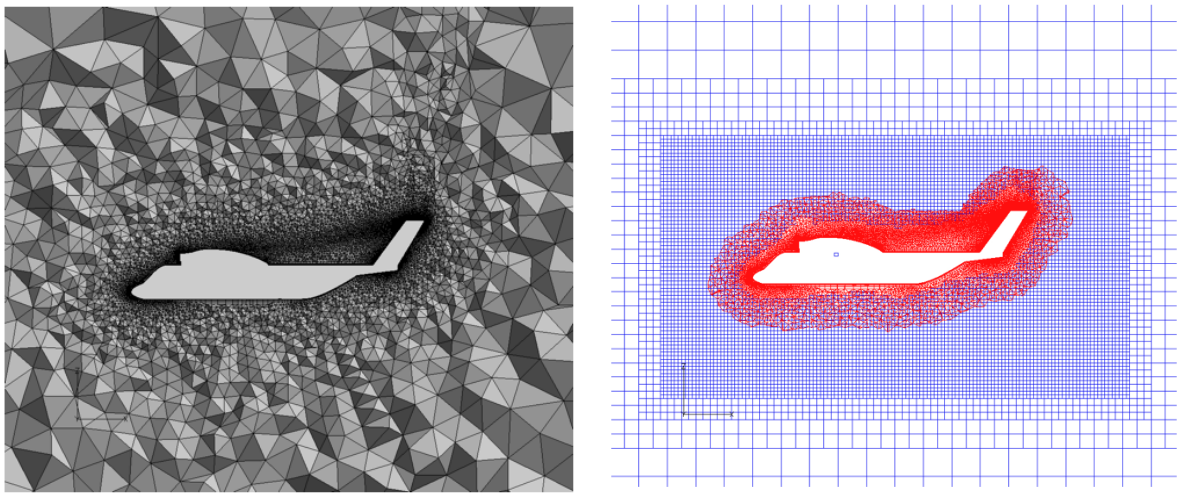

Figure 7. Typical computational meshes used for the slowed rotor compound fuselage: fully unstructrued NSU3D mesh (left) and dual-mesh with unstructured near-body and Cartesian off-body (right).

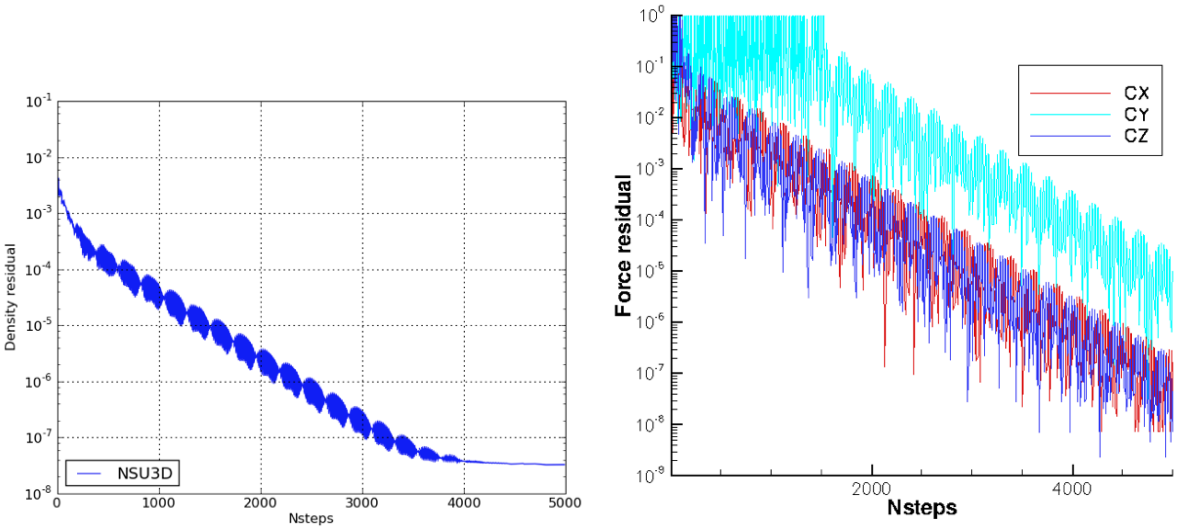

Figure 8. Density residual (left) and force residual (right) convergence as a function of iteration steps for fully-unstructured (NSU3D) computations of the slowed rotor compound fuselage.

Comparisons of the lift and drag coefficients with the experimental data are carried out for a range of angles of attack using the dual-mesh mode with the coarse off-body mesh. These results are summarized in Fig. 10. Generally, the Helios code appears to over-predict the lift coefficient (shown on the left-side of the figure) and is in close agreement with the OVERFLOW results for negative angles of attack. At slightly positive angles of attack, the discrepancy with the experiments (and the OVERFLOW results) grows somewhat and may be attributed to significant flow separation effects. Drag comparisons shown on the right side of the figure show similar trends. There is fairly good agreement observed at negative angles of attack between the Helios prediction and the experimental data, but there is significant discrepancy near zero and slightly positive angles. More detailed OVERFLOW predictions reported by Allan *et al.*[44] included the effects of the support and walls. These results were somewhat inconclusive since they indicated better drag-coefficient agreement, but a greater over-prediction of the lift-coefficient.

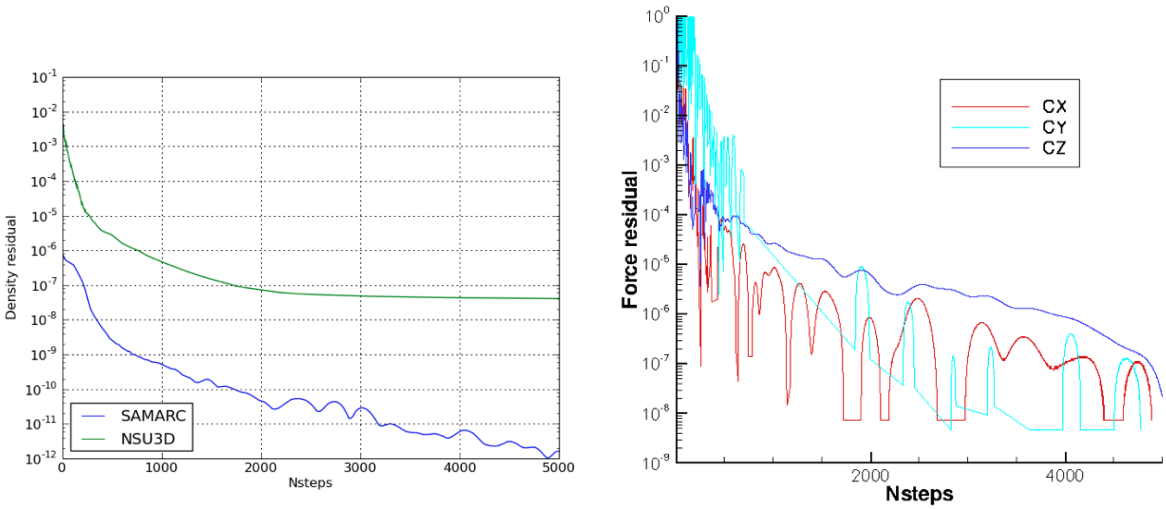

Figure 9. Density residual (left) and force residual (right) convergence as a function of iteration steps for dual-mesh (NSU3D-SAMARC) computations of the slowed rotor compound fuselage.

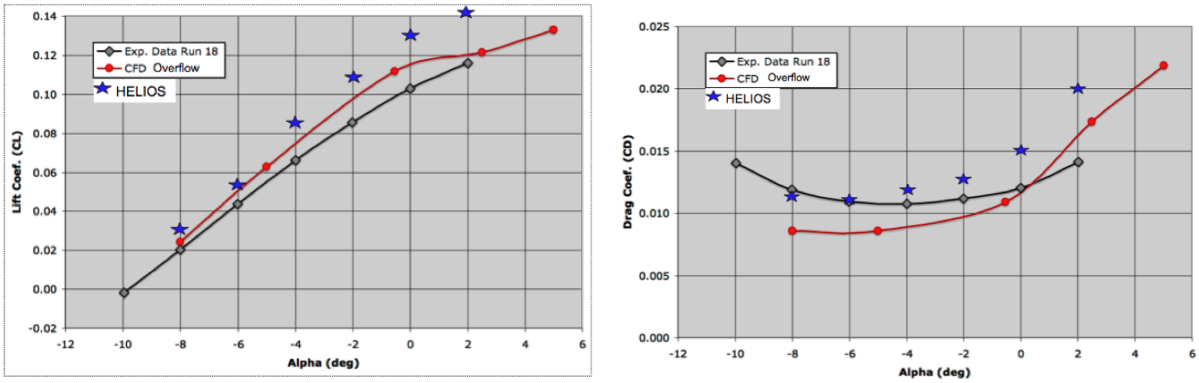

Figure 10. Comparison of Helios dual-mesh predictions of lift and drag coefficients as a function of angle of attack with experimental data and OVERFLOW code predictions. Overflow results from Allan *et al.*[44]

IV.B. Georgia Tech Rotor-Body with Momentum-Disk Model

The Georgia Tech rotor-body is a two-bladed teetering rotor with stiff blades over a hemisphere-cylinder fuselage. It was tested in the John J. Harper 7×9-ft wind tunnel at Georgia Tech to investigate rotor-fuselage interaction on a simple configuration[45,46] and has been widely analyzed over the years.[47,48] Static pressure taps and condenser microphones are installed in a row along the length of the fuselage, which allowed mean and unsteady surfaces pressures to be measured anywhere on the body by simply rotating the fuselage about its centerline. Specific conditions calculated are listed in Table 2 and correspond to the experimental test conditions. To minimize hub interference this model was not equipped with collective or cyclic controls, but instead was given a fixed collective pitch of 10 degrees and allowed to freely flap about the teetering hinge.

The case was exercised in Helios using the single (unstructured) mesh mode and the grid used is shown in Fig. 11. In addition to the mixed-element (prisms and tets) mesh shown, a fully tetrahedral mesh is also used in the Helios study. Both meshes had about 1.9M nodes. The momentum-disk case was set-up using the SMEMRD module. We note that the disk surface is not directly inserted into the unstructured mesh since SMEMRD generates its own structured mesh and distributes the momentum (and energy) source terms to the unstructured mesh. Comparisons are also made with FUN3D code[32,49] on the fully tetrahedral grid. We note that the FUN3D results were also obtained by coupling the code with the same SMEMRD module.[38] In Helios, the low speed case was run using low Mach preconditioning (LMP) due to the very low freestream Mach number. In contrast, the FUN3D results employed the artificial compressibility formulation.

Table 2. CFD Test Conditions for Georgia Tech Rotor Body Configuration

Advance Ratio	0.1
Free-Stream Mach Number	0.0295
C_T	≈ 0.009 (fixed controls)
Fixed collective (pitch) angle	10.0 deg
Flapping (β_{1s}, β_{1c})	-2.02, -1.94 deg
Reynolds Number	315,000
Shaft Tilt	6 deg forward
Rotor RPM	2100

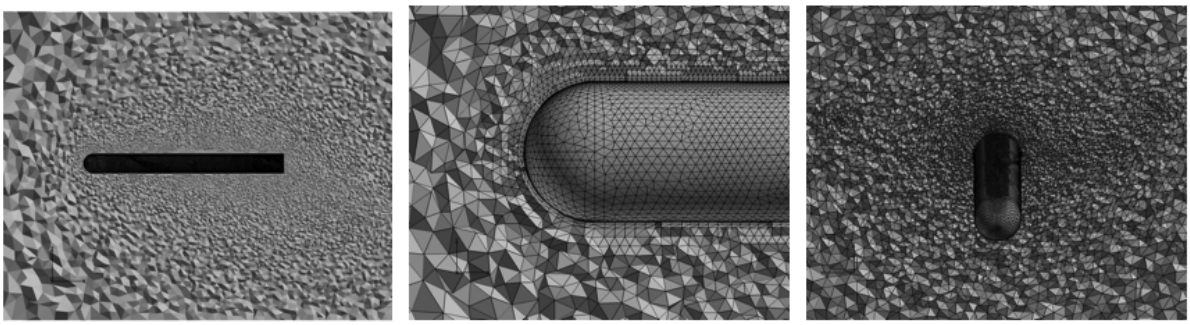

Figure 11. Computational mesh used for the Georgia Tech rotor-body configuration. Helios is operated in fully unstructured mode for momentum-disk computations.

Helios convergence results are shown in Figure 12 for the density residual, and fuselage lift and drag coefficients. It is seen that converged results are obtained after about 1000 iterations. The predicted

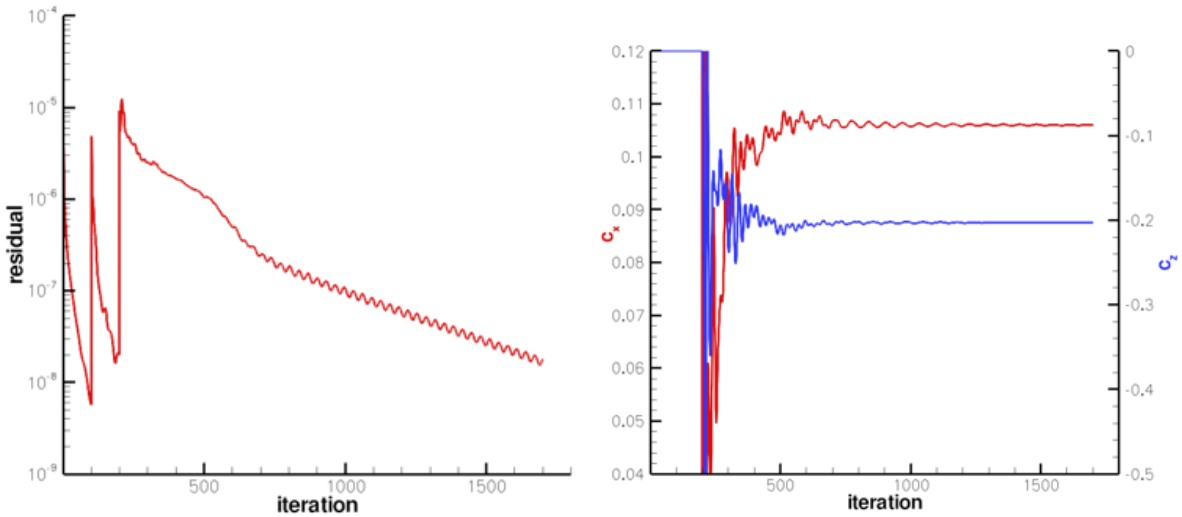

Figure 12. Residual (left) and force (right) convergence as a function of iteration steps for fully-unstructured (NSU3D) computations of the Georgia Tech rotor body configuration with momentum-disk model.

thrust comparisons using Helios and FUN3D are shown in Table 3. The NSU3D results used both central-differencing and Roe upwind schemes, while FUN3D used the Roe upwind scheme. Moreover, since FUN3D does not have provision for low Mach preconditioning, computations were carried out with the standard compressible equations as well as with the incompressible artificial compressibility formulation. Finally, NSU3D was run on both mixed-element and tetrahedral meshes, while FUN3D was exercised only on the

tetrahedral mesh. The measured experimental thrust is 0.009, making the FUN3D result somewhat more accurate. Compared with FUN3D at the fixed 10 deg collective, the thrust predicted by Helios is about 7% lower. Therefore, for direct comparison, NSU3D has been trimmed to the FUN3D thrust, resulting in a 0.75 deg increase in collective to 10.75 deg. It is noted that at the same thrust, the FUN3D torque is up to 12% lower. We further note that the mixed element and tetrahedral grids in NSU3D result in the same fuselage and rotor forces.

Table 3. Comparisons of CFD Predictions for Georgia Tech Rotor Body Configuration

Solver	Scheme	Mesh	Collective	ct	cq
Experiment	—	—	10.0	0.009	—
NSU3D	CD-LMP	Mixed	10.0	0.00854	0.000733
NSU3D	Roe-LMP	Mixed	10.0	0.00869	0.000729
NSU3D	CD-LMP	Mixed	10.75	0.00921	0.000831
NSU3D	Roe-LMP	Tet	10.75	0.00921	0.000831
FUN3D	Roe-Comp	Tet	10.0	0.00991	0.000735
FUN3D	Roe-Incomp	Tet	10.0	0.00921	0.000728

CD : Central-Differencing, Roe: Roe upwind-flux differencing

LMP: Low Mach Preconditioning, Comp: compressible gas model, Incomp: incompressible model

Figure 13 shows surface pressure predictions on the upper and lower centerline (left) and on the $z = 0$ waterline. In addition to the experimental data, Helios results using NSU3D and SMEMRD are shown for both the mixed-element and tetrahedral meshes. Also, FUN3D results are shown for the tetrahedral mesh. Interestingly, both codes miss the forward peak in the upper centerline prediction (left figure), which appears to be due to an inherently unsteady effect. Also, looking at the results on the waterline (right), it is apparent that the tetrahedral-mesh results of the two codes agree quite well, while the mixed-element Helios result shows some differences on one side of the fuselage. To examine the differences between the results

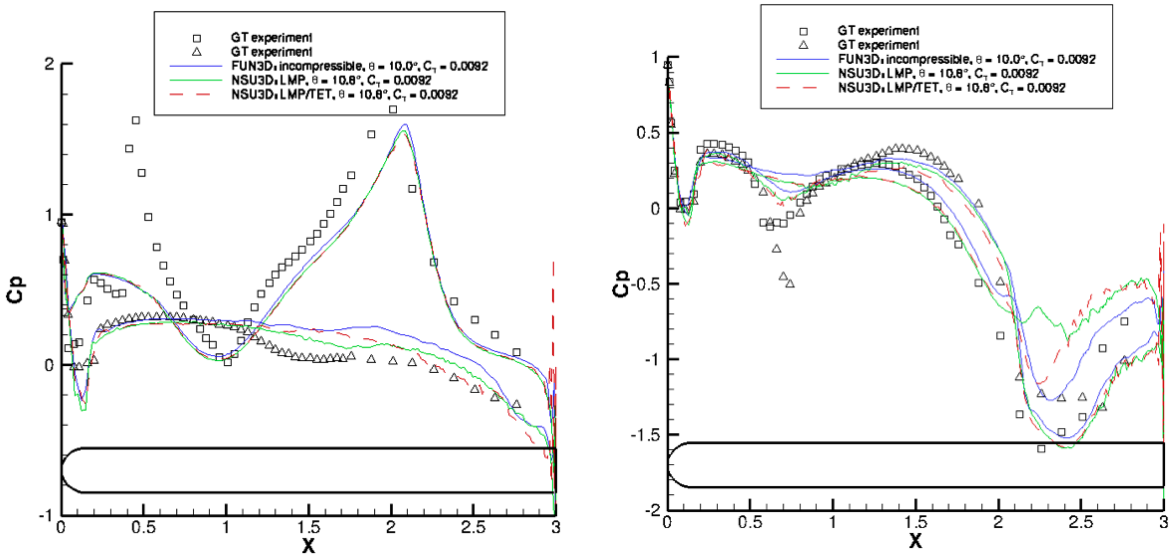

Figure 13. Comparison of Helios predictions for tet and mixed-element meshes with experimental data and FUN3D using tet-meshes. Upper and lower centerline (left); z = 0 waterline (right); Mach = 0.0295.

more closely, Figure 14 shows the pressure contours on the body for the three cases: mixed-element-Helios, tet-mesh-Helios and tet-mesh-FUN3D. In the case of the mixed element mesh, which shows the discrepancy in the pressure comparisons, the low pressure region on the side of the fuselage appears to be broken into two pieces, with the z = 0 waterline cut intersecting the high pressure ridge. Indeed, the tetrahedral mesh is in somewhat better agreement on the side of the fuselage. Although these studies are not exhaustive, there is evidence to conclude that the Helios and FUN3D results are in reasonable agreement when the same tetrahedral mesh is used in both codes and the collective is adjusted to account for the thrust discrepancy.

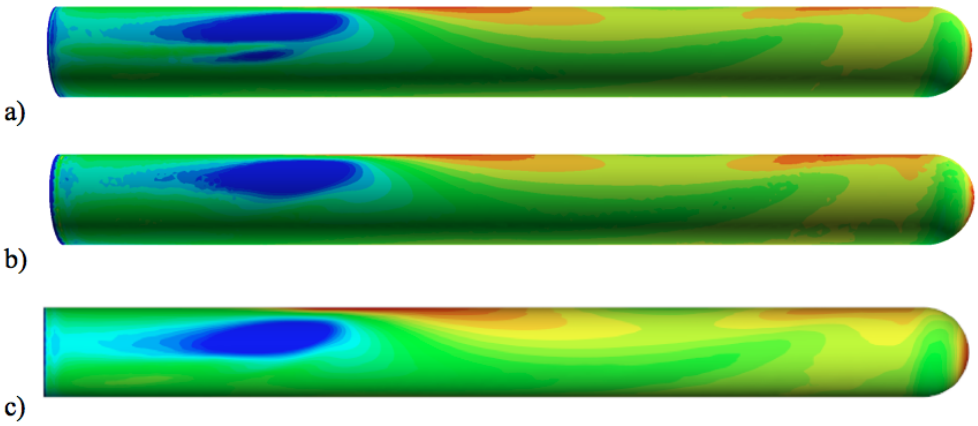

Figure 14. Pressure contours on fuselage: a) Helios (mixed-element), b) Helios (tet), c) FUN3D (tet), M = 0.0295

IV.C. Ideal Hover for Isolated TRAM Rotor

The next case describes the application of Helios to the experimental TRAM rotor. The Tilt Rotor Aeroacoustics Model (TRAM) is a quarter-scale model of the Bell/Boeing V-22 Osprey tiltrotor aircraft. Isolated rotor experimental measurements were taken from the 20-×26-ft open jet section of the Duits-Nederlandse Wind Tunnel (DNW). A full-span dual rotor model was tested in the 40-×80-ft test section of the National Full-scale Aerodynamic Complex (NFAC) at NASA Ames[50] and previously modeled by many researchers.[51] The TRAM rotor has been modeled within Helios in three different ways: (1) in the rotational frame using a single unstructured mesh over the whole domain, (2) in the rotational frame using the dual-mesh mode with unstructured near-body and Cartesian off-body, and (3) in the inertial frame using the dual-mesh mode. We restrict our attention in the present document to the single- and dual-mesh rotating frame studies.

Figure 15 shows a single-block fully unstructured mesh with about 5M nodes. The grid is set up for the fixed 14-deg collective. The same mesh was also subset by trimming the mesh to a specified distance from the surface for use as the near-body mesh in the dual-mesh mode. Figure 16 shows the trimmed near-body

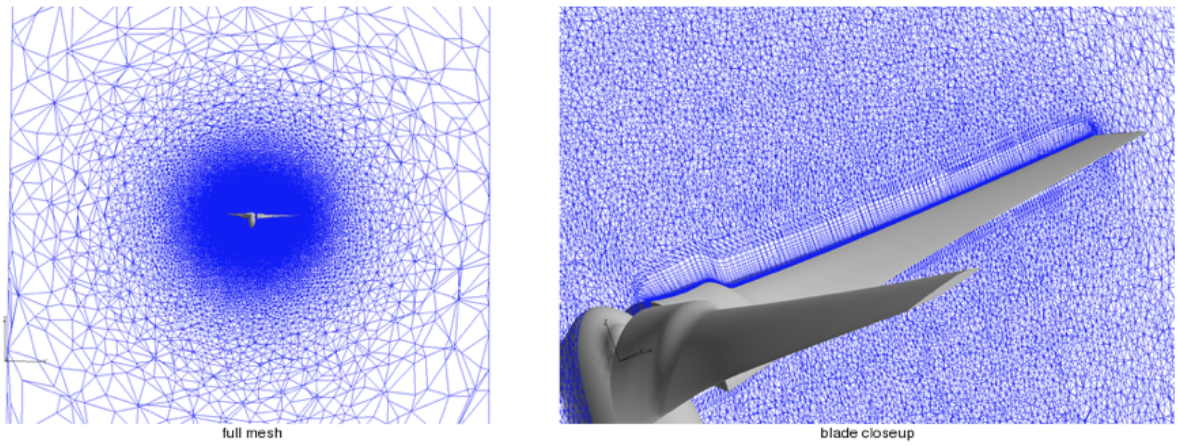

Figure 15. Single-block unstructured computational mesh used for the TRAM configuration.

mesh embedded within the off-body Cartesian mesh, which is automatically generated by the SAMARC component of Helios during execution. In the latter case, the near-body mesh contains about 2.9 million nodes while the off-body mesh is obtained using 6 levels of mesh refinement, contains about 36 million nodes and has a fine mesh resolution of about 10% of the tip chord. The mesh point data as well as the CPU execution times per iteration step are summarized in Table 4.

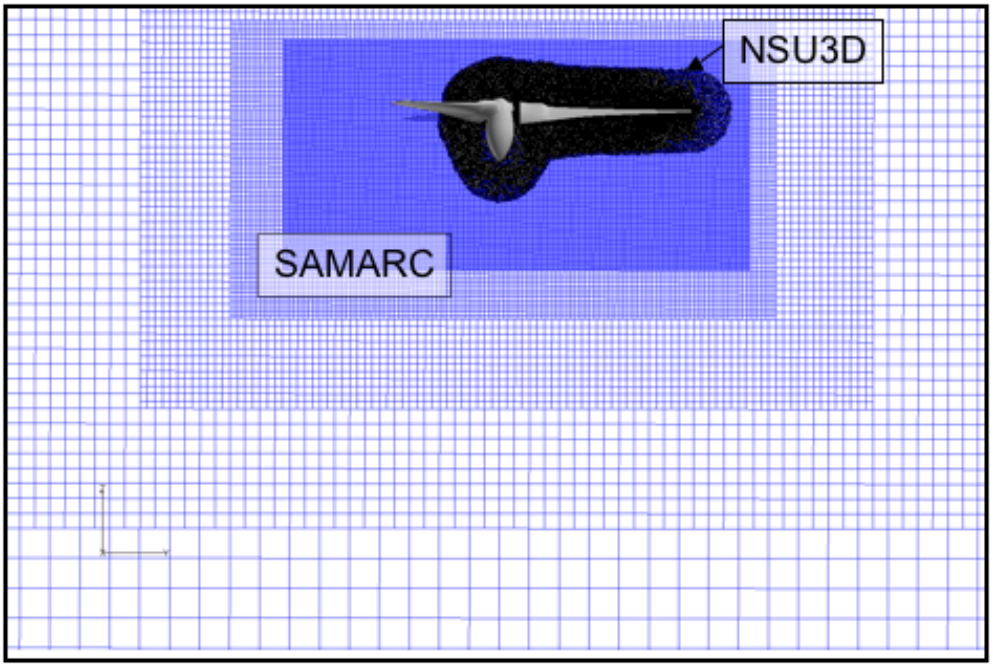

Figure 16. Dual-mesh with unstructured near-body and Cartesian off-body meshes used for TRAM configuration. Near-body mesh is cut from full single-block unstructured mesh.

Table 4. TRAM Computational Performance on 64 processors

Solver	Unstructured	Cartesian	Total
Single-Mesh			
Mesh Size	5M	—	5M
CPU Time	3.3s	—	3.3s
Dual-Mesh			
Mesh Size	2.9M	36M	38.9M
CPU Time	1.6s	3.3s	4.9s

The computed flowfields from the fully unstructured (NSU3D) and dual-mesh (NSU3D-SAMARC) runs are shown in Fig. 17. Iso-surface contours of the q-criterion are shown for $q = 0.00005$ in both cases. In the fully unstructured case, it is apparent that the wake resolution is generally poor, with the vortex core being preserved only about a quarter of a revolution, although weaker cores are observable for about 3 passes below the blade. In comparison, the wake resolution is clearly better in Helios, due in part to the finer mesh spacing and in part to the fifth-order spatial accuracy of the Cartesian off-body. In fact, the dual-mesh results show that the vortex structure is maintained intact for as many as 5 revs. Moreover, finally when the vorticity dissipates, it appears to do so as a result of going into the coarser grid system, and not because of dissipation losses on the fine grid.

The computed thrust and power coefficients from the two sets of calculations are given along with the experimental data in Table 6. Results from the OVERFLOW code are also included for comparison purposes. It is clear that the dual-mesh Helios code achieves comparable results to Overflow and are generally better than the results obtained by the fully unstructured mesh (NSU3D). Overall, Helios over-predicts C_T by about 2%, C_Q by about 6% and under-predicts the Figure of Merit by about 3%.

It is also worth noting the CPU execution times of the single-mesh (or fully unstructured) and dual-mesh Helios computations listed in Table 4. The single-mesh system with 5 million nodes takes a total CPU time of 3.3s/iteration, while the dual-mesh system with 39 million points takes only 4.9s or approximately 50% more. Further examination of the total number of unstructured and Cartesian points in the dual-mesh system indicates that the Cartesian mesh system has 36 million off-body points compared with the 2 million

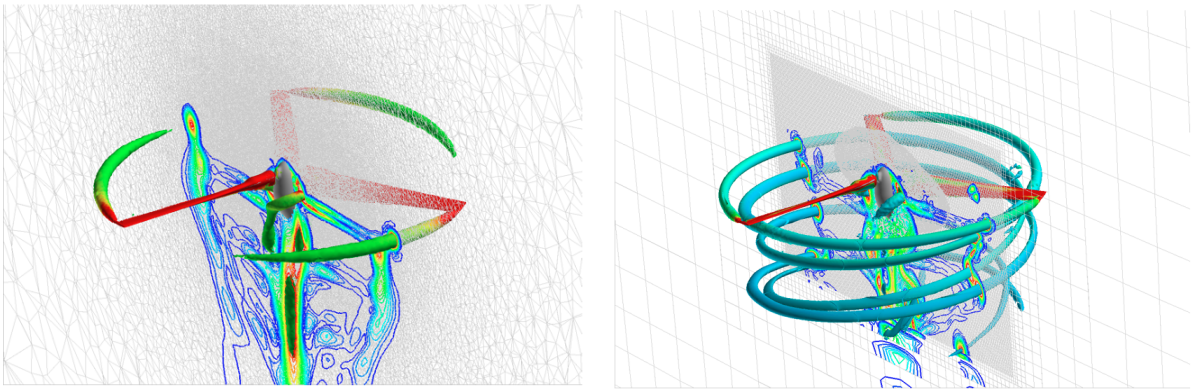

Figure 17. Comparison of iso-surface of the q criterion for fully unstructured (NSU3D: top) and unstructured-Cartesian (NSU3D-SAMARC: bottom) meshes. NSU3D is second-order accurate, while SAMARC off-body solution is fifth-order accurate.

Table 5. TRAM Load Predictions

Solver	C_T	C_Q	FM
Experiment	0.0149	0.00165	0.7794
NSU3D	0.01505	0.001834	0.7120
Helios	0.0152	0.00174	0.7557
Overflow	0.0148	0.00168	0.7578

off-body tetrahedra in the fully unstructured mesh (obtained by subtracting the 3 million near-body mesh from the 5 million fully unstructured mesh). Thus, the Cartesian solver is able to compute the 36-million point mesh at about twice the CPU time as the unstructured solver requires for 2-million points. In other words, the Cartesian solver is almost an order of magnitude faster in computational speed per grid point than the unstructured solver. We note that these timings are somewhat approximate because of differences in the computational requirements for tetrahedra versus prisms as well as dependence on the number of processors used for the estimation. Nevertheless, the results confirm the main impetus for the dual-mesh approach in Helios, i.e., Cartesian off-body meshes provide solutions at much lower expense than unstructured meshes of similar resolution. Moreover, the use of Cartesian meshes permits the use of high-order discretizations at minimal additional expense—in fact, the present results for the TRAM were obtained using fifth-order accuracy in the off-body domain, while the unstructured solution is limited to second-order accuracy.

Scalability performance of the dual-mesh Helios results are shown for the 39-million point case in Fig. 18. The results are obtained by running the problem of fixed size on 8, 16, 32, 64 and 128 processors on the Hawk cluster at AFRL. Specifically, Fig. 18a shows the parallel speed-up as a function of number of processors. It is evident that good scalability of over 90% is obtained for 64 processors, with the scalability dropping slightly to 81% for the 128-processor run. It should be pointed out that this performance is very good considering that we used a fixed problem size in these studies. Moreover, the Helios-Whitney code does not have any special load-balancing schemes and both the near-body and off-body solutions are carried out on the same number of processors. Consequently, it is difficult to maintain peak scalability performance simultaneously for both near- and off-body solvers. The break-up of the computational time spent in the three key modules: near-body CFD (NSU3D), off-body CFD (SAMARC) and domain connectivity (PUNDIT) are shown in FIg. 18b. For the current grid size, it is observed that NSU3D scales close to the ideal parallel performance, while SAMARC performs close to ideal for smaller numbers of processors, but deteriorates in performance for the 128-processor run. For this static mesh case, we further observe that the portion of CPU time spent on domain connectivity is relatively low. These numbers also scale fairly well for up to 64 processors with modest slow-down being observed for the 128-processor run.

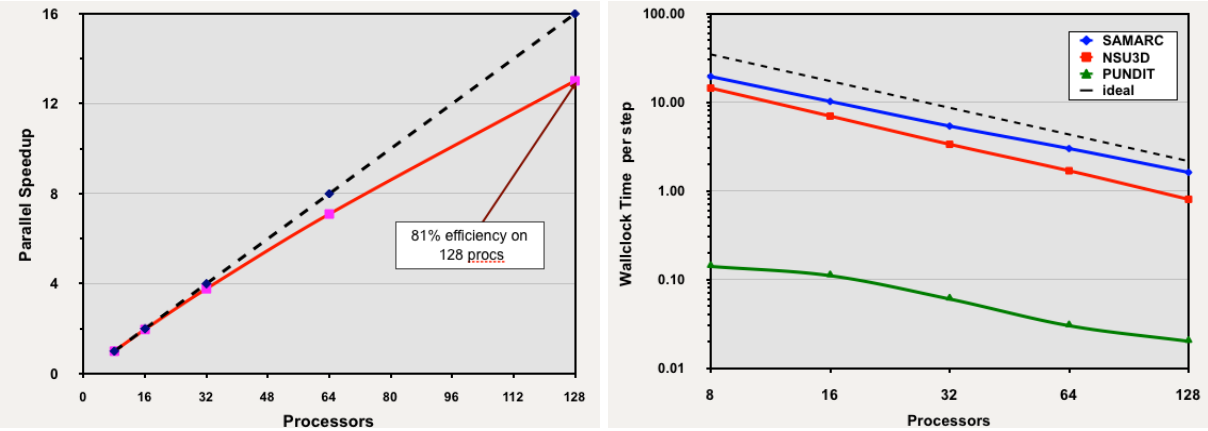

Figure 18. Scalability performance of the dual-mesh Helios code up to 128 processors for a fixed problem size. Scalability results were obtained on the Hawk system at AFRL.

IV.D. Isolated UH-60A Rotor in Forward Flight with Aeroelasticity and Trim Coupling

As a final validation study, we focus on the CFD-CA coupling capability in Helios. The test case is an isolated UH-60A rotor in three critical forward flight test cases taken from the U. S. Army/NASA UH-60A BLACKHAWK Airloads Program Flight Tests.[52] This section documents the test cases, the flight conditions in each, Helios computational specifics, validation of important measures of accuracy, and accuracy metrics for each measure. Note that, in the current study, the measures are limited at present to performance and blade airloads, and does not include structural loads, hub vibratory loads, wake, or acoustics.

The test cases are the Flight Counters 8534 (high speed), 8513 (low speed), and 9017 (moderate speed, highly loaded). These cases are chosen because they represent important design conditions, and in addition, cover all of the three important aerodynamic mechanisms—3D transonic unsteady pitching moments, vortex induced airloads, and multiple dynamic stall cycles—separately. The test cases are shown in Fig. 19 and Table 6. The three cases represent important design calculations. The high speed case, C8534, is the

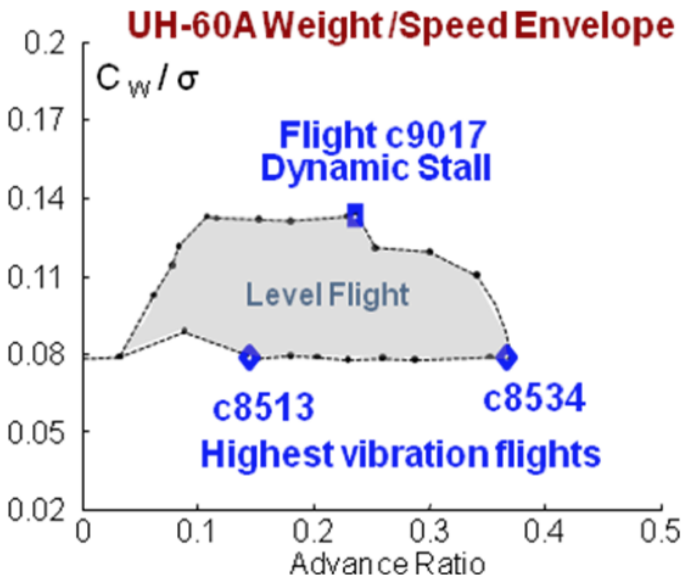

Figure 19. UH60 weight/speed envelope.

highest vibration condition for the UH-60A rotor. It is also the highest level flight speed for this rotor. The low speed case, C8513, is representative of a high vibration low speed transition (hover to forward flight)

Table 6. Flight Test Conditions

Flight Counter	C8534 (high speed)	C8513 (low speed)	C9017 (dyn stall)
Advance Ratio (Speed)	0.368 (155 kts)	0.1535 (65 kts)	0.234 (100 kts)
C_T	0.084	0.0076	0.126
Shaft Angle of attack	-7.31 deg	0.76 deg	-0.19 deg
Main airload mechanism	Unsteady transonics	Wake-induced airloads	Dyn. stall cycles

condition. By itself, this flight is not a high vibration case, but it has an airload mechanism similar to one. The dynamic stall case, C9017, is representative of stall loads in maneuver. It is a highly loaded condition (trimmed at high altitude) occurring near the thrust boundary. Finally, we note that the three cases are well understood, and CFD/CA coupled results well-benchmarked with structured near-body CFD solvers for both airloads and structural loads.[18,20,22] The present Helios studies utilize an unstructured near-body mesh with about 5M nodes and a Cartesian off-body mesh with about 10M nodes. The latter corresponds to a fine-mesh spacing of about 18% of the tip-chord. An azimuthal time-step of quarter degree was used in all cases. The calculations were performed on 128 processors on the Hawk (AFRL) and mjm (ARL) clusters.

All validation cases are observed to converge smoothly. For instance, the azimuthal variation of normal forces at two key radial stations are shown in Fig. 20. The normal force at 77.5% R shows the effect of inboard wake interaction at the junction of first and second quadrants. The normal force at 96.5% R, a station on the swept portion of the blade, shows the large negative lift due to high elastic twist deformation. The figures show the gradual convergence of the normal forces from one coupling iteration to the next. By iteration 4, the nominal waveform of the oscillatory harmonics are observed to be in place.

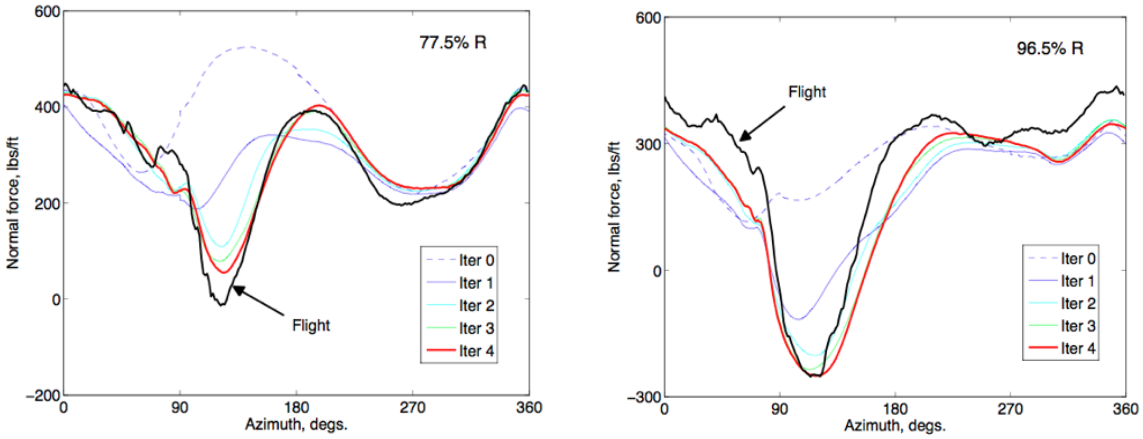

Figure 20. Normal force convergence at (a) an inboard station 77.5% R and (b) on the swept tip portion of blade 96.5% R.

In general, the sectional normal forces are satisfactorily predicted on all the nine stations of the blade as seen in Fig. 21. These validations are almost identical to the state-of-the-art structured code predictions (see for example Ref.[17,18,22]). The vibratory airloads are shown in Fig. 22. These are the 3/rev and higher harmonics, the dominant of which are the 3, 4, and 5/rev harmonics, that generate the shaft transmitted vibration in the fixed frame. The phases of these vibratory harmonics are satisfactorily predicted. However, the peak-to-peak magnitude of these harmonics are not satisfactory—with about 30% error. The quarter-chord pitching moment predictions are shown in Fig. 23. Predictions at all radial stations are in agreement with existing state of the art. Note that the measured values have recognized discrepancies in mean magnitudes due to gauge-out near the trailing edge. The critical pitching moments occur near the tip (86.5% R outboard) and are the key contributors to the large elastic twist deformation that drive the vibratory normal forces shown earlier. The pitching moments at the tip are driven by 3-D unsteady relief on the transonic shocks near the tip. These shocks first appear on the lower side and then move to the upper and determine the high nose down pitching moments at the junction of the first and second quadrants.

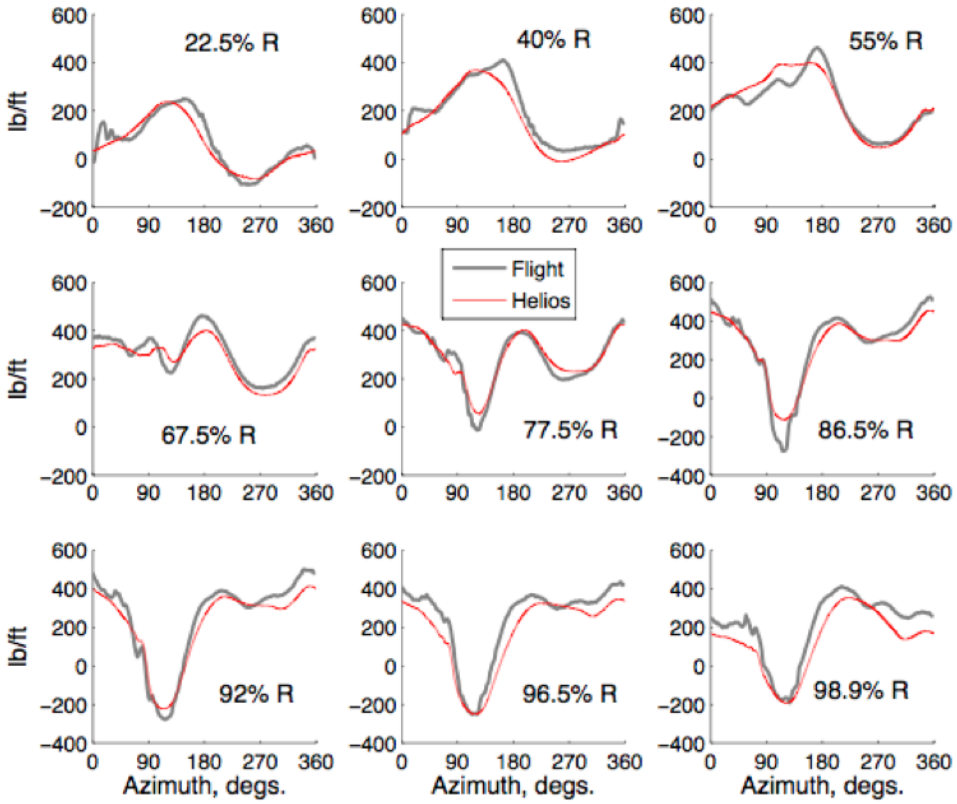

Figure 21. Calculated sectional normal forces compared with flight test measurements at nine radial stations; all harmonics.

Predicted airloads for the low speed flight C8513 are validated with flight test data in Fig. 24. The wake airloading on the retreating side is not as well-predicted in extent as that on the advancing side but otherwise predictions are satisfactory. The error in the vibratory harmonics are shown in Fig. 25 and again good agreement is observed.

For the dynamic stall case, the target thrust was arbitrarily reduced by 5% from the previously published value due to well-known uncertainties associated with this flight. Unlike the other two flights, this flight occurs at the stall boundary of the rotor, and achieving the correct trim state is a critical pre-requisite to any meaningful stall prediction. The two stall cycles are well-predicted (Fig. 26, 77.5% R and 92% R). However, the pre-stall lift trough is not consistent with previously published structured code results. Furthermore, the advancing blade lift waveform, shows a spurious higher harmonic waveform in the second quadrant. Reasons for these discrepancies are under investigation, but overall the validation trends are in good agreement with the existing state-of-the-art.

V. Summary

Helios is an innovative multi-disciplinary platform for rotorcraft aeromechanics. It is comprised of a flexible Python-based infrastructure that couples together modules responsible for the aerodynamics, structural dynamics, vehicle flight dynamics and controls. The Python infrastructure treats each module as an object in the main script and orchestrates the execution sequence of the solution algorithm with minimal storage or performance overheads. At the heart of the Helios platform is an innovative mesh framework for the aerodynamics, that involves using mixed-element unstructured meshes in the near-body region and Cartesian meshes in the off-body region. This combination allows for accurate viscous solutions around complex geometries and efficient and high-order accurate solutions in the off-body region. The latter aspect is of particular importance for rotorcraft simulations, where there is a need to accurately resolve the rotor

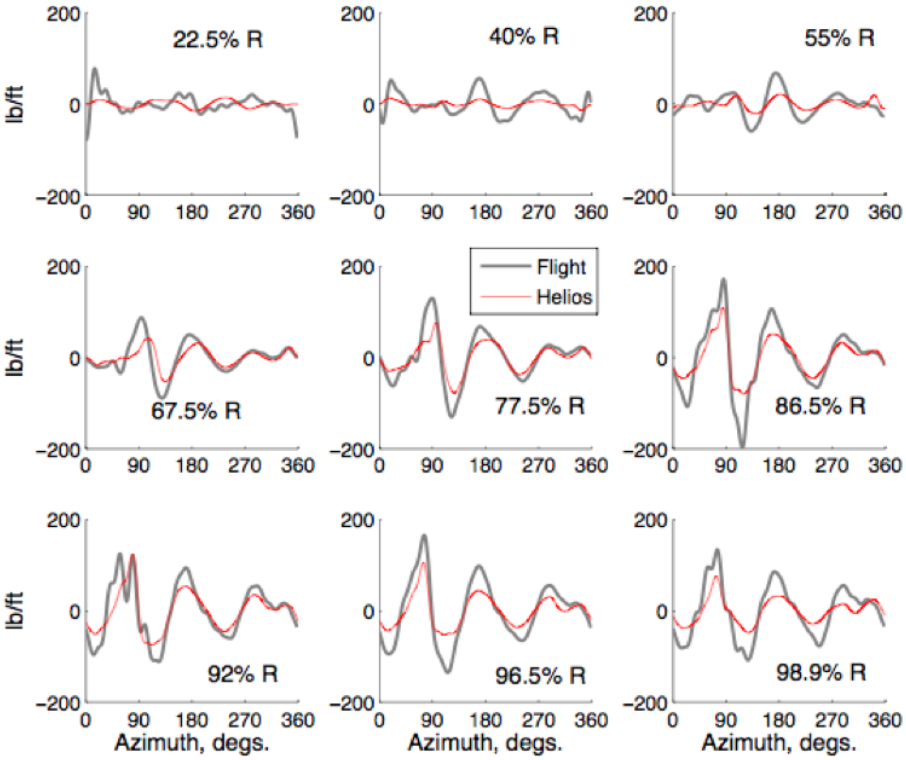

Figure 22. Calculated vibratory normal forces compared with flight test measurements at nine radial stations; harmonics 3/rev and above.

tip vortices for long distances in the far-wake.

The first version of Helios (Whitney) software platform provides three main capabilities: (1) fuselage aerodynamics with and without a momentum-disk model for rotor effects, (2) isolated rotor in ideal hover (i.e., rigid rotor) in both rotational (steady) and inertial (unsteady) frames, and (3) isolated rotor in forward flight with aeroelastic and trim coupling. Validation studies of these capabilities have been presented. These include the slowed rotor compound fuselage tests, Georgia Tech rotor body (for momentum-disk validation), TRAM rotor in hover and UH-60A in forward flight. In each case, Helios results are compared against experimental data as well as results from proven codes such as OVERFLOW and FUN3D (when available). In general, the results reveal that Helios provides reliable performance and the validation studies compare well with the current state-of-the-art.

VI. Future Plans

Helios development will follow an annual cycle of development, testing and release. Accordingly, the second version of Helios, called "Shasta", is slated for development in 2010 and will be beta-released in the beginning of FY 2011 following Product Acceptance Tests by the CREATE-AV ShadowOps team. The Helios-Shasta version will include the following new capabilities: (1) rotor-fuselage combination with free-flight trim coupling, (2) tight-coupling of CFD and CSD for maneuvering flight,[53] (3) modeling of flapped and slatted rotors such as the SMART rotor, and (4) helicopter modeling with store separation with prescribed motion of the stores. In addition to these basic capability enhancements, a number of functional enhancements are envisaged that include: (1) automated off-body adaptive mesh refinement (AMR) using feature-detection methodology,[54] (2) development of a load-balancing module to facilitate dynamic repartitioning of the near-body and off-body problems to accommodate for the growth in off-body mesh points due to the AMR process, (3) enhanced interface specification strategy for all components to facilitate interoperability of components amongst different CREATE-AV products,[31] and (4) generalized interfaces for structural dynamics and trim to allow plugging in of appropriate comprehensive analysis packages. It is

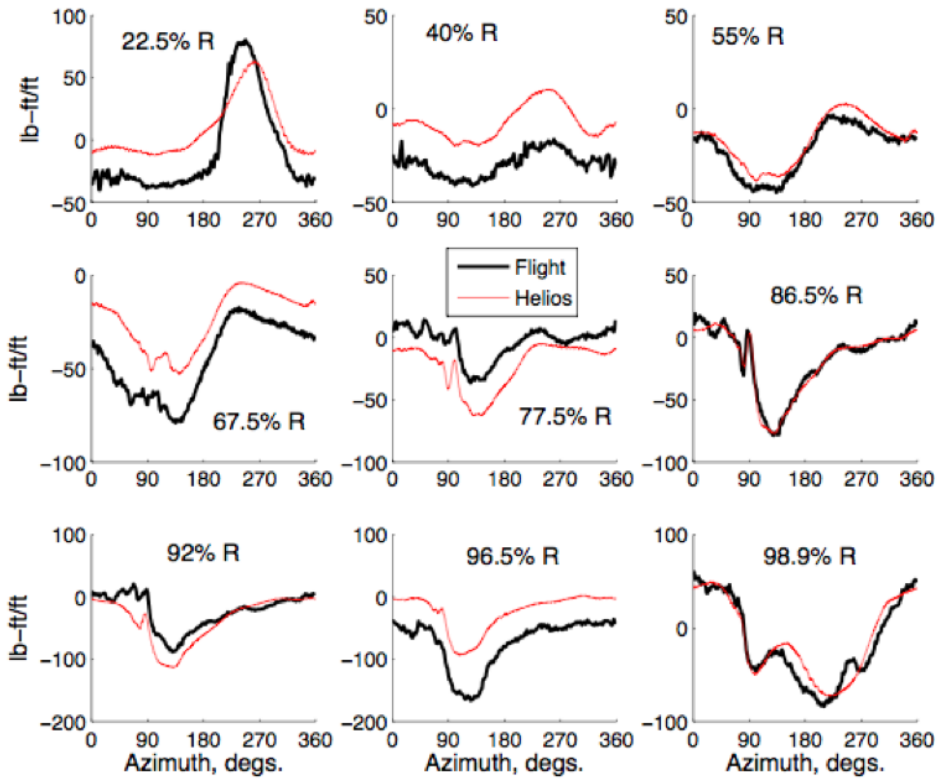

Figure 23. Calculated quarter chord pitching moments compared with flight test measurements at nine radial stations; all harmonics

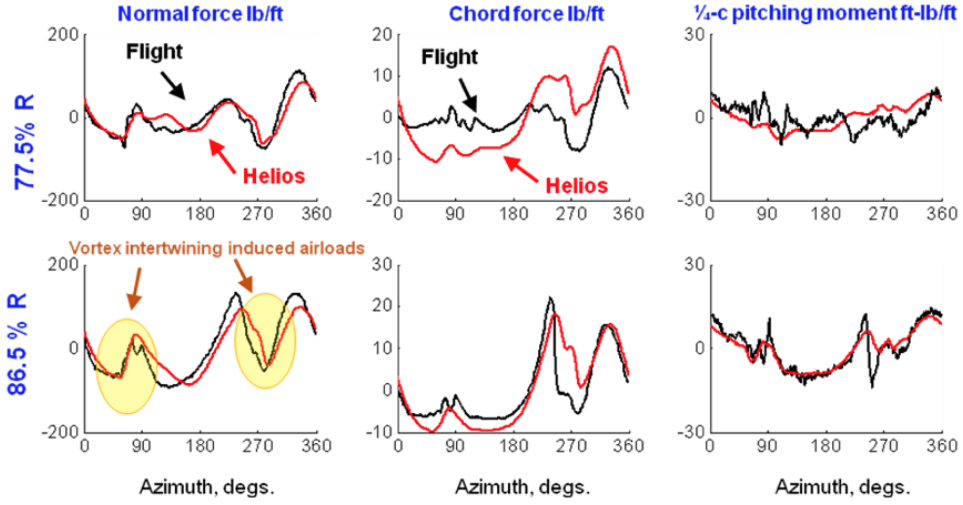

Figure 24. Calculated and measured airloads (steady removed) in low speed fight C8513.

anticipated that these enhancements would lead to improved performance predictions, better resolution of the tip vortices in the far-wake as well as enhanced scalability. Moreover, the generalized interfaces would enable component inter-operability amongst CREATE-AV products as well as facilitate collaboration with the broader research community.

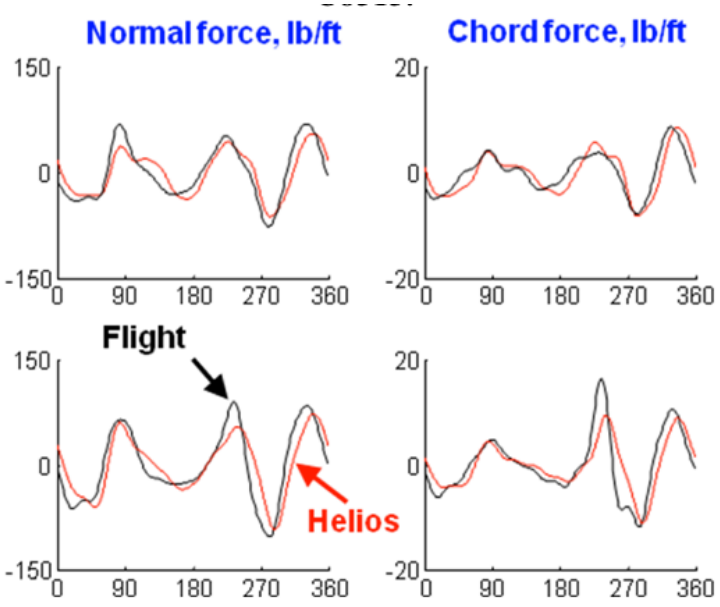

Figure 25. Calculated and measured vibratory (3-12/rev) normal and chord forces for low speed flight C8513.

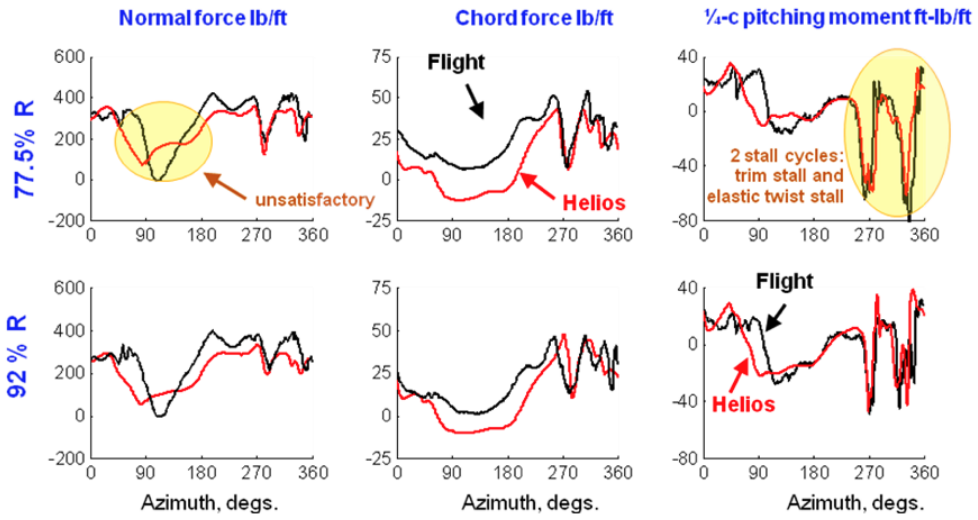

Figure 26. Calculated and measured airloads (steady removed) in dynamic stall fight C9017.

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. Development was performed at the HPC Institute for Advanced Rotorcraft Modeling and Simulation (HI-ARMS) located at the US Army Aeroflightdynamics Directorate at Moffett Field, CA. The authors would like to acknowledge Dr. Robert Meakin and Dr. Chris Atwood of CREATE-AV for their support and encouragement. We are grateful to Dr. Thomas Pulliam of NASA Ames Research Center for the development of the higher-order Cartesian-mesh solver (ARC3DC), Dr. Scott Morton and Mr. Todd Tuckey of the CREATE-AV Kestrel team for the development of the GUI engine, Dr. Sameer Shende of Paratools, Inc. for the development of the PTOOLS run-time environment, Dr. Robert Ormiston of AFDD for his support of the RCAS integration effort and Dr. Wayne Johnson of

NASA Ames for his advice and encouragement. We are also grateful to Dr. Yikloon Lee of NAVAIR for his assistance during the ShadowOps testing phase of the Helios code.

References

[1]Post, D.E., "A new DoD initiative: the Computational Research and Engineering Acquisition Tools and Environments (CREATE) program," *Journal of Physics*, Conference Series 125, 2008.

[2]Wissink, A., Sitaraman, J., Mavriplis, D., Pulliam, T., and Sankaran, V., "A Python-Based Infrastructure for Overset CFD with Adaptive Cartesian Grids," *46th AIAA Aerospace Sciences Meeting*, AIAA-2008-0927, Reno, Nevada, January 2008.

[3]Sitaraman, J., Katz, A., Jayaraman, B., Wissink, A., and Sankaran, V., "Evaluation of a Multi-Solver Paradigm for CFD using Overset Unstructured and Structured Adaptive Cartesian Grids," *AIAA Aerospace Sciences Meeting*, AIAA-2008-0660, Reno, Nevada, 2008.

[4]Jespersen, D., Pulliam, T. and Buning, P., "Recent Enhancements to OVERFLOW," AIAA 1997-0644, 35th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, Aug, 1997.

[5]P. Buning, R. Gomez, W. Scallion, "CFD Approaches for Simulation of Wing- Body Stage Separation," AIAA 2004-4838, 22nd Applied Aerodynamics Conference and Exhibit, Providence, RI, Aug, 2004.

[6]Mavriplis, D. J. and Pirzadeh, S., Large-Scale Parallel Unstructured Mesh Computations for 3D High-Lift Analysis, *Journal of Aircraft*, Vol. 36, No. 6, pp. 987 - 998, Dec 1999.

[7]Mavriplis, D. J., and Venkatakrishnan, V., A Unified Multigrid Solver for the Navier- Stokes Equations on Mixed Element Meshes, *International Journal of Computational Fluid Dynamics*, Vol. 8, pp. 247-263, 1997.

[8]Mavriplis, D. J., Viscous Flow Analysis using a Parallel Unstructured Multigrid Solver, *AIAA Journal*, Vol 38, No. 11, pp. 2067-2076, November 2000.

[9]Mavriplis, D. J., Unstructured Mesh Discretizations and Solvers for Computational Aerodynamics, *AIAA Journal*, Vol 46, No. 6, pp. 1281 - 1298, June 2008.

[10]Hornung, R. D., and S. R. Kohn, Managing Application Complexity in the SAMRAI Object-Oriented Framework, *Concurrency and Computation: Practice and Experience*, Vol. 14, 2002, pp. 347-368.

[11]Hornung, R. D., A. M. Wissink, and S. R. Kohn, "Managing Complex Data and Geometry in Parallel Structured AMR Applications," *Engineering with Computers*, Vol. 22, No. 3-4, Dec. 2006, pp. 181-195. Also see `www.llnl.gov/casc/samrai`.

[12]Wissink, A. M., R. D. Hornung, S. Kohn, S. Smith, and N. Elliott, Large-Scale Parallel Structured AMR Calculations using the SAMRAI Framework, Proceedings of Supercomputing 2001 (SC01), Denver CO, Nov 2001.

[13]Wissink, A. M., D. A. Hysom, and R. D. Hornung, "Enhancing Scalability of Parallel Structured AMR Calculations", Proceedings of the 17th ACM International Conference on Supercomputing (ICS03), San Francisco CA, June 2003, pp. 336–347.

[14]Pulliam, T. H., "Euler and Thin-Layer Navier-Stokes Codes: ARC2D, and ARC3D," Computational Fluid Dynamics Users Workshop, The University of Tennesse Space Institute, Tullahoma, Tennessee, March 12-16, 1984.

[15]Sitaraman, J., Floros, M., Wissink, A. and Potsdam, M.,"Parallel Unsteady Overset Mesh Methodology for a Multi-Solver Paradigm with Adaptive Cartesian Grids", *AIAA Applied Aerodynamics Meeting and Exhibit*, AIAA-2008-7177, Honolulu, Hawaii, August, 2008.

[16]Tung, C., Caradonna, F. X. and Johnson, W. R., "The Prediction of Transonic Flows on an Advancing Rotor," *Journal of the American Helicopter Society*, Vol. 31, No. 3, July 1986, pp. 4-9.

[17]Datta, A., and Chopra, I., "Validation of Structural and Aerodynamic Modeling Using UH-60A Airloads Program Data," *Journal of the American Helicopter Society*, Vol. 51, (1), January 2006, pp. 43- 58.

[18]Potsdam, M., Yeo, H., and Johnson, W., "Rotor Airloads Prediction Using Loose Aerodynamic/Structural Coupling," American Helicopter Society 60th Annual Forum Proceedings, Baltimore, MD, June 7-10, 2004; and in *Journal of Aircraft*,Vol. 43, (3), May-June, 2006, pp. 732- 742.

[19]Strawn, R. C., Caradonna, F. X., and Duque, E. P. N., "30 Years of Rotorcraft Computational Fluid Dynamics Research and Development," *Journal of the American Helicopter Society*, Vol. 51, (1), January 2006, pp. 5-21.

[20]Datta, A., Sitaraman, J., Chopra, I., and Baeder, J., "Analysis Refinements for Prediction of Rotor Vibratory Loads in High-Speed Forward Flight," American Helicopter Society 60th Annual Forum Proceedings, Baltimore, MD, June 2004; and in "CFD/CSD Prediction of Rotor Vibratory Loads in High Speed Flight," *Journal of Aircraft*, Vol. 43, (6), November-December 2006, pp. 1698-1709.

[21]Lim, J. W., Nygaard, T. A., Strawn, R., and Potsdam, M., "Blade-Vortex Interaction Airloads Prediction using Computational Fluid and Structural Dynamics," *Journal of the American Helicopter Society*, Vol. 52 (4), October 2007, pp. 318-328.

[22]Datta, A. and Chopra, I., "Prediction of the UH-60A Main Rotor Structural Loads Using CFD/Comprehensive Analysis Coupling," *Journal of the American Helicopter Society*, Vol. 53, (4), October 2008, pp. 351-365.

[23]Datta, A., Nixon, M., and Chopra, I., "Review of Rotor Loads Prediction with the Emergence of Rotorcraft CFD," *Journal of the American Helicopter Society*, Vol. 52, (4), October 2007, pp. 287-217.

[24]Datta, A. and Johnson, W., "An Assessment of the State-of-the-art in Multidisciplinary Aeromechanical Analysis," American Helicopter Society Technical Specialist's Meeting, San Francisco, January 22-24, 2008.

[25]Servera, G., Beaumier, P., and Costes, M., "AWeak Coupling Method between the Dynamics Code Host and the 3D Unsteady Euler Code Waves," 26th European Rotorcraft Forum, The Hague, Netherlands, September 14-16, 2000.

[26]Altmikus, A. R. M., Wagner, S., Beaumier, P., and Servera, G., A Comparison:Weak versus Strong Modular Coupling for Trimmed Aeroelastic Rotor Simulation, American Helicopter Society 58th Annual Forum Proceedings, Montreal, Quebec, Canada, June 11-13, 2002.

[27]Pomin, H., and Wagner, S., "Aeroelastic Analysis of Helicopter Rotor Blades on Deformable Chimera Grids," *Journal of Aircraft*, Vol. 41, (3), May-June 2004, pp. 577-584.

[28]Pahlke, K., and van der Wall, B., "Chimera Simulations of Multibladed Rotors in High-Speed Forward Flight with Weak Fluid-Structure- Coupling,",*Aerospace Science and Technology*, Vol. 9, (5), July 2005, pp. 379-389.

[29]Saberi, H., Khoshlahjeh, M., Ormiston, R., and Rutkowski, M. J., "Overview of RCAS and Application to Advanced Rotorcraft Problems," 4th AHS Decennial Specialists' Conference on Aeromechanics, San Francisco, CA, January 21-23, 2004.

[30]Alonso, J., P. LeGresley, E. v. d. Weide, J. Martins, J. Reuther, pyMDO: A Framework for High-Fidelity Multi- Disciplinary Optimization, AIAA-2004-4480, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany NY, Aug 2004.

[31]Morton, S., "Rigid Maneuvering and Aeroelastic Results for Kestrel—A CREATE Simulation Tool," AIAA Paper2010-1233, 48th Aerospace Sciences Meeting, Jan 2010, Orlando, FL.

[32]Anderson, W., and Bonhaus, D., An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids, *Computers & Fluids*, Vol. 23, No. 1, 1994, pp. 1-21.

[33]O'Brien, D., *SMEMRD User Manual*, V 1.4, 2009.

[34]Mavriplis, D. J., Grid Resolution Study of a Drag Prediction Workshop Configuration using the NSU3D Unstructured Mesh Solver, AIAA paper 2005-4729, presented at the 23rd AIAA Applied Aerodynamics Conference, Toronto, Canada, June 2005.

[35]Mavriplis, D. J., Third Drag Prediction Workshop Using the NSU3D Unstructured Mesh Solver, AIAA Journal of Aircraft, Vol. 45, No. 3, pp. 750 - 761, March 2008

[36]Mavriplis, D. J. and Yang, Z., "Construction of the Discrete Geometric Conservation Law for High-Order Time Accurate Simulations on Dynamic Meshes," *Journal of Computational Physics*, Vol. 213, (1), April 2006, pp. 557-573.

[37]Mavriplis, D. J., Aftosmis, M. J., and Berger, M., "High Resolution Aerospace Applications Using the NASA Columbia Supercomputer," *The Int. J. of High Performance Computing Appl.*, Vol. 21, (1), pp. 106-126, 2007.

[38]O'Brien, D.M. and Smith, M.J., "Improvements in the Computational Modeling of Rotor / Fuselage Interactions Using Unstructured Grids," 61st Annual Forum of the American Helicopter Society, Grapevine, TX, June 2005.

[39]Rajagopalan, R.G. and Kim, C.K., "Laminar Flow Analysis of a Rotor in Hover," Journal of the American Helicopter Society, Vol. 36, No.1, pp.12-23, 1991.

[40]Yikloon Lee and James D. Baeder. "Implicit hole cuttinga new approach to overset grid connectivity," In 16th AIAA Computational Fluid Dynamics Conference, Washington, DC, June 2003. AIAA. Orlando, Florida.

[41]Choi, S. and Datta, A., "CFD Prediction of Rotor Loads Using Time-Spectral Method and Exact Fluid-Structure Interface", Journal of American Helicopter Society, AIAA Paper 2008-7325, 26th AIAA Applied Aerodynamics Conference, Aug. 2008, Honolulu, Hawaii.

[42]Shende, S., "An Infrastructure for Deploying Multi-Language CFD Applications," Final report, PET CE-KY8-SP1, 2009. https://okc.erdc.hpc.mil

[43]Jenkins, L. N., Yao, C. S., Bartram, S., Harris, J., Allan, B., Wong, O., and Mace, D., Development of a Large Field-of-View PIV System for Rotorcraft Testing in the 14x22 Subsonic Wind Tunnel, Tech. Rep., 65th Annual American Helicopter Society Forum, Grapevine, TX, May 2009.

[44]Allan, B.G., Jenkins, L.N., Yao, S.H., Bartram, S.M., Hallissy, J.B., and Harris, J., Noon, K. W., Wong, O. D., Jones, H. E., Malovrh, B. D., and Reis, D. G., "Navier-Stokes Simulation of a Heavy Lift Slowed-Rotor Compound Helicopter Configuration," Presented at the American Helicopter Society 65th Annual forum, Grapevine, Texas, May 27-29, 2009.

[45]Brand, A.G., McMahon, H.M., and Komerath, N.M., "Surface Pressure Measurements on a Body Subject to Vortex-Wake Interaction," *AIAA Journal*, Vol. 27, No. 5, May 1989, pp. 569-574.

[46]Komerath, N.M., McMahon, H.M., Brand, A.G., Liou, S.G., and Mavris, D.M., "Measurement and Prediction of the Aerodynamic Interactions Between a Rotor and an Airframe in Forward Flight," Proceedings of the 45th Annual Forum of the American Helicopter Society, Washington, D.C. pp. 323-335, May 1989.

[47]Chaffin, M. S., and Berry, J. D., "Helicopter Fuselage Aerodynamics Under a Rotor by Navier-Stokes Simulation," *Journal of the American Helicopter Society*, Vol. 42, (3), July 1997, pp. 235-244.

[48]Steijl, R. and Barakos, G. N., "Computational Study of Helicopter Rotor-Fuselage Aerodynamic Interactions," *AIAA Journal*, Vol. 47, (9), September 2009, pp. 2143-2157.

[49]Lee-Rauch, E., M and Biedron, R. T., "Simulation of an Isolated Tiltrotor in Hover with an Unstructured Overset-Grid RANS Solver", Presented at the American Helicopter Society 65th Annual Forum, Grapevine, TX, May 27-29,2009.

[50]Swanson, S. M., McCluer, M. S., Yamauchi, G. K., and Swanson, A. A., Airloads Measurements from a 1/4-Scale Tiltrotor Wind Tunnel Test, *25th European Rotorcraft Forum*, Rome, Italy, September 1999.

[51]Potsdam, M. A., and Strawn, R. C., "CFD Simulations of Tiltrotor Configurations in Hover," *Journal of the American Helicopter Society*, Vol. 50, (1), January 2005, pp. 82-94.

[52]Bousman, W. G. and Kufeld, R. M, *UH-60A Airloads Catalog*, NASA/TM-2005-212827, AFDD/TR-05-003, August 2005.

[53]Bhagwat, M. J., Ormiston, R. A., Saberi, H. A. and Xin, H., Application of CFD/CSD Coupling for Analysis of Rotorcraft Airloads and Blade Loads in Maneuvering Flight, American Helicopter Society 63rd Annual Forum Proceedings, Virginia Beach, VA, May 1-3, 2007.

[54]Kamkar, S. J., Jameson, A. J., and Wissink, A. M., "Feature-Driven Cartesian Adaptive Mesh Refinement in the Helios Code," *48th AIAA Aerosciences Conference*, AIAA Paper, Orlando, FL, Jan 4-7, 2010.