Journal of Computational Physics 229 (2010) 4703–4723

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids

Jayanarayanan Sitaraman [a,*], Matthew Floros [b], Andrew Wissink [c], Mark Potsdam [c]

[a] University of Wyoming, Laramie, Wyoming, USA
[b] US Army Research Laboratory, Aberdeen Proving Ground, MD, USA
[c] Army Aeroflightdynamics Directorate, Moffet Field, CA, USA

ARTICLE INFO

Article history:
Received 28 May 2009
Received in revised form 12 February 2010
Accepted 7 March 2010
Available online 15 March 2010

Keywords:
Overset grid
Domain connectivity
Adaptive grids
Multi-solver paradigm
Parallel execution

ABSTRACT

This paper describes the algorithms and functionality of a new module developed to support overset grid assembly associated with performing time-dependent and adaptive moving body calculations of external aerodynamic flows using a multi-solver paradigm (i.e. different CFD solvers in different parts of the computational domain). We use the term "domain connectivity" in this paper to denote all the procedures that are involved in an overset grid assembly, and the module developed is referred henceforth as the domain-connectivity module. The domain-connectivity module coordinates the data transfer between different solvers applied in different parts of the computational domain – body fitted structured or unstructured to capture viscous near-wall effects, and Cartesian adaptive mesh refinement to capture effects away from the wall. The execution of the CFD solvers and the domain-connectivity module are orchestrated by a Python-based computational infrastructure. The domain-connectivity module is fully parallel and performs all its operations (identification of grid overlaps and determination of data interpolation strategy) on the partitioned grid data. In addition, the domain connectivity procedures are completely automated such that no user intervention or manual input is necessary. The capabilities and performance of the package are presented for several test problems, including flow over a NACA 0015 wing and an AGARD A2 slotted airfoil, hover simulation of a scaled V-22 rotor, and dynamic simulation of a UH-60A rotor in forward flight. A modification to the algorithm for improved domain connectivity solutions in problems with tight tolerances as well as heterogeneous grid clustering is also presented.

Published by Elsevier Inc.

1. Introduction

Traditional CFD codes are often written to support a single gridding and solution paradigm. Grids fall under three main classifications: Cartesian (structured or unstructured), structured-curvilinear (body-fitted) or unstructured (tetrahedral, hexahedral or prismatic). Each meshing paradigm has specific advantages and disadvantages. For example, Cartesian grids are easy to generate, to adapt, and to extend to higher-order spatial accuracy, but they are not well-suited for resolving boundary layers around complex geometries. Structured curvilinear grids work well for resolving boundary layers, but the grid generation process for complex geometries remains tedious and requires considerable user expertise. General unstructured grids are well-suited for complex geometries and are relatively easy to generate, but their spatial accuracy

* Corresponding author. Tel.: +1 301 741 3216.
E-mail addresses: jsitaram@uwyo.edu (J. Sitaraman), matt.floros@us.army.mil (M. Floros), awissink@us.army.mil (A. Wissink), mpotsdam@us.army.mil (M. Potsdam).

0021-9991/$ - see front matter Published by Elsevier Inc.
doi:10.1016/j.jcp.2010.03.008

is often limited to second-order, and the associated data structures are less computationally efficient than their structured-grid counterparts.

Thus, while a single gridding paradigm brings certain advantages in some portions of the flowfield, it also imposes an undue burden on others. A computational platform that supports multiple mesh paradigms provides the potential for optimizing the gridding strategy on a local basis. However, integrating different meshing paradigms into a single large monolithic code is complex and usually relegates at least one of the models to be less accurate, less optimized, or less flexible than the original standalone solver.

This work describes an approach to mitigate these issues by using a multiple-mesh strategy that is implemented through the use of *multiple* CFD codes, each optimized for a particular mesh type. It was developed for analysis of rotorcraft, but features several enabling technologies for extension to other types of unsteady, moving-body problems. In addition to supporting heterogeneous grids and solvers, the software was developed to minimize the analysis burden on the user. While an interface procedure is required for each specific solver, this interface must only be written once. Once the interface exists, the domain connectivity software retrieves the information it needs directly from the flow solvers and the grids themselves, so the human analyst does not need to provide input specific to each problem.

For rotorcraft analysis, the approach is to apply unstructured or body-fitted curvilinear grids near the body surface to capture complex geometry and viscous boundary layers. A short distance from the body surface, the flow is determined by a high-order block-structured adaptive Cartesian solver that adapts time-dependently to capture wake effects. Previous work [1,2] has addressed the development of a Python-based multi-solver infrastructure using both unstructured (NSU3D [3]) and structured (UMTURNS [4]) near-body solvers and an adaptive Cartesian off-body solver (SAMARC). Using this high-level framework, discrete codes can be coupled together to obtain a contiguous solution over the entire computational domain. The Python code merely manages data pointers to pass information between the solvers and the domain connectivity package and adds negligible overhead to the analysis. Fig. 1 shows an example calculation of flow over a sphere using this approach for which both the viscous boundary layer and the shed vorticity are accurately captured.

A critical aspect of the multi-mesh/multi-solver approach is the need for data exchange between the different meshes, which is facilitated in this work using the well-established Chimera-based overset procedure. As shown in Fig. 2, the near-body and off-body meshes are constructed to be overlapping and the data exchange is facilitated using the domain-connectivity module. The focus of the present paper is on the development of a new domain-connectivity module that can efficiently support the overset multiple-mesh paradigm for large-scale unsteady moving-body computations.

The domain connectivity procedure entails identifying hole regions, where one grid cuts out another, and interpolating data between the grids at boundaries, both hole boundaries and outer boundaries. Specifically, the domain connectivity procedures involve the evaluation of fringe data points (points that receive data), donor cells (points that provide data), hole points (points that do not need to be solved) and interpolation weights. The process of identifying fringes and holes (termed hole-cutting) can be done in one of two ways, explicitly or implicitly.

Explicit hole cutting means the hole boundaries are explicitly defined according to a representation of the underlying component surface. Holes are cut in the background grids in "cookie-cutter" fashion using this representation. Some of the earliest practical explicit hole-cutting techniques proposed by Meakin [5] for OVERFLOW applied analytic shapes to surround the solid component. This approach was later refined to use a surface ray-casting [6], the approach currently used in

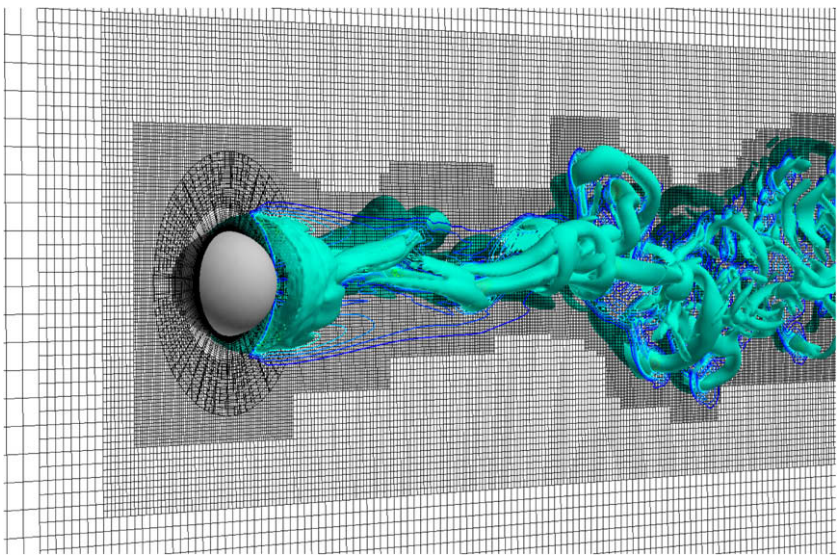

Fig. 1. Unsteady flow over a sphere at Re = 1000 using the multiple-solver approach. The NSU3D unstructured solver is used near the body surface, the high-order Cartesian AMR solver SAMARC is used in the field, and data are interpolated between the solvers using Chimera-based interpolation [1].

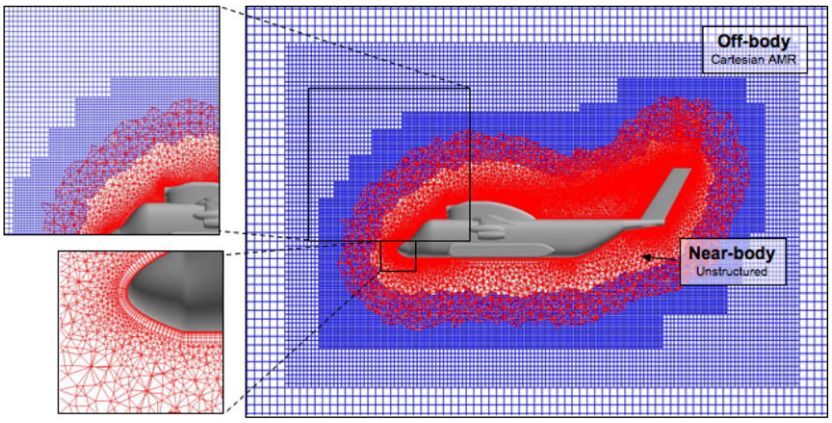

Fig. 2. Overset near/off-body gridding. Unstructured or curvilinear grids to capture geometric features and boundary layer near-body surface, adaptive block-structured Cartesian grids to capture far-field flow features.

OVERFLOW [7] today. Petersson developed a similar ray tracing approach in the CHALMESH [8] software, an overset grid generator for the Overture [9] package. Explicit hole cutting has seen widespread use for practical applications but has the drawback that it requires additional inputs and steps to form the explicit hole boundary during the overset assembly process. Implicit hole cutting means there is no explicit definition of the hole boundary. Instead, the hole is computed purely by comparing volume grid information and flow boundary conditions, Whichever grid has the best resolution, referred to as *resolution capacity*, cuts a hole in all other grids that occupy the same region in space. Implicit hole cutting method was first demonstrated by Chessire and Henshaw [10] in the CMPGRD package, an early predecessor to Overture. The Ogen [11] software in Overture [9] uses it today. The PEGASUS5 [12] package was one of the first to demonstrate implicit hole cutting for practical 3D static-grid applications. Lee [13,14] demonstrated it for moving-grid rotorcraft applications. The benefit of implicit hole cutting is that it is completely automated; it does not require any definition of the hole boundary *apriori*. The drawback is that it tends to be complicated to program and more computationally demanding than explicit hole cutting. We note following Lee [14] that implicit hole cutting methodology, by itself, does not tag the portions of the grids inside the boundaries of a solid body as hole-points. It can only identify the fringe region surrounding the solid body. We refer to the elimination of the points inside solid bodies as minimum hole-cutting. In our methodology, we include the capability to perform minimum hole-cutting as well. Additionally, our algorithm, unifies both donor search and minimum hole-cutting in to one single procedure obviating the need for separate procedure outlined by Lee [14].

Other notable domain connectivity packages include SUGGAR/DiRTlib [15], CHIMPS [16], BEGGAR [17], and FASTRAN [18]. These packages (including those mentioned in the paragraphs above) all have their merits and drawbacks. Most are intended to be used within a single solver and do not have the capability to support multiple solvers. Most are intended for exclusively structured or exclusively unstructured grids; they do not efficiently handle a mix of grid types. Many are not parallel, which limits their applicability for large-scale problems on HPC systems.

In this work we seek to develop a domain connectivity package that functions in the Python-based multi-solver paradigm, efficiently exploiting the mix of structured and unstructured grid types. It will employ implicit hole cutting for purposes of automation but, unlike most of the earlier implementations of implicit hole cutting, it is implemented on distributed memory parallel systems using MPI-based communication in order to take advantage of HPC systems widely used for CFD today.

The new domain-connectivity module developed as part of this work is termed PUNDIT, which is an acronym for Parallel UNsteady Domain Information Transfer. As outlined earlier, our objective is to create a package which is modular, parallel, and supports efficient automated domain connectivity operations for all participant grid types (unstructured, structured curvilinear and adaptive Cartesian).

2. Methodology

The coupling of the near-body solver, off-body grid manager, off-body solver, and domain-connectivity module is accomplished through a Python-based infrastructure with emphasis on preserving the modularity of the participating solvers. In addition to the advantages in efficiency and ease in code development, coupling existing mature simulation codes through a common high-level infrastructure provides a natural way to reduce the complexity of the coupling task and to leverage the large amount of verification, validation, and user experience that typically go into the development of each separate model. The infrastructure discussed here currently couples the following codes through a Python infrastructure: (a) the parallel NSU3D code [3] for the unstructured near-body solver, (b) the parallel UMTURNS code as the structured near-body solver, (c) the SAMRAI framework [19] for the off-body Cartesian grid generation, adaptation, and parallel communication, (d) the serial high-order ARC3DC code for solution on the Cartesian blocks, and (e) the domain-connectivity module (PUNDIT).

Brief descriptions of the implementations of the first four components are outlined below. More detailed descriptions of the implementations as well as performance statistics and validation are described in more detail in Refs. [1,2]. This paper is devoted to detailed descriptions of algorithms and implementation of the domain-connectivity module (PUNDIT). The infrastructure which integrates the simulation capabilities of all the modules mentioned above is called HELIOS (Helicopter Overset Simulations) and is developed under the DoD HPC HI-ARMS program. Because of the modular development approach being taken, the individual components of HELIOS, such as the solvers and the domain connectivity package, can be used independently for moving body problems other than rotorcraft.

2.1. Python infrastructure

Python-based computational frameworks have been developed previously by several researchers [20] as a means of coupling together existing legacy codes or modules. Such a framework has a number of advantages over a traditional monolithic code structure: (1) it is easier to incorporate well-tested and validated legacy codes rather than to build the capabilities into an entirely new code, (2) there is less code complexity in the infrastructure itself, so maintenance and modification costs are less, and (3) it is easier to test and optimize the performance of each module separately, often yielding better performance for the code as a whole. Essentially, Python enables the legacy solvers to execute independently of one another and reference each other's data without memory copies or file I/O. Further, the Python-wrapped code may be run in parallel using pyMPI or myMPI, with each of the solvers following its native parallel implementations. Further details of the Python infrastructure used here can be found in [1,2].

2.2. Flow solver modules

Well-established legacy codes are employed for all of the independent modules in this study. For the unstructured solver in the near-body region, we employ the NSU3D [3] code, which is an implicit node-centered Reynolds-Averaged Navier–Stokes (RANS) code capable of handling arbitrary unstructured mesh elements. In addition, we also use a curvilinear structured mesh solver for the near-body region, namely the UMTURNS code, which is also an implicit RANS solver. For the Cartesian grids in the off-body region, we utilize a Cartesian grid derivative of the well-known ARC3D [21] code, referred to here as ARC3DC. The ARC3DC code employs third-order temporal discretizations using a multi-stage Runge–Kutta time-stepping framework and is capable of up to fifth-order accurate spatial discretizations. Further, the Cartesian grids in the off-body are automatically generated and managed for parallel execution by the SAMRAI infrastructure [19]. As mentioned earlier, these codes or modules are combined together using a Python-based framework that orchestrates the execution of these modules and the associated data transfers.

2.3. Meshing paradigm

The meshing paradigm consists of separate near-body and off-body grid systems. The near-body grid typically extends a short distance from the body, sufficient to contain the boundary layer. This grid can be a structured curvilinear grid or an unstructured tetrahedral or prismatic grid that has been extracted from a standard unstructured volume grid or generated directly from a surface triangulation using hyperbolic marching. The reason for using curvilinear or unstructured grids in the near-body region is to properly capture the geometry and viscous boundary layer effects, which are difficult or impossible to capture with Cartesian grids alone. We further note that either structured or unstructured grids will work equally well in the near-body region from the point of view of our infrastructure. Away from the body the near-body grid solution is interpolated onto a Cartesian background mesh with the aid of the domain connectivity algorithm. This transition normally occurs at a distance wherein the sizing of the near-body grid cells is approximately commensurate with the sizing of the Cartesian mesh in the off-body region.

The off-body grid system consists of a hierarchy of nested refinement levels, generated from coarsest to finest. In the Structured Adaptive Mesh Refinement (SAMR) paradigm [22], the coarsest level defines the physical extents of the computational domain. Each finer level is formed by selecting cells on the coarser level and then clustering the marked cells together to form the regions that will constitute the new finer level. Solution-based refinement progresses as follows: physical quantities and gradients are computed at each Cartesian cell using the latest available solution and those cells that hold values deemed to require refinement are marked. The marked cells are then clustered to form the new set of patches that constitute the next finer level. The process is repeated at each new grid level until the geometry and solution features are adequately resolved. We note that this entire procedure is automated within the software and no user intervention is required. The procedure of adaptive mesh refinement is graphically illustrated in Fig. 3.

An example of the overset near-body/off-body meshing strategy is given in Fig. 2, which shows the meshes for flow computations over a helicopter fuselage. Here, the mixed-element unstructured near-body grid envelops the fuselage, while a multi-level Cartesian off-body grid extends from the outer boundary of the near-body grid to the far-field boundary. The two sets of meshes overlap in the so-called fringe region, where data are exchanged between the grids.

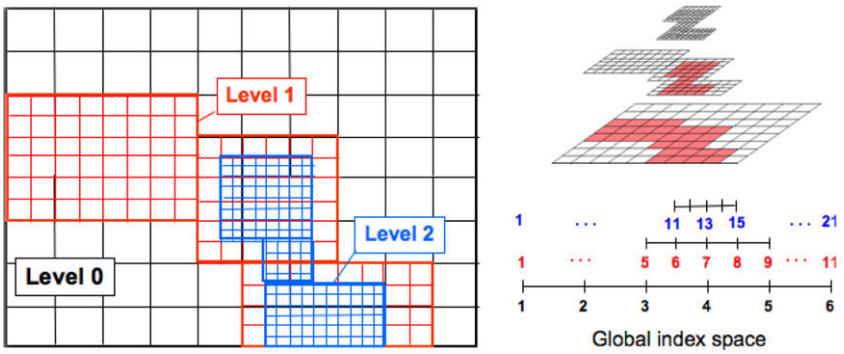

Fig. 3. Block structured AMR grid composed of a hierarchy of nested levels of refinement. Each level contains uniformly-spaced logically-rectangular regions, defined over a global index space.

2.4. Overset methodology

The overall solution procedure for the overset meshes is as follows. At each iteration step, the solutions of the fluid equations in each mesh are obtained independently with the solution in the each fringe region being specified by interpolation from the overlapping "donor" mesh as a Dirichlet boundary condition. At the end of the iteration, the fringe data are exchanged between the solvers so that the evolution of the global solution is faithfully represented in the overset methodology. Further, if one or more of the meshes is moving or changing, the fringe regions and the interpolation weights are recalculated at the beginning of the time step. This procedure is repeated for each iteration step until solution convergence is attained in both the near- and off-body meshes.

2.5. Domain connectivity module (PUNDIT)

For each solver, the solution at the fringe cells is obtained by interpolation from the overlapping "donor" mesh. The domain connectivity module (PUNDIT) is responsible for determining the appropriate interpolation weights for each fringe point. Further, in the case of multiple overlapping meshes, PUNDIT must also identify one mesh as the donor for each fringe point in each mesh. For static meshes, these operations are done once, at the beginning of the computation, while, for the more general case of moving or adapting meshes, the determinations of donors and weights has to be repeated within the time-stepping or iteration loop.

PUNDIT adopts the implicit hole cutting procedure followed by NAVAIR-IHC [13]. 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. Moreover, the implicit hole-cutting procedure produces an arbitrary number of fringe points based on mesh density compared to traditional methods which use a fixed fringe width (usually single or double). The critical parameter that quantifies the quality of a grid cell or node is termed as "resolution capacity". PUNDIT nominally uses the cell volume as the resolution capacity for a grid cell and the average of cell volumes of all associated grid cells as resolution capacity for a grid node. A modification to this procedure to account for wall proximity is described in Section 5. In addition, PUNDIT separates the near-body to near-body and near-body to off-body domain connectivity procedures to facilitate automatic off-body grid generation and to improve efficiency [23].

Following are the steps followed by PUNDIT to determine the domain connectivity information in a parallel environment. Partitioning of grid and solution data is assumed to be performed using an appropriate solver-based load balancing scheme before these steps are executed.

1. *Registration*: On each processor, the flow solvers register grid and solution data pointers for every grid block (both near-body and off-body) with PUNDIT. The grid data consist of the Cartesian coordinates and cell to node connectivity. The solution data are composed of flow variables at each of the grid nodes.
2. *Profiling*: PUNDIT profiles each of the grid blocks and forms meta-data representations to facilitate faster donor search operations. The main procedures that are executed in this step are:
   - Minimal *bounding box* computation: Oriented bounding boxes are constructed instead of axis-aligned bounding boxes to minimize the search space (Fig. 4(b)).
   - Division of minimal bounding box in to *vision space bins*: The vision space bins are smaller Cartesian boxes within the bounding box. The size of the vision space bins is determined by dividing the volume of the bounding box by the total number of cells contained in it (Fig. 4(c) and (d)).
   - Generation of a *cumulative fill-table*: The cumulative fill table is a bin-wise reordering of the cell indices that are contained in each vision space bin. They facilitate fast identification of all cells contained within each vision space bin.

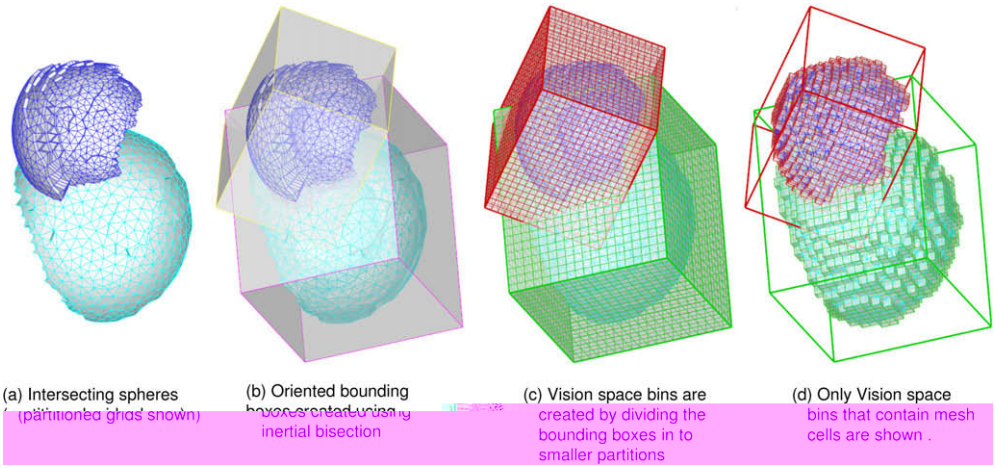

Fig. 4. Oriented bounding boxes and vision space bins that are created during preprocessing to accelerate donor search processes.

- Estimation of element *resolution capacity*: Element resolution capacity is the grid quality measure for each cell and each grid node. Nominally, cell volumes are used as resolution capacity for each grid cell and average of all the associated cell volumes is used as the resolution capacity for each grid node. A smaller value of resolution capacity indicates better grid quality, or improved solution accuracy. A method to modify resolution capacity to retain cells near a solid wall in the grid containing the wall is described in Section 5.
- Evaluation of *cell-to-cell connectivity*: The containment search strategy uses a gradient based approach involving walking from cell to cell until the containment criteria are satisfied. Cell-to-cell connectivity is required to aid in this process. Since the input grid data consist of only cell-to-node connectivity, the cell-to-cell connectivity is evaluated as a preprocessing step. We utilize hash table based sorting for rapid evaluation of cell-to-cell connectivity from cell-to-node connectivity. Nonetheless this step still remains the most performance intensive among all the profiling operations. However, if the mesh topology is preserved, this step needs to be performed only once even for unsteady moving mesh problems.

3. *Near-body to near-body domain connectivity*: In this step the domain connectivity operations are performed between all near-body blocks on all processors. The sequence of the operations is as follows:
   - *Bounding box exchange and intersection checks*: The bounding boxes constructed in each processor in the *profiling* step are gathered in every processor. Each processor checks for intersection of bounding boxes of its grid blocks with the global set of bounding boxes, which aids in identifying candidate donor grid blocks. The bounding box intersection checks are optimized such that detection of intersection as well as identification of vision space bins containing potential receiver grid points are simultaneously processed. The identification of vision space bins accelerates the identification of potential receiver grid points as only those nodes which belong in these vision space bins are checked for containment in the donor bounding box.
   - *Communication of potential receivers:* Following the intersection checks a communication packet is set up and exchanged between all the processors. The communication packet consists of a list of coordinates of all potential receiver grid points and their resolution capacity organized such that the donor search can be directed to the appropriate grid block in the candidate processor. Upon completion of this communication, each grid block in each processor will form a list of points for which it needs to locate donor cells.
   - *Donor search*: Fig. 5 shows a graphical illustration and flow chart of the donor search algorithm outlined below. The donor search algorithm proceeds as follows: For any potential receiver point, a localization is performed by locating the vision space bin that contains it. This is a trivial operation since the vision space bins follow a Cartesian structure within the oriented bounding box. Subsequently a spiral search (i.e. vision space bins are tagged along an outward spiral beginning at the initial location) is performed until a vision space bin with at least one grid cell centroid is located. This grid cell is chosen as the seed for initiating the so-called "stencil walk process". Fig. 6 graphically illustrates the stencil walk process using a two dimensional analogy. A line (termed *stencil walk path*) is created by connecting a point inside a potential donor cell (centroid for the initial cell and face intersection point for subsequent cells) with the receiver point. A check is made to determine if this edge intersects any of the faces of the cell. If an intersection is located, the cell which forms the neighbor at that face is chosen as the next potential donor cell. This procedure continues until a cell with no intersections is located which would be the donor cell for the given point. For a face which shows intersection but has no neighbor (i.e. a boundary face), a check is conducted using the spiral search technique to check if the edge intersects any other boundary faces in the neighborhood. If an intersection is found then stencil walking proceeds to the cell which owns that boundary face. This procedure addresses complex grid boundaries such as those found in partitioned unstructured grids or thin trailing edges. If a donor cell is not located, the receiver point is pronounced to have no donor cell in the current grid block.

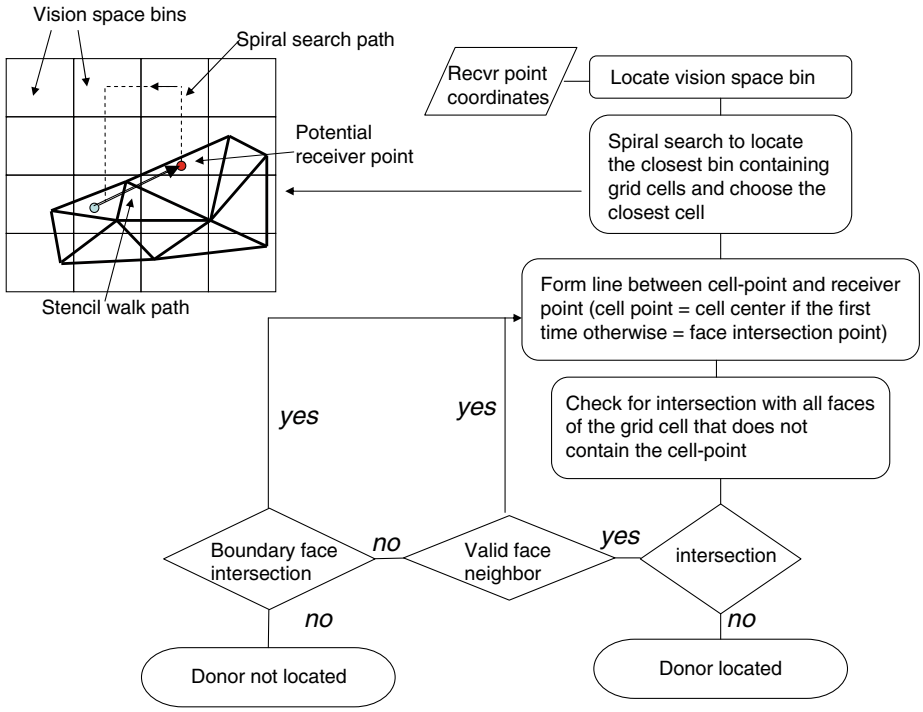

Fig. 5. Flowchart of donor finding algorithm that uses vision space bin based localization and stencil-walk based search.

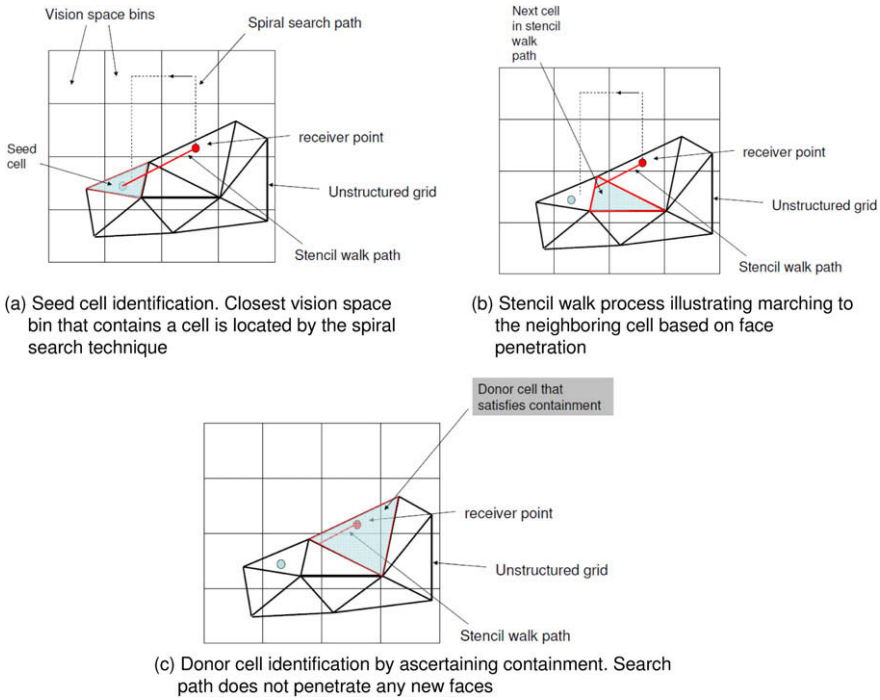

Fig. 6. Illustration of the stencil walk search which locates donor cells that satisfy containment.

- *Minimum hole cutting*: Hole points are those which are located inside a solid body and hence do not need to be solved or interpolated. Hole points are located as part of the donor search process. The hole-cutting logic is invoked whenever the donor search method terminates with a boundary face penetration. If the final boundary face that the stencil walk

path intersects is a wall boundary face, the receiver point will be tagged as a hole point (i.e. it is inside the solid body). A two dimensional analogy of this procedure is graphically illustrated in Fig. 7. In this case the donor search proceeds along the chosen path and finally intersects a wall boundary face indicating that the receiver point is indeed inside the solid body. Once a point is tagged as a hole by a near-body grid partition, it can be tagged as a fringe point only by another grid partition from the same near-body grid. Hence it can no longer receive data even if a suitable donor is found by another grid partition (belonging to another near-body grid or a off-body block). This ensures that the volume within the body is blanked for the nodes in all grids which overlap within the body.

- *Generation of the communication table:* Once a list of donor cells is obtained in every processor, it is communicated back to the processors that requested the donor search. Every processor then performs an evaluation of potential donor cells, such that the donor of the best resolution capacity is chosen for every receiver grid point. Indices of all the potential donors that are not acceptable are communicated back to the appropriate donor processors such that they can update their donor list. Additionally a quality check is performed to make sure that no donor cell has a receiver point as its vertex. If such a cell is located it is deleted from the donor list and this information is communicated to the receiver processors who adjust their receiver lists accordingly as well. This process rigorously establishes donor quality since all donors will be composed of only nodes which are being solved and are not themselves receiver points. The final product of this step is a communication table consisting of a list of donor cells and receiver points in each processor synchronized for data interpolation.

4. *Off-body grid generation*: The adaptive Cartesian mesh generation is dictated by the resolution of the near-body meshes at their boundaries. In order to achieve this, a list is generated which is composed of all outer boundary points of near-body meshes across processors that did not find a near-body donor in the previous step. This list is communicated to the off-body grid manager (SAMRAI) which automatically generates/adjusts nested Cartesian meshes such that the resolution is commensurate at the near-body boundaries.
5. *Off-body to Near-body connectivity*: First valid donors (i.e those with better resolution capacity) for all points in near-body grid blocks are searched in the off-body Cartesian blocks. This search process is extremely efficient because of the isotropic nature of the Cartesian blocks and the global grid indexing followed by the off-body grid manager. A donor Cartesian cell can be located in just a single step. In addition, this process identifies all the Cartesian blocks which might contain potential receiver points. Following this, valid donors in the near-body grids are searched for all potential off-body grid points using the same search process that was outlined for the near-body to near-body domain connectivity. Once the donors and receiver points are located for both the off-body and near-body grids, a communication pattern is followed which updates the communication tables that synchronize the donor and receiver lists in each processor. Donor quality is again guaranteed by removing low quality donors and their corresponding receivers from the communication tables.

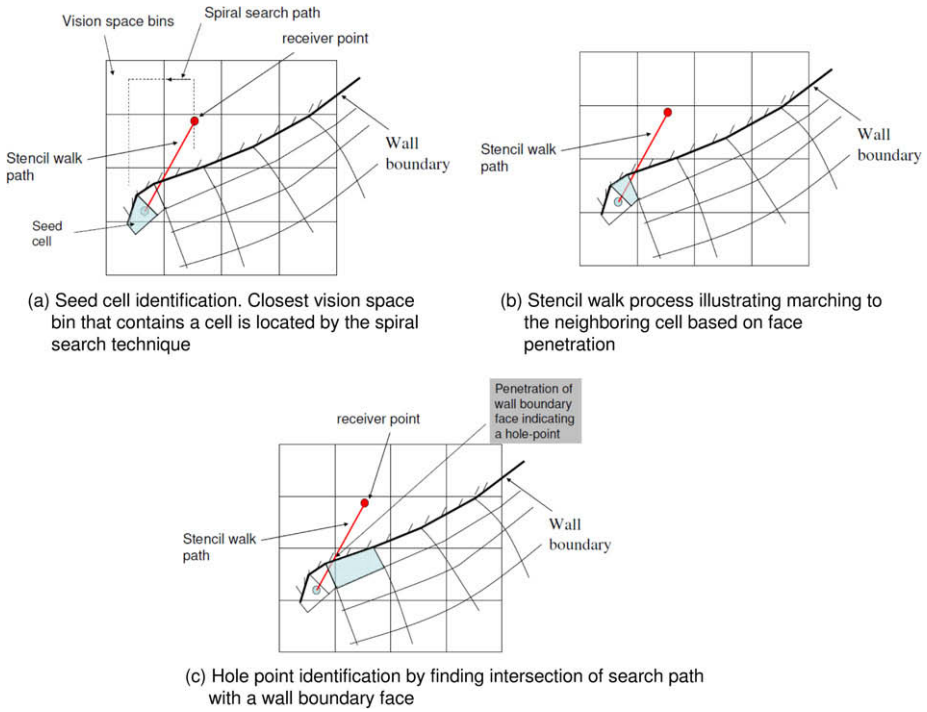

Fig. 7. Illustration of the hole cutting methodology using wall boundary face penetration criteria

Once the communication tables are finalized, the interpolation weights for each grid node of each donor cell in the donor list are computed. Tri-linear basis functions are used for data interpolation and the interpolation weights are computed using a Newton–Raphson procedure.

6. *Data interpolation*: Data are interpolated in a three step process. The first step consists of populating a communication buffer using the solution data and the donor list from the communication tables. These data buffers are exchanged between processors through inter-processor communication in the second step. In the third and final step the solution data are updated in each processor based on the receiver list from the communication tables. In contrast to many of the existing domain connectivity solvers, PUNDIT minimizes the communication overhead by maintaining the interpolation weights in the host processor. The client processor only receives solution data and is capable of assigning them appropriately because of the synchronization in communication tables. Moreover, interpolation weights are evaluated only for the final list of donors, minimizing the amount of floating point operations. In order to support interfacing of solvers which follow different non-dimensionalizations for flow variables, PUNDIT maintains non-dimensionalizing factors for each flow variable from each participant solver. The data buffers that are exchanged are in the dimensional form and appropriate non-dimensionalization is applied at the update step based on the factors provided by each participant code. It is worth noting that the implementation of data interpolation is quite general and does not impose any restriction on the type or number of flow variables.

3. Software verification and application results

We present results using PUNDIT within the flow solver package HELIOS for several examples. The results section is organized as follows: first, test cases for interpolation accuracy and scalability are presented. Then, application results are presented which demonstrate the use of the software for adapting and moving body meshes. The first result is a NACA0015 fixed wing test case which demonstrates the advantages of adaptive grids and higher-order methods for wake capturing. The next set of results illustrates the capability of PUNDIT to perform domain connectivity operations which automate hole cutting and generate optimal overlaps. The domain connectivity solutions are shown for the AGARD A2 test case (NHLP-2D slotted airfoil) in steady and unsteady motion. Following that we present results for rotorcraft simulations for the 1/4th scaled V-22 and full scale UH-60A rotors. Additional application results are presented in Ref. [24].

3.1. Interpolation accuracy

All the polyhedra that are common in unstructured mixed-element meshes (tetrahedra, pyramids, prisms and hexahedra) are supported in PUNDIT. Tri-linear basis functions are used to perform data interpolation inside each of these polyhedra. The accuracy of interpolation was verified by constructing unstructured meshes in a unit cube of incremental resolutions. PUNDIT is used then to perform search and interpolation for a random set of one million points distributed in the unit cube. The data at the grid nodes of the unstructured grid are set up using a set of chosen test functions listed in Table 1. The interpolated values at each of the random points are then correlated with the exact test function values to estimate the interpolation error (it is computed as the infinity norm $\|e\|_\infty := max(|e_1|,\ldots,|e_n|)$, where $e_i = f_i - f_i^{exact}$) between corresponding values. Fig. 8 shows the variation of interpolation error with improving resolution. For linear test functions, the tri-linear interpolation gives exact solutions within machine precision. For a non-linear test function, interpolations in all types of polyhedra show 2nd order accuracy. Interpolation accuracy is also verified to be 2nd order for a test function that uses the Lamb–Oseen line vortex solution. We note that any point, that got a incorrect donor or did not get a donor (it will retain its degenerate function value of zero), will pollute the interpolation error norm and degrade the order of accuracy. Therefore, the 2nd order accuracy verified in Fig. 8, also verifies the robustness of the search algorithm in locating the correct donors for all of the random set of points.

3.2. Scalability Studies for UH-60A and TRAM Rotors

The performance of the domain-connectivity module is studied by varying the number of processors used for program execution. Note that partitioning of the grid data is handled by flow solvers and PUNDIT operates on this partitioned grid data. Fig. 9 shows the speedup in wall-clock time with increasing number of processors. Three sets of grid systems are studied to calibrate scalability. The TRAM grid system consists of 0.8 million near-body nodes and 5.1 million off-body nodes. The

Table 1
Definition of test functions used for verification of interpolation accuracy

Linear test function	$f(x,y,z) = x + y + z$
Non-linear test function	$f(x,y,z) = cos(x)cos(y)cos(z)$
Lamb–Oseen line vortex exact solution for velocity	$\vec{f}(\vec{r}) = \frac{\Gamma}{2\pi h}(1 - e^{-\sigma h^2})\hat{e}_\theta$ where Γ is the circulation strength, $\vec{r}$ is the position vector, h is the orthogonal distance from $\vec{r}$ to the axis of the vortex and $\hat{e}_\theta$ is the vector orthogonal to the plane containing $\vec{r}$ and the axis of the vortex

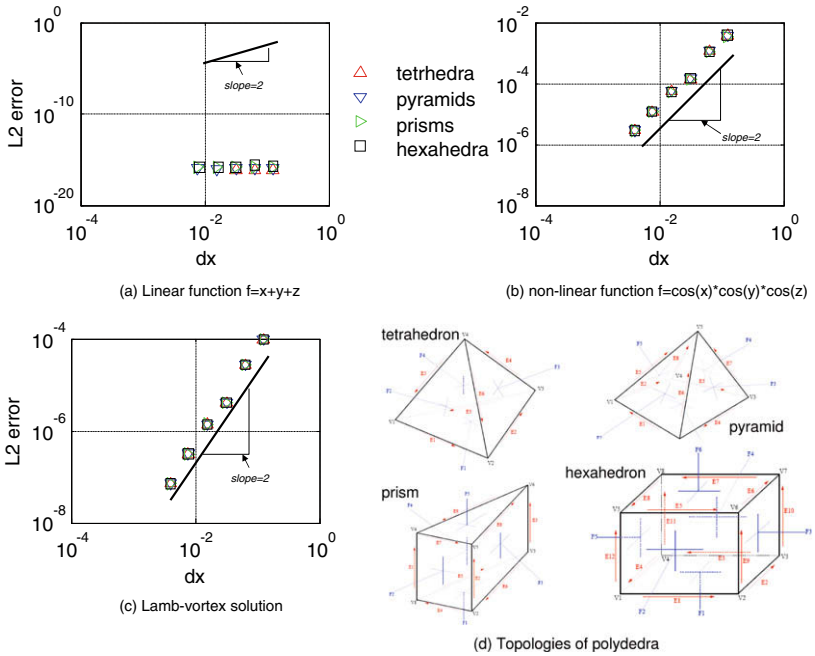

Fig. 8. Variation of interpolation errors for polyhedra shown in subplot (d): subplot (a) shows error variation for a linear test function, subplot (b) shows error variation for a non-linear test function and subplot (c) shows error variation for the Lamb-vortex solution.

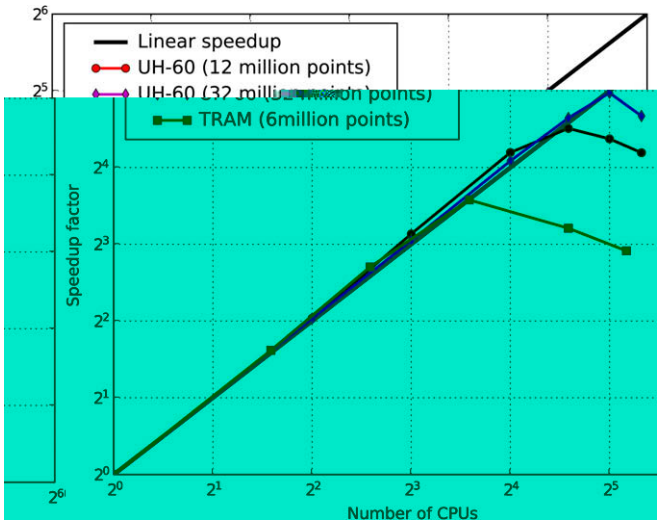

Fig. 9. Scalability of PUNDIT with increasing number of processors for TRAM and UH-60A datasets.

coarse UH-60A grid system is composed of 1 million near-body nodes and 11 million off-body nodes. The fine UH-60A grid system contains 4.6 million near-body nodes and 28 million off-body nodes. More details on these test cases are described in the application results Sections 3.6 and 3.5. Good linear scalability was observed for all three cases up to 12 processors. Beyond that the scalability drops off depending on grid size. For larger mesh systems such as the fine grid for UH-60A, scalability begins to drop-off at about 32 processors, while for the TRAM rotor it starts to drop much earlier at about 12 processors. The reason for scalability drop-off is the load imbalance created by non-optimality of solver based partitioning for domain connectivity procedures. Redistribution of the computational work for searching donors evenly after an initial evaluation of donor processors could be an approach to mitigate this problem (for example the "rendezvous" approach described in [25]).

3.3. Flow Over NACA0015 Wing

The multi-code approach is used to study the flow-field over a NACA 0015 rectangular wing. The primary objective of this computation is to validate the multi-code simulation capability for a high-Reynolds number turbulent external flow problem. The experimental data used for validation are those measured by McAlister et al. [26]. The wing geometry is a rectangular planform with a square tip and an aspect ratio of 3.3. The test point for which validation results are presented is at the following operating condition: $M = 0.1235,\ Re = 1.5e6,\ \alpha = 12^\circ$. For this case, computations are performed in the near-body region using the UMTURNS flow solver using structured curvilinear meshes. The wing is modeled in the full-span configuration. The Spalart–Allmaras turbulence model is used for RANS closure.

Fig. 10 shows the domain connectivity solution obtained for the NACA0015 grids. Sections of the grid system in the X–Y plane and X–Z plane are shown. The implicit domain connectivity algorithm employed not only produces optimal overlap but also ensures commensurate grid spacing of off-body and near-body computational domains at regions of overlap. The pressure distributions obtained from the predictions are compared with measured experimental data in Fig. 11. Fair agreement can be noted between the measurement and prediction at all the span locations, although the leading edge suction peak is consistently under-predicted, This may be attributed to wind-tunnel wall effects which are not modeled in the analysis presently.

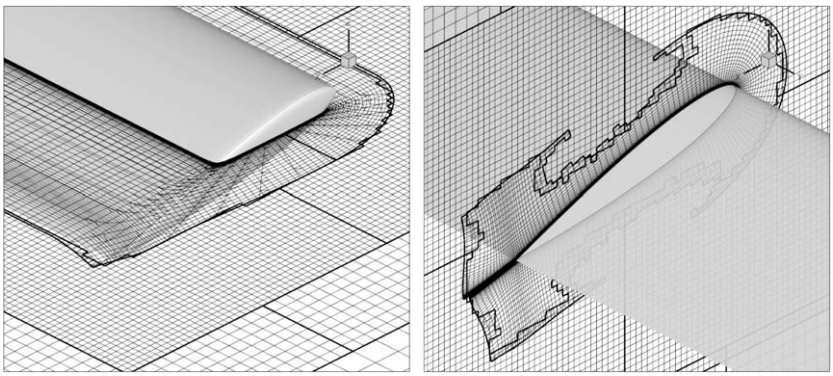

Fig. 10. Domain connectivity solutions with adaptive Cartesian grids for steady flow over a NACA0015 wing at $M = 0.1235,\ \alpha = 12^\circ$ and $Re = 1.5e6$ using HELIOS.

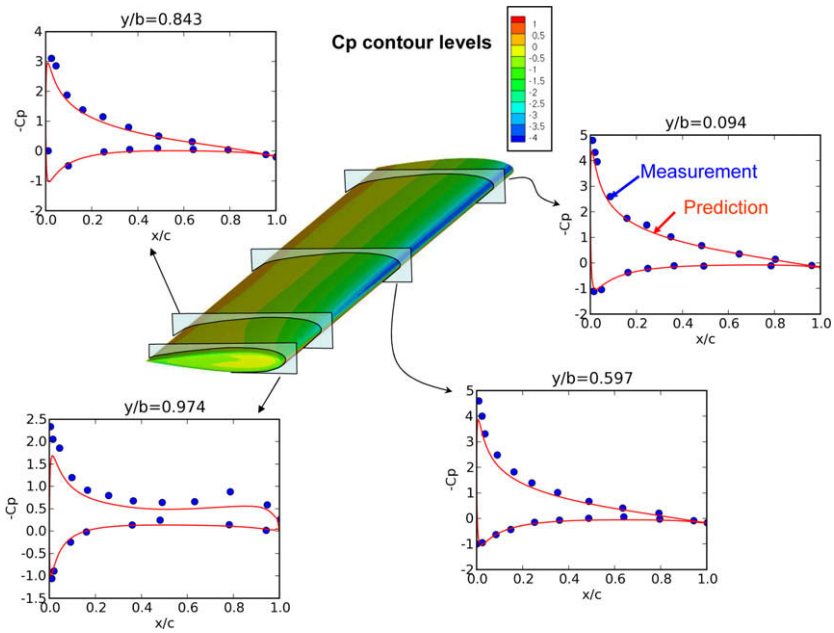

Fig. 11. Prediction of pressure distributions for steady flow over NACA0015 wing at $M = 0.1235,\ \alpha = 12^\circ$ and $Re = 1.5e6$ using HELIOS.

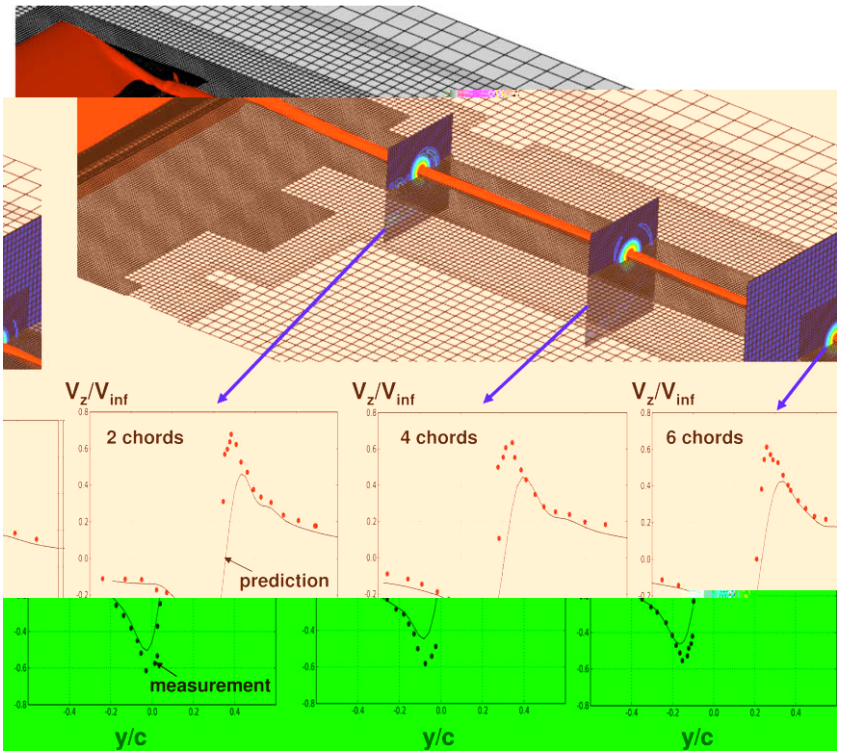

Fig. 12. Prediction of steady flow over NACA0015 wing at $M = 0.1235$, $\alpha = 12^\circ$ and $Re = 1.5e6$ using HELIOS; Predicted tip-vortex trajectory and comparison of predicted swirl velocities with experimental data.

Fig. 12 shows the evolution of the vortex structures from the tip of the wing. The flowfield is dominated by the presence of the vortex sheet which rolls up into a strong tip vortex quite close to the wing. This tip vortex remains coherent up to about 15 chords behind the wing and then begins to break down in to smaller structures. Vorticity magnitude is used as the mesh adaptation criterion for the off-body solver. The meshes represented in the figure show four levels of adaptation. It can be observed that the mesh system is able to track the vorticity and automatically adapt to regions of high vorticity (around the area of tip vortices). The figure also shows comparison of swirl velocities predicted at the tip-vortex locations with measured experimental data. The peak magnitudes are under predicted by about 20% which is consistent with the under-prediction of the leading edge suction peaks. However, the vortex shows minimal diffusion because of numerical dissipation. The use of a 5th-order accurate numerical scheme (6th-order central difference with 5th-order dissipation terms) in the off-body solver in conjunction with adaptive Cartesian grids facilitates the improved resolution of the tip vortex structure. This test case illustrates how the domain-connectivity module can be used with an adaptive grid solver to capture localized flow features in the solution.

3.4. AGARD A2 slotted airfoil test case

This section illustrates the capabilities of PUNDIT for the AGARD A2 (NHLP 2D slotted airfoil) test case in static and dynamic conditions. Although topologically simple, this problem is challenging for computation of domain connectivity because of multiple overlapping meshes in close proximity. In particular, when the flap and slat are fully retracted, there is a very narrow gap between the moving parts and the stationary airfoil body. For this case, there are independent grids for the flap, slat, and main airfoil body, plus a background grid that extends to approximately 10 chords around the airfoil. For this test case, the UMTURNS code was used to calculate the flow solution in all four grids. This slotted airfoil is one of the well-established cases for validation of CFD codes [27,28].

The experimental data for this case are obtained from the electronic supplement of the AGARD 303 report [29,30]. Only static data are available in the supplement. The surface pressure distributions obtained by the computations and their comparison with the experimental data for the 4° angle of attack case are shown in Fig. 13. Excellent agreement can be noted in the suction peaks and chordwise variation of pressure coefficients for all three elements of the slotted airfoil. This validation study provides confidence in the accuracy of the domain-connectivity module.

Fig. 14(a) shows the original overlapping meshes that describe the geometry of the slotted airfoil problem. Fig. 14(b) and (c) shows the detail of the leading and trailing edge section which contain regions which have three overlapping meshes. Finally, Fig. 14(d) and (e) shows the results after performing the domain connectivity operation. In these figures only the

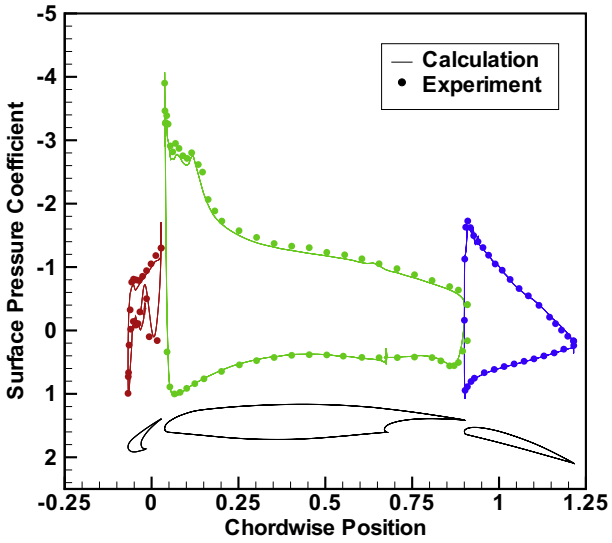

Fig. 13. Prediction of steady 2-D flow over NHLP-2D slotted airfoil at $M = 0.1937$, $\alpha = 4^{\circ}$ and $Re = 3.52e6$ using HELIOS; Comparison of surface pressure distributions to experimental data

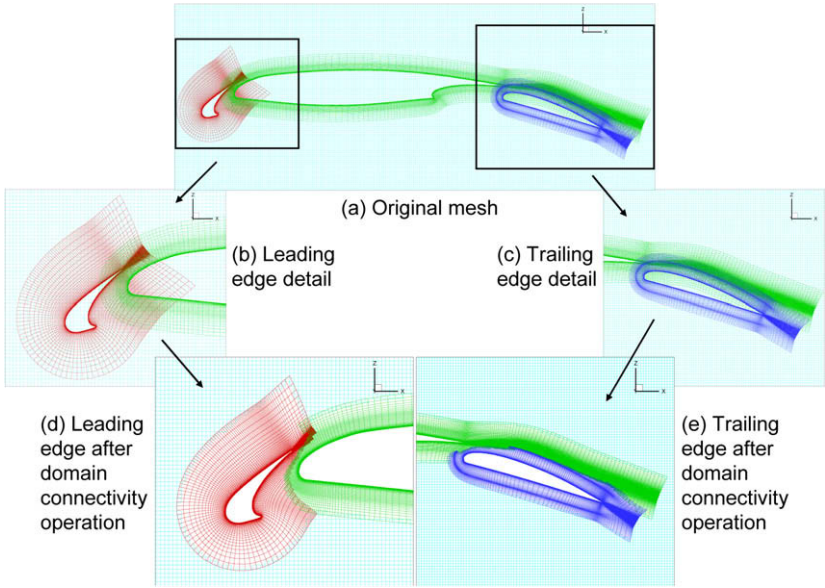

Fig. 14. Domain connectivity operations for NHLP airfoil-flap-slat test case using PUNDIT.

solver points are shown. It can be noticed that at the solver point boundaries the grid resolutions of all overlapping meshes are commensurate with each other which improves the quality of data interpolation. In addition the amount of overlap is optimized, which in turn minimizes the amount of double solving and redundancy in the flow solution.

Fig. 15 shows the contours of streamwise momentum near the leading edge slat and trailing edge flap. The continuity of the contours across the mesh boundaries illustrates the quality of data interpolation achieved by maintaining optimal connectivity. Data interpolation between mesh cells and nodes of differing resolution capacity often causes large, non-physical oscillations in contours of flow variables at grid interfaces.

Following the static computations, calculations were performed under dynamic conditions to illustrate the utility of the domain-connectivity module for moving body problems. The flap and slat were cyclically oscillated at a 1 Hz frequency for two cycles with 8000 time steps per cycle. Although validation data for the flow solution are not available for the dynamic case, the performance of the domain connectivity software can be qualitatively shown. This type of problem demonstrates the power of the implicit hole cutting procedure. For the computation, the flap and slat grids were translated and rotated

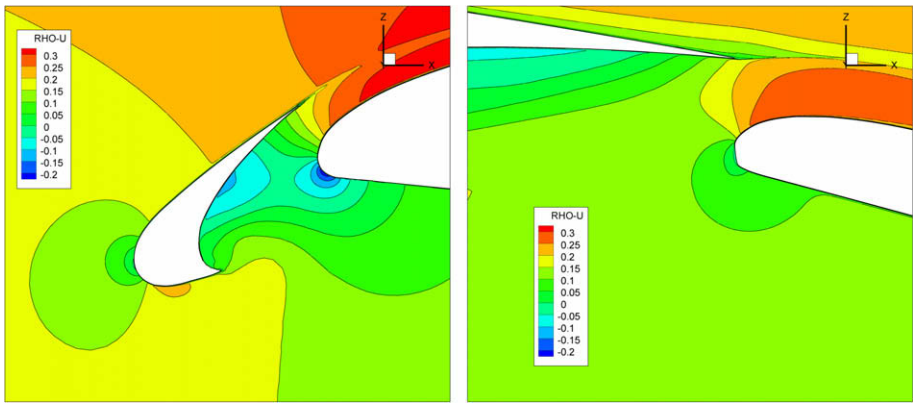

Fig. 15. x-momentum contours of flow solutions for NHLP airfoil-flap-slat test case using PUNDIT and UMTURNS; conditions are $M = 0.197$, $\alpha = 4^\circ$, $Re = 3.52e6$

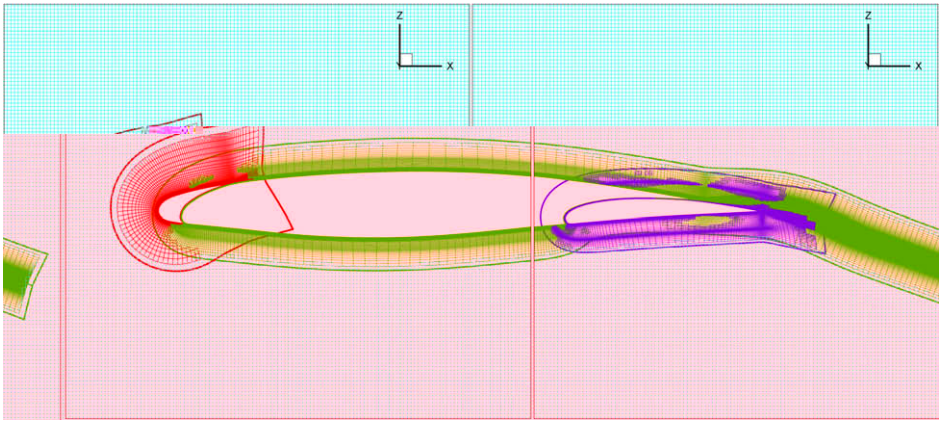

Fig. 16. AGARD A2 airfoil grids with flap and slat fully retracted after domain connectivity operation. The thick lines represent the boundaries of the grids before blanking.

incrementally at each time step and the domain connectivity re-calculated with the new grid coordinates. The airfoil body and background meshes were stationary. As the flap and slat approach the airfoil body, an increasing number of grid points become hole points and background points become solver points. When fully retracted, the gaps between the airfoil body and the control elements are very thin, but PUNDIT is able to perform the hole cutting reliably and automatically without any input from the user, see Fig. 16. Explicit hole cutting for a complex shape with such a tight gap would be challenging and would require significant user expertise and effort.

For a sinusoidal oscillation, the velocity of the flap and slat are highest when they are 50% deployed. Fig. 17 shows the streamwise momentum solution for the airfoil with the flap and slat in the same position while retracting and deploying. The large orange and red areas on the upper surface represent the flow accelerating over the airfoil[1]. Because of the lag in the unsteady lift, the air is moving faster over the airfoil as the elements retract than as they deploy.

3.5. Hover simulation of TRAM rotor

Hover simulation for the TRAM (Tilt Rotor Aeroacoustics Model, a 1/4th scaled V-22) rotor was performed using the HELIOS infrastructure with PUNDIT for domain connectivity. The near-body solver used for this simulation is the unstructured NSU3D code.

Fig. 18 illustrates the mesh system and connectivity solution and Fig. 19 shows flow solution obtained for this test case. This test case demonstrates the capability of PUNDIT to handle multiple unstructured meshes.

The TRAM mesh system consists of three unstructured grid blocks generated by rotating and combining a single blade unstructured mesh. PUNDIT performs domain connectivity between these grid blocks first and determines the resolution

[1] For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.

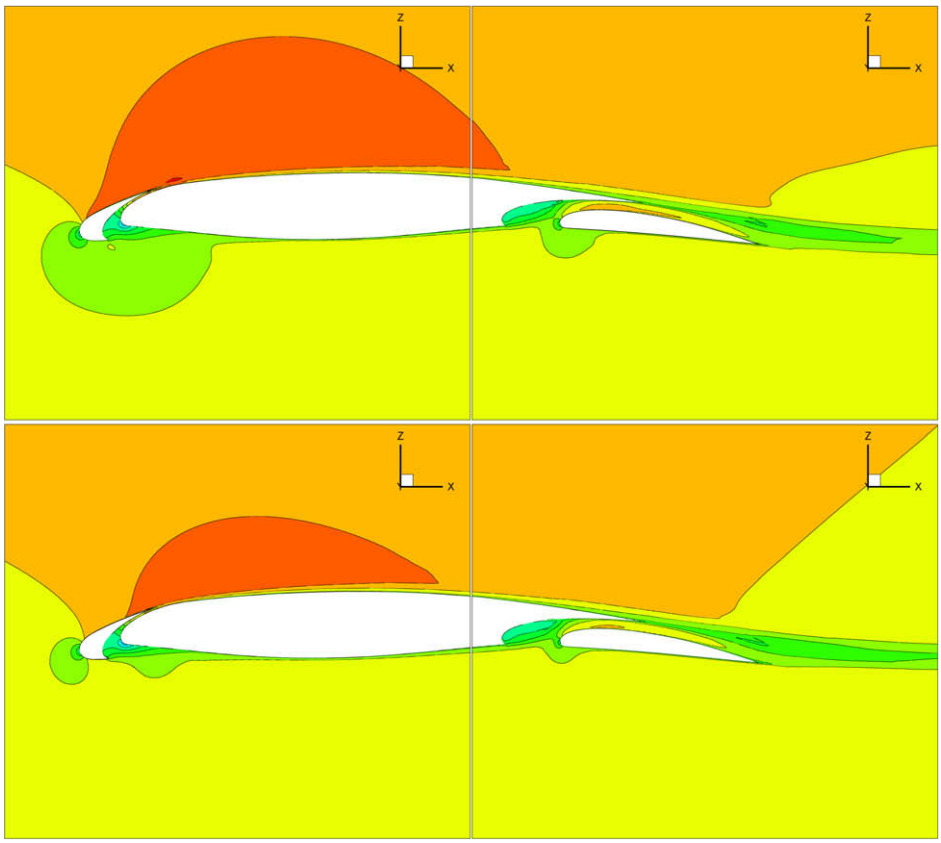

Fig. 17. AGARD A2 airfoil solutions with the flap and slat in mid-motion. In the upper figure, the elements are retracting; in the lower figure, they are deploying.

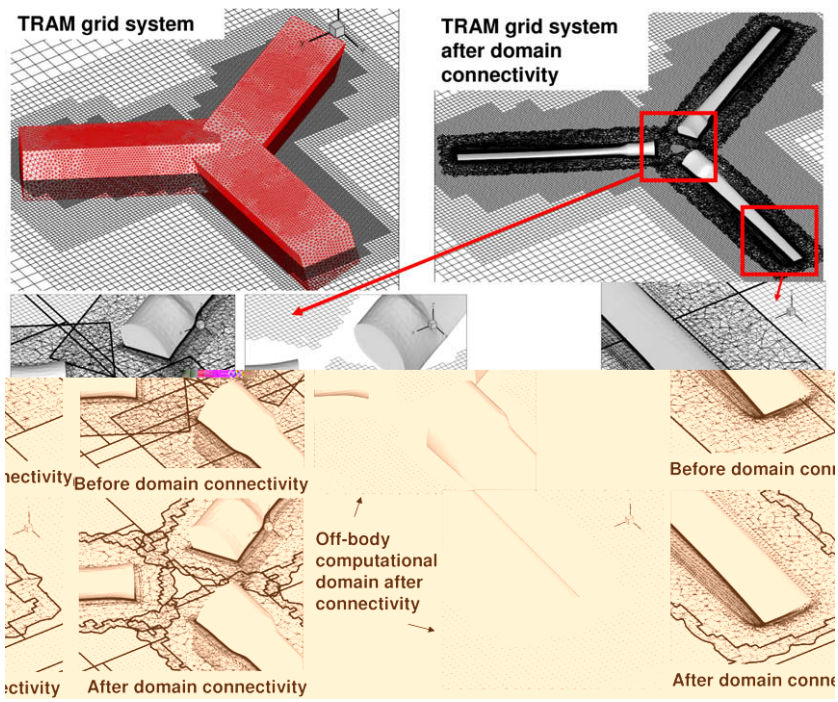

Fig. 18. Mesh system and domain connectivity for 1/4th scaled V-22 (TRAM) rotor using HELIOS; Plots illustrate grid overlap and hole geometry.

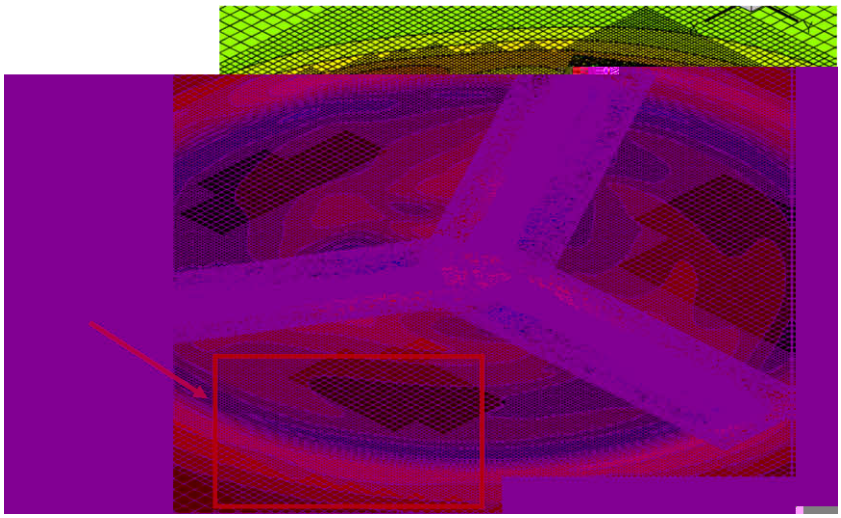

Fig. 19. Flow solutions for 1/4th scaled V-22 (TRAM) rotor using HELIOS; Plot illustrates dynamic mesh adaptation to flow features.

and geometry requirements for the off-body meshes. The off-body mesh generation is performed subsequently such that the resolution of the grid cells near the outer boundaries of near-body meshes is commensurate with the off-body grid cells in that region. Consequently PUNDIT performs the domain connectivity solution between the off-body and near-body systems and determines the off-body and near-body computational domains. The off-body computational domains are shown overlapped with near-body meshes to illustrate the extent of overlap created. In addition the off-body domains are shown by themselves to illustrate the hole geometry created by the implicit hole-cutting procedure. Overall, it can be observed that PUNDIT is capable of producing optimal domain connectivity solutions for the mesh system used in this case, even at the center of the rotor where there are multiple overlap regions. It is worth noting that an explicit hole cutting methodology needs sufficient user intervention to make the appropriate hole cuttings for a mesh system such as this, especially at the overlap regions.

The flow solution obtained from the hover simulation, shown in Fig. 19, illustrates the dynamic adaptation of the off-body grid system to the tip vortex structure. The solution slice shows contours of downwash momentum superimposed on the grid system. Large gradients of downwash indicate a larger presence of vorticity as observed in the tip regions where the mesh density is automatically increased. Note that domain connectivity is performed on the fly in this simulation after each off-body grid adaptation step.

3.6. Moving mesh computations for rotor aeromechanics

The last case shown is the moving mesh problem for the UH-60A helicopter in forward flight. The meshes used for this problem are shown Fig. 20. Fig. 20(a) shows the grid system with a curvilinear C–O type topology (near-body) overlapped by a Cartesian mesh (off-body). The flow solutions for the near-body grids are performed by the UMTURNS code and those for the off-body grids are performed by the SAMARC code. Fig. 20(b) and (c) shows sections along the $z = 0$ plane and $x = 0$ plane for the entire grid system illustrating the computational domains (solver points) in the near- and off-body regions. The problem includes grid deformation to account for blade elasticity as well as rigid body rotation of the near-body meshes around the rotor shaft axis. Prescribed elastic deformations which are obtained from previous CFD/CSD computations [31] are used for verification of the methodology. Since the near-body meshes are moving, the domain connectivity has to be performed at every time step. The domain connectivity procedures required about 5% of the total time taken for a time step. Computations were conducted on 32 processors, which is the scalability limit of PUNDIT for this problem.

Fig. 20(d) shows the contours of the z-momentum (downwash) for the flow solution obtained. The vorticity shed from the rotor blades induces the downwash observed in the figure. During the advancing azimuth the aerodynamic loading towards the tip of the blades becomes negative causing them to produce much less downwash compared to the retreating side where the aerodynamic loading is positive. The downwash distribution is also an indicator of the vortex wake geometry shed from the rotor blade; large gradients indicate the presence of strong vortex structures. A comparison of aerodynamic loading (sectional lift and pitching moment at 86% radial station) is shown in Fig. 20(e) which compares favorably with the measured flight test data [32].

4. Performance data for test cases presented

Table 2 summarizes the performance data of the domain connectivity procedure for the test cases that were presented in this paper. The *time* presented in the table represents the actual wall-clock time that elapsed between different operations,

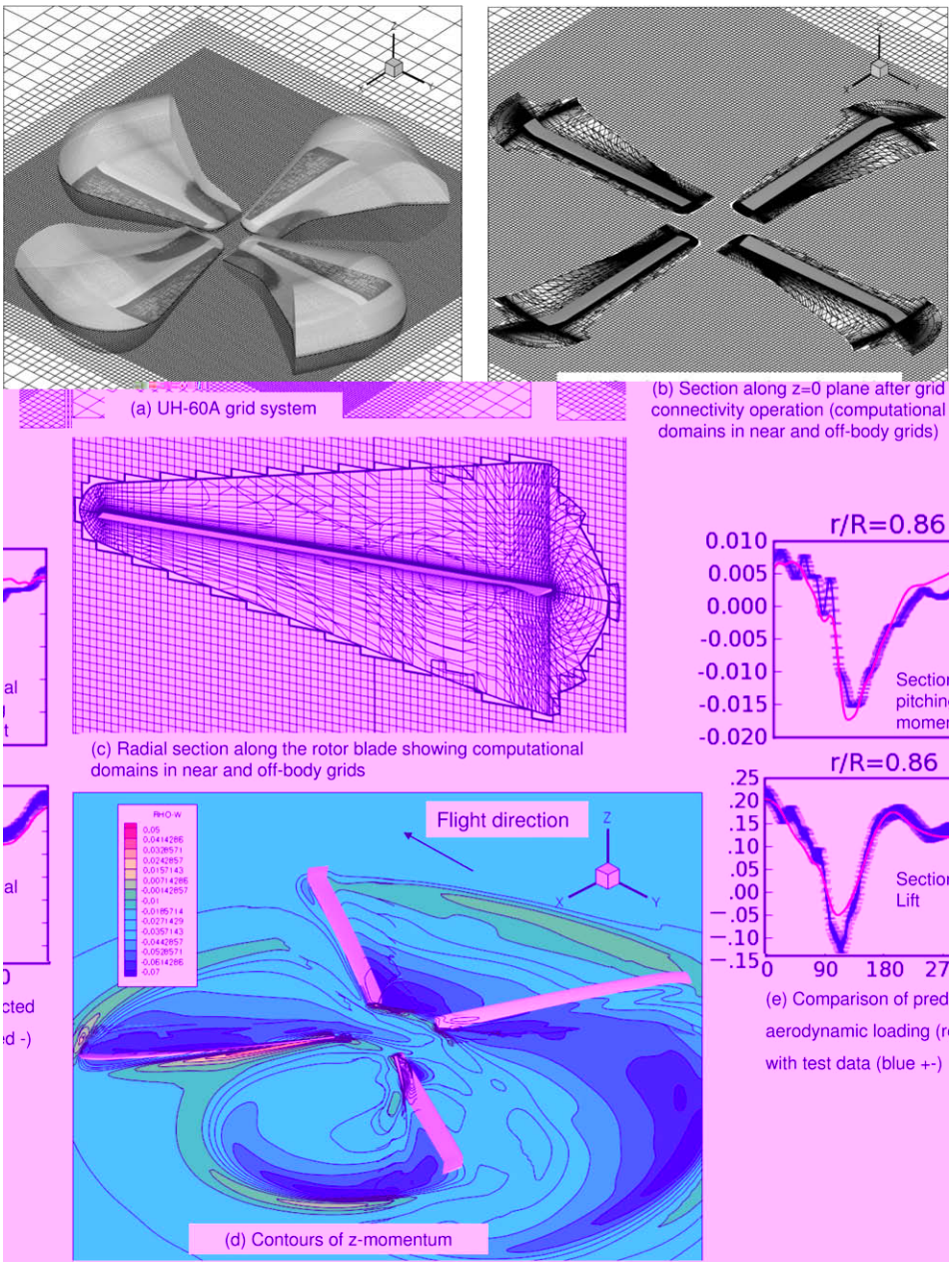

Fig. 20. Mesh system, domain connectivity and flow solution for aeroelastic simulation of UH-60A rotor using HELIOS; Plots illustrate grid overlaps, hole geometry and comparison with measured flight test data.

such as domain connectivity procedures and near and off-body flow solutions. Time accurate computations were performed for all these cases and timings are provided for one time step of unsteady flow solution. In all the cases shown, twenty sub-iterations were used in the near-body solver to reduce the unsteady flow residual by two orders of magnitude compared to its initial value. The off-body flow solution is on average an order of magnitude faster than the near-body flow solution. This is primarily because of two reasons: (1) lean cache-friendly data structure and flow solution procedures for isotropic Cartesian grids and (2) explicit time stepping (using 3-stage Runge–Kutta which does not require any sub-iterations or matrix inversion).

The data does indicate a weak correlation with total domain connectivity time and total number of grid points. The nature of partitioning and grid overlap strongly contribute to the performance of domain connectivity procedures. Therefore it is often difficult to identify global trends between CPU time and number of grid points. The computational overhead metric (calculated as fraction of domain connectivity time/total CPU time) shows encouraging trends and is less than 10% for all test cases.

Table 2
Performance data for the four test cases presented in the paper. Detailed timings are provided for the domain connectivity procedure. The key for domain connectivity timings are as follows, *NB*: near-body, *OB*: off-body, *nbpr*: near-body preprocessing time, *nbdc*: near-body to near-body connectivity time, *obpr*. off-body preprocessing time, *obdc*: near-body to off-body connectivity time, *interp*: data interpolation time

Case	Number of processors	Number of grid nodes		Number of fringes and holes		Time taken in seconds per time step for overset grid assembly	Time taken in seconds for one time step of flow solution		Computational overhead
		NB	OB	NB	OB		NB	OB	
Slotted airfoil	4	128,327	263,169	Fringes: 12,431 holes: 1900	Fringes: 9346 holes: 4745	nbpr: 0.54 nbdc: 1.14 obpr: 0.005 obdc: 0.32 interp: 0.01 total: 2.015	17.63	2.12	9.26%
1/4 scaled V-22 (TRAM)	12	763,965	5,120,710	Fringes: 35,586 holes: 0	Fringes: 33,069 holes: 1725	nbpr: 0.28 nbdc: 0.25 obpr: 0.005 obdc: 0.25 interp: 0.02 total: 0.805	49.31	3.38	1.5%
NACA 0015	16	2,470,737	5,649,812	Fringes: 147,569 holes: 0	Fringes: 404,526 holes: 50,964	nbpr: 0.70 nbdc: 0.14 obpr: 0.01 obdc: 1.66 interp: 0.04 total: 2.55	44.43	3.54	5.05%
Blackhawk UH-60	32	457,5256	28,007,666	Fringes: 85,378 holes: 0	Fringes: 264,771 holes: 7450	nbpr: 0.53 nbdc: 1.11 obpr: 0.014 obdc: 1.81 interp: 0.06 total: 3.52	101.48	7.16	3.15%

5. Resolution capacity modification

As stated in Section 2.5, the nominal resolution capacity that determines the solution quality at a given point in a given grid is based on the cell volume. For a cell, it is the volume of that cell, and for a point, it is the average of the volumes of the cells which share that point. In a purely mathematical sense, this provides the best solution by selecting closely spaced points over more sparsely spaced points. However, physically, the points with the closest spacing may not be the points which should be solved. Two particular instances were observed where the default selection process was not selecting the correct points. Both can be illustrated by the slotted airfoil case presented in Section 3.4.

The first case is near a solid wall. For overset grids, only the solver assigned to the grid block which contains the wall has information that the wall is present. Processors solving background grid blocks do not have the wall information from the near-body grids, and each near-body grid does not have information about walls in other near-body grids. Wall distance is important for turbulence modeling in particular, so it is important that points near a solid wall are solved in the grid which contains the wall.

In the second case, a wall may not be involved but an "island" can form where one grid is locally more refined than another. Computationally, it may be important for solution points to be contiguous rather than being broken up by receiver points which get data from other processors. Localized areas of dense points normally occur because of the proximity of a wall, so often the first issue is the cause of the second.

In order to address these issues, a wall correction scheme was investigated. A scaling factor ξ was applied to artificially promote the resolution capacity of cells near a solid wall. The scaling factor is identically 1.0 for grid blocks which do not contain any wall nodes and varies from 0 to 1 in grid blocks which do have wall nodes. This is performed by calculating ξ for all nodes, and then non-dimensionalizing by $\xi_{\max}$ for the grid block so all values of ξ fall between 0 and 1.

The first method was a linear scale factor. As the resolution capacity is being calculated, the distance to the nearest wall node, x_w is calculated, and the scale factor was determined simply as

$$\xi = x_w \tag{1}$$

For this case, the $\xi_{\max}$ is the distance of the boundary node farthest from a wall in the grid block. A failure mode of this approach is seen with the AGARD A2 airfoil grids. The grid for the main airfoil body extends far beyond the trailing edge compared to the grids for the flap and slat. Since the scale factor is relative, the farthest point from the wall, the non-dimensionalized ξ is smaller for the main body than for the flap and slat for the same distance from the wall.

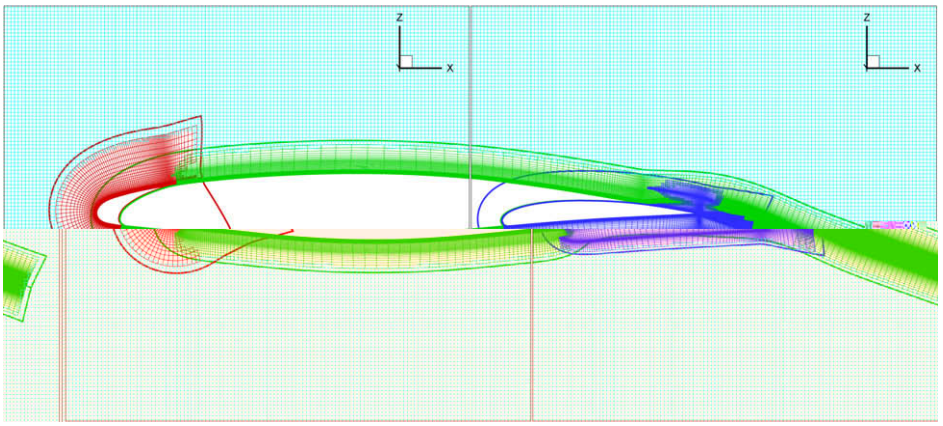

Fig. 21. AGARD A2 airfoil grids with flap and slat fully retracted after domain connectivity operation with wall distance correction. The thick lines represent the boundaries of the grids before blanking. Grid scattering and formation of islands noted in Fig. 16 are observed to be substantially mitigated.

For the general case where grids can be heterogeneous and can contain complex shapes or protuberances, ξ should reflect the proximity to the wall relative to the local vicinity of the grid. To accomplish this, for each point, the shortest distance to an outer boundary x_o is also determined. With both the minimum distances to a wall and to an outer boundary, ξ was determined as

$$\xi = \left(\frac{x_w}{x_w + x_o}\right)^2 \tag{2}$$

Although the nearest wall point and the nearest outer point are not generally along the same line, this ensures that ξ approaches 0 near the wall and approaches 1 near the outer boundary. And for the case of the airfoil body mentioned above, the resolution capacity of points near the body are not excessively modified because the grid extends so far beyond the trailing edge. For this investigation, the ξ values are still divided by $\xi_{\max}$ even though it is close to unity. This ensures that the maximum scale factor for points at the outer boundary of grid blocks is exactly 1 to match that of background grid blocks. By squaring the term in parentheses in Eq. (2), points near the wall are assigned a very small resolution capacity, so they are solved in the local grid with an appropriate wall distance for the turbulence model.

The grids from Fig. 16 are shown with the wall distance correction in Fig. 21. Comparing the two figures, the effect of wall distance is clear. The small "island" above the slat is eliminated as the points in the slat grid are promoted to higher resolution capacity in the vicinity of the slat surface. The slat grid extends around the bottom of the slat and transitions to the airfoil body grid at the gap. Similarly, for the transition to the flap grid on the lower surface, the airfoil body grid extends to the gap and the flap grid is used for points through the trailing edge of the flap without the "island" of airfoil body grid in Fig. 16. The airfoil body grid is very dense along its centerline behind the trailing edge of the airfoil body, but the transition from the airfoil body grid to the flap grid on the upper surface of the flap is still smooth, with enough points retained in the flap grid near the surface to resolve the boundary layer for this high Reynolds number flow. This is a particularly challenging case because both bodies are in such close proximity and the upper surface of the flap is so near a dense region of the airfoil body grid.

Note that the correction represented in Eq. (2) is intended to be a proof of concept only and has not been evaluated beyond the AGARD A2 test case. It was investigated to determine if the cell and point resolution capacities could be modified in a simple way to obtain a computationally better selection of grid points. As the code matures and the base of users and applications increases, a more appropriate algorithm may be developed.

6. Summary and conclusions

The potential of the multi-solver paradigm and automated domain connectivity in resolving flow physics in unsteady problems which involve moving bodies has been demonstrated. The domain-connectivity module (PUNDIT) shows encouraging trends in obtaining optimal domain connectivity, improving interpolation accuracy and automated parallel execution. The improvements that PUNDIT brings over the existing domain connectivity technology can be categorized into three areas:

1. *Automation*: There is no user input or intervention required for PUNDIT, which makes it completely automated and shows potential for large productivity gains. Existing technologies often require high-levels of user expertise because of the need to perform explicit hole cutting.
2. *Scalability*: None of the existing technologies provides scalable execution for all the domain connectivity procedures. We have demonstrated linear-scalability of execution if at least 1 million points are available per processor. However, this is still quite short of meeting future production demands of highly-scalable execution. The main reason for the drop-off in

scalability is because of solver-based mesh-partitioning, which is non-optimal for domain connectivity. This is because only a subset of processors will have intensive domain connectivity operations as the grid system becomes more finely partitioned, leaving the overall solution procedure unbalanced. Algorithms for redistribution of search load on the fly are envisioned to mitigate the load imbalance.

3. *Arbitrary element types and relative motion capability*: PUNDIT supports multiple solvers (both intra-code and inter-code) and multiple mesh types and performs all the domain connectivity operations in a distributed computing platform for moving body problems. Existing technologies support many of these features but none of them support all of the features together.

Following are the summaries of the specific observations that pertain to the various application test cases that were studied:

1. The NACA0015 test case shows the capability of adaptive meshes and high-order solvers to preserve the coherence of vortex structures. Comparison with experimental data shows under-prediction of peak swirl velocity magnitudes. However these are attributed to the lift deficiency observed from the pressure distributions. Comparison with experimental data is likely to improve with the inclusion of wind-tunnel wall modeling.
2. The AGARD A2 airfoil-flap-slat test case shows the ability of PUNDIT to obtain optimal domain connectivity and facilitate smooth solution interpolation. Excellent correlation obtained with experimental data for the pressure distributions provides validation for the methodology.
3. Studies for the TRAM rotor show the capability of automated mesh adaptation and domain connectivity solution for a rotorcraft problem in a parallel processing environment.
4. Application to the aeroelastic simulation of a UH-60A helicopter further illustrates the capability of dynamic domain connectivity management in a parallel processing environment with reasonable cost.
5. Promoting the resolution capacity of grid points near solid walls can be used to obtain a more computationally effective grid. For the AGARD A2 airfoil test case, application of the wall boundary correction resulted in additional points being retained in the grid containing the solid wall.

Acknowledgments

Development was performed at the HPC Institute for Advanced Rotorcraft Modeling and Simulation (HIARMS) located at the US Army Aeroflightdynamics Directorate at Moffett Field, CA, which is supported by the Department of Defense High Performance Computing Modernization Office (HPCMO). The authors would like to acknowledge Dr. Robert Meakin for many of the original ideas and critical concepts that have been used in the development of the domain-connectivity module. We also gratefully acknowledge the contributions of Dr. Venke Sankaran in reviewing the manuscript, Dr. Willam Chan for providing test cases and insights, Dr. Buvana Jayaraman for verifying the robustness of the simulation capability and Dr. Beatrice Roget for discussions on search algorithms. In addition, we are grateful to Dr. Thomas Pulliam for the development of the higher-order Cartesian-mesh solver (ARC3DC). We also thank Dr. Dimitri Mavriplis and Dr. Zhi Yang at the University of Wyoming for providing the NSU3D code.

References

[1] A. Wissink, J. Sitaraman, D. Mavriplis, T. Pulliam, V. Sankaran, A python-based infrastructure for overset cfd with adaptive cartesian grids, in: AIAA 48th Aerospace Sciences Meeting, No. AIAA-2008-0927, AIAA, Washington, DC, 2008.
[2] J. Sitaraman, A. Katz, B. Jayaraman, A. Wissink, V. Sankaran, Evaluation of a multi-solver paradigm for cfd using overset unstructured and structured adaptive cartesian grids, in: AIAA 48th Aerospace Sciences Meeting, No. AIAA-2008-0660, AIAA, Washington, DC, 2008.
[3] Z. Yang, D. Mavriplis, Higher-order time integration schemes for aeroelastic applications on unstructured meshes, in: AIAA 44th Aerospace Sciences Meeting, No. AIAA-2006-0441, AIAA, Washington, DC, 2006.
[4] J. Sitaraman, J.D. Baeder, Evaluation of the wake prediction methodologies used in cfd based rotor airload computations, in: AIAA 24th Conference on Applied Aerodynamics, No. AIAA-2006-3472, AIAA, Washington, DC, 2006.
[5] R.L. Meakin, N.E. Suhs, Unsteady aerodynamic simulation of multiple bodies in relative motion, in: AIAA 9th Computational Fluid Dynamics Conference, No. AIAA-89-1996-CP, AIAA, Washington, DC, 1989, pp. 643–657.
[6] R.L. Meakin, Object X-rays for cutting holes in composite overset structured grids, in: AIAA 15th Computational Fluid Dynamics Conference, No. AIAA 2001-2537, AIAA, Washington, DC, 2001.
[7] P.G. Buning, D.C. Jesperson, T.H. Pulliam, W.M. Chan, J. Slotnick, S.E. Krist, K.J. Renze, Overflow users manual, Research report, NASA Langley Research Center, 1998.
[8] N.A. Petersson, An algorithm for assembling overlapping grid systems, SIAM J. Sci. Comp. 20 (1999) 1995–2021.
[9] W.D. Henshaw, D.L. Brown, D.J. Quinlan, Overture: Object-oriented tools for overset grid applications, in: AIAA 17th Conference on Applied Aerodynamics, No. AIAA 1999-3130, AIAA, Washington, DC, 1999.
[10] G.S. Chesshire, W.D. Henshaw, Composite overlapping meshes for the solution of partial differential equations, JCP 90 (1) (1990) 1–64.
[11] W.D. Henshaw, Ogen: an overlapping grid generator for Overture, Research Report LA-UR-96-3466, Los Alamos National Laboratory, 1996.
[12] S.E. Rogers, N.E. Suhs, W.E. Dietz, Pegasus 5: an automated preprocessor for overset-grid computational fluid dynamics, AIAA J. 41 (6) (2003) 1037–1045.
[13] Y. Lee, J.D. Baeder, Implicit hole cutting – a new approach to overset grid connectivity, in: AIAA 16th Conference on Computational Fluid Dynamics, No. AIAA 2003-4128, AIAA, Washington, DC, 2003.
[14] Y. Lee, On overset grids connectivity and vortex tracking in rotorcraft cfd, Ph.D. Thesis, University of Maryland, College Park, Maryland, 2008.

[15] R.W. Noack, Suggar: A general capability moving body overset grid assembly, in: AIAA 17th Computational Fluid Dynamics Conference, No. AIAA 2005-5117, AIAA, Washington, DC, 2005.
[16] J.J. Alonso et al., Chimps: a high performance scalable module for multi-physics simulations, in: AIAA/ASME/SAE 42nd Joint Propulsion Conference, No. AIAA 2006-5274, AIAA, Washington, DC, 2006.
[17] D.M. Belk, R.C. Maple, Automated assembly of structured grids for moving body problems, in: AIAA 12th Computational Fluid Dynamics Conference, No. AIAA 1995-1680, AIAA, Washington, DC, 1995, pp. 381–390.
[18] Z.J. Wang, V. Parthasarathy, N. Hariharan, A fully automated chimera methodology for multiple moving body problems, in: AIAA 36th AIAA Aerospace Sciences Meeting, No. AIAA 1998-217, AIAA, Washington, DC, 1998.
[19] R.D. Hornung, A.M. Wissink, S.R. Kohn, Managing complex data and geometry in parallel structured amr applications, Eng. Comp. 22 (3) (2006) 181–195.
[20] J.J. Alonso, P. LeGresley, E. Van Der Weide, pymdo: a framework for high-fidelity multi-disciplinary optimization, in: AIAA/ISSMO 10th Conference on Multidisciplinary Analysis and Optimization, No. AIAA 2004-4480, AIAA, Washington, DC, 2004.
[21] T.H. Pulliam, Euler and thin-layer navier-stokew codes: Arc2d and arc3d, in: Computational Fluid Dynamics Users Workshop, Tullahoma, Tennesse, 1984.
[22] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, JCP 82 (1) (1989) 65–84.
[23] R.L. Meakin, A.M. Wissink, W.C. Chan, S.A. Pandya, J. Sitaraman, On strand grids for complex flows, in: AIAA 18th Conference on Computational Fluid Dynamics, No. AIAA 2007-3834, AIAA, Washington, DC, 2007.
[24] J. Sitaraman, M.W. Floros, A.M. Wissink, M. Potsdam, Parallel unsteady overset mesh methodology for a multi-solver paradigm with adaptive cartesian grids, in: AIAA 26th Conference on Applied Aerodynamics, No. AIAA 2008-7177, AIAA, Washington, DC, 2008.
[25] S.J. Plimpton, B. Hendrickson, J.R. Stewart, A parallel rendezvous algorithm for interpolation between multiple grids, J. Parallel Distributed Comput. 64 (2) (2004) 266–276.
[26] K.W. McAlister, R.K. Takahashi, Naca 0015 wing pressure and trailing vortex measurements, Research Report NASA TP 3151, NASA Ames Research Center, 1991.
[27] A. Elsenaar et al., A selection of experimental test cases for the validation of cfd codes volume 1, Research Report AR 303, AGARD, Advisory Group for Aerospace Research and Development, 7 Rue Ancelle, 92200 Neuilly-Sur-Seine, France, 1994.
[28] M. Burt, Summary of test cases, in: Selection of Experimental Testcases For the Validation of CFD Codes Volume 1, no. Chapter 5, AGARD, Neully-Sur-Seine, France, 1994, pp. 55–133.
[29] I.R.M. Moir, Measurements on two-dimensional aerofoil with high-lift devices, in: Selection of Experimental Testcases For the Validation of CFD Codes Vol. 2. AGARD, Neully-Sur-Seine, France, 1994, pp. A2–1–A2–12.
[30] A.G. for Aerospace Research, Development, A selection of experimental test cases for the validation of cfd codes, supplement: datasets a–e, No. AGARD-AR-303-SUPPL, Electronic Data, AGARD, Neully-Sur-Seine, France, 1994.
[31] M. Potsdam, H. Yeo, W. Johnson, Rotor airloads prediction using loose aerodynamic/structural dynamic coupling, J. Aircraft 43 (3) (2006) 732–742.
[32] R.M. Kufeld, D.L. Balough, J.L. Cross, K.F. Studebaker, C.D. Jennison, W.G. Bousman, Flight testing the UH-60A airloads aircraft, in: American Helicopter Society 50th Annual Forum Proceedings, AHS International, Washington, DC, 1994, pp. 557–577.