Extension of CHIMPS for unstructured overset simulation and higher-order interpolation

By S. Hahn, G. Iaccarino, S. Ananthan† AND J. D. Baeder‡

1. Motivation and objectives

Rapid progress of computational science and engineering has allowed our technical knowledge to mature in individual areas of science over the past few decades. Nevertheless, many of the complex interactions between two or more elementary phenomena, such as multi-phase flows, fluid–structure interactions, or reactive flows, remain poorly understood. Analysis of multi-disciplinary large-scale systems has emerged as an important task of modern computational science and engineering.

To address this kind of problem, it is more advantageous to build a flexible integration infrastructure in which several independent solvers can be easily coupled with one another, rather than to newly implement all the necessary functionalities and physical/numerical models onto a single code. The reason is twofold: From a physical and numerical point of view, different physical phenomena often involve drastically different ranges of eigenvalues, in which case it is prohibitively difficult to develop a universally efficient numerical method to account for all relevant physics. Therefore, coupling individual solvers best suited for each purpose would be the most effective way to obtain the solution. On the other hand, from a technical point of view, the ability to easily integrate individual solvers ensures that new features or updated models can be included with reasonable effort without disrupting the entire structure. An additional benefit of this strategy is that various existing component modules or solvers can be rapidly combined to solve new problems without a tremendous overhead to create a new environment from scratch.

A majority of multi-physics-coupled applications concerns a situation where different physical phenomena occur locally in space. Two typical examples among the recent studies conducted at Stanford University are the simulations for an entire jet engine (Medic *et al.* 2007) and for rotor blades (Hahn *et al.* 2006). In the former example, transonic flows in the compressor and turbine are solved by a structured multi-block compressible URANS code, whereas the mixing of low-speed air with liquid fuel from the intricate injector passages and the subsequent chemical reactions are solved by an unstructured reactive large-eddy simulation (LES) solver. In the latter example, by contrast, near-blade transonic flows are solved using a compressible URANS solver, whereas the highly vortical wake region is solved by an energy-conserving incompressible finite-volume URANS solver to better preserve tip vortices. In these cases, code-to-code coupling is realized by each solver receiving data at the domain boundary from a domain of another solver and imposing these data as a boundary condition. Since the grids of each participating solver do not necessarily match in general, search and interpolation are the essential elements for integrated simulations. In modern computing environments, participating codes are usually domain-decomposed onto a very large number of processors, and the data to

† Postdoctoral research fellow, University of Maryland.
‡ Professor, University of Maryland.

be interpolated can reside anywhere in the distributed memory of a parallel computer. Search and information exchange under these circumstances have often been referred to in the literature as the $M \times N$ problem (Plimpton *et al.* 2004), and an efficient and scalable (both in CPU time and memory usage) solution to the $M \times N$ problem is not straightforward.

To address the issues stated above and hence facilitate a multi-code integration process, Stanford University has developed the software named CHIMPS (Coupler for High-performance Integrated Multi-Physics Simulations; Alonso *et al.* 2006), which automates the search, interpolation, communication, and data/information transfer based on the general unstructured-mesh format in a user-friendly and scalable manner. In this paper, we describe two newly extended capabilities of CHIMPS which have been recently implemented: One is the full support for the hole-cutting process for unstructured overset-mesh simulations in a parallel environment, and the other is a higher-order interpolation on unstructured meshes. Although our primary interest currently lies in the rotorcraft applications, we believe this implementation is readily applicable to unstructured overset simulations in any area.

In this article, we describe the overset hole-cutting algorithm and its validation in sections 2 and 3, respectively, and the higher-order interpolation algorithm is provided in section 4.

2. Hole cutting for overset-mesh simulation on unstructured meshes

In an overset-mesh simulation, the entire domain is decomposed into several overlapping patches or blocks, and the meshes are generated independently on each of them. Ever since it was first introduced two decades ago by Steger *et al.* (1983), the overset-mesh computation has become increasingly popular as an efficient technique for complex geometry. Meanwhile, a coupled simulation using CHIMPS has a close similarity to an overset-mesh simulation in the sense that the interface conditions in one block are sought and interpolated from other blocks in the entire computational domain. This is our main motivation in using CHIMPS as a main coordinator of multi-code coupled overset-mesh simulations. Regarding the extension of CHIMPS to support overset-mesh simulations, our particular interest lies in the generalization for both static and moving unstructured meshes, robust automation to minimize human intervention, and efficiency both in speed and scalability.

Two important preprocessing steps related to an overset-mesh calculation are the hole cutting and donor-cell identification. Hole cutting is a process to identify the interior and exterior of given solid-body or hole surfaces on the volume meshes of each block. The final goal of hole cutting is to provide an integer array i_b (called iblank), which has the values, for example, 1 and 0 for interior and exterior of hole surfaces. Mathematically, it is identical to the judgment whether a given point lies inside a given polyhedron (Milgram 1989), and several algorithms have been suggested so far for efficient hole cutting: In the analytic shape method (Meakin 1991), the entire solid-body surfaces are first expressed as a collection of several analytical shapes, a process which is known to be efficient but hard to automate and requires considerable expertise. Later, Chiu & Meakin (1995) suggested the Cartesian hole-map method, which utilizes the efficiency of "search by truncation" from inverse maps. Wang & Parthasarathy (2000) considered hole cutting for general unstructured meshes based on the ADT structure. Meakin (2001) developed the object X-ray method, which stores only two-dimensional arrays for hole maps and

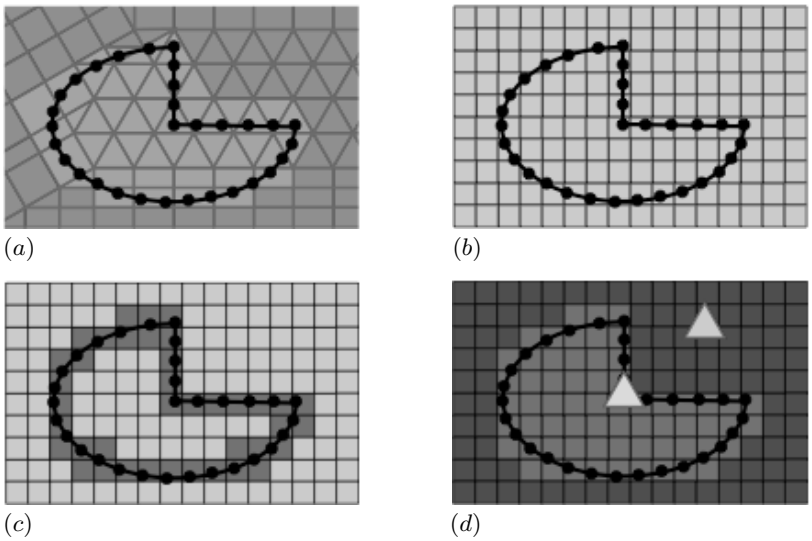


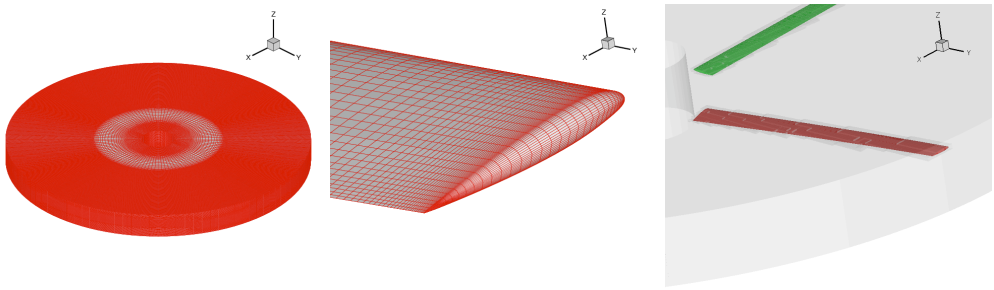
FIGURE 1. Schematic diagram of the Cartesian hole-map method: (*a*) hole surface (black curve with •) and underlying unstructured meshes; (*b*) Cartesian hole map (light grey squares); (*c*) hole surface approximated on the Cartesian hole map (medium grey squares); (*d*) interior (medium grey squares) and exterior (dark grey squares) identified on the Cartesian hole map.

leads to significant memory savings for very large three-dimensional problems. Note that many of the existing hole-cutting algorithms assume a hole to form a closed surface. For problems where any open hole surface is involved, users may artificially close the surface just for hole-cutting purposes using Delauney triangulation, as was suggested by Chiu & Meakin (1995).

In order to apply CHIMPS to hole cutting, the easiest idea would be to directly search for the solid-body surface mesh points and identify hole-boundary cells within a given set of volume meshes, followed by a certain kind of painting algorithm to determine the interior and exterior of solid-body surfaces. However, two major difficulties arise in this case: First, depending on the actual extent of surface-mesh resolution, the resulting hole-boundary cells may not form a closed surface. A systematic refinement of surface meshes may be necessary to guarantee the formation of closed surfaces and hence the success of the later painting algorithm. Second, the current version of CHIMPS keeps only the node connectivity information for a given set of volume meshes, but the painting algorithm essentially requires face-to-face connectivity as well. Therefore, we decided to make the hole-cutting capability completely independent of already existing search/interpolation functionalities and newly implemented a few API's exclusively for the hole-cutting purpose using the efficient Cartesian hole-map method (Chiu & Meakin 1995). Once hole-cut boundaries are known, the donor-cell identification is essentially a search/interpolation procedure and hence API routines from the original version of CHIMPS library are applicable virtually without any modification.

2.1. *Cartesian hole-map method*

Figure 1 shows a schematic illustration of the Cartesian hole-map method. As stated above, the ultimate goal of a hole-cutting process is to identify the interior and exterior of a given hole surface on the volume meshes of each block, as demonstrated in figure 1(*a*). In order to accomplish this task, the Cartesian hole-map method first constructs



(*a*) (*b*) (*c*)

FIGURE 2. Hole cutting for HART-II rotor blades: (*a*) full view of outer wake meshes; (*b*) zoomed view of blade meshes; (*c*) result of hole cutting.

auxiliary uniform Cartesian meshes (called a Cartesian hole map) with a proper mesh size ds within the bounding box of the hole surface (figure 1*b*). Usually, the average hole-surface mesh resolution is chosen as the hole-map mesh size ds. Then the given hole surface is approximated on the Cartesian hole map by searching the hole-surface mesh elements on it (figure 1*c*). The final step is to further discriminate the interior from the exterior of the hole surface on the Cartesian hole map by a painting algorithm. Once the interior and exterior are successfully defined on the Cartesian hole map, whether a mesh cell is interior or exterior to the hole surface can also be readily determined by searching for it on the hole map and checking the color (integer value stored) there (figure 1*d*).

One of the main advantages of the Cartesian hole-map method is that all the relevant search processes can be conducted very efficiently using the "search-by-truncation" method. For example, for a given mesh element with a bounding box $(x_{j,\min}, x_{j,\max})$ $(j = 1, 2, 3)$, the range of hole-map indices over which this mesh element spans can be easily determined by $i_{j,\min} = \text{int}[(x_{j,\min} - X_{j,\min})/\text{d}s]$ and $i_{j,\max} = \text{int}[(x_{j,\max} - X_{j,\min})/\text{d}s] + 1$, where $X_{j,\min}$ are the lower bound coordinates of the entire Cartesian hole map.

2.2. *Implementation of the Cartesian hole-map method onto CHIMPS*

To implement the Cartesian hole-map method onto CHIMPS, we first defined a new class of object (named '*hole*' object) in CHIMPS to represent the solid-body surface meshes. A hole object is composed of the Cartesian coordinates of solid-body surface mesh points and their node connectivity. Currently, triangular and quadrilateral elements are supported for the hole object. The first step for hole cutting is to register all the solid-body surfaces of each solver to CHIMPS as hole objects, a step which is handled by the newly implemented API routine `chimps_setHoleGeom`. For each hole object registered, the local bounding box and average surface resolution are computed locally on each processor. The global bounding box and global average surface resolution then can be easily constructed by collective communications. These two items are shared by all the participating processors and used to create the Cartesian hole map for each solid-body surface. Once the Cartesian hole map is constructed, the hole-boundary elements are first identified by checking the truncated indices of bounding-box coordinates of each surface-mesh element. Next, the interior and exterior of the hole boundary surface points are identified by the painting algorithm illustrated by Cho *et al.* (1999). Finally, the iblank array is defined rather conservatively by checking whether the Cartesian bounding box of a mesh element is fully contained in the exterior part of the Cartesian hole map. All of these procedures are realized by a single API routine `chimps_cutHoles` and the resulting iblank array can

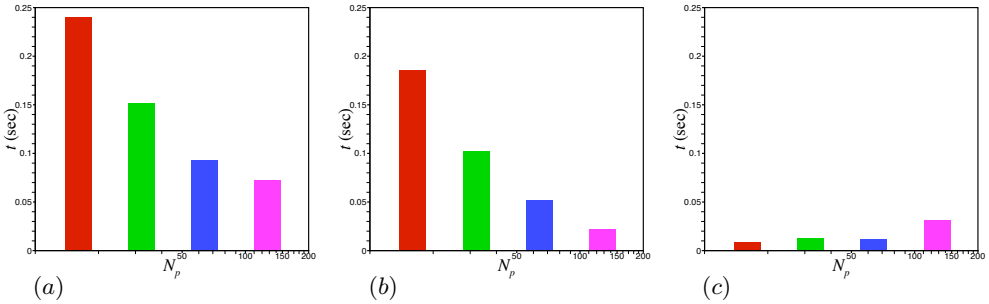


FIGURE 3. Scalability test for hole-cutting capability of CHIMPS: (*a*) total amount of CPU time; (*b*) CPU time required for iblank definition; (*c*) CPU time required for painting algorithm.

be returned to the driver program by calling the API routine `chimps_getIblank`. Figure 2 demonstrates how the hole-cutting functionalities of CHIMPS work for the HART-II rotor blades (Yu *et al.* 2002). In this sample case, four blades (figure 2*b*) are located in a cylindrical wake mesh (figure 2*a*), from where we want to cut holes around each blade. The outer cylindrical mesh has 16.73 million cells, whereas each blade has 8064 (112×74) surface elements. Figure 2(*c*) shows that CHIMPS correctly identifies the interior and exterior of blade surface meshes and successfully performs the hole-cutting procedure.

Figure 3 shows the scalability result for the CHIMPS hole-cutting capability performed on the same set of meshes in figure 2. Overall the scalability is found to be reasonable up to 128 and 64 processors for wake and blade meshes, respectively. In addition, the main portion of the computational cost is shown to be comprised of iblank definition in figure 3(*b*), which is almost perfectly scalable because each processor can work completely independently for this task. However, there is no parallelism involved in the painting algorithm that discriminates the interior and exterior on the Cartesian hole map (figure 3*c*). It is indeed almost unchanged with respect to the increase of processor number. The overall trend shows that the parallel performance would be somewhat saturated as the number of processors is further increased. However, the overall fraction of the hole-cutting procedure out of the entire simulation load is expected to be still minor in typical rotor applications. If the geometry of rotor blades is not too complex, further improvement would be also possible by adjusting the hole-map resolution, which is especially important in maintaining the load of painting algorithm smaller. It is usual to choose the hole-map resolution as the average resolution of the hole-surface meshes, but the number of three-dimensional hole-map meshes will be inversely proportional to its cube, which would result in an enormous increase in the cost of the painting algorithm. A mild relaxation of the hole-map resolution can lead to a substantial reduction in the operation count of the painting algorithm.

3. Validation for the overset-mesh capabilities

In this section, we present the results of actual CHIMPS-integrated multi-code coupled overset-mesh computations as a proof of validation.

3.1. *Flow over a circular cylinder*

The overset-mesh capabilities are first applied to a two-dimensional simulation of flow over a circular cylinder at the Reynolds number of $\mathrm{Re} = u_\infty d/\nu = 100$, where u_∞, d, and

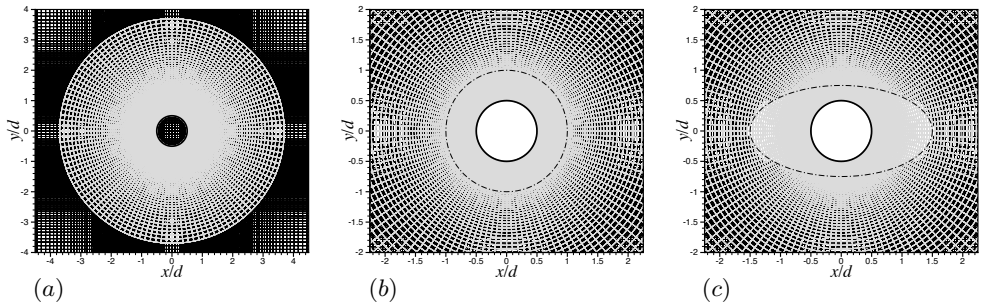


FIGURE 4. Overset-mesh configuration for flow over a circular cylinder: (*a*) cylindrical near-body (grey) and Cartesian exterior (black) meshes; (*b*) results of hole cutting for a circular stationary hole; (*c*) results of hole cutting for an elliptic moving hole. Dash-dotted lines denote hole surfaces.

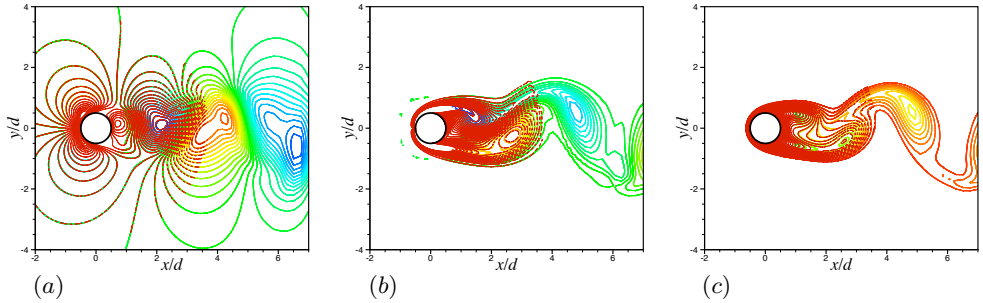


FIGURE 5. Contours of flow variables for the overset-mesh simulation with a static hole: (*a*) v; (*b*) ω_z (*c*) passive scalar. Dashed and solid lines are for near-body and exterior (blanked) meshes, respectively.

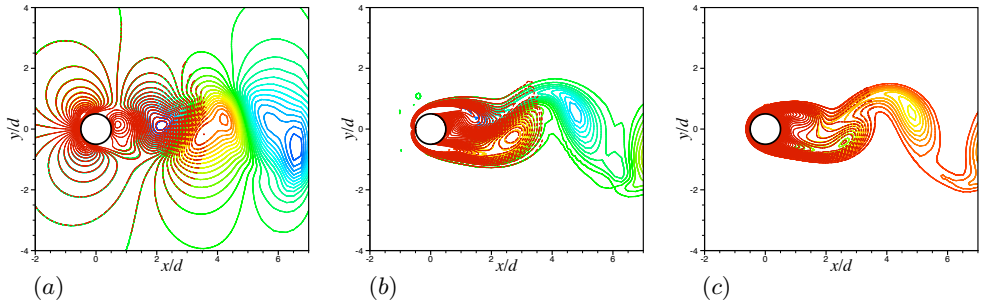


FIGURE 6. Contours of flow variables for the overset-mesh simulation with a moving hole: (*a*) v; (*b*) ω_z (*c*) passive scalar. Dashed and solid lines are for near-body and exterior (blanked) meshes, respectively.

ν are the free-stream velocity, cylinder diameter, and the kinematic viscosity, respectively. This problem is solved coupling two different instances of the incompressible CDP code (Ham & Iaccarino 2004). Both stationary and moving holes are tested. Note that the hole-cutting subroutines have to be called only once at the beginning of a run for the case of a stationary hole. Figure 4(*a*) shows the overset-mesh configuration, where the entire computational domain is decomposed into near-body cylindrical and exterior Cartesian domains. Instead of directly using the circular cylinder as a hole surface, a fictitious surface which is reasonably distant from the cylinder wall is chosen as the hole surface in

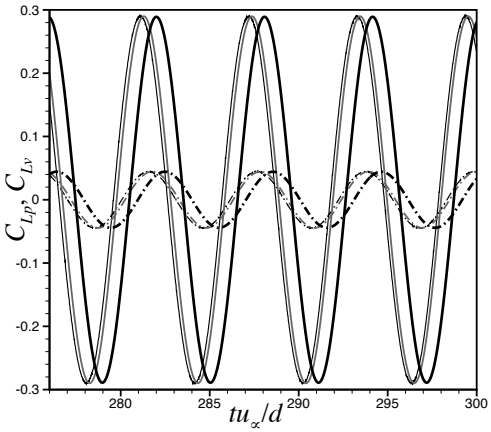


FIGURE 7. Time histories of the lift coefficient: Pressure (C_{Lp}) and viscous (C_{Lv}) lift components for solid and dash-dotted lines, respectively. Thin black and grey lines for a stationary and a moving hole, respectively. Thick black lines for the reference data from a single-code simulation.

this test in order to avoid near-wall solutions being interpolated and transferred. For a static hole, a circle with the radius of $r = d$ is chosen as the hole surface (figure 4*b*). On the other hand, for a moving hole, an ellipse centered at the origin is used with the lengths of major and minor axes of $3d$ and $1.5d$, respectively (figure 4*c*), and the angle between the major axis and x direction sinusoidally varies in time with $\theta(t) = \theta_0 \sin(2\pi t/T)$, where the amplitude and period are set to $\theta_0 = 30^\circ$ and $T = 3d/u_\infty$, respectively. Figures 4(*b*) and (*c*) show the result of hole cutting for each case. It is clear that the newly extended CHIMPS correctly completes the hole-cutting task.

Figures 5 and 6 show instantaneous contours of various flow variables for a static and a moving hole, respectively. In general, the solution quality is satisfactory for both static and moving holes. It is especially notable that all the contours are smoothly connected and no artificial flow structures or disturbances are generated at the domain interface. In addition, the two solutions almost perfectly agree within the overlap region, confirming that the newly implemented hole-cutting capability and hole-boundary treatment of CDP are accurate.

Figure 7 shows time histories of the pressure and viscous lift coefficients. The airload prediction from the present overset-mesh computations is also shown to be accurate. Even though slight high-frequency oscillations are found in the moving-hole case, overall the prediction accuracy for amplitude and frequency of unsteady airload variations is shown to be reasonable.

3.2. *Flow over a hovering rotor*

In order to verify our overset-mesh capabilities for a more realistic rotor problem, the newly implemented tools are also applied to the hovering rotor from Martin *et al.* (2001), where a single NACA2415 blade with the radius and chord of 406 mm and 44.5 mm, respectively, rotates at $\Omega = 35.0$ Hz, corresponding to the tip Mach number and chord-based Reynolds number of 0.26 and 272,000, respectively. The effective blade loading and the collective pitch measured from the chord line are $C_T/\sigma = 0.064$ and 4.5°. Figure 8 shows the domain configuration. For the near-blade flow solver, the fully compressible Stanford University multi-block code (SUmb; formerly known as TFLO, Yao *et al.* 2001)

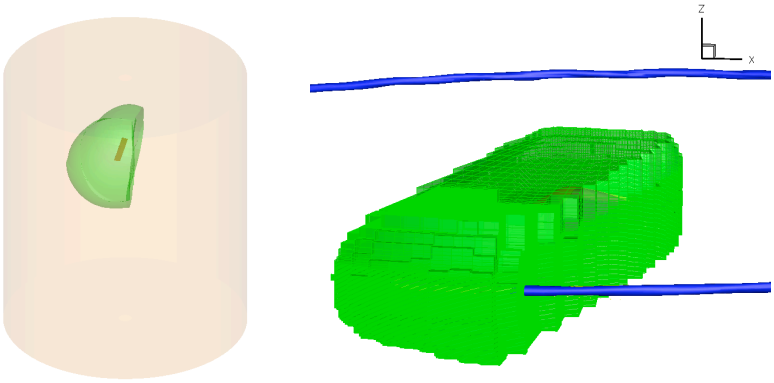


FIGURE 8. Domain configuration (left) and the result of hole cutting (right) for the hovering rotor case. Hemisphere-type and cylindrical surfaces for the SUmb and CDP domain boundaries, respectively.

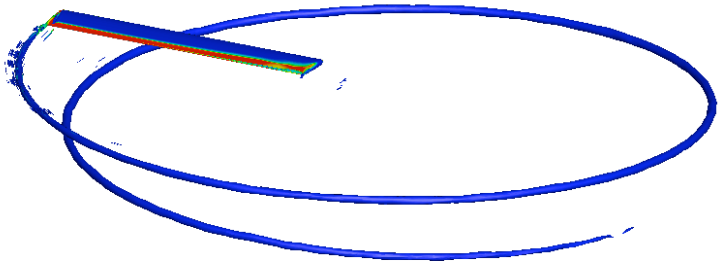


FIGURE 9. Iso-surfaces of q_2 for the hovering rotor case.

is used, whereas the incompressible CDP solver is used for the outer wake region. Simple cylindrical hexahedra are used for the outer CDP domain, where 216 uniform meshes are used in the azimuthal direction. On the other hand, meshes are clustered near the blade tip in the radial and vertical directions. The computation is performed with 2.96 and 16.77 million mesh elements for SUmb and CDP, respectively, and the time marching was completed until the tenth revolution from the impulsive start with 360 time steps per revolution. For turbulence modeling, the v^2–f turbulence model (Durbin 1995) is used.

Figure 9 shows the iso-surfaces of $q_2 = -\partial u_i/\partial x_j \times \partial u_j/\partial x_i = 2 \times 10^6$ after ten revolutions from the impulsive start. With this value of q_2, a long trail of thin strong tip vortex is shown to be well captured almost up to the wake age of $\zeta = 540°$. Figure 10 shows the comparison of azimuthal velocity profiles of the tip vortex at different wake-age locations. Even though a large overshoot is observed at $\zeta = 3°$, which corresponds to the SUmb domain in the overset simulation, and the tip vortex is located at more inboard locations in the simulation than in the experiment, the overall behavior of the tip vortex with respect to the wake-age angle is reasonably well captured, including the decay rate and asymmetry of the vortex showing higher azimuthal velocity on the inboard side.

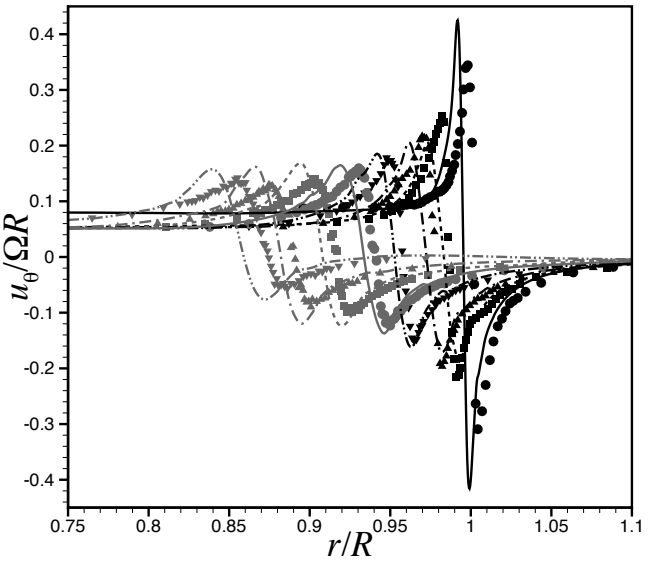


FIGURE 10. Swirl-velocity profiles at various wake-age locations for the hovering rotor case. Lines and symbols are for the present overset computation and experiment (Martin *et al.* 2001), respectively: ——— and ●, $\zeta = 3°$; - - - - and ■, $\zeta = 25°$; —·—·— and ▲, $\zeta = 45°$; —··—·· and ▼, $\zeta = 90°$; ——— and ●, $\zeta = 137°$; - - - - and ■, $\zeta = 209°$; —·—·— and ▲, $\zeta = 295°$; —··—·· and ▼, $\zeta = 371°$.

4. Higher-order interpolation on unstructured meshes

The interpolation engine for the current version of CHIMPS is based on trilinear interpolations and provides a very stable and robust solution reconstruction. Since many applications use higher-order discretization, it would be desirable for CHIMPS to support higher-order interpolations as well. However, as will be shown below, the unstructured multivariate higher-order interpolation is not as straightforward as univariate higher-order or multivariate linear ones. Important factors to consider in the unstructured interpolation algorithm include the order of accuracy, efficiency, unisolvence, and well-posedness. Regarding the efficiency, it would be much more desirable for the interpolation scheme to be local and hence matrix solution required for a global scheme could be avoided, because solution to a matrix system is potentially very expensive on unstructured meshes. For this reason, we assume that, together with a scalar function ϕ, three Cartesian components of its gradient $\partial\phi/\partial x_i$ are additionally given at each node point, and we pursue a higher-order interpolation (more specifically, accuracy higher than the second order) on this configuration. A univariate counterpart for this problem is the well-known Hermite interpolation for ϕ_P at $x = x_P \in [x_i, x_{i+1}]$:

$$\begin{aligned}
\phi_P &= \alpha_i\phi_i + \alpha_{i+1}\phi_{i+1} + \beta_i \left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_i + \beta_{i+1} \left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_{i+1} + \mathcal{O}(\Delta^4) \\
\alpha_i &= (x_{i+1} - x_P)^2(x_{i+1} + 2x_P - 3x_i)/(x_{i+1} - x_i)^3, \\
\alpha_{i+1} &= (x_P - x_i)^2(3x_{i+1} - 2x_P - x_i)/(x_{i+1} - x_i)^3, \\
\beta_i &= (x_{i+1} - x_P)^2(x_P - x_i)/(x_{i+1} - x_i)^2, \\
\beta_{i+1} &= (x_P - x_i)^2(x_{i+1} - x_P)/(x_{i+1} - x_i)^2,
\end{aligned} \tag{4.1}$$

which is fourth-order accurate. For an unstructured element with N nodes, one may expect to obtain a similar multivariate expression as follows:

$$\phi_P = \sum_{n=1}^{N} \alpha_n \phi_n + \sum_{n=1}^{N} \beta_{1,n} \left.\frac{\partial \phi}{\partial x_1}\right|_n + \sum_{n=1}^{N} \beta_{2,n} \left.\frac{\partial \phi}{\partial x_2}\right|_n + \sum_{n=1}^{N} \beta_{3,n} \left.\frac{\partial \phi}{\partial x_3}\right|_n, \tag{4.2}$$

which has $4N$ coefficients (α_n, $\beta_{1,n}$, $\beta_{2,n}$, $\beta_{3,n}$) to be determined. A straightforward procedure to obtain the necessary equations is to expand the right-hand side as the Taylor series about $(x_{1,P}, x_{2,P}, x_{3,P})$ and equate the coefficients term by term between the left- and right-hand sides, which results in the following:

- Zeroth-order derivative: 1 equation

$$\sum_{n=1}^{N} \alpha_n = 1. \tag{4.3}$$

- First-order derivatives: 3 equations

$$\sum_{n=1}^{N} \alpha_n (x_{i,n} - x_{i,P}) + \sum_{n=1}^{N} \beta_{i,n} = 0 \qquad \text{for} \quad i = 1, 2, 3. \tag{4.4}$$

- Second-order derivatives: 6 equations

$$\sum_{n=1}^{N} \alpha_n \frac{(x_{i,n} - x_{i,P})^2}{2} + \sum_{n=1}^{N} \beta_{i,n} (x_{i,n} - x_{i,P}) = 0 \qquad \text{for} \quad i = 1, 2, 3. \tag{4.5}$$

$$\begin{aligned} &\sum_{n=1}^{N} \alpha_n (x_{i,n} - x_{i,P})(x_{j,n} - x_{j,P}) + \sum_{n=1}^{N} \beta_{i,n} (x_{j,n} - x_{j,P}) \\ &+ \sum_{n=1}^{N} \beta_{j,n} (x_{i,n} - x_{i,P}) = 0 \qquad \text{for} \quad (i,j) = (1,2), (2,3), (3,1). \end{aligned} \tag{4.6}$$

- Third-order derivatives: 10 equations

$$\sum_{n=1}^{N} \alpha_n \frac{(x_{i,n} - x_{i,P})^3}{6} + \sum_{n=1}^{N} \beta_{i,n} \frac{(x_{i,n} - x_{i,P})^2}{2} = 0 \qquad \text{for} \quad i = 1, 2, 3. \tag{4.7}$$

$$\begin{aligned} &\sum_{n=1}^{N} \alpha_n \frac{(x_{i,n} - x_{i,P})^2 (x_{j,n} - x_{j,P})}{2} + \sum_{n=1}^{N} \beta_{i,n} (x_{i,n} - x_{i,P})(x_{j,n} - x_{j,P}) \\ &+ \sum_{n=1}^{N} \beta_{j,n} \frac{(x_{i,n} - x_{i,P})^2}{2} = 0 \quad \text{for } (i,j) = (1,2), (2,1), (1,3), (3,1), (2,3), (3,2). \end{aligned} \tag{4.8}$$

$$\begin{aligned} &\sum_{n=1}^{N} \alpha_n (x_{1,n} - x_{1,P})(x_{2,n} - x_{2,P})(x_{3,n} - x_{3,P}) + \sum_{n=1}^{N} \beta_{1,n} (x_{2,n} - x_{2,P})(x_{3,n} - x_{3,P}) \\ &+ \sum_{n=1}^{N} \beta_{2,n} (x_{3,n} - x_{3,P})(x_{1,n} - x_{1,P}) + \sum_{n=1}^{N} \beta_{3,n} (x_{1,n} - x_{1,P})(x_{2,n} - x_{2,P}) = 0. \end{aligned} \tag{4.9}$$

Therefore, twenty equations in total should be solved to obtain interpolation weights for

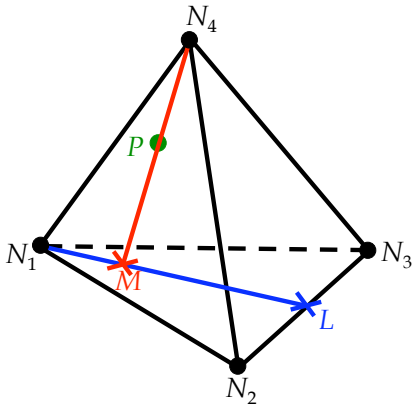


FIGURE 11. Schematic diagram for serial applications of univariate Hermite interpolation.

fourth-order accuracy. However, there is in general a fundamental inconsistency between the numbers of unknowns and equations, leading to non-unisolvence and arbitrariness. In the case of $N = 5$ (pyramids), the numbers of unknowns and equations are equal and unisolvence seems to be formally satisfied. Nevertheless, 20×20 matrix solution at each element would be considerably inefficient, particularly for a large set of unstructured data. More importantly, the above set of equations is a multivariate analogue of the Vandermonde matrix and indeed a direct matrix solution was very ill-behaving. Obviously, it is not straightforward to satisfy the higher-order accuracy, efficiency, and unisolvence at the same time on unstructured meshes. For example, Gasca & Maeztu (1982) developed an algorithm where unisolvence is always guaranteed. Furthermore, their resulting equations are simply reduced to a triangular matrix, which can be efficiently solved by a forward elimination. However, higher-order accuracy was not necessarily guaranteed in their algorithm.

Our current approach acquires higher-order accuracy by serially applying univariate Hermite interpolations along the properly chosen paths. Figure 11 shows a schematic diagram for a serial application of univarite Hermite interpolaiton. For four nodal points (N_1, N_2, N_3, N_4) of a given tetrahedron and an interpolation point P within it, the procedure can be summarized as follows:

1. Find the intersection point M between $\overline{N_4P}$ and $\triangle N_1N_2N_3$.
2. Find the intersection point L between N_1 and $\overline{N_2N_3}$.
3. Perform univariate Hermite interpolation at L along $\overline{N_2N_3}$ using given data at the two nodes.
4. Perform univariate Hermite interpolation at M using N_1 and L.
5. Finally, perform Hermite interpolation at P using N_4 and M.

Note that β_i and β_{i+1} are in the order of $\mathcal{O}(\Delta)$ in (4.2). Therefore, at every stage of univariate Hermite interpolation, the gradient components should be at least third-order accurate in order to guarantee the fourth-order accuracy. In the present implementation, we simply used second-order accurate linear interpolations for the gradient components and hence it guarantees a third-order accuracy in total. One may obtain a fourth-order accuracy by applying higher-order interpolations also for the gradient components at L and M, but it would not be an easy task because it will additionally require higher-order derivatives to be given at each node point. For elements other than tetrahedra, they

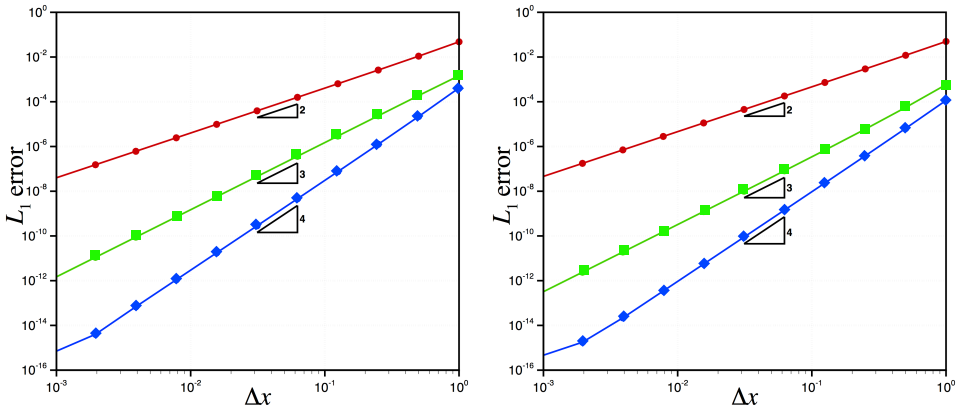


FIGURE 12. L_1 error versus mesh spacing for triangles (left) and tetrahedra (right): —●—, linear; —■— Hermite interpolations without higher-order derivatives; —◆— Hermite interpolations with higher-order derivatives.

are first decomposed into a number of disjoint tetrahedra, and one of them having the given point P inside is arbitrarily chosen. This procedure cannot satisfy the unisolvence, but the interpolated value will asymptotically converge to the true value with the mesh refinement.

Figure 12 shows the result of an accuracy test for the proposed interpolation procedure for two-dimensional triangular and three-dimensional tetrahedral elements. This figure shows that the present algorithm is indeed third- and fourth-order accurate with linear and higher-order interpolations applied to the gradient components, respectively.

Acknowledgments

This work was funded by NASA NRA's Subsonic Rotary Wing Program under the "Innovative Parallel Framework for Coupling Advanced Rotorcraft Aeromechanic Simulations" project, with Dr. Guru Guruswamy as technical monitor.

REFERENCES

ALONSO, J. J., HAHN, S., HAM, F., HERRMANN, M., IACCARINO, G., KALITZIN, G., LEGRESLEY, P., MATTSSON, K., MEDIC, G., MOIN, P., PITSCH, H., SCHLÜTER, J., SVÄRD, M., VAN DER WEIDE, E., YOU, D. & WU, X. 2006 CHIMPS: A high-performance scalable module for multi-physics simulations. *AIAA Paper* 2006–5274.

CHIU, I. T. & MEAKIN, R. 1995 On automating domain connectivity for overset grids. *AIAA Paper* 1995–0854.

CHO, K. W., KWON, J. H. & LEE, S. 1999 Development of a fully systemized Chimera methodology for steady/unsteady problems. *J. Aircraft* **36** (6), 973–990.

DURBIN, P. A. 1995 Separated flow computations with the k–ϵ–v^2 model. *AIAA J.* **33** (4), 659–664.

GASCA, M. & MAEZTU, J. I. 1982 On Lagrange and Hermite interpolation in r^k. *Numer. Math.* **39**, 1–14.

HAHN, S., DURAISAMY, K., IACCARINOX, G., NAGARAJAN, S., SITARAMAN, J., WU, X., ALONSO, J. J., BAEDER, J. D., LELE, S. K., MOIN, P. & SCHMITZ, F.

2006 Coupled high-fidelity URANS simulation for helicopter applications. Annual Research Briefs 2006. Center for Turbulence Research, Stanford University.

Ham, F. & Iaccarino, G. 2004 Energy conservation in collocated dicretization schemes on unstructured meshes. Annual Research Briefs 2004. Center for Turbulence Research, Stanford University.

Martin, P. B., Pugliese, G. J. & Leishman, J. G. 2001 High resolution trailing vortex measurements in the wake of a hovering rotor. In *Proc. of the AHS 57th Annual Forum, Washington, DC*.

Meakin, R. L. 1991 A new method for establishing intergrid communication among systems of overset grids. *AIAA Paper* 1991–1586.

Meakin, R. L. 2001 Object X-rays for cutting holes in composite overset structured grids. *AIAA Paper* 2001–2537.

Medic, G., You, D., Kalitzin, G., Pitsch, H., van der Weide, E. & Alonso, J. J. 2007 Integrated computations of an entire jet engine. In *Proc. of the ASME/IGTI Turbo Expo*. GT2007–27094.

Milgram, M. S. 1989 Does a point lie inside a polygon? *J. Comput. Phys.* **84**, 134–144.

Plimpton, S. J., Hendrickson, B. & Stewart, J. R. 2004 A parallel rendezvous algorithm for interpolation between multiple grids. *J. Parallel Distr. Com.* **64**, 266–276.

Steger, J. L., Dougherty, F. C. & Benek, J. A. 1983 A Chimera grid scheme. In *Advances in Grid Generation* (ed. K. N. Ghia & U. Ghia), pp. 59–69. American Society of Mechanical Engineers, Fairfield, NJ.

Wang, Z. J. & Parthasarathy, V. 2000 A fully automated Chimera methodology for multiple moving body problems. *Int. J. Numer. Meth. Fluids* **33**, 919–938.

Yao, J., Jameson, A. & Alonso, J. J. 2001 Development and validation of a massively parallel flow solver for turbomachinery flows. *J. Propul. Power* **17**, 659–668.

Yu, Y. H., Tung, C., van der Wall, B., Pausder, H., Burley, C., Brooks, T., Beaumier, P., Yves, D., Mercker, E. & Pengel, K. 2002 The HART-II test: Rotor wakes and aeroacoustics with higher harmonic pitch control (HHC) inputs – the joint German/French/Dutch/US project. In *Proc. of the AHS 58th Annual Forum, Montreal, Canada*.