EXPLICIT INTERPOLATION-BASED CFD MESH MORPHING

Ivan Malcevic[1], Arash Mousavi[2]

[1]General Electric - Research Center, Niskayuna NY., U.S.A. malcevic@ge.com
[2]General Electric - Aerospace, Evendale, OH., U.S.A. arash.mousavi@ge.com

ABSTRACT

This paper describes mesh morphing methodology based on explicit interpolation. The method handles large displacements while maintaining high mesh quality, has a fast run time, scales linearly with mesh size, and can easily be parallelized to many processing units. The method was originally developed to morph multi-block structured CFD meshes. Blocking topology is used to establish connectivity between domain boundary surfaces and to guide domain decomposition strategy. Blocking layout also defines interpolation direction and drives decisions on interpolation order and smoothness which are important to control the mesh quality. The method is combined with several pre-processing concepts to extend it to the morphing of unstructured CFD meshes. Shrink-wrap, domain decomposition, and automated blocking are combined to generate a structured background mesh used as the backbone of the morphing process. The resulting process efficiently morphs unstructured meshes with large surface displacements. Morphing capabilities are illustrated in several turbomachinery and external flow applications.

Keywords: mesh morphing, computational fluid dynamics, interpolation, background mesh, structured and unstructured mesh

1. INTRODUCTION

A CFD simulation cycle consists of pre-processing (geometry modeling and meshing), flow solution computation, and post-processing of the results. Traditionally, the most expensive part of the cycle is obtaining a converged flow solution. Improvements in numerical schemes, and the emergence of massively parallel CPU and GPU solvers drastically reduced flow solution wall clock time. Today, a well-converged solution can be obtained in just a few hours even for computationally demanding simulations. At the same time, the progress in geometry modeling and meshing has been modest. Parallel scalability is limited to a small number of processing units. A typical run time to generate an unstructured CFD mesh of an aircraft model or a wind turbine with a few hundred million elements is at least several hours. In many cases, mesh generation is already the dominant component in a CFD simulation cycle, with no clear solution in sight. As a result, the importance of alternative paths grows rapidly, mesh morphing being one of the most popular.

Mesh morphing can be loosely described as a modification of the baseline mesh while preserving mesh structure, also referred to as connectivity or topology. It should be noted that morphing cannot fully replace mesh generation but can provide a useful path to CFD mesh if a prototype exists.

Compared to mesh generation, mesh morphing offers several advantages, but has its limits as well. A major advantage is a much shorter workflow cycle. A morphing input is a baseline mesh, which has built-in know-how of mesh generation. If the existing mesh has already been used in a previous CFD run, it is reasonable to expect that it has proper surface and volume feature resolution (e.g. wake, leading or trailing edge resolution), wall-cell spacing (y+ distribution), design-practice "approved" boundary layer, defeaturing parameters applied, etc. These aspects (and much more) are part of a full mesh generation workflow and are rarely fully automated. They come for "free" in a mesh morphing workflow. Because morphing "just" deforms the existing mesh, one can expect a much shorter turnaround time. And, since mesh connectivity is unchanged, uncertainty associated with differences in meshing two similar (but still different) models is eliminated – an important aspect when comparing CFD results in re-design and optimization applications. However, fixed connectivity is also the main constraint since it limits the scope of a morphing workflow to similar models with limited geometry variations before the deformed mesh becomes

unusable. As a result, morphing has seen limited use in small amplitude optimization and fluid-structure workflows.

The success of an industrial morphing system is measured by how it addresses the major questions: i) what its useful application space is, ii) what displacement types and amplitudes it can handle, iii) parallel capabilities, and iv) scalability. The history of research on mesh morphing is as long as the mesh generation. Early methods targeting mapped meshes were succeeded by methods using elastic and spring analogy, followed by mesh-based PDE solutions including Laplacian and other smoothing methods. Examples of review papers on the comparison of various morphing methods can be found in [1], and [2]. In recent years, the radial basis function approach (RBF) is the method of choice. See [3], [4], [5], and [6] among others. RBF represents a meshless approach to morphing and offers an interpolation-like technique for distributing displacement fields into the domain. Much of the recent work on RBF focuses on the choice and support of the interpolation functions targeting specific application space. For examples see [7], [8], and [9].

Most morphing methods, including the popular RBF morph, are implicit which assumes solving a large system of equations (meshless or not) to compute displacement field across a domain. The scalability and run-time are directly affected by the size of this system. For the mesh-based methods, the size of the system is typically the size of the background mesh used to discretize the domain (which may or may not be the original CFD mesh). In the case of meshless methods, system size is driven by the size of the displacement source array. This depends on the nature of the application and how the system is set. In some applications, like fluid-structure interaction, it is driven by the number of surface mesh nodes, and the system size can easily grow into tens or hundreds of thousands. In such cases, the performance of the implicit methods could be significantly degraded. Various approximation techniques have been proposed to improve the performance in these cases, and this is still an area of active research.

The other success criterion, arguably more important than time performance, is the question of mesh quality. The morphed mesh needs to be usable for CFD simulation. This statement contains dual requirements. First, mesh quality needs to exceed minimum thresholds. Those are usually specified in terms of element quality metrics like aspect ratio, minimum element angles, or surface and volume ratios. However, equally important is the question of preserving mesh properties, not necessarily in the form of element quality metrics. The baseline (nominal) mesh contains the built-in engineering design practice that qualifies it as the "golden standard". Mesh properties like wake resolution, local surface refinement, and boundary layer characteristics are important features of any CFD mesh and should be preserved during the morphing. Remember, morphing serves as the alternate for full mesh generation, and full mesh generation workflow would most likely account for all design changes in deformed configuration, much like in the case of the original mesh.

To date, the question of maintaining the mesh quality during morphing has not been fully answered. Certain classes of methods are known to produce higher-quality elements than others. Other methods are known to be able to better tolerate lower initial mesh quality. Even within the same class of methods, the choice of underlying shape functions is very important, since it significantly influences the performance for specific problem categories. The requirement of preserving desired mesh properties adds to the complexity of implementation. Depending on the morphing method, these additions may be hard to implement or may result in a reduced smoothness level of the resulting displacement field. For example, the requirement of preserving the properties of the boundary layer translates into additional terms of the system matrix such that locally, the stiffness of the region near walls becomes very high. This in turn could potentially lead to a less numerically stable system and produce lower quality elements than desired. Additional requirements like re-positioning the wake region of the airfoil mesh in case of airfoil turn or moving the region of mesh refinement to keep up with expected shock location change due to airfoil redesign are even harder to implement.

The consequence of the above-mentioned difficulties is limited application space of morphing methods, usually expressed through small displacement amplitudes, and limited practical mesh size in a certain class of applications like fluid-structure interaction. In addition, the morphing step is usually accompanied by subsequent corrective action typically aimed at locally (and sometimes globally) improve on mesh quality. Such actions include smoothing, local node repositioning, and in some cases other mesh-improving techniques like swapping, refinement and coarsening. These actions complicate practical implementation and do not guarantee that the mesh quality issue will be resolved.

To address both runtime requirements and be able to better control mesh quality during morphing, we propose an explicit interpolation-based morphing method. The method was originally developed for structured mesh turbomachinery CFD applications to enable handling large displacements for various applications including airfoil design and re-design, optimization, and fluid-structure interaction but also applications like geometry feature additions and system applications. The method utilizes the structured nature of the CFD meshes to automatically determine the optimal domain decomposition and the order and direction of interpolation. The explicit nature of the method allows for a very fast run time and produces an algorithm that scales linearly with mesh size. Finally, the explicit method allows for easy control of local features like boundary layer properties, region-focused mesh control, local element quality check, and local real-time adjustments during mesh movement (rather than doing it post-factum).

The remainder of this paper is organized as follows: The next section focuses on the morphing environment and the interaction of morphing with other pre-processing (geometry and meshing) techniques. It is important to understand these elements and their interactions since the performance of the proposed morphing methodology can be properly understood only within the larger framework where several techniques come together to form a fully functional system with capabilities exceeding the capabilities of each technique applied separately. The main section describing the method to morph structured CFD meshes then follows. Finally, the

extensions of the method to unstructured CFD mesh morphing are detailed. The capabilities and usability of the proposed method are illustrated in the examples shown throughout this paper.

2. MESH MORPHING ENVIRONMENT

To fully understand the method described below, it is important to learn how it fits within a mesh morphing environment and, more generally, how it interacts with various modules of a dynamic pre-processing environment.

2.1 Elements of Morphing Process

The crucial aspect for the success of any morphing application (and a more general pre-processing application) is a holistic approach. In the case of a morphing workflow this means the inclusion of three fundamental elements:

- Motion definition
- Surface mesh morphing
- Volume mesh morphing

Morphing workflow buildup starts with the motion definition step. In this stage, information on surface and volume motion and related constraints should be gathered and organized. The result of this stage is a set of rules that describe the motion of the domain boundary and internal surfaces, and additionally, the motion of regions of special interest. The set of rules should describe a conformal, unambiguous, and well-defined motion field (particularly at intersection regions). In general, motion definition constraints are derived from two sources: target application and modeling system/tools environment. An example of motion definition for a turbomachinery CFD application is illustrated in Figure 1. A domain consists of a single passage (slice of a full wheel) around an airfoil. A cross-section of the model mesh is shown in the figure. Air flows from left (inlet surface) to right (outlet surface). Circumferential symmetry employed in the simulation allows the model to be reduced to a single airfoil with bottom and top periodic surfaces, with 1-1 node and element match. Radially, the CFD domain is bounded between the inner (hub) and outer (casing) surfaces of the revolution. In addition to airfoil shape change, the motion definition set of rules should specify how each of the six bounding surfaces moves for the CFD simulation intent. For some surfaces, the motion could be explicitly specified. For example, in the case of a fluid-structure interaction, the motion of the airfoil surface mesh is specified in the form of a displacement field obtained from a model mechanical analysis. In this case, the motion is specified directly on mesh nodes. In a case of an airfoil redesign, the surface motion might be specified in the form of a new CAD representation while mesh motion needs to be computed separately. Some of the surfaces might be stationary (hub and casing surfaces), while the associated mesh moves (slides) along the surface (slip motion condition). The motion of inlet and exit surfaces might depend on a wider CFD setup. If there are additional blade rows in this simulation forward and aft of this airfoil, the axial (horizontal) location of the inlet and exit surfaces should be constrained. Depending on the capabilities of the CFD solver and the type of simulation, inlet and exit surface meshes might be allowed to slide in the circumferential direction (top-bottom direction in the figure) or might need to be frozen. For the example in Figure 1 inlet mesh (left boundary) was allowed to slide and the exit mesh (right side) was kept frozen

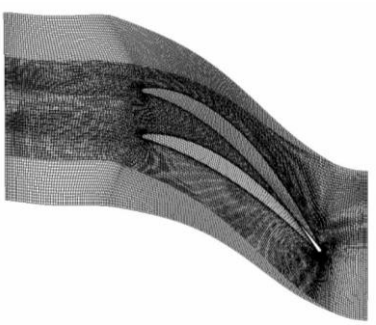


Figure 1. Single airfoil passage mesh morphing

The decision on periodic surface and mesh motion depends on solver capabilities, and the type of simulation. In this case, the inlet surface mesh slides circumferentially, hence, the periodic surface mesh must move as well. At the top and bottom, the motion is constrained to the hub and casing surface. The motion in the middle of the periodic surface is decided based on simulation intent. One should choose to move the periodic mesh nodes to mimic airfoil motion to maximize the morphed elements' quality. (Figure 2).

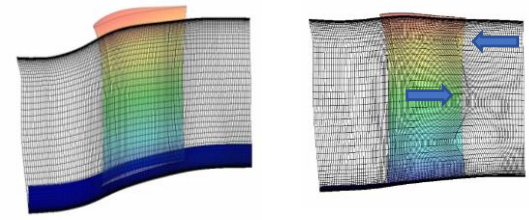


Figure 2. Periodic surface mesh motion

Using the above example as a guide it can be observed that the set of rules describing target application and motion definition are not the same. This is an important distinction when considering CFD versus, for example, structural analysis. Consider the target application of the airfoil re-design described above. For an engineer working on airfoil optimization, airfoil shape change fully defines the problems. The same airfoil shape change is sufficient for a mechanical analyst as well since the new airfoil shape fully defines the structural model. However, an airfoil is just one of the surfaces of the larger CFD domain. As described above, other considerations that stem from a larger hardware system (of which the airfoil is just one component), the type of the CFD simulation, and modeling system capabilities play a role in how the CFD domain changes its shape. Hence, while in both cases the target application is the airfoil re-design, one ends with a significantly different (wider) set of motion rules in the case of a CFD workflow.

Once the motion definition is complete, the surface mesh morphing can commence. Depending on the nature of surface motion the computation of new surface mesh could be done in a separate step (explicit surface mesh motion) or can be performed simultaneously with volume morphing (implicit surface mesh motion constraint). For the example shown in Figure 1, inlet surface motion could be precomputed by prescribing the amount of the circumferential shift or could be left to be computed at the time of volume mesh morphing as a fallout from the volume mesh morphing interpolation scheme with the additional constraint that the axial motion of the inlet surface nodes is set to 0. The exact workflow depends on the capability of the morphing method used, and the existence of additional CFD constraints like the need to impose a specified flow angle in the inlet region for a smooth transition to the upstream blade row. Note how the motion definition (circumferential slide), surface mesh morphing (explicit or implicit, constrained, left to be computed during volume morph), and volume mesh morph elements of the morphing environment merge into a single workflow decision which has a direct consequence on the quality of the morphed mesh. Also, note the flexibility of the modular environment. Depending on the requirements, each of the steps can operate in a different mode. For example, we could freeze the inlet mesh, or in the case of circumferential slide precompute new inlet mesh. The decision on how to operate should be derived from motion definition constraints, capabilities of the morphing, pre-processing, and simulation systems, and projected morphed mesh quality for each of the possible workflows.

Surface mesh displacement (explicitly computed or implicitly constrained) serves as a boundary condition for volume mesh morph. The role of volume morphing is to distribute surface mesh displacements into the domain volume in an "optimal" way. The definition of "optimal" depends on several factors including the original mesh quality, displacement amplitudes, the local and global mesh quality criteria (e.g. boundary layer properties), etc.

When discussing the morphing environment, the emphasis is on the modular structure. Both surface mesh and volume mesh morphing parts of the environment should contain several different morphing modules instead of focusing solely on one method. Every modern commercial pre-processing package offers several meshing methods. Workflows for meshing complex models (manual or automated based on feature recognition) break the process into several stages each using the right meshing tool for the job. The same approach should be taken in mesh morphing. Parts of the model might be best morphed using projection, others by parametric mapping, while the rest might best be served by interpolation.

2.2 Preprocessing Environment Interaction

The discussion now shifts one level up to focus on the interaction of the morphing environment with the wider set of pre-processing modules. A pre-processing environment can be loosely described as a collection of modules/capabilities each performing specific geometry or meshing task. The morphing environment is the subset of this pre-processing module set. Other subsets might include mesh generation modules (Delaunay, advancing front, overset, etc.), and geometry handling modules, but also capabilities like smoothing, domain decomposition, splicing, inflation, grafting, etc. In a dynamic environment, a set of such modules is pulled together into a workflow serving the target application space.

The previous section illustrated how the interaction between morphing modules results in different system performances. Similarly, the interaction between a morphing environment and the wider pre-processing module set brings a new quality to modeling and enables morphing methods to work in their sweet zones. For example, this paper discusses the extension of the volume morphing method originally developed for the morphing of structured meshes. The method is grouped with shrink wrap, automated blocking, and domain decomposition techniques to extend it to the morphing of unstructured meshes.

The illustration in Figure 3 shows how morphing, and mesh inflation can be paired to enable a transformation of a standard airfoil mesh shown on the left, to a more complex model that contains an airfoil root fillet (right). In this case, the combination of inflation and morphing enabled a transition between two geometrically topologically different models. A qualitatively new, higher fidelity CFD simulation is enabled. Workflows like this are referred to as morphing-enabled systems emphasizing the morphing role (although morphing is one of several technologies used to realize the application).

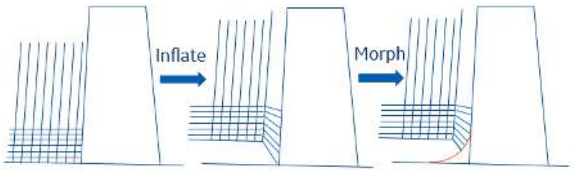


Figure 3. Morphing + inflation workflow

2.3 Morphing Application Space

This section concludes with a few examples illustrating the CFD mesh morphing application space. The examples shown represent a small sample of what morphing can be used for. In each example, an emphasis is made on how the morphing capability was paired-up with other pre-processing modules to realize additional simulation benefits.

It is worth noting that all the models shown were generated using the morphing method described in the next section.

When discussing morphing application space, the usual association is re-design and optimization. Morphing-enabled workflow operates in existing capability space and the primary benefit is the short simulation cycle time. In a typical meshing workflow, time-to-mesh is anywhere from several minutes to an hour. Efficient morphing implementation can reduce time-to-mesh to just a few seconds. The redesign and optimization cycle usually requires hundreds of runs which then translates to big computational and design time savings. In Figure 4 several examples of airfoil re-designs are shown.

In such workflows, morphing is usually paired with the underlying geometry (CAD) system used to design the component of interest. Knowledge of the CAD system is used to understand the best approach to surface mesh morphing. It is important to note that this is a morphing-to-target-geometry application where airfoil displacements are given in the form of a new geometry model, as opposed to a point cloud model where a displacement field is explicitly specified on a point set surrounding model. Note the ability to handle large changes in airfoil shapes, one of the features of the proposed morphing method.

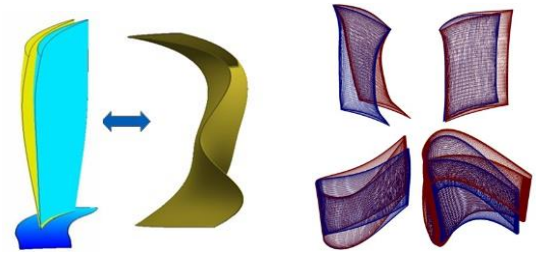


Figure 4. Airfoil re-design and optimization

The example shown in Figure 5 represents the use of morphing for so-called cold-to-hot and hot-to-cold shape changes. Typical aero design is performed in the so-called hot state which, for aircraft engines, might correspond to airplane cruise speed. For manufacturing, these shapes need to be converted to so-called cold shapes (hot-to-cold transformation) and later transformed into shapes that represent different operating conditions (partial or overspeed). In these workflows, morphing is usually paired with mapping/interpolation used to map the displacements computed on a coarse structural mechanical model to a much finer CFD mesh. The primary benefit of morphing-based workflow is the direct coupling of mechanical/thermal with aero modeling compared to one-way and reduced order implicit information sharing, leading to more accurate predictions.

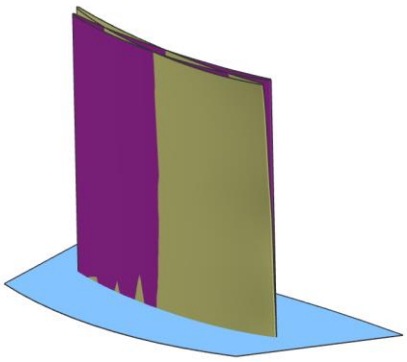


Figure 5. Hot-cold transformation

Morphing can also be used to extend the modeling capabilities of existing systems. Examples shown in Figures 3, and 6 show such morphing use. In the example in Figure 6, morphing is used to study the secondary flow effects on airfoil thermal and aero performance by introducing a so-called leading edge bulb fillet. The primary benefit of morphing-based workflows in these cases is a much shorter lead time to modeling new capabilities compared to modifying existing design systems to accept new shapes. It allows for fast new concept evaluation and down selection. Accepted concepts can then be added to production system modeling capabilities at a later stage.

Finally, the example in Figure 7 illustrates the use of morphing in a system modeling application. In this case, the morphing of a single airfoil passage mesh (like the one shown in Figure 1) was paired with domain decomposition and splicing to enable the generation of a full-wheel turbomachinery mesh to study the effects of airfoil resequencing.

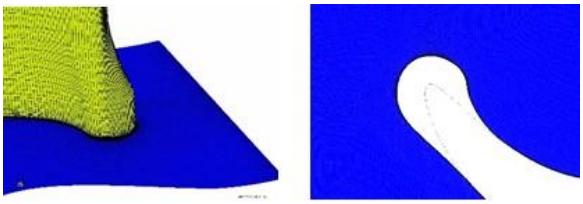


Figure 6. Study of secondary flow effects on airfoil performance using leading edge bulb fillet

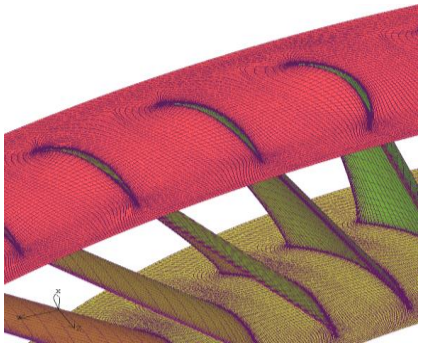


Figure 7. Morphing for CFD system applications

3. STRUCTURED MESH MORPHING

In this section, the algorithm to distribute surface mesh displacements into the volume of a CFD domain is described. The discussion is restricted to structured CFD meshes.

Structured CFD meshes (also known as body-fitted or mapped meshes) consist of one or more blocks. Each block has the topology of a cube. Block faces are associated with external surfaces of the domain or coincide with a face of another block (internal interfaces). Structured meshes do not have an explicit element structure. Mesh nodes are accessed via a block number and local indices. There is no explicit node connectivity data stored in memory since index values are sufficient to compute any "element" related values (e.g. stiffness matrix).

Structured meshes are very efficient in discretizing high aspect ratio structures like aircraft wings and fuselage, wind turbines, leading and trailing edges of airfoils, and other

regions where high aspect ratio elements are desirable (e.g. clearances). The major drawback is the difficulty associated with discretizing complex domains, which limits their application space. However, when available, structured CFD meshes are preferred and used as the golden standard to benchmark solution quality.

Structured meshes have properties that lend them to rather a straight-forward morphing process as well. If one considers the domain as a collection of cuboids, the problem of domain deformation reduces to deforming each chunk to reasonably preserve a cube-like shape. In such a case, we can expect that the mesh inside each block will remain valid and conform to quality thresholds. The important message is that, instead of working at the element or vertex level, the process abstracts to the block level. This is a major benefit. Even in the most complex structured meshes, the block count rarely goes into the hundreds. Figures 8 and 9 illustrate blocking layouts for two model types at both ends of the spectrum. The airfoil passage mesh shown in Figure 8 contains a very low number of blocks, often not more than 10. On the other side of the application space is a full aircraft model for which a section detail is shown in Figure 9. Such mesh typically consists of several hundred blocks and is considered high-end in terms of complexity. Even for these models, block count is negligible compared to the number of nodes and elements which are in millions. Hence, operations performed at the block level are computationally insignificant.

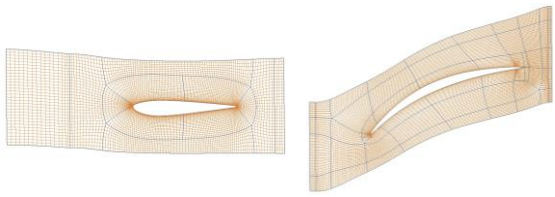


Figure 8. Sample block layout: airfoil passage

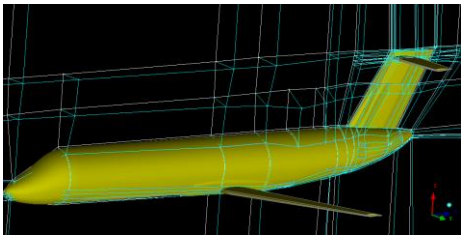


Figure 9. Sample block layout for aircraft model

The second important property of structured meshes is the alignment with main flow features. When looking at a cross-section of a structured mesh, one can follow a grid line in the boundary layer region that "flows" around the structure (e.g. wing), a streamwise grid line direction aligned with the flow, and a cross grid line direction flowing from one domain boundary to another. These grid lines connect opposite surfaces of the domain where the boundary conditions are specified in the form of surface displacements.

The facts observed above can be used in an efficient two-step iterative scheme to interpolate boundary displacements into the domain. At the start of an iteration, a part of the domain is fully morphed. The rest of the domain is still in the baseline state and is active in the sense that morphing still needs to be performed on it. In the first step, a search is performed on the active domain to identify a block chain connecting opposite surface boundaries. Once the connecting block chain is established, the mesh in this connecting region is morphed using interpolation. This completes the iteration, after which the additional part of the domain (defined by block chain) has been morphed. The iterative process repeats until the domain is fully morphed.

The two-step approach has important distinguishing features.

First, the connectivity search step operates at the block level and naturally lends itself to built-in domain decomposition. Due to low block count, even the most complex search and optimization algorithms are cheap. One has the freedom to collect all valid connectivity paths, and then to use a variety of priority-driven decision algorithms to choose which ones to tackle first. The priority criteria can vary. "The most important region (to be morphed first)" could be defined by application workflow. It might be the region where a shock is expected to form and maintaining the morphed mesh quality is critical. It might also be the wake region where tighter mesh resolution must be maintained. The order of morphing is important since the boundary constraints accumulate and the active domain gets smaller with each added iteration. Having the opportunity to decide the morphing order, increases the chances of getting valid morphed mesh (for example by morphing first the regions with the smallest elements or the region with the lowest mesh quality).

The second aspect in which this approach stands out is its explicit interpolation nature. The connectivity search step (with associated priority decision) identifies the block chain to be morphed. This region has the topology of a cube, with opposite sides being surface mesh patches on domain surfaces (or the surface of the previously morphed region). A set of grid lines (topologically parallel to each other) connect these patches. For each connecting grid line, displacements at ends are known. To distribute them into the domain, one simply interpolates along the grid line using a chosen distribution law, the simplest one being linear interpolation (by grid line length). Irrespective of the interpolation function, (a few choices are discussed at end of this section) the computation of the displacement for each internal node is explicit and depends only on the displacements of the end nodes. There is no system of equations to be solved, there is no matrix–vector multiplication involved. Computational cost is reduced to a single nodal sweep, with a single displacement calculation per node. Thus, the computational cost scales linearly with mesh (nodal) count.

The iterative domain decomposition nature of the process allows the addition of steps to the morphing workflow to improve robustness and target CFD-specific features.

First, note that in a typical CFD mesh, elements near surfaces have very small thicknesses and high aspect ratios (typically in thousands) (Figure 10). This region requires special treatment during mesh morphing since thin boundary layer elements cannot tolerate distortion. Relative vertex motion needs to be kept at a minimum, particularly in the direction normal to the wall. In CFD workflows, the boundary layer mesh (region 0 in Figure 12) is morphed first. It is morphed

in a pointwise-rigid manner: each surface mesh point has an associated grid line emanating from the (airfoil) surface. Each node on the grid line is assigned the same displacement as the corresponding surface node. For most practical applications, this procedure generates a boundary layer mesh of the same quality as the original by preserving the orthogonality and the spacing of the wall. Figure 10 shows details of a boundary layer mesh for an airfoil re-design with an airfoil section rotated 10 degrees, and a chord shortened by 5%. Note how two meshes look almost identical near the airfoil surface.

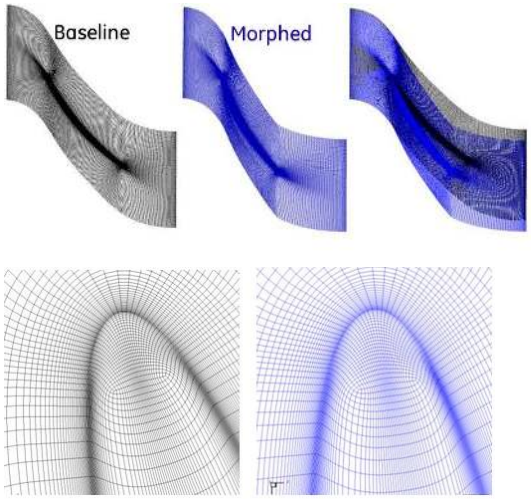


Figure 10. Boundary layer mesh preservation

The other important requirement is to ensure that boundary surface constraints are fully satisfied. For a CFD mesh morphing workflow this requirement typically includes 1-1 periodic surface mesh match and the requirement that bottom and top mesh layers lie on prescribed surfaces (hub or inner surface, and case or outer surface). To satisfy these requirements one can modify the interpolation step described above to bi-directional (or tri-directional) interpolation for block chains that touch the surfaces in question. However, practical implementation typically uses a simpler decoupled approach. During the iterative volume morphing stage, standard interpolation is performed along grid lines. No special consideration is given to the block chains that touch the hub or case surfaces. As a result, outer mesh layers may not lie on designated surfaces. To rectify this, a post-interpolation step is added in which surface mesh nodes are projected to designated surfaces, and associated grid lines (emanating from surface nodes) are appropriately stretched/contracted by interpolating the surface node displacement on the grid line (much like in the regular interpolation step during iterative morphing process). To ensure 1-1 periodic match, surface mesh projection should be along the radial direction (passing through the axis of revolution).

3.1 Morphing Workflow Example

The above method is embedded in a workflow to morph a structured CFD airfoil passage mesh shown in Figure 12. The workflow follows the three-part approach to morphing discussed in Section 2. The volume morph algorithm described above is the element of part 3 of the workflow. The steps of the workflow are shown in Figure 11.

1 Motion Definition

Surface	Motion description
Airfoil	User given
Inlet/Exit	Circumferential slide
Periodics	Midchord: replicate airfoil mid-camber motion
	Fwd/Aft: taper to conform to inlet/exit motion
Hub/Case	No surface deformation. Slip condition

2 Surface Morph

Surface	Surface Mesh Morphing Details
Airfoil	User Given
Inlet/Exit	Constant circumferential motion defined by airfoil LE/TE
Periodics	Midchord: replicate airfoil mid-camber motion
	Fwd/Aft: linear taper to conform to inlet/exit motion
Hub/Case	Slip condition. To be enforced during volume morphing stage.

3 Volume Morph

3.0	Boundary layer motion: constant displacement along grid lines off the airfoil	
3.1	Initiate active boundary to domain outer boundary and boundary layer outer sur	
	Initiate active domain to full domain minus boundary layer	
	Set interpolation law to linear along grid line arc length	
	Set priority path criteria. Initiate priority path queue	
		1. By boundary condition: airfoil walls, periodic bc, inlet bc, exit bc
		2. By element quality: skewness (minimum angle)
		3. By path aspect ratio: from low to high aspect ratio
3.2.	Iterative search and interpolate	
	While active domain exists	
		1. Search all valid paths in active domain
		2. Place paths in priority queue
		3. Interpolate surface displacements into the highest path block chain
		4. Update active domain, surface mesh displacements on active domain bdry
3.3.	Finalize:	
		1. Project surface mesh nodes of first and last layer to hub/case
		2. Interpolate: distribute endwall displacements along spanwise grid lines

Figure 11. Morphing workflow: Airfoil passage

The workflow begins with surface motion definition. Whenever given a choice, the motion should be chosen to allow maximum flexibility during morphing. In this example of an airfoil redesign, airfoil deformation is a user input. The redesign shown in Figure 12 consists of 10% chord reduction at 15% and 90% of the airfoil span and 10% chord extension at 50% span. CFD workflow constraints axial (flow direction) location of inlet and exit surfaces, and inner and outer diameter surfaces (hub and case) do not deform. Other than that, external surfaces are allowed to move. To allow for maximum flexibility during morphing (and the best chances of good mesh quality after morphing), inlet and exit surfaces are allowed to slide circumferentially, and periodic surfaces are allowed to slide to replicate airfoil motion in the middle of the domain. To ensure conformity, periodic movement tapers from the value in the mid-domain to match the corner motions of inlet and exit surfaces. While hub and case surfaces do not deform, the surface mesh is allowed to slide along the surfaces.

In the surface morph part of the workflow motion definition is used to compute a morphed surface mesh in cases where sufficient information exists or impose constraints on the volume morph if that is not the case. For this airfoil redesign,

inlet and exit morphed surface meshes are computed using angle shifts defined by airfoil leading and trailing edge motion. Constant shift angle is applied to grid points on the same circumferential grid line. Airfoil surface motion morphed inlet and exit surface meshes with the periodic motion definition in part 1 of the workflow, provide sufficient information to compute morphed periodic surface mesh (Figure 2). Hub and case surface meshes are not computed at this time. The constraint that these meshes lie on respective surfaces needs to be built in the volume morph part of the workflow.

The volume part of the workflow begins with the morphing of the boundary layer region (region 0 in Figure 12). For every grid line emanating from the airfoil surface, and for every node on the grid line, a displacement of the corresponding surface node is applied. After completion of this step, several initialization steps are performed to allow for the iterative part of the workflow to begin. The active domain and its boundary are defined (all the domain except region 0). Interpolation law is set to linear along grid line length. A set of priority rules are defined to facilitate the priority sorting of connectivity paths during the search step of an iteration. The sets of rules are divided into subgroups, the lower group of rules serving as a tiebreaker in case two or more equivalent paths exist. The order of the groups and the order of boundary condition types are not arbitrary and are driven by the application type. At the top of the priority are paths that connect parts of the model (for example two airfoil surfaces in the case of a multi-airfoil model). The reason: the mesh quality near airfoil surfaces (or interfacing with airfoil boundary layer) is of the highest importance for the quality of a CFD simulation. Similarly, the periodic boundary condition is placed in front of the inlet and exit since in most airfoil meshes, the skewness of the mesh in the mid-passage (the region between the airfoil and periodic surfaces) plays important role in accurately resolving flow features like vortices and shocks. However, if the flow is expected to be subsonic (no shocks), one might prioritize wake mesh quality and place the exit surface in front. Similarly, if this simulation targets the interaction between blade rows in an aircraft engine, and this is the second airfoil in the setup, the quality of the inlet mesh region would be important to ensure proper flow feature transition to the front of this airfoil. In such cases, inlet boundary condition should be prioritized in the workflow. (Note the importance of the flexibility of the workflow. The one-size-fits-all approach does not apply here.)

With priority rules and a priority queue in place, the iterative search-and-interpolate part of the process can begin. During the search step of an iteration, a set of paths (block chains) connecting the surfaces of the active domain is collected. For this example, the paths found in the first iteration are shown in the upper left part of Figure 12 (labeled 1a thru 1d). Using a priority set of rules, paths labeled as “1” are selected (upper right of Figure 12). In the interpolate part of an iteration a sweep is performed thru grid lines along direction “1” connecting the airfoil boundary layer and the periodic surface. For each grid line, displacements of the end nodes are known (one end node on periodic surface mesh, the other end node on the boundary layer region morphed in step 3.0). Using linear interpolation, end node displacements are interpolated on the nodes of the grid line. At the conclusion of the iteration, the intermediate mesh shown at the bottom right of Figure 12 is obtained (labeled as “1”).

The process continues with iterating on the updated active domain (top left of Figure 13), searching and obtaining valid paths (labeled 2 and 3), selecting paths labeled 2, and interpolating along inlet grid lines to obtain the intermediate mesh stage “2” at the end of the second iteration (bottom right of Figure 13). In the final iteration, the remaining exit region of the mesh gets morphed in a similar way.

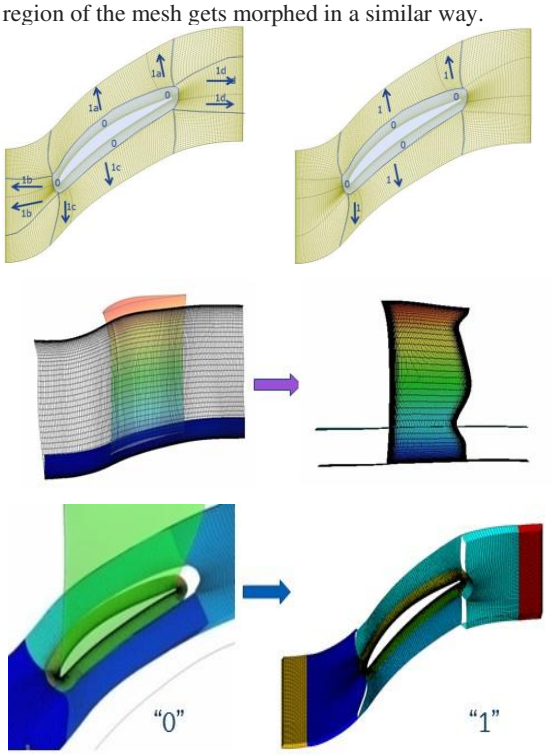


Figure 12. Morphing process: iteration 1

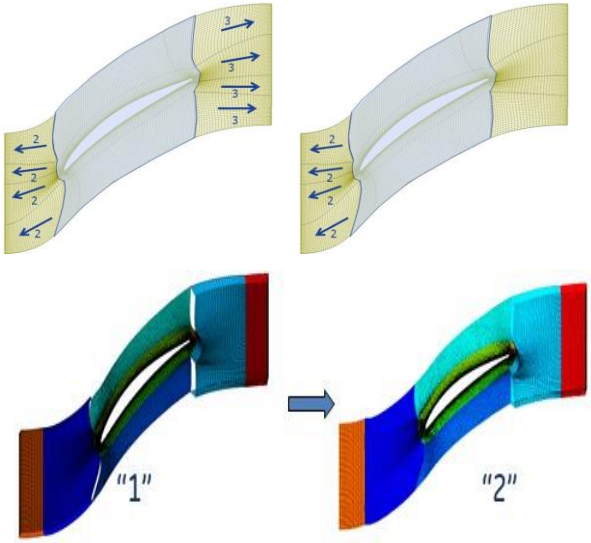


Figure 13. Morphing process: iteration 2

After the completion of the search-and-interpolate stage, the remaining task is to ensure any surface constraints set in part 2 of the workflow are satisfied. In this example, surface mesh constraints are applied to the bottom and the top layer of the mesh such that the final morphed mesh conforms to the target domain. Mesh nodes at the bottom layer are projected to the hub and nodes at the top layer are projected to the case surface. Radial projection (vector passing thru the axis of rotation) is used to maintain rotational periodicity. In the final step displacements of the end nodes are interpolated on the nodes of the grid lines connecting the hub and case surfaces.

3.2 Method Performance

The discussion on method performance includes an assessment of runtime, scalability, and the ability to handle large displacements.

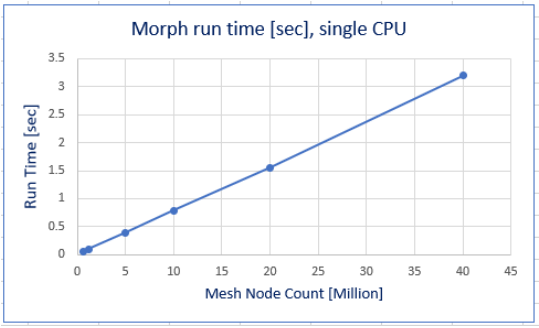
As previously noted, the bulk of the computational cost lies with the interpolation stage of the volume morph which is implemented as a single nodal sweep. Hence, it is expected that the method scales linearly with the nodal count and that the run time is short. The graph in Figure 14 shows the runtime performance of the method for the workflow example described in the preceding section. For the scalability test, the mesh count was varied from 0.1 to 40 million nodes. Method indeed achieves linear scalability with respect to the nodal count. From the same graph, one can see that mesh with 25 million nodes morphs in about 2 seconds on a single CPU. (The reader should focus on ball-park run-time performance rather than the value. The exact run time will depend on the system hardware, and implementation of the workflow).

The ability to handle large displacements is tied to the key feature of the method: the interpolation of surface displacements along grid lines connecting domain surfaces. The goal is to produce a morphed mesh that satisfies quality thresholds. In real-life applications, the quality of a mesh is determined using a set of metrics depending on application specifics. For this presentation, aspect ratio and element determinant are chosen to demonstrate the method's performance. Aspect ratio plays important role in characterizing the boundary layer mesh region and is used on a relative scale. For reliable CFD the aspect ratio of morphed and baseline meshes in this region should be very similar. Determinant, on the other hand, is used on the absolute scale: all elements should have a determinant larger than the threshold value set by the design practice.

The quality of the mesh should be assessed separately for boundary and far-field regions. (CFD mesh has different properties, and the morphing method differs in these regions.)

As described above, the application of constant displacement on the nodes of grid lines emanating from boundary wall surfaces preserves element shape. Therefore, minimal changes are expected during morphing for this mesh region. The table in Figure 14 illustrates the performance of the method for the airfoil redesign example: aspect ratio and determinant in the near-field mesh region are essentially unchanged during morphing.

For the far-field region, during every iteration, end nodal displacements are interpolated on all topologically parallel grid lines of the block chain that connects domain surfaces.



Mesh	Aspect ratio		Determinant (3x3x3)	
	Near Field	Far Field	Near Field	Far Field
Baseline	688.4	7.5	0.734	0.593
Morphed	688.6	7.9	0.731	0.577

Figure 14. Method performance: airfoil redesign

Critical to quality is the level of distortion added to the element during interpolation, which in turn is directly related to relative displacements of element vertices. Neighboring grid lines undergo the same interpolation law on similar arc lengths and similar nodal distribution along the grid lines. If the displacements of the neighboring end nodes on the surfaces of the domain are not drastically different, the interpolation process will result in an element distortion level that can be tolerated by the original element shape. A good quantitative measure to gauge whether a given displacement field can be handled by interpolation is the non-dimensional relative displacement metric. It is computed at the element level as the maximum nodal relative displacement divided by the average element length. (Note that this metric is directional – a separate value is computed for each of the three grid directions). The value of 1 would mean that the maximum relative displacement of nodes of the element equals the element size. This metric depends mainly the on the surface displacements slope. The choice of the interpolation law does have some influence but to a much lesser degree. During morphing, this relative displacement is "added" to the pre-existing element distortion of the baseline mesh, so the answer on how large displacements the morphing can handle depends on the quality of the originating mesh as well. Experience in using the proposed method shows that fields with maximum nondimensional relative displacement metric smaller than 0.25 are handled well during morphing, which spans many real-life CFD applications like optimization, airfoil flutter, wind turbine vibrations, and similar applications illustrated in Section 1 of this paper. The table in Figure 14 illustrates the performance of the method for the airfoil redesign example: changes to both determinant and aspect ratio metrics in the far-field mesh region are minimal.

As hinted above, the choice of interpolation law plays a role in the final mesh quality. But more importantly, the choice of the interpolation law determines the shape and smoothness of

the displacement field in the mesh domain. The simplest choice is a linear interpolation (by grid line length). It results in the C0 displacement field, with the slope discontinuities occurring at the boundaries of the domains defined in each iterative step. Higher-order interpolation can be used to ensure the desired level of smoothness of the displacement field. Figure 15 illustrates a sample grid line in the morphing region "1" of the first iterative step of the airfoil redesign example. The grid line connects the boundary region (marked as "0" with the periodic surface. The straight blue line depicts the displacement field distribution along the grid lines "1" (in region 1) and "0" (in boundary layer region) when linear interpolation is used and with the assumption of frozen (no motion) periodic surface. Dashed lines correspond to the motion of the respective region on the opposite side of the airfoil. The displacement field exhibits slope discontinuities at the interface to the boundary layer region and the location of the periodic boundary. For some applications, this discontinuity may not pose a problem. The morphed mesh may have a slightly higher or lower element growth ratio at locations of discontinuities, but this change may be within the variations already present in the original mesh. However, the smoothness of the displacement field might be of importance for other applications (e.g. fluid-structure interaction) where mesh motion plays a role in a numerical scheme. In such cases, a switch to higher-order interpolation might be needed. (shown as red curve in Figure 15).

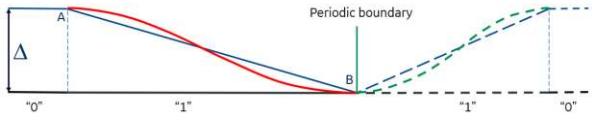


Figure 15: Interpolation law choice

4. UNSTRUCTURED MESH MORPHING

The morphing approach outlined in the previous section has several strong points. It robustly handles large mesh motion, is very fast, and scales linearly with mesh count. The main drawback is that it is limited to structured CFD mesh applications.

This section describes the extension of the above approach to morphing unstructured meshes. The goal is to retain the good sides of the method but to relax the requirement of structured baseline mesh. The main idea for the extension originates from the way how the structured mesh gets morphed. Special attention was paid to the boundary layer mesh region in the vicinity of walls, where relative mesh vertex motion is kept to a minimum, and prescribed displacement is diffused into the volume only as one moves away from the walls. This idea can be extended to unstructured meshes, provided the domain can be split into near and far-field regions.

To realize such a split, two actions need to be performed. A boundary between two regions needs to be defined, followed by a decomposition step.

Boundaries between near and far-field mesh regions should be defined automatically (any user intervention defeats the purpose of fast morphing). The shape of the boundary should be simple, not dependent on surface details, but resemble global model features to stay reasonably close to the structure. This is precisely the same set of criteria used in shrink-wrap technology where the idea is to walk over small features and retain only the global shape of the structure. For mesh split, the emphasis is on simplicity and automation. The length of surface offset used during shrink wrap is not of the primary concern. In the previous section, we used simple airfoil models to showcase the structured morphing approach. Imagine now that those airfoil models come with all the complexity of real hardware: tip squealers, cooling holes, vortex generators, tip shrouds, and part-span shrouds, to name a few. Inevitably, such models would need to be discretized in an unstructured manner. However, from a 10-foot distance, the model would still look like an airfoil. One can provisionally scan the baseline surface mesh model, remove all local features for example section by section), compute simplified cross-sections, and "wrap" all exotic dimples, undulations, and similar, by offsetting sufficiently enough outward. The shrink wrap technology is not new, and it is readily available in many commercial meshing packages today. Additionally, many structures of interest can be shrink-wrapped with very simple actions: airfoil wing and fuselage, wind turbines, and turbomachinery blades can all be approximated with primitive cylindrical structures.
Once the wrap surface is defined, domain decomposition can be used to separate the originating mesh into near-field (elements inside) and far-field (elements outside) regions.

A simple model of a swinging wrecking ball is used to illustrate the steps of the unstructured mesh morphing process. The baseline mesh model and prescribed displacement field are shown in Figure 16. The model consists of a ball hanging on a rope attached to a crane. The ball swings toward the crane as shown. Figure 17 illustrates the shrink-wrap model around the rope-ball model. Note that in a real application, rope and ball could be replaced with, for example, an aircraft wing with arbitrary geometry detail level. The shrink-wrap model could still be a circular cylinder. Mesh decomposition into the near-field region around rope (red), on-the-fence (white), and far-field mesh regions (light blue) is shown in Figure 18.

The Mesh morphing process splits into two stages. First, the nodes (elements) of the near-field mesh region are moved. Like in the case of structured mesh morphing, the goal is to preserve the mesh characteristics as much as possible. The approach, again, is to distribute surface displacements within the near-field region in a pointwise rigid way to minimize boundary layer mesh distortion.

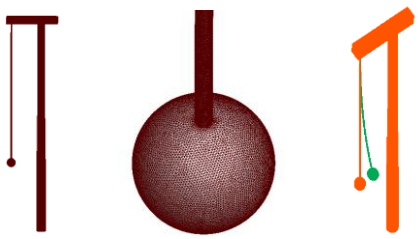


Figure 16. Swinging wrecking ball model

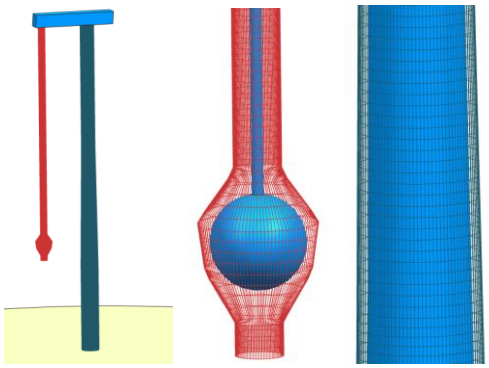


Figure 17. Shrink wrap of the ball-crane model

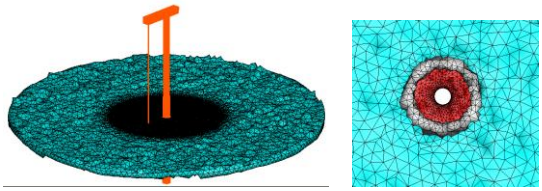


Figure 18. Mesh decomposition

Surface displacements are propagated through the prism boundary layer in the same way as for structured meshes by applying constant motion on the mesh line emanating from the surface and going through the thickness of the boundary layer. From there, displacements are propagated one element layer at a time until all elements of the inner mesh region are exhausted. In doing so, the vertices of the tetrahedral elements of the previous layer serve as the sources to compute the displacements of the vertices of the elements in the next layer. Various weighting schemes can be used to propagate the displacements in the outward direction, starting from simple unweighted averaging, but also considering local model curvature, element quality, distance proximity, etc. Layer structure and displacement propagation scheme for a cross-section of the rope in the ball model and simple airfoil model are shown in Figure 19.

After all vertices of the inner mesh region are assigned the displacement value, the morphing of the far field can start. Following the logic of the structured mesh morphing process, the outer boundary of the inner mesh region takes the role of the active region boundary. Unlike the structured mesh case, there is no underlying block structure and structured grid lines to serve as the interpolation medium.

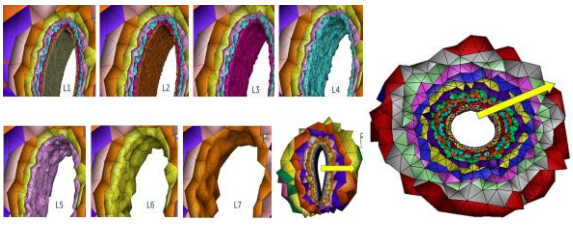


Figure 19. Layer progression for inner mesh displacement distribution

The shape of the domain boundary is simple (from the shrink wrap), and that boundary surface is readily discretized using structured surface mesh (see Figure 17 for the example). This is where another tool from the pre-processing suite – automated blocking generation – is used to generate structured background mesh over the far-field mesh region with shrink-wrap surface mesh at its boundary. Assuming such mesh is generated the following actions can be taken: i) Project displacements of the outer layer of the inner mesh region onto the shrink-wrap surface, ii) morph the structured background mesh to distribute the displacement field into the domain, iii) project computed displacement field from structured background mesh onto the nodes of the baseline unstructured mesh.

The steps listed above are analyzed in more detail.

At the heart of the process is the assumption that the structured background mesh can be readily generated. Indeed, the far-field mesh region has a simple shape (shrink-wrap boundary). Automated block generation methods can be used to chunk and generate background structured mesh. Such processes can be found in commercial packages that operate on turbomachinery CFD meshing. More sophisticated methods capable of discretizing complex domains are described in [10], [11], [12], and [13] among others. Using these methods, background structured mesh can be automatically generated for the target class of applications (airfoil turbomachinery, fuselage-wing, wind turbine, etc.). Background meh for the rope-ball model is shown in Figure 20. Note, that the structured mesh serves only as the background medium to compute and distribute the target displacement field into the far-field mesh region where the elements are typically large (much larger than in the near-field region – see Figures 18 and 19 for examples. These elements can tolerate larger relative nodal displacements, including numerical "imperfections" that might arise from using a coarser background mesh. The message here is that the background mesh does not need to be perfect in terms of resolution and smoothness. The resolution should be such to allow for a sufficiently smooth displacement field but can be much coarser than the original unstructured mesh.

Once the background mesh is available, the next step is to compute the displacements on its surface – the shrink-wrap surface. The shrink-wrap surface is fully covered by the outer layer of near-field region elements (white elements in Figure 18). Simple, element-wise projection can be used to compute the displacements for each of the nodes on the shrink-wrap surface. Note that this step is local (localized to each element of the interface), and explicit in nature. Displacements for each node on the shrink-wrap surface are computed by interpolating the inside mesh element of the original mesh that contains it.

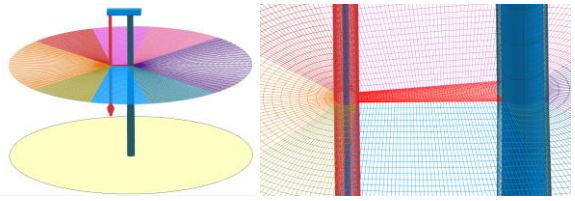


Figure 20. Structured background mesh

From there, the structured background mesh gets morphed. The method described in the previous section is used. This step is the key element in the process since allows for explicit, fast, and scalable displacement field computation. By switching to structured mesh to compute the displacement field, the time-consuming connectivity-related issues of unstructured meshes are bypassed, presented in the form of expensive implicit schemes for displacement computation.

Finally, the displacement field computed on the nodes of the structured meshes needs to be projected to the nodes of the original unstructured mesh. Like in the forward projection step done above, this process is also local and explicit in nature – no system of equations involved, just pure elementwise interpolation.

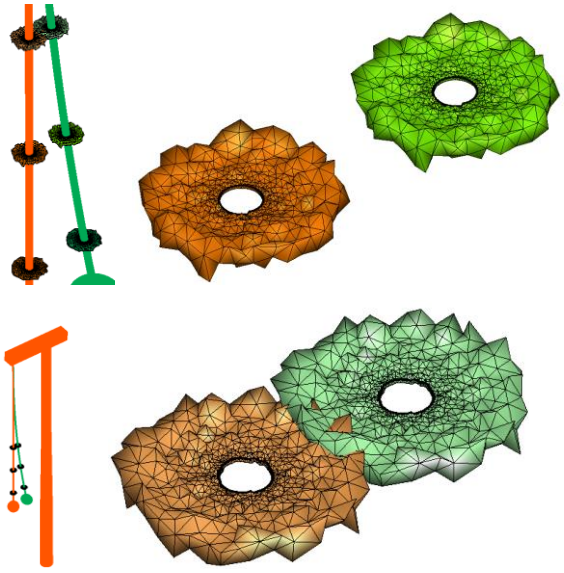


Figure 21. Near-field mesh morphing

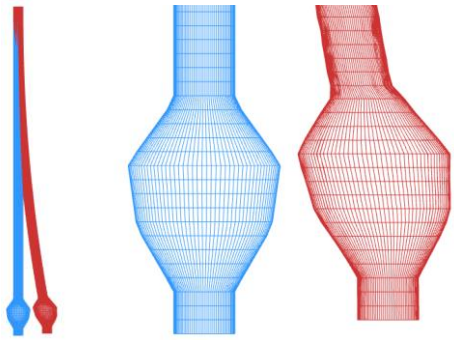


Figure 22. Displacement of the shrink-wrap mesh

Figures 21, 22, and 23 illustrate the steps outlined above for our swinging ball model. Figure 21 shows a comparison of the baseline and morphed meshes for a few cross-sections of the inner mesh region. The displacement of the shrink-wrap surface is shown in Figure 22, and the motion of the background mesh is shown in Figure 23. Finally, mesh motion for the far-field (outer) mesh region for a cross-section of the rope-ball model is shown in Figure 24.

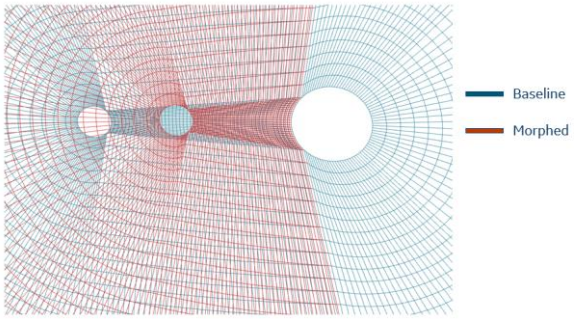


Figure 23. Motion of the background mesh

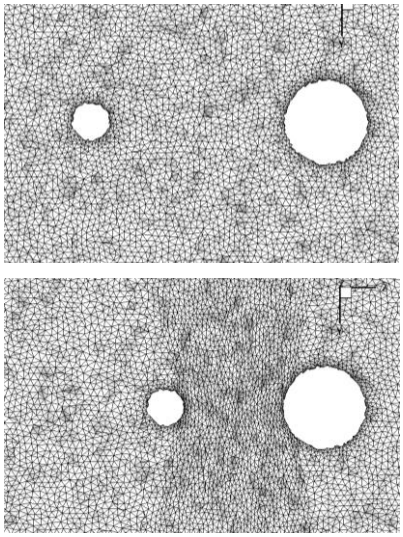


Figure 24. Baseline and morphed mesh comparison

5. CONCLUDING REMARKS

This presentation on mesh morphing has two goals. The first half of this paper discusses the elements of the mesh morphing environment, and the interaction of the morphing system with a wider set of pre-processing techniques. Mesh morphing can reach its full potential only when approached holistically. Several examples are shown demonstrating how such interactions lead to qualitatively new simulation capabilities.

In the second part, the focus is narrowed to volume mesh morphing. An explicit interpolation-based approach to the morphing of structured meshes is proposed. The method handles large displacements, has a fast run time, and scales linearly with mesh count. The proposed method is then extended to the morphing of unstructured meshes by combining it with several pre-processing techniques, demonstrating the benefit of the system-level approach.

REFERENCES

[1] M. L. Staten, S.J. Owen, S. M. Shontz, A.G. Salinger, T.S. Coffey, "A comparison of mesh morphing methods for 3D shape optimization". *Proceedings of the 20th International Meshing Roundtable*, pp. 293–311 (2011)

[2] M. Alexa, "Recent Advances in Mesh Morphing", *Computer Graphics Forum,* Vol 21(2), pp:173-192 (2002)

[3] A. de Boer, M.S. Van der Schoot, H. Bijl, "Mesh Deformation Based on Radial Basis Function Interpolation", *Computers & structures*, Vol 85(11), pp. 784-795 (2007)

[4] M.E. Biancolini, "Mesh Morphing and Smoothing by Means of Radial Basis Functions (RBF): A Practical Example Using Fluent and RBF Morph", *Handbook of Research on Computational Science and Engineering: Theory and Practice*, Vol 2 pp. 347-380 (2011)

[5] T. Rendall, C.B. Allen, "Efficient Mesh Motion Using Radial Basis Functions with Data Reduction Algorithms", Journa of Computational Physics Vol 228(17) pp:6231-6249 (2009)

[6] D. Seiger, S. Menzel, M. Botsch, "High Quality Mesh Morphing Using Triharmonic Radial Basis Functions", *Proceedings of the 21th International Meshing Roundtable*, pp. 1-15 (2012)

[7] M.E. Biancolini , A. Chiappa , U. Cella, E. Costa, C. Growth, S. Porziani," A Comparison Between the Bi-harmonic Spline and the Wendland C2 Radial Function", *Proceedings 2020 International Conference on Computational Science,* pp. 294-308 (2020)

[8] M.E. Biancolini , A. Chiappa , F. Giorgetti , S. Porziani, M. Rochette," Radial basis functions mesh morphing for the analysis of cracks propagation", *Proceedings of AIAS 2017 International Conference on Stress Analysis,* pp. 433-443 (2018)

[9] Myles Morelli, Tommaso Bellosta , Alberto Guardone, "Efficient Radial Basis Function Mesh Deformation Methods for Aircraft Icing", Journal of Computational and Applied Mathematics, Vol 392, (2021)

[10] David L. Rigby, "TopMaker: A Technique for Automatic Multi-Block Topology Generation Using the Medial Axis", NASA Technical Report NASA/CR—2004-213044 (2004)

[11] Damrong Guoy, Jeff Erickson, "Automatic Blocking Scheme for Structured Meshing in 2D Multiphase Flow Simulation", *Proceedings of the 13th International Meshing Roundtable*, pp. 121–132 (2004)

[12] I. Malcevic, "Automated Blocking for Structured CFD Gridding with an Application to Turbomachinery Secondary Flows, 20th AIAA Computational Fluid Dynamic Conference AIA 2011-3049 (2011)

[13] Harold J. Fogg, Cecil G. Armstrong, Trevor T. Robinson, "New techniques for enhanced medial axis based decompositions in 2-D", *Proceedings of the 23th International Meshing Roundtable*, pp. 162–174 (2014)