Techniques of Mesh Movement

Aditya Kashi
under the supervision of
Dr Hong Luo

December 2015

Abstract

This article describes methods of mesh movement for computational fluid dynamics (CFD) problems. The methods described can be classified into two families - elasticity-based methods and interpolation methods. The former class includes lineal and torsional spring analogy methods and linear elasticity method and its variants. Interpolation methods described here include the 'Delaunay graph' mapping, interpolation by radial basis functions, and a combination of Delaunay graph mapping and radial basis function interpolation. The merits and disadvantages of these methods have been discussed, and some two-dimensional test cases have been presented.

Contents

1 Introduction

Moving meshes are required in various problems in computational fluid dynamics (CFD), such as aeroelasticity, aerodynamic shape optimization [4] and some kinds of multi-phase/multi-component flow simulations. Mesh movement may also be used in generating curved meshes for spatially high-order computational methods [7]. In general, good properties of a mesh movement scheme are as follows.

1. It should be robust. In many kinds of meshes, such as meshes required for viscous flow computations, even small boundary movements can invalidate the mesh. The scheme should be able to preserve mesh validity, at the very least.

2. Mesh quality should be preserved to a great extent. Elements should not become highly distorted after movement, as this can impact flow computations on the deformed mesh.

3. It should be computationally inexpensive. Many applications need very fast mesh movement schemes as they require the mesh to be moved many times during the simulation, sometimes every time step.

Some mesh movement schemes that we present here satisfy only one or two of the above criteria. Our aim is to find a scheme that satisfies all three criteria well.

2 Review of methods

2.1 Elasticity methods

Elasticity-based methods are among the oldest mesh-movement methods in use. These include lineal spring analogy, torsional spring analogy, and various kinds of elastic solid mechanics analogies. These methods model the mesh as some kind of solid entity, impose a boundary displacement and solve equilibrium equations to propagate the motion to the interior nodes.

2.1.1 Spring analogies

Lineal spring analogy is the simplest method that can be used to move internal mesh points in response to movement of the boundary, invented by J.T. Batina. Each edge of the mesh is assumed to represent a lineal spring connecting the two nodes which make up the edge. The idea is that after imposing boundary movement, the mesh is allowed to "go to equilibrium". At every mesh point, we have

$$\sum_j k_{ij}(\Delta \mathbf{r}_i - \Delta \mathbf{r}_j) = \mathbf{0} \quad \forall i \tag{1}$$

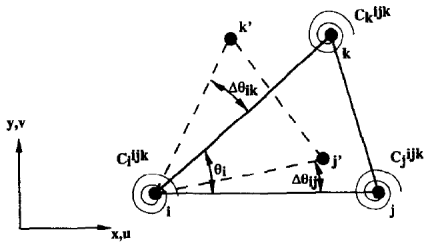

Figure 1: Movement of an element in torsion spring analogy

where i ranges over all nodes, j ranges over points surrounding node i and $\Delta\mathbf{r}_i$ is the displacement of node i. k_{ij} is the stiffness of the spring between nodes i and j, which can be taken as

$$k_{ij} = \frac{1}{||\mathbf{r}_i - \mathbf{r}_j||}. \tag{2}$$

This method is not reliable as it generates invalid elements (having zero or negative value of the Jacobian determinant of the reference-to-physical element mapping) for even relatively small displacements, depending on the mesh. However, for the application it was developed, it sometimes gives acceptable results. It is used, for example, by Mavriplis *et. al.* [9] in an optimization problem involving inviscid flow only. It is also less computationally expensive compared to other elasticity-based methods.

Note that separate problems are solved in each coordinate direction, which are uncoupled. It is the author's opinion that this is the main reason for its unreliability for many kinds of mesh and movement combinations. However, because of this, extension to 3D is trivial.

The torsional spring analogy is an extension to the previous method, which adds torsional springs at each node in each element (see Farhat *et.al.* [2]). These torsional springs resist the relative angular motion between edges of an element. This leads to a substantially more robust movement. In the scheme of Farhat *et. al.*, the problem is divided into two parts, one related to lineal spring analogy and one to the torsional spring analogy. The lineal spring model is not the same as that used by Batina; in this case the lineal spring model also leads to a coupled system which is more robust than the uncoupled scheme. The torsional part adds even more resistance to invalid elements. As shown in figure 1, the torsional spring analogy adds torsion springs at each node of each element. The stiffness of a torsion spring associated with a node of a certain element is a function of the angle formed by the edges of that element meeting at the node.

While this method is significantly more robust than the lineal spring analogy of Batina, it is also much more computationally expensive. A fully coupled system of $n_{dim}N_n$ equations must be solved, where N_n is the number of nodes that are to be moved in the mesh and n_{dim} is the dimension of the problem.

2.1.2 Elastic solid analogy ('Elasticity')

The deformable elastic solid analogy, employing the equations of (linear or non-linear) elasticity, is one of the widely-used mesh movement techniques. This class of schemes often lead to very robust mesh movement. The mesh is assumed to model a deformable solid body, which is then deformed according to the equations of solid mechanics, that is, linear or non-linear elasticity.

The simplest approach is linear elasticity.

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{0} \quad \text{in}\, \Omega \tag{3}$$

Assuming an isotropic material, the constitutive relation can be taken as

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \lambda(\text{tr}\boldsymbol{\epsilon})\boldsymbol{I} \tag{4}$$

where $\boldsymbol{\sigma}$ is the stress and $\boldsymbol{\epsilon}$ is the strain. Finally, the linear strain-displacement relation is given by

$$\boldsymbol{\epsilon} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \tag{5}$$

where $\boldsymbol{u}$ is the displacement field, with Dirichlet boundary conditions (prescribed boundary displacement $\boldsymbol{u}_b$)

$$\boldsymbol{u} = \boldsymbol{u}_b \quad \text{on}\, \partial\Omega \tag{6}$$

The weak form of this set of equations, solved by Galerkin finite element method (FEM) leads to a linear system of size $n_d N_n$, where n_d is the dimension of the problem and N_n is the number of nodes to be moved.

While the basic linear elasticity scheme is adequate for some applications, it is not adequate for stretched, boundary-layer meshes for viscous flow, among others. The scheme is often modified by 'stiffening' the mesh appropriately, so as to attain some control over the propagation of deformation into the interior of the mesh, as done, for instance, by Hartmann and Leicht [3]. In their work, the material is stiffened based on the determinant of the local Jacobian matrix of the reference-to-physical mapping, that is to say smaller elements are stiffer than larger ones.

Non-linear elasticity is claimed to be a highly robust method for mesh movement by Persson and Peraire [7]. As long as the mesh is fine enough to resolve the displacements, they claim that element validity is guaranteed. In their work, for non-linear elasticity, the constitutive equation (4) and strain-displacement relation (5) are replaced by the 'neo-Hookean' constitutive model

$$\boldsymbol{P} = \mu((\boldsymbol{F}^T\boldsymbol{F})\boldsymbol{F}^{-T} - \boldsymbol{F}^{-T}) + \lambda(\ln \det \boldsymbol{F})\boldsymbol{F}^{-T} \tag{7}$$

where the deformation gradient $\boldsymbol{F}$ is given by

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}.$$

Here, $\boldsymbol{x}$ is the physical position vector of a point with coordinate $\boldsymbol{X}$ in the reference configuration. The system is solved using Newton-GMRES iterations.

In our opinion, this method probably cannot be used in any unsteady moving-geometry simulations, as the cost will be prohibitive. However, it leads to good results for curved mesh generation, as can be seen in Persson and Peraire's work.

The advantage of elasticity-based methods is that they can be made highly robust. Their disadvantage is that generally, the more robust the scheme, the more expensive it is. The cost of the system of equations that needs to be solved usually scales with the total number of mesh nodes to be moved; this can get very expensive for 3D viscous meshes. Further, the implementation is usually not very easy; different cell types require different treatments (different basis functions, for instance, in case of elastic-solid approaches) and extension from 2D to 3D may not be trivial (as in case of torsional springs).

2.2 Interpolation Methods

Also called algebraic methods, these methods do not involve any physics-based modeling of the mesh. They are purely mathematical manipulations of the mesh. Unlike elasticity-based methods, schemes of this type tend to be fast and independent of mesh topology.

2.2.1 Delaunay graph mapping (DGM)

Developed by Liu, Qin and Xia [6], DGM is a fast method of mesh movement. It consists of the following steps.

- A Delaunay triangulation is constructed from the boundary points of the mesh. This triangulation is referred to as the Delaunay graph.
- Each internal mesh point is located in the Delaunay graph, and its barycentric coordinates are calculated with respect to the Delaunay element it lies in.
- Using the prescribed boundary displacements, the Delaunay graph is moved.
- Using the calculated barycentric coordinates, the interior mesh points are mapped back to the deformed Delaunay elements keeping the barycentric coordinates constant, thus moving the mesh.

Thus, no linear system needs to be solved. The bulk of the work is in computing the Delaunay triangulation and traversing it, which can be done efficiently. This method results in a valid mesh as long as the boundary displacement does not invalidate the Delaunay graph itself. If the Delaunay graph becomes invalid, the displacement must be carried out in several steps. The Delaunay graph is re-computed each step, and thus much more boundary movement is possible without invalidating the sequence of Delaunay graphs.

In our experiments, Delaunay graph mapping is not very robust when it comes to rotational motion, or otherwise very large deformation (see the results section).

2.2.2 Radial basis functions

Radial basis functions (RBFs) are radially-symmetric functions. They are used as a basis for the displacement field in the moving mesh [1]. Considering n_b boundary points in the mesh, we express the displacement as

$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{n_b} \mathbf{a}_j \phi(\|\mathbf{x} - \mathbf{x}_{bj}\|) \tag{8}$$

where $\mathbf{s}$ represents the displacement field, $\mathbf{x}$ is the position vector of any point in the domain, $\mathbf{x}_{bj}$ is the position of the ith boundary point, ϕ is a radial basis function, $\phi(\|\mathbf{x} - \mathbf{x}_{bj}\|)$ is the jth basis function for the displacement field, and $\mathbf{a}_j$ is the jth coordinate or coefficient in that basis.

Since we know the displacements of the boundary nodes, we can solve for the coefficients $\mathbf{a}_j$ using

$$\mathbf{s}(\mathbf{x}_{ib}) = \sum_{j=1}^{n_b} \mathbf{a}_j \phi(\|\mathbf{x}_{bi} - \mathbf{x}_{bj}\|). \tag{9}$$

In each coordinate direction, this leads to a system of n_b equations in n_b unknowns:

$$\mathbf{A}\mathbf{a}_k = \mathbf{s}_k \tag{10}$$

where the (i, j) component of $\mathbf{A}$ is $\phi(\|\mathbf{x}_{bi} - \mathbf{x}_{bj}\|)$, $\boldsymbol{a}_k \in \mathbb{R}^{n_b}$ is the coefficient vector in the kth direction and $\boldsymbol{s}_k \in \mathbb{R}^{n_b}$ is the vector of boundary displacements in the kth coordinate direction.

In our implementation, we assemble $\mathbf{A}$ as a sparse matrix by evaluating the radial basis function between every pair of boundary points. This process can be parallelized easily.

Next, we need to evaluate equation (8) for each interior point, which requires one evaluation of the RBF for each interior point (in each direction). This too, can be parallelized easily.

The quality of the final mesh depends on which RBF is used. Many kinds of RBFs have been used in literature [1, 8]. In this work, we mainly use Wendland's C2 function

$$\phi(x) = (1 - x)^4(4x + 1), \tag{11}$$

or rather, a modification,

$$\phi(x) = \begin{cases} \left(1 - \frac{x}{r_s}\right)^4 \left(4\frac{x}{r_s} + 1\right) & x < r_s \\ 0 & x \geq r_s \end{cases} \tag{12}$$

where r_s is a real number called the 'support radius'. This modified function has a compact support. In light of this, we see the matrix $\mathbf{A}$ is sparse if the support radius is less than the

characteristic dimension of the domain. For curved mesh generation, this support radius can be kept quite small relative to the whole domain, giving us a very sparse linear system which is solved quite fast.

The scaling of the argument by the support radius serves to make the value of the RBF 1.0 at the boundary. This means that points very close to the boundary will deform just like the boundary itself. This results in an essentially rigid motion of points very close to the boundary, which causes the near-boundary elements to retain their quality after the mesh movement. The support radius is chosen big enough to accommodate the expected deformation of the boundary, but small enough that the linear system is sparse. An algorithm to determine a good support radius will be very useful in this regard.

2.2.3 Interpolation using radial basis function on the Delaunay graph (DGRBF)

This method proposed by Wang, Qin and Zhao [10] combines both the Delaunay graph mapping and interpolation by RBF. The general scheme of the mesh movement is the same as in case of DGM, but with the important difference that interpolation is not done using barycentric coordinates of the nodes with respect to the Delaunay graph elements. Here, the interpolation is done via radial basis functions in each Delaunay graph element. The displacement of a node with initial position $\boldsymbol{x}$ is given by

$$\mathbf{s}(\mathbf{x}) = \sum_{j=1}^{n_t} \mathbf{a}_j \phi(\|\mathbf{x} - \mathbf{x}_{tj}\|) \tag{13}$$

where $\boldsymbol{x}_{tj}$ are positions of nodes of the Delaunay simplex that contains the node at $\boldsymbol{x}$, and n_t is the number of nodes in that simplex (3 in 2D and 4 in 3D). We need to solve for the $\boldsymbol{a}_j$, the RBF coefficients, by solving a 3×3 system in 2D or a 4×4 system in 3D, in each Delaunay element.

$$\mathbf{s}(\mathbf{x}_{ti}) = \sum_{j=1}^{n_t} \mathbf{a}_j \phi(\|\mathbf{x}_{ti} - \mathbf{x}_{tj}\|). \tag{14}$$

DGRBF method provides flexibility in choosing the RBF and the support radius, and with a good choice of these, robust mesh movement can be achieved for translational motion and possibly some other kinds of deformation. It has been observed in [10] and in our tests, that interpolation of displacements by DGRBF often does not give very good results for large rotational deformations. The remedy for this is to interpolate the rotational angles instead of the displacements directly, and then calculate the displacements at each interior node using these interpolated rotation angles using the rotation matrix. For example, in 2D,

$$x_{new} = (x - x_0)\cos a_z - (y - y_0)\sin a_z + x_0 \tag{15}$$

$$y_{new} = (x - x_0)\sin a_z + (y - y_0)\cos a_z + y_0 \tag{16}$$

where (x_0, y_0) is the center of rotation, and a_z is the interpolated rotation angle at the node in question. For interpolation of the angles, the same formulae (13) and (14) are used, with the rotation angles replacing the displacements. Angle interpolation can be combined with displacement interpolation for problems involving both translation and rotation. This angle interpolation scheme, and its combination with the displacement interpolation scheme, is referred to as 'DGRBF2' in [10].

2.3 Mesh quality

In order to judge the effectiveness of mesh-movement methods, we need to measure the quality of the deformed mesh. Some mesh quality measures have been derived for linear 2D and 3D elements by Knupp [5]. These include the relative size, shape and size-shape metrics for triangles and tetrahedra, and size, shape, skew, size-shape and size-skew metrics for quadrangles and hexahedra.

The size metric distinguishes elements having very small or very large volume with respect to some reference volume. The shape metric identifies elements that have unequal edges or unequal angles between edges; it is independent of size of the element. Finally, the skew metric is a measure of unequal angles between edges of the element only, irrespective of lengths of edges and volume of element; it is different from shape in case of non-simplicial (like quadrangular and hexahedral) elements. While dealing with quadrangles, we use the skew metric.

All metrics are normalized so that they are 1.0 for ideal elements with regard to the characteristic they measure, and they are 0.0 for degenerate elements.

3 Results

We present a case of rotation of an interior object inside a far-field boundary using interpolation methods. We show results for a 60-degree rotation in case of DGM (fig. 3), RBF (fig. 4) and DG-RBF2 (fig. 5) methods. Note that DGRBF2 refers to the interpolation of rotation angles by DGRBF. The case has been referred from Wang *et. al.* [10]. Figure 2 shows the un-deformed mesh.

We note that the RBF and DGRBF2 methods can handle the rotation case quite well. The stretched, fine elements near the inner boundaries are well-preserved, while the elements further in the interior, which are larger and can thus take more deformation, are distorted somewhat. Though RBF gives slightly better mesh quality than DGRBF2 for this amount of rotation, RBF takes 1.926 seconds of total CPU time (using conjugate gradient solver), while DGRBF2 takes only 0.08 seconds of total CPU time. Further, shown in figure 6 is a case of extreme rotation of 120 degrees, which DGRBF2 solves while maintaining mesh validity. None of the other methods are able to do this. RBF has been found to give valid meshes upto about 110 degrees of rotation.

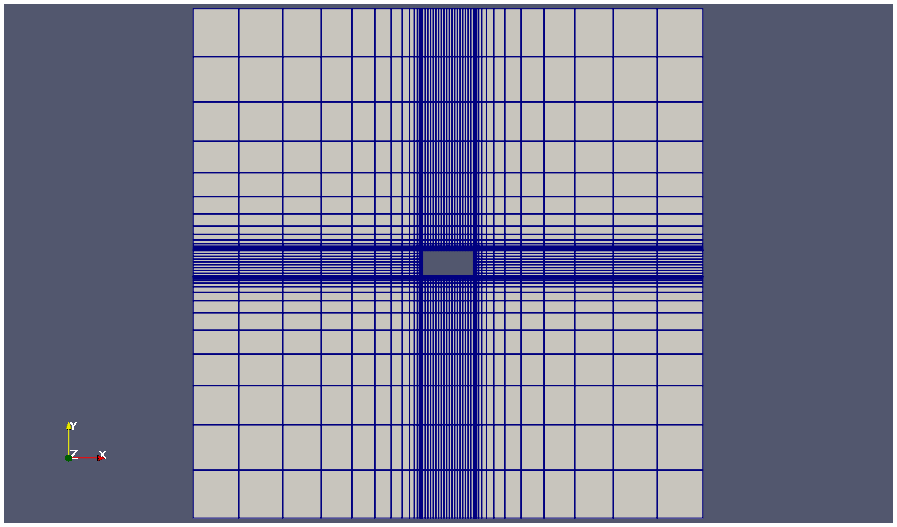

Figure 2: Original mesh

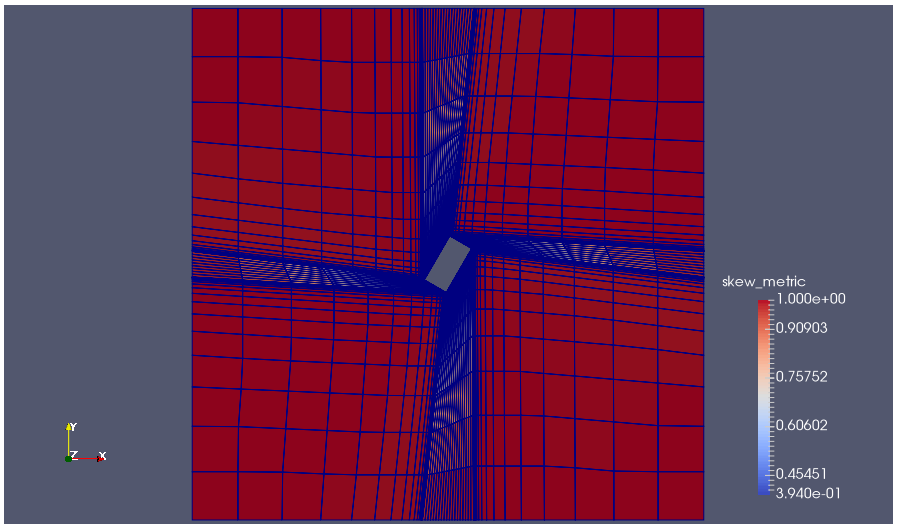

Figure 3: 60 degrees rotation by DGM

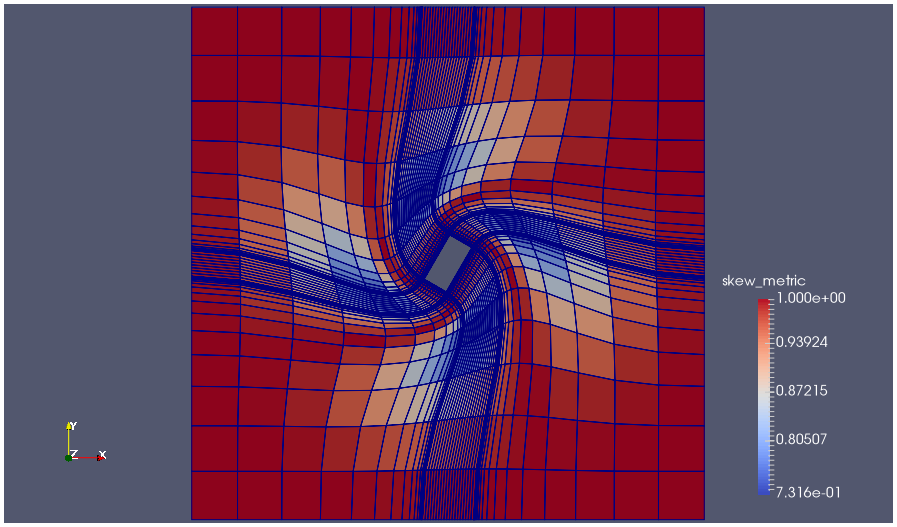

Figure 4: 60 degrees rotation by RBF

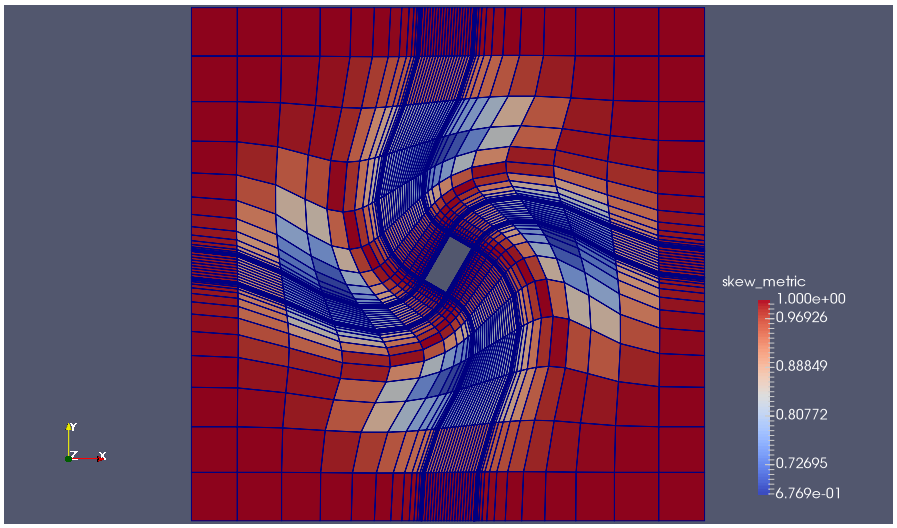

Figure 5: 60 degrees rotation by DGRBF2

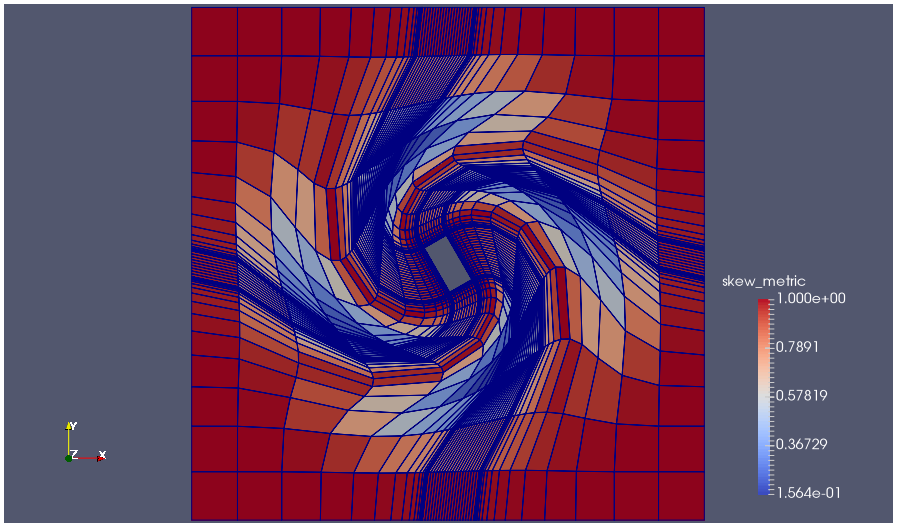

Figure 6: Large rotational motion carried out by DGRBF2

While solving certain problems with the pure RBF method, we find that the linear system to be solved is sometimes quite ill-conditioned, depending upon the mesh. However, in some of the cases that we worked on (such as curved mesh generation of hybrid meshes for viscous simulations of airfoils), sparse direct methods are capable of solving the system efficiently and effectively. This is likely to hold true in general as the systems to be solved will usually not be very large (having size of the number of boundary nodes).

4 Conclusion

A lot of choice is available for mesh-movement techniques, but only a few will be good for a given application. We conclude that interpolation methods are generally well-suited for unsteady simulations that require mesh movement, as they are very fast. When deformations are relatively small and not highly rotational, DGM is a good choice while when larger and more general deformations are needed, DGRBF is best. The pure RBF method, while usually giving good robust results, is more expensive than other interpolation methods considered here.

RBF, and elastic solid analogy methods, are found to be effective for curved mesh generation, where computational cost is less of an issue. We have used RBF for curved mesh generation with good results for certain viscous flow cases.

References

[1] A. de Boer, M.S. van der Schoot, and M. Bijl. Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85:784–795, 2007.

[2] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer methods in applied mechanics and engineering*, 163:231–245, 1998.

[3] Ralf Hartmann and Tobias Leicht. High-order unstructured grid generation and discontinuous Galerkin discretization applied to a 3D high-lift configuration. 53$^{\text{rd}}$ AIAA Aerospace Sciences Meeting (SciTech 2015), AIAA 2015-0819, Kissimmee, Florida, 2015.

[4] S. Jakobsson and O. Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape opitimization. *Computers & Fluids*, 36:1119–1136, 2007.

[5] P.M. Knupp. Algebraic mesh quality measures for unstructured initial meshes. *Finite Elements in Analysis and Design*, 39:217–241, 2003.

[6] X. Liu, N. Qin, and H. Xia. Fast dynamic grid deformation based on Delaunay graph mapping. *Journal of Computational Physics*, 211:405–423, 2006.

[7] P.-O. Persson and J. Peraire. Curved mesh generation and mesh refinement using lagrangian solid mechanics. In *47th AIAA Aerospace Sciences Meeting*, 2009.

[8] T.C.S. Rendall and C.B. Allen. Efficient mesh motion using radial basis functions with data reduction algorithms. *Journal of Computational Physics*, 228:6231–6249, 2009.

[9] L. Wang, D.J. Mavriplis, and W.K. Anderson. Adjoint sensitivity formulation for high-order discontinuous Galerkin discretizations in unsteady inviscid flow problems. *AIAA journal*, 48(12), 2010.

[10] Y. Wang, N. Qin, and N. Zhao. Delaunay graph and radial basis function for fast quality mesh deformation. *Journal of Computational Physics*, 294:149–172, 2015.