A DIRECT METHOD OF GENERATING QUADRATIC CURVILINEAR TETRAHEDRAL MESHES USING AN ADVANCING FRONT APPROACH

Fariba Mohammadi[1] Suzanne M. Shontz[2]

[1]*Department of Mechanical Engineering, Information and Telecommunication Technology Center, The University of Kansas, 1520 W. 15th Street, Lawrence, KS, 66045, USA, fariba_m@ku.edu*
[2]*Department of Electrical Engineering and Computer Science, Bioengineering Program, Information and Telecommunication Technology Center, University of Kansas, 1520 W. 15th Street, Lawrence, KS 66045, USA, shontz@ku.edu*

ABSTRACT

Computational modeling and simulation of real-world problems, e.g., various applications in the automotive, aerospace, and biomedical industries, often involve geometric objects which are bounded by curved surfaces. The geometric modeling of such objects can be performed via high-order meshes. Such a mesh, when paired with a high-order partial differential equation (PDE) solver, can realize more accurate solution results with a decreased number of mesh elements (in comparison to a low-order mesh). There are several types of high-order mesh generation approaches, such as direct methods, *a posteriori* methods, and isogeometric analysis (IGA)-based spline modeling approaches. In this paper, we propose a direct, high-order, curvilinear tetrahedral mesh generation method using an advancing front technique. After generating the mesh, we apply mesh optimization to improve the quality and to take advantage of the degrees of freedom available in the initially straight-sided quadratic elements. Our method aims to generate high-quality tetrahedral mesh elements from various types of boundary representations including the cases where no computer-aided design files are available. Such a method is essential, for example, for generating meshes for various biomedical models where the boundary representation is obtained from medical images instead of CAD files. We present several numerical examples of second-order tetrahedral meshes generated using our method based on input triangular surface meshes.

Keywords: high-order mesh generation, advancing front, tetrahedral meshes

1. INTRODUCTION

The use of high-order partial differential equation (PDE) solvers for various computational mechanics problems have seen an increase in recent years due to their ability to deliver more accurate results at a lower computational cost [1, 2, 3]. However, to obtain such results, these solvers must be paired with a high-order mesh while working with geometries that involve curved boundaries [4, 5]. Most real-world problems, such as various applications in the automotive [6, 7], aerospace [8, 9], and biomedical [10, 11] industries, involve geometries containing such curved surfaces. Computational modeling and simulation of these problems would greatly benefit from the use of high-order meshes, giving more accurate results with a decreased number of mesh elements compared to using its low-order counterpart. Since high-order meshes can be composed of both curved and straight elements, using these meshes ensures that the various curves and features present in the geometries are preserved and well-captured in the corresponding mesh. Thus, robust and high-quality high-order meshes are an essential part of accurate and efficient computational simulations.

There are two main categories of methods that can be used to generate a high-order mesh. The first category is known as *a posteriori* methods [4, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. This is the most common approach for generating high-order meshes. These methods consist of three main steps: first, a low-order mesh of the geometry is generated and additional nodes are added to the low-order mesh. Next, the newly-added boundary nodes are projected onto the curved domain so that they conform to the boundary, and finally the new positions of the interior nodes are computed, followed by moving these nodes to their updated positions. The mesh topology is maintained throughout the process. There are two popular techniques for deformation of the low-order mesh. The first one involves optimizing an objective function [13, 12, 15, 16, 18, 19, 21]. The second technique is based on the solution of PDEs, e.g., a linear elasticity approach [20], a nonlinear elasticity approach [4], a thermo-elastic analogy based approach [23], or other strategies [14]. There are challenges associated with the *a posteriori* approach. For example, often the boundary transformation step can create tangled elements in the mesh resulting in an invalid mesh. Also, with this approach, it is required to possess the geometry of the desired high-order mesh to represent the curved boundary. Such geometries are often attained from computer-aided design (CAD) files which are not always available, for example, in case of patient-specific anatomical models.

Direct methods make up the second category of high-order mesh generation approaches. With this approach, a high-order mesh is generated directly from a curved geometry. In earlier work [24], we developed a high-order triangular mesh generation method based on an advancing front technique. To the best of our knowledge, this is the only direct method for high-order mesh generation which generates the meshes from a high-order edge mesh of the boundary. In addition, it generates the high-order meshes for use with the finite element method.

Other geometric approaches have been developed for use with PDE solvers. Such examples include isogeometric analysis (IGA) [25, 26], Non-Uniform Rational B-splines (NURBS)-Enhanced Finite Element Method (NEFEM) [27], and Conforming to Interface Structured Adaptive Mesh Refinement (CISAMR) [28, 29]. IGA aims to integrate analysis and design by employing the same basis functions to describe the geometric representation and numerical simulation, thus making it possible to perform analysis directly on CAD models [30, 31]. The mesh used in IGA is generated by parameterization of the geometric domain over which the PDE is posed. This is different than the direct approach of mesh generation mentioned above, where the mesh and the geometry are separate entities. Two of the most popular techniques used in IGA are NURBS and T-splines [25, 26, 31]. Similarly the objective of NEFEM is the integration of the CAD boundary representation of the domain and FEM [27]. In contrast with the approach taken in IGA and NEFEM, we aim to generate tetrahedral meshes for FEM using direct methods. To date, there are no direct high-order tetrahedral mesh generation methods available for use with the traditional FEM approach.

In this paper, we will discuss our direct method for generation of high-order tetrahedral meshes using an advancing front technique, which extends our earlier work in [24]. Among the many algorithms that have been developed for generating unstructured 3D meshes over the years, the Delaunay tetrahedralization methods and the advancing front-based methods are the most popular [32]. Our mesh generation method uses the advancing front approach [33, 34, 35, 36, 37, 38] to generate high-order tetrahedral meshes for geometric models in which the surface is represented by high-order triangular elements. To utilize the degrees of freedom available in each element, and to improve the quality of the meshes, we then employ a mesh optimization method. To this end, we use the optimization algorithm for regularization of high-order elements available in Gmsh [17, 39]. Our current implementation can be used to generate curvilinear quadratic tetrahedral meshes. However, the algorithm can also be used to generate curvilinear tetrahedral meshes of higher-order. We plan to extend our implementation to handle such cases as part of our future work. Our present focus is on generating curvilinear quadratic tetrahedral meshes for geometries that are represented by quadratic triangular surface meshes with uniform element size distribution.

The novelty of our work lies in our method's ability to generate high-order meshes directly from curved boundaries. Since our method uses a direct approach instead of an *a posteriori* approach, it can generate meshes for various biomedical applications where patient-specific models are obtained from medical images and no CAD representation is available. One such example is cardiovascular modeling, where the use of high-order curvilinear meshes would aid in capturing the various features and curves present in the heart. Various research groups in recent years have used different high-order mesh generation schemes to generate cardiovascular meshes [10, 11]. In [10], cubic Hermite and cubic Lagrange cardiac ventricular meshes are generated, with the final meshes having 1, 12, and 6 and 2, 9, and 8 elements in radial, circumferential, and longitudinal directions, resulting in a fairly coarse mesh. The mesh generation technique used in [11] involved a manual vertex placement strategy to generate cubic Hermite meshes. In contrast to these techniques, using our method, it is possible to generate quadratic tetrahedral biomedical meshes where the mesh is fine enough to capture the boundary representation properly and does not involve manual intervention.

In Section 2, we present our mesh generation method. Results of our high-order tetrahedral mesh generation method on several geometries are shown in Section 3. Finally, in Section 4, we summarize our results and discuss several possibilities for future work.

2. HIGH-ORDER TETRAHEDRAL MESH GENERATION

In this section, we describe our high-order curvilinear tetrahedral mesh generation algorithm using an advancing front approach.

2.1 Initial front setup

In the proposed algorithm, we start with a high-order triangular surface mesh as input and use it to generate a curvilinear high-order tetrahedral mesh. The input triangular surface mesh can be based on various types of boundary representations, e.g., from CAD files or patient-specific boundary meshes obtained from medical images, e.g. MRI or CT scans. To generate the tetrahedral mesh, first we assign the triangular surface mesh as the initial active front. Each triangular face is considered an element of the active front. First, we check to ensure that the active front is not empty. Assuming it is not empty, we start with the first triangular element in the active front. As the method progresses and new elements are generated, we update the active front by deleting the triangular faces that are already used to generate the tetrahedral elements and by adding the new faces that are created after generating the tetrahedra.

2.2 Calculating reference height

Next, we calculate the area of the surface triangles. Since the surface triangles are curved, we need to use shape functions for the high-order elements to calculate the area. For a second-order mesh, we use the shape functions for a two-dimensional second-order Lagrange element and high-degree Gaussian quadrature rules developed by Dunavant [40]. We use a polynomial of degree 6 with 12 Gaussian points and weights. We calculate the area A_j of each triangular element using the Jacobian of the transformation in two dimensions. Next, we average the areas of all the curved triangular elements and denote this average area by A_{avg}. We use this average area to calculate a reference height h of a tetrahedron using the following equation:

$$h = \frac{2\sqrt{2}}{\sqrt{3}\sqrt[4]{3}} A_{avg}, \tag{1}$$

Equation 1 calculates the height of a regular/equilateral tetrahedron using the area of one of its triangular faces. The constant value in equation 1 comes from the properties of an equilateral tetrahedron [37]. Our aim is to generate tetrahedral elements as close to a regular tetrahedron as possible. Calculating the height in this manner helps us to ensure this goal. We use the average area instead of individual, local triangle areas to calculate this reference height. This helps us to maintain a uniform search radius and a uniform element size distribution. This matches well with our current focus which is the generation of high-order, curvilinear tetrahedral meshes for geometries whose surface meshes mainly have uniform element size distribution.

2.3 Searching for candidate vertices

Once we have calculated h, we begin the process of generating the tetrahedral elements from the surface mesh. To this end, we first need to search for candidate vertices within a specific search radius. In a high-order triangular surface mesh, each triangular element has both low-order and high-order vertices. The low-order vertices are the three endpoints of the triangle edges. The high-order vertices depend on the degree of polynomial of the triangular element. For a quadratic element, high-order vertices are the midpoints of each edge. Hence, only the low-order vertices are considered while searching for candidates to serve as the fourth vertex of the tetrahedron. We start with selecting the first triangular face from the active front and calculate the centroid c of that element. Next, we calculate the inward surface normal of the same element. Since the normal of the triangle face is the normal of its tangent plane, we first calculate the tangent plane to the triangular face and use that to calculate the surface normal. We use the centroid as a reference point and insert a vertex a at a distance h from the centroid of the selected triangular face. The vertex is inserted in the direction of the inward normal vector of the surface mesh. Next, we search for other suitable candidate vertices within a specific radius, $r = \alpha h$, of a. Here, α is a user-defined constant value that can be varied according to the size of the geometry and mesh. If α is set as 1.5, the search radius would be 1.5 times the length of h. The choice of α depends on the how coarse or fine the mesh is. If the users wish to search within a larger radius, they may increase the value of α, if they wish to search within a smaller radius, they can reduce the value of α. As a general guideline, the coarser the mesh is, the larger the search radius should be in order to capture enough candidate vertices. Once we set an

α, we use it for all the elements of that particular mesh.

While generating a tetrahedron, initially the candidate vertices consist of vertex a and the low-order vertices that are within radius r of the selected triangular face. Since a fixed h is used to search and generate all tetrahedral elements, element size uniformity throughout the mesh is ensured. Currently in Matlab, we are using the findNearestNeighbors function to search for the nearest neighbors from a point cloud of low-order vertices. Figure 1 shows an example of various low-order vertices present in the mesh which give the several possible candidate tetrahedra.

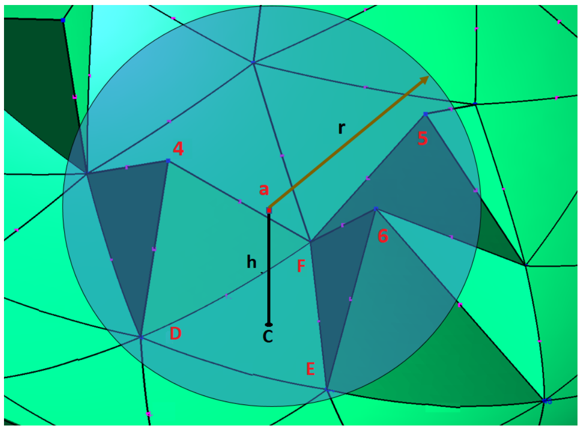


Figure 1: Example of various vertices present in the mesh giving possible candidate tetrahedra. Here, a is the newly added vertex that is inserted at a distance h from centroid c of triangular surface DEF. Vertices 4,5, and 6 are pre-existing vertices in the mesh that fall within search radius r.

2.4 Vertex selection

For each candidate tetrahedron C_i, $i = 1, 2, ..., n$, we perform several validity checks and quality metric calculations to ensure we generate the best possible tetrahedron among from the available candidate vertices. We describe the validity and quality checks in detail in Sections 2.5 and 2.6. The candidate tetrahedron that passes all validity and quality checks is selected as the final tetrahedron for that surface triangular face. If no such candidate is available, we perform several mesh modification operations around the selected triangular face to obtain a valid tetrahedral element. The mesh modification operations include local remeshing, e.g., removing the tetrahedron near the triangular face and remeshing the resulting void, 2-3 swap, 3-2 swap, 2-2 swap, and edge contraction [37, 41, 42, 43].

The fourth vertex of a generated tetrahedron could be either a new vertex or a pre-existing vertex. Once the low-order vertex is finalized, we generate the high-order nodes for the newly-created triangular faces of the tetrahedron. For a second-order tetrahedron, the high-order nodes will be the midpoints of each edge of its triangular faces. Next, we update the active front by deleting the initially selected triangular face and adding the newly-created triangle faces. These steps are repeated until our active front is empty. Algorithm 1 gives the pseudocode for our high-order curvilinear tetrahedral mesh generation method.

Algorithm 1: High-order tetrahedral mesh generation

```
Input: Second-order curvilinear triangular
  surface mesh as active front
Output: Second-order curvilinear tetrahedral
  mesh
Calculate $A_j$, $A_{avg}$, and $h$
if active front is not empty then
    for each triangular face do
        if the face exists in the active front then
            1. Calculate the inward surface normal
            2. Calculate the position of the
              centroid $c$ of the triangular element
            3. Insert a vertex $a$ in the inward
              direction at distance $h$ from $c$
            4. Search for more candidate vertices
              within radius $r$ of $a$
            5. Run element validity tests as
              described in Section 2.5
            6. Run element quality tests as
              described in Section 2.6
            if no suitable candidate element
              present then
                7. Perform mesh modification
            end
            if tetrahedron selection criteria are
              met as shown in Algorithm 2 then
                8. Generate tetrahedron
            end
            9. Insert high-order nodes
            10. Update active front
        end
    end
end
11. Optimize the vertices in the final tetrahedral
  mesh
```

2.5 Validity Tests

Once we have obtained all the candidate tetrahedra $C_i, i = 1, 2, ..., n$, we start performing the validity checks on the candidates. We conduct the checks in the specified order, so that we can remove the unsuitable candidates one-by-one and preserve the best candidates. Once a test is performed, the candidate tetrahedra are updated accordingly, e.g., if a candidate tetrahedron fails a test, that candidate is discarded, and the remaining candidates are considered for the next test.

Degenerate and inverted element test: This test checks the volume of a candidate tetrahedron to ensure that it is nonzero (indicating the element is not degenerate) and nonnegative (indicating the element is not inverted). For this test,we use the shape functions for a second-order Lagrange tetrahedron and the 11-point Gauss-Lobatto rule [44]. We perform this test on each candidate tetrahedron. If a candidate tetrahedron has four vertices that are nearly coplanar or collinear, this test will omit that candidate and will update the candidate tetrahedra list.

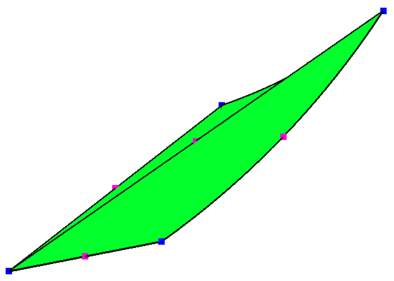


Figure 2: A degenerate tetrahedral element.

Figure 2 shows an example of a degenerate tetrahedral element in which the four vertices are nearly coplanar.

Point inside tetrahedron test: We take the updated candidate tetrahedra from the previous test and perform the point inside tetrahedron check. This is done using orientation and incircle tests discussed in Section 4.2 of [45]. This test examines each candidate tetrahedron to ensure no existing vertex of the mesh is located inside the tetrahedron. If there is one such candidate tetrahedron, it will be discarded. We check each candidate vertex against other candidate points that are found within the specified search radius.

Edge-face intersection test: Again, we take the updated candidates from the previous test and check for possible edge-face intersections. We conduct this test to detect intersections between the edges of a candidate tetrahedron with the faces of the surrounding existing tetrahedra. To this end, we check for intersections between the candidate tetrahedron edges with existing faces that are within distance r of the candidate tetrahedron. To check for edge-face intersection, we use the principles of line-plane intersection. To represent the curved second-order surface triangles, we use the shape functions for a two-dimensional second-order Lagrange element. We first calculate the tangent

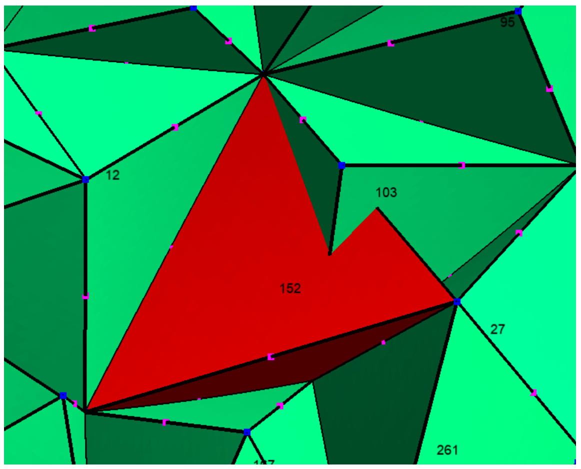


Figure 3: Example of edge-face intersection between element 152 (red) and element 103 (green).

planes to the curves and use them to calculate the surface normal of the face we are checking against. This normal vector represents the face. Next, we select an arbitrary point that lies on the face. For the edge, we take the two endpoints of the selected edge.

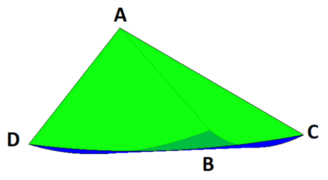
Figure 3 shows an example of an edge-face intersection in which one candidate element edge (element 152 shown in red) intersects with two planes of an existing element (element 103).

Face-edge intersection test: Next, we perform another test on the updated candidate tetrahedra to detect intersections between the faces of a candidate tetrahedron and edges of surrounding existing tetrahedra that are within distance r of that specific candidate. The face-edge intersection test follows the same principle outlined for edge-face intersection test.

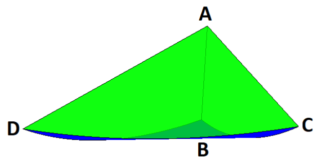
2.6 Element Quality Tests

The candidates that pass the validity tests are considered for element quality checks. Similar to the validity tests, the quality checks are performed in the specified order. If there exists a candidate that does not meet the minimum quality requirement set for a particular test, that candidate is discarded, and the rest of the candidate tetrahedra move forward to the next test.

Dihedral angle: We start the quality checks with a dihedral angle calculation. We calculate the dihedral angles between the faces of a candidate tetrahedron and discard the candidates that have a dihedral angle less than 15°. Since the mesh is curvilinear, we first calculate the tangent planes to the curves and use them to calculate the surface normals. The angle between the normals of the two planes gives us the dihedral angle.



(a)



(b)

Figure 4: Example of various dihedral angles: (a) a dihedral angle of 22° between plane ABC and plane BCD; (b) a dihedral angle of 45° between plane ABC and plane BCD.

Figure 4 shows two different tetrahedra with different minimum dihedral angle values. Fig. 4(a) shows a tetrahedron with a dihedral angle of 22° between plane ABC and plane BCD, and Fig. 4(b) shows a tetrahedron with a dihedral angle of 45° between the same planes.

Equiangular skewness: We use the dihedral angles and measure equiangular skewness of the tetrahedral element. The equiangular skewness [46] is given by:

$$\max\left[\frac{\theta_{max}-\theta_e}{180-\theta_e}, \frac{\theta_e-\theta_{min}}{\theta_e}\right], \tag{2}$$

where

$$\begin{aligned}\theta_{min} &= \text{ smallest angle of the element,}\\ \theta_{max} &= \text{ largest angle of the element, and}\\ \theta_e &= \text{ angle for equiangular element, i.e., } 70.52^\circ\\ &\quad \text{for a regular tetrahedron.}\end{aligned}$$

A low minimum dihedral angle would result in a higher skewness value. The tetrahedra shown in Fig. 4(a) with minimum dihedral angle 22° has an equiangular skewness of 0.68, and the tetrahedron in Fig. 4(b) with minimum dihedral angle 45° has an equiangular skewness of 0.36. We discard candidate tetrahedra that have an equiangular skewness value greater than 0.9 unless there are no other candidates with a lower skewness value available.

Edge-face angle calculation: Next, we calculate the angles between candidate tetrahedral edges and faces of both surrounding existing tetrahedral elements and triangular boundary surfaces. If a candidate exists that creates an edge-face angle of less than 10°, we discard that candidate. This reduces the possibility of creating low-quality elements, such as slivers, needles, and caps on future iterations.

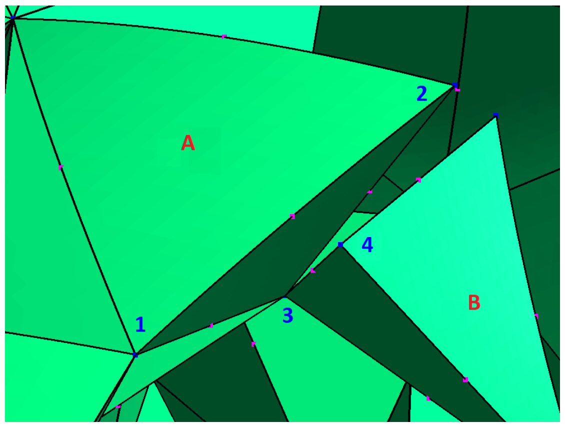


Figure 5: Example of an edge-face angle with the potential to create invalid elements.

In Fig. 5, face 1-2-3 of tetrahedron A is creating an angle of approximately 8° with edge 3-4 of tetrahedron B. This narrow angle could generate low-quality elements on future iterations. We discard candidate the tetrahedra that might create such angles in the mesh.

Scaled Jacobian: As a final quality metric, we calculate the scaled Jacobian of our candidate tetrahedra. The scaled Jacobian is defined as:

$$\frac{\min J(\boldsymbol{\xi})}{\max J(\boldsymbol{\xi})}, \tag{3}$$

where $J(\boldsymbol{\xi}) = \det(\partial\mathbf{x}/\partial\boldsymbol{\xi})$.

This is the Jacobian of the mapping from the reference coordinate $\boldsymbol{\xi}$ to the physical coordinate $\mathbf{x}$. Scaled Jacobian values can range from $-\infty$ to 1, where a value of 1 indicates a straight-sided element, a value less than 1 indicates a curved element and a negative value indicates an inverted element.

We calculate the scaled Jacobian using the shape functions for a second-order Lagrange tetrahedron and the 11-point Gauss-Lobatto rule [44].

2.7 Mesh modification operations

In the event that there are no suitable candidate element present after performing the validity and quality checks, we conduct several mesh modification operations to generate a valid element. These modification operations include:

- **Local remeshing:** We remove the tetrahedron near the selected surface triangle and remesh that region.

- **Face swapping and edge contraction:** If there are low-quality elements in the mesh, e.g., sliver, wedge, needle, or cap elements, we perform face swaps [42, 43] and/or edge contractions [41] to obtain an element that has higher quality. If these operations do not improve the element quality or they reduce the quality of the adjacent elements, we keep the original element in the mesh. Among the various types of face-swapping operations, our method currently employs 2-2, 2-3, and 3-2 swaps.

2.8 Selection of the best possible tetrahedral element

Once we have the candidate tetrahedra that passed all of the validity and quality checks, we use our tetrahedral selection algorithm described in Algorithm 2 to select the best quality tetrahedron and insert it into the mesh.

Since a high-order mesh consists of both straight-sided and curved elements, we use two different selection criteria for these two different types of elements. For curved elements, the selection criterion is based on the scaled Jacobian, and for straight-sided elements, it is based on the equiangular skewness.

Curved elements: Often the candidate with the highest scaled Jacobian value does not generate the best tetrahedral element in terms of the overall quality of the mesh. The candidate vertices can include both existing vertices from the mesh, as well as one newly added vertex a as shown in Fig. 1. If the newly added vertex a gives the highest scaled Jacobian but there are other existing candidate vertices that are very close to a, then selecting the element with the highest scaled Jacobian will create two vertices that are very close to each other, which will generate low-quality elements, such as slivers, caps, and needles on future iterations. Thus, if two such vertices exist in the candidate list, we select the existing vertex over the new vertex.

Straight-sided elements: For straight-sided elements, the scaled Jacobian value is always 1. Thus, we use equiangular skewness to measure the quality of such elements. We use the same analogy described above for curved elements and give preference to selection of existing vertices over the addition of new vertices.

Algorithm 2: Selection of the best possible tetrahedral element

```
Input: Candidate tetrahedra $C_i, i = 1, 2, ..., n$ that pass validity and quality checks
Output: The most suitable candidate tetrahedron $C$
if more than one candidate tetrahedron present then
    if scaled Jacobian $\in$ (0,1) then
        if new vertex $a$ is the fourth vertex of the candidate tetrahedron with highest scaled Jacobian value then
            1. Calculate distance $d$ between $a$ and the fourth vertex of every other candidate tetrahedra present $C_i, i = 1, 2, ..., n-1$
            if $d < h$ then
                select candidate tetrahedron $C$ with lowest $d$
            else
                select the candidate tetrahedron $C$ with the highest scaled Jacobian value
            end
        else
            select the candidate tetrahedron $C$ with the highest scaled Jacobian value
        end
    else
        if scaled Jacobian = 1 then
            if new vertex $a$ is the fourth vertex of the candidate tetrahedron with lowest equiangular skewness then
                1. Calculate distance $d$ between $a$ and the fourth vertex of every other candidate tetrahedra present $C_i, i = 1, 2, ..., n-1$
                if $d < h$ then
                    select candidate tetrahedron $C$ with lowest $d$
                else
                    select the candidate tetrahedron $C$ with the lowest equiangular skewness
                end
            else
                select the candidate tetrahedron $C$ with the lowest equiangular skewness
            end
        end
    end
else
    select the only candidate tetrahedron available
end
```

2.9 Mesh Optimization

After generation of the straight-sided quadratic tetrahedral meshes is complete, we apply mesh optimization. This ensures that the high-order degrees of freedom available in each element are utilized and the quality of the mesh is further improved. After mesh optimization, we obtain quadratic curvilinear tetrahedral meshes. We use the mesh optimization algorithm for regularization of high-order meshes available in Gmsh. We use the disjoint strong strategy with 100 maximum iterations and 25 barrier updates [39].

3. NUMERICAL RESULTS

In this section, we show the results from applying our mesh generation algorithm to generate several second-order tetrahedral meshes from second-order triangular surface meshes. We also report the wall-clock time required to generate the meshes and to optimize them. The straight-sided quadratic tetrahedral mesh generation algorithm was run using Matlab R2019b. The optimization algorithm available in Gmsh is implemented in C++. All the execution times were measured on a machine with 16GB of RAM and an Intel(R) Core(TM) i7-6700HQ CPU. All mesh visualizations were generated using Gmsh.

For our examples, the low-order surface meshes were generated using Gmsh, Netgen [38], and from boundary representations obtained from segmented medical images. To show that our method can be used to generate curvilinear quadratic tetrahedral meshes from various types of boundary representations, such as when a CAD model is either available or absent, we show both types of examples. To this end, we show examples when the geometry is generated by us versus when the geometry or the surface mesh is obtained from other sources. The high-order surface meshes were generated using Gmsh and meshCurve [47]. For example 1 and example 3, the geometries and the second-order surface meshes were directly generated using Gmsh. For example 2, the geometry was obtained from Netgen, and for example 4, the linear surface mesh was obtained from Netgen. For example 5, the geometric model of the left ventricle (LV) myocardium was obtained from cine cardiac magnetic resonance (MR) images using the Automated Cardiac Diagnosis Challenge (ACDC) dataset [48]. To generate the low-order surface mesh, the Lewiner marching cubes [49] algorithm was used and followed by mesh simplification in MeshLab [50]. Once we have the low-order surface meshes for examples 2,4, and 5, we then use Gmsh to introduce the high-order nodes to the low-order surface mesh. Finally meshCurve is used to obtain the curvilinear second-order surface mesh from the linear surface mesh. Figure 6 shows an overview of the high-order surface mesh generation process.

For our first example, we generate a curvilinear, quadratic tetrahedral mesh using the second-order surface mesh of a torus. The 3D mesh generated using our method has 3852 tetrahedral elements. For this example, we used a search radius of $1.5h$. For mesh optimization, we used a target Jacobian range of $0.7 - 2.0$. Figure 7(a) shows the volume mesh of the torus generated using our method. Figures 7(b,c) show two cross sections of the mesh. The 3D mesh does not contain inverted elements. The runtime and element quality information before and after applying mesh optimization are shown in Fig. 7(d). After optimization, the minimum scaled Jacobian value increased from 0.541 to 0.681, and the maximum equiangular skewness value decreased from 0.864 to 0.857. Figures 7(e,f) show the histograms of the quality measures. As seen in Fig. 7(f), only around 0.2 percent of the total elements have a skewness value higher than 0.85.

For our second example, we generated a second-order tetrahedral mesh of a screw. The CAD file of the screw was obtained from Netgen. To generate this mesh, we used a search radius of $1.8h$. The 3D mesh of the screw has 272 elements. There are several skinny triangular elements present in the surface mesh near the screw head region that can be seen in the circled areas in Figs. 8(d,e). The surface mesh also contains several highly-curved, skinny triangular elements on the screw end as shown by the circled area in Fig. 8(f). These lower-quality triangular elements initially resulted in a higher skewness and lower scaled Jacobian value than normal. However, incorporation of the mesh optimization step further improved the quality of the meshes. For mesh optimization, we used a target Jacobian range of $0.1 - 2.0$. Figures 8(a-c) show the 3D mesh of the screw. The runtime and element quality information are shown in Fig. 8(g). Before optimization, the minimum scaled Jacobian value of the mesh was 0.028. After optimization this value increased to 0.278. Similarly, the maximum equiangular skewness value before optimization was 0.931. After optimizing the mesh, the maximum skewness value reduced to 0.927. Figures 8(h,i) show the histograms of the quality measures. It can be seen from the histograms that only around 0.3 percent elements have a skewness value higher than 0.9.

Our third example is a second-order tetrahedral mesh of a hollow cylinder with no ends. The 3D mesh generated using our method has 6050 elements. For this example, we used a search radius of $1.5h$. For mesh optimization, we used a target Jacobian range of $0.8-2.0$. The tetrahedral mesh generated using our method is shown in Fig. 9. Figures 9(b,c) show the hollow cylinder using various cutting planes. The runtime statistics and mesh quality information are shown in Fig. 9(d). Optimizing the mesh increased the minimum scaled Jacobian value from 0.504 to 0.618. The maximum equiangular skewness value decreased from 0.869 to 0.863. The histograms of scaled Jacobian values and equiangular skewness are shown in Figs. 9(e,f). Only around 0.4 percent of the mesh elements have a skewness value greater than 0.85.

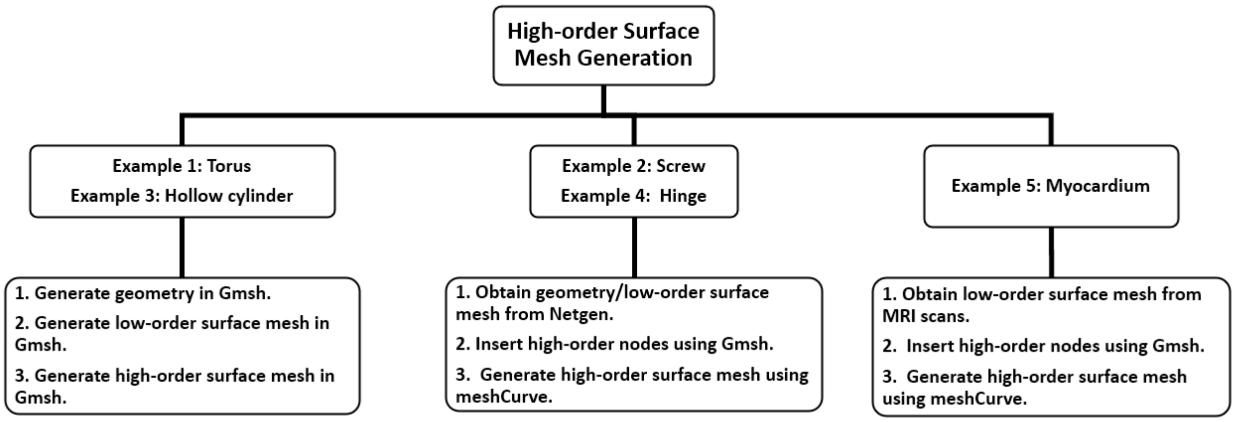

High-order Surface Mesh Generation
Example 1: Torus
Example 3: Hollow cylinder
Example 2: Screw
Example 4: Hinge
Example 5: Myocardium
1. Generate geometry in Gmsh.
2. Generate low-order surface mesh in Gmsh.
3. Generate high-order surface mesh in Gmsh.
1. Obtain geometry/low-order surface mesh from Netgen.
2. Insert high-order nodes using Gmsh.
3. Generate high-order surface mesh using meshCurve.
1. Obtain low-order surface mesh from MRI scans.
2. Insert high-order nodes using Gmsh.
3. Generate high-order surface mesh using meshCurve.

Figure 6: High-order surface mesh generation.

Example	**Software package**	**Quality metric**	**Min quality metric value**	
			Using software package	**Using our method**
Torus	Gmsh	Scaled Jacobian	0.581	0.681
Torus	meshCurve	Jacobian determinant	-0.002	N/A
Cylinder	Gmsh	Scaled Jacobian	0.450	0.618
Hinge	Netgen	Scaled Jacobian	-0.214	0.173
Screw	N/A	Scaled Jacobian	N/A	0.278
Myocardium	N/A	Scaled Jacobian	N/A	0.156

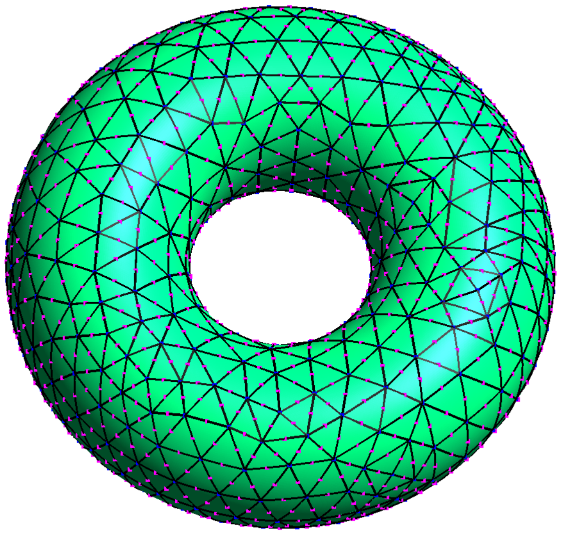
Table 1: Comparison of meshes generated using different software packages.

For our fourth example, we used the geometry of a hinge obtained from Netgen. The second-order tetrahedral mesh of the hinge contains 2996 elements. For this example, we used a search radius of $1.5h$. The surface mesh of the hinge has a combination of very large and small elements. This initially resulted in a minimum scaled Jacobian value of 0.034 and maximum skewness value of 0.903 for this mesh. Similar to the screw, the mesh optimization step improved the quality of the mesh, increasing the scaled Jacobian value to 0.173 and decreasing the skewness value to 0.887. For mesh optimization, we used a target Jacobian range of $0.1 - 2.0$. Figure 10 shows the tetrahedral mesh of the hinge. Figures 10(b-d) show the hinge using various cutting planes. Figure 10(e) shows the runtime statistics and mesh quality information. The histograms of the scaled Jacobian values and equiangular skewness are shown in Figs. 10(f,g). In this mesh, only around 0.6 percent of the elements have a skewness value greater than 0.85.

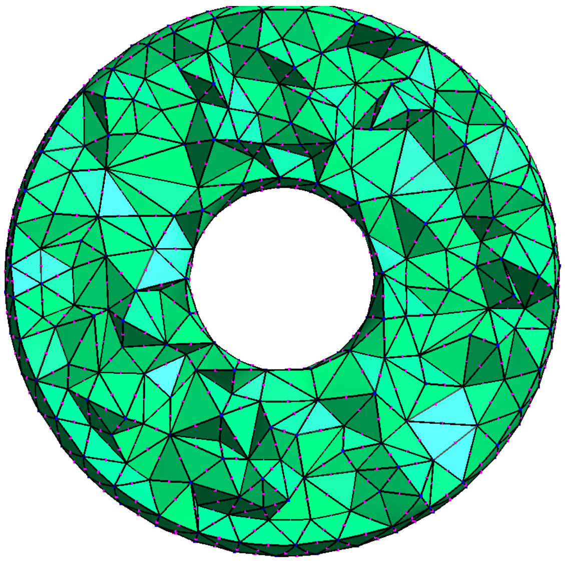
Finally, we generated a second-order tetrahedral mesh of the left ventricle myocardium of a normal human heart. For this example, we used a search radius of $1.5h$. For mesh optimization, we used a target Jacobian range of $0.2 - 2.0$. Figures 11(a,b) show the top and side view of the myocardium. Figure 11(c) shows the myocardium using a cutting plane. The runtime statistics and mesh quality information are shown in Fig. 11(d). Mesh optimization increased the minimum scaled Jacobian value from 0.123 to 0.156 and decreased the maximum skewness value from 0.895 to 0.883. The histograms of scaled Jacobian values and equiangular skewness are shown in Figs. 11(e,f). For this mesh, around 0.8 percent of the elements have a skewness value greater than 0.85.

For coarser meshes, such as the screw example shown in Fig. 8, to obtain enough candidate vertices during the vertex search, the search radius needs to be larger compared to that of a finer mesh, such as the hollow cylinder shown in Fig. 9.

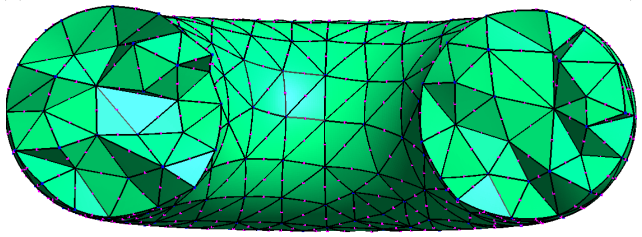
We compare meshes generated using our method with those generated by existing open-source high-order meshing software such as Gmsh, meshCurve, and Netgen. We note that, these software packages use a mesh generation approach that is different from ours and require CAD files to generate their meshes. For the torus example, we generated the second-order surface mesh using Gmsh and used it as an input for our method. We used Gmsh to generate a second-order tetrahedral mesh from the same surface mesh. The resulting mesh had 7 elements with a negative scaled Jacobian value.



(a)



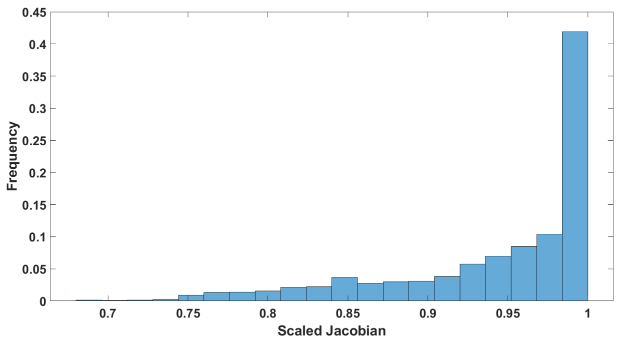
(b)



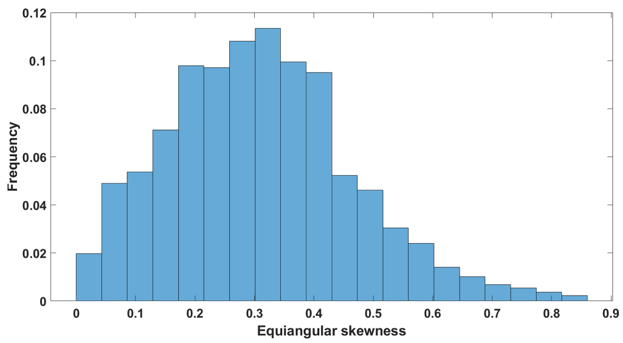
(c)

No. of Elements	Runtime (s)		Scaled Jacobian				Equiangular skewness			
			Initial		Final		Initial		Final	
	Mesh generation	Mesh optimization	min		min	max	min	max	min	max
3852	2192	1.542	0.541	1.000	0.681	1.000	0.006	0.864	0.006	0.857

(d)

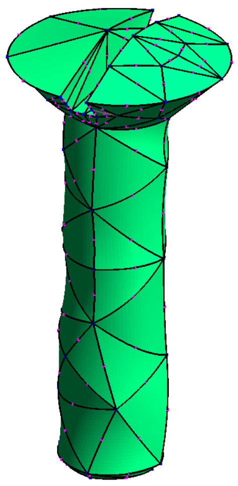


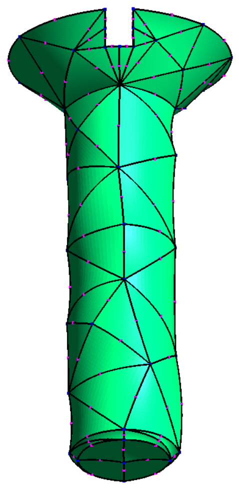
(e)

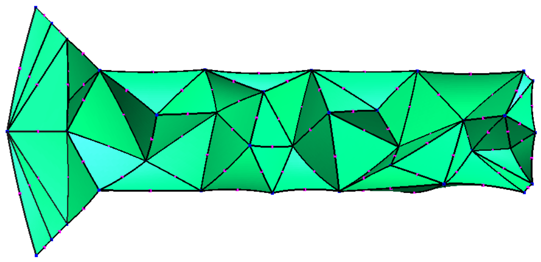


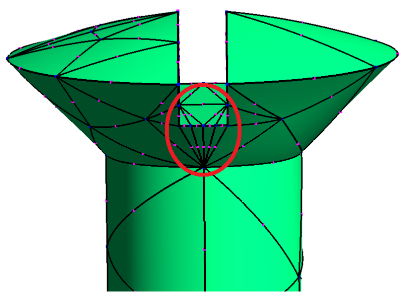
(f)

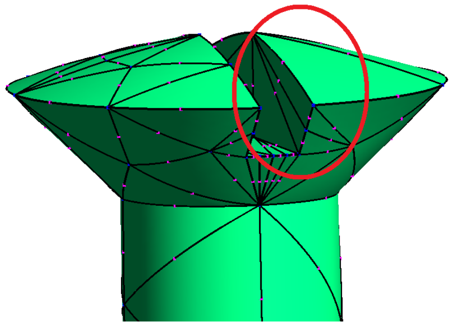
Figure 7: Second-order curvilinear tetrahedral mesh of a torus after optimization: (a) torus; (b) cross section 1; (c) cross section 2; (d) algorithm runtime statistics and mesh quality metrics, and (e,f) histogram plots of scaled Jacobian and equiangular skewness.

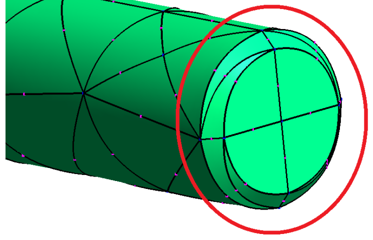

(a)


(b)


(c)

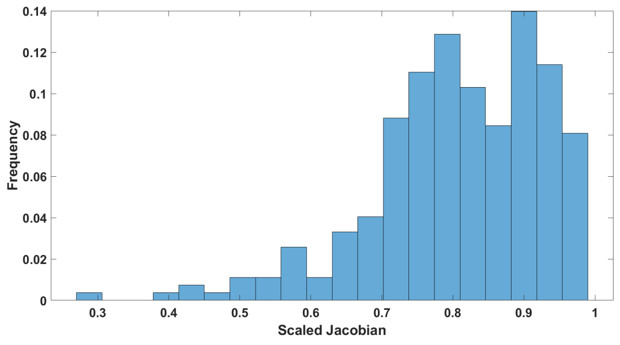

(d)


(e)

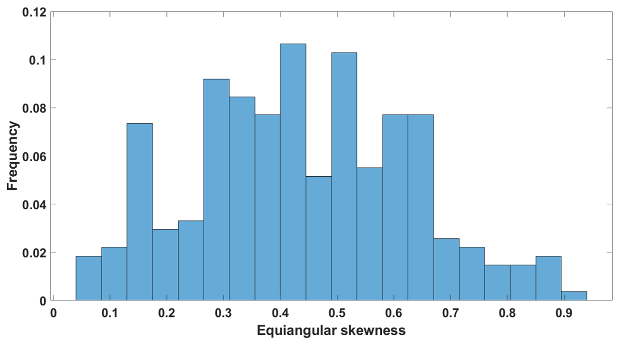

(f)

No. of Elements	Runtime (s)		Scaled Jacobian				Equiangular skewness			
			Initial		Final		Initial		Final	
	Mesh generation	Mesh optimization	min		min	max	min	max	min	max
272	192	0.469	0.028	1.000	0.278	1.000	0.063	0.931	0.056	0.927

(g)

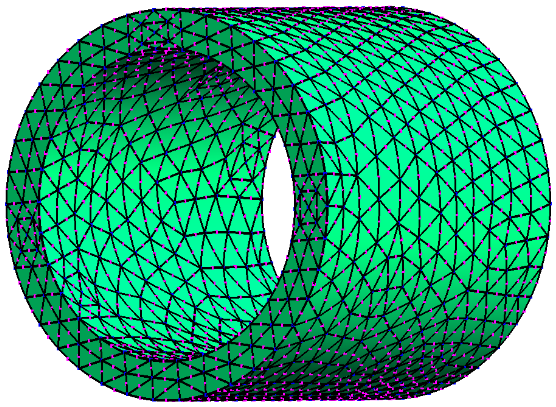


(h)

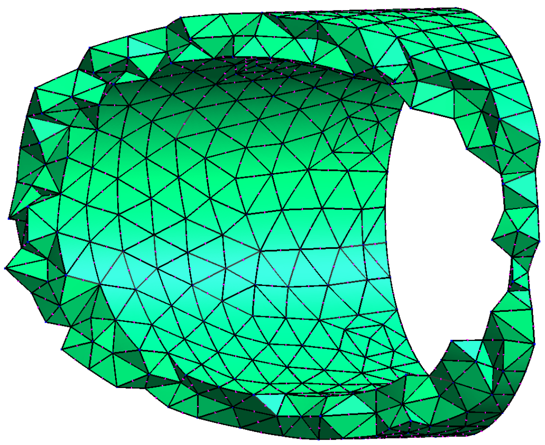


(i)

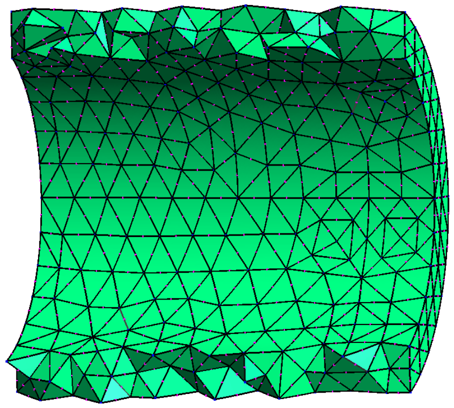
Figure 8: Second-order curvilinear tetrahedral mesh of a screw after optimization: (a) and (b) 3D mesh of the screw (views 1 and 2); (c) cross section of the screw; (d) and (e) zoomed-in views of the screw head; (f) zoomed-in view of the screw end; (g) algorithm runtime statistics and mesh quality metrics, and (h) and (i) histogram plots of scaled Jacobian and equiangular skewness.



(a)



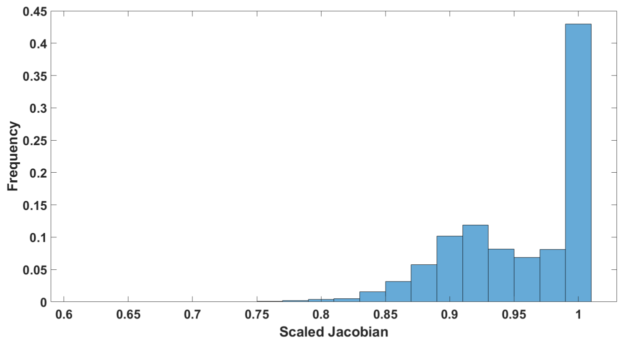
(b)



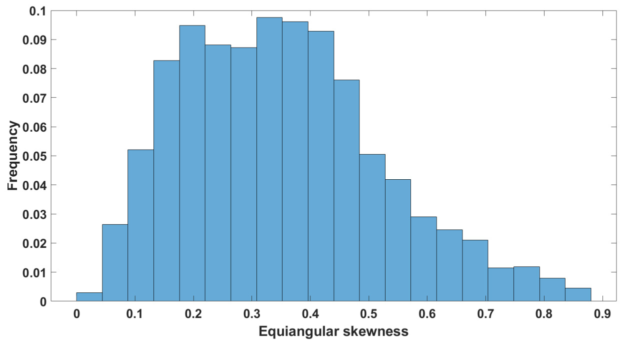
(c)

No. of Elements	Runtime (s)		Scaled Jacobian				Equiangular skewness			
			Initial		Final		Initial		Final	
	Mesh generation	Mesh optimization	min		min	max	min	max	min	max
6050	6570	0.187	0.504	1.000	0.618	1.000	0.031	0.869	0.012	0.863

(d)

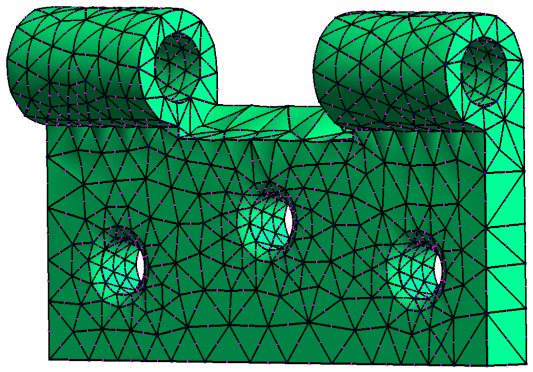


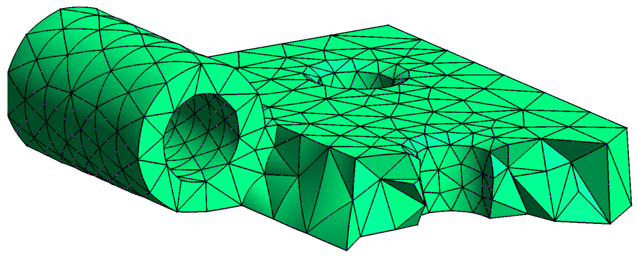
(e)

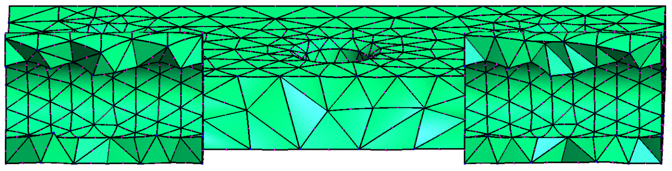


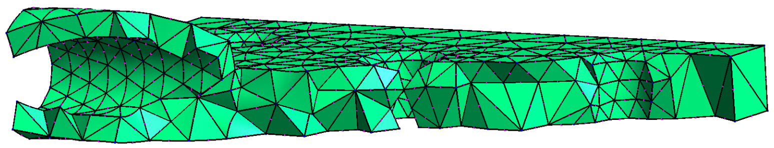
(f)

Figure 9: Second-order curvilinear tetrahedral mesh of a hollow cylinder with no ends after optimization: (a) 3D mesh of the cylinder; (b) cross section 1; (c) cross section 2; (d) algorithm runtime statistics and mesh quality metrics, and (e,f) histogram plots of scaled Jacobian and equiangular skewness.


(a)

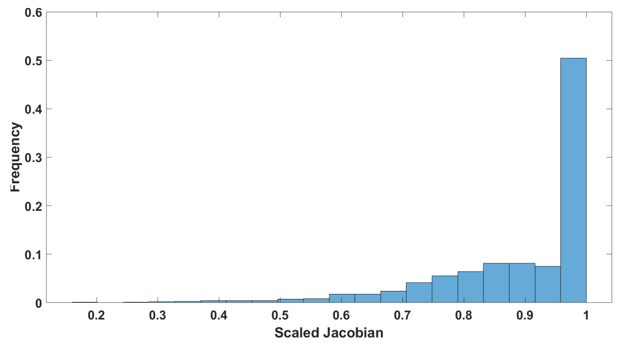

(b)


(c)

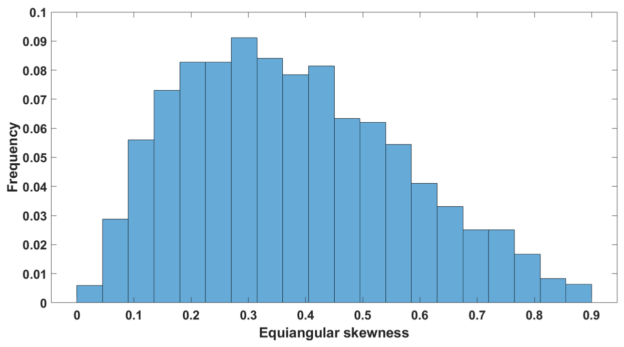

(d)

No. of Elements	Runtime (s)		Scaled Jacobian				Equiangular skewness			
			Initial		Final		Initial		Final	
	Mesh generation	Mesh optimization	min		min	max	min	max	min	max
2996	3166	0.156	0.034	1.000	0.173	1.000	0.032	0.903	0.008	0.887

(e)

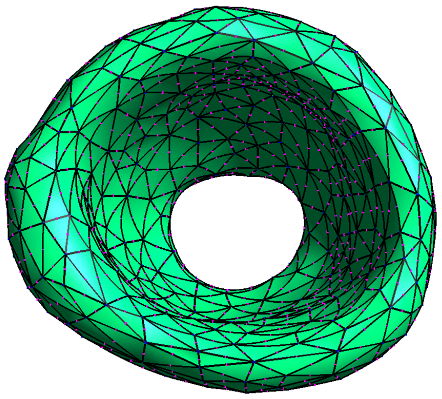


(f)

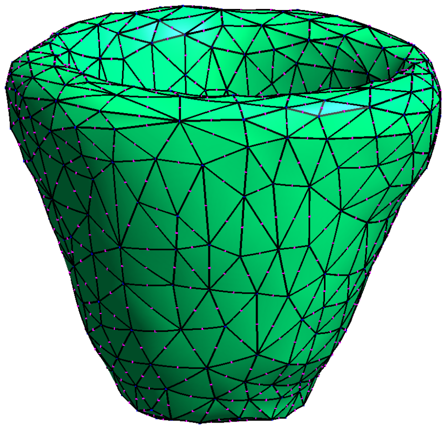


(g)

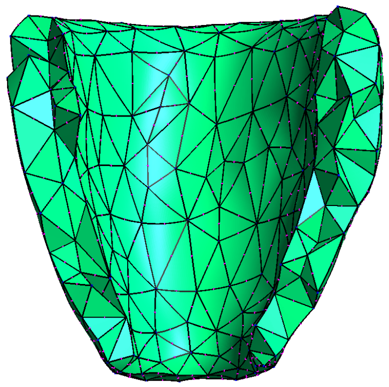
Figure 10: Second-order curvilinear tetrahedral mesh of a hinge after optimization: (a) 3D mesh of the hinge; (b) cross section 1; (c) cross section 2; (d) cross section 3; (e) algorithm runtime statistics and mesh quality metrics, and (f,g) histogram plots of scaled Jacobian and equiangular skewness.



(a)



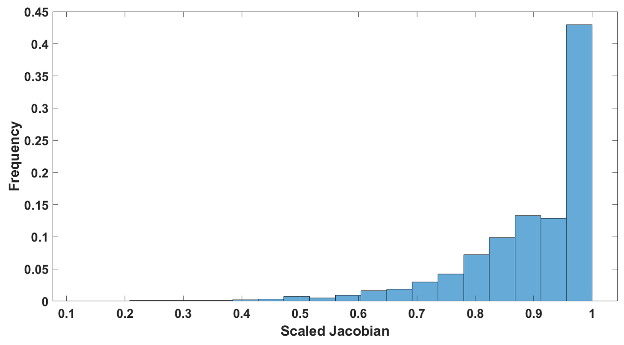
(b)



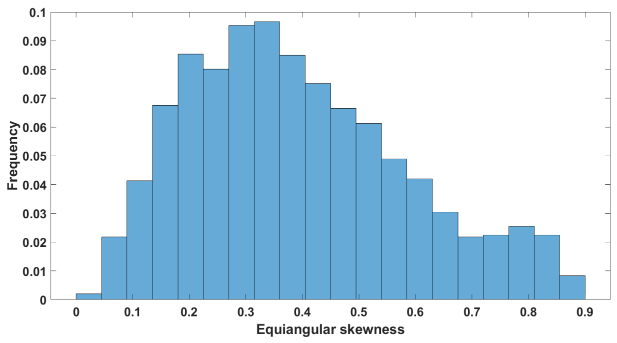
(c)

No. of Elements	Runtime (s)		Scaled Jacobian				Equiangular skewness			
			Initial		Final		Initial		Final	
	Mesh generation	Mesh optimization	min		min	max	min	max	min	max
3022	3283	0.203	0.123	1.000	0.156	1.000	0.041	0.895	0.013	0.883

(d)



(e)



(f)

Figure 11: Second-order curvilinear tetrahedral mesh of the left ventricle (LV) myocardium of a normal human heart after optimization: (a) top view; (b) side view; (c) cross section 1; (d) algorithm runtime statistics and mesh quality metrics, and (e,f) histogram plots of scaled Jacobian and equiangular skewness.

We used mesh regularization with a target Jacobian range of $0.7 - 2.0$ in Gmsh to optimize the mesh. The optimized mesh had a minimum scaled Jacobian value of 0.581. Next, we tried to generate a second-order tetrahedral mesh of the same torus using meshCurve. After the surface curving, the mesh had a minimum Jacobian value of -0.0018365 resulting in a tangled mesh. Since meshCurve does not have a mesh optimization option, we could not improve the equality of this mesh further.

For the hollow cylinder with no end, we obtained the second-order surface mesh from Gmsh. We attempted to generate a second-order tetrahedral mesh from the same surface mesh using Gmsh. The resulting mesh was valid but had a minimum scaled Jacobian value of 0.000213. After applying mesh regularization in Gmsh, the final tetrahedral mesh had a minimum scaled Jacobian value of 0.450. For mesh optimization, we used a target Jacobian range of $0.8 - 2.0$.

For the hinge example, the surface mesh was obtained from Netgen. We used Netgen to generate a second-order mesh of the hinge. The resulting mesh had a minimum scaled Jacobian value of -0.214. The mesh optimization algorithm available in Netgen failed to untangle the mesh.

For the screw example, although the initial geometry was obtained from Netgen, we used Gmsh and meshCurve to obtain the final curved surface mesh and hence a similar volume mesh was not possible to generate using Netgen.

For the myocardium example, since it was not generated from a CAD file, we could not compare it with the results of any existing high-order mesh generation software. Table 1 shows a comparison of the quality of the meshes generated using the software packages mentioned previously and using our method. While Gmsh successfully generated meshes of the torus and the cylinder, the scaled Jacobian values of these meshes were lower than those generated using our method. Meshes of the torus and the hinge generated using meshCurve and Netgen had inverted elements and resulted in tangled meshes which the respective software packages failed to untangle. Overall, the meshes generated using our method are of better quality.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a novel direct method for generation of high-order tetrahedral meshes based on use of an advancing front technique. Our method assumes as input a high-order triangular surface mesh for the boundary of the geometry. In contrast with *a posteriori* mesh generation methods, our method does not depend on the availability of a CAD file.

The quality of our high-order tetrahedral mesh is somewhat dependent on the quality of the input high-order triangular surface mesh. If the input surface mesh has skinny elements, this will affect the quality of the 3D mesh. Also, if the surface mesh is composed of heterogeneous-sized elements, this will also affect the quality of the 3D mesh generated using our method. While the meshes generated using our method are initially mostly of high element quality, examples such as the screw and the hinge were improved using the mesh optimization step. Our method has been successfully used to generate isotropic meshes. In the future, we plan to extend our method to generate anisotropic meshes, as well. Also, currently we have implemented the method to generate quadratic tetrahedral meshes. In the future, we plan to extend our implementation to generate meshes of higher orders.

Finally, the method is currently implemented in Matlab and serves as an initial prototype. Our future work will include implementing the method in C++ to reduce the runtime so that it can handle larger, real-world meshes, such as those required for biomedical simulations.

5. ACKNOWLEDGMENTS

The work of the first author was funded by NSF OAC grant 1808553. The work of the second author was funded in part by NSF grants OAC 1808553 and CCF 1717894. We would also like to thank Dr. Cristian A. Linte from the Biomedical Modeling, Visualization and Image-guided Navigation Lab at Rochester Institute of Technology for providing us with the myocardium surface meshes. Finally, we would like to thank the referees for their insightful comments which significantly improved the quality of the paper.

References

[1] Yosibash Z., Trabelsi N., Milgrom C. "Reliable simulations of the human proximal femur by high-order finite element analysis validated by experimental observations." *Journal of Biomechanics*, vol. 40, no. 16, 3688–3699, 2007

[2] Castonguay P., Williams D., Vincent P., Lopez M., Jameson A. "On the development of a high-order, multi-GPU enabled, compressible viscous flow solver for mixed unstructured grids." *20th AIAA Computational Fluid Dynamics Conference*, p. 3229. 2011

[3] Mavriplis D., Nastase C., Wang L., Shahbazi K., Burgess N. "Progess in high-order discontinuous Galerkin methods for aerospace applications." *47th AIAA Aerospace Sciences Meeting including*

The New Horizons Forum and Aerospace Exposition, p. 601. 2009

[4] Persson P.O., Peraire J. "Curved mesh generation and mesh refinement using Lagrangian solid mechanics." *Proceedings of the 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, p. 949. 2009

[5] Stees M., Dotzel M., Shontz S.M. "Untangling high-order meshes based on signed angles." *Proceedings of the 28th International Meshing Roundtable*, pp. 267–282. Zenodo, 2020

[6] Carrigan T., Landon M., Pita C. "Meshing considerations for automotive shape design optimization." Tech. rep., SAE Technical Paper, 2016

[7] Dawes W.N. "Automated meshing for aerothermal analysis of complex automotive geometries." Tech. rep., SAE Technical Paper, 2011

[8] Johnston C., Barnes S. "Development of high-order meshing for industrial aerospace configurations." *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, pp. 65–78. Springer, 2015

[9] Wang Z.J., Fidkowski K., Abgrall R., Bassi F., Caraeni D., Cary A., Deconinck H., Hartmann R., Hillewaert K., Huynh H.T., et al. "High-order CFD methods: current status and perspective." *International Journal for Numerical Methods in Fluids*, vol. 72, no. 8, 811–845, 2013

[10] Lamata P., Sinclair M., Kerfoot E., Lee A., Crozier A., Blazevic B., Land S., Lewandowski A.J., Barber D., Niederer S., et al. "An automatic service for the personalization of ventricular cardiac meshes." *Journal of The Royal Society Interface*, vol. 11, no. 91, 20131023, 2014

[11] Gonzales M.J., Sturgeon G., Krishnamurthy A., Hake J., Jonas R., Stark P., Rappel W.J., Narayan S.M., Zhang Y., Segars W.P., et al. "A three-dimensional finite element model of human atrial anatomy: new methods for cubic Hermite meshes with extraordinary vertices." *Medical Image Analysis*, vol. 17, no. 5, 525–537, 2013

[12] Karman S.L., Erwin J.T., Glasby R.S., Stefanski D. "High-order mesh curving using WCN mesh optimization." *Proceedings of the 46th AIAA Fluid Dynamics Conference*, p. 3178. 2016

[13] Dey S., O'bara R.M., Shephard M.S. "Curvilinear mesh generation in 3D." *Proceedings of the 8th International Meshing Roundtable, 1999*, pp. 407–417. 1999

[14] Fortunato M., Persson P.O. "High-order unstructured curved mesh generation using the Winslow equations." *Journal of Computational Physics*, vol. 307, 1–14, 2016

[15] Gargallo-Peiró A., Roca X., Peraire J., Sarrate J. "Distortion and quality measures for validating and generating high-order tetrahedral meshes." *Engineering with Computers*, vol. 31, no. 3, 423–437, 2015

[16] Gargallo-Peiró A., Roca X., Peraire J., Sarrate J. "Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes." *International Journal for Numerical Methods in Engineering*, vol. 103, no. 5, 342–363, 2015

[17] Geuzaine C., Remacle J.F. "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities." *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, 1309–1331, 2009

[18] Roca X., Gargallo-Peiró A., Sarrate J. "Defining quality measures for high-order planar triangles and curved mesh generation." *Proceedings of the 20th International Meshing Roundtable*, pp. 365–383. Springer, 2011

[19] Ruiz-Gironés E., Sarrate J., Roca X. "Generation of curved high-order meshes with optimal quality and geometric accuracy." *Proceedings of the 25th International Meshing Roundtable*, vol. 163, pp. 315–327. Procedia Engineering, 2016

[20] Xie Z.Q., Sevilla R., Hassan O., Morgan K. "The generation of arbitrary order curved meshes for 3D finite element analysis." *Computational Mechanics*, vol. 51, no. 3, 361–374, 2013

[21] Stees M., Shontz S.M. "A high-order log barrier-based mesh generation and warping method." *Proceedings of the 26th International Meshing Roundtable*, vol. 203, pp. 180–192. Procedia Engineering, 2017

[22] Feng L., Alliez P., Busé L., Delingette H., Desbrun M. "Curved optimal Delaunay triangulation." *ACM Transactions on Graphics*, vol. 37, no. 4, 16, 2018

[23] Moxey D., Ekelschot D., Keskin Ü., Sherwin S.J., Peiró J. "High-order curvilinear meshing using a thermo-elastic analogy." *Computer-Aided Design*, vol. 72, 130–139, 2016

[24] Mohammadi F., Dangi S., Shontz S.M., Linte C.A. "A direct high-order curvilinear triangular mesh generation method using an advancing front

technique." *International Conference on Computational Science*, vol. 12138 of *Lecture Notes in Computer Science*, pp. 72–85. Springer, 2020

[25] Hughes T.J., Cottrell J.A., Bazilevs Y. "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement." *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 39-41, 4135–4195, 2005

[26] Cottrell J.A., Hughes T.J., Bazilevs Y. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, 2009

[27] Sevilla R., Fernández-Méndez S., Huerta A. "NURBS-enhanced finite element method (NEFEM)." *International Journal for Numerical Methods in Engineering*, vol. 76, no. 1, 56–83, 2008

[28] Mohmadsalehi M., Nagarajan A., Soghrati S. "Higher-order mesh generation using CISAMR: A case study on bias in presentation and interpretation of results." *Computer Methods in Applied Mechanics and Engineering*, vol. 372, 113360, 2020

[29] Soghrati S., Nagarajan A., Liang B. "Conforming to interface structured adaptive mesh refinement: new technique for the automated modeling of materials with complex microstructures." *Finite Elements in Analysis and Design*, vol. 125, 24–40, 2017

[30] Mothes Hansvold K.A. *Mesh Generation Techniques for Isogeometric Analysis*. Master's thesis, NTNU, 2015

[31] Lai Y., Zhang Y.J., Liu L., Wei X., Fang E., Lua J. "Integrating CAD with Abaqus: A practical isogeometric analysis software platform for industrial applications." *Computers & Mathematics with Applications*, vol. 74, no. 7, 1648–1660, 2017

[32] Owen S.J. "A survey of unstructured mesh generation technology." *Proceedings of the 7th International Meshing Roundtable*, pp. 239–267. 1998

[33] Lo S.H. "Volume discretization into tetrahedra—II. 3D triangulation by advancing front approach." *Computers & Structures*, vol. 39, no. 5, 501–511, 1991

[34] Löhner R., Parikh P. "Generation of three-dimensional unstructured grids by the advancing-front method." *International Journal for Numerical Methods in Fluids*, vol. 8, no. 10, 1135–1149, 1988

[35] Mavriplis D.J. "An advancing front Delaunay triangulation algorithm designed for robustness." *Journal of Computational Physics*, vol. 117, no. 1, 90–101, 1995

[36] Merriam M. "An efficient advancing front algorithm for Delaunay triangulation." *Proceedings of the 29th Aerospace Sciences Meeting*, p. 792. 1991

[37] Ito Y., Shih A.M., Soni B.K. "Reliable isotropic tetrahedral mesh generation based on an advancing front method." *Proceedings of the 13th International Meshing Roundtable*, pp. 95–106. 2004

[38] Schöberl J. "NETGEN An advancing front 2D/3D-mesh generator based on abstract rules." *Computing and Visualization in Science*, vol. 1, no. 1, 41–52, 1997

[39] Geuzaine C., Johnen A., Lambrechts J., Remacle J.F., Toulorge T. "The generation of valid curvilinear meshes." *IDIHOM: Industrialization of High-Order Methods - A Top-Down Approach*, pp. 15–39. Springer, 2015

[40] Dunavant D. "High Degree efficient symmetrical Gaussian quadrature rules for the triangle." *International Journal for Numerical Methods in Engineering*, vol. 21, no. 6, 1129–1148, 1985

[41] Dassi F., Formaggia L., Zonca S. "Degenerate tetrahedra removal." *Applied Numerical Mathematics*, vol. 110, 1–13, 2016

[42] Freitag L.A., Ollivier-Gooch C. "Tetrahedral mesh improvement using swapping and smoothing." *International Journal for Numerical Methods in Engineering*, vol. 40, no. 21, 3979–4002, 1997

[43] Klingner B.M., Shewchuk J.R. "Aggressive tetrahedral mesh improvement." *Proceedings of the 16th International Meshing Roundtable*, pp. 3–23. Springer, 2008

[44] Maas S., Rawlins D., Weiss J., Ateshian G. "FEBio theory manual." *Musculoskeletal Research Laboratories, University of Utah, Salt Lake City, UT*, 2011. URL `https://help.febio.org/Manuals/FEBioTheory/index.html`

[45] Shewchuk J.R. "Adaptive precision floating-point arithmetic and fast robust geometric predicates." *Discrete & Computational Geometry*, vol. 18, no. 3, 305–363, 1997

[46] "TGrid 5.0 user's guide.", 2008. URL `https://romeo.univ-reims.fr/documents/fluent/tgrid/ug/tgrid50-ug.pdf`

[47] Ims J., Duan Z., Wang Z.J. “meshCurve: An automated low-order to high-order mesh generator.” *Proceedings of The 22nd AIAA Computational Fluid Dynamics Conference*, p. 2293. 2015

[48] Upendra R.R., Wentz B.J., Simon R., Shontz S.M., Linte C.A. “CNN-based cardiac motion extraction to generate deformable geometric left ventricle myocardial models from Cine MRI.” *Proceedings of the 11th Biennial International Conference on Functional Imaging and Modeling of the Heart*, to appear, 2021

[49] Lewiner T., Lopes H., Vieira A.W., Tavares G. “Efficient implementation of marching cubes’ cases with topological guarantees.” *Journal of Graphics Tools*, vol. 8, no. 2, 1–15, 2003

[50] Cignoni P., Callieri M., Corsini M., Dellepiane M., Ganovelli F., Ranzuglia G. “MeshLab: an Open-Source Mesh Processing Tool.” V. Scarano, R.D. Chiara, U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008