A high-order log barrier-based mesh generation and warping method

Mike Stees\textsuperscript{a}, Suzanne M. Shontz\textsuperscript{b}

\textsuperscript{a}Department of Electrical Engineering and Computer Science, Information and Telecommunication Technology Center, mstees@ku.edu, The University of Kansas, Lawrence, KS, 66045, USA

\textsuperscript{b}Department of Electrical Engineering and Computer Science, Bioengineering Graduate Program, Information and Telecommunication Technology Center, shontz@ku.edu, The University of Kansas, Lawrence, KS, 66045, USA

Abstract

The ability to generate high-order meshes that conform to the boundary of curved geometries is a hurdle in the adoption of high-order computational methods for the numerical solution of partial differential equations. In this paper, we propose a method for generating and warping second-order Lagrange triangular and tetrahedral meshes based on a log barrier method. In the case of generation, the approach consists of modifying an initial linear mesh by first, adding nodes at the midpoint of each edge; second, displacing the newly added boundary midpoints to the curved boundary, and third, solving for the final positions of the interior nodes based on the boundary deformation. By allowing all of the boundary nodes to move, the approach can also be used to warp second-order triangular and tetrahedral meshes. We present several numerical examples in both two and three dimensions which demonstrate the capabilities of our method in generating and warping second-order curvilinear meshes.

Keywords: high-order mesh generation, curvilinear high-order mesh, high-order mesh warping, numerical optimization

1. Introduction

High-order computational methods for solving partial differential equations have piqued the interest of the scientific computing community because of their potential to deliver more accurate solutions at a lower cost than their low-order counterparts. To take full advantage of these high-order methods in the presence of curved boundaries, the methods need to be paired with a high-order mesh that accurately captures the curvature of the boundary. Unfortunately, there is a lack of robust high-order mesh generation methods that are suitable for generating curvilinear meshes for complex geometries [1].

Existing approaches for high-order mesh generation can be classified into two categories [2]: \textit{direct} methods, which generate a curved high-order mesh directly from a representation of the geometry, such as a CAD file, and the more common \textit{a posteriori} methods, which generate a high-order mesh by transforming an initial linear mesh [3–11]. A typical \textit{a posteriori} approach consists of the following three steps: (1) additional nodes are added to the linear mesh, (2) the newly added boundary nodes are projected onto the curved boundary, and (3) the interior nodes are moved to their final positions. Within the space of \textit{a posteriori} approaches, there are three subcategories of methods. The first category involves deforming the mesh based on the solution to a PDE. In the first category, several approaches have been proposed. Persson and Peraire [3] considered a nonlinear elasticity approach, while Xie et al. [6] employed...
a linear elasticity approach. Fortunato and Persson [4] posed the problem in terms of the Winslow equations, and Moxey et al. [12] investigated a thermo-elastic model. The second category is based on optimization of an objective function with the goal of minimizing element distortion and accurately representing the geometry [8–11,13]. Finally, methods in the third category curve the mesh boundary, and then identify invalid elements and eliminate them through a combination of local refinement, edge and face swaps, and node relocation [2,14,15].

The purpose of this paper is to describe a new method for generating and warping second-order Lagrange triangular and tetrahedral meshes based on the Log Barrier-Based Mesh Warping (LBWARP) method. We demonstrate our method on several numerical examples in both two and three dimensions. The remainder of this paper is organized as follows. In Section 2, we review the original LBWARP method. In Section 3, we present our new method for second-order mesh generation and warping. The performance of our method is illustrated in Section 4. Finally, in Section 5, we summarize our findings and give some possibilities for future work.

2. The Log Barrier-Based Mesh Warping Method

The LBWARP method was originally proposed by Shontz and Vavasis in [16]. At a high level, the LBWARP algorithm follows these steps: (1) for each interior node, a strictly convex optimization problem is solved to calculate a set of local weights that relates the interior node to its neighbors; (2) a user-supplied deformation is applied to the boundary nodes; (3) after deforming the boundary, the new positions of the interior nodes are calculated by solving a linear system of equations using the weights from (1) and the new boundary positions from (2).

Before proceeding further, let us introduce the following notation for the 2D formulation of the problem. The 3D formulation is similar. Let \((x_i, y_i)\) represent the \(x\)- and \(y\)-coordinates of the \(i^{th}\) interior node. Furthermore, let the \(x\)- and \(y\)-coordinates of the adjacent vertices of node \(i\) be given by \(\{(x_j, y_j) : j \in N_i\}\), where \(N_i\) is the set of neighbors of node \(i\). With this notation in mind, we proceed to the formulation of the optimization problem. In order to find the set of weights \(\lambda_{ij}\), where \(\lambda_{ij}\) denotes the weight of node \(j\) on interior node \(i\), the following optimization problem is formulated using the log barrier function for each interior node \(i\):

\[
\max_{\lambda_{ij}, \forall j \in N_i} \sum_{j \in N_i} \log(\lambda_{ij}) \tag{1}
\]
subject to

\[
\lambda_{ij} > 0 \tag{2}
\]

\[
\sum_{j \in N_i} \lambda_{ij} = 1 \tag{3}
\]

\[
x_i = \sum_{j \in N_i} \lambda_{ij} x_j \tag{4}
\]

\[
y_i = \sum_{j \in N_i} \lambda_{ij} y_j, \tag{5}
\]

where

\[
N_i = \begin{cases} 
\{\text{all nodes with which } i \text{ shares an edge}\}, & \text{if generating a low-order mesh} \\
\{\text{all nodes of the triangles to which } i \text{ belongs}\}, & \text{if generating a high-order mesh.}
\end{cases}
\]

The log barrier function serves two important purposes in this formulation. In particular, it prevents negative weights and more evenly distributes the weights. We are not the first group to use a log barrier in a high-order mesh generation context. Toulorge et al. [11] use a log barrier to prevent element Jacobians from becoming too small as part of an objective function for untangling invalid curved elements. As posed in (1)-(5), this is a strictly convex optimization problem for which there is a unique minimum. By starting with an initial feasible point, this optimization problem is solved using the projected Newton method [17]. To calculate the initial feasible point, three of the interior
node’s adjacent vertices are chosen to write \((x_i, y_i)\) as a convex combination of the positions of the three nodes. The other adjacent neighbors are initially assigned a weight of \(e\), a small positive constant. This initial feasible point serves as the starting point for the projected Newton method which is used to calculate the weights. After calculating the weights, a boundary deformation is applied. The final step is to solve for new locations of the interior nodes by solving the following linear equations:

\[
\sum_{j \in N_i} A_{ij} x_j = x_i \tag{6}
\]

\[
\sum_{j \in N_i} A_{ij} y_j = y_i \tag{7}
\]

While looking at the linear equations for each interior node individually can be helpful, they can also be viewed in global form. Following the notation of [16], let \(b\) and \(m\) represent the numbers of boundary and interior nodes respectively, and define \(x_B\) and \(y_B\) to be vectors of length \(b\) that contain the initial x- and y-coordinates of the boundary nodes. The initial x- and y-coordinates of the interior nodes can likewise be placed in vectors \(x_I\) and \(y_I\), respectively. With this notation in mind, \([x_B, y_B]\) and \([x_I, y_I]\) contain the original boundary and interior node positions, respectively. Next, denote by \(L\) the weighted Laplacian matrix for a weighted graph \(G(V;E;\lambda)\) defined as follows:

\[
L(i, j) = \begin{cases} 
-\lambda_{ij} & \text{if } i \neq j
\sum_{k \in V} \lambda_{ik} & \text{if } i = j.
\end{cases}
\]

where \(\lambda_{ij} = 0\) iff \((i, j) \notin E\). The interior nodes are assumed to be numbered first, while the boundary nodes are assumed to be numbered last. In addition, denote by \(A = [A_I, A_B]\) the matrix that is derived from \(L\) by deleting the last \(b\) rows. The weights corresponding to the interior nodes are contained in \(A_I\), while the weights corresponding to the boundary nodes are contained in \(A_B\). Given these definitions, we can express (6)-(7) as

\[
A_I [x_I, y_I] = -A_B [x_B, y_B]. \tag{8}
\]

3. High-Order LBWARP

In this section, we present our method for high-order mesh generation and warping.

3.1. High-Order Mesh Generation

After adding midpoints to the linear mesh, our algorithm has steps similar to LBWARP. Unlike for the low-order case, the set of neighbors that should be included for an interior point is not necessarily clear. More specifically, not only do we have the traditional low-order nodes, but we also have the added midpoints of each edge. With this in mind, we define the set of neighbors for an interior node \(i\) as \(N_i = \{\text{all nodes of the triangles to which } i \text{ belongs}\}\). In Fig. 1, we illustrate an example of this definition.

Using this new definition of neighbors, we cannot guarantee that our problem is strictly convex. In fact, it may be nonconvex. See Fig. 2 for an example of this. With this in mind, we can no longer solve the optimization problem using the projected Newton method. Instead, we employ a Sequential Quadratic Programming (SQP) method with a backtracking line search based on the following sufficient decrease condition:

\[
\phi(x_k + \alpha p_k) \leq \phi(x_k) + \eta \alpha (\nabla f_k^T p_k - \|c_k\|_1),
\]

where \(p_k\), \(f_k\), and \(c_k\) are \(p(x_k), f(x_k),\) and \(c(x_k)\), respectively. The constant \(\eta\) was chosen to be 0.9, and \(\phi(x) = f(x) + \|c(x)\|_1\). Within the line search, we start with \(\alpha = 1.0\) and scale the value of \(\alpha\) by 0.9 for each iteration of the line search until the sufficient decrease condition is satisfied. Due to the monotonic nature of the solution generated by the backtracking line search and the log barrier term in (1), we can ignore the inequality constraints given by (2)
when solving the problem given by (1)-(5).

To frame our discussion of a typical SQP method, assume that we have the following model problem [18]:

\[
\begin{align*}
\min f(x) \\
\text{subject to } c(x) &= 0.
\end{align*}
\] (9) (10)

The Lagrangian function for the model problem is:

\[ L(x, \lambda) = f(x) - \lambda^T c(x), \]

where \( \lambda \) are the Lagrange multipliers. Next, define \( A(x) \) as the Jacobian matrix of the constraints, that is,

\[ A(x)^T = [\nabla c_1(x), \nabla c_2(x), \ldots, \nabla c_m(x)], \]

where \( c_i(x) \) is the \( i^{th} \) component of \( c(x) \). The Karush-Kuhn-Tucker (KKT) conditions are first-order necessary conditions for the solution of the nonlinear programming problem to be optimal. For convenience, we reproduce the KKT conditions for the problem in (9)-(10) in equations (11)-(12), where \( \mathcal{E} \) is the set of equality constraints.

\[
\nabla_x L(x^*, \lambda^*) = 0 \text{ (stationarity)},
\]

\[
c_i(x^*) = 0, \text{ for all } i \in \mathcal{E} \text{ (primal feasibility)}. \] (11) (12)

The gradient of the Lagrangian is given by:

\[
\nabla_x L(x, \lambda) = \nabla f(x) - \lambda^T \nabla c(x).
\]

The KKT conditions of (9)-(10) can be written as a system of equations in the unknowns \( x \) and \( \lambda \):

\[
F(x, \lambda) = \begin{bmatrix} \nabla f(x) - A(x)^T \lambda \\ c(x) \end{bmatrix} = 0.
\]
Any solution \((x^*, \lambda^*)\) of the problem defined in (9)-(10) for which \(A(x^*)\) has full rank satisfies the system above. The resulting system of nonlinear equations is then solved using Newton’s method. The Jacobian of this system with respect to the unknowns is

\[
J(x, \lambda) = \begin{bmatrix}
\nabla^2_{xx} L(x, \lambda) & -A(x)^T \\
A(x)^T & 0
\end{bmatrix}.
\]

By solving the Newton-KKT system defined as

\[
\begin{bmatrix}
\nabla^2_{xx} L(x_k, \lambda_k) & -A(x_k)^T \\
A(x_k)^T & 0
\end{bmatrix} \begin{bmatrix}
p_k \\
p_{\lambda}
\end{bmatrix} = \begin{bmatrix}
-\nabla f(x_k) + A(x_k)^T \lambda_k \\
c(x_k)
\end{bmatrix}
\]

for \(p_k\) and \(p_{\lambda}\), where \(p_k\) and \(p_{\lambda}\) are \(p(x_k)\) and \(p(\lambda_k)\), respectively, we can define our new iterate \((x_{k+1}, \lambda_{k+1})\) as

\[
\begin{bmatrix}
x_{k+1} \\
\lambda_{k+1}
\end{bmatrix} = \begin{bmatrix}
x_k \\
\lambda_k
\end{bmatrix} + \begin{bmatrix}
p_k \\
p_{\lambda}
\end{bmatrix}.
\]

Solutions of the KKT conditions yield stationary points. For our problem, the method converges to a constrained local max due to the monotonic behavior of the line search and the logarithmic barrier term. To find an initial feasible point for the optimization problem for each interior node, we choose three of the node’s neighbors and write the node as a convex combination of the three neighbors, where the neighbors not chosen in the three are given a weight of \(\epsilon\), a small positive constant. As an example of this process, suppose that we are calculating the initial feasible point for node \(v_{13}\) as shown in Fig. 3. If we choose \(v_1, v_2,\) and \(v_3\) as our set of three neighbors, then the resulting linear system has the following form:

\[
\begin{bmatrix}
v_1.x & v_2.x & v_3.x \\
v_1.y & v_2.y & v_3.y \\
1 & 1 & 1
\end{bmatrix} \begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\lambda_3
\end{bmatrix} = \begin{bmatrix}
v_{13}.x - \epsilon \times \left( \sum_{i=4}^{12} v_i.x \right) \\
v_{13}.y - \epsilon \times \left( \sum_{i=4}^{12} v_i.y \right) \\
1 - 9 \times \epsilon
\end{bmatrix}
\]

where \(\lambda_4, \ldots, \lambda_{12} = \epsilon\).

The optimization problem given in (1)-(5) can now be solved for the weights. After solving for the weights, we proceed to the boundary deformation. In the case of high-order mesh generation, the high-order boundary nodes
are projected onto the curved boundary. For this projection step, we assume that the appropriate destinations of the boundary midpoints on the surfaces or curves are given. Now that the weights have been calculated and the boundary has been deformed, we can now turn our attention to calculating the final positions of the interior nodes by solving the linear system defined in (8).

In contrast with the low-order case, in the high-order case, $A_I$ and $A_B$ in (8) represent how interior nodes (including interior midpoints) and boundary nodes (including boundary midpoints) relate to their neighbors, respectively. Based on the construction of $A_I$, we expect the matrix to be structurally symmetric. In other words, if $A_I(i,j) \neq 0$, then $A_I(j,i) \neq 0$. To confirm this, we can examine a spy plot of $A_I$. In Fig. 4, we show the spy plots of the $A_I$ matrices for the set of high-order mesh generation test cases in Section 4. To solve (8), we use an LU factorization of $A_I$, followed by the solution of the two triangular systems involving $L$ and $U$, respectively.

3.2. High-Order Mesh Warping

As a natural extension of the process that we have described above, we can move all boundary nodes as opposed to only the boundary midpoints. This extension allows more general mesh deformations, in other words, mesh warping. The process for warping a second-order mesh is similar to the process for generating a second-order mesh. Assuming that our method generated the second-order mesh, we can reuse the set of local weights relating each interior node to its neighbors. Given these weights, the deformation is applied to the boundary nodes and the final positions of the interior nodes are calculated as in (8). For warping a second-order mesh that was not generated by the method, we would need to first solve the optimization problem given in (1)–(5) for the set of local weights. After solving for the weights, the method would proceed as before. More specifically, the boundary would be deformed and the new interior node locations would be calculated. When the mesh is warped multiple times, the $L$ and $U$ factors are reused which greatly simplifies the complexity of the method. Furthermore, our method has high potential for improved performance in parallel as was shown for the parallel low-order log barrier warping method [19]. In contrast with some of the other high-order mesh generation methods recently developed, our method presents a unified approach for generating the curved high-order mesh and then warping it.

4. Numerical Results

In this section, we will show several examples which demonstrate the use of our method for high-order mesh generation and warping. For each example, we will show the meshes and discuss the element distortion as measured by the scaled Jacobian [2]. We used the modified definition of the scaled Jacobian as described in [6]. The definition is reproduced in (13) for convenience:

$$\text{scaled Jacobian} = \frac{\min \{ |J\phi(\xi)| \}}{\max \{ |J\phi(\xi)| \}},$$

(13)

where $\xi$ denotes a point from the discrete set of quadrature points of the reference element, and $J\phi(\xi)$ is the Jacobian of the isoparametric mapping from local to physical coordinates. The quadrature points are chosen to be the original low-order nodes and the newly added midpoints. The combination of these points gives us one set of Gaussian quadrature rules that is defined on triangles and tetrahedra.

The first two examples demonstrate our method on 2D examples. The first example involves generating the second-order mesh of an annulus. Figure 5(a-c) show an initial linear mesh, the curved second-order mesh, and a histogram plot of the scaled Jacobian for the second-order mesh. The second example is a parametric gear. In Fig. 6, we show the initial linear mesh, the curved second-order mesh, and the histogram plot of the scaled Jacobian. From the histogram plots, it is evident that the second-order mesh of the annulus has elements with minimal distortion, while the parametric gear has some elements with more distortion in the areas of high curvature.

For our third and fourth examples, we illustrate the usage of our method on 3D examples. For each 3D example, we show the linear mesh, the curved second-order mesh, and a histogram plot of the scaled Jacobian. With this in mind, our third example is a door hinge as shown in Fig. 7. Our fourth and final generation example is a cylindrical shell of
Fig. 4: Spy plots of $A_I$: (a) annulus, (b) gear, (c) door hinge, and (d) shell.

height 0.15 created from an extrusion of a coarsened version of the annulus as shown in Fig. 8. With the exception of the parametric gear and door hinge, our method generates second-order meshes with minimal distortion. By using a finer initial mesh for the parametric gear and door hinge examples, the distortion would likely decrease. In contrast with some of the other high-order mesh generation methods, our method does not guarantee a valid high-order mesh (e.g., elements with a positive scaled Jacobian) since the interior nodes are moved based on their weights and the deformation applied to the boundary. In practice, if our initial mesh is fine enough, then the deformation applied to the high-order boundary midpoints to place them on the true curved boundary will be small, and the mesh elements will generally be valid.

Aside from second-order generation, our method can also be used for warping a generated second-order mesh. In Fig. 9, we warp the second-order annulus that we generated for Fig. 5. For this example, we increase the size of the inner ring of the annulus. In Figs. 9(c,e), we show the mesh after increasing the ring by 1.25x and 1.5x, respectively. We also show the histogram plots of scaled Jacobian at each stage. As we can see from the figures, as the size of the inner ring increases, so does the distortion of the elements along the inner boundary. In addition to dilation, we also explore the effect of rotation. In Fig. 10, we rotate the inner ring of the shell 15 degrees counterclockwise. In Fig. 11, we rotate the inner ring of the shell by an additional 15 degrees, for a total rotation of 30 degrees counterclockwise. In each case, an incremental rotation of 1 degree was applied. Similar to the rotation test conducted in [16], the distortion of the elements along the inner cylinder increases as the angle of rotation increases. These two examples demonstrate that minor boundary deformations can result in element distortion. For more severe boundary deformations, it is expected that the mesh quality will continue to degrade. In particular, the mesh may even become tangled for sufficiently large boundary deformations. This suggests that the method is suitable for small- to medium-sized deformations, which is consistent with what is observed for the original LBWARP method. To address...
Fig. 5: Annulus example: (a) a linear mesh of the annulus, (b) the curved second-order mesh, and (c) a histogram plot of the scaled Jacobian of the second-order mesh in (b).

Fig. 6: Parametric gear example: (a) a linear mesh of the gear, (b) the curved second-order mesh, (c) a histogram plot of the scaled Jacobian of the second-order mesh in (b).
these limitations, we plan to explore the impact of different choices of the neighboring set for interior points. We will also try recalculating the weights before the next warping step for larger deformations. Finally, we will develop a hybrid warping and untangling method for use with large deformations in an attempt to correct the tangled elements.

In Table 1, we list the number of elements and wall clock execution time for each of our numerical examples. The code was run using Matlab R2016b. The execution times were measured on a Surface Pro 4 with 8GB of RAM and a Core i7 2.2GHZ CPU.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
<th>Number of Elements</th>
<th>Wall Clock Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>generation</td>
<td>annulus</td>
<td>268</td>
<td>22.01</td>
</tr>
<tr>
<td>generation</td>
<td>gear</td>
<td>424</td>
<td>35.35</td>
</tr>
<tr>
<td>generation</td>
<td>door hinge</td>
<td>1112</td>
<td>17.42</td>
</tr>
<tr>
<td>generation</td>
<td>shell</td>
<td>1368</td>
<td>106.09</td>
</tr>
<tr>
<td>warping</td>
<td>annulus</td>
<td>268</td>
<td>22.05</td>
</tr>
<tr>
<td>warping</td>
<td>shell</td>
<td>1368</td>
<td>111.39</td>
</tr>
</tbody>
</table>

Table 1: Table showing the number of elements and the wall clock time for each of the numerical examples.
5. Conclusions

We have presented a new \textit{a posteriori} method for generating and warping curved, second-order meshes. In order to generate a high-order mesh, we first add midpoints to the linear mesh. After adding midpoints to the given linear mesh, our method solves an optimization problem for each interior node to calculate a set of local weights that relate the interior node to its neighbors. Next, our method projects the newly added boundary nodes onto the curved boundary. Lastly, the final positions of the interior nodes are calculated based on the optimal set of weights and the new boundary node positions. Aside from mesh generation, we have also shown that our method can be used for warping second-order meshes by allowing all boundary nodes to be moved.

We note that presently, our numerical examples are not representative of real-world engineering meshes. Toward this end, we plan to move beyond our Matlab implementation by implementing our method in C++ to allow us to tackle larger problems. With this in mind, our future work will include applying our method to more real-world geometries, as well as developing the method for higher orders. In addition, as our warping examples showed, it is expected that mesh quality will continue to degrade for severe boundary deformations. With this in mind, we will explore the strategies described at the end of Section 4 to attempt to improve warping performance.
Fig. 9: Annulus example: (a) the starting second-order mesh, (b) the histogram of the initial second-order mesh, (c) the mesh after increasing the inner ring by 1.25x, (d) the histogram after increasing the size of the inner ring, (e) the mesh after increasing the inner ring by 1.5x, and (f) the histogram after increasing the size of the inner ring.

Acknowledgements

The work of the first author was funded in part by the Madison and Lila Self Graduate Fellowship. The work of the second author was supported in part by NSF CAREER Award ACI-1500487 (formerly ACI-1330054 and OCI - 1054459).
Fig. 10: Cylindrical shell example: (a) the starting second-order mesh, (b) the histogram of the initial second-order mesh, (c) the mesh after rotating the inner ring 15 degrees counter clockwise, and (d) the histogram after rotation of the inner ring.

Fig. 11: Cylindrical shell example: (a) the second-order mesh after rotating the inner ring 15 degrees counter clockwise, (b) the histogram after rotation of the inner ring, (c) the second-order mesh from (a) after rotating the inner ring an additional 15 degrees counter clockwise, and (d) the histogram after the second rotation of the inner ring.
References


