Mesh Curving and Refinement Based on Cubic Bézier Surface for High-Order Discontinuous Galerkin Methods

Shu-Jie Li

Abstract A 3-D curved mesh generator is prescribed for converting linear elements to quadratic elements required by high-order methods, which is based on the reconstruction of Cubic Bézier surfaces. Successive curved mesh refinement is also supported by inquiring the middle nodes of the edges and faces of the reconstructed quadratic elements via the Cubic Bézier surface method. Numerical test cases are shown to demonstrate the capability of both mesh curving and refinement around three-dimensional geometries.

1 Introduction

High-order methods of computational fluid dynamics (CFD) are proven useful in many aspects which can achieve low-dissipation results, e.g., the discontinuous Galerkin methods (DG) [1–8]. However, the total numerical error of using a high-order method can be affected by the approximation error of surface geometries, e.g., see reference [7, 8], where using curved elements are shown to be necessary to eliminate the unphysical entropy wake in DG solvers. So curved elements are widely used in the high-order CFD community and considering the curvature of surface elements is the minimum requirement of achieving high-fidelity, high-order accurate simulations, especially on coarse grids. However, for practical applications, such a requirement is usually not met. In most of the cases, only straight-sided meshes are provided while the CAD files are inaccessible, so that mesh curving has to be considered if high-quality solutions are pursued with high-order methods.

In this paper, a 3-D curved mesh generator is prescribed for converting linear elements to quadratic elements, which is required by most high-order methods, especially DG. To recover curvature from the provided linear mesh, cubic Bézier

S.-J. Li (✉)
Beijing Computational Science Research Center (CSRC), Beijing, China
e-mail: shujie@csrc.ac.cn

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 131,
https://doi.org/10.1007/978-3-030-23436-2_15

surfaces [9, 10] are reconstructed upon the numerical approximation of nodal normals.

Successive curved mesh refinement is also realized by inquiring the middle nodes of the edges and faces of the reconstructed quadratic elements. The mesh curving and refinement are applied to the DLR-F6 configuration [11].

2 Mesh Curving

In this section, mesh curving strategies are studied based on the cubic Bézier surface reconstruction [9, 10]. A discontinuous Galerkin flow solver with the physical orthogonal basis [1–6] is used for assessing the solution quality with approximated curved boundaries. Finally, the mesh curving is applied to the DLR-F6 configuration.

2.1 Cubic Bézier Surface

In the first mesh curving strategy, we consider the mesh curving problem under the framework of cubic Bézier surface reconstruction [9, 10]. In most cases, the surface mesh only consists of triangles and/or quadrilaterals. Correspondingly, the cubic Bézier triangle and quadrilateral are used, as illustrated in Fig. 1a, b.

Compared with linear elements, extra mid-edge points (such as p_{m1}, p_{m2}, p_{m3} in Fig. 1a, b) are needed to be computed with the cubic Bézier curves for defining the quadratic elements. In this section, we outline the theory of cubic Bézier Surface firstly. Without loss of generality, considering a n-order of Bézier curve in the form of

$$\mathbf{B}(t) = \sum_{i=0}^{n} \mathbf{b}_{i,n}(t)\,\mathbf{P}_i, \qquad t \in [0, 1], \tag{1}$$

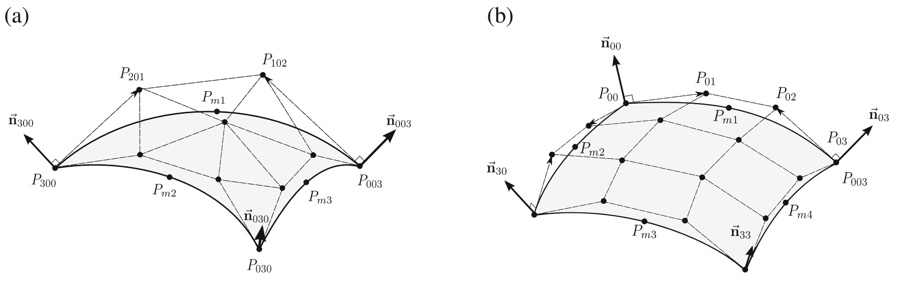

Fig. 1 Cubic Bézier surface. (**a**) Triangle. (**b**) Quadrilateral

where $\mathbf{P}_i$ is the ith control node of the Bézier curve, $\mathbf{B}(t)$ is the resulting parametric curve and term $\mathbf{b}_{i,n}(t)$ is called the *Bernstein basis polynomials* which may be written as

$$\mathbf{b}_{i,n}(t) = \binom{n}{k} t^i \, (1-t)^{(n-i)}, \quad i = 1, \dots, n \tag{2}$$

Note that $t^0 = 1$, $(1-t)^0 = 1$ for any t value, and that the *binomial coefficient* $\binom{n}{k}$ is defined as

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3}$$

The first and last control nodes coincide with the end points of the edge, so only the two internal control nodes are computed. The derivative of the Bézier curve with a respect to t is actually a tangential vector $\mathbf{T}$:

$$\mathbf{T}(t) = \mathbf{B}'(t) = n \cdot \sum_{i=0}^{n-1} (\mathbf{P}_{i+1} - \mathbf{P}_i) \cdot \mathbf{b}_{i,n-1}(t) \tag{4}$$

The cubic Bézier curve with $n = 3$ is shown in Fig. 1a. This triangle consists of the three corner nodes $\mathbf{P}_{300}$, $\mathbf{P}_{030}$, $\mathbf{P}_{003}$, and the three computed surface normals $\mathbf{n}_{300}$, $\mathbf{n}_{030}$, and $\mathbf{n}_{003}$. To reconstruct the cubic Bézier curve for each edge of the triangle, for instance on the edge $\mathbf{P}_{003} \rightarrow \mathbf{P}_{300}$, the tangential vector is computed as (4) where $\mathbf{T}(0)$ and $\mathbf{T}(1)$ must be in the same vector direction.

$$\begin{aligned} \mathbf{T}(0) &= \mathbf{P}_{300} - \mathbf{P}_{003} - ((\mathbf{P}_{300} - \mathbf{P}_{003}) \cdot \mathbf{n}_{030})\, \mathbf{n}_{030} \\ \mathbf{T}(1) &= \mathbf{P}_{300} - \mathbf{P}_{003} - ((\mathbf{P}_{300} - \mathbf{P}_{003}) \cdot \mathbf{n}_{300})\, \mathbf{n}_{300} \end{aligned} \tag{5}$$

By solving (5), the control nodes are expressed as

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{P}_{003} \\ \mathbf{P}_1 &= \mathbf{P}_{102} = (2\mathbf{P}_{003} + \mathbf{P}_{300} - ((\mathbf{P}_{300} - \mathbf{P}_{003}) \cdot \mathbf{n}_{030})\, \mathbf{n}_{030})/3 \\ \mathbf{P}_2 &= \mathbf{P}_{201} = (2\mathbf{P}_{300} + \mathbf{P}_{003} + ((\mathbf{P}_{300} - \mathbf{P}_{003}) \cdot \mathbf{n}_{300})\, \mathbf{n}_{300})/3 \\ \mathbf{P}_3 &= \mathbf{P}_{300} \end{aligned} \tag{6}$$

With the cubic Bézier curve formula (1) defined by (6), the middle node of each edge can be computed by taking $t = 0.5$. For both six-node triangle and eight-node quadrilateral, not mid-face nodes exist, thus the reconstruction is actually done on edges but not surfaces. The same procedure can be applied to all the edges on the wall surfaces. Finally, the curved elements can be output into a mesh file for high-order solvers.

2.1.1 Calculation of Surface Normals

The cubic Bézier surface is defined by the nodal coordinates and normals, so only the normals are required. The normal $\mathbf{n}_i$ at the ith node is calculated by the inverse-distance weighting (IDW) [12] of its N surrounding surface normals as in (7), where r_j is the distance from the ith node to its jth neighboring surface center.

$$\omega_i = \frac{{r_i}^{-2}}{\sum_{j=1}^{N} {r_j}^{-2}}, \qquad \mathbf{n}_i = \sum_{j=1}^{N} \omega_j \mathbf{n}_j \tag{7}$$

IDW satisfies $\sum_{j=1}^{N} \omega_j = 1$ and it is thus linear preserving. And, IDW also produces nonlinear interpolation when the scalar vectors are distributed nonlinearly. One of the greatest advantages of using IDW is that the interpolated normals can be bounded by the maximum and minimum of the data points. Thus, the safety of normal reconstruction is enhanced without introducing any dangerous new extremum.

The normals are then computed for all the wall nodes which are the most significant parts causing the geometric error. At each node, a single normal is computed despite that its surrounding surface cells could be on different patches. When the node is at the intersection of multiple patches, the computed normal is not well defined. The order of mesh curving should be conducted firstly on the symmetric wall, then the solid wall and followed by the far-field boundary surfaces. By doing this the normal singularity can be partially removed. The remaining singular normals probably have bad vector directions which differ greatly to their surrounding computed normals. So the angle α between two end-point normals of an edge is thus can be used as a measure of singularity.

A threshold angle control strategy for avoiding the singularity is tested effective in which the edges with the singular angle $\alpha \geq 30 \sim 60^\circ$ are kept straight and the midpoints are obtained by geometric averages of end-point nodes while all other edges are recovered to curves. While the volume mid-edge nodes are simply obtained by geometric averages of end-point nodes. Note that the entire algorithm is fully automatically without human inventions.

2.1.2 Numerical Tests for the Mesh Curving

In this section, the mesh curving strategies are firstly tested for the numerical solutions of the inviscid Euler equations by a high-order discontinuous Galerkin flow solver [1, 2]. Smooth inviscid flows past a sphere are computed which admits a symmetric flow pattern and a constant entropy across the computational domain in theory, so that the error due to curved mesh approximations can be computed quantitatively.

Both the linear mesh and the curved method are tested, which show that the unphysical wake flow produced by using the linear grids is greatly improved by using the curved mesh, see also [7, 8]. In this test, the fourth-order DG is used for spatial discretization and a fast first-order exponential time integration (EXP1) [4] with CFL $= 10^3$ is used for the steady-state time marching. Figure 2 presents the steady-state residual convergences of the linear and curved meshes. It is so obvious that meaningful high-order accurate solutions can be obtained only if the curved mesh is employed, while the convergence of the linear mesh shows stagnation due to the production of unphysical wakes, as shown in Fig. 3. The error is measured quantitatively with the entropy variable which has an exact value for smooth inviscid flows. The entropy error is defined as

$$E_{L_2(\Omega)}(s) := \sqrt{\frac{1}{|\Omega|}\int_{\Omega}\left(\frac{s}{s_0}\right)^2 \mathrm{d}V - 1} \tag{8}$$

where Ω is the cell element and $|\Omega|$ is the cell volume. $s := P/\rho^{\gamma}$ is the entropy, P the pressure and ρ the density. s_0 is the exact entropy value of the free stream. In Fig. 2 (left), it is observed that a steady-state convergence cannot be realized with the linear mesh. The entropy error with the linear mesh stagnates at $10^{-3.42}$, while the one with the curved mesh converged at $10^{-5.32}$, as shown in Fig. 2 (right). So the curve mesh reconstruction does help on reducing the errors induced by the geometry approximations. In Fig. 3 (left), large entropy errors with the linear mesh occur in the wake flow region due to the straight-sided surface approximations and the flow symmetry is broken as shown in the pressure contour of Fig. 4 (left) which is unphysical. In contrast, using the recovered curved mesh indeed gives theoretically consistent results along with the fourth-order discontinuous Galerkin methods, in which the pressure contour is symmetric, as shown in Fig. 4 (right) and the local entropy error $S_{err} := P/\rho^{\gamma} - P_0/\rho_0^{\gamma}$ is much smaller than the linear one, see Figs. 3 (left) and 2 (right). This case shows that using the mesh curving method is indeed helpful in reducing the errors produced by the linear surface approximations.

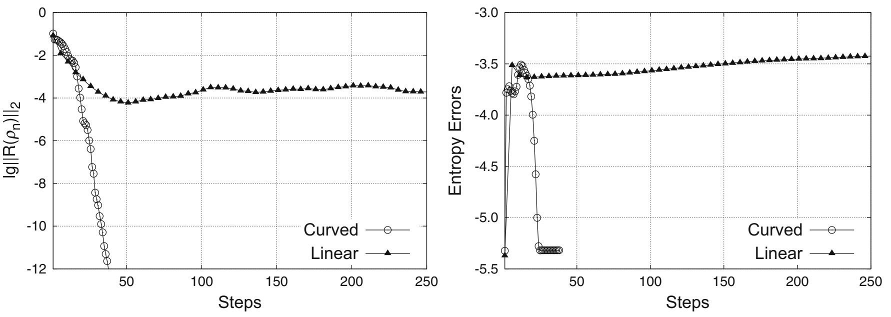

Fig. 2 Smooth flow past a sphere: convergence history of the density residuals $\lg \|R(\rho_n)\|_2$ with the curved mesh (left) and the linear mesh (right). Rapid convergence can be achieved by using the curved mesh while the unphysical wake flow with the linear mesh leads to residual stagnation

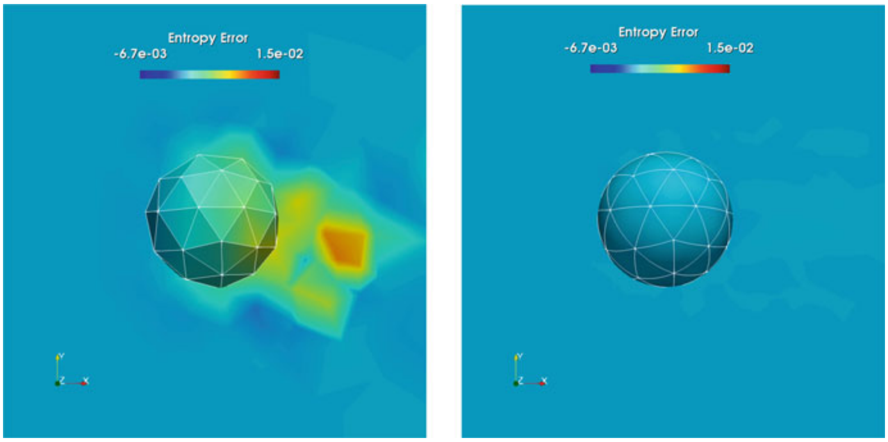

Fig. 3 Smooth flow past a sphere: comparison of entropy error with the curved mesh (left) and the linear mesh (right). The local entropy error $S_{err} := P/\rho^\gamma - P_0/\rho_0^\gamma$. The linear-element approximation of curved surface induces a large entropy wake

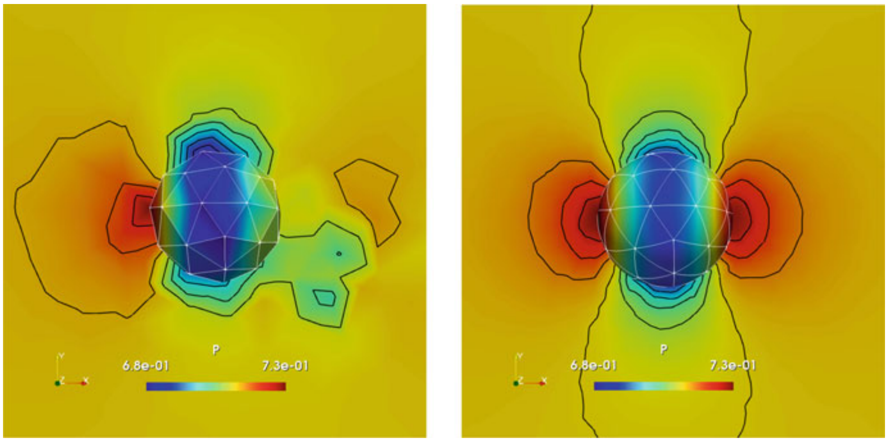

Fig. 4 Smooth flow past a sphere: comparison of pressure contours with the curved mesh (left) and the linear mesh (right). The result of the linear mesh gives a unphysical wake flow pattern

Next, the mesh curving strategies are applied to a three-dimensional complex geometry: DLR-F6 configuration. The surface hybrid grids of DLR-F6 has 34,676 nodes, 54,804 linear elements, and 11,150 linear edges. First, nodal normals are computed according to Sect. 2.1.1. In Fig. 5, the computed normals at each node are shown in blue. Then by applying the mesh curving using the Cubic Bézier surface reconstruction with setting $\alpha = 60°$, 98.6% of 11,150 edges are converted to curves and 161 edges are kept straight. The uncurved, straight edges are marked with red spheres and are shown in Fig. 6. From which we can see that most of the edges are

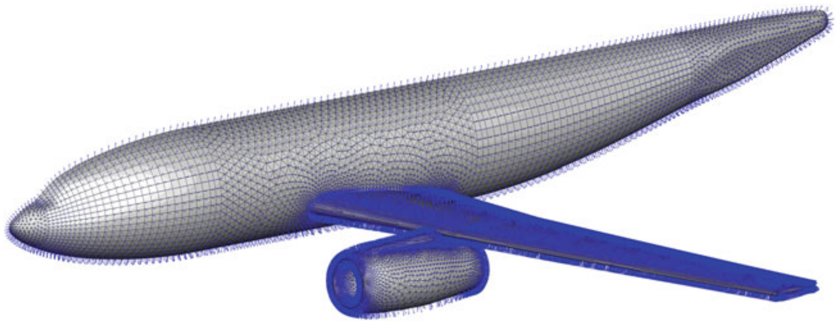

Fig. 5 DLR-F6 configuration: computed nodal normals; normal (blue), edge (black)

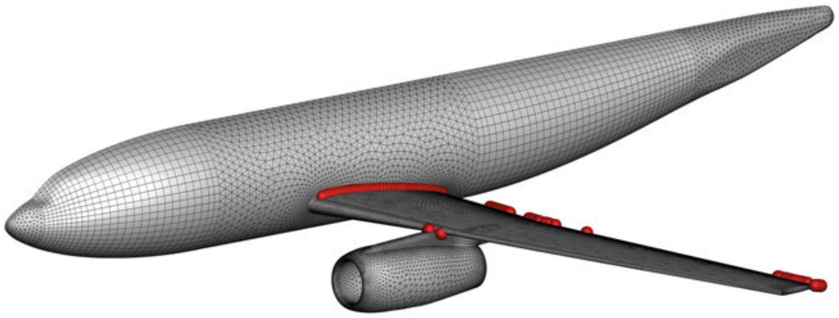

Fig. 6 DLR-F6 configuration: marking uncurved edges with red spheres

converted to curves except four tiny regions which contain uncurved edges. Zoom-in views of these four regions are shown in Fig. 7, where the straight edges are marked with red spheres and the curved edges are denoted by multiple black nodes. Finally, the curved surface mesh is shown in the left one of Fig. 8, which is also used for curved mesh refinements in Sect. 3.

3 Curved Mesh Refinement

In this section, a curved mesh refinement tool is developed based on the mesh curving developed in Sect. 2. Global mesh refinements are realized by inquiring curved mesh information such as the midpoints of edges and faces. Edge- and face-based arrays are thus created for storing newly generated middle nodes of edges and faces without duplicating the node storage. A convention of elemental node numbering has to be defined to form subcell topologies which can be chosen arbitrarily but must be consistent during mesh refinements. The full procedure mesh refinement algorithm is shown in Algorithm 1 in details.

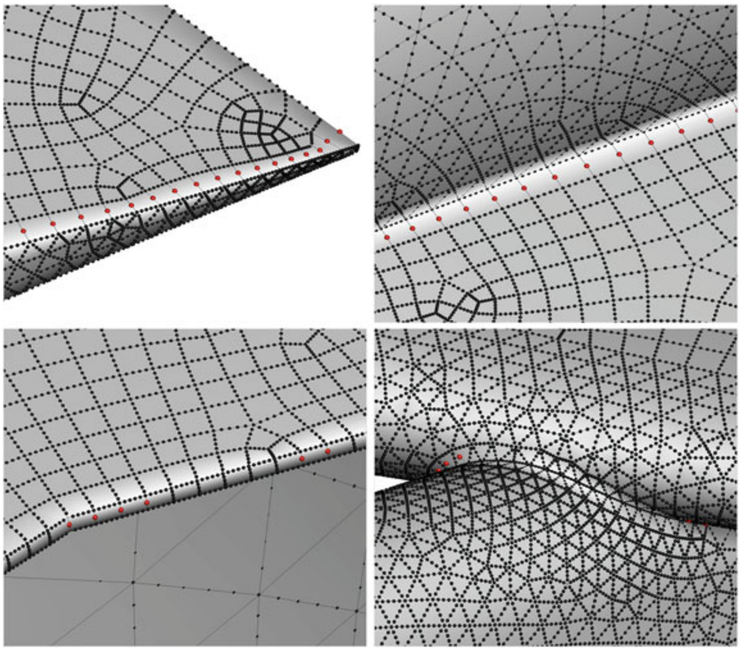

Fig. 7 DLR-F6 configuration: marking uncurved edges with the red sphere

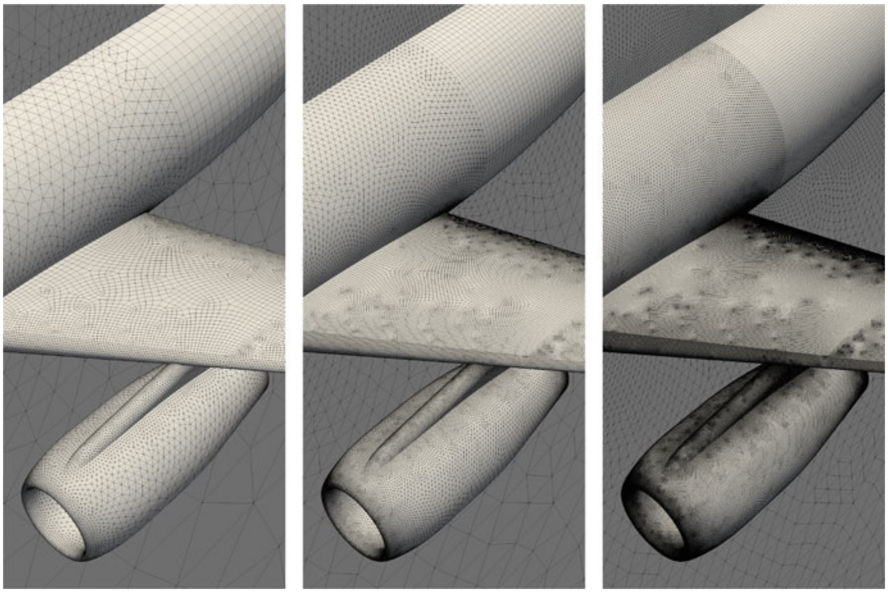

Fig. 8 DLR-F6 configuration: successive curved mesh refinements. Number of cells: 1,345,335 (left), 10,791,182 (middle), 86,500,468 (right)

Algorithm 1 Curved mesh refinement

```
1:  Recover cubic Bézier surfaces from the provided linear mesh
2:  for Each level of mesh refinement $iLevel$ do
3:      if $iLevel = 1$ then
4:          Compute mid-edge & mid-face nodes from the cubic Bézier surfaces
5:      else
6:          Read in internal mesh file
7:      end if
8:      Form quadratic isoparametric elements
9:      Update face list & edge list
10:     if find tetrahedron then
11:         Split tetrahedron with considerations of mesh quality [13]
12:     end if
13:     Split quadratic isoparametric elements and store quadratic children cells
14:     Dump the internal mesh file
15: end for
```

This quadratic elements refinement process is iteratively called with storing an internal mesh file for each refinement level. Note that only the initialization stage requires the participation of cubic Bézier curves and the refinement levels greater than two use the shape function interpolations for saving computational cost. The resulting quadratic elements approximate the curvature of cubic Bézier surfaces but are immediately compatible with traditional finite element solvers.

In the quadratic elements approximation, the mapping from reference coordinate $(\xi, \eta) \in [(0, 1), (0, 1)]$ to physical coordinate $\mathbf{x}$ on a triangle is defined as $\mathbf{x} = \sum_{i=1}^{6} N_i \mathbf{x}_i$ where $\zeta = 1 - \xi - \eta$. For a quadrilateral, the reference coordinate is $(\xi, \eta) \in [(-1, 1), (-1, 1)]$ while $\mathbf{x} = \sum_{i=1}^{8} N_i \mathbf{x}_i$. The middle points of edges can be computed easily instead of using Bézier curves. The shape functions of the quadratic surface elements are listed in Table 1. In the step 13 of Algorithm 1, the quadratic children cells are computed by computing the mid-edge points using the shape functions of Table 1. In the step 12, according to [13], there are three ways of conducting tetrahedron splitting. Among all the splitting possibilities, we split the tetrahedron with choosing the shortest average edge length of the new children cells.

The mesh refinement method is applied to the DLR-F6 configuration used for the mesh curving in Sect. 2.1.2. Two levels of successive mesh refinements are applied. The resulting meshes are shown in Fig. 8 with volume cell number 1,345,335, 10,791,182 and 86,500,468, from left to right. The zoom-in views of the inlet of the nacelle are displayed in Fig. 9, from which we can see that the feature of the high-curvature surface of the inlet is smoothly captured for each level of curved mesh refinement. Because the mesh curving is only conducted on the boundaries, the mesh refinement results in linear volume elements away from the boundaries. It deserves a future study to spread the mesh curving effects from boundaries to the whole computational domain.

Table 1 Quadratic surface elements

Shape function	Six-node triangle	Eight-node quadrilateral
N_1	$\xi\,(2\xi - 1)$	$\frac{1}{4}\,(1-\xi)\,(\eta-1)\,(\xi+\eta+1)$
N_2	$\eta\,(2\eta - 1)$	$\frac{1}{4}\,(1+\xi)\,(\eta-1)\,(\eta-\xi+1)$
N_3	$\zeta\,(2\zeta - 1)$	$\frac{1}{4}\,(1+\xi)\,(\eta+1)\,(\xi+\eta+1)$
N_4	$4\xi\eta$	$\frac{1}{4}\,(\xi-1)\,(\eta+1)\,(\xi-\eta+1)$
N_5	$4\eta\zeta$	$\frac{1}{2}\,(1-\eta)\,\left(1-\xi^2\right)$
N_6	$4\xi\zeta$	$\frac{1}{2}\,(1-\xi)\,\left(1-\eta^2\right)$
N_7	n/a	$\frac{1}{2}\,(1+\eta)\,\left(1-\xi^2\right)$
N_8	n/a	$\frac{1}{2}\,(1-\xi)\,\left(1-\eta^2\right)$

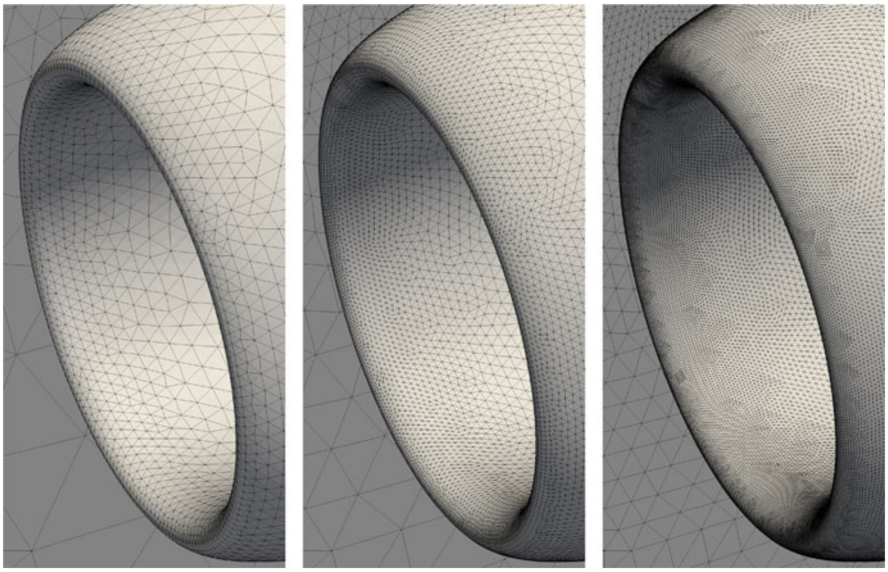
Fig. 9 DLR-F6 configuration: zoom-in views of the nacelle grids. The high-curvature surfaces are smoothly approximated with the curved mesh refinements

4 Conclusion

Mesh curving and refinement algorithms applied to high-order discontinuous Galerkin methods are presented. The mesh curving algorithm uses the cubic Bézier surface/curves reconstruction to recover curvature from the provided linear mesh, thus, can upgrade a linear mesh to a quadratic one. Comparing with other mesh curving techniques [14–16], the current strategy is much easier to implement while does not require surface fitting. Moreover, it supports both hybrid element types curving and also global curved mesh refinement. Numerical tests demonstrate that by using curved elements along with the high-order methods, unphysical entropy errors can be significantly reduced. The mesh curving and refinement can be applied to three-dimensional geometries, such as the DLR-F6 configuration, as shown in the paper.

Finally, even if the mesh curving and refinement approaches turned out to be robust, further research is needed to increase the accuracy of the curved surface approximation. Moreover, the feature sharping/feature detection and volume elements curving algorithms need to be further investigated, especially for more complex geometries.

Acknowledgements This work is funded by the National Natural Science Foundation of China (NSFC) under the Grant U1530401. The author thanks Dr. Hang Si of WIAS, Germany, for the discussion and collaboration in a broad sense of computational geometry. SJL would further like to thank the reviewers for their helpful comments.

References

1. Li, S.-J.: A parallel discontinuous Galerkin method with physical orthogonal basis on curved elements. Proc. Eng. **61**, 144–151 (2013)
2. Li, S.-J., Wang, Z.J., Ju, L., Luo, L.-S.: Explicit large time stepping with a second-order exponential time integrator scheme for unsteady and steady flows. AIAA Paper, 2017–0753
3. Li, S.-J., Wang, Z.J., Ju, L., Luo, L.-S.: Fast time integration of Navier-Stokes equations with an exponential-integrator scheme. AIAA Paper, 2018–0369
4. Li, S.-J., Luo, L.-S., Wang, Z.J., Ju, L.: An exponential time-integrator scheme for steady and unsteady inviscid flows. J. Comput. Phys. **365**, 206–225 (2018)
5. Li, S.-J.: Efficient p-multigrid method based on an exponential time discretization for compressible steady flows. arXiv:1807.0115
6. Li, S.-J., Ju, L.: Exponential time-marching method for the unsteady Navier-Stokes equations. AIAA Paper, 2019-0907
7. Bassi, F., Rebay, S.: High-order accurate discontinuous finite element solution of the 2D Euler equations. J. Comput. Phys. **138**(2), 251–285 (1997)
8. Krivodonova, L., Berger, M.: High-order accurate implementation of solid wall boundary conditions in curved geometries. J. Comput. Phys. **211**(2), 251–285 (2006)
9. Yamaguchi, F.: Curves and Surfaces in Computer Aided Geometric Design. Springer, Heidelberg (1988)
10. Vlachos, A., Peters, J., Boyd, C., et al.: Curved PN triangles. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics. ACM Press, New York (2001)
11. Brodersen, O., Stürmer, A.: Drag prediction of engine-airframe interference effects using unstructured Navier-Stokes calculations. AIAA Paper, 2001–2414
12. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 ACM National Conference, pp. 517–524 (1968)
13. Zhang, S.: Successive subdivisions of tetrahedra and multigrid methods on tetrahedral meshes. Houston J. Math. **21**, 541–556 (1995)

14. Hindenlang, F., Bolemann,T., Munz, C.-D.: Mesh curving techniques for high order discontinuous Galerkin simulations. In: IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach, pp. 133–152. Springer, Berlin (2015)
15. Ims, J., Duan, Z., Wang, Z.J.: meshCurve: an automated low-order to high-order mesh generator. AIAA Paper, 2015–2293
16. Jiao, X., Wang, D.: Reconstructing high-order surfaces for meshing. Eng. Comput. **28**(4), 361–373 (2012)