Automatic metric 3D surface mesh generation using subdivision surface geometrical model. Part 2 : mesh generation algorithm and examples

Lee, Chi King

2003

Lee, C. K. (2003). Automatic metric 3D surface mesh generation using subdivision surface geometrical model. Part 2: Mesh generation algorithm and examples. International journal for numerical methods in engineering, 56(11), 1615-1646.

<https://hdl.handle.net/10356/103270>

<https://doi.org/10.1002/nme.631>

© 2003 John Wiley & Sons, Ltd. This is the author created version of a work that has been peer reviewed and accepted for publication by International Journal for Numerical Methods in Engineering, John Wiley & Sons, Ltd. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at:
[DOI:<http://dx.doi.org/10.1002/nme.631>].

Downloaded on 26 Apr 2023 20:24:23 SGT

Automatic metric 3D surface mesh generation using subdivision surface geometrical model

Part II: Mesh generation algorithm and examples

*C. K. Lee

*School of Civil and Environmental Engineering
Nanyang Technological University
Nanyang Avenue
Singapore 639798*

*E-mail address: ccklee@ntu.edu.sg

(Pre-printed version of the paper in IJNME Vo. 56, 1615-1646, 2003)

Summary

In this paper, a new metric advancing front surface mesh generation scheme is suggested. This new surface mesh generator is based on a new geometrical model employing the interpolating subdivision surface concept. The target surfaces to be meshed are represented implicitly by interpolating subdivision surfaces which allow the presence of various sharp and discontinuous features in the underlying geometrical model. While the main generation steps of the new generator are based on a robust metric surface triangulation kernel developed previously, a number of specially designed algorithms are developed in order to combine the existing metric advancing front algorithm with the new geometrical model. As a result, the application areas of the new mesh generator are largely extended and can be used to handle problems involving extensive changes in domain geometry. Numerical experience done indicates that, by using the proposed mesh generation scheme, high quality surface meshes with rapid varying element size and anisotropic characteristics can be generated in a short time by using a low-end PC. Finally, by using the pseudo-curvature element-size controlling metric to impose the curvature element-size requirement in an implicit manner, the new mesh generation procedure can also generate finite element meshes with high fidelity to approximate the target surfaces accurately.

KEYWORDS: *Metric 3D anisotropic adaptive surface mesh generations; Advancing front technique; Interpolating subdivision surface; Sharp and discontinuous features on 3D surface meshes*

1. Introduction

The problem of anisotropic surface mesh generations currently received much attention since it is a prerequisite for many useful applications such as shell analyses [1,2], metal forming studies [3] and in biomechanics modellings [4]. Recently, with the rapid advancement in the element size and shape controls using the metric specifications approach [5-8], the problem of generating good surface triangular meshes over smooth analytical parameterized surfaces can be considered as largely solved [9,10]. In the area of surface mesh generation, the two major problems that remained to be solved are

- (1) The development of versatile while robust algorithms for the generation of anisotropic quadrilateral meshes with good element shape and size controls [11,12].
- (2) The development of surface mesh generation algorithms for the discretization of poorly parameterized and unsmooth surfaces [13-16].

The objective of the present study is to develop a new automatic mesh generator for the generation of well-graded anisotropic meshes for poorly parameterized or unsmooth surfaces. In the first part of this study [13], a novel geometrical model based on the interpolating subdivision surface concept [18-20] is presented. In addition, methods to create sharp features such as cusp nodes and crease lines by the tagging procedure have also been described. Furthermore, surfaces generation examples were also presented to demonstrate the flexibility of the model to create surfaces with different smoothness and curvature properties. In the second part of this study, attentions will be focused on the development of a new mesh generation scheme for the discretization of surfaces. In addition to provide a general description of the mesh generation scheme, this paper will emphasize on the algorithmic developments for the following special features of the new mesh generator:

- (1) Parameterization of the subdivision surface model
- (2) Computations of surface geometrical properties
- (3) Boundary and domain discretizations
- (4) Special treatments for sharp features

In the next section, a brief review of all the basic elements needed for metric surface triangulation will be given. It will then be followed by the description of the parameterization procedure. Note that when comparing with meshing algorithms based on analytical surface patches, the parameterization procedure used in the current study is more complicated. When analytical surface patches are used, the parametric space of the geometrical model is stationary (constant). However, by using subdivision surfaces as the underlying geometrical

model, it is possible that the parameterization can be updated and changed after each mesh refinement step. After the description of the parameterization procedure, the algorithms for the computations of some important surface geometrical properties will be presented. They will then be followed by the descriptions of the algorithms used for boundary and domain discretizations. As the metric advancing front technique (MAFT) for surface triangulation can be considered as a matured method, in this paper, attentions will be focused on those algorithms specially designed for combining the MAFT with the subdivision surface model rather than the implementation of MAFT itself. Finally, several mesh generation examples will be provided to demonstrate the performance, robustness and flexibility of the new mesh generation scheme.

2. Basic elements for metric surface triangulation

In this section, brief summaries of some essential elements needed for metric surface triangulation will be given. Details of them can be found in references [5], [7] and [9].

(1) The first fundamental form of surface mapping

In this study, it is assumed that the target surface to be gridded is represented by a bi-variate mapping of the form

$$(x, y, z)^T = \mathbf{r}(u, v) \quad (1)$$

where the mapping function $\mathbf{r}$ could be defined either implicitly or explicitly. The first fundamental form of Eqn. 1 which governs the length scale transformation of the mapping is defined as

$$\mathbf{dx} = (dx, dy, dz)^T = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix} = (\mathbf{r}_{,u} \quad \mathbf{r}_{,v}) \mathbf{du} \quad (2)$$

(2) Metric specifications

The metric tensor in 3D space that defines the user specification of element size characteristics is a 3×3 matrix, $\mathbf{M}_{3D}$, of the form

$$\mathbf{M}_{3D} = \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \mathbf{e}_3] \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \mathbf{e}_3]^T \quad (3)$$

such that $\text{Det}(\mathbf{M}_{3D}) > 0$. $(\mathbf{e}_i, \lambda_i)$, $i=1,2,3$ are the eigenpairs of $\mathbf{M}_{3D}$ such that $\mathbf{e}_i \cdot \mathbf{e}_j = \delta_{ij}$ with $\lambda_i = \frac{1}{(h_i)^2} > 0$ and h_i is the principal element size in the $\mathbf{e}_i$ direction. The eigenpair $(\mathbf{e}_i, \lambda_i)$ will determinate the stretching ratio and direction of the mesh. Furthermore, the length of an elementary vector, $\mathbf{dx}$, in the normalized space can be expressed as

$$\mathbf{dx}^T \mathbf{dx} = \mathbf{dx}^T \mathbf{M}_{3D} \mathbf{dx} \quad (4)$$

By combing Eqn. 4 with Eqn. 2, the relationship between $\mathbf{dx}$ and $\mathbf{du}$ is given by

$$\mathbf{dx}^T \mathbf{dx} = \mathbf{dx}^T \mathbf{M}_{3D} \mathbf{dx} = \mathbf{du}^T (\mathbf{r}_u \quad \mathbf{r}_v)^T \mathbf{M}_{3D} (\mathbf{r}_u \quad \mathbf{r}_v) \mathbf{du} = \mathbf{du}^T \mathbf{M}_{\text{sur}} \mathbf{du} \quad (5)$$

Thus, the 2×2 matrix, $\mathbf{M}_{\text{sur}}$, defined as

$$\mathbf{M}_{\text{sur}} = (\mathbf{r}_u \quad \mathbf{r}_v)^T \mathbf{M}_{3D} (\mathbf{r}_u \quad \mathbf{r}_v) \quad (6)$$

is the metric tensor which combines the effects of the user specification and the surface mapping and will be used for element size control during mesh generation.

(3) Curvature element-size control

In many surface mesh generation applications, it is necessary to ensure that the finite element meshes are good adaptive approximations of the target surfaces. It was found that the maximum allowable element size needed to obtain a good approximation of the target surface will depend on the principal curvatures and directions of the surface. One convenient way to carry out curvature element-size control is to combine the *pseudo-curvature element-size controlling metric*, $\mathbf{M}_{\text{pc3D}}$, which can be expressed in the form [7]

$$\mathbf{M}_{\text{pc3D}} = [\mathbf{d}_{\min}, \mathbf{d}_{\max}, \hat{\mathbf{n}}] \begin{bmatrix} \left(\frac{\bar{\kappa}_{\min}}{\omega \cdot g(\epsilon)} \right)^2 & 0 & 0 \\ 0 & \left(\frac{\bar{\kappa}_{\max}}{\omega \cdot g(\epsilon)} \right)^2 & 0 \\ 0 & 0 & k \end{bmatrix} [\mathbf{d}_{\min}, \mathbf{d}_{\max}, \hat{\mathbf{n}}]^T \quad (7)$$

with the metric tensor $\mathbf{M}_{\text{sur}}$ (Eqn. 6). In Eqn. 7, $\mathbf{d}_{\min}$ and $\mathbf{d}_{\max}$ are the principal directions of the surface at the point under consideration. $\hat{\mathbf{n}}$ is the normal vector at the point while $g(\epsilon)$ is a function of ϵ , the surface accuracy tolerance defined by the user and k is any constant ≥ 0 . The terms ω , $\bar{\kappa}_{\min}$ and $\bar{\kappa}_{\max}$ are functions of the principal curvatures of the

surface and the bounds of the element size defined by the user. Since the principal curvatures and directions of the surface are functions of the second order derivatives $\mathbf{r}_{uu}$, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$ of the surface mapping, in order to carry out curvature element-size control, it is required that the underlying geometrical model must satisfy some minimum continuity requirements so that $\mathbf{r}_{uu}$, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$ can be computed at least at some discrete points in the parametric space.

3. Parameterization of the geometrical model

In this study, interpolation subdivision surfaces are employed to construct the geometrical model for the mesh generator. Details of the construction method and the properties of the model have been described in Part I of the current study [13]. This section will be focused on the parameterization procedure of the subdivision surface model which consists of the following steps:

- (1) The problem domain is represented as a surface patch with a finite number of sub-surfaces.
- (2) The shape (and other geometrical properties) of a sub-surface is solely defined by a *limiting surface definition mesh* (LSDM) associates with it (Fig. 1a). The interpolating subdivision scheme described in Part I of the current study will be applied on the LSDM to implicitly generate the *limiting sub-surface* which defines the domain to be meshed (Fig. 1b). Note that the limiting surface is *not* a supporting surface and no trimming curve is needed to define the boundary of the surface. It exactly defines the domain to be gridded and may contain openings, cusps, cracked lines and other geometrical features.
- (3) Each LSDM is enclosed by a number of *limiting boundary curve definition edges* (LBCDE). These edges are simply the union of a finite number of boundary segments of the LSDM. The boundary subdivision scheme described in reference [13] is applied to generate the *limiting boundary curves* (LBC) which define the boundary of the limiting sub-surface and may contain boundary cusp nodes (Fig. 1b). Note that the LBCDEs also define the topology of the surface patch by separating different sub-surfaces (Fig. 2). Different topological details such as branching and crack lines can be obtained by arranging the LBEs in an appropriate order.
- (4) The LBCDEs of the surface patch are separated by some *edge termination nodes* (ETN). Note that an ETN only indicates the end of a LBCDE but may not always corresponding to a sharp corner. An example is shown in Fig. 3 for a case that an ETN is a smooth boundary node.

- (5) For every LSDM in the surface patch, there is a corresponding *associated parametric mesh* (APM) defined on a planar $u-v$ space. The APM is the 2D parameterization of the 3D LSDM and defines the parameterization of the geometrical model. To obtain a valid parameterization, it is required that the APM will possess the following properties:
- (i) The range of the APM is always within the unit square $[0,1] \times [0,1]$.
 - (ii) Every element (node) of the LSDM is uniquely mapped to an element (node) in the APM with finite area (Fig. 4). Similarly, the boundary segments on the LSDM is uniquely mapped to the boundary segments of the APM.
 - (iii) Both the LSDM and the APM contain *no* singular point for which a point in the LSDM will be mapped to a line or an element in the APM or vice versa.
 - (iv) The local node numbering of elements in the APM is in anti-clockwise direction and follows exactly the same order as its corresponding element in the APM (Fig. 5).

Note that in step (5), there is no restriction on how the APM is formed. In fact, for a given LSDM, it is possible to construct more than one APM mesh. In many cases, the simplest way to create the APM is by projecting the LSDM onto one of the coordinate plane in the 3D space (Fig. 6). However, for more complicated surfaces, it is sometimes necessary to create artificial cut lines to open up the LSDM to form the APM (Fig. 7). In practice, even it is not an essential condition, it is deserved that the parameterization (or flattening) procedure employed to obtain the APM mesh should be able to preserve the surface metric structure (length and angles etc) of the surface. In fact, quite a number of efficient flattening algorithms have been developed and interested readers may refer to references [21] and [22] for more details. The APM provides a continuous parameterization for the LSDM which is expressed in an element-by-element manner. Note that in order to provide a valid continuous mapping between the 3D and the parametric spaces, whenever the LSDM mesh is subdivided (either uniformly or adaptively), the APM will be subdivided accordingly. However, the edges of the APM will be subdivided at their *midpoints* and it is not necessary to apply the interpolating subdivision scheme on the APM. The simple midpoint rule is adopted for parametric space subdivision because (i) it will simplify a lot the computation of various geometrical properties of the limiting surface, (ii) it will lead to a faster searching algorithm during adaptive subdivision and finite element generation and (iii) it will not deteriorate the parameterization of the mapping. An example of subdivision of the LSDM and the APM is given in Fig. 8 in which one subdivision step is applied to the meshes in shown in Fig. 7.

4. Computations of surface geometrical properties

By using the APM to define the surface parameterization, various geometrical properties of the limiting surface (e.g. position vector, surface tangents and normal) can be computed by considering the mapping relationship between the LSDM and the APM. The typical problem that required to be solved during mesh generation can be stated as:

Given a point $\mathbf{u}_p=(u_p, v_p)$ in the parametric space, compute the following geometrical properties at the corresponding point, $\mathbf{x}_p$, on the limiting surface (Fig. 9):

- (1) The position vector $\mathbf{x}_p=(x_p, y_p, z_p)$.
- (2) The surface tangents, $\mathbf{r}_u, \mathbf{r}_v$, for element length scale transformation. Note that in here $\mathbf{r}$ denotes the implicit mapping between the LSDM and the APM defined by the interpolation subdivision scheme.
- (3) The second order derivatives $\mathbf{r}_{uu}, \mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$ for curvature element-size control.

Since interpolating subdivision scheme is used for the construction of geometrical model and no explicit form of the mapping function $\mathbf{r}$ is available, the above problem can only be solved approximately by using the adaptive evaluation procedure described in Section 3.3.3 of Part I of the current study. The basic operation of adaptive refinement is to first carry out a sufficient numbers of local subdivisions around the point under concern (Fig. 10) so that a small element enclosing the target point is created. The geometrical properties required will then be estimated locally. The details of the computational methods used are given below.

4.1 Computation of position vector

Consider the typical case that, after a number of adaptive refinements, the targeted point $\mathbf{u}_p=(u_p, v_p)$ is enclosed by an element A'B'C' in the parametric space while the corresponding point $\mathbf{x}_p=(x_p, y_p, z_p)$ in the 3D space is enclosed by the triangle ABC. Since the interpolating subdivision scheme is used, the points A, B and C are lying on the target limiting surface so that the triangle ABC is a local linear approximation of the limit surface around $\mathbf{x}_p$. In order to compute the position of $\mathbf{x}_p$, an auxiliary normalized ξ - η space is set up and the element in the u - v space is mapped to a right angle triangle (Fig. 11). If $\mathbf{u}_\alpha=(u_\alpha, v_\alpha)$ where $\alpha=A', B'$ and C' are the u - v coordinates of the nodes A', B' and C' respectively, then by the standard mapping procedure of a linear triangular element, $\mathbf{u}_p$ can be expressed as

$$\mathbf{u}_p = \begin{bmatrix} u_p \\ v_p \end{bmatrix} = \begin{bmatrix} u_{C'} \\ v_{C'} \end{bmatrix} + \begin{bmatrix} u_{A'C'} & u_{B'C'} \\ v_{A'C'} & v_{B'C'} \end{bmatrix} \begin{bmatrix} \xi_p \\ \eta_p \end{bmatrix} \quad (8a)$$

In Eqn. 8, $u_{\alpha\beta}=u_\alpha-u_\beta$ and $v_{\alpha\beta}=v_\alpha-v_\beta$ with $\alpha, \beta=A', B'$ and C' . (ξ_p, η_p) are the coordinates of the point $\mathbf{u}_p$ ($\mathbf{x}_p$) in the ξ - η space and can be obtained by re-arranging Eqn. 8a in the form

$$\begin{bmatrix} u_{A'C'} & u_{B'C'} \\ v_{A'C'} & v_{B'C'} \end{bmatrix} \begin{bmatrix} \xi_p \\ \eta_p \end{bmatrix} = \begin{bmatrix} u_{pC'} \\ v_{pC'} \end{bmatrix} \quad (8b)$$

Obviously, Eqn. 8b will only be singular if and only if the term $J=u_{A'C'}v_{B'C'}-u_{B'C'}v_{A'C'}=0$ which implies that the area of the triangle $A'B'C'$ is equal to zero. After (ξ_p, η_p) are found, by using the mapping relationship between the normalized and the 3D space (Fig. 11), the position vector of $\mathbf{x}_p$ can be computed as

$$\mathbf{x}_p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} + \xi_p \begin{bmatrix} x_{AC} \\ y_{AC} \\ z_{AC} \end{bmatrix} + \eta_p \begin{bmatrix} x_{BC} \\ y_{BC} \\ z_{BC} \end{bmatrix} \quad (9)$$

In Eqn. 9, $x_{\alpha\beta}=x_\alpha-x_\beta$ for $\alpha, \beta=A, B$ and C and so on. As in the above calculations, the position of $\mathbf{x}_p$ is obtained by linear interpolation, one can expect that the error of $\mathbf{x}_p$ computed will be of the order $O(h^2)$ where h is the size of the element enclosing the point in the 3D space.

4.2 Computation of first order derivatives, $\mathbf{r}_u$ and $\mathbf{r}_v$

For the computation of the first order derivatives, it is required to first generalize Eqn. 8b for a general point (ξ, η) in the ξ - η space such that

$$\begin{bmatrix} u_{A'C'} & u_{B'C'} \\ v_{A'C'} & v_{B'C'} \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} u_{uC'} \\ v_{vC'} \end{bmatrix} \quad (10a)$$

where $u_{uC'}=u-u_C$ and $v_{vC'}=v-v_C$. From Eqn. 10a, the values of ξ and η can be computed as

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \frac{1}{J} \begin{bmatrix} v_{B'C'} & -u_{B'C'} \\ -v_{A'C'} & u_{A'C'} \end{bmatrix} \begin{bmatrix} u_{uC'} \\ v_{vC'} \end{bmatrix} \quad (10b)$$

Similarly, the general form of Eqn. 9 can be written as

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} + \xi \begin{bmatrix} x_{AC} \\ y_{AC} \\ z_{AC} \end{bmatrix} + \eta \begin{bmatrix} x_{BC} \\ y_{BC} \\ z_{BC} \end{bmatrix} \quad (11a)$$

Using Eqn. 10b, one can eliminate ξ and η from Eqn. 11a and obtains

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} + \frac{1}{J} (v_{B'C'} u_{uC'} - u_{B'C'} v_{vC'}) \begin{bmatrix} x_{AC} \\ y_{AC} \\ z_{AC} \end{bmatrix} + \frac{1}{J} (v_{vC'} u_{AC'} - v_{A'C'} u_{uC'}) \begin{bmatrix} x_{BC} \\ y_{BC} \\ z_{BC} \end{bmatrix} \quad (11b)$$

Note that in Eqn. 11b, only the terms $u_{uC'}$ and $v_{vC'}$ are linear functions of u and v respectively.

Thus, the first order derivatives $\mathbf{r}_u$ and $\mathbf{r}_v$ can be computed explicitly as

$$\mathbf{r}_u = \begin{bmatrix} \partial x / \partial u \\ \partial y / \partial u \\ \partial z / \partial u \end{bmatrix} = \frac{1}{J} \begin{bmatrix} x_{AC} v_{B'C'} - x_{BC} v_{A'C'} \\ y_{AC} v_{B'C'} - y_{BC} v_{A'C'} \\ z_{AC} v_{B'C'} - z_{BC} v_{A'C'} \end{bmatrix} \quad (12a)$$

and

$$\mathbf{r}_v = \begin{bmatrix} \partial x / \partial v \\ \partial y / \partial v \\ \partial z / \partial v \end{bmatrix} = \frac{1}{J} \begin{bmatrix} -x_{AC} u_{B'C'} + x_{BC} u_{A'C'} \\ -y_{AC} u_{B'C'} + y_{BC} u_{A'C'} \\ -z_{AC} u_{B'C'} + z_{BC} u_{A'C'} \end{bmatrix} \quad (12b)$$

From Eqn. 12, one can see that $\mathbf{r}_u$ and $\mathbf{r}_v$ are constant within the element. Hence, if h is the size of the element enclosing the point in the 3D space, the errors of $\mathbf{r}_u$ and $\mathbf{r}_v$ will be of order $O(h)$ and is less accurate when comparing with the computation of $\mathbf{x}_p$. For this reason, whenever it is required to compute $\mathbf{r}_u$ and $\mathbf{r}_v$, one additional level of refinement will be carried out automatically to improve the accuracy of $\mathbf{r}_u$ and $\mathbf{r}_v$ (Fig. 12).

4.3 Computations of second order derivatives $\mathbf{r}_{uu}$, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$

As the first derivatives are constant within an element, it is not possible to directly compute the values of the second order derivatives using linear interpolation. The second order derivatives are obtained by combining the four triangles obtained in two successive subdivisions into one 6-node quadratic triangular element (Fig. 13). Note that in the 3D space, the 6-node triangular element is a true 3D surface element since its mid-edge nodes are obtained by the interpolating subdivision scheme. However, in the $u-v$ space, as the simple midpoint rule is used, the element is sub-parametric with the edge nodes lying on the midpoints of the edges. As a result, the relationship between (u,v) and (ξ,η) will again be defined by Eqn. 10. The Jacobian and the inverse of the Jacobian matrix of the transformation are given by

$$\mathbf{J} = \begin{bmatrix} \partial u / \partial \xi & \partial v / \partial \xi \\ \partial u / \partial \eta & \partial v / \partial \eta \end{bmatrix} = \begin{bmatrix} u_{1'3'} & v_{1'3'} \\ u_{2'3'} & v_{2'3'} \end{bmatrix} \quad \text{and} \quad \mathbf{J}^{-1} = \frac{1}{J} \begin{bmatrix} v_{2'3'} & -v_{1'3'} \\ -u_{2'3'} & u_{1'3'} \end{bmatrix} \quad (13)$$

For the relationship between (x,y,z) and (ξ,η) , it can be expressed in the form

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1^x \\ a_1^y \\ a_1^z \end{bmatrix} + \eta \begin{bmatrix} a_2^x \\ a_2^y \\ a_2^z \end{bmatrix} + \xi \begin{bmatrix} a_3^x \\ a_3^y \\ a_3^z \end{bmatrix} + \xi \eta \begin{bmatrix} a_4^x \\ a_4^y \\ a_4^z \end{bmatrix} + \eta^2 \begin{bmatrix} a_5^x \\ a_5^y \\ a_5^z \end{bmatrix} + \xi^2 \begin{bmatrix} a_6^x \\ a_6^y \\ a_6^z \end{bmatrix} \quad (14a)$$

where the coefficients a_i^α with $\alpha=x,y,z$ and $i=1,...,6$ are given by

$$\begin{aligned} a_1^\alpha &= \alpha_3, a_2^\alpha = 4\alpha_4 - 3\alpha_3 - \alpha_2, a_3^\alpha = 4\alpha_5 - 3\alpha_3 - \alpha_1 \\ a_4^\alpha &= 4(\alpha_3 - \alpha_4 - \alpha_5 + \alpha_6), a_5^\alpha = 2\alpha_2 + 2\alpha_3 - 4\alpha_4, a_6^\alpha = 2\alpha_1 + 2\alpha_2 - 4\alpha_5 \end{aligned} \quad (14b)$$

Hence,

$$\mathbf{r}_\xi = \begin{bmatrix} \partial x / \partial \xi \\ \partial y / \partial \xi \\ \partial z / \partial \xi \end{bmatrix} = \begin{bmatrix} a_3^x \\ a_3^y \\ a_3^z \end{bmatrix} + \begin{bmatrix} a_4^x \\ a_4^y \\ a_4^z \end{bmatrix} \eta + 2 \begin{bmatrix} a_6^x \\ a_6^y \\ a_6^z \end{bmatrix} \xi, \quad \mathbf{r}_\eta = \begin{bmatrix} \partial x / \partial \eta \\ \partial y / \partial \eta \\ \partial z / \partial \eta \end{bmatrix} = \begin{bmatrix} a_2^x \\ a_2^y \\ a_2^z \end{bmatrix} + \begin{bmatrix} a_4^x \\ a_4^y \\ a_4^z \end{bmatrix} \xi + 2 \begin{bmatrix} a_5^x \\ a_5^y \\ a_5^z \end{bmatrix} \eta \quad (15)$$

Thus, to obtain the terms $\mathbf{r}_u$ and $\mathbf{r}_v$, one should consider the transformation between the $u-v$ and the $\xi-\eta$ spaces together with Eqns. 13 and 15 such that

$$\begin{aligned} [\mathbf{r}_u, \mathbf{r}_v]^T &= \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \end{bmatrix} \\ &= \mathbf{J}^{-1} \begin{bmatrix} a_3^x + a_4^x \eta + 2a_6^x \xi & a_3^y + a_4^y \eta + 2a_6^y \xi & a_3^z + a_4^z \eta + 2a_6^z \xi \\ a_2^x + a_4^x \xi + 2a_5^x \eta & a_2^y + a_4^y \xi + 2a_5^y \eta & a_2^z + a_4^z \xi + 2a_5^z \eta \end{bmatrix} \end{aligned} \quad (16)$$

From Eqn. 16, since the matrix $\mathbf{J}^{-1}$ is a constant matrix, $\mathbf{r}_u$ and $\mathbf{r}_v$ are linear functions of ξ and η . To obtain the second derivatives $\mathbf{r}_{uu}$, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$, one can treat the terms $\partial x / \partial u$ and $\partial x / \partial v$ as functions of ξ and η and reappplies the transformation. For example,

$$\begin{aligned} \begin{bmatrix} \partial^2 x / \partial u^2 \\ \partial^2 x / \partial u \partial v \end{bmatrix} &= \begin{bmatrix} \frac{\partial(\partial x / \partial u)}{\partial u} \\ \frac{\partial(\partial x / \partial u)}{\partial v} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial^2 x}{\partial \xi^2} \\ \frac{\partial^2 x}{\partial \xi \partial \eta} \end{bmatrix} \\ &= \mathbf{J}^{-1} \left[\frac{\partial}{\partial \xi} \left(\frac{1}{J} (a_3^x v_{2'3'} - a_2^x v_{1'3'} + \eta(a_4^x v_{2'3'} - 2a_5^x v_{1'3'}) + \xi(2a_6^x v_{2'3'} - a_4^x v_{1'3'})) \right) \right. \\ &\quad \left. \frac{\partial}{\partial \eta} \left(\frac{1}{J} (a_2^x u_{1'3'} - a_3^x u_{2'3'} + \eta(2a_5^x u_{1'3'} - a_4^x u_{2'3'}) + \xi(a_4^x u_{1'3'} - 2a_6^x u_{2'3'}) \right) \right] \\ &= \frac{\mathbf{J}^{-1}}{J} \begin{bmatrix} (2a_6^x v_{2'3'} - a_4^x v_{1'3'}) \\ (2a_5^x u_{1'3'} - a_4^x u_{2'3'}) \end{bmatrix} \end{aligned} \quad (17)$$

Obviously, other terms such as $\frac{\partial^2 x}{\partial v^2}$, $\frac{\partial^2 y}{\partial u^2}$ etc. can also be computed explicitly in a similar manner. Note that in order to express the second derivatives explicitly in the above simple form, the Jacobian matrix and its determinant (Eqn. 13) must be independent of ξ and η . Otherwise, Eqn. 17 will involve differentiation of rational functions. Hence, one can see that the use of the simple midpoint rule for the subdivision of the APM greatly simplifies the second order derivatives computations.

Similar to the computations of $\mathbf{r}_u$ and $\mathbf{r}_v$, in order to maintain the accuracy of $\mathbf{r}_{uu}$, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$, addition refinements will be carried out around the target point. As shown in Fig. 14, for second order derivatives computations, *two* additional refinements will be carried out and the refined elements will be further subdivided to generate the 6-node element for calculation.

From the above computation procedures, it can be seen that while $\mathbf{x}_p$ can be obtained in a continuous form, both the surface tangents and curvatures information of the limiting surface can only be obtained in a discontinuous, element-by-element manner. However, this will not lead to any problem in the mesh generation procedure. The reasons are that (i) during the surface mesh generation step, the surface tangent and curvatures information will only be evaluated at some discrete points in the parametric space and (ii) if needed, the adaptive subdivision scheme can be applied to improve the results to any desire accuracy.

4.4 *Appropriate number of subdivisions*

During the mesh generation step, in order to ensure that the 3D positions of new nodes are computed accurately, a sufficient number of subdivisions must first be carried out before the method described in Section 4.1 is used. Towards this end, it can be shown that [7] the minimum number of subdivisions needed will depends on (i) κ , the local curvature of the surface at the evaluation point and (ii) an accuracy parameter ϵ which limit the ratio $(s-h)/h$ such that

$$\frac{(s-h)}{h} \leq \epsilon \quad (18)$$

In Eqn. 18, h is the edge length of the element enclosing the point and s is the local arc length the element approximating (Fig. 15). In order to ensure that Eqn. 18 is satisfied, it can be proved that the maximum value of h is given by [22]

$$h = \frac{(1 - \varepsilon)}{\kappa} \sqrt{40(1 - (1 - 1.2\varepsilon)^{0.5})} \quad (19)$$

Normally, the value of ε is pre-selected by the user and a value of $\varepsilon=0.01$ will be sufficient for most applications. However, in general, the curvature κ at the point is not known in advance. Thus, the appropriate number of subdivisions needed for accurate computation of the position vector can only be established by an iterative procedure shown below.

- (i) A minimum number of $NS_{\min}$ subdivisions will first be carried out around the target point. Set the current subdivision level, NL as $NL=NS_{\min}$.
- (ii) The position of the point will be computed based on the current subdivision using the method described in Section 4.1.
- (iii) Two additional subdivisions will be carried out so that the first and second order derivatives will be computed using the methods described in Sections 4.2 and 4.3.
- (iv) The derivatives obtained in (iii) will be used for the computation of the minimum principal curvatures at the target point by the methods described in reference [7].
- (v) Check whether the current element size satisfies Eqn. 19 or not. If Eqn. 19 is satisfied, exit with the current point position and derivatives.
- (vi) If $NL < NS_{\max}$, the maximum subdivision level allowed, carry out one more level of subdivision and set $NL \leftarrow NL+1$ and goto step (ii).

Note that in the above algorithm, $NS_{\min}$ is employed to ensure that a minimum number of subdivisions must be carried out while $NS_{\max}$ is employed to ensure the termination of the iterative procedure. In the current implementation, $NS_{\min}$ is set equal to 2 while a value of $NS_{\max}=6$ is used. However, for a value of $\varepsilon=0.01$, for all the examples given in Section 6, it is found that the number of subdivisions needed seldom exceeds 3.

5. Boundary and domain discretizations

Similar to the previous works of surface mesh generation [7,9], finite element meshes will be generated in a bottom up manner and the following three distinct phases of generation can be identified.

- (1) Control point formation;
- (2) Boundary segments generation; and
- (3) Sub-surface meshes generation.

The MAFT will be employed for the sub-surface mesh generation. After a sub-surface mesh is triangulated, a series of mesh quality enhancement procedures [9] will be performed to

enhance the quality of the sub-surface mesh before it is combined with the global mesh. As the MAFT has been described in detail in references [7], [9] and [23], the objective of this section is to provide some essential descriptions of those algorithms that are specially designed for the use of MAFT with the subdivision surface geometrical model.

5.1 Control point formation

Control points is a set of points with positions that are invariant with respect to the target mesh density and they consist of the following two types of nodes.

- (1) Terminal nodes which define the end points of the limiting boundary edges (in both the parametric space and the 3D space).
- (2) Boundary cusp points on the limiting boundary edges (Fig. 16).

As shown in Fig. 16, in general, control points in the 3D space are not one-to-one correspondence to the control points in the parametric space. In addition, a given control point may has different statuses with respect to different sub-surfaces.

Boundary cusp nodes are not terminal points for boundary edges. However, their positions are invariant with respect to element size density since they define sharp features along the domain boundary. As a result, their positions must first be fixed during the control point formation phase. Note that in order to generate a C^0 surface patch, the statuses of a boundary cusp point with respect to different sub-surfaces must be consistent (e.g. node 8 in Fig. 16). Otherwise, the geometrical model will contain discontinuous features.

5.2 Boundary segment generation

After all the control points are established, new boundary segments will first be generated along the all LBCs in the parametric space and then mapped to the 3D space using the subdivision surface geometrical model. In order to ensure compatibility along the interfaces of different sub-surfaces, all the LBCs in the surface patches will be classified into either *free curves* or *depended curves* according to the connectivity of the surface patch [9]. New boundary segments will first be generated on those free curves. New nodes and segments generated are then transferred to the depended curves (Fig. 17). In the current implementation, when coincident curves are present (e.g. curve 1 and curve 7 in Fig. 17), the curve with the lowest curve number will be discretized first. Note that the finial boundary discretization is independent of the order of discretization of the boundary curves. The reason is that new nodes and segments will be generated based on the geometrical definition and the metric tensors of the curves which are identical for coincident curves.

The major difference between the present boundary discretization scheme and the one used in the previous study [9] is the presence of boundary cusp nodes. All the boundary curves which contain boundary cusp nodes will be further subdivided into a number of sections during boundary discretization (Fig. 18). Different sections of the boundary curve will be separated by boundary cusp nodes and discretized separately. In this case, boundary cusp nodes will be treated in a way similar to the corner nodes with the difference that the two-point template (Section 4.1 of [13]) will be used instead of the 3-point template for position computation.

5.3 *Sub-surface mesh generation*

Once all the boundary curves are discretized, sub-surface meshes will be generated by the MAFT. The boundary segments enclosing the sub-surfaces will be retrieved as the initial generation front [9]. The MAFT will then be employed to generate an optimal anisotropic mesh in terms of size and shape in the parametric space. Finally, the mesh generated will be mapped back to the 3D space. In general, the whole parametric space mesh generation process is similar to the one described in reference [9] with the following differences:

(1) *Description of geometrical model and background mesh*

In the present triangulation scheme, totally three finite element meshes will be involved during the sub-surface mesh generation process. They are, namely, (i) the limiting surface definition mesh (LSDM) which defines the geometrical properties of the target surface, (ii) the background mesh (BGM) which defines the metric specification for element shape and size controls and (iii) the new mesh generated. In general, the LSDM and the BGM are two different meshes but in many cases, it is possible to use one single mesh to define both the geometrical model and the metric specification. In the current implementation, three different sets of data with *identical* format are employed for the storage of these meshes. Even though such a practice will increase the amount of work space required, it will significantly increase the flexibility of the mesh generator. By storing the LSDM and the BGM separately, the definition of them can be changed in any refinement step. Towards this end, two different modeling strategies can be identified.

(i) *Meshing with stationary geometrical model*

When a stationary geometrical model is used, the LSDM will remain unchanged during successive refinement steps in which the element size distribution is altered (Fig. 19). In this case, the BGM will be simply replaced by the new mesh generated. This approach is

frequently used in the cases when the geometry of the problem domain is unchanged while the mesh density is required to be updated due to various reasons [1].

(ii) Meshing with non-stationary geometrical model

In many applications such as large deformation shell analysis or crack growth modelling, the geometry of the surface may be changed in every incremental analysis step. In such a case, it is required to update both the geometrical model and the BGM. If conventional modelling tools such as analytical surface patches are used, the updating process will be rather difficult and tedious. However, if the geometrical model is represented by the subdivision model, the updating process can be easily accomplished by replacing the LSDM and the BGM with the newly generated mesh (Fig. 20). In particular, if both the LSDM and the BGM are stored in identical format, all the state variables can be transferred directly from the new mesh generated.

(2) *Treatment of interior cusp points*

As interior cusp points are sharp features on the sub-surface, their positions will remain constant with respect to the element density of the mesh. Hence, it is required that nodes will always be generated exactly at the positions of interior cusp points. In the current implementation, instead of designing a special algorithm to generate nodes exactly at interior cusp points, during the parametric mesh generation step, nodes will first be generated freely without any constraints imposed on their positions. After the parametric mesh generation is completely, cusp nodes will be re-established by a recovery procedure consists of the following steps:

- (i) For any interior cusp point S on the surface, the element $A'B'C'$ which enclosing it in the parametric space will be identified (Fig. 21).
- (ii) The local parent coordinates of the interior cusp point with respect to the element $A'B'C'$, (ξ_s, η_s) , are computed.
- (iii) According to the values of (ξ_s, η_s) , the cusp node will be recovered by either (i) creating an addition cusp node inside the element $A'B'C'$ (and also the element ABC in the 3D space) and subdivided the element into two or three elements or (ii) by moving one of the existing element corner nodes to the position of the cusp point. Details of different recovery schemes used are listed in Table I and Fig. 21.

Table I Recovery of interior cusp nodes

Range of (ξ_s, η_s) .	Actions	Figures
$\xi_s \geq 1/5$ and $\eta_s \geq 1/5$ and $\zeta_s \geq 1/5$	subdivide A'B'C' into 3 elements	Fig. 21a
$\xi_s < 1/5$ and $\eta_s \geq 1/5$ and $\zeta_s \geq 1/5$	Subdivide A'B'C' along edge B'C'	Fig. 21b
$\xi_s \geq 1/5$ and $\eta_s < 1/5$ and $\zeta_s \geq 1/5$	Subdivide A'B'C' along edge A'C'	Fig. 21b
$\xi_s \geq 1/5$ and $\eta_s \geq 1/5$ and $\zeta_s < 1/5$	Subdivide A'B'C' along edge A'B'	Fig. 21b
$\eta_s < 1/5$ and $\zeta_s < 1/5$	move node A' to cusp point	Fig. 21c
$\xi_s < 1/5$ and $\zeta_s < 1/5$	move node B' to cusp point	Fig. 21c
$\xi_s < 1/5$ and $\eta_s < 1/5$	move node C' to cusp point	Fig. 21c

Note: $\zeta_s = 1 - \xi_s - \eta_s$

(3) Curvature element size control

Since the element size characteristics of the present generation procedure is controlled by using the metric specifications and the local curvatures of the surface can be estimated accurately by using the method described in Section 4.3, the fidelity of the mesh generated can be ensured by first intersect the pseudo-curvature element-size controlling metric, $\mathbf{M}_{pc3D}$ (Eqn. 7), with the user defined metric $\mathbf{M}_{3D}$ (Eqn 3). Eqn. 6 is then applied on the resulting metric to combine the effect of the parametric mapping to generate the final element size controlling surface metric $\mathbf{M}_{sur}$. By this approach, the element-size controlling information can be prescribed implicitly at the nodal points of the BGM as the inputs of the mesh generator.

6. Mesh generation examples

In this section seven examples will be presented to demonstrate the performance of the surface mesh generation scheme developed. The examples given in this section can be classified into two groups. In the first group, the geometrical definitions of the target surfaces are stationary and are defined by the coarse LSDMs. Four adaptive refinements will be carried out to generate meshes with element size distribution compatible with the input metric specifications. After each adaptive refinement is finished, the newly generated mesh will be employed as the BGM for the next refinement. In the second group, the geometrical definitions of the target surfaces are non-stationary. The whole generation process will be divided into a number of *increments*. The initial geometrical model is defined by the input LSDM. During each increment, four cycles of adaptive refinements will again be carried out to obtain an adaptive mesh fulfilling the metric specification. The final adaptive mesh produced will then be slightly deformed to generate the new LSDM for the next increment. By repeating such deformation increment for a number of times, adaptive meshes with shapes very different from the initial geometrical model are obtained.

In all the examples given in this section, in order to ensure that the positions of new nodes created will be placed accurately on the target surface, a value of $\epsilon=0.01$ (Eqn. 18) was used for the computation of 3D coordinates of the new nodes. In addition, the values of $NS_{\min}$ and $NS_{\max}$ (Section 4.4) were set equal to 2 and 6 respectively.

6.1 Examples with stationary geometrical definitions

Example 1: Mesh generation for a hemisphere

In the first example, the target surface is a hemisphere with no sharp feature. The geometry of the target surface is defined by a simple mesh shown in Fig. 22a. In this example, the element size and shape required was specified in terms of the parameter ϵ for the curvature element-size control. Three values of ϵ corresponding to $\epsilon=0.005$, 0.001 and 0.0001 were applied to generate the corresponding pseudo-curvature element-size controlling metric, $\mathbf{M}_{pc3D}$ (Eqn. 7). The final meshes obtained after four adaptive refinements are shown in Figs. 22b to 22d for different values of ϵ . From these figures one can see that by using a coarse mesh to define the underlying geometrical model, the limiting surface is not a perfect hemisphere with constant curvature. However, one can see that the uneven curvature effect is quite small and can be detected only by setting a very low value of $\epsilon=0.0001$.

Examples 2-4: Mesh with shape features

In these three examples, the target surfaces contain sharp and discontinuous features. A model contains an interior cusp node is shown in Fig. 23. For this model, two different sets of meshes were generated. In the first set of meshes, node A shown in Fig. 23a was treated as a normal interior node while in the second set of meshes, it was tagged as an interior cusp node. The final meshes generated were shown in Figs. 23b and 23c for the untagged and tagged models respectively. From Fig. 23, it can be seen that the shape of the mesh is changed locally after node A is tagged and the cusp point was reproduced exactly by the mesh generator.

Fig. 24 shows the LSDM for a model contains crease lines which writes out the three letters "CoM". Two different sets of meshes were generated by tagging and not tagging the letters as crease lines. The final meshes obtained are shown in Figs. 24b and 24d respectively. Zoom views near the letter 'C' of the untagged and the tagged model are shown in Fig. 24c and 24e respectively. From these figures, one can see that the crease lines with discontinuity in gradient were produced in the tagged model. Regarding the mesh quality and edge length distributions of the meshes generated, the convergence histories for the element quality α [9], and the normalized edge length distribution for the meshes generated with crease lines are

plotted in Figs. 25 and 26 respectively. From these figures, it can be concluded that both the quality and the edge length distribution of the meshes generated are good with most of the elements and edges with shape quality and normalized edge length close to the ideal value of unity. Furthermore, the convergence rate of the mesh generation process is fast such that the meshes are converged to their final forms after one or two iterations.

Fig. 27 shows a model with discontinuous feature. The LSDM is shown in Fig. 27a in which the nodes A, B and C were separated to create an internal crack lines. Along one side of the cracked line all the nodes are tagged as normal boundary nodes while node B on the other side of the cracked line was tagged as a boundary cusp node. The final mesh produced and a zoom view near the discontinuous feature produced are shown in Figs. 27b and 27c respectively. This example demonstrates that discontinuous features can be obtained by using the tagging technique appropriately.

Example 5: Model without boundary

In this example, the LSDM used is the one corresponding to Example 4 in Part I of the study. The LSDM mesh used is shown in Fig. 7a and contains no boundary segment. However, in order to create the parameterization model, cut lines were created and the model was opened up in the $u-v$ space. The LSDM in the $u-v$ space is shown in Fig. 7b and the final mesh with small elements generated at the eight vertices of the cube is shown in Fig. 28. Note that in this example, even though the domain in the 3D space is an object without boundary, good parameterization can be obtained by creating cut lines and no singular point is present in the problem domain. If traditional surface modelling tools such as NURBS surface patches are employed to parameterize the 3D surface, one must either create singular point in the mapping or divide the target surface into a number of sub-surfaces. Thus, in this case, the use of subdivision surfaces will be more convenient and lead to a simpler representation of the surface.

6.2 Examples with non-stationary geometrical definitions

Example 6: Model with two sub-surfaces

In the first example of non-stationary geometrical model, the initial LSDM for the target surface consists of two cylindrical surfaces as shown in Fig. 29a. After four iterations were carried out to generate the final adaptive mesh for the initial surface model (Fig. 29b), a new geometrical model was set up by deforming the mesh in such a way that nodes near the end of the horizontal cylindrical surface were moved downward to the surface of a fictitious sphere centred at point A. The radius of the fictitious sphere is set equal to $1.5 \times N_{inc}$ where N_{inc} is the

number of increments carried out. After the new geometrical model is set up, the metric specification is re-defined such that small elements will be generated at where the fictitious sphere intersects with the horizontal cylindrical surface. Totally five increments were carried out and the final meshes obtained from increment one to five are shown in Fig. 29c to 29g respectively. From these figures, it can be seen that small elements are generated near the intersection between the horizontal cylindrical surface and the fictitious sphere. Furthermore, the shape of the final mesh obtained (Fig. 29g) is very different from the initial geometry (Fig. 29b).

Example 7: Model of a bridge deck

In the last example, the surface model under consideration is a section of a bridge deck. The initial LSDM mesh used which consists of six sub-surfaces is shown in Fig. 30a. The first final mesh generated is shown in 30b. For this problem, deformed geometrical models are obtained by applying incremental rotations at the end A of the bridge deck. During each increment, end A was rotated for 30° while end B remained horizontal. Thus, after twelve increments, end A will be horizontal again while all other parts of the bridge deck will undergo large rotations. For this example, the metric specification for the mesh density was unchanged during each increment. The final mesh obtained after 1,3,5,6,7,9 and 12 increments are shown in Fig. 30c to 30i respectively. In practice, such a large rotational deformation will seldom be encountered and this example is given to demonstrate that the proposed mesh generation algorithm is able to generate finite element meshes even for such applications.

Regarding the speed of the present mesh generation scheme, since it is developed based on the MAFT, it is expected that the order of complexity is similar to the previous scheme developed base in the NURBS surface model and is proportional to $O(NE^{3/2})$ (NE is the number of element generated). This prediction is confirmed by plotting a graph of $NE^{3/2}$ against the CPU time used as shown in Fig. 31. In general, as the geometrical model is not expressed in an explicit form, more CPU time is needed to evaluate the 3D positions of new nodes created. However, since an adaptive evaluation procedure is used, it was found that the CPU time needed for position calculations only accounts for a small part of the total CPU time used for the whole mesh generation process. In the current implementation, by using a low-end PC equipped with an Intel PII 450MHz CPU and 64Mb of memory, it requires 52 seconds of CPU time to generate a mesh with 6000 elements (the BGM of the model contains 6000 elements). Hence, the algorithm should be fast enough for most practical applications.

7. Conclusions and future extensions

In this paper, a new metric advancing front mesh generator has been proposed for the generation of anisotropic mesh for 3D surfaces. The new mesh generation scheme suggested is based on a new geometrical model using the subdivision surface concept in which the geometry of the target surface is represented by a finite element mesh. By using the subdivision concept and a simple tagging procedure, various sharp and discontinuous features such as cusp points, crease lines and cracked lines can be included in the geometrical model without the need of using complicated geometrical primitives. Furthermore, with a careful implementation, the data employed to define the geometrical model of the target surface can be stored in a format identical with the formats of the background mesh and the newly generated mesh. As a result, in the present model, it becomes a trivial task to change the geometrical definition of the target surface without the need of re-parameterization. Hence, the proposed mesh generation algorithm will be useful in applications in which the surface geometry is constantly changing. The mesh generation examples demonstrated that high quality finite element meshes with element size distribution compatible with the user specifications can be generated by using a low-end PC in a short time. Furthermore, with the use of the curvature element-size controlling technique, meshes which are good approximations of the target surfaces can be generated up to any user defined accuracy.

A potential area of extension for the present study is to develop a mesh generation algorithm using the quadrilateral subdivision surfaces as the underlying geometrical model. In such a case, the area in which major work done is needed is the development of a robust anisotropic quadrilateral surface mesh generator for the discretization of target surfaces created by the quadrilateral subdivision surface scheme.

References

1. C. K. Lee and R. E. Hobbs, Automatic adaptive refinement for shell analysis using nine-node assumed strain element, *Int. J. Numer. Methods Engrg.*, 40, 3601-3638 (1997)
2. C. K. Lee, K. Y. Sze and S. H. Lo, On using degenerated solid shell elements in adaptive refinement analysis, *Int. J. Numer. Methods Engrg.*, 45 627-659 (1999)
3. J. C. Cavendish, W. H. Frey and S. P. Marin, Feature-based design and finite element mesh generation for functional surfaces, *Adv. Engrg. Software*, 13, 226-237 (1991)
4. T. J. R. Hughes, C. A. Taylor and G. Engel, Finite element simulation of pulsatile flow through distensible blood vessels, Abstract in the Proceeding of the Fourth World Congress on Computational Mechanics, Buenos Aires, Argentina, 26 June - 2 July 1998 Vol. II page 1006 (CD-ROM proceeding)
5. H. Borouchaki, P. L. George, F. Hecht, P. Laug and E. Saltel, Delaunay mesh generation governed by metric specifications. Part I. Algorithms, *Fin. Ele. Anal. Des.*, 25, 61-83 (1997)
6. H. Borouchaki, P. L. George, F. Hecht, P. Laug and E. Saltel, Delaunay mesh generation governed by metric specifications. Part II. Applications, *Fin. Ele. Anal. Des.*, 25, 85-109 (1997)
7. C. K. Lee, On curvature element-size control in metric surface mesh generation, *Int. J. Numer. Methods. Engrg.*, 50, 787-807 (2001)
8. H. Borouchaki, R. Hecht and P. J. Frey, Mesh gradation control, *Int. J. Numer. Methods. Engrg.*, 43, 1143-1165 (1998)
9. C. K. Lee, Automatic metric advancing front triangulation over curved surfaces, *Engrg. Computations*, 17, 48-74 (2000)
10. H. Borouchaki, P. Laug and P. L. George, Parametric surface meshing using a combined advancing-front generalized Delaunay approach, *Int. J. Numer. Methods. Engrg.*, 49, 233-259 (2000)
11. S. J. Owen, M. L. Staten, S. A. Canann and S. Saigal, Q-Morph: An indirect approach to advancing front quad meshing, *Int. J. Numer. Methods. Engrg.*, 44, 1317-1340 (1999)
12. Y. K. Lee and C. K. Lee, Automatic Generation of Anisotropic Quadrilateral Meshes on Three Dimensional Surfaces using Metric Specifications, *Int. J. Numer. Methods Engrg.*, **53**, 2673-2700 (2002)
13. C. K. Lee, Automatic metric 3D surface mesh generation using subdivision surfaces geometrical model. Part I: Construction of underlying geometrical model. To appear.
14. R. Lohner, Regridding surface triangulations, *J. Comput. Physics*, 126, 1-10, (1996)
15. L. Kobbelt, Iterative mesh generation for FE-computations on free form surfaces, *Engineering, Comput.*, 14, 806-820 (1997)

16. A. Rassineux, P. Villon, J. M. Savignat and O. Stab, Surface meshing by local hermite diffuse interpolation, *Int. J. Numer. Methods Engrg.*, 49, 31-49 (2000)
17. N. Dyn, D. Levin and J. A. Gregory, A butterfly subdivision scheme for surface interpolation with tension control, *Proceedings of Computer Graphic, ACM SIGGRAPH'90*, 160-169 (1990)
18. D. Zorin, P. Schroder and W. Sweldens, Interpolating subdivision with arbitrary topology, *Proceedings of Computer Graphic, ACM SIGGRAPH'96*, 189-192 (1996)
19. N. Dyn, D. Levin and J. A. Gregory, A 4point interpolatory subdivision scheme for curve design, *Comput. Aided Design*, 4, 257-268 (1987)
20. J. McCartney, B. K. Hinds and B. L. Seow, The flattening of triangulated surfaces incorporating darts and gussets, *Comp. Aided,Design.*, 31, 249-260 (1999)
21. A. Sheffer and E. de Sturler, Surface parameterization for meshing by triangulation flattening, *Proceedings of the 9th International Meshing Roundtable, Sandia National Laboratories, U. S. A* 161-172 October (2000)
22. R. J. Cass and S. E. Benzley, R. J. Meyers and T. D. Blacker, Generalized 3-D paving: An automated quadrilateral surface mesh generation algorithm, *Int. J. Numer. Methods Engrg.*, **39**, 1475-1489 (1996)
23. C. K. Lee, Automatic adaptive mesh generation using metric advancing front approach, *Engr g. Comput.*, 16, 230-263 (1999)

Fig. 1 Definition of a single target surface: (a) limiting surface definition mesh, limiting boundary curve definition edge and edge termination node; (b) limiting surface and limiting boundary curve.

Fig. 2 Topology definition of the surface patches by LBCDEs: (a) surface patch with no branching or crack lines; (b) surface patch with branching and crack line.

Fig. 3 Case that an ETN is a smooth boundary point: (a) limiting surface definition mesh; (b) limiting surface.

Fig. 4 Parameterization of the LSDM by using an APM: (a) LSDM in 3D space; (b) APM in u-v space.

Fig. 5 Local node numbering of elements in LSDM and APM.

Fig. 6 Construction of APM by projection: (a) LSDM; (b) APM.

Fig. 7. Construction of APM by cuttings and opening operations: (a) LSDM; (b) APM.

Fig. 8 Subdivision of LSDM and APM: (a) LSDM after one subdivision;
(b) APM after one subdivision.

Fig. 9. Evaluation of geometrical properties at a point on the limiting surface.

Fig. 10 Adaptive subdivision around a point.

Fig. 11 Computation of $\mathbf{x}_p$.

Fig. 12 Computations of $\mathbf{r}_u$ and $\mathbf{r}_v$ by carrying out one more level of subdivision.

Fig. 13 Computations of $\mathbf{r}_{uu}$, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$ using a six-node element formed by two subdivision steps.

Fig. 14 Computations of $\mathbf{r}_{uu}$, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$ by carrying out two more levels of subdivisions.

Fig. 15 Curvature element-size deviation measurement.

Nodes in 3D space	Statuses of nodes*	Corresponding nodes in u-v spaces
1	I-4, II-4	2,5
2	II-4, III-4	6,8
3	III-4, IV-2	9,12,13
4	III-4, IV-5	10,15,14
5	I-4, II-4, III-4	7,3,11
6	I-4	4
7	I-4	1
8	II-3, III-3	16,17

*Legend for statuses of nodes:
 2=boundary node, 3=boundary cusp node,
 4=corner node, 5=corner cusp node
 I-4 = the node has a status of 4 on surface I
 and so on

Fig. 16 Control points for different sub-surfaces.

Sub-surfaces	Boundary curves ordering	Boundary segments generation sequence
I	1→2→3→4	1→2→3→4
II	5→6→7	5→6→7 (transfer from 1)
III	10→8→12→11	10→8→12→(11 transfer from 6)
IV	14→9→13→15	14→(9 transfer from 8)→13→(15 transfer from 8)

Fig. 17 Discretization of domain boundary.

Existing boundary discretization: ▲ corner nodes ● normal boundary node ■ boundary cusp node

New boundary discretization: ▲ corner nodes ● normal boundary node ■ boundary cusp node

Fig.18 Discretization of boundary curve contains cusp node: (a) existing boundary curve; (b) discretization section by section separated by boundary cusp points.

Fig. 19 Meshing with stationary geometrical model: (a) geometrical model unchanged during each mesh refinement; (b) meshes generated with different element density.

Fig. 20 Meshing with non-stationary geometrical model.

Fig. 21 Recovery of interior cusp points: (a) case for subdividing element ABC into three elements; (b) cases for subdividing element ABC into two elements along the edge; (c) cases for moving the positions of corner nodes.

(a)

(c)

(b)

(d)

Fig. 22 Example 1, model for a hemisphere: (a) LSDM; (b) final mesh generated by setting $\epsilon=0.005$; (c) final mesh generated by setting $\epsilon=0.001$; (d) final mesh generated by setting $\epsilon=0.0001$.

(a)

(b)

(d)

Fig. 23 Example 2, model with interior node: (a) the LSDM; (b) final mesh generated by not tagging interior node A; (d) final mesh generated by tagging interior node A as cusp node.

(a)

(b)

(c)

Fig. 24 Example 3, model with crease lines: (a) LSDM; (b) final mesh without crease lines; (c) zoom view near the letter "C" for final mesh without crease lines (d) final mesh with crease lines; (e) zoom view near the letter "C" for final mesh with crease lines.

(d)

(e)

Fig. 24 (Cont'd) Example 3, model with crease lines: (a) LSDM; (b) final mesh without crease lines; (c) zoom view near the letter "C" for final mesh without crease lines (d) final mesh with crease lines; (e) zoom view near the letter "C" for final mesh with crease lines.

Fig. 25 Convergence of element quality (α) for meshes generated for Example 3 with crease lines.

Fig. 26 Convergence of edge length for meshes generated for Example 3 with crease lines.

(a)

(b)

(c)

Fig. 27 Model with discontinuity: (a) LSDM; (b) final mesh after 4 refinements; (c) zoom view near the discontinuity.

Fig. 28 Example 5, surface model without boundary.

Fig. 29 Example 6, model of intersection of two surfaces: (a) initial LSDM; (b) final mesh obtained from initial LSDM; (c) to (g) final meshes obtained after one to five increment steps.

(d)

(f)

(e)

(g)

Fig. 29 (Cont'd) Example 6: model of intersection of two surfaces: (a) initial LSDM; (b) final mesh obtained from the initial LSDM; (c) to (g) final meshes obtained after one to five increment steps.

Fig. 30 Example 7, model of a bridge deck: (a) initial LSDM; (b) final mesh obtained from the initial LSDM; (c) to (i) final meshes obtained after rotating the end B for 30° , 90° , 150° , 180° , 210° , 270° and 360° respectively.

(d)

(e)

(f)

Fig. 30 (Cont'd) Example 7: model of a bridge deck: (a) initial LSDM; (b) final mesh obtained from the initial LSDM; (c) to (i) final meshes obtained after rotating the end B for 30° , 90° , 150° , 180° , 210° , 270° and 360° respectively.

(g)

(h)

(i)

Fig. 30 (Cont'd) Example 7: model of a bridge deck: (a) initial LSDM; (b) final mesh obtained from the initial LSDM; (c) to (i) final meshes obtained after rotating the end B for 30° , 90° , 150° , 180° , 210° , 270° and 360° respectively.

Fig. 31 A graph of $NE^{3/2}$ against the CPU time needed for mesh generation