INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **46**:961–982
Published online 28 September 2004 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.791

Non-self-overlapping structured grid generation on an *n*-sided surface

Charlie C. L. Wang[1,‡] and Kai Tang[2,*,†]

[1]*Department of Automation and Computer-Aided Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong*
[2]*Department of Mechanical Engineering, Hong Kong University of Science and Technology, Clear Water Bay, N.T., Hong Kong*

SUMMARY

Most existing meshing algorithms for a 2D or shell figure requires the figure to have exactly four sides. Generating structured grids in the *n*-sided parametric region of a trimmed surface thus usually requires to first partition the region into four-sided sub-regions. We address the automatic structured grid generation problem in an *n*-sided region by fitting a planar Gregory patch so that the partition requirement is naturally avoided. However, self-overlapping may occur in some portions of the algebraically generated grid; this severely limits its usage in most of engineering and scientific applications where a grid system with no self-intersecting is strictly required. To solve the problem, we use a functional optimization approach to move grid nodes in the $u-v$ domain of the trimmed surface to eliminate the self-overlapping. The derivatives of a Gregory patch, which are extremely difficult to compute analytically, are not required in our method. Thus, our optimization algorithm compares favourably at least in terms of speed with some other mesh optimization algorithms, such as the elliptic PDE method. In addition, to overcome the difficulty of guessing a good initial position of every grid node for the conjugate gradient method, a progressive optimization algorithm is incorporated in our optimization. Experiment results are given to illustrate the usefulness and effectiveness of the presented method. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: self-overlapping; structured grid; quadrilateral grid; Gregory patch; *n*-sided; trimmed surface

1. INTRODUCTION

In computer-aided engineering, geometric modelling, computer graphics, and many other applications, trimmed parametric surfaces are widely adopted [1]. After a parametric surface S intersects with other surfaces, only a portion of the surface patch is used in defining a meaningful shape, which is called a trimmed (parametric) surface S_T. S_T is constrained by the same mathematical surface equation as $S(u,v)$, but its parametric domain is only a portion of

*Correspondence to: Kai Tang, Department of Mechanical Engineering, Hong Kong University of Science and Technology, Clear Water Bay, N.T., Hong Kong.
†E-mail: mektang@ust.hk
‡E-mail: cwang@acae.cuhk.edu.hk

Received 8 May 2003
Revised 19 June 2004

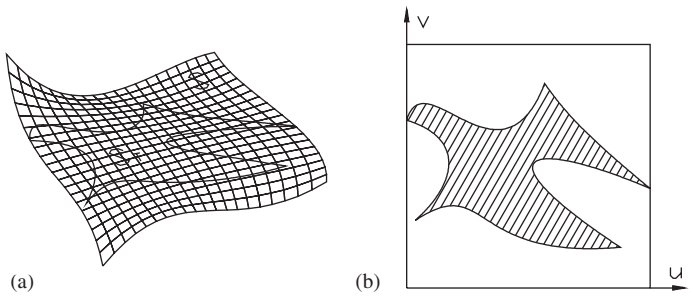

Figure 1. A trimmed n-sided surface and its parametric area: (a) an example surface; (b) the parametric area in $u-v$ domain.

that of S. The parametric area of S_T, to be denoted as P_{S_T}, lies inside $(u,v) \in [0,1] \times [0,1]$ (assuming the $u-v$ domain of S is normalized) and is bounded by a number of curves (see Figure 1). Each boundary curve of P_{S_T} is expressed as a parametric equation of the form $b_i = [u_i(t)\ v_i(t)]$, where $t \in [0,1]$. Formally, a trimmed n-sided surface patch is defined below.

Definition 1.1
A trimmed parametric surface S_T, whose n boundary curves form a Jordan curve in its parametric area P_{S_T}, is defined as a *trimmed n-sided* (*parametric*) *surface patch*.

An example of a trimmed n-sided patch and its parametric area is given in Figure 1. The task of approximating a trimmed surface by a complex of simple planar elements (either triangular or quadrilateral) plays an important role in engineering computing; this is referred to as the surface *meshing* operation, which has been studied for many years [2–16]. The two most powerful analysis tools in engineering are the finite element methods and the finite difference methods. The finite element method usually adopts either triangular grids or quadrilateral grids, and the grids can be structured or unstructured. However, the finite difference method generally uses *structured quadrilateral grids* (or simply called *structured grids*). The structured grids can be generated algebraically or as the solution of *partial deferential equations* (PDEs). Surface grid generation algorithms are based on a projection strategy, consisting in generating a grid on the surface parametric plane, and then to project it on the surface. However during the projection step the grid characteristics, such as point distribution and orthogonality, are often not correctly transported. In this approach, we solve the structured grid generation problem in an n-sided region while preserving the non-overlapping property in a numerical optimization manner.

Within all the surface grid generation approaches, algebraic grid generation is some form of interpolation from boundary points—different approaches use different kinds of interpolation [3–5]. Overlapping may, however, happen on some portions of algebraically generated grids, which must be corrected in order for the mesh to be usable for almost any application. Also, error may be generated when converting generic boundary curves into curves with a specified representation. Grid generation is actually a boundary-value problem, so grids can be generated from point distribution on boundaries by solving elliptic PDEs in the field [6–10]. The smoothness properties and extremum principles of some such PDE systems can serve to produce smooth grids without boundary overlapping. The most recent work of Arina [11] overcame the difficulties of point distribution and orthogonality on the mapped a surface by a

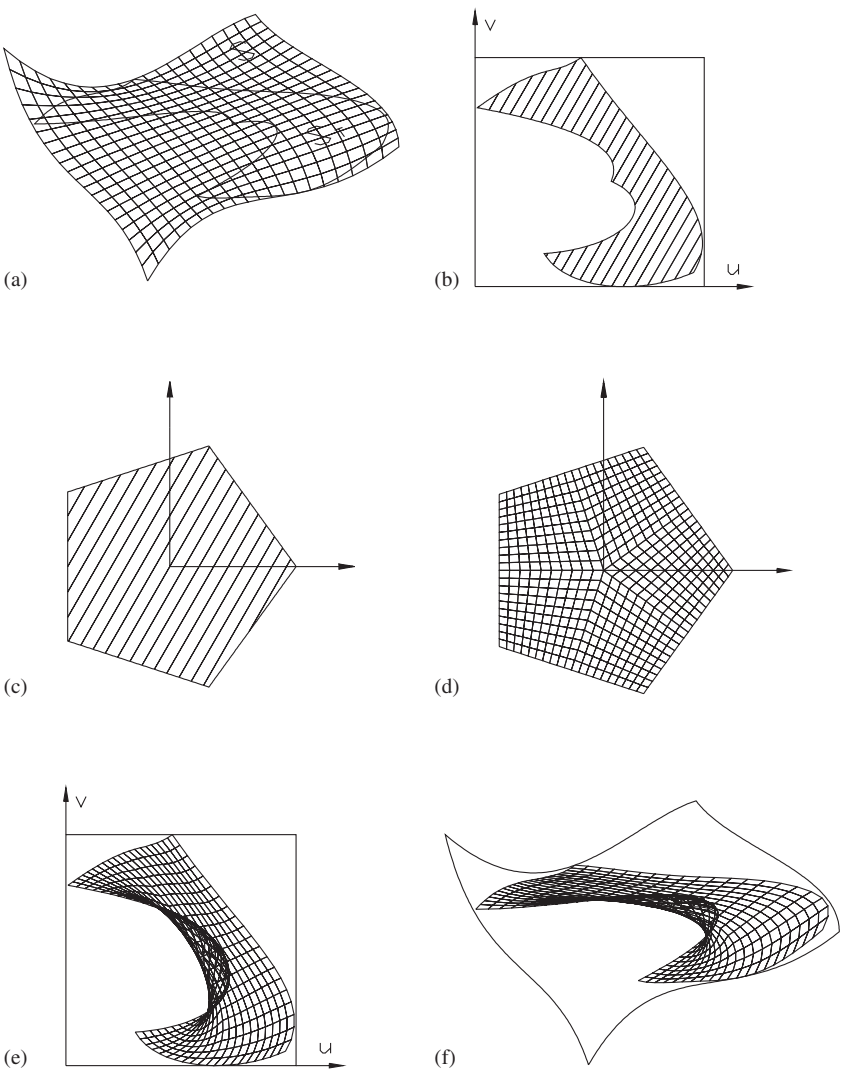

Figure 2. Mapping from the $x-y-z$ space to the $\xi-\eta$ domain via the $u-v$ domain (Example I): (a) a trimmed n-sided surface $S_T(u,v)$ in the $x-y-z$ space; (b) the parametric area of $S_T(u,v)$ in the $u-v$ domain; (c) the parametric area of $G(\xi,\eta)$ in the $\xi-\eta$ domain; (d) the grid in the $\xi-\eta$ domain; (e) the grid in the $u-v$ domain (with overlapping); (f) the grid for the surface S_T (with overlapping).

conformal map, preserving angles and scale-length ratios. His method consists of two major steps: the parameterization of surface by isothermal co-ordinates; the generation of a 2D grid on the conformal parametric plane and its projection on the surface. All the above approaches, however, cannot be directly applied to a region with n sides (when $n>4$), so the n-sided region has to be partitioned into several non-overlapping blocks with four sides. For the region with a complex shape (as shown in Figures 2(a) and 2(b)), generating such partitions is not a

straightforward work—skeleton structure [12, 13] or background triangulation [14, 15] may be required to construct. However, their resultant grids are usually unstructured.

In this paper, a novel method is presented for generating structured quadrilateral grids with no self-overlapping on a trimmed n-sided patch by automatically fitting non-self-overlapping multi-block planar grids into the parametric region of the given surface. The idea is to first construct an initial structured algebraic grid based on the Gregory patch mapping [16] and then eliminate any possible self-overlapping on the mesh by performing a functional optimization. In our approach, a planar Gregory patch $G(\xi,\eta)$ is adopted to fit an algebraic grid in the $u-v$ domain of S_T; and since the $\xi-\eta$ domain P_G of $G(\xi,\eta)$ is a regular n-sided polygon, it is easy to divide P_G into n four-sided sub-regions and grid each sub-region automatically (see Figures 2(c) and 2(d)). The mapping between the $\xi-\eta$ domain and the $u-v$ domain by a Gregory patch nevertheless may generate overlapping (see Figures 2(e) and 2(f)). To mend that, we develop a functional optimization method to eliminate such overlapping in the $u-v$ domain; the shapes of grid elements in the $u-v$ space are also adjusted in the optimization. One question arising naturally is: why not using PDEs for this optimization? Our answer is that, when enhancing the quality of grids by the PDE methods, the derivatives such as G_ξ, G_η, $G_{\xi\xi}$, $G_{\eta\eta}$, and $G_{\xi\eta}$ are needed; for a Gregory patch (detail in Section 3), it is hard to give the analytical formulas for these terms, and computing them numerically is a very time-consuming process. Our method compares favourably to PDEs in this regard, as only the positions of every grid nodes in the $u-v$ domain are required. Thus, the speed of our method is faster.

The paper is organized as follows. In Section 3, after giving some necessary definitions and preliminaries for the Gregory patch mapping, we introduce the method to determine the initial algebraic grid by a planar Gregory patch. We then formulate the self-overlapping problem in the $u-v$ domain as a singularity problem in the derivatives on the Gregory patch, in Section 4.1. Based on this formulation, the corresponding objective function of our optimization is then derived in Section 4.2, with a shape control term also added to the objective function. The details of the numerical implementation of the optimization are given in Section 4.3. As our numerical implementation is iterative, to overcome the difficulty of 'guessing' a good initial grid, the idea of progressive optimization is introduced in Section 4.4. Finally, some experimental results are shown in Section 5 to demonstrate the power of our approach, with some conclusion remarks offered in the last section.

2. GREGORY PATCH MAPPING

The necessary definitions and preliminaries of a Gregory patch are first given here.

Definition 2.1
Let $P(u): 0\leqslant u\leqslant 1$ and $Q(v): 0\leqslant v\leqslant 1$ be two regular curves in $\Re^3$ with $P(0)=Q(0)$, and $T_P(u)$: $0\leqslant u\leqslant 1$ and $T_Q(v)$: $0\leqslant v\leqslant 1$ be two C^1 vector functions in $\Re^3$ satisfying

$$T_P(0)=\left.\frac{\mathrm{d}Q(v)}{\mathrm{d}v}\right|_{v=0} \quad \text{and} \quad T_Q(0)=\left.\frac{\mathrm{d}P(u)}{\mathrm{d}u}\right|_{u=0}$$

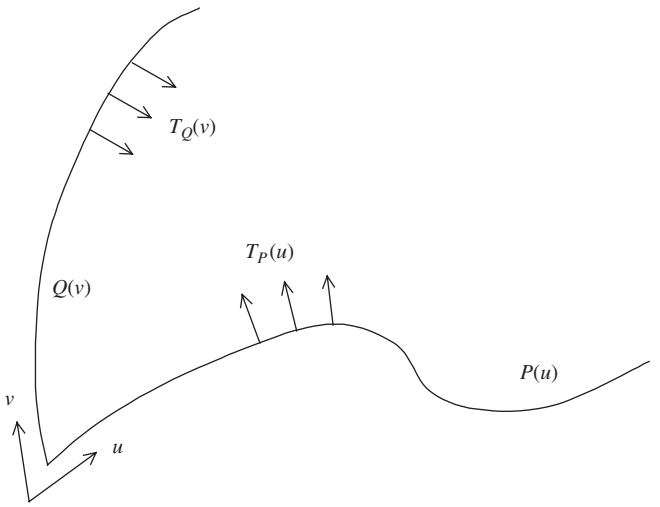

Figure 3. Define a Gregory corner interpolator.

the *Gregory corner interpolator* of the four, $\{P(u), Q(v), T_P(u), T_Q(v)\}$, is a surface in $\Re^3$ (Figure 3) defined by

$$r(u,v) = P(u) + vT_P(u) + Q(v) + uT_Q(v) - P(0) - vT_P(0) - uT_Q(0) - uv\frac{vT'_P(0) + uT'_Q(0)}{u+v} \tag{1}$$

The Gregory corner interpolator function $r(u,v)$ agrees with $P(u)$ and $Q(v)$ along the two sides (i.e. $r(u,0) = P(u)$ and $r(0,v) = Q(v)$). Also, its partial derivatives with respect to u and v agree with $T_P(u)$ and $T_Q(v)$ along the respective sides

$$-\left.\frac{\partial r(u,v)}{\partial v}\right|_{v=0} = T_P(u) \quad \text{and} \quad \left.\frac{\partial r(u,v)}{\partial u}\right|_{u=0} = T_Q(v)$$

since $T_P(0) = Q'(0)$ and $T_Q(0) = P'(0)$. For an n-sided 3D surface, n such interpolator functions can be defined on the n corners; the final surface is the weighted sum of the n functions [17, 18]. The details are defined as follows.

Definition 2.2
The parametric domain of a Gregory patch with n sides is defined as a unit length regular n-gon in the $\xi - \eta$ domain.

We name the parametric domain of a Gregory patch G as P_G, where all corners X_k $(k = 0, 1, \ldots, n-1)$ are ordered in the anti-clockwise (as shown in Figure 4). Given a point $X = (\xi_0, \eta_0)$ inside P_G, when computing its 3D position defined by a Gregory corner interpolator $r_k(u_k, v_k)$, the parameters (u_k, v_k) of the point corresponding to the kth corner X_k

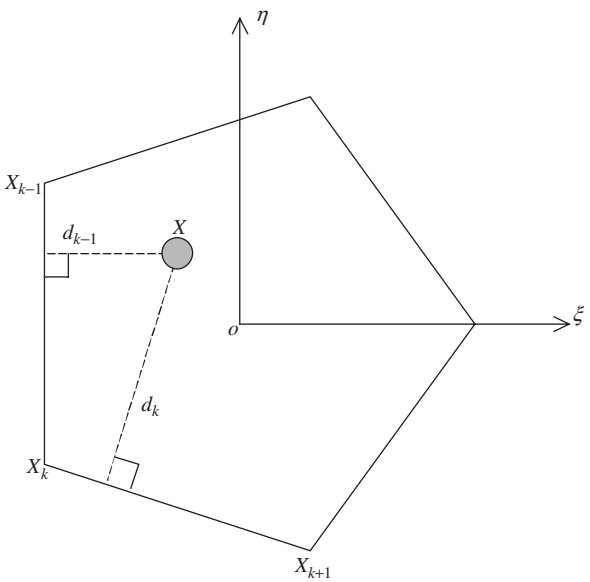

Figure 4. P_G of a Gregory patch with five sides.

are defined as

$$(u_k, v_k) = \left(\frac{d_{k-1}}{d_{k-1} + d_{k+1}}, \frac{d_k}{d_{k-2} + d_k} \right) \tag{2}$$

where d_k represents the perpendicular distance from X to the side $X_k X_{k+1}$. It is easy to find that if (ξ_0, η_0) lies on the side $X_k X_{k+1}$, $v_k = 0$ since $d_k = 0$; if (ξ_0, η_0) is on $X_{k-1} X_k$, $u_k = 0$ since $d_{k-1} = 0$; when (ξ_0, η_0) and X_{k+1} coincides, we have $u_k = 1$ by Equation (2); and when (ξ_0, η_0) and X_{k-1} coincides, we have $v_k = 1$.

Definition 2.3
If $C_0(u)$, $C_1(u), \ldots, C_{n-1}(u)$ are n regular 3D curves that form a closed loop in 3D space, that is $C_k(1) = C_{(k+1) \bmod n}(0)$ $(k = 0, 1, \ldots, n-1)$, and $T_{C_0}(u)$, $T_{C_1}(u), \ldots, T_{C_{n-1}}(u)$ are n continuous 3D vector functions defined on the $C_k(u)$s, respectively, the *Gregory patch* of $C_k(u)$s and $T_{C_k}(u)$s is defined as a mapping from P_G to $\Re^3$ (Figure 5)

$$G(X) = \sum_{k=0}^{n-1} w_k(X) r_k(u_k(X), v_k(X)) \tag{3}$$

where

$$w_k(X) = \frac{\prod_{j \neq k-1, k} d_j^2}{\sum_{l=0}^{n-1} \prod_{j \neq l-1, l} d_j^2}$$

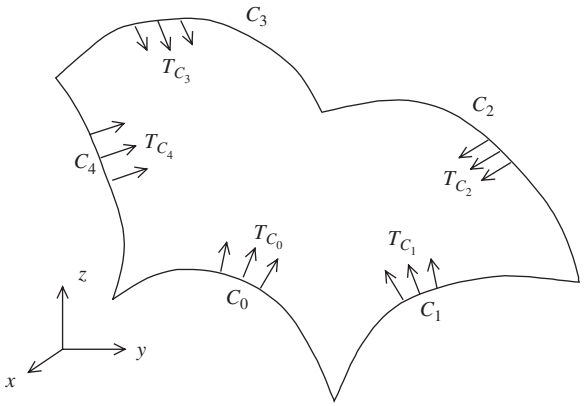

Figure 5. Define a Gregory patch.

and $r_k(u_k, v_k)$ represents the Gregory corner interpolator function for the kth corner of the four items $\{C_k(u),\ \bar{C}_k(v),\ T_{C_k}(u),\ \bar{T}_{C_k}(v)\}$, $\bar{C}_k(u) = C_k(1-u)$, and $\bar{T}_{C_k}(u) = T_{C_k}(1-u)$.

Note that $w_k(X)$ is unity at the vertex X_k and is zero on those edges of the n-gon not incident with X_k. Thus, it can be verified that the Gregory patch $G(X)$ is boundary conforming. That is, if an X is on the boundary of P_G, then $G(X)$ must be on one of the $C_k(u)$s, and conversely, for any point $p \in \Re^3$ on any $C_k(u)$, there must be an X on the boundary of P_G such that $p = G(X)$. If the n boundary curves of a Gregory patch $G(X)$ all lie in a common plane, obviously $G(X)$ also lies in that plane, i.e. it is a *planar Gregory patch*. Next, we elaborate on how the Gregory patch mapping can be utilized to mesh an n-side trimmed parametric surface patch.

3. ALGEBRAIC GRID GENERATION BASED ON GREGORY MAPPING

Similar to other algebraic grid generation methods, the algebraic grid construction in our approach also consists of three steps: (1) forward mapping; (2) grid generation; and (3) inverse mapping. The forward mapping is the mapping of the 3D physical surface S_T to its underlying parametric area $\mathrm{P}_{S_\mathrm{T}}$. By fitting a planar Gregory patch G into $\mathrm{P}_{S_\mathrm{T}}$, $\mathrm{P}_{S_\mathrm{T}}$ of the surface S_T is further mapped into the parametric n-gon P_G of G. In this planar Gregory patch G, all the $C_k(u)$s and $T_{C_k}(u)$s are determined by the 2D boundary curves of $\mathrm{P}_{S_\mathrm{T}}$ in the parametric domain instead of the 3D boundary curves of S_T. Grids will be generated in P_G and then mapped back into $\Re^3$, generating a structured grid system of S_T.

In the $\xi - \eta$ domain of G, the co-ordinates of parametric n-gon P_G's corners are defined by

$$X_k = \left(\cos\frac{2k\pi}{n}, \sin\frac{2k\pi}{n}\right) \tag{4}$$

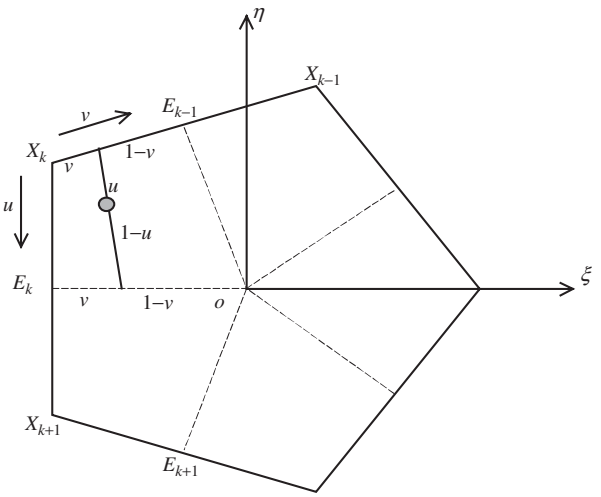

Figure 6. Multi-block grids generation in the $\xi - \eta$ domain in P_G.

where n is the number of corners, and $k = 0, \ldots, n-1$; so the position of middle points on the n boundary sides can be computed by

$$E_k = \left(\frac{1}{2}\left(\cos\frac{2k\pi}{n} + \cos\frac{2(k+1)\pi}{n} \right), \frac{1}{2}\left(\sin\frac{2k\pi}{n} + \sin\frac{2(k+1)\pi}{n} \right) \right) \tag{5}$$

By the X_ks, E_ks, and the origin o, the n-gon P_G is divided into n blocks. When establishing the $M \times N$ grid in the kth sub-domain of P_G, the co-ordinate of every grid node $(\xi^k_{i,j}, \eta^k_{i,j})$ is determined by (as illustrated in Figure 6)

$$\begin{aligned} \xi^k_{i,j} &= \left(1 - \frac{i}{M}\right)\left[\left(1 - \frac{j}{N}\right)\xi(X_K) + \frac{j}{N}\xi(E_{K-1})\right] + \frac{i}{M}\left[\left(1 - \frac{j}{N}\right)\xi(E_K) + \frac{j}{N}\xi(o)\right] \\ \eta^k_{i,j} &= \left(1 - \frac{i}{M}\right)\left[\left(1 - \frac{j}{N}\right)\eta(X_K) + \frac{j}{N}\eta(E_{K-1})\right] + \frac{i}{M}\left[\left(1 - \frac{j}{N}\right)\eta(E_K) + \frac{j}{N}\eta(o)\right] \end{aligned} \tag{6}$$

where $\xi(\cdots)$ and $\eta(\cdots)$ represent the ξ and η co-ordinates of a point in the $\xi - \eta$ plane. In our approach, the n sub-domains are meshed with the same number of M and N, so the boundary nodes of adjacent sub-domains are coincident. To benefit the later grid optimization algorithm, we store the topological structure of the final grid M^G by a pair of complex (V^G, K^G), where V^G is the set of grid nodes (the coincident vertices can be stored *only once* in V^G), and K^G is a simplex complex specifying the connectivity of the grid simplices (the *vertex–face* and *vertex–vertex* adjacency information). Therefore, every node lying on the boundary of P_G has two incident faces, and the node coinciding with o has n incident faces, while all other inner nodes have four incident faces.

After the grid M^G is constructed in P_G, the co-ordinates of grid nodes are mapped backwards into $\Re^3$ by the algebraic equation of G and S_T. However, when fitting a planar Gregory patch G into the parametric domain P_{S_T} of S_T in the $u - v$ plane, only the boundary curves $-C_k(u)$s are given by P_{S_T}. To determine the algebraic equation of G (Equation (3)), the $T_{C_k}(u)$s are

also needed. Here, a linear blending function is chosen. By the compatibility conditions at corners given in Definition 2.1, we have

$$T_{C_i}(0) = \left.\frac{\mathrm{d}\bar{C}_{i-1}(u)}{\mathrm{d}u}\right|_{u=0} = -\left.\frac{\mathrm{d}C_{i-1}(u)}{\mathrm{d}u}\right|_{u=1} \quad \text{and} \quad T_{C_i}(1) = \left.\frac{\mathrm{d}C_{i+1}(u)}{\mathrm{d}u}\right|_{u=0}$$

the function of $T_{C_i}(u)$ is then given as

$$T_{C_i}(u) = (1-u)T_{C_i}(0) + uT_{C_i}(1) \tag{7}$$

Now, the final co-ordinates of grid nodes on the physical trimmed space can be expressed in the following form:

$$S_{T_m} = S(u(G(\xi_m, \eta_m)), v(G(\xi_m, \eta_m)))$$

where the functions $u(\cdots)$ and $v(\cdots)$ represent the u and v co-ordinates of a point on the Gregory patch.

As already exemplified in Figure 2, however, when the boundary of $\mathrm{P}_{S_\mathrm{T}}$ is complicated and convoluted, self-overlapping may occur in the $u-v$ plane by the planar Gregory patch mapping, which leads to a self-intersecting grid on S_T. In the following section, we introduce an iterative functional minimization method for the purpose of eliminating or reducing the self-overlapping in the $u-v$ domain.

4. GRID OPTIMIZATION

Once an initial structured mesh M^G is generated on $\mathrm{P}_{S_\mathrm{T}}$ by means of Gregory patch mapping, we next proceed to eliminate any possible self-overlapping in the mesh. This is achieved by first modelling this elimination process as a functional minimization problem and then introducing the formulas to optimize the derived objective function. To avoid 'guessing' a good initial value in the iterative optimization process, a progressive optimization algorithm is also proposed.

4.1. Non-self-overlapping property

First, it is necessary to give a formal mathematical characterization of self-overlapping. Let us add a virtual axis w perpendicular to the $u-v$ plane as $w = u \times v$. A Gregory patch $G(\xi,\eta)$ now is a planar region embedded in the $u \times v \times w$ space, denoted as $G(\xi,\eta) = [U(\xi,\eta)\ V(\xi,\eta)\ W(\xi,\eta)]^\mathrm{T}$, where $U(\xi,\eta)$, $V(\xi,\eta)$, and $W(\xi,\eta)$ are the components of $G(\xi,\eta)$ on the u, v, and w axis, respectively (note that $W(\xi,\eta)$ is a constant zero). We define the unit '*normal vector*' at any point (ξ_0,η_0) on the patch $G(\xi,\eta)$ as

$$N(\xi_0,\eta_0) = \left.\frac{G_\xi \times G_\eta}{\|G_\xi \times G_\eta\|}\right|_{(\xi_0,\eta_0)}$$

A point (ξ_0,η_0) is said to be *singular* if its corresponding $\|G_\xi \times G_\eta\|$ is a zero vector. The lemma below is important as it stipulates the condition for guaranteeing the non-self-overlapping property.

Lemma 4.1
The mapping $G(\xi,\eta)=[U(\xi,\eta)\ V(\xi,\eta)\ W(\xi,\eta)]^{\mathrm{T}}$ has no self-overlapping if and only if there is no any singular point in the n-gon P_G.

Proof
Let us argument the $G(\xi,\eta)$ by $G^*(\xi,\eta)=[U^*(\xi,\eta)\ V^*(\xi,\eta)\ W^*(\xi,\eta)]^{\mathrm{T}}$ with $U^*(\xi,\eta)=U(\xi,\eta)$, $V^*(\xi,\eta)=V(\xi,\eta)$, and $W^*(\xi,\eta)=a\xi+b\eta$ for some real number a and b. By properly choosing a and b, one can enforce $G^*(\xi,\eta)$ to have non-zero length normal vector everywhere, thus it is a smooth regular surface in the $u\times v\times w$ space. Suppose first that $G(\xi,\eta)$ is self-overlapped. This means that there exist two distinct pairs $(\xi_0,\eta_0)\in\mathrm{P}_G$ and $(\xi_1,\eta_1)\in\mathrm{P}_G$, such that $(U^*(\xi_0,\eta_0),V^*(\xi_0,\eta_0))=(U^*(\xi_1,\eta_1),V^*(\xi_1,\eta_1))$. By properly selecting a and b, one can also ensure that $W^*(\xi_0,\eta_0)\neq W^*(\xi_1,\eta_1)$. Let us intersect $G^*(\xi,\eta)$ with a plane Π that is parallel to the $u-w$ plane and contains the two points

$$p_0=(U^*(\xi_0,\eta_0),V^*(\xi_0,\eta_0),W^*(\xi_0,\eta_0)) \text{ and } p_1=(U^*(\xi_1,\eta_1),V^*(\xi_1,\eta_1),W^*(\xi_1,\eta_1))$$

resulting in a regular curve σ. Consider the portion $\sigma*$ of σ between the two points, since $\sigma*$ is regular and bounded, it must have a u-extreme point $p=(U^*(\xi^*,\eta^*),V^*(\xi^*,\eta^*),W^*(\xi^*,\eta^*))$ where the normal vector n to the curve is parallel to the u-axis, as shown in Figure 7(a). Since the projection of the normal N to the surface $X^*(\xi,\eta)$ at point p in plane Π can be easily seen to identify with n, we have $N\cdot w=0$. This translates to

$$U^*_\xi(\xi^*,\eta^*)V^*_\eta(\xi^*,\eta^*)=U^*_\eta(\xi^*,\eta^*)V^*_\xi(\xi^*,\eta^*)$$

i.e. $U_\xi(\xi^*,\eta^*)V_\eta(\xi^*,\eta^*)=U_\eta(\xi^*,\eta^*)V_\xi(\xi^*,\eta^*)$. This means (ξ^*,η^*) is a singular point of $G(\xi,\eta)$.

Conversely, let (ξ^*,η^*) be a singular point of $G(\xi,\eta)$; hence, $U_\xi(\xi^*,\eta^*)V_\eta(\xi^*,\eta^*)=U_\eta(\xi^*,\eta^*)V_\xi(\xi^*,\eta^*)$. Consequently, the normal N to the surface $G^*(\xi,\eta)$ at (ξ^*,η^*) is perpendicular to the w-axis. Without loss of generality, we can assume N is parallel to the u-axis. Intersecting $G^*(\xi,\eta)$ with the plane $v=V^*(\xi^*,\eta^*)$, we obtain a regular curve σ. As $p^*=(U^*(\xi^*,\eta^*),V^*(\xi^*,\eta^*),W^*(\xi^*,\eta^*))$ is a local u-extreme point on this curve, one can find a real number $\delta>0$ such that the vertical line $u=U^*(\xi^*,\eta^*)-\delta$ intersects σ at least twice

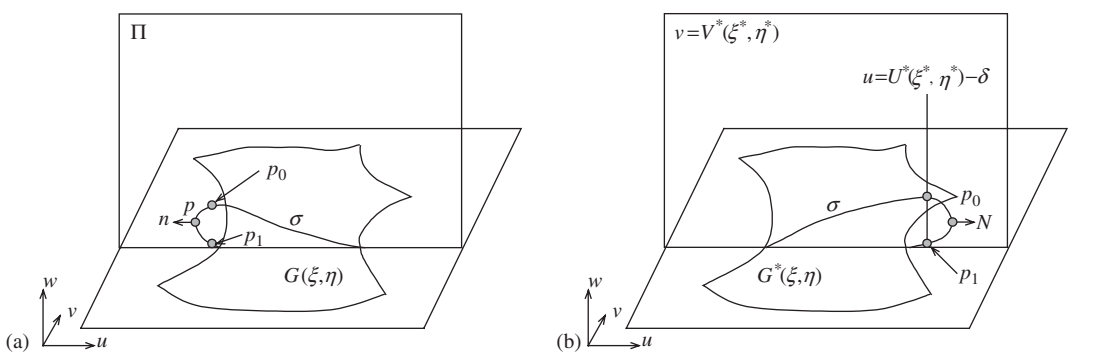

Figure 7. Proof of Lemma 4.1.

(assuming p^* is a u-maximum point). Let

$$p_0 = (U^*(\xi_0, \eta_0), V^*(\xi_0, \eta_0), W^*(\xi_0, \eta_0)) \quad \text{and} \quad p_1 = (U^*(\xi_1, \eta_1), V^*(\xi_1, \eta_1), W^*(\xi_1, \eta_1))$$

be two such intersection points for some $(\xi_0, \eta_0) \neq (\xi_1, \eta_1)$, as shown in Figure 7(b). Obviously, we have $(U^*(\xi_0, \eta_0), V^*(\xi_0, \eta_0)) = (U^*(\xi_1, \eta_1), V^*(\xi_1, \eta_1))$. Since $(U(\xi, \eta), V(\xi, \eta)) = U^*(\xi, \eta), V^*(\xi, \eta)$, we conclude that $G(\xi, \eta)$ maps two distinct points in the $\xi - \eta$ domain to a same point in the region P_G. This completes the proof. □

Based on the above lemma, the following useful proposition is in order.

Proposition 4.1
If the normal at every point to the Gregory patch $G(\xi, \eta)$ has the same sign in w, then $G(\xi, \eta)$ has no self-overlapping.

Without loss of generality, we can assume that the sign in w of the normal vectors of a non-self-overlapping mapping $G(\xi, \eta)$ is always positive.

4.2. *Objective function of the optimization*

To facilitate the discussion of the objective function, we use the following terms for self-overlapping.

Definition 4.1
A point (ξ_d, η_d) is called a *shadow point* of $G(\xi, \eta)$ if the normal at $G(\xi_d, \eta_d)$ is along the negative w-axis.

Definition 4.2
The set R_d of all the shadow points of $G(\xi, \eta)$ is defined as the *shadow region* of $G(\xi, \eta)$.

From the above analysis, we find that the non-self-overlapping term in the objective function should be a function indicating the area of the shadow region of $G(\xi, \eta)$. During the algebraic grid generation, the four nodes in every facet are sorted in the counter-clockwise order as (u_{j-1}, v_{j-1}), (u_j, v_j), (u_{j+1}, v_{j+1}), and (u_{j+2}, v_{j+2}). Thus, the area of facet i in M^G can be computed by

$$\begin{aligned} A_i = \tfrac{1}{2}[&(u_j - u_{j+1})(v_j + v_{j+1}) + (u_{j+1} - u_{j+2})(v_{j+1} + v_{j+2}) \\ &+ (u_{j+2} - u_{j-1})(v_{j+2} + v_{j-1}) + (u_{j-1} - u_j)(v_{j-1} + v_j)] \end{aligned} \tag{8}$$

When A_i is negative, all the points in facet i are shadow points. Rather than simply adding all the negative A_is and taking the sum as the minimization objective function, we adopt an exponential function defined as

$$J_1 = \sum_{i=0}^{F-1} \mathrm{e}^{-aA_i} \tag{9}$$

where A_i is the area of facet i in the mesh, F is the total number of facets in the mesh, and $a = 1/\max\{|A_i|\}$. The figure of the function $f(t) = \mathrm{e}^{-at}$ is shown below with $a = 2$ (see

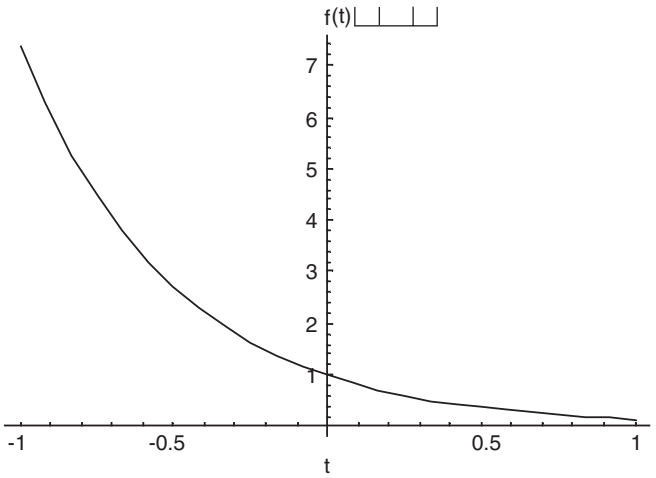

Figure 8. Shape of function $f(t)=\mathrm{e}^{-at}$ with $a=2$.

Figure 8). The selection of this function over simple summation is motivated by considerations: (1) this function usually converges better than the simple linear summation in the iterative minimization process, and (2) it also tends to compute a global minimization. Why not simply applying a spring model (i.e. $f(t)=\frac{1}{2}at^2$) as the format of our objective function? It is because that we want the first term J_1 effecting on the grids only when having some negative A_is. Thus, the ideal format should have $f(t)\approx 0$ when $t>0$. Also, the function $f(t)$ is expected to be C^1 continuous since our objective function will be minimized by a gradient method. The exponential function satisfies these factors but a spring model does not.

To achieve a smooth grid with good facet shape, Laplacian smoothing is usually applied on structured grids, which arises from solving a pair of partial differential equations [19]: $u_{\xi\xi}+u_{\eta\eta}=0$ and $v_{\xi\xi}+v_{\eta\eta}=0$. In practice, node positions are the average of points of its neighbouring nodes in Laplacian smoothing. Similarly, we add the following smoothing term into our objective function:

$$J_2=\sum_{j=0}^{E-1}\left\|\bar{v}_j-\frac{1}{L}\sum_{m\in j^*}\bar{v}_m\right\|^2 \tag{10}$$

where j^* represents an index complex of the *neighbour nodes* of the inner node $\bar{v}_j=(u_j,v_j)$ in M^G, L is the element number of j^*, and E is the number of inner nodes in the mesh. The formula of J_2 can be rewritten as

$$J_2=\sum_{j=0}^{E-1}\left[\left(u_j-\frac{1}{L}\sum_{m\in j^*}u_m\right)^2+\left(v_j-\frac{1}{L}\sum_{m\in j^*}v_m\right)^2\right] \tag{11}$$

By summing together the non-self-overlapping term and the smoothing term, the final objective function is given as

$$J = \sum_{i=0}^{F-1} \mathrm{e}^{-aA_i} + \frac{1}{\bar{J}_2} \sum_{j=0}^{E-1} \left[\left(u_j - \frac{1}{L} \sum_{m \in j^*} u_m \right)^2 + \left(v_j - \frac{1}{L} \sum_{m \in j^*} v_m \right)^2 \right] \tag{12}$$

where $\bar{J}_2$ is the initial value of J_2 before the optimization. This factor is utilized to balance the relative importance of the non-self-overlapping term and the smoothing term so that J_1 leads the nodes' movement when self-overlapping occurs, and J_2 directs the nodes to archive better grid shape when $J_1 \to 0$.

4.3. Numerical implementation

During the process of optimization, we move the inner nodes of M^G in the $u - v$ plane to achieve the functional optimum (minimum). Therefore, the u and v components of all the inner nodes of M^G form the solution vector χ. The conjugate gradient method [20] is applied to obtain the functional optimum, where the explicit form of gradient at every inner node to the objective function is desired. For an inner node (u_j, v_j), from Equation (8), we have

$$\frac{\partial A_i}{\partial u_j} = \frac{1}{2}(v_{j+1} - v_{j-1}) \quad \text{and} \quad \frac{\partial A_i}{\partial v_j} = \frac{1}{2}(u_{j-1} - u_{j+1})$$

so the gradient direction of (u_j, v_j) with respect to the non-self-overlapping term J_1 in the objective function is

$$\begin{bmatrix} \dfrac{\partial J_1}{\partial u_j} \\ \dfrac{\partial J_1}{\partial v_j} \end{bmatrix} = \begin{bmatrix} \displaystyle\sum_{k=0}^{N-1} (-a\mathrm{e}^{-aA_k}) \frac{\partial A_k}{\partial u_j} \\ \displaystyle\sum_{k=0}^{N-1} (-a\mathrm{e}^{-aA_k}) \frac{\partial A_k}{\partial v_j} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{2} \displaystyle\sum_{k=0}^{N-1} -a\mathrm{e}^{-aA_k}(v_{j+1} - v_{j-1}) \\ \dfrac{1}{2} \displaystyle\sum_{k=0}^{N-1} -a\mathrm{e}^{-aA_k}(u_{j-1} - u_{j+1}) \end{bmatrix} \tag{13}$$

where A_k is the area of the kth incident facet of (u_j, v_j), and N is the number of incident facets around (u_j, v_j). By Equation (11), the gradient of (u_j, v_j) with respect to the smoothing term J_2 in the objective function is determined as

$$\begin{bmatrix} \dfrac{\partial J_2}{\partial u_j} \\ \dfrac{\partial J_2}{\partial v_j} \end{bmatrix} = \begin{bmatrix} 2\left(u_j - \dfrac{1}{L} \displaystyle\sum_{m \in j^*} u_m \right) - \displaystyle\sum_{m \in j^*} \frac{2}{L_m} \left(u_m - \frac{1}{L_m} \sum_{n \in m^*} u_n \right) \\ 2\left(v_j - \dfrac{1}{L} \displaystyle\sum_{m \in j^*} v_m \right) - \displaystyle\sum_{m \in j^*} \frac{2}{L_m} \left(v_m - \frac{1}{L_m} \sum_{n \in m^*} v_n \right) \end{bmatrix} \tag{14}$$

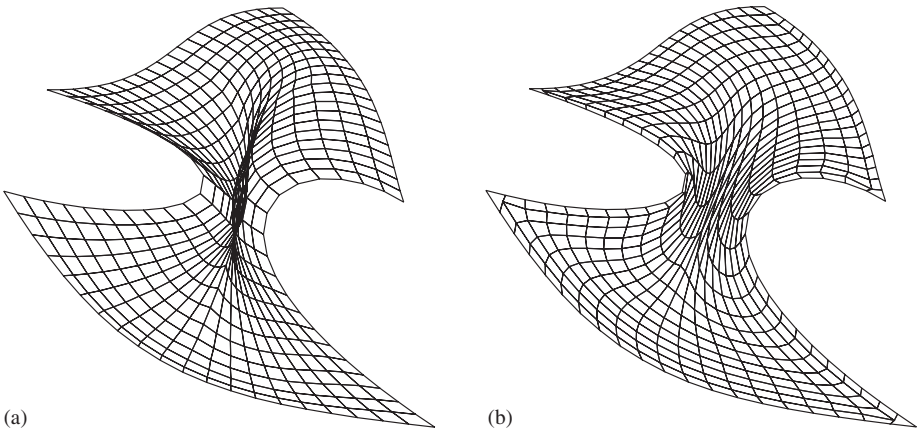

Figure 9. Example II—grid before vs after optimization: (a) before optimization; (b) after optimization.

where m^* represents an index complex of the *neighbouring nodes* of (u_j, v_j), and L_m is the element number of m^*. Therefore, in summary, we have

$$\begin{bmatrix} \dfrac{\partial J}{\partial u_j} \\ \dfrac{\partial J}{\partial v_j} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{2}\displaystyle\sum_{k=0}^{N-1} -a\mathrm{e}^{-aA_k}(v_{j+1} - v_{j-1}) \\ \quad + \dfrac{2}{\bar{J}_2}\left[\left(u_j - \dfrac{1}{L}\displaystyle\sum_{m\in j^*} u_m\right) - \displaystyle\sum_{m\in j^*}\dfrac{1}{L_m}\left(u_m - \dfrac{1}{L_m}\displaystyle\sum_{n\in m^*} u_n\right)\right] \\ \dfrac{1}{2}\displaystyle\sum_{k=0}^{N-1} -a\mathrm{e}^{-aA_k}(u_{j-1} - u_{j+1}) \\ \quad + \dfrac{2}{\bar{J}_2}\left[\left(v_j - \dfrac{1}{L}\displaystyle\sum_{m\in j^*} v_m\right) - \displaystyle\sum_{m\in j^*}\dfrac{1}{L_m}\left(v_m - \dfrac{1}{L_m}\displaystyle\sum_{n\in m^*} v_n\right)\right] \end{bmatrix} \tag{15}$$

Using the above formula, by a conjugate gradient method, we can iteratively determine the value of components in χ that makes the objective function as defined in Equation (12) minimum. Thus, the u and v parameters of every inner node in a non-self-overlapping grid M^{G^*} are determined. By the parametric equation of S_T, the final position of all grid nodes can be computed by their u, v parameters.

Figure 9 shows an example of the grid (in $u - v$ plane) generated before and after the functional optimization when choosing a 10×10 grid in each sub-domain. As demonstrated in the figure, the self-overlapping is eliminated in the optimized grid.

4.4. Progressive optimization

As our objective function in the functional optimization is concave with respect to its solution vector χ (there can be many local minima), the success of the numerical optimization algorithm depends critically on the initial position of χ. We take the algebraic grid generated

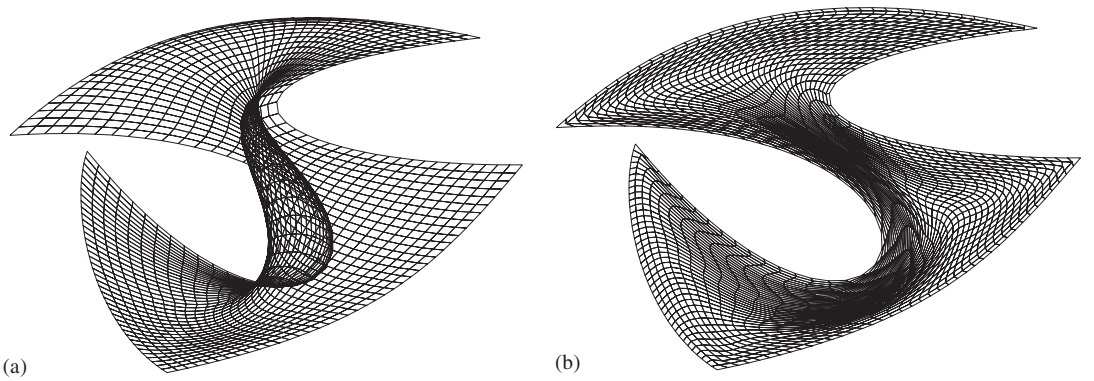

Figure 10. Example III—result of progressive optimization: (a) before optimization; (b) after optimization.

by a Gregory patch in Section 3 as an initial position of the solution vector; it however may not be a very satisfying one for some strongly concaved boundaries (e.g. the one shown in Figure 10(a)). 'Guessing' a good initial vector is hard. The basic idea we take to overcome this difficulty is to progressively achieve the optimum by gradually deforming the grid M^0 from a regular n-gon Ω_R into the grid M^f in the region $\mathrm{P}_{S_\mathrm{T}}$ bounded by the given curves in the $u-v$ plane, which we discuss in detail in this section.

The grid M^0 generated in the initial shape, a regular n-gon, by a Gregory patch $G^0(\xi,\eta)$ is non-self-overlapping. For any grid node $\bar{v}_j$, its co-ordinates determined by the Gregory patch $G^\mathrm{T}(\xi,\eta)$ on $\mathrm{P}_{S_\mathrm{T}}$ in the $u-v$ plane are $(u_j^\mathrm{T}, v_j^\mathrm{T})$, and the co-ordinates determined by $G^0(\xi,\eta)$ are (u_j^0, v_j^0). At the beginning of the deformation, the moving direction of $\bar{v}_j$ is $(\mathrm{d}u_j, \mathrm{d}v_j) = (u_j^\mathrm{T} - u_j^0, v_j^\mathrm{T} - v_j^0)$. When deforming M^0 into M^f by changing a deformation factor λ_t ($\lambda_t \in [0,1]$), the position of every grid node $\bar{v}_j$ is determined by

$$\begin{aligned} u_j^t &= u_j^0 + \lambda_t \,\mathrm{d}u_j \\ v_j^t &= v_j^0 + \lambda_t \,\mathrm{d}v_j \end{aligned} \tag{16}$$

During the deformation, we check if self-overlapping occurs; once it is detected, the numerical optimization method presented in Section 4.3 is applied to current grid M^{λ_t} to construct a new grid $M^{\lambda_t *}$ without or with less self-overlapping. The position of grid node $\bar{v}_j$ in $M^{\lambda_t *}$ changes from (u_j^t, v_j^t) to $(u_j^{t^*}, v_j^{t^*})$, so we use the following equation to alternate the moving direction of $\bar{v}_j$.

$$\begin{aligned} \mathrm{d}u_j &= (u_j^{t^*} - u_j^0)/\lambda_t \\ \mathrm{d}v_j &= (v_j^{t^*} - v_j^0)/\lambda_t \end{aligned} \tag{17}$$

The deformation will then continue along the new directions guided by Equation (16). The deformation and functional optimization are applied alternatively until the final non-self-overlapping grid M^f is obtained. Since the positions of boundary grid nodes are not adjusted

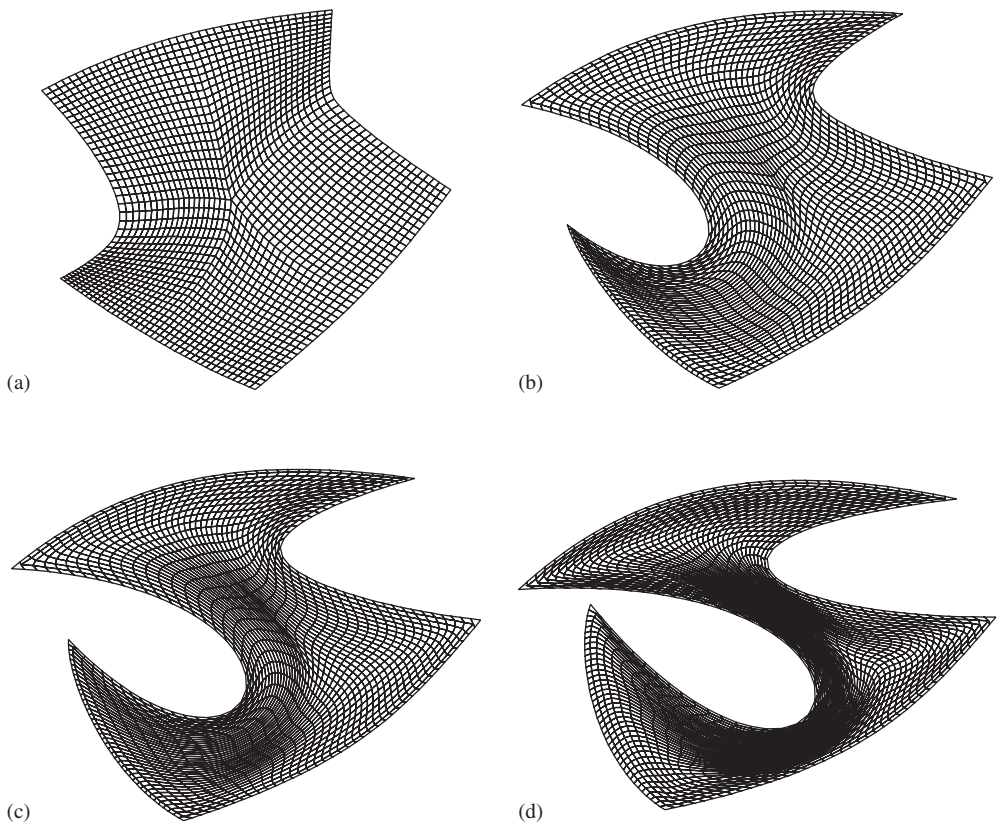

Figure 11. Example III—progressive results: (a) $\lambda_t = 0.2$; (b) $\lambda_t = 0.413$; (c) $\lambda_t = 0.709$; (d) $\lambda_t = 1.0$.

during the optimization, their moving directions do not change during the deformation—so the result grid M^f of the progressive optimization is still boundary conforming to $\mathrm{P}_{S_\mathrm{T}}$ exactly as the algebraic grid M^T generated by the Gregory patch $G^\mathrm{T}(\xi, \eta)$ on $\mathrm{P}_{S_\mathrm{T}}$.

During the deformation, the deformation factor λ_t increases from zero to one adaptively to the value increase of the objective function. The overall procedure of progressive optimization is outlined in pseudo-code in ***Algorithm*** ProgressiveOptimization( ) below. As a demonstration of the power of this progressive optimization, to Example III (shown in Figure 10), we first tried the pure numerical optimization which though failed to achieve a non-self-overlapping grid even after iterating 10 000 times; using ***Algorithm*** ProgressiveOptimization( ), we easily obtain a final result without self-overlapping by applying the numerical optimization only eight times with summed 479 iterations. The progressive results are shown in Figure 11. The progressive algorithm does not success in any case, so in step 7 of ***Algorithm*** ProgressiveOptimization( ) we detect whether the moving step is less than a threshold. If so, report failure and stop running the algorithm. However, in all of our testing examples, the algorithm works well.

Algorithm ProgressiveOptimization (P_{S_T})
Input : A region P_{S_T} in the $u-v$ parametric space.
Output : The non-self-overlapping grid M^f on P_{S_T}.
1. Compute the algebraic grid M^T on P_{S_T}, M^T has n_T facets;
2. Compute position of every grid node on M^0;
3. Compute the initial moving direction of every grid node by $G^0(\xi,\eta)$ and $G^T(\xi,\eta)$;
4. $\lambda_t \leftarrow 0$;
5. **do**{
6. $\Delta\lambda \leftarrow 2(1-\lambda_t)$;
7. When $\Delta\lambda < 10^{-16}$, report *failure* and terminate the algorithm;
8. **do**{
9. $\Delta\lambda \leftarrow \Delta\lambda/2$, and $\lambda_t \leftarrow \lambda_t + \Delta\lambda$;
10. Change the position of every grid node by Equation (16)—obtain the current grid M^{λ_t};
11. Compute the number n_- of facets with negative area on the current grid M^{λ_t};
12. }**while**$((n_-/n_T) > \tau)$;
13. Compute the numerical optimum $M^{\lambda_t *}$ of the current grid M^{λ_t};
14. Use $M^{\lambda_t *}$ to update the moving direction of every grid node by Equation (17);
15. }**while**$(\lambda_t < 1)$;
16. $M^f \leftarrow M^{\lambda_t *}$;
17. **return** M^f;
(* in our testing, we choose $\tau = 20\%$)

5. EXPERIMENTAL RESULTS

The proposed mesh algorithm has been implemented using Java language and separately tested on a PIII 900 MHz PC with a basic configuration and on a PIV 2.6 GHz PC with a modernist configuration; a number of test cases are tried. Figure 12 gives the non-self-overlapping grid generation result of the trimmed surface in Example I (initially given in Figure 2); Figure 13 shows the result of the patch in Example II with a mesh denser than the one shown in Figure 9; and the final surface grids in Example III are displayed in Figure 14. Example IV is a trimmed surface with six sides; its shape in the parametric surface is very convoluted so that the six-side region cannot be easily divided into two four-side sub-regions by straight lines since all the pertinent diagonals (shown as dash lines in Figure 15(a)) intersect the boundaries. As shown in Figure 15(b), the original algebraic grids from the Gregory patch mapping incurs severe self-overlapping. Using the presented mesh method, the self-overlapping is successfully eliminated in the final optimized grid, as given in Figure 15(c). The final surface grids are shown in Figure 15(d). For a trimmed surface patch with holes, we can first partition its parametric domain into several regions without holes and then apply the proposed mesh algorithm independently to these regions, e.g. Example V shown in Figure 16. All the boundary curves in our examples are presented by fourth-order Bezier curves. The control points of the Bezier boundary curves are listed in Table I.

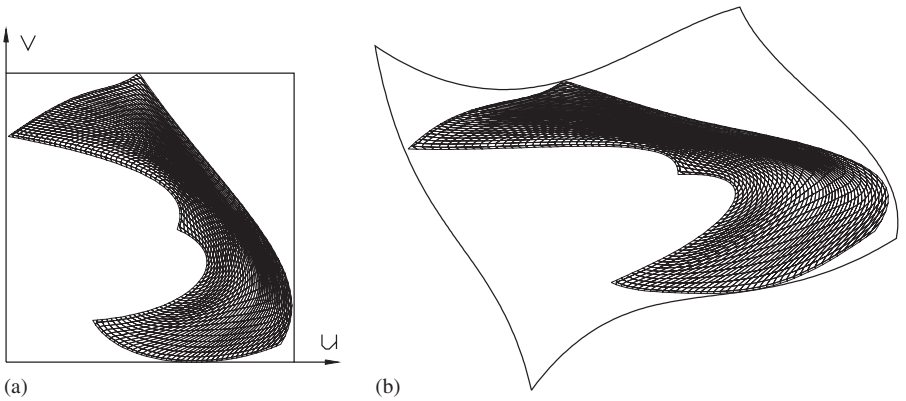

Figure 12. Result of Example I: (a) result grids in $u-v$ space; (b) final grids on the trimmed surface.

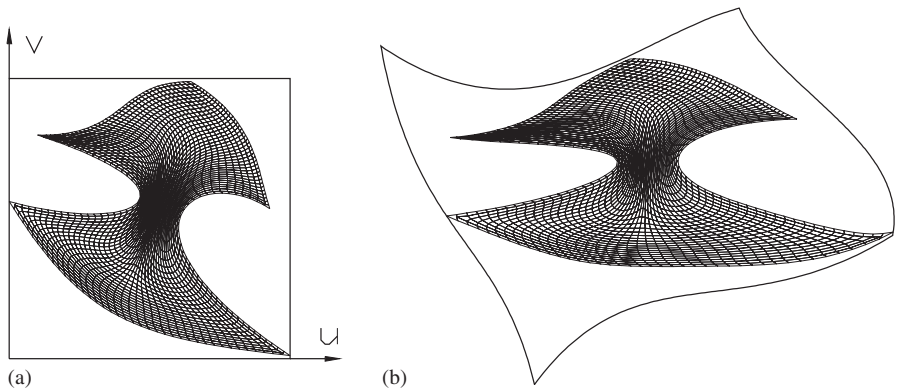

Figure 13. Result of Example II: (a) result grids in $u-v$ space; (b) final grids on the trimmed surface.

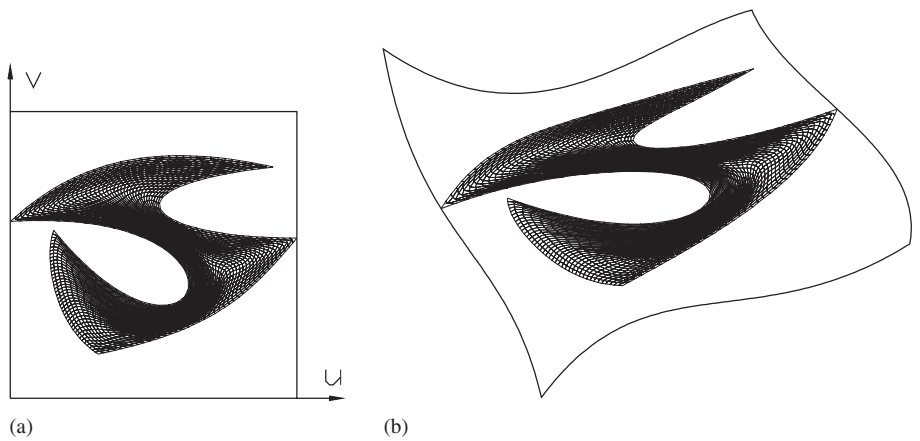

Figure 14. Result of Example III: (a) result grids in $u-v$ space; (b) final grids on the trimmed surface.

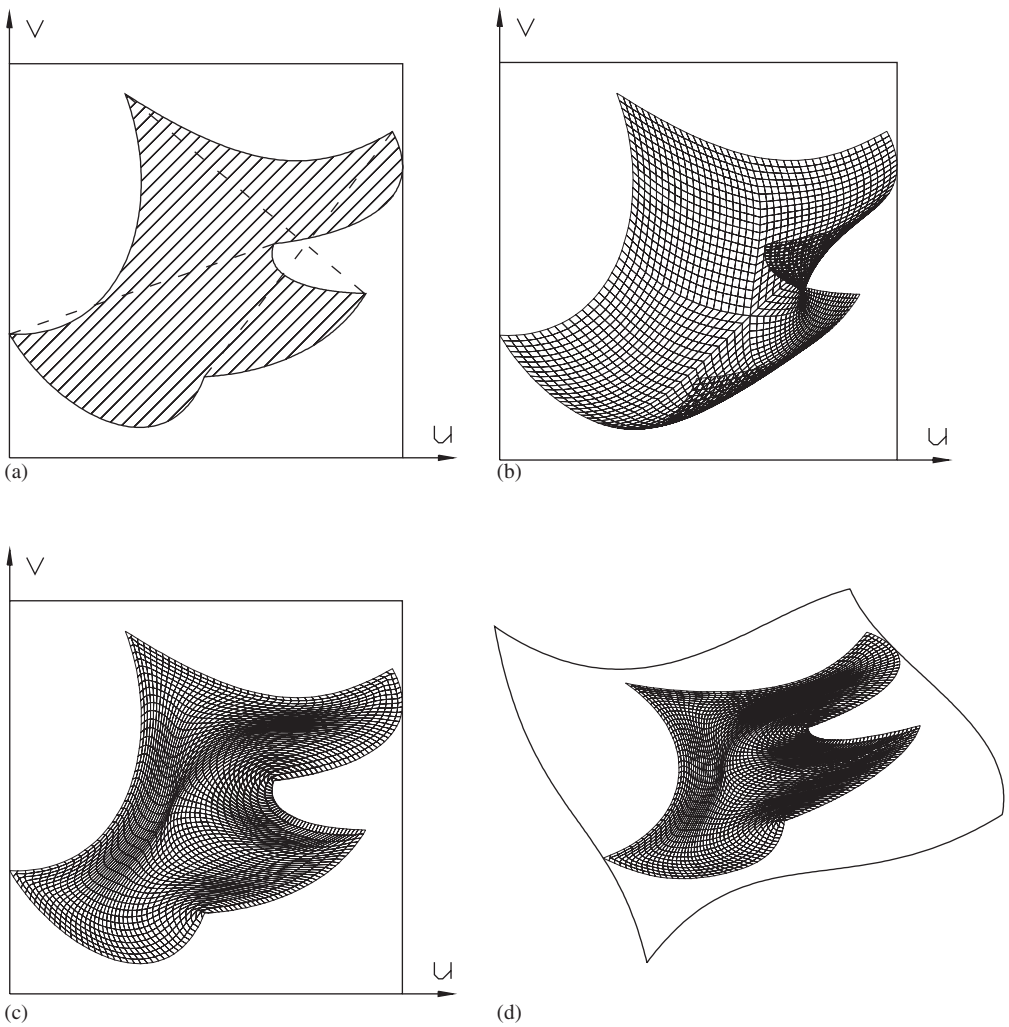

Figure 15. Example IV—a patch with six sides: (a) the parametric area in $u-v$ domain; (b) algebraic grids in $u-v$ domain; (c) grids after progressive optimization; (d) grids on the trimmed surface.

The computer running times of the given examples are tabulated in Table II. Usually, several minutes are needed. If the standard tools in commercial software are conducted to generate the grids on trimmed n-sided parametric patches, manual partitioning is required and it is a trial and error process—so it usually takes several hours of a human working. Also, it is hard to control the time cost. In some cases, especially the cases with convex boundaries, the traditional process can generate a good result very fast. However, if one is unluckily dealing with a much concaved parametric region, he may spend a lot of time on it. It is believed that with the increasing processing power available on the computer and with more efficient optimization algorithms, the running time can be shortened significantly. Consider about the

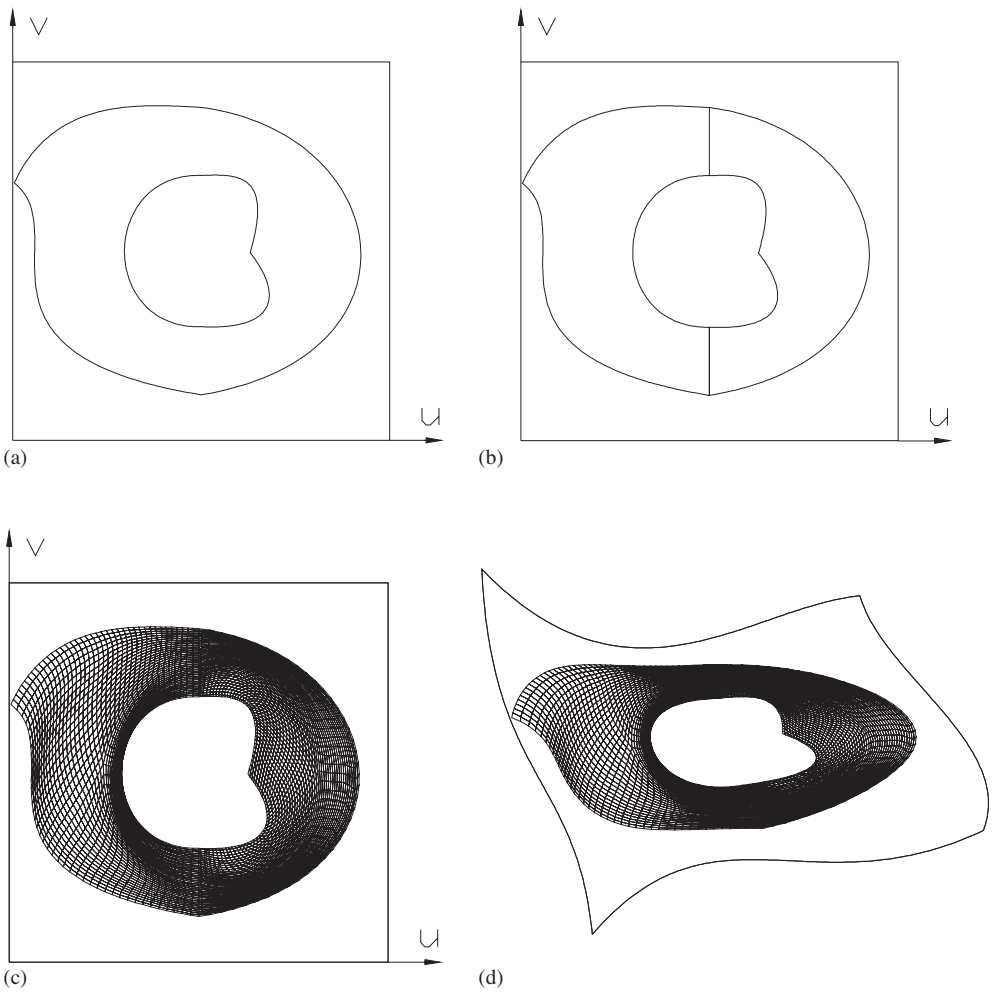

Figure 16. Example V—complex region with multiple patches: (a) given region in $u - v$ space; (b) given region is subdivided into two patches; (c) final grids in $u - v$ space; (d) final grids on the trimmed surface.

differences of computing time on PCs with basic and modernist configurations, the computing time of all the examples on PIV 2.6 GHz is almost one third of the time on PIII 900. On the PIV 2.6 GHz, it has already achieved an acceptable speed in the work of computer-aided engineering.

6. CONCLUSION

In this paper, we present a method for automatically constructing a structured grid system in an n-sided planar region bounded by parametric boundary curves of any form with only C^1 continuity, using a planar Gregory patch. This averts the need of manually partitioning

Table I. Control points of boundary curves.

Example	Curve no.	Control points
I	1	(0.86,0.90), (1.0,0.65), (0.67,0.40), (0.41,0.0)
	2	(0.41,0.0), (0.30,0.08), (0.26,0.01), (0.0,0.21)
	3	(0.0,0.21), (0.53,0.31), (0.56,0.42), (0.53,0.52)
	4	(0.53,0.52), (0.73,0.61), (0.60,0.79), (0.27,0.82)
	5	(0.27,0.82), (0.34,0.96), (0.63,1.0), (0.86,0.90)
II	1	(0.93,0.46), (0.88,0.33), (0.89,0.14), (0.65, 0.01)
	2	(0.65,0.01), (0.38,0.0), (0.42,0.20), (0.10,0.20)
	3	(0.10,0.20), (0.72,0.40), (0.46,0.57), (0.0,0.44)
	4	(0.0,0.44), (0.26,0.87), (0.46,0.94), (1.0,0.99)
	5	(1.0,0.99), (0.32,0.54), (0.66,0.29), (0.93,0.46)
III	1	(1.0,0.36), (0.50,0.37), (0.26,0.18), (0.92,0.10)
	2	(0.92,0.10), (0.46,0.0), (0.21,0.10), (0.0,0.30)
	3	(0.0,0.30), (0.94,0.24), (0.67,1.0), (0.15,0.34)
	4	(0.15,0.34), (0.12,0.44), (0.15,0.69), (0.31,0.79)
	5	(0.31,0.79), (0.70,0.70), (0.82,0.59), (1.0,0.36)
IV	1	(0.62,0.40), (0.74,0.38), (1.0,0.31), (0.90,0.10)
	2	(0.90,0.10), (0.73,0.21), (0.58,0.23), (0.27,0.0)
	3	(0.27,0.0), (0.37,0.29), (0.28,0.65), (0.0,0.64)
	4	(0.0,0.64), (0.12,0.86), (0.37,1.0), (0.46,0.75)
	5	(0.46,0.75), (0.62,0.74), (0.77,0.66), (0.84,0.53)
	6	(0.84,0.53), (0.71,0.52), (0.59,0.49), (0.62,0.40)

Table II. Time cost of examples.

Example	Result figures	Optimization type	Side number*	Time cost on PIII 900 MHz	Time cost on PIV 2.6 GHz
I	12	Progressive	5	4 min 30 s	1 min 41 s
II	9, 13	Pure numerical	5	2 min 11 s	49 s
III	10, 11, 14	Progressive	5	12 min 7 s	4 min 13 s
IV	15	Progressive	6	15 min 13 s	5 min 19 s
V	16	Progressive	2×5	4 min 9 s	1 min 33 s

*The grid size in each block is 20×20.

the n-sided region into four-sided sub-regions, which is a popular solution by most commercial meshing software. However, the algebraic grid thus generated may have self-overlapping which makes the mesh useless in most of engineering and scientific applications. A functional optimization method is then presented to eliminate such self-overlapping in the grid. Unlike PDE methods, the derivatives of a Gregory patch, which are very difficult to compute, are not required in our method. Thus, the speed of the optimization process of our approach is relatively fast. To resolve the difficulty of guessing good initial positions of every grid node for the conjugate gradient method, a progressive optimization algorithm is introduced, which has been shown to be very effective in a variety of practical examples. In summary, our approach provides a promising meshing tool in engineering.

Some new ideas about introducing additional terms to the Gregory definition have been proposed recently [18], which will allow extra controlling to the patch. One possible extension of our current work is to see if an algebraic mapping which guarantees non-self-overlapping can be determined by adjusting those additional terms of a Gregory patch. The process may also be a functional optimization approach like what is described in this paper. Or we can alternate the 3D vector functions along the boundaries so that the positions of inner nodes are adaptively modified to achieve a non-self-overlapping grid.

REFERENCES

1. Mortenson ME. *Geometric Modeling* (2nd Edn). Wiley: New York, 1997.
2. Thompson JF, Soni BK, Weatherill NP. *Handbook of Grid Generation*. CRC Press: Florida, 1999.
3. Brakhage KH, Muller S. Algebraic-hyperbolic grid generation with precise control of intersection of angles. *International Journal for Numerical Methods in Fluids* 2000; **33**:89–123.
4. Shih TIP, Bailey RT, Nguyen HL, Roelke RJ. Algebraic grid generation for complex geometries. *International Journal for Numerical Methods in Fluids* 1991; **13**:1–31.
5. Smith RE, Eriksson LE. Algebraic grid generation. *Computer Methods in Applied Mechanics and Engineering* 1987; **64**:285–300.
6. Kim S. Control functions and grid qualities measurements in the elliptic grid generation around arbitrary surfaces. *International Journal for Numerical Methods in Fluids* 2000; **33**:81–88.
7. Knupp PM. Jacobian-weighted elliptic grid generation. *SIAM Journal on Scientific Computing* 1996; **17**: 1475–1490.
8. Khamayseh A, Hamann B. Elliptic grid generation using NURBS surfaces. *Computer Aided Geometric Design* 1996; **13**:369–386.
9. Soni BK. Elliptic grid generation system: control functions revisited I. *Applied Mathematics and Computation* 1993; **59**:151–163.
10. Thompson JF. A survey of dynamically-adaptive grids in the numerical solution of partial differential equations. *Applied Numerical Mathematics* 1985; **1**:3–27.
11. Arina R. Orthogonal and conformal surface grid generation. *Applied Numerical Mathematics* 2003; **46**(3–4): 249–262.
12. Tam TKH, Armstrong CG. 2D Finite element mesh generation by medial axis subdivision. *Advances in Engineering Software and Workstations* 1991; **13**:313–324.
13. Yamakawa S, Shimada K. Quad-layer: layered quadrilateral meshing of narrow two-dimensional domains by bubble packing and chordal axis transformation. *Journal of Mechanical Design*, *Transactions of the ASME* 2002; **124**:564–573.
14. Owen SJ, Staten ML, Canann SA, Saigal S. Q-Morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering* 1999; **44**:1317–1340.
15. Lee YK, Lee CK. Automatic generation of anisotropic quadrilateral meshes on three-dimensional surfaces using metric specifications. *International Journal for Numerical Methods in Engineering* 2002; **53**:2673–2700.
16. Gregory JA. *N*-sided surface patches. *Mathematics of Surfaces. Proceedings of a Conference*. Clarendon Press: Oxford, UK, 1986; 217–232.
17. Gregory JA, Yuen PK. An arbitrary mesh network scheme using rational splines. *Mathematical Methods in Computer Aided Geometric Design II*, Lyche T, Schumaker LL (eds). Academic Press: New York, 1992; 321–329.
18. Hall R, Mullineux G. Shape modification of Gregory patches. *The Mathematics of Surfaces VII*, Goodman T, Martin R (eds). Information Geometers: University of Wales, Cardiff, UK, 1997; 393–408.
19. Knupp P, Steinberg S. *The Fundamentals of Grid Generation*. CRC Press: Florida, 1993.
20. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C*: *The Art of Scientific Computing*. Cambridge University Press: Cambridge, 1992.

