Algebraic grid generation on trimmed parametric surface using non-self-overlapping planar Coons patch

Charlie C. L. Wang
Department of Automation and Computer-Aided Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
E-mail: cwang@acae.cuhk.edu.hk

Kai Tang*
Department of Mechanical Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
E-mail: mektang@ust.hk

Abstract

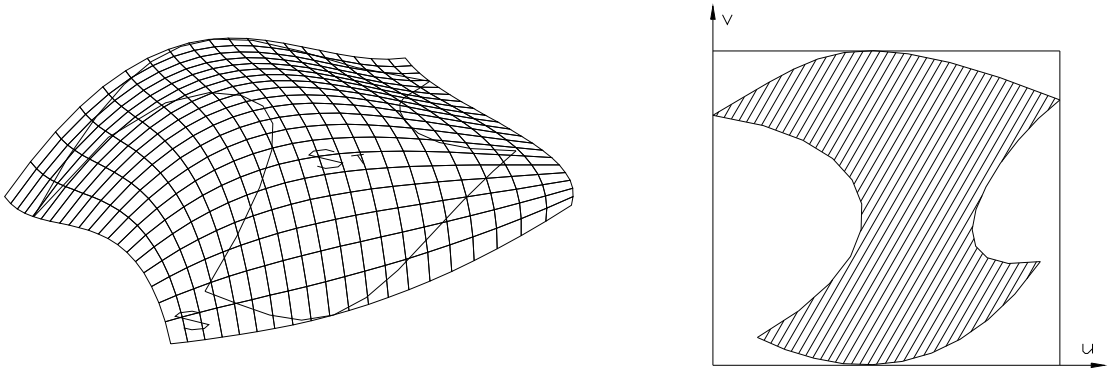
Using a Coons patch mapping to generate the structured grid in the parametric region of a trimmed surface can avoid the singularity of elliptic PDE methods when only C^1 continuous boundary is given; the error of converting generic parametric C^1 boundary curves into a specified representation form is also avoided. However, overlap may happen on some portions of the algebraically generated grid when a linear or naïve cubic blending function is used in the mapping; this severely limits its usage in most of engineering and scientific applications where a grid system of non-self-overlapping is strictly required. To solve the problem, non-trivial blending functions in a Coons patch mapping should be determined adaptively by the given boundary so that self-overlapping can be averted. We address the adaptive determination problem by a functional optimization method. The governing equation of the optimization is derived by adding a virtual dimension in the parametric space of the given trimmed surface. Both one-parameter and two-parameter blending functions are studied. To resolve the difficulty of guessing good initial blending functions for the conjugate gradient method used, a progressive optimization algorithm is then proposed which has been shown to be very effective in a variety of practical examples. Also, an extension is on the objective function to control the element shape. Finally, experiment results are shown to illustrate the usefulness and effectiveness of the presented method.

Keywords: algebraic grid generation, trimmed surface, parametric space, self-overlap, and Coons patch

* Corresponding author: mektang@ust.hk

1. Introduction

This paper presents a method for generating algebraic grids on a trimmed surface by fitting a non-self-overlapping planar Coons patch into the parametric region of the given surface. In computer-aided engineering, geometric modeling, computer graphics, and many other applications, a parametric surface patch usually intersects with other surfaces, and thus only a portion of the surface patch is used in defining a meaningful shape [1]. The remaining portion of parametric surface patch S after trimming by other surfaces 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 that of S. The parametric area of S_T – 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 Fig. 1-1). Each boundary curve of P_{S_T} is expressed as a parametric equation of the form $b_i = \begin{bmatrix} u_i(t) & v_i(t) \end{bmatrix}$, where $t \in [0,1]$.



(a) a trimmed surface **(b) the parametric area in $u-v$ domain**

Fig. 1-1 A trimmed surface and its related parametric area

An essential task in many engineering disciplines is to approximate a trimmed surface by an aggregate of simple planar elements, e.g., triangles and quadrilateral elements; this is referred to as the surface meshing operation. Surface mesh generation methods have been studied for many years. Both structured and unstructured grids can be constructed on three-dimensional surfaces [2]. The two most powerful analysis tools in engineering are the finite element method and the finite difference method. The finite element method usually conducts either triangular grids or quadrilateral grids, and the grids can be structured or unstructured. Quadrilateral grids in general have better quality than the triangular grids. Unstructured 2D quadrilateral grids can be constructed directly by geometry decomposition [3-6], boundary advancing [7-10], and skeleton construction [11, 12]; or indirectly by the background triangular grids [13-17]. 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). Algebraic grid generation is some form of interpolation from boundary points – different approaches use different kinds of interpolation [18-20]. Overlap may, however, happen on some portions of algebraically generated grids. 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 [21-25]. The smoothness properties and extremum principles of some such PDE systems can serve to produce smooth grids without boundary overlap. However, since the elliptic PDE methods require second derivative, singularity appears when any boundary curve of P_{S_T} is only C^1 continuity.

Similar to other algebraic grid generation approaches, our grid construction method also consists of three steps: 1) forward mapping; 2) grid generation; and 3) back mapping. The forward mapping is the mapping of the three-dimensional physical surface S_T to its underlying parametric area P_{S_T}. Once the forward mapping is finished, the grids are produced in P_{S_T} by fitting a planar Coons patch $X(\xi,\eta)$. After establishing the uniform $M \times N$ grid in the parametric domain of $S_C(\xi,\eta)$ by

$$\begin{cases} \xi_i = \dfrac{i}{M} & (i = 0,\cdots,M) \\ \eta_j = \dfrac{j}{N} & (j = 0,\cdots,N) \end{cases}, \tag{1-1}$$

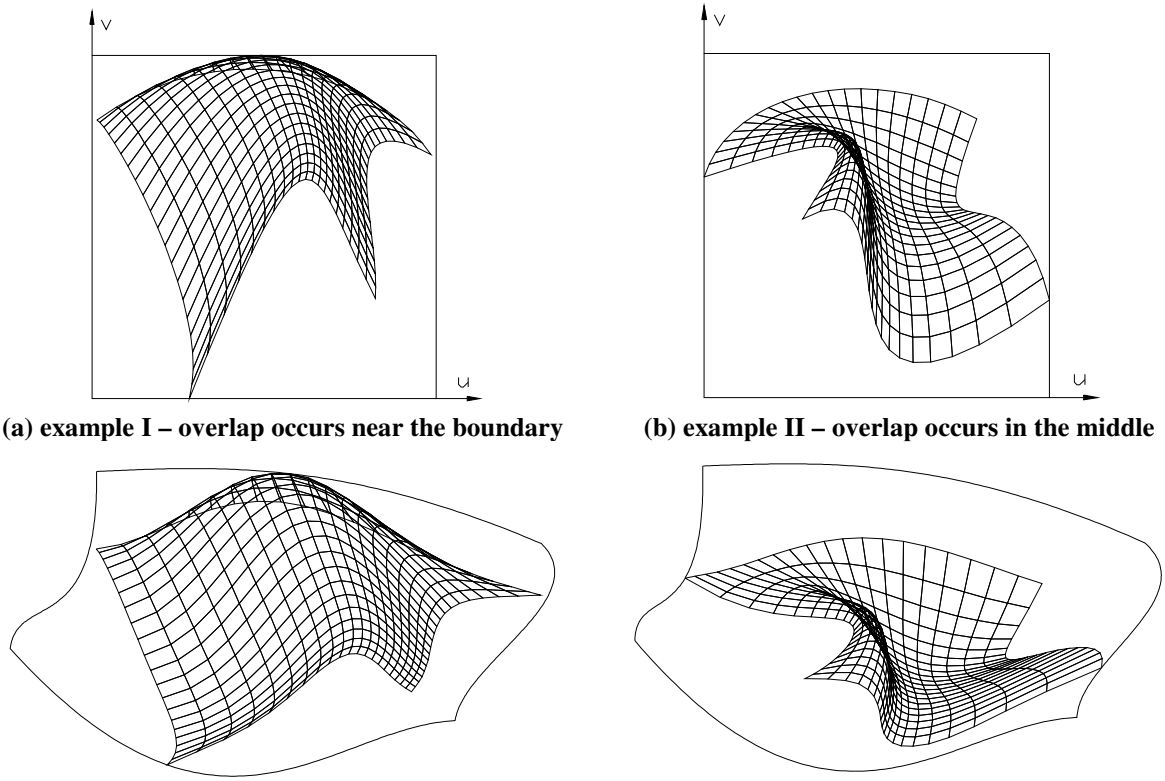
the $M \times N$ grid on the physical trimmed space can be expressed by the back mapping in the form

$$S_{T_{i,j}} = S(u(X(\xi_i,\eta_j)), v(X(\xi_i,\eta_j))), \tag{1-2}$$

where the functions $u(\cdots)$ and $v(\cdots)$ represent the u and v coordinates of a point on the Coons patch. However, as mentioned above, overlap may happen on the boundary of P_{S_T} or even in the middle of P_{S_T}, which of course influences the final surface grid (see Fig. 1-2). Aimed at resolving this perplexing issue, a method based on finding proper blending functions in $X(\xi,\eta)$ is explored in this paper, which assures the *non-self-overlapping* property. Since using a Coons patch does not need to convert the representation of boundary curves, no converting error is generated in this method. For the P_{S_T} with complex boundaries, the whole P_{S_T} can be subdivided into several sub-regions, where each region has only four boundary curves and its corner points satisfy the non-self-overlapping condition.

We organize the paper as follows. In section 2, some necessary definitions and preliminaries are first given. Next, the governing equation is derived in section 3 by the normal of the Coons patch in a virtual direction – w.

Based on the governing equation, the formulas to compute the functional optimum of one-parameter blending functions in the Coons patch is presented in section 4; and in section 5, the blending functions are extended to two-parameter functions to achieve more flexibility. In section 6, to overcome the difficulty of "guessing" a good initial blending function, an algorithm for computing the optimum progressively is given. Finally, section 7 shows the possible extension of our method to control the shape of elements.



(a) example I – overlap occurs near the boundary **(b) example II – overlap occurs in the middle**

(c) surface grid of example I **(d) surface grid of example II**

Fig. 1-2 Overlap occurs near the boundary or in the middle of P_{S_T}

2. Preliminary

We first give necessary definitions and preliminaries.

Definition 2.1 Given four C^1 curves $Q_0(\xi)$, $Q_1(\xi)$, $P_0(\eta)$ and $P_1(\eta)$ $(0 \le \xi, \eta \le 1)$ in three-dimensional space that form a closed curve, the *Coons patch* defined on them is

$$X(\xi,\eta) = \begin{bmatrix}\alpha(\xi) & 1-\alpha(\xi)\end{bmatrix}\begin{bmatrix}P_0(\eta)\\ P_1(\eta)\end{bmatrix} + \begin{bmatrix}Q_0(\xi) & Q_1(\xi)\end{bmatrix}\begin{bmatrix}\beta(\eta)\\ 1-\beta(\eta)\end{bmatrix} - \begin{bmatrix}\alpha(\xi) & 1-\alpha(\xi)\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\beta(\eta)\\ 1-\beta(\eta)\end{bmatrix}$$

where $\alpha(\xi)$ is a C^1 function of ξ in $[0, 1]$ satisfying $\alpha(0) = 1$ and $\alpha(1) = 0$, $\beta(\eta)$ is a C^1 function of η in $[0, 1]$ satisfying $\beta(0) = 1$ and $\beta(1) = 0$, and the parametric space of the Coons patch is $(\xi, \eta) \in [0, 1] \times [0, 1]$.

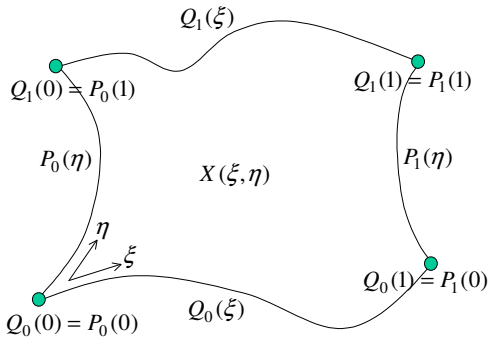


Fig. 2-1 Illustration of the boundary curves on a Coons patch

The two functions $\alpha(\xi)$ and $\beta(\eta)$ are referred to as the *blending functions* of a Coons patch. As the four curves are all C^1 continuous, $X(\xi,\eta)$ has partial derivatives in the entire parameter domain. By its definition, it can be easily verified that the mapping $X(\xi,\eta)$ is boundary conforming; that is, we have $X(\xi,0) = Q_0(\xi)$, $X(\xi,1) = Q_1(\xi)$, $X(0,\eta) = P_0(\eta)$, and $X(1,\eta) = P_1(\eta)$. Therefore, $Q_0(\xi)$, $Q_1(\xi)$, $P_0(\eta)$ and $P_1(\eta)$ are the *boundary curves* of the corresponding Coons patch.

In $X(\xi,\eta)$, the four curves are connected in the way that $Q_0(1) = P_1(0)$, $P_1(1) = Q_1(1)$, $Q_1(0) = P_0(1)$, and $P_0(0) = Q_0(0)$ (illustrated in Fig. 2-1); points $Q_0(0)$, $Q_0(1)$, $Q_1(0)$, and $Q_1(1)$ are the *corner points* of $X(\xi,\eta)$. Since the four curves are given conditions, they cannot be changed in our computation. The blending function $\alpha(\xi)$ gives an interpolation between $P_0(\eta)$ and $P_1(\eta)$, and $\beta(\eta)$ gives the interpolation between $Q_0(\xi)$ and $Q_1(\xi)$. They are not restricted to be one-parameter functions though; two-parameter functions can also be used.

If the four boundary curves of a Coons patch $X(\xi,\eta)$ all lie in a common plane, obviously $X(\xi,\eta)$ also lies in that plane, i.e., it is a *planar Coons patch*. Unless specially noted in the context, hereafter a planar Coons patch is assumed. As it is planar, the set of all the points of $X(\xi,\eta)$ for $(\xi,\eta) \in [0,1]\times[0,1]$, denoted as Ω_X, form a compact region in the $u-v$ plane. On the other hand, since the four defining curves of $X(\xi,\eta)$ – $Q_0(\xi)$, $Q_1(\xi)$, $P_0(\eta)$ and $P_1(\eta)$ – also all lie in this same plane, they enclose a simple region Ω. We next define an important criterion on the relationship between the two regions.

Definition 2.2 A Coons patch $X(\xi,\eta)$ is said to be a *graph-mapping* if it is a one-to-one mapping between the square $[0,1]\times[0,1]$ in the $\xi-\eta$ plane and the region Ω_X.

Since an $X(\xi,\eta)$ is always boundary conforming, one can prove that the two regions Ω_X and Ω are identical to each other if $X(\xi,\eta)$ is a graph-mapping. Moreover, because a graph-mapping $X(\xi,\eta)$ is one-to-one, one can mesh the region Ω_X, and hence Ω, by simply first uniformly meshing the square $[0,1]\times[0,1]$ in the $\xi-\eta$ plane and then map this mesh to Ω_X (and Ω) by the mapping $X(\xi,\eta)$, which is guaranteed to be free of self-overlapping. As Ω can be taken as the parametric area in the $u-v$ plane of a trimmed surface S_T, a meshing of Ω thus introduces a valid meshing of S_T.

But, exactly how should this graph-mapping be rigorously and mathematically characterized? This calls for the introduction of the governing function as described in the next section.

3. Governing Equation

In this section, the governing equation to eliminate the self-overlap is derived. Let's add a virtual axis w perpendicular to the $u-v$ plane, i.e., $w=u\times v$. A Coons patch $X(\xi,\eta)$ now is a planar region embedded in the $u\times v\times w$ space, denoted as $X(\xi,\eta)=\begin{bmatrix}U(\xi,\eta)\\V(\xi,\eta)\\W(\xi,\eta)\end{bmatrix}$, where $U(\xi,\eta)$, $V(\xi,\eta)$, and $W(\xi,\eta)$ are the components of $X(\xi,\eta)$ 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 $X(\xi,\eta)$ as $N(\xi_0,\eta_0)=\left.\dfrac{X_\xi\times X_\eta}{\left\|X_\xi\times X_\eta\right\|}\right|_{(\xi_0,\eta_0)}$. A point (ξ_0,η_0) is said to be *singular* if its corresponding $\left\|X_\xi\times X_\eta\right\|$ is a zero vector. The lemma below is important as it stipulates the condition for guaranteeing the graph-mapping property.

Lemma 3.1 The Coons patch $X(\xi,\eta)=\begin{bmatrix}U(\xi,\eta)\\V(\xi,\eta)\\W(\xi,\eta)\end{bmatrix}$ is a graph-mapping if and only if there is no any singular point in the square $[0,1]\times[0,1]$.

(The proof is given in Appendix A))

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

Proposition 3.1 If the normal at every point to the Coons patch $X(\xi,\eta)$ has the same sign in w, then $X(\xi,\eta)$ is a graph-mapping.

Without loss of generality, we can assume that the sign in *w* of the normal vectors of a graph-mapping $X(\xi,\eta)$ is always positive. To facilitate the formulation, we use the following two definitions.

Definition 3.1 A point (ξ_d,η_d) is called a *shadow point* of $X(\xi,\eta)$ if the normal at $X(\xi_d,\eta_d)$ is along the negative *w*-axis.

Definition 3.2 The set R_d of all the shadow points of $X(\xi,\eta)$ is defined as the *shadow region* of $X(\xi,\eta)$.

Shadow points must be prevented at the corners. It is because the normal direction at a corner point is determined and only determined by the two tangents on its related two boundary curves. No matter how you change the blending functions of the Coons patch, the tangents on the boundary curves are not changed. Thus, the following assumption is imposed on the four defining curves.

Assumption 3.1 The normals of $Q_0, Q_1, P_0,$ and P_1 at the four corners respectively are along the positive *w*-axis.

From the above analysis, we find that the governing equation should be a function indicating the area of the shadow region of $X(\xi,\eta)$. Since the projection of $X_\xi \times X_\eta$ along the *w*-axis is $X_{\xi_u} X_{\eta_v} - X_{\eta_u} X_{\xi_v}$ (where $X_{\xi_u} = U_\xi$, $X_{\eta_u} = U_\eta$, $X_{\xi_v} = V_\xi$, and $X_{\eta_v} = V_\eta$), the final governing equation to construct a non-self-overlapping planar Coons patch on four given boundaries is

$$\iint_\Psi \mathrm{H}[-(X_{\xi_u} X_{\eta_v} - X_{\eta_u} X_{\xi_v})]d\xi d\eta = 0, \tag{3-1}$$

where $\mathrm{H}(\cdots)$ is the *Heaviside* function, and Ψ is the overall domain of $(\xi,\eta) \in [0,1]\times[0,1]$. Actually, the left part of equation (3-1) is exactly the area of the shadow region in the $u-v$ space.

4. One-parameter Blending Function

To achieve (3-1), we use the functional optimization method to determine the algebraic function of a non-self-overlapping planar Coons patch within the Jordan curve formed by $Q_0(\xi)$, $Q_1(\xi)$, $P_0(\eta)$, and $P_1(\eta)$. In

the definition of a Coons patch, only the two blending functions, i.e., $\alpha(\xi)$ and $\beta(\eta)$, are flexible to be changed during the optimization. As polynomials are most widely used in computer-aided geometric design due to its many merits, we also use polynomials for the blending functions. They are one-parameter functions. Thus, the following proposition comes.

Proposition 4.1 If a blending function $\alpha(\xi)$ of degree n in ξ direction and a blending function $\alpha(\eta)$ of degree m in η direction are used, we have totally $n+m-2$ degree of freedom to control the distribution of grid points inside $X(\xi,\eta)$ in the $u-v$ parametric space.

When $\alpha(\xi)$ is a polynomial of degree n, it can be represented by $\alpha(\xi)=\sum_{i=0}^{n} a_i\xi^i$. Since $\alpha(\xi)$ should satisfy $\alpha(0)=1$ and $\alpha(1)=0$, we can determine that $a_0=1$ and $a_1=-1-\sum_{i=2}^{n} a_i$. Therefore, $\alpha(\xi)$ is expressed as

$$\alpha(\xi)=1-(1+\sum_{i=2}^{n} a_i)\xi+\sum_{i=2}^{n} a_i\xi^i \tag{4-1}$$

where $a_i, i=2,...,n$, are variables to be determined. Similarly, we can represent $\beta(\eta)$ by

$$\beta(\eta)=1-(1+\sum_{j=2}^{m} b_j)\eta+\sum_{j=2}^{m} b_j\eta^j \tag{4-2}$$

where $b_j, j=2,...,m,$ are variables to be determined. Therefore, totally $n+m-2$ variables can be adjusted to determine the optimized algebraic function of $X(\xi,\eta)$; they form the solution vector of the functional optimization as $\chi=[a_2 \quad \cdots \quad a_n \quad b_2 \quad \cdots \quad b_m]$.

From the governing equation (3-1), the objective function in the functional optimization is given as

$$A=\iint_{\Psi} \mathrm{H}[-(X_{\xi_u}X_{\eta_v}-X_{\eta_u}X_{\xi_v})]d\xi d\eta\,. \tag{4-3}$$

From the definition of Coons patch (Definition 2.1), the partial derivatives of $X(\xi,\eta)$ are

$$X_\xi=\frac{\partial X(\xi,\eta)}{\partial \xi}=\begin{bmatrix}U_\xi\\V_\xi\end{bmatrix}=[\alpha'(\xi) \quad -\alpha'(\xi)]\begin{bmatrix}P_0(\eta)\\P_1(\eta)\end{bmatrix}+[Q_0'(\xi) \quad Q_1'(\xi)]\begin{bmatrix}\beta(\eta)\\1-\beta(\eta)\end{bmatrix}-[\alpha'(\xi) \quad -\alpha'(\xi)]\begin{bmatrix}Q_0(0) & Q_1(0)\\Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\beta(\eta)\\1-\beta(\eta)\end{bmatrix}$$

and

$$X_\eta = \frac{\partial X(\xi,\eta)}{\partial \eta} = \begin{bmatrix} U_\eta \\ V_\eta \end{bmatrix} = \begin{bmatrix} \alpha(\xi) & 1-\alpha(\xi) \end{bmatrix} \begin{bmatrix} P_0'(\eta) \\ P_1'(\eta) \end{bmatrix} + \begin{bmatrix} Q_0(\xi) & Q_1(\xi) \end{bmatrix} \begin{bmatrix} \beta'(\eta) \\ -\beta'(\eta) \end{bmatrix} - \begin{bmatrix} \alpha(\xi) & 1-\alpha(\xi) \end{bmatrix} \begin{bmatrix} Q_0(0) & Q_1(0) \\ Q_0(1) & Q_1(1) \end{bmatrix} \begin{bmatrix} \beta'(\eta) \\ -\beta'(\eta) \end{bmatrix}.$$

From (4-1) and (4-2), we obtain $\alpha'(\xi) = -1 - \sum_{i=2}^{n} a_i + \sum_{i=2}^{n} i a_i \xi^{i-1}$ and $\beta'(\eta) = -1 - \sum_{j=2}^{m} b_j + \sum_{j=2}^{m} j b_j \eta^{j-1}$, after substituting them into $X_\xi(\xi,\eta)$ and $X_\eta(\xi,\eta)$, the following formulas are determined.

$$\begin{aligned} X_\xi(\xi,\eta) = {} & (P_0(\eta) - P_1(\eta)) \left(-1 - \sum_{i=2}^{n} a_i + \sum_{i=2}^{n} i a_i u^{i-1} \right) + Q_0'(\xi) + (Q_0'(\xi) - Q_1'(\xi)) \left(-(1 + \sum_{j=2}^{m} b_j)\eta + \sum_{j=2}^{m} b_j \eta^j \right) \\ & - \left(-1 - \sum_{i=2}^{n} a_i + \sum_{i=2}^{n} i a_i \xi^{i-1} \right) \begin{bmatrix} Q_0(0) - Q_0(1) \\ Q_1(0) - Q_1(1) \end{bmatrix}^T \begin{bmatrix} 1 - (1 + \sum_{j=2}^{m} b_j)\eta + \sum_{j=2}^{m} b_j \eta^j \\ (1 + \sum_{j=2}^{m} b_j)\eta - \sum_{j=2}^{m} b_j \eta^j \end{bmatrix} \end{aligned} \tag{4-4}$$

$$\begin{aligned} X_\eta(\xi,\eta) = {} & P_0'(\eta) + (P_0'(\eta) - P_1'(\eta)) \left(-(1 + \sum_{i=2}^{n} a_i)\xi + \sum_{i=2}^{n} a_i \xi^i \right) + (Q_0(\xi) - Q_1(\xi)) \left(-1 - \sum_{j=2}^{m} b_j + \sum_{j=2}^{m} j b_j \eta^{j-1} \right) \\ & - \begin{bmatrix} 1 - (1 + \sum_{i=2}^{n} a_i)\xi + \sum_{i=2}^{n} a_i \xi^i \\ (1 + \sum_{i=2}^{n} a_i)\xi - \sum_{i=2}^{n} a_i \xi^i \end{bmatrix}^T \begin{bmatrix} Q_0(0) - Q_1(0) \\ Q_0(1) - Q_1(1) \end{bmatrix} \left(-1 - \sum_{j=2}^{m} b_j + \sum_{j=2}^{m} j b_j \eta^{j-1} \right) \end{aligned} \tag{4-5}$$

By the above formulas, the objective function to determine a non-self-overlapping planar Coons patch can be computed numerically. When optimizing the planar Coons patch, we conduct the *Conjugate Gradient Method* [27] to search for the final solution vector in the problem space. From (4-3), the gradient direction of χ is

$$\frac{\partial A}{\partial a_i} = \iint_\Psi \delta[-(X_{\xi_u} X_{\eta_v} - X_{\eta_u} X_{\xi_v})] \left(\frac{\partial X_{\eta_u}}{\partial a_i} X_{\xi_v} + \frac{\partial X_{\xi_v}}{\partial a_i} X_{\eta_u} - \frac{\partial X_{\xi_u}}{\partial a_i} X_{\eta_v} - \frac{\partial X_{\eta_v}}{\partial a_i} X_{\xi_u} \right) d\xi d\eta, \tag{4-6}$$

$$\frac{\partial A}{\partial b_j} = \iint_\Psi \delta[-(X_{\xi_u} X_{\eta_v} - X_{\eta_u} X_{\xi_v})] \left(\frac{\partial X_{\eta_u}}{\partial b_j} X_{\xi_v} + \frac{\partial X_{\xi_v}}{\partial b_j} X_{\eta_u} - \frac{\partial X_{\xi_u}}{\partial b_j} X_{\eta_v} - \frac{\partial X_{\eta_v}}{\partial b_j} X_{\xi_u} \right) d\xi d\eta. \tag{4-7}$$

where $\delta(t) = \frac{d}{dt} \mathrm{H}(t)$. To calculate the equations (4-6) and (4-7), we need the formulas of $\frac{\partial X_\xi}{\partial a_i}$, $\frac{\partial X_\eta}{\partial a_i}$, $\frac{\partial X_\xi}{\partial b_i}$, and $\frac{\partial X_\eta}{\partial b_i}$. They are determined from (4-4) and (4-5) as

$$\frac{\partial X_\xi(\xi,\eta)}{\partial a_i} = \left(P_0(\eta) - P_1(\eta) - \begin{bmatrix} Q_0(0) - Q_0(1) \\ Q_1(0) - Q_1(1) \end{bmatrix}^T \begin{bmatrix} 1 - (1 + \sum_{j=2}^{m} b_j)\eta + \sum_{j=2}^{m} b_j \eta^j \\ (1 + \sum_{j=2}^{m} b_j)\eta - \sum_{j=2}^{m} b_j \eta^j \end{bmatrix} \right) (-1 + i\xi^{i-1}), \tag{4-8}$$

$$\frac{\partial X_\eta(\xi,\eta)}{\partial a_i} = \left(P_0'(\eta) - P_1'(\eta) - (Q_0(0) - Q_1(0) - Q_0(1) + Q_1(1)) \left(-1 - \sum_{j=2}^{m} b_j + \sum_{j=2}^{m} j b_j \eta^{j-1} \right) \right) (-\xi + \xi^i), \quad (4\text{-}9)$$

$$\frac{\partial X_\xi(\xi,\eta)}{\partial b_j} = \left(Q_0'(\xi) - Q_1'(\xi) - (Q_0(0) - Q_0(1) - Q_1(0) + Q_1(1)) \left(-1 - \sum_{i=2}^{n} a_i + \sum_{i=2}^{n} i a_i \xi^{i-1} \right) \right) (-\eta + \eta^j), \quad (4\text{-}10)$$

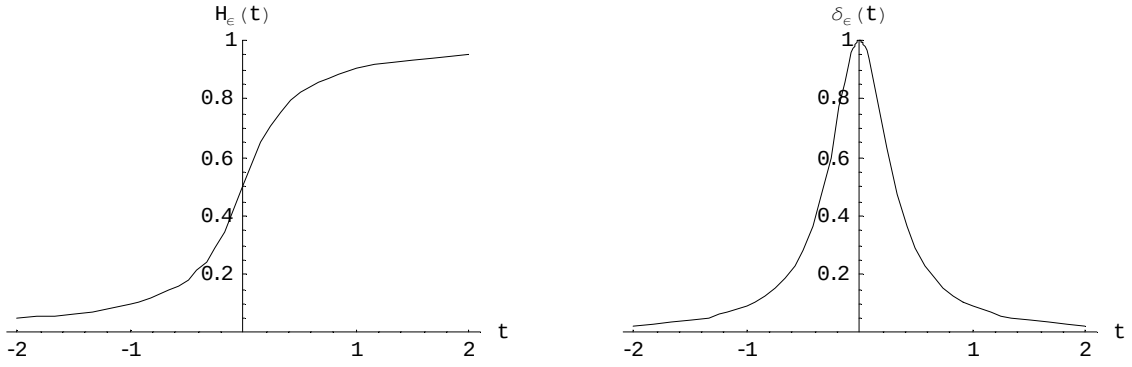
$$\frac{\partial X_\eta(\xi,\eta)}{\partial b_j} = \left(Q_0(\xi) - Q_1(\xi) - \begin{bmatrix} 1 - (1 + \sum_{i=2}^{n} a_i)\xi + \sum_{i=2}^{n} a_i \xi^i \\ (1 + \sum_{i=2}^{n} a_i)\xi - \sum_{i=2}^{n} a_i \xi^i \end{bmatrix}^T \begin{bmatrix} Q_0(0) - Q_1(0) \\ Q_0(1) - Q_1(1) \end{bmatrix} \right) (-1 + j\eta^{j-1}). \quad (4\text{-}11)$$

The final optimized algebraic functions of $X(\xi,\eta)$ are computed iteratively; when $A / \overline{A} < \mu$, the iteration stops (where A is the value of objective function, and $\overline{A}$ is the area of the Coons patch $X(\xi,\eta)$). μ is an empirical threshold number and in our system is set to 0.125%. To have the tendency to compute a global minimization, the following approximation for $\mathrm{H}(t)$ and $\delta(t)$ are usually adopted [28]:

$$\mathrm{H}_\varepsilon(t) = \frac{1}{2}(1 + \frac{2}{\pi}\arctan(\frac{t}{\varepsilon})), \quad (4\text{-}12)$$

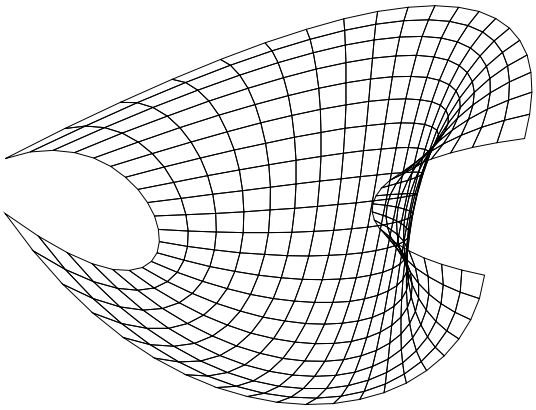
$$\delta_\varepsilon(t) = \frac{\varepsilon}{\pi(\varepsilon^2 + t^2)}. \quad (4\text{-}13)$$

As $\varepsilon \to 0$, the approximation converges to give theoretical $\mathrm{H}(t)$ and $\delta(t)$. In our testing examples, we usually use $\varepsilon = \frac{1}{\pi}$, which makes $\delta_\varepsilon(0) = 1$. The figures of $\mathrm{H}_\varepsilon(t)$ and $\delta_\varepsilon(t)$ with $\varepsilon = \frac{1}{\pi}$ are given as below.

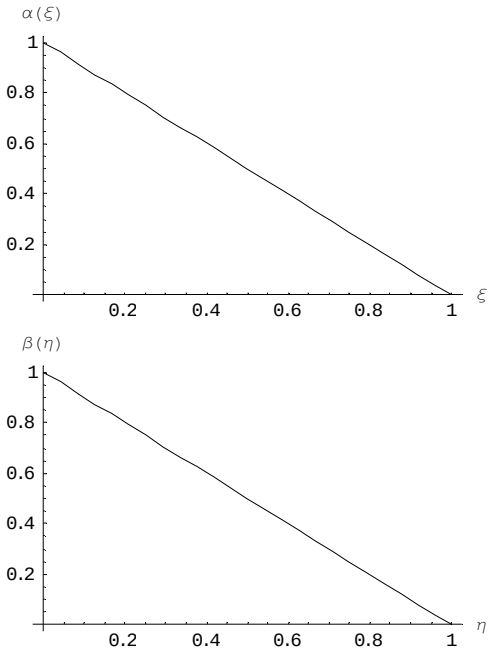


(a) $\mathrm{H}_\varepsilon(t)$ **(b)** $\delta_\varepsilon(t)$

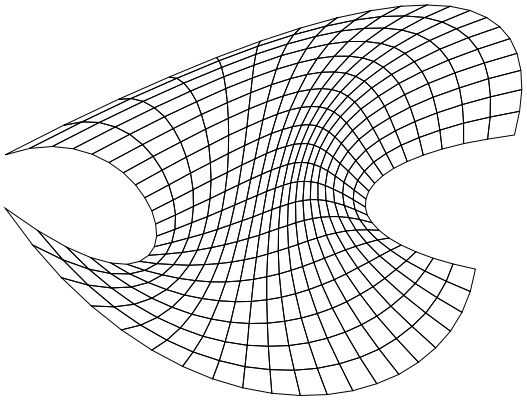
Fig. 4-1 Approximation of $\mathrm{H}(t)$ and $\delta(t)$



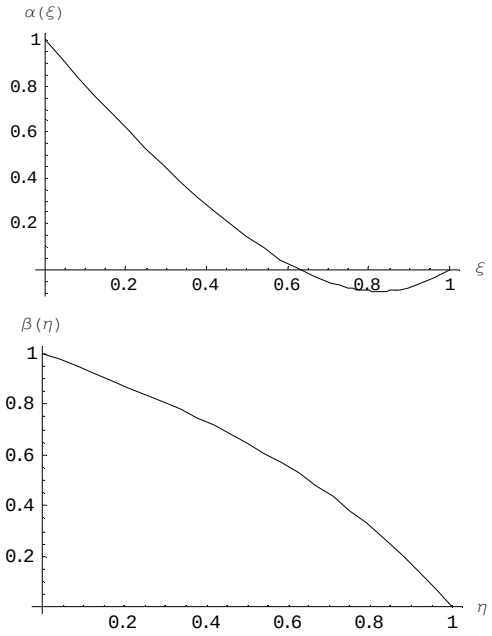
(a) before optimization



(b) $\alpha(\xi)$ and $\beta(\eta)$ in (a)



(c) after optimization



(d) $\alpha(\xi)$ and $\beta(\eta)$ in (c)

Fig. 4-2 Example III – algebraic grid generation after the functional optimization

Fig. 4-2 shows an example of the grid (in $u-v$ plane) generated before and after the functional optimization when choosing $n=m=4$. At the beginning of the functional optimization, we initially choose $\alpha(\xi)=1-\xi$ and $\beta(\eta)=1-\eta$, which are straight lines (see Fig. 4-2b). After the functional optimization, they become polynomial curves of degree four (Fig. 4-2c). As demonstrated in the figure, the new non-linear blending functions successfully eliminate the self-overlapping.

5. Two-parameter Blending Function

In our above solution, the blending function between $P_0(\eta)$ and $P_1(\eta)$, $\alpha(\xi)$, is independent of η and the blending function between $Q_0(\xi)$ and $Q_1(\xi)$, $\beta(\eta)$, is independent of ξ, which makes the Coons patch not flexible enough to the given boundary curves. In this section, we explore the feasibility of dedicating the two blending functions as functions of both parameters, i.e., they become $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$. Accordingly, the equation of the planar Coons patch $X(\xi,\eta)$ changes to

$$\begin{aligned} X(\xi,\eta) = &\begin{bmatrix}\alpha(\xi,\eta) & 1-\alpha(\xi,\eta)\end{bmatrix}\begin{bmatrix}P_0(\eta)\\ P_1(\eta)\end{bmatrix} + \begin{bmatrix}Q_0(\xi) & Q_1(\xi)\end{bmatrix}\begin{bmatrix}\beta(\xi,\eta)\\ 1-\beta(\xi,\eta)\end{bmatrix} \\ &- \begin{bmatrix}\alpha(\xi,\eta) & 1-\alpha(\xi,\eta)\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\beta(\xi,\eta)\\ 1-\beta(\xi,\eta)\end{bmatrix} \end{aligned}, \tag{5-1}$$

where $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ should also satisfy the constraints of Coons patch:

$$\alpha(0,\eta)=1,\ \alpha(1,\eta)=0,\ \beta(\xi,0)=1, \text{ and } \beta(\xi,1)=0\ \ ((\xi,\eta)\in[0,1]\times[0,1]).$$

Here, we express $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ as

$$\alpha(\xi,\eta)=\sum_{i=0}^{m}\sum_{j=0}^{n}a_{i,j}\mathrm{B}_{i,m}(\xi)\mathrm{B}_{j,n}(\eta), \tag{5-2}$$

$$\beta(\xi,\eta)=\sum_{i=0}^{m}\sum_{j=0}^{n}b_{i,j}\mathrm{B}_{i,m}(\xi)\mathrm{B}_{j,n}(\eta), \tag{5-3}$$

where $\mathrm{B}_{i,m}(\xi)$ and $\mathrm{B}_{j,n}(\eta)$ are Bernstein basis functions of degree m and n – $\mathrm{B}_{j,k}(t)=\binom{j}{k}t^{j}(1-t)^{k-j}$. To satisfy the constraints of Coons patch for $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$, by the properties of Bernstein basis function [1], we have $a_{0,j}\equiv 1$, $a_{m,j}\equiv 0$, $b_{i,0}\equiv 1$, and $b_{i,n}\equiv 0$. Thus, $a_{i,j}$ ($i=1,\cdots,m-1$ and $j=0,\cdots,n$) and $b_{i,j}$ ($i=0,\cdots,m$ and $j=1,\cdots,n-1$) are variables to be determined.

Recall the objective function

$$A=\iint_{\Psi}\mathrm{H}[-(X_{\xi_u}X_{\eta_v}-X_{\eta_u}X_{\xi_v})]d\xi d\eta,$$

during the optimization, the following formulas of partial derivatives of $X(\xi,\eta)$ are needed.

$$\begin{aligned} X_{\xi}(\xi,\eta) = &\begin{bmatrix}\alpha_{\xi}(\xi,\eta) & -\alpha_{\xi}(\xi,\eta)\end{bmatrix}\begin{bmatrix}P_0(\eta)\\ P_1(\eta)\end{bmatrix} + \begin{bmatrix}Q_0'(\xi) & Q_1'(\xi)\end{bmatrix}\begin{bmatrix}\beta(\xi,\eta)\\ 1-\beta(\xi,\eta)\end{bmatrix} + \begin{bmatrix}Q_0(\xi) & Q_1(\xi)\end{bmatrix}\begin{bmatrix}\beta_{\xi}(\xi,\eta)\\ -\beta_{\xi}(\xi,\eta)\end{bmatrix} \\ &- \begin{bmatrix}\alpha_{\xi}(\xi,\eta) & -\alpha_{\xi}(\xi,\eta)\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\beta(\xi,\eta)\\ 1-\beta(\xi,\eta)\end{bmatrix} - \begin{bmatrix}\alpha(\xi,\eta) & 1-\alpha(\xi,\eta)\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\beta_{\xi}(\xi,\eta)\\ -\beta_{\xi}(\xi,\eta)\end{bmatrix} \end{aligned} \tag{5-4}$$

$$X_\eta(\xi,\eta)=\begin{bmatrix}\alpha(\xi,\eta) & 1-\alpha(\xi,\eta)\end{bmatrix}\begin{bmatrix}P_0'(\eta)\\ P_1'(\eta)\end{bmatrix}+\begin{bmatrix}\alpha_\eta(\xi,\eta) & -\alpha_\eta(\xi,\eta)\end{bmatrix}\begin{bmatrix}P_0(\eta)\\ P_1(\eta)\end{bmatrix}+\begin{bmatrix}Q_0(\xi) & Q_1(\xi)\end{bmatrix}\begin{bmatrix}\beta_\eta(\xi,\eta)\\ -\beta_\eta(\xi,\eta)\end{bmatrix}$$
$$-\begin{bmatrix}\alpha_\eta(\xi,\eta) & -\alpha_\eta(\xi,\eta)\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\beta(\xi,\eta)\\ 1-\beta(\xi,\eta)\end{bmatrix}-\begin{bmatrix}\alpha(\xi,\eta) & 1-\alpha(\xi,\eta)\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\beta_\eta(\xi,\eta)\\ -\beta_\eta(\xi,\eta)\end{bmatrix} \tag{5-5}$$

where

$$\begin{aligned}\alpha_\xi(\xi,\eta)&=\sum_{i=0}^{m}\sum_{j=0}^{n}a_{i,j}\mathrm{B}'_{i,m}(\xi)\mathrm{B}_{j,n}(\eta)\\ \alpha_\eta(\xi,\eta)&=\sum_{i=0}^{m}\sum_{j=0}^{n}a_{i,j}\mathrm{B}_{i,m}(\xi)\mathrm{B}'_{j,n}(\eta)\\ \beta_\xi(\xi,\eta)&=\sum_{i=0}^{m}\sum_{j=0}^{n}b_{i,j}\mathrm{B}'_{i,m}(\xi)\mathrm{B}_{j,n}(\eta)\\ \beta_\eta(\xi,\eta)&=\sum_{i=0}^{m}\sum_{j=0}^{n}b_{i,j}\mathrm{B}_{i,m}(\xi)\mathrm{B}'_{j,n}(\eta)\end{aligned} \tag{5-6}$$

can be obtained from (5-2) and (5-3), and $\mathrm{B}'_{j,n}(t)=\frac{d}{dt}\mathrm{B}_{j,n}(t)=n(\mathrm{B}_{j-1,n-1}(t)-\mathrm{B}_{j,n-1}(t))$. By the objective function, the gradient direction of the solution vector is determined by

$$\frac{\partial A}{\partial a_{i,j}}=\iint_\Psi \delta[-(X_{\xi_u}X_{\eta_v}-X_{\eta_u}X_{\xi_v})](\frac{\partial X_{\eta_u}}{\partial a_{i,j}}X_{\xi_v}+\frac{\partial X_{\xi_v}}{\partial a_{i,j}}X_{\eta_u}-\frac{\partial X_{\xi_u}}{\partial a_{i,j}}X_{\eta_v}-\frac{\partial X_{\eta_v}}{\partial a_{i,j}}X_{\xi_u})d\xi d\eta\,, \tag{5-7}$$

$$\frac{\partial A}{\partial b_{i,j}}=\iint_\Psi \delta[-(X_{\xi_u}X_{\eta_v}-X_{\eta_u}X_{\xi_v})](\frac{\partial X_{\eta_u}}{\partial b_{i,j}}X_{\xi_v}+\frac{\partial X_{\xi_v}}{\partial b_{i,j}}X_{\eta_u}-\frac{\partial X_{\xi_u}}{\partial b_{i,j}}X_{\eta_v}-\frac{\partial X_{\eta_v}}{\partial b_{i,j}}X_{\xi_u})d\xi d\eta\,. \tag{5-8}$$

Similar to the single parameter case, to compute the equations (5-7) and (5-8), we need to know $\frac{\partial X_\xi}{\partial a_{i,j}}$, $\frac{\partial X_\eta}{\partial a_{i,j}}$, $\frac{\partial X_\xi}{\partial b_{i,j}}$, and $\frac{\partial X_\eta}{\partial b_{i,j}}$. They are derived from (5-4) and (5-5) as

$$\begin{aligned}\frac{\partial X_\xi}{\partial a_{i,j}}&=(P_0(\eta)-P_1(\eta))\frac{\partial\alpha_\xi(\xi,\eta)}{\partial a_{i,j}}\\ &\quad-[(Q_1(0)-Q_1(1))+(Q_0(0)+Q_1(1)-Q_1(0)-Q_0(1))\beta(\xi,\eta)]\frac{\partial\alpha_\xi(\xi,\eta)}{\partial a_{i,j}}\\ &\quad-(Q_0(0)+Q_1(1)-Q_1(0)-Q_0(1))\beta_\xi(\xi,\eta)\frac{\partial\alpha(\xi,\eta)}{\partial a_{i,j}}\end{aligned} \tag{5-9}$$

$$\begin{aligned}\frac{\partial X_\eta}{\partial a_{i,j}}&=(P_0'(\eta)-P_1'(\eta))\frac{\partial\alpha(\xi,\eta)}{\partial a_{i,j}}+(P_0(\eta)-P_1(\eta))\frac{\partial\alpha_\eta(\xi,\eta)}{\partial a_{i,j}}\\ &\quad-(Q_0(0)+Q_1(1)-Q_1(0)-Q_0(1))\beta_\eta(\xi,\eta)\frac{\partial\alpha(\xi,\eta)}{\partial a_{i,j}}\\ &\quad-[(Q_1(0)-Q_1(1))+(Q_0(0)+Q_1(1)-Q_1(0)-Q_0(1))\beta(\xi,\eta)]\frac{\partial\alpha_\eta(\xi,\eta)}{\partial a_{i,j}}\end{aligned} \tag{5-10}$$

$$\frac{\partial X_\xi}{\partial b_{i,j}} = (Q_0'(\xi) - Q_1'(\xi))\frac{\partial \beta(\xi,\eta)}{\partial b_{i,j}} + (Q_0(\xi) - Q_1(\xi))\frac{\partial \beta_u(\xi,\eta)}{\partial b_{i,j}}$$
$$- (Q_0(0) + Q_1(1) - Q_1(0) - Q_0(1))\alpha_u(\xi,\eta)\frac{\partial \beta(\xi,\eta)}{\partial b_{i,j}} \tag{5-11}$$
$$- [(Q_0(1) - Q_1(1)) + (Q_0(0) + Q_1(1) - Q_1(0) - Q_0(1))\alpha(\xi,\eta)]\frac{\partial \beta_u(\xi,\eta)}{\partial b_{i,j}}$$

$$\frac{\partial X_\eta}{\partial b_{i,j}} = (Q_0(\xi) - Q_1(\xi))\frac{\partial \beta_\eta(\xi,\eta)}{\partial b_{i,j}}$$
$$- (Q_0(0) + Q_1(1) - Q_1(0) - Q_0(1))\alpha_\eta(\xi,\eta)\frac{\partial \beta(\xi,\eta)}{\partial b_{i,j}} \tag{5-12}$$
$$- [(Q_0(1) - Q_1(1)) + (Q_0(0) + Q_1(1) - Q_1(0) - Q_0(1))\alpha(\xi,v)]\frac{\partial \beta_\eta(\xi,\eta)}{\partial b_{i,j}}$$

where

$$\frac{\partial \alpha(\xi,\eta)}{\partial a_{i,j}} = \mathrm{B}_{i,m}(\xi)\mathrm{B}_{j,n}(\eta),\ \frac{\partial \beta(\xi,\eta)}{\partial b_{i,j}} = \mathrm{B}_{i,m}(\xi)\mathrm{B}_{j,n}(\eta),$$

$$\frac{\partial \alpha_\xi(\xi,\eta)}{\partial a_{i,j}} = \mathrm{B}'_{i,m}(\xi)\mathrm{B}_{j,n}(\eta),\ \frac{\partial \alpha_\eta(\xi,\eta)}{\partial a_{i,j}} = \mathrm{B}_{i,m}(\xi)\mathrm{B}'_{j,n}(\eta),$$

$$\frac{\partial \beta_\xi(\xi,\eta)}{\partial b_{i,j}} = \mathrm{B}'_{i,m}(\xi)\mathrm{B}_{j,n}(\eta),\ \frac{\partial \beta_\eta(\xi,\eta)}{\partial b_{i,j}} = \mathrm{B}_{i,m}(\xi)\mathrm{B}'_{j,n}(\eta).$$

Now we can compute the functional optimum of the two-parameter blending functions by the *Conjugate Gradient Method* using the same terminal condition as that of the optimization of the one-parameter blending functions – $A / \overline{A} < \mu$.

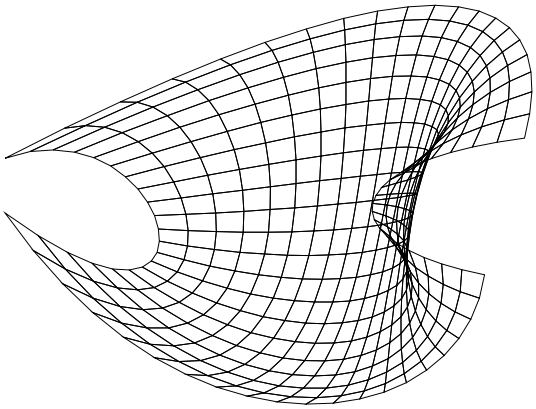
Fig. 5-1 depicts the grid generation result of Example III by optimizing the planar Coons patch of two-parameter blending functions ($n = m = 3$). We initially set

$$(a_{i,j})_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2/3 & 2/3 & 2/3 & 2/3 \\ 1/3 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } (b_{i,j})_{4\times4} = \begin{bmatrix} 1 & 2/3 & 1/3 & 0 \\ 1 & 2/3 & 1/3 & 0 \\ 1 & 2/3 & 1/3 & 0 \\ 1 & 2/3 & 1/3 & 0 \end{bmatrix},$$

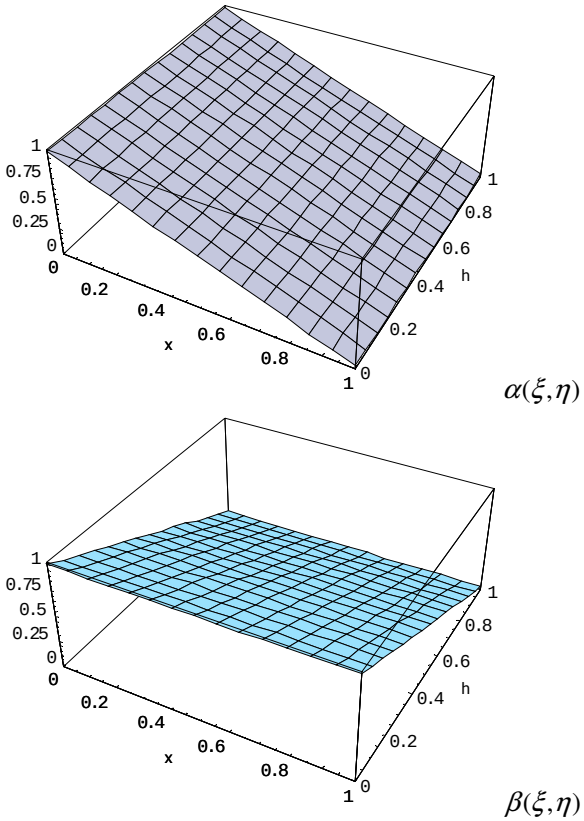
their related $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ are displayed in Fig. 5-1b. When the functional optimization is completed, they become

$$(a_{i,j})_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.4149 & 0.1328 & 0.0963 & 0.3666 \\ 0.0899 & -0.1294 & -0.5643 & -0.5712 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } (b_{i,j})_{4\times4} = \begin{bmatrix} 1 & 0.6612 & 0.3255 & 0 \\ 1 & 0.6146 & 0.2382 & 0 \\ 1 & 0.4619 & -0.0418 & 0 \\ 1 & 0.3446 & -0.1754 & 0 \end{bmatrix},$$

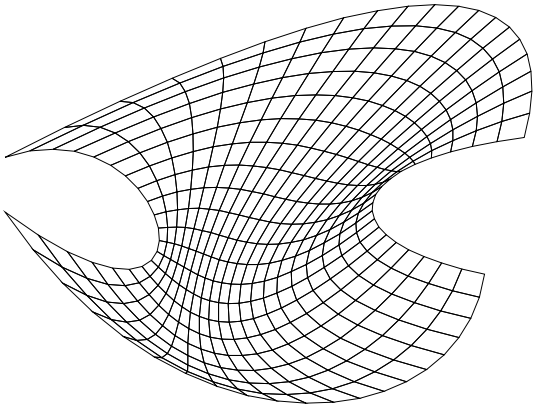
with which the final $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ are shown in Fig. 5-1d.



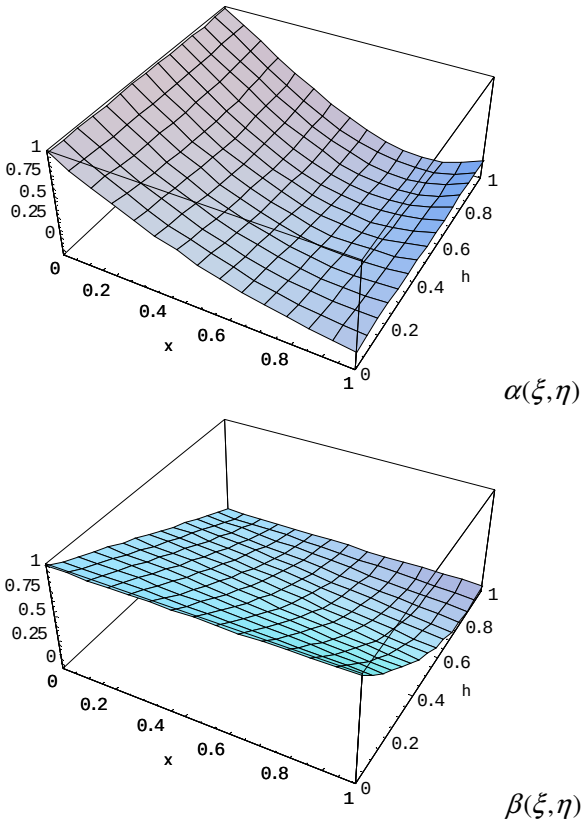
(a) before optimization



(b) $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ in (a)



(c) after optimization



(d) $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ in (c)

Fig. 5-1 Example III – functional optimization using two-parameter blending functions

6. Progressive Optimization

Because our objective function in the functional optimization is concave (allowing many local minima – it is not the form of our objective function that leads these local minima; they are tightly related to the given boundary curves), the success of the numerical optimization algorithm depends critically on the initial position of the solution vector. We usually take the bilinear interpolation as an initial position of the solution vector; it however may be a very unsatisfying one to some strongly concaved boundaries (e.g., the one shown in Fig. 6-1a). "Guessing" a good initial vector is hard. The basic idea we take to overcome this difficulty is to progressively achieve the functional optimum by gradually deforming a rectangle boundary region into the region Ω bounded by the given curves, which we discuss in detail in this section.

The initial shape, a rectangle, does not generate overlapping with a bilinear Coons patch $X^0(\xi,\eta)$. When deforming the rectangle into the final shape Ω by a linear interpolation, we check if overlap occurs and, once it is detected, the numerical optimization method presented is applied to $X^0(\xi,\eta)$ to determine a new Coons patch mapping $X^1(\xi,\eta)$ without self-overlap; the deformation will then continue. The deformation and functional optimization are applied alternatively on $X^i(\xi,\eta)$ (where i represents the Coons patch mapping determined after i times of the numerical optimization) until the final non-self-overlapping planar Coons patch $X^f(\xi,\eta)$ of Ω is obtained. We use the box $\{R_{00}, R_{01}, R_{11}, R_{10}\}$ of Ω as the initial rectangle region, where $R_{00}, R_{01}, R_{11}, R_{10}$ are its four corner points in anti-clockwise sense. Thus, its four boundary curves are

$$\begin{aligned} Q_{R0}(\xi) &= \xi R_{01} + (1-\xi)R_{00} \\ Q_{R1}(\xi) &= \xi R_{11} + (1-\xi)R_{10} \\ P_{R0}(\eta) &= \eta R_{10} + (1-\eta)R_{00} \\ P_{R1}(\eta) &= \eta R_{11} + (1-\eta)R_{01} \end{aligned} \tag{6-1}$$

where $Q_{R0}(\xi)$ - $Q_0(\xi)$, $Q_{R1}(\xi)$ - $Q_1(\xi)$, $P_{R0}(\eta)$ - $P_0(\eta)$, and $P_{R1}(\eta)$ - $P_1(\eta)$ are related. Therefore, the four boundary curves of $X^i(\xi,\eta)$ are

$$\begin{aligned} Q_0^i(\xi) &= \lambda_i Q_0(\xi) + (1-\lambda_i)(\xi R_{01} + (1-\xi)R_{00}) \\ Q_1^i(\xi) &= \lambda_i Q_1(\xi) + (1-\lambda_i)(\xi R_{11} + (1-\xi)R_{10}) \\ P_0^i(\eta) &= \lambda_i P_0(\eta) + (1-\lambda_i)(\eta R_{10} + (1-\eta)R_{00}) \\ P_1^i(\eta) &= \lambda_i P_1(\eta) + (1-\lambda_i)(\eta R_{11} + (1-\eta)R_{01}) \end{aligned} \tag{6-2}$$

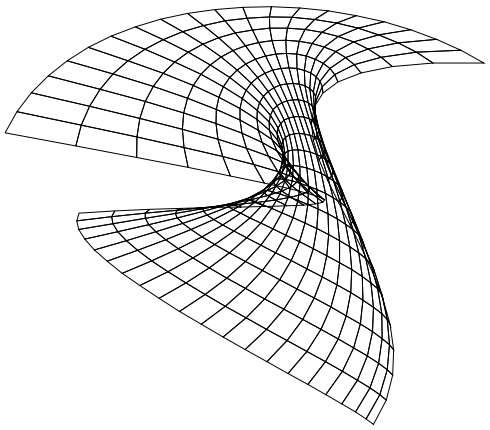
where λ_i is the deformation factor, or the *time*, when the functional optimization is applied to $X^i(\xi,\eta)$ ($\lambda_i \in [0,1]$). At the beginning of the deformation, $\lambda_i = 0$; after the deformation is completed, $\lambda_i = 1$. Also, we can get

$$\begin{aligned}
\frac{\partial}{\partial \xi} Q_0^i(\xi) &= \lambda_i Q_0'(\xi) + (1-\lambda_i)(R_{01} - R_{00}) \\
\frac{\partial}{\partial \xi} Q_1^i(\xi) &= \lambda_i Q_1'(\xi) + (1-\lambda_i)(R_{11} - R_{10}) \\
\frac{\partial}{\partial \eta} P_0^i(\eta) &= \lambda_i P_0'(\eta) + (1-\lambda_i)(R_{10} - R_{00}) \\
\frac{\partial}{\partial \eta} P_1^i(\eta) &= \lambda_i P_1'(\eta) + (1-\lambda_i)(R_{11} - R_{01})
\end{aligned} . \tag{6-3}$$

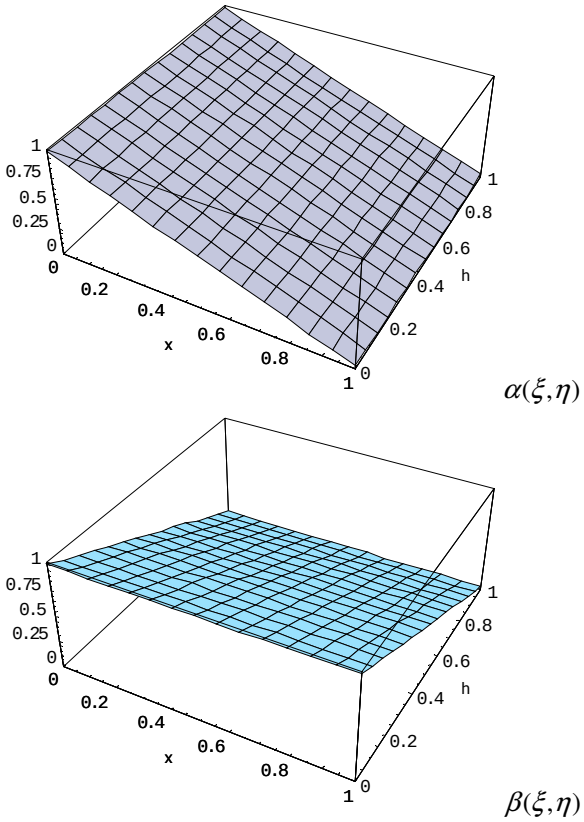
Using $Q_0^i(\xi)$, $Q_1^i(\xi)$, $P_0^i(\eta)$, $P_1^i(\eta)$, $\frac{\partial}{\partial \xi} Q_0^i(\xi)$, $\frac{\partial}{\partial \xi} Q_1^i(\xi)$, $\frac{\partial}{\partial \eta} P_0^i(\eta)$, and $\frac{\partial}{\partial \eta} P_1^i(\eta)$ in (6-2) and (6-3) to take place of $Q_0(\xi)$, $Q_1(\xi)$, $P_0(\eta)$, $P_1(\eta)$, $Q_0'(\xi)$, $Q_1'(\xi)$, $P_0'(\eta)$, and $P_1'(\eta)$ in the equations of section 5, the functional optimum of $X^i(\xi,\eta)$ can be determined by the same method.

During the deformation, the deformation factor increases from zero to one adaptively to the value increase of the objective function. The overall procedure of progressive optimization is given in pseudo-code in ***Algorithm*** ProgressiveOptimization() below. In Example IV (shown in Fig. 6-1), the non-self-overlapping planar Coons patch cannot be obtained by the pure numerical optimization even after iterating 10000 times; using ***Algorithm*** ProgressiveOptimization(), we obtain a final result without self-overlapping by only 30 applications of the numerical optimization routine – in each time, less than 10 iterations are needed. The progressive results are shown in Fig. 6-2.

Algorithm ProgressiveOptimization (Ω)
Input: A region Ω in the $u-v$ parametric space.
Output: The non-self-overlap planar Coons patch $X^f(\xi,\eta)$ on Ω.
1. Compute the bounding box of Ω, and use it to generate $Q_{R0}(\xi)$, $Q_{R1}(\xi)$, $P_{R0}(\eta)$, and $P_{R1}(\eta)$;
2. Build $X^0(\xi,\eta)$ on $Q_{R0}(\xi)$, $Q_{R1}(\xi)$, $P_{R0}(\eta)$, and $P_{R1}(\eta)$ as a bilinear planar Coons patch;
3. $i \leftarrow 1$ and $\lambda_0 \leftarrow 0$;
4. **do**{
5. $\Delta\lambda \leftarrow 2(1-\lambda_{i-1})$;
6. **do**{
7. $\Delta\lambda \leftarrow \Delta\lambda/2$, and $\lambda_i \leftarrow \lambda_{i-1} + \Delta\lambda$;
8. Change the shape of $X^i(\xi,\eta)$ by λ_i, and compute the objective function A of $X^i(\xi,\eta)$;
9. Compute the area $\overline{A}$ of $X^i(\xi,\eta)$;
10. }**while**($(A/\overline{A}) > \tau$);
11. Compute the numerical optimum of blending functions in $X^i(\xi,\eta)$;
12. $i \leftarrow i+1$;
13. }**while**($\lambda_{i-1} < 1$);
14. $X^f(\xi,\eta) \leftarrow X^{i-1}(\xi,\eta)$;
15. **return** $X^f(\xi,\eta)$;
(* in our testing, we choose $\tau = 0.5\%$)



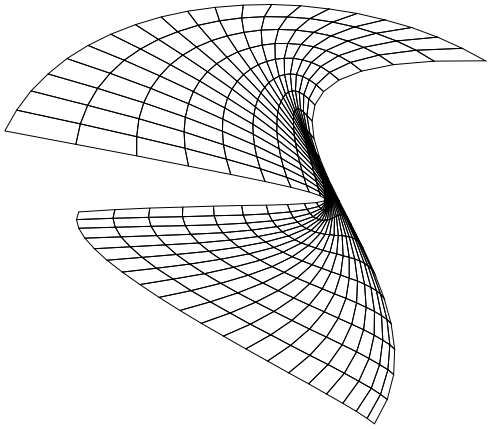
(a) before optimization



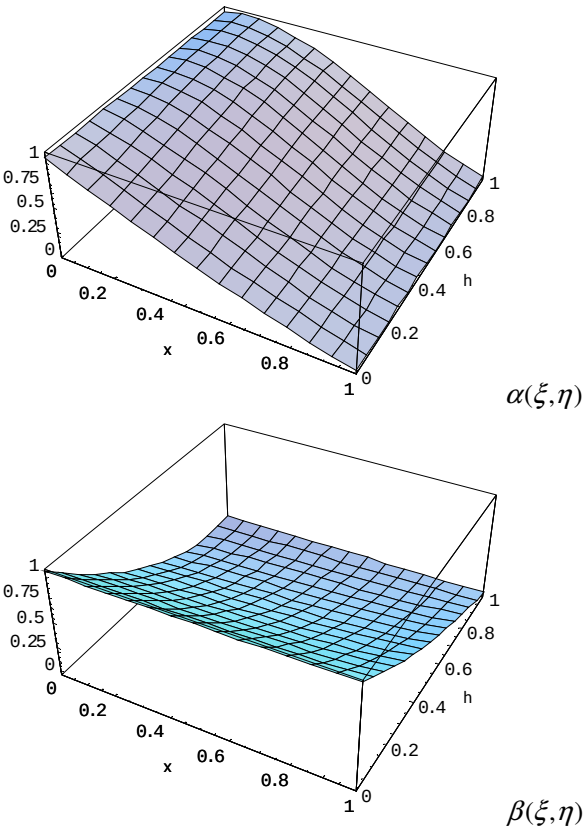
$\alpha(\xi,\eta)$

$\beta(\xi,\eta)$

(b) $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ in (a)



(c) final result

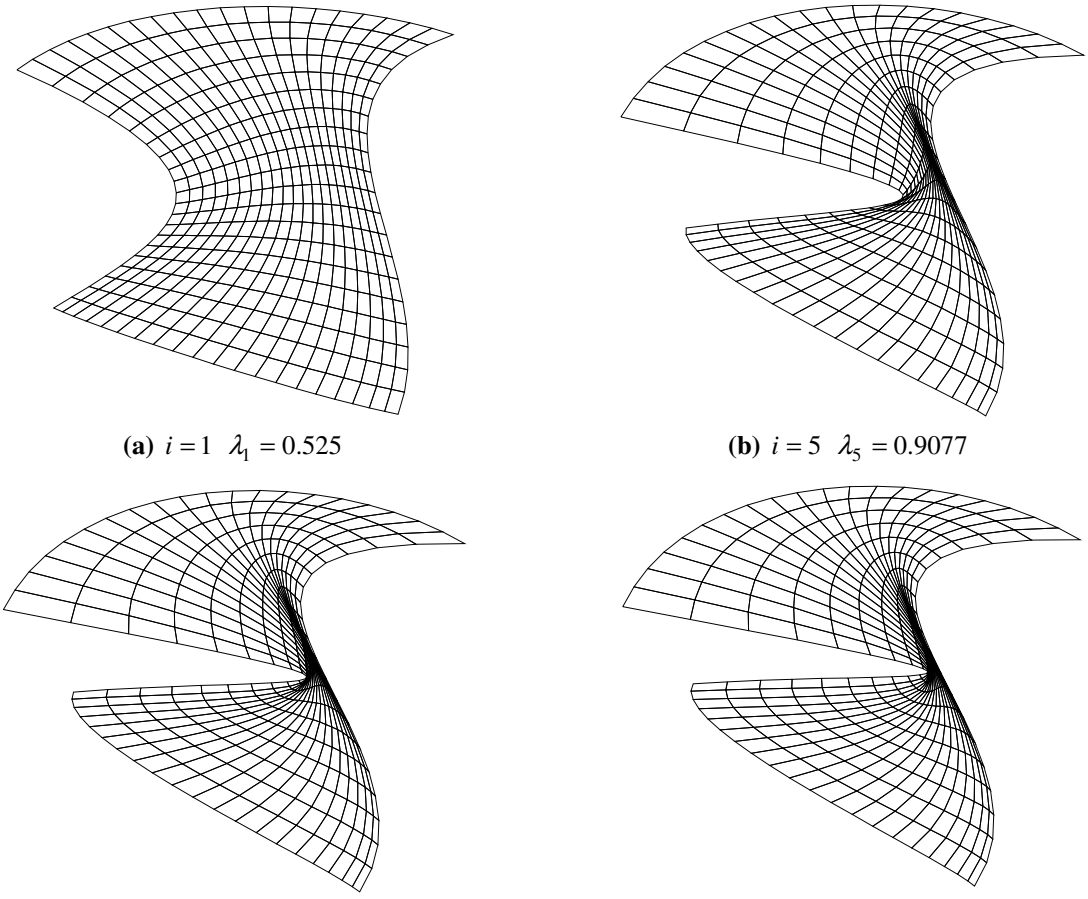


$\alpha(\xi,\eta)$

$\beta(\xi,\eta)$

(d) $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ in (c)

Fig. 6-1 Example IV – result of progressive optimization ($n = m = 3$)



(a) $i = 1 \quad \lambda_1 = 0.525$ **(b)** $i = 5 \quad \lambda_5 = 0.9077$

(c) $i = 16 \quad \lambda_{16} = 0.9853$ **(d)** $i = 30 \quad \lambda_{30} = 1.0$

Fig. 6-2 Example IV – progressive results

7. Extension of Shape Control

In the previous parts of this paper, the objective function of optimization (4-3) does *not* consider the shape control of each element. Actually, it is possible to add a shape control term in the objective function. The basic idea that if $X_\xi \cdot X_\eta \to 0$ was maintained all over the patch, the shape of every element in the grid could be guaranteed in the $\xi - \eta$ plane. Thus, we add a new term

$$B = \iint_\Psi (X_{\xi_u} X_{\eta_u} + X_{\xi_v} X_{\eta_v})^2 d\xi d\eta \,, \tag{7-1}$$

to the objective function. The modified objective function is

$$J = w_A A + w_B B \tag{7-2}$$

where A is as given in (4-3), and w_A, w_B are the factors to balance the weight between the term *A* and the term *B*. When the term B is minimized to zero, the $X_\xi \cdot X_\eta \to 0$ is guaranteed all over the patch.

In order to minimize the objective function, we need to have its gradients with respect to the solution vector. By equation (7-1), we have

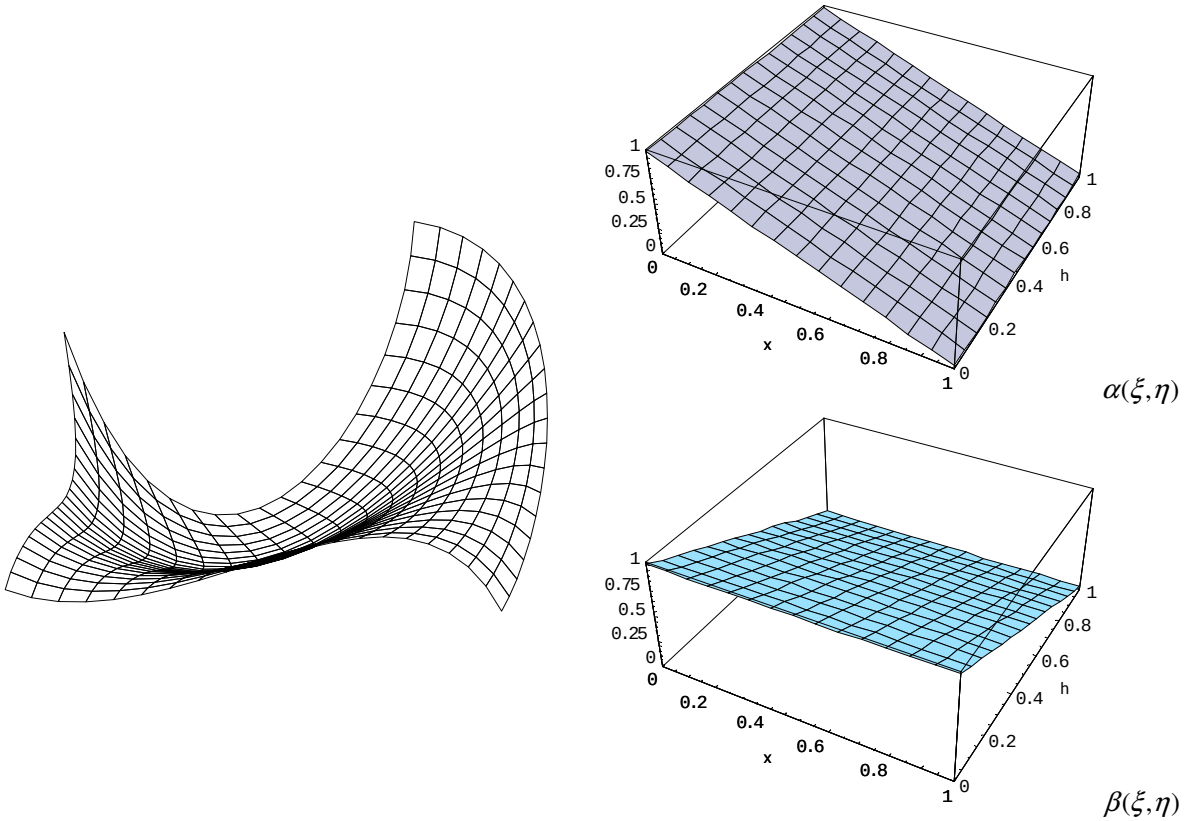
$$\frac{\partial B}{\partial a_{i,j}} = \iint_{\Psi} 2(X_{\xi_u} X_{\eta_u} + X_{\xi_v} X_{\eta_v})(\frac{\partial X_{\xi_u}}{\partial a_{i,j}} X_{\eta_u} + \frac{\partial X_{\eta_u}}{\partial a_{i,j}} X_{\xi_u} + \frac{\partial X_{\xi_v}}{\partial a_{i,j}} X_{\eta_v} + \frac{\partial X_{\eta_v}}{\partial a_{i,j}} X_{\xi_v}) d\xi d\eta \,, \tag{7-3}$$

$$\frac{\partial B}{\partial b_{i,j}} = \iint_{\Psi} 2(X_{\xi_u} X_{\eta_u} + X_{\xi_v} X_{\eta_v})(\frac{\partial X_{\xi_u}}{\partial b_{i,j}} X_{\eta_u} + \frac{\partial X_{\eta_u}}{\partial b_{i,j}} X_{\xi_u} + \frac{\partial X_{\xi_v}}{\partial b_{i,j}} X_{\eta_v} + \frac{\partial X_{\eta_v}}{\partial b_{i,j}} X_{\xi_v}) d\xi d\eta \,. \tag{7-4}$$

Thus, together with equations (5-7) and (5-8), the gradients of J with respect to $a_{i,j}$ and $b_{i,j}$ are computed by

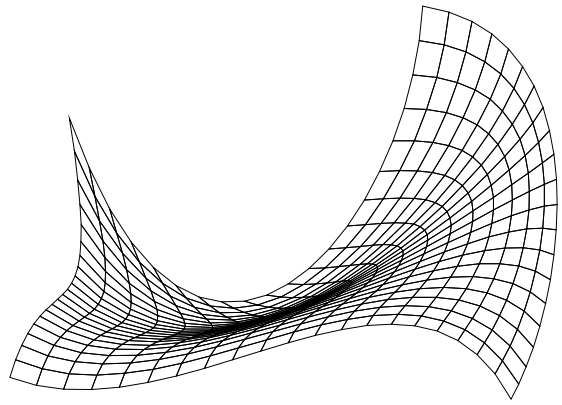
$$\frac{\partial J}{\partial a_{i,j}} = w_A \frac{\partial A}{\partial a_{i,j}} + w_B \frac{\partial B}{\partial a_{i,j}} \text{ and } \frac{\partial J}{\partial b_{i,j}} = w_A \frac{\partial A}{\partial b_{i,j}} + w_B \frac{\partial B}{\partial b_{i,j}} \,. \tag{7-5}$$

The comparison of the non-self-overlapping Coons patch generation with and without shape control is given in the following figures. Fig. 7-1 shows the bilinear Coons patch before optimization. Obviously, the patch is self-overlapped. Fig. 7-2a gives the optimized patch with weights $w_A = 1.0$ and $w_B = 0$ – the shape control is not cared (the result blending functions without shape control are shown in Fig. 7-2b). The mesh is stretched in the middle. Fig. 7-2c and 7-2d illustrate the result with shape control by changing w_B from 0 to 10^{-13}, which leads to less stretch inside the Coons patch.

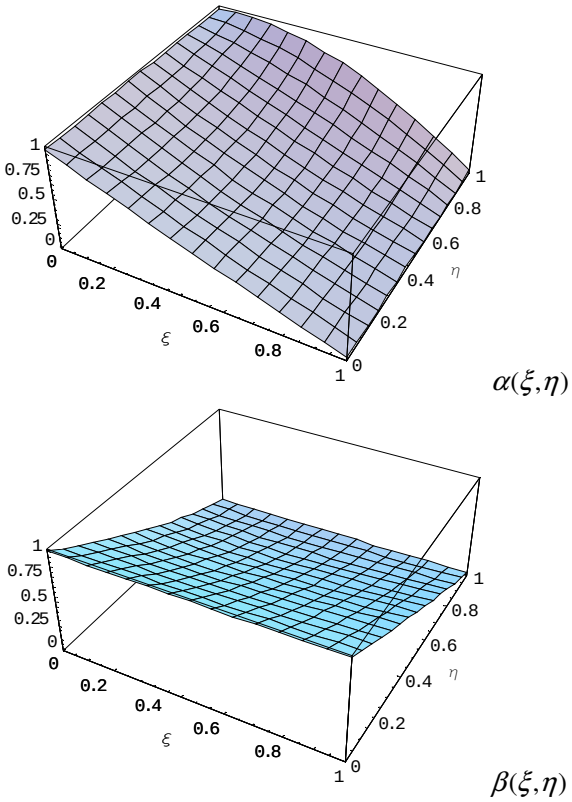


(a) mesh before optimization **(b) $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ in (a)**

Fig. 7-1 Example V – before optimization



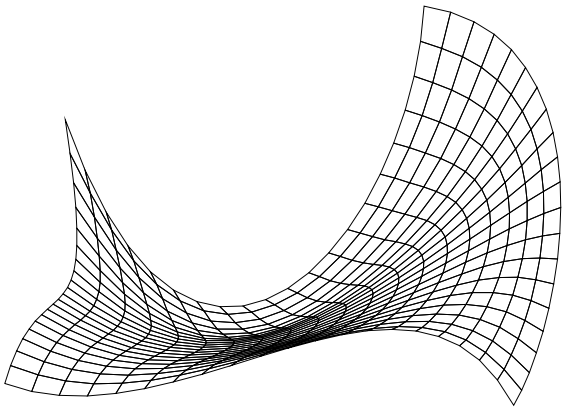
(c) after optimization without shape control
($w_A = 1.0$ **and** $w_B = 0$)



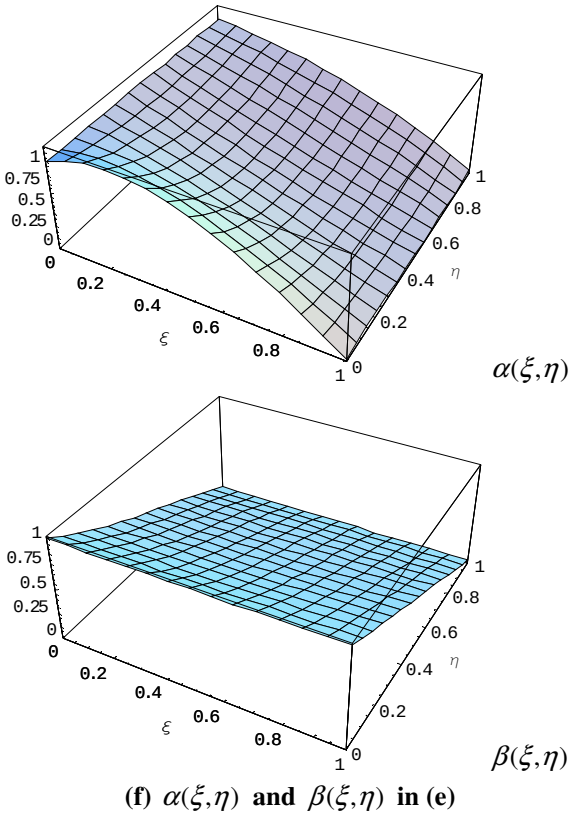
$\alpha(\xi,\eta)$

$\beta(\xi,\eta)$

(d) $\alpha(\xi,\eta)$ **and** $\beta(\xi,\eta)$ **in (c)**



(e) after optimization with shape control
($w_A = 1.0$ **and** $w_B = 10^{-13}$)



$\alpha(\xi,\eta)$

$\beta(\xi,\eta)$

(f) $\alpha(\xi,\eta)$ **and** $\beta(\xi,\eta)$ **in (e)**

Fig. 7-2 Example V – optimization with vs. without shape control

This shape control idea can be further extended to control the shape of elements on the given parametric surface. If $S_\xi \cdot S_\eta \to 0$ was maintained all over the patch, the shape of every element in the grid could be guaranteed on the parametric surface S. The term B of the objective function could be modified to

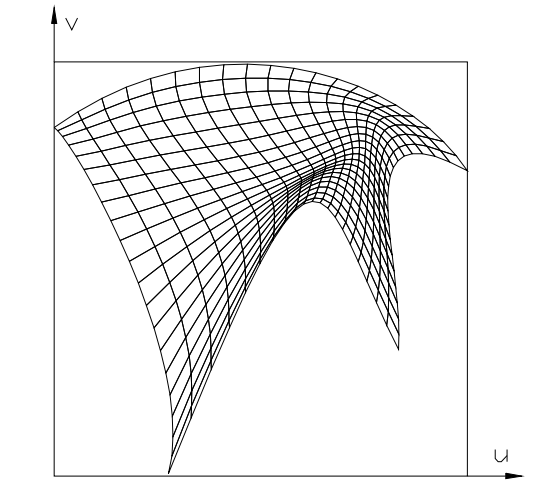
$$B^* = \iint_\Psi (S_{\xi_x} S_{\eta_x} + S_{\xi_y} S_{\eta_y} + S_{\xi_z} S_{\eta_z})^2 \, d\xi d\eta$$

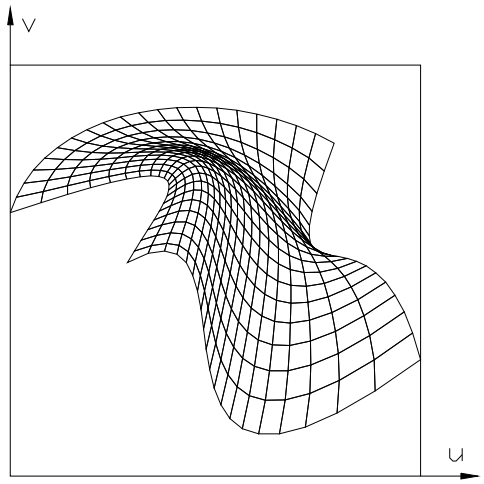
to achieve this condition. Therefore, the new objective function might be $J^* = w_A A + w_S S$.

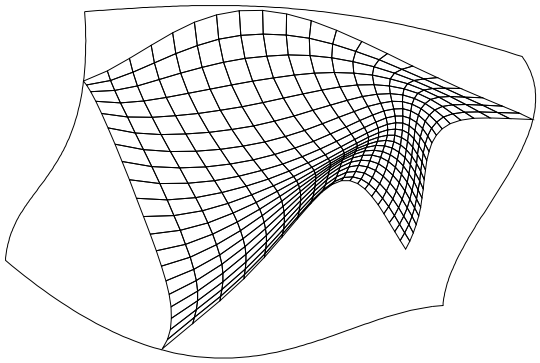
8. Experimental Results and Discussion

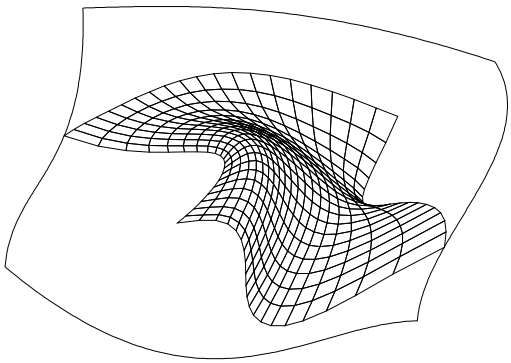
Fig. 8-1 gives the algebraic grid generation results of trimmed surfaces in Example I and II (initially given in Fig. 1-2), and their final blending functions ($n = m = 3$) of the non-self-overlapping planar Coons patch are also shown. When the shape of the trimmed surface in the parametric space is very convoluted (e.g., an *n*-sided or with a hole – such as Example V shown in Fig. 8-2a), the given region is subdivided into several 4-sided sub-patches (Fig. 8-2b), where all patches satisfy *Assumption 3.1*; then the approach presented in this paper can be applied to generate the final algebraic grids (Fig. 8-2c and 8-2d show the result of example V).

When applying the functional optimization to obtain the algebraic equation of the Coons patch, we should choose the order of blending functions carefully. If the order of blending functions is lower than the order of the given four boundary curves, the functional optimization may not be convergent; however, higher order polynomials tend to make the final shape of the blending functions vibrate violently. Thus, we usually let the order of blending functions to be the same as that of the boundary curves or just one order higher.

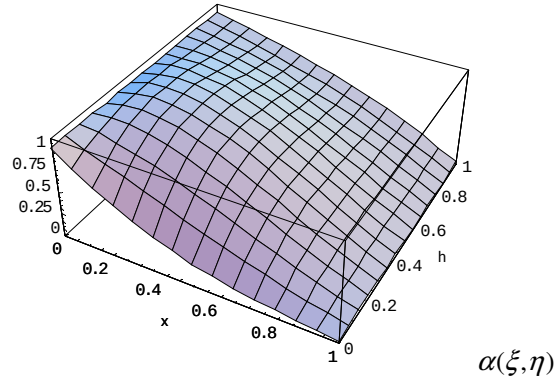




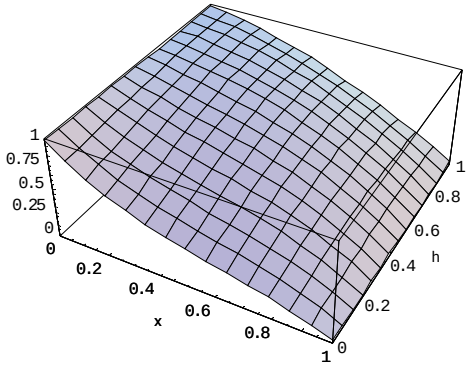




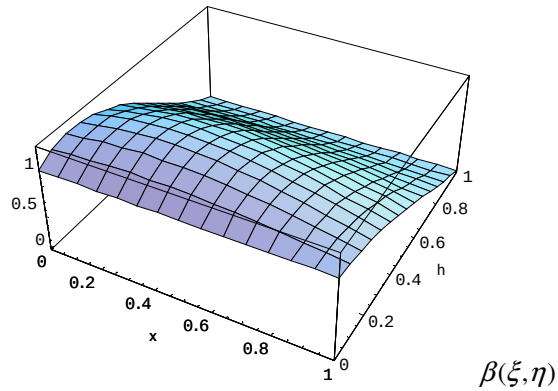
(a) result of Example I **(b) result of Example II**



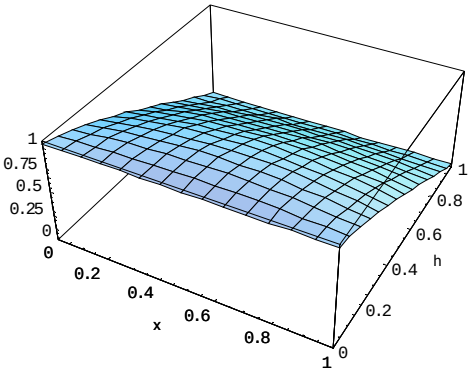
$\alpha(\xi,\eta)$



$\alpha(\xi,\eta)$



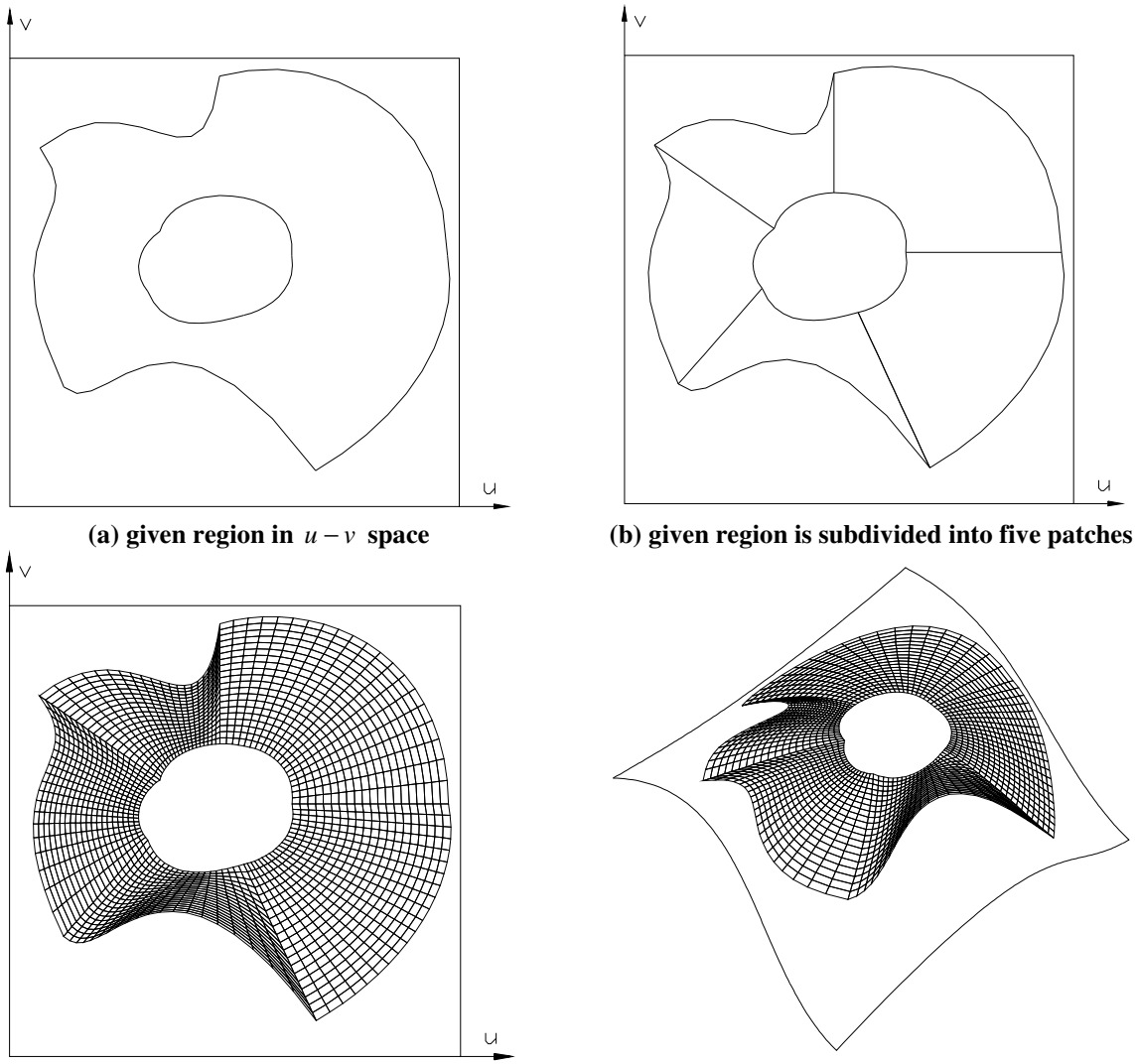
$\beta(\xi,\eta)$



$\beta(\xi,\eta)$

(c) $\alpha(\xi,\eta)$ **and** $\beta(\xi,\eta)$ **of Example I** **(d)** $\alpha(\xi,\eta)$ **and** $\beta(\xi,\eta)$ **of Example II**

Fig. 8-1 Algebraic grid generation results of Example I and II



(a) given region in $u-v$ space **(b) given region is subdivided into five patches**

(c) final grids in $u-v$ space **(d) final grids on the trimmed surface**

Fig. 8-2 Example VI – complex region with multiple patches

9. Conclusion

The ability to generate a grid system in a 4-sided region bounded by parametric boundary curves of any form with only C^1 continuity is a significant advantage of the Coons patch method over other algebraic methods for building a structured grid. This not only averts the singularity of elliptic PDE methods when only C^1 continuous boundary available, but also avoids the error generated when converting generic parametric C^1 boundary curves into curves with a specified representation form. However, the self-overlapping phenomenon frequently encountered in Coons patch mapping has been perplexing both theoreticians and practitioners for a long time and limiting its usage in many scientific and engineering practices. In this paper, we present an algorithm that uses the functional optimization method to determine the blending functions in a Coons patch so that the self-overlap can be prevented. Our initial test results are encouraging: in a variety of cases where self-

overlapping occurs if linear or naïve blending functions were used, the found blending functions after the functional optimization successfully avert the self-overlapping. A possible extension is also given to demonstrate how to control the shape of elements by adding a new term on the objective function. Coupled with any mesh smoothing methods [2], our algorithm provides a promising meshing tool in engineering.

The main disadvantage of our Coons patch method is the computing time. The time cost of examples in this paper is shown in Table 9-1. Usually, several minutes are needed. All tests are performed on a PIII 500 PC with a program written in Java. In the current implementation, we use a very primitive numerical method to compute the functional optimization. It is believed that with more efficient optimization algorithms and with the increasing processing power available on every desktop, the running time can be shortened significantly. Moreover, the following topics or improvements are worth future research:

- In our extension of element shape control, the shapes of elements are controlled in the $\xi - \eta$ plane; however, as described at the end of section 7, our experiment has already shown the possibility of controlling the shape of elements on the given trimmed parametric surface.
- Also, it will be interesting to see if the ideas of this paper can be extended into three-dimensional space to solve the algebraic grid generation problem in a given closed space using a Coons solid. Then, the governing equation may come from the same idea as Equation (3-1) by detecting some properties of the Coons solid in a virtual newly added axis direction.

Table 9-1 Time cost of examples

Example	Result Figures	Optimization Type	Blending Function Type	Time cost
I	8-1a 8-2c	Progressive	Two-parameter ($n = m = 3$)	3.4 min.
II	8-1b 8-1d	Progressive	Two-parameter ($n = m = 3$)	5.7 min.
III	4-2	Pure numerical	One-parameter ($n = m = 4$)	1.7 min.
	5-1	Pure numerical	Two-parameter ($n = m = 3$)	2.3 min.
IV	6-1	Progressive	Two-parameter ($n = m = 3$)	12.1 min.
V	7-1c 7-1d	Pure numerical	Two-parameter ($n = m = 3$)	3.7 min.
	7-1e 7-1f	Pure numerical	Two-parameter ($n = m = 3$)	4.1 min.
VI	8-2	Progressive	Two-parameter ($n = m = 3$)	4.5 min.

*The grid size of our numerical integration is 20×20 .

10. Reference

[1] Mortenson ME. *Geometric Modeling (2nd Edition)*. Wiley: New York, 1997.

[2] Thompson JF, Soni BK, Weatherill NP. *Handbook of Grid Generation*. CRC Press: Florida, 1999.

[3] Baehmann PL, Wittchen SL, Shephard MS, Grice KR, Yerry MA. Robust, geometrically based, automatic two-dimensional mesh generation. *International Journal for Numerical Methods in Engineering* 1987;24:1043-1078.

[4] Talbert JA, Parkinson AR. Development of an automatic, two-dimensional finite element mesh generation using quadrilateral elements and bezier curve boundary definition. *International Journal for Numerical Methods in Engineering* 1990; 29:1551-1567.

[5] Joe B. Quadrilateral mesh generation in polygonal regions. *Computer-Aided Design* 1995; 7:209-222.

[6] Josep S., Antonio H. Efficient unstructured quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 2000; 49:1327-1350.

[7] Lo SH. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering* 1985; 21:1403-1426.

[8] Blacker TD, Stephenson MB. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; 32:811-847.

[9] Cass RJ, Benzley SE, Mayers RJ, BlackerTD. Generalized 3-D Paving: An automated quadrilateral surface mesh generation algorithm. *International Journal for Numerical Methods in Engineering* 1996; 39:1475-1489.

[10] White DR, Kinney P. Redesign of the Paving algorithm: Robustness enhancements through element by element meshing. *Proceedings of the Sixth International Mesh Roundtable*, Park City, Utha; Oct. 1997, pp.323-335.

[11] Tam TKH, Armstrong CG. 2D Finite element mesh generation by medial axis subdivision. *Advances in Engineering Software & Workstations* 1991; 13:313-324.

[12] 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.

[13] Lau TS, Lo SH, Lee CK. Generation of quadrilateral mesh over analytical curved surfaces. *Finite Elements in Analysis & Design* 1997; 27:251-272.

[14] Borouchaki H., Frey PJ. Adaptive triangular-quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1998; 41:915-934.

[15] 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.

[16] 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.

[17] Itoh T, Shimada K. Automatic conversion of triangular meshes into quadrilateral meshes with directionality. *International Journal of CAD/CAM* 2001; 1:20-38.

[18] 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.

[19] 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.

[20] Smith RE, Eriksson LE. Algebraic grid generation. *Computer Methods in Applied Mechanics & Engineering* 1987; 64:285-300.

[21] 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.

[22] Knupp PM. Jacobian-weighted elliptic grid generation. *SIAM Journal on Scientific Computing* 1996; 17: 1475-1490.

[23] Khamayseh A, Hamann B. Elliptic grid generation using NURBS surfaces. *Computer Aided Geometric Design* 1996; 13:369-386.

[24] Soni BK. Elliptic grid generation system: control functions revisited. I. *Applied Mathematics & Computation* 1993; 59:151-163.

[25] Thompson JF. A survey of dynamically-adaptive grids in the numerical solution of partial differential equations. *Applied Numerical Mathematics* 1985; 1:3-27.

[26] Fomenko AT, Kunii TL. *Topological modeling for visualization*. Springer: Hong Kong, 1997.

[27] Press WH, Teukolsky SA, Vetterling WT, and Flannery BP. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 1992.

[28] Chan TF, Vese LA. Active contours without edges. *IEEE Transactions of Image Processing* 2001; 10: 266-277.

Appendix A

Lemma The Coons patch $X(\xi,\eta)=\begin{bmatrix} U(\xi,\eta) \\ V(\xi,\eta) \\ W(\xi,\eta) \end{bmatrix}$ is a graph-mapping if and only if there is no any singular point in the square $[0,1]\times[0,1]$.

Proof. Let us argument the $X(\xi,\eta)$ by $X^*(\xi,\eta)=\begin{bmatrix} U^*(\xi,\eta)=U(\xi,\eta) \\ V^*(\xi,\eta)=V(\xi,\eta) \\ W^*(\xi,\eta) \end{bmatrix}$ with $W^*(\xi,\eta)=a\xi+b\eta$ for some real number a and b. By properly choosing a and b, one can enforce $X^*(\xi,\eta)$ to have normal vector everywhere, thus it is a smooth regular surface in the $u\times v\times w$ space. Suppose first that $X(\xi,\eta)$ is not a graph-mapping. This means that there exist two distinct pairs $(\xi_0,\eta_0)\in[0,1]\times[0,1]$ and $(\xi_1,\eta_1)\in[0,1]\times[0,1]$, 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 $X^*(\xi,\eta)$ 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 σ^* of σ between the two points. Since σ^* 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 Fig. A-1a. 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 $X(\xi,\eta)$.

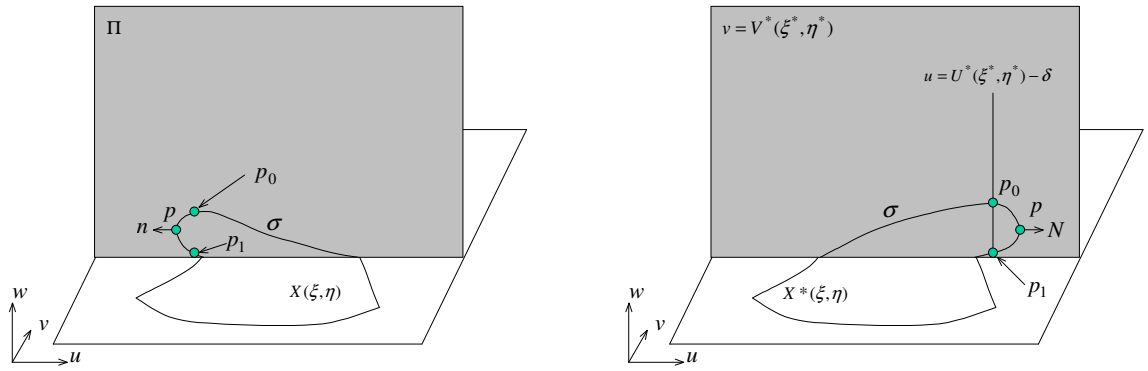
Conversely, let (ξ^*,η^*) be a singular point of $X(\xi,\eta)$; hence, $U_\xi(\xi^*,\eta^*)V_\eta(\xi^*,\eta^*)=U_\eta(\xi^*,\eta^*)V_\xi(\xi^*,\eta^*)$. Consequently, the normal N to the surface $X^*(\xi,\eta)$ at (ξ^*,η^*) is perpendicular to the w-axis. Without loss of generality, we can assume N is parallel to the u-axis. Intersecting $X^*(\xi,\eta)$ 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 (assuming $p*$ is a u-maximum point). Let

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

be two such intersection points for some $(\xi_0, \eta_0) \neq (\xi_1, \eta_1)$, as shown in Fig. A-1b. 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 $X(\xi, \eta)$ maps two distinct points in the $\xi - \eta$ domain to a same point in the region Ω_X. This completes the proof.

Q.E.D.



(a) **(b)**

Fig. A-1 Proof of Lemma