Non-self-overlapping Hermite interpolation mapping: a practical solution for structured quadrilateral meshing

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, N.T., Hong Kong
E-mail: mektang@ust.hk

Abstract

This paper addresses the problem of constructing a structured quadrilateral grid inside a given four-sided 2D region by a particular boundary-conforming mapping scheme – Hermite Interpolation Mapping (HIM). When the four given boundary curves are concave and convoluted, all boundary-conform mapping methods suffer from potential self-overlapping problem. Under HIM, the geometry of the grid depends on both the four boundary curves and the tangent vector functions associated with the curves. While the four boundary curves are fixed, the tangent functions in HIM can be varied to suit the need of controlling the characteristics of the mesh inside the given region so to prevent self-overlapping. Besides tangent functions, the four twist vectors at the corners of the region can also be adjusted to influence the distribution of the inner grid elements. In our approach, a constrained functional optimization scheme is adopted to adjust the tangent functions and the twist vectors, adaptive to the geometry of the boundary curves, so that the resulting HIM will be free of self-overlapping. The optimization is carried out on the shape control energy that measures the overall mesh quality of the underlying HIM while the self-overlapping is strongly prevented in the form of constraints to the optimization. Experimental results show the promise of the proposed method as a practical and effective solution for structured grid generation.

Keywords: structured grid, quadrilateral mesh, parametric space, self-overlap, and Hermite interpolation.

* Corresponding author

1. Introduction

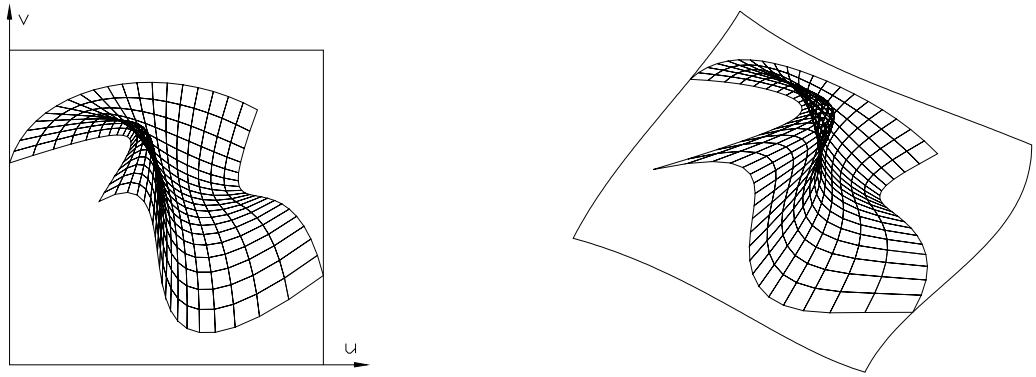
A task frequently encountered in the computer-aided engineering process is finding an aggregate of simple planar elements, e.g., triangles and quadrilaterals, which fills a given region R in the plane bounded by four curves. This is referred to as a (2D) meshing operation. The region R can be either a planar (thin plate) part that itself needs to be meshed or the parametric region of a trimmed 3D surface that will be readily meshed once its parametric region is meshed. As thousands or even hundreds of thousands of elements are usually required to accurately approximate a part surface, such a meshing operation inevitably has to be done by a computer, either fully automatically or semi-automatically with minimum human involvement.

The finite element method and the finite difference method are the two most powerful analysis tools in engineering. The finite element method usually works on either triangular grids or quadrilateral grids, and the grids can be structured or unstructured. Quadrilateral grids in general are preferred over triangular grids as the former ones usually generate smaller analysis errors than the latter. Unstructured 2D quadrilateral grids can be applied to the finite element computing packages. However, the finite difference method generally uses *structured quadrilateral grids* (or simply called *structured grids*) that are associated historically with rectangular Cartesian grids, since such a regular lattice structure provides easy identification of neighboring points to be used in the representation of derivatives. The purpose of our work in this paper is to construct non-self-overlapping structured quadrilateral grids in a give four-sided 2D region.

Automatic 2D mesh generation has been studied for decades [1-3]. Among various approaches, the three most popular ones are Delaunay-triangulation method [4-8], the partial differential equation (PDE) method [9-13], and the boundary-conforming mapping (BCM) method [14-17]. The Delaunay-triangulation approach meshes a region R by incrementally inserting points into the interior of R while maintaining certain special geometric properties on the resulting triangles. Although fully automatic, it can only generate triangular elements, which are less favored than quadrilaterals, and is very susceptible to geometric degeneracy and numerical instability, in addition to the high complexity of implementing the algorithm. Also, the result mesh can hardly be converted into structured grids. The PDE technique generates grids by first distributing points on the boundary curve and then solving elliptic PDEs in the field. Elegant formulation notwithstanding, it requires delicate numerical solutions and usually runs very slow; moreover, since it requires second-order derivatives, singularities appear when the boundary curve is only C^1 continuous or its second-order derivative is difficult to compute. The BCM method requires no algorithmic computation like the other two, thus it is fast and immune

to any numerical instability problem. Basically, a BCM is a continuous mapping from a square to a planar region R enclosed by its four curves. By trivially sampling the square into a rectangular grid and then mapping it into R, one obtains a quadrilateral mesh of R. Because of its simplicity, BCM method is widely used in 2D meshing. It is not an overstatement that every commercial surface meshing software package has a meshing utility module based on some kind of BCM.

Unfortunately, while being extremely simple and numerically robust, all BCM schemes suffer from a well-known deficiency: when the four bounding curves are concave and very convoluted, self-overlapping often tends to occur in the mesh, i.e., some quadrilateral elements overlap with each other, which makes the mesh useless to downstream applications. As an illustration, we made a trimmed surface with four trimming curves and fed the corresponding four parametric trimming curves to a leading commercial mesh software package using a bilinear Coons patch mapping (which is a popular BCM scheme favored by many mesh software packages); the output result is shown in Fig. 1 where the self-overlapping is obvious.



(a) in the parametric region **(b) on the 3D surface**

Fig. 1 Self-overlapping mesh from Coons patch mapping

Abundance of research can also be found in literature relating to mesh smoothing [18-24]. Most of them are based on the idea of relocating one node in a mesh to optimize the shapes of adjacent elements, where the Laplacian and Winslow smoothing techniques are widely utilized. However, all of these smoothing methods require a *valid* initial grid. For structured quadrilateral meshes, the initial grids are usually generated algebraically, e.g., via a BCM method. The smoothing approaches preserve non-self-overlapping condition only when the initial grid is valid, i.e., self-overlapping free.

In this paper, we study this self-overlapping behavior of a particular boundary-conforming mapping scheme – the *Hermite Interpolation Mapping* (HIM) [25-27], and investigate what measures can be taken to prevent the potential self-overlapping from occurring in the mesh. Under HIM, the geometry of the mesh depends on both the four bounding curves and the tangent vector fields associated with the bounding curves. While the four

bounding curves are given and hence fixed, the tangent vector fields are user-specified and can vary to suit our needs of controlling the characteristics of the mesh. This is the primary motivation behind. Explicitly, we set forth to develop a methodology that will find suitable tangent vector fields, adaptive to the input bounding curves, so that the resultant HIM will be free of self-overlapping. Rather than trying to seek the exact and analytical solution, which is achievable conceivably only for very simple types of bounding curves and tangent field vector functions due to high non-linearity of the differential equations involved, we model the solution as a functional optimization problem and propose a numerical method for solving it.

The paper is organized as follows. After giving necessary preliminaries and propositions in section 2, the tangent functions of the Hermite Interpolation Mapping are defined in section 3 as four Bézier polynomials. In section 4, the non-self-overlapping mesh generation problem is formulated as a constrained functional optimization problem, where the objective function is derived for the shape control of grid elements, and the non-self-overlapping condition is set as a constraint to the optimization. Section 5 gives the numerical implementation method of the constrained optimization, and section 6 considers the influence of sample points. Finally, in section 7 we show some experimental results to demonstrate the effectiveness of the proposed non-self-overlapping HIM approach, and in section 8 we conclude the paper with pointers to some future research.

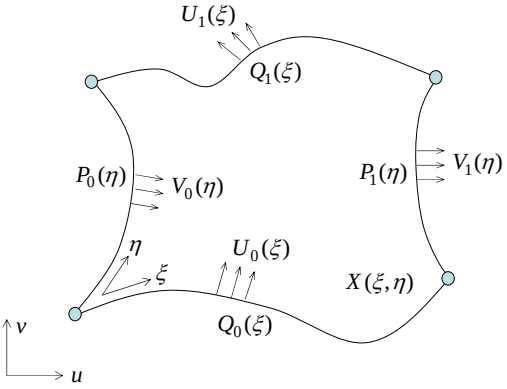


Fig. 2 Definition of Hermite Interpolation Mapping

2. Preliminaries

Refer to Fig. 2, let $Q_0(\xi)$, $Q_1(\xi)$, $P_0(\eta)$ and $P_1(\eta)$ $(0 \le \xi, \eta \le 1)$ be four C^1 continuous curves in the $u-v$ plane that form a simple closed curve, and $U_0(\xi)$, $U_1(\xi)$, $V_0(\eta)$, and $V_1(\eta)$ $(0 \le \xi, \eta \le 1)$ be four C^1 continuous vector functions that meet the following boundary conditions

$$U_0(0) = P_0'(0),\ U_0(1) = P_1'(0),\ U_1(0) = P_0'(1),\ U_1(1) = P_1'(1), \tag{1}$$

$$V_0(0) = Q_0'(0),\ V_0(1) = Q_1'(0),\ V_1(0) = Q_0'(1),\ V_1(1) = Q_1'(1).$$

We define the following three mappings from parametric $\xi - \eta$ domain to the $u - v$ plane

$$X_1(\xi,\eta) = \begin{bmatrix} f_0(\xi) & f_1(\xi) & f_2(\xi) & f_3(\xi) \end{bmatrix} \begin{bmatrix} P_0(\eta) \\ P_1(\eta) \\ V_0(\eta) \\ V_1(\eta) \end{bmatrix}, \tag{2}$$

$$X_2(\xi,\eta) = \begin{bmatrix} Q_0(\xi) & Q_1(\xi) & U_0(\xi) & U_1(\xi) \end{bmatrix} \begin{bmatrix} f_0(\eta) \\ f_1(\eta) \\ f_2(\eta) \\ f_3(\eta) \end{bmatrix}, \tag{3}$$

$$X_3(\xi,\eta) = \begin{bmatrix} f_0(\xi) & f_1(\xi) & f_2(\xi) & f_3(\xi) \end{bmatrix} \begin{bmatrix} Q_0(0) & Q_1(0) & U_0(0) & U_1(0) \\ Q_0(1) & Q_1(1) & U_0(1) & U_1(1) \\ V_0(0) & V_0(1) & t_{00} & t_{01} \\ V_1(0) & V_1(1) & t_{10} & t_{11} \end{bmatrix} \begin{bmatrix} f_0(\eta) \\ f_1(\eta) \\ f_2(\eta) \\ f_3(\eta) \end{bmatrix}, \tag{4}$$

where t_{00} through t_{11} are called *twist vectors* and can be of arbitrary values, and the $f_i(t)$ s are Hermite interpolation functions: $f_0(t) = 1 - 3t^2 + 2t^3$, $f_1(t) = 3t^2 - 2t^3$, $f_2(t) = t - 2t^2 + t^3$, $f_3(t) = -t^2 + t^3$. The Hermite Interpolation Mapping (HIM) is then defined as:

$$X(\xi,\eta) = X_1(\xi,\eta) + X_2(\xi,\eta) - X_3(\xi,\eta) \quad (0 \le \xi, \eta \le 1) \tag{5}$$

The HIM of Eq. (5) can be verified to satisfy the boundary conditions of derivatives below:

$$X_\xi(0,\eta) = V_0(\eta), \quad X_\xi(1,\eta) = V_1(\eta), \quad X_\eta(\xi,0) = U_0(\xi), \quad X_\eta(\xi,1) = U_1(\xi), \tag{6}$$

and

$$X_{\xi\eta}(0,0) = t_{00}, \quad X_{\xi\eta}(0,1) = t_{01}, \quad X_{\xi\eta}(1,0) = t_{10}, \quad X_{\xi\eta}(1,1) = t_{11}. \tag{7}$$

The mapping of Eq. (5) is easily seen to be *boundary conforming*, that is, we have the following one-to-one mapping between the four sides of the $\xi - \eta$ square and the four bounding curves in the *u*-*v* plane:

$$X(\xi,0) = Q_0(\xi), \quad X(\xi,1) = Q_1(\xi), \quad X(0,\eta) = P_0(\eta), \quad X(1,\eta) = P_1(\eta). \tag{8}$$

For a region R in the *u*-*v* plane bounded by $Q_0(\xi)$, $Q_1(\xi)$, $P_0(\eta)$ and $P_1(\eta)$, the mapping $X(\xi,\eta)$ is further said to be *region conforming* if the set $\{X(\xi,\eta) : 0 \le \xi,\eta \le 1\}$ is equal to region R. This is one necessary condition for meshing R using $X(\xi,\eta)$ – a grid element must not go outside the region. However, a region-conforming $X(\xi,\eta)$ does not yet ensure non-self-overlapping on the mesh, the mapping also has to be one-to-one, as we stipulate in the proposition below.

Proposition 1 The meshing of region R under mapping $X(\xi,\eta)$ is free of self-overlapping if and only if $X(\xi,\eta)$ is one-to-one; that is, $X(\xi_0,\eta_0)=X(\xi_1,\eta_1)$ if and only if $(\xi_0,\eta_0)=(\xi_1,\eta_1)$ for any (ξ_0,η_0), $(\xi_1,\eta_1)\in[0,1]$.

Consider the vector function $\frac{\partial X}{\partial \xi}\times\frac{\partial X}{\partial \eta}$, since R is a planar region in the $u-v$ plane, $\frac{\partial X}{\partial \xi}\times\frac{\partial X}{\partial \eta}$ is always perpendicular to the $u-v$ plane. We use the scalar function $J(\xi,\eta)=w(\frac{\partial X}{\partial \xi}\times\frac{\partial X}{\partial \eta})$ to denote the *w*-component of the vector $\frac{\partial X}{\partial \xi}\times\frac{\partial X}{\partial \eta}$, and call it the *Jacobian* of the mapping $X(\xi,\eta)$, where *w* designates the direction perpendicular to both *u*- and *v*-axis. We can compute it by the equation

$$J(\xi,\eta)=X_{\xi_u}X_{\eta_v}-X_{\eta_u}X_{\xi_v}, \tag{9}$$

where X_{ξ_u} and X_{ξ_v} represent the *u*- and *v*-components of X_ξ, and X_{η_u} and X_{η_v} represent the *u*- and *v*-components of X_η. As $X(\xi,\eta)$ is C^1 continuous, from classical functional analysis and differential geometry [28] (see also [29]), its Jacobian directly relates to the one-to-one property of the mapping; the following proposition is in order.

Proposition 2 The mapping $X(\xi,\eta)$ is one-to-one if and only if its Jacobian $J(\xi,\eta)$ is positive in the entire parametric domain $(0\le\xi,\eta\le 1)$.

Therefore, our task now becomes how to control those variables in Eq.(5) so that the resulting Jacobian $J(\xi,\eta)$ is guaranteed to be positive in the entire $u-v$ square. This calls first to look into the formulation of the tangent vector functions $U_0(\xi)$, $U_1(\xi)$, $V_0(\eta)$, and $V_1(\eta)$, as to be discussed next.

3. Formulation of Boundary Tangent Functions

Since $U_0(\xi)$, $U_1(\xi)$, $V_0(\eta)$, and $V_1(\eta)$ are four C^1 continuous vector functions, they can be approximated as parametric curves. Here, we deal with them as Bézier curves

$$U_0(\xi)=\sum_{i=0}^{m}a_i\mathrm{B}_{i,m}(\xi),\ U_1(\xi)=\sum_{i=0}^{m}b_i\mathrm{B}_{i,m}(\xi),\ V_0(\eta)=\sum_{j=0}^{n}c_j\mathrm{B}_{j,n}(\eta),\ V_1(\eta)=\sum_{j=0}^{n}d_j\mathrm{B}_{j,n}(\eta), \tag{10}$$

where $\mathrm{B}_{i,m}(\xi)$ and $\mathrm{B}_{j,n}(\eta)$ are the Bernstein basis functions of degree *m* and *n* with $\mathrm{B}_{j,k}(t) = \binom{j}{k} t^j (1-t)^{k-j}$, and a_i, b_i, c_i, and d_i are vectors that control the shape of the tangent functions – they are called *control vectors*. To satisfy the boundary condition of an HIM for $U_0(\xi)$, $U_1(\xi)$, $V_0(\eta)$, and $V_1(\eta)$ – Eq.(1), by the property of Bernstein basis function [27], we have

$$a_0 \equiv P_0'(0),\ a_m \equiv P_1'(0),\ b_0 \equiv P_0'(1),\ b_m \equiv P_1'(1), \tag{11}$$

$$c_0 \equiv Q_0'(0),\ c_n \equiv Q_1'(0),\ d_0 \equiv Q_0'(1),\ d_n \equiv Q_1'(1).$$

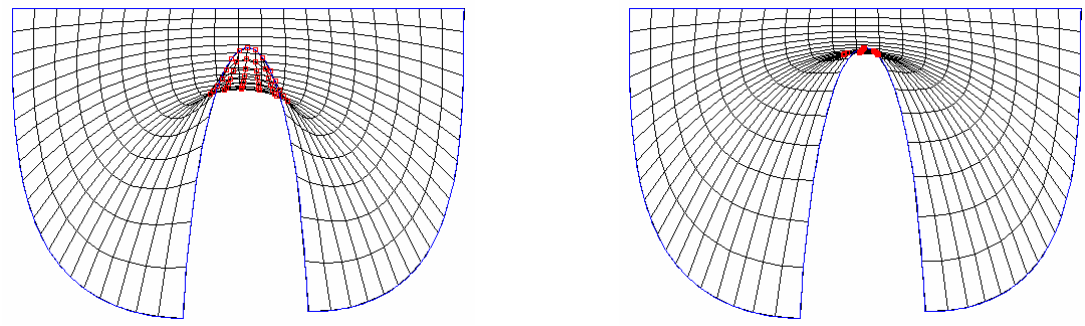
The rest of the control vectors, i.e., $\{a_1,\cdots,a_{m-1},b_1,\cdots,b_{m-1},c_1,\cdots,c_{n-1},d_1,\cdots,d_{n-1}\}$, can then be adjusted to influence the distribution of the tangent on the boundary of a Hermite Interpolation Mapping. By varying the tangent distribution on the boundary of $X(\xi,\eta)$, one thus can control the distribution of the Jacobian $J(\xi,\eta)$ inside $X(\xi,\eta)$. For example, as in Fig. 3, when changing $U_0(\xi)$ from the form of

$$U_0(\xi) = (1-\xi)P_0'(0) + \xi P_1'(0)$$

to the form of

$$U_0(\xi) = (1-\xi)^2 P_0'(0) + 2\xi(1-\xi)a_1 + \xi^2 P_1'(0), \text{ with } a_1 = (0, 500),$$

the number of points with negative Jacobian is decreased tremendously (the points with negative Jacobians are shown with small cubes). Therefore, as the control vectors influence the shape of the tangent functions which in turn influence the Jacobian on $X(\xi,\eta)$, they play the central role in our non-self-overlapping HIM construction algorithm.



(a) **(b)**

Fig. 3 Example I – the influence of the tangent functions on the Jacobian distribution

(a) $U_0(\xi) = (1-\xi)P_0'(0) + \xi P_1'(0)$ **;** (b) $U_0(\xi) = (1-\xi)^2 P_0'(0) + 2\xi(1-\xi)a_1 + \xi^2 P_1'(0)$ with $a_1 = (0, 500)$

Let $\{a_1,\cdots,a_{m-1}\}$, $\{b_1,\cdots,b_{m-1}\}$, $\{c_1,\cdots,c_{n-1}\}$, and $\{d_1,\cdots,d_{n-1}\}$ be the *adjustable* control vectors of boundary tangent functions $U_0(\xi)$, $U_1(\xi)$, $V_0(\eta)$, and $V_1(\eta)$ respectively. The Jacobian then is a function of all these control vectors, plus the four twist vectors t_{00}, t_{01}, t_{10}, and t_{11}, i.e.,

$$J(a_1,\cdots,a_{m-1},b_1,\cdots,b_{m-1},c_1,\cdots,c_{n-1},d_1,\cdots,d_{n-1},t_{00},t_{01},t_{10},t_{11}).$$

By carefully "designing" these vectors so that the corresponding Jacobian $J(\xi,\eta)$ is non-negative in the entire parametric space $(0\le\xi,\eta\le 1)$, the self-overlapping in the mapping $X(\xi,\eta)$ is effectively eliminated. Thus, these vectors are called *design vectors*. When solving this "design" problem, the *differentiation simplicity* is an important criterion for selecting the *base type* of the tangent vector functions (i.e., their derivatives and other differentiation operations can be performed easily). Since our tangent vector functions are formulated as Bézier curves which are polynomials, this criterion is satisfied due to the superb algebraic simplicity of Bezier functions.

Once the base type of the tangent functions and the design vectors are decided, the next task is then to assign proper values to these design variables that will enforce the non-negativity of the Jacobian in the square $(0\le\xi,\eta\le 1)$. Originally, we contemplated on deriving a general analytical and exact solution. However, owing to the complexity of relationship between the design variables and the Jacobian, this attempt was not successful. Instead, we propose to model this assignment problem as a functional optimization problem and present a numerical solution for determining the optimum, as to be detailed next.

4. Problem Formulation

An intuitive way of finding good values for the design variables is to formulate them into the following functional minimization problem

$$\textbf{min}\ E=\iint_{\Omega}\left[H(-J(a_1,\cdots,a_{m-1},b_1,\cdots,b_{m-1},c_1,\cdots,c_{n-1},d_1,\cdots,d_{n-1},t_{00},t_{01},t_{10},t_{11}))\right]d\xi d\eta$$

where Ω stands for the $\xi-\eta$ square and H(t) is the *Heaviside* function (i.e., H(t) = 1 if t > 0 and H(t) = 0 otherwise). As E counts exactly the number of points with negative Jacobian in the $\xi-\eta$ square, by minimizing it we also minimize the degree of self-overlapping, and a zero E indicates a mapping with no self-overlapping.

One caveat of the above formulation is the outright omission of the points in the $\xi-\eta$ square that have positive Jacobian. As an immediately consequence, the shape quality of the elements in the final mesh is completely ignored. Actually, the element shape quality is an essential issue in a grid generation algorithm; the

ideal shape of elements in a quadrilateral mesh is rectangular [3]. Similar to other algebraic grid generation approaches, the grid is established by uniformly subdividing the square $(0 \le \xi, \eta \le 1)$ into $M \times N$ elements, where the parameters of the element node (i, j) in $X(\xi, \eta)$ is given by

$$\begin{cases} \xi_i = \dfrac{i}{M} & (i = 0, \cdots, M) \\ \eta_j = \dfrac{j}{N} & (j = 0, \cdots, N) \end{cases}. \tag{12}$$

It is thus plausible to inquire if we can reformulate the problem with the element shape quality also considered.

Proposition 3 Every element in the mesh under a mapping $X(\xi, \eta)$ is rectangular if $X_\xi \perp X_\eta$ (i.e., $X_\xi \cdot X_\eta = 0$) at every $(\xi, \eta) \in [0,1] \times [0,1]$.

Combining this requirement for shape control and the original objective of maintaining non-negative $J(\xi, \eta)$ in $\Omega = (0 \le \xi, \eta \le 1)$, we now reformulate the problem as a *constrained* functional minimization problem. In this new formulation, the solution space remains the same, i.e., they are the design variables, or $\Pi = \{a_1, \cdots, a_{m-1}, b_1, \cdots, b_{m-1}, c_1, \cdots, c_{n-1}, d_1, \cdots, d_{n-1}, t_{00}, t_{01}, t_{10}, t_{11}\}$; the minimization objective however is now the shape control energy E_S, augmented by the constraint of positive Jacobian $J(\xi, \eta): 0 \le \xi, \eta \le 1$. Explicitly, we now have the following problem to solve:

$$\begin{aligned} &\textbf{min} \iint_\Omega (X_{\xi_u} X_{\eta_u} + X_{\xi_v} X_{\eta_v})^2 \, d\xi d\eta \\ &\textit{subject to } (X_{\xi_u} X_{\eta_v} - X_{\eta_u} X_{\xi_v}) > 0 : (\xi, \eta) \in \Omega . \end{aligned} \tag{13}$$

In our objective function, in order to simplify the variational computation process, only the orthogonal control is conducted. Other factors, such as aspect ratio and element size, can be left to the mesh smoothing procedures after obtaining an initial non-self-overlapping grids from our approach. Done this way is more convenient than considering all the factors at one time and in one functional.

5. Numerical Implementation

This section describes the necessary formulas and procedures for solving the constrained optimization problem as defined in Eq. (13). A penalty function based method is adopted to convert this constrained optimization problem into an unconstrained optimization problem, where our numerical implementation minimizes the objective function by means of the conjugate gradient method.

We convert the constrained optimization problem of Eq. (13) into its unconstrained counter-part by adding the constraint as a penalty term on the objective function [30]. The objective function to be optimized now becomes

$$J_O(\Pi) = \sum_{k\in I}\left[\left(X_{\xi_u} X_{\eta_u} + X_{\xi_v} X_{\eta_v}\right)_k\right]^2 + \lambda \sum_{k\in I}\left\{\max\left[\left(X_{\eta_u} X_{\xi_v} - X_{\xi_u} X_{\eta_v}\right)_k, 0\right]\right\}^2 \tag{14}$$

where I symbolizes the set of sample points on the patch X and λ is the coefficient to balance the weight between the penalty term and the shape control energy term. The static choice of λ is not easy; for a smaller λ, which deemphasizes the non-self-overlapping part, the computing process converges to a non-self-overlapping HIM very slowly; for a larger λ, on the other hand, the control on the shapes of the final grid elements would be weak. In our implementation, we incrementally increase λ during the optimization. Theoretically, we arrive at an optimum Π^* in the limit as λ tends to infinity. The optimized J_O with respect to Π is determined by a conjugate gradient method which includes the iterative process of computing gradients at the current point and searching an optimum point along the conjugate direction [30]. The values of all design vectors in Π are adjusted during the optimization along the conjugate direction, after each iteration. In this approach, the gradients are computed analytically, so most computing time during the optimization is spent on the optimum point search along the conjugate direction. The unnecessary details of the conjugate gradient method and its related gradient formulas are omitted here.

The initial values for the design vectors at the start of the optimization, i.e., Π^0, are set by us as:

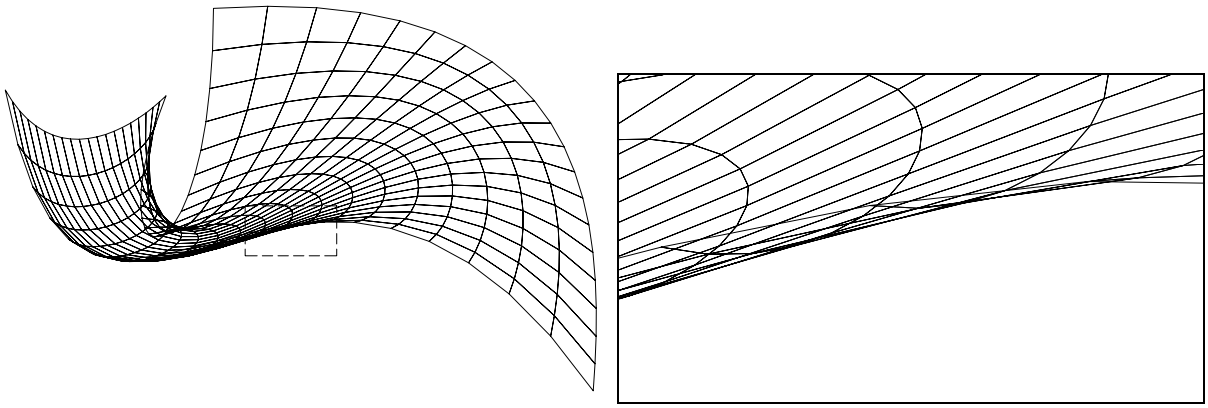
$$\begin{gathered} t_{00} = t_{01} = t_{10} = t_{11} = 0 \\ a_i = (1-\frac{i}{m})a_0 + \frac{i}{m}a_m, \quad b_i = (1-\frac{i}{m})b_0 + \frac{i}{m}b_m, \quad (i = 1,\cdots,m-1) \\ c_i = (1-\frac{j}{n})c_0 + \frac{j}{n}c_n, \quad d_j = (1-\frac{j}{n})d_0 + \frac{j}{n}d_n, \quad (j = 1,\cdots,n-1)\,. \end{gathered} \tag{15}$$

These initial conditions set up a “natural” state where every tangent function is a linear interpolation of its two end vectors as determined by the derivative boundary condition (Eq. (11)) and the four corners of the mesh are essentially flat due to the zero-value twist vectors. Any single closed 2D region can be elastically deformed from a rectangle without overlap. When the region R is a rectangle, the above initial condition (Eq.(15)) gives a non-self-overlapping mapping with rectangular elements. When an arbitrary region R is given, choosing the above initial condition is based on simulating the elastic deformation by the re-location of four corners. Therefore, at the corners, non-self-overlapping and element shape are preserved by the above initial condition.

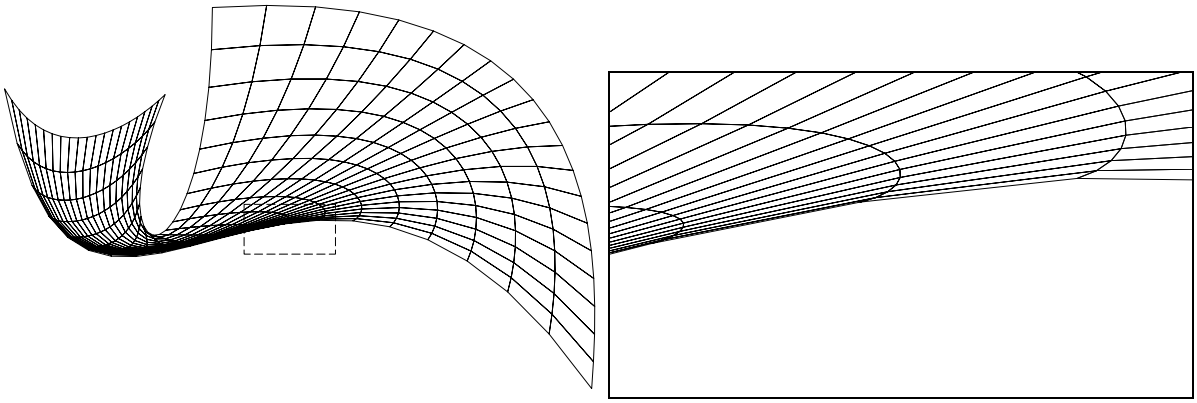
Before testing the experimental examples, we introduce the following global grids quality measurement factor:

$$E_S = \frac{1}{(M+1)(N+1)} \sum_{i=0}^{M} \sum_{j=0}^{N} \left\| \frac{X_\xi(\xi_i, \eta_j)}{\left\| X_\xi(\xi_i, \eta_j) \right\|} \cdot \frac{X_\eta(\xi_i, \eta_j)}{\left\| X_\eta(\xi_i, \eta_j) \right\|} \right\|. \tag{16}$$

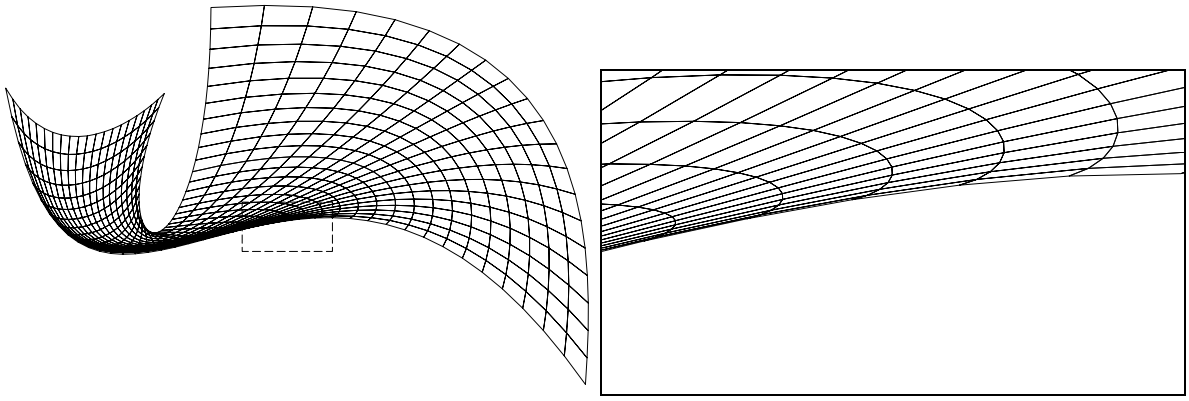
Obviously, when $E_S = 0$, it indicates $X_\xi \perp X_\eta$ at every (ξ_i, η_i) – so very element is rectangular. The smaller E_S, the better mesh quality.



(a) the grid generated with the initial HIM, with self-overlapping (M=20, N=20 with $E_S = 0.6383$)



(b) M = N = 20 (after optimization with 400 sample points) - $E_S = 0.6147$



(c) M = 40, N = 20 (by the HIM obtained from the optimization with 400 sample points) - $E_S = 0.6177$

Fig. 4 Example II – independence between the sample points and the grid

6. Sampling Points

How to sample the points in domain Ω is critical to expediting the convergence of the functional optimization as given in Eq. (13). In our numerical implementation, the sample points set I is generated randomly – the sample points are not required to coincide with the grid nodes. This is an important feature due to the fact that our optimization objective is not limited to the individual grid nodes, but rather pertains to the HIM that governs the final grid generation. In all our examples, the population size of sample points is set to be 400. Of course, more sample points give more accurate result, but at the same time, slow down the computation of optimization. Therefore, we need to seek a balance between the speed and the accuracy. After our testing, we found that 400 sample points work well for the grids with size $M \times N$ less than 1000. It is because that one sample point can have effect on four grid nodes when it is not coincident to a grid node; after considering the probability of being coincident with grid node and two sample points fall in the same grid, the possible grid size could maximally be 2.5 times the number of sample points. This decoupling between the grid nodes and the sample points is very helpful, from the performance point of view at least, since the size of the grid elements is usually required to be very small by downstream applications, which otherwise would result in a large number of sample points if the two are coincided and consequently consumes longer computing time. In Fig. 4, we show example II that demonstrates our point.

To further accelerate the convergence, we did not use the *normal random sampling* for region $\Omega(0 \le \xi, \eta \le 1)$; instead, a *Gaussian random sampling distribution* is used. This new sampling strategy favors places where severe self-overlapping occurs but at the same time still maintains the randomness nature. First, a uniform distribution of sample points is generated randomly, and the mean point q_c of the sample points with a negative Jacobian is computed. Then, we re-generate the sample points around q_c by following a Gaussian distribution. A Gaussian distribution can be converted from a planar distribution by the polar form of the Box-Muller transformation [31, 32]. Our algorithm of generating sample points around q_c with a Gaussian distribution is as shown in Table 1. Why would Gaussian random sampling distribution expedite the convergence rate? This distribution gives more weights at the regions where self-overlap occur during the constrained optimization; correspondingly, greater gradients at those regions are achieved during the optimization – leading to faster convergence.

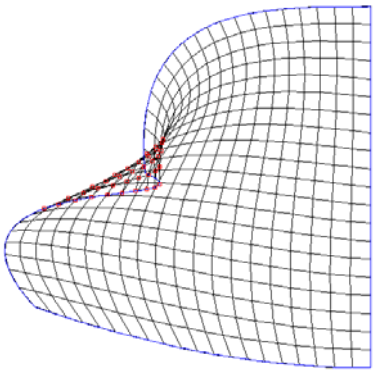
Table 1 Algorithm *GaussianSamplePointsGeneration*(n, q_c)

Algorithm *GaussianSamplePointsGeneration*(n, q_c)

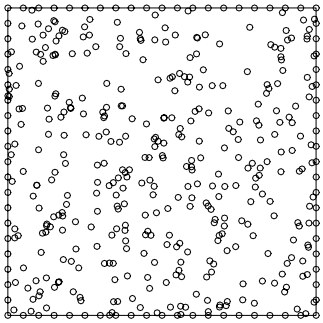
Input: the number of sample points n, and the location of the Gaussian distribution's center q_c

Output: the sample point list L_s

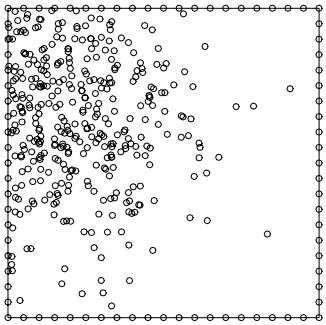
1. $L_s \leftarrow \phi$ and $i \leftarrow 0$;
2. Generate sample points on the boundaries uniformly;
3. **Do** {
4. **Do** {
5. **Do** {
6. $p_\xi \leftarrow 2\ ranf() - 1$;
7. $p_\eta \leftarrow 2\ ranf() - 1$;
8. $w \leftarrow \|p\|$;
9. } **while**($w >= 1.0$);
10. $w \leftarrow \sqrt{(-2\ln w)/w}$;
11. $p \leftarrow wp + q_c$;
12. } **while**($p \notin [0,1] \times [0,1]$);
13. Add p into L_s;
14. $i \leftarrow i + 1$;
15. } **while**($i < n$);



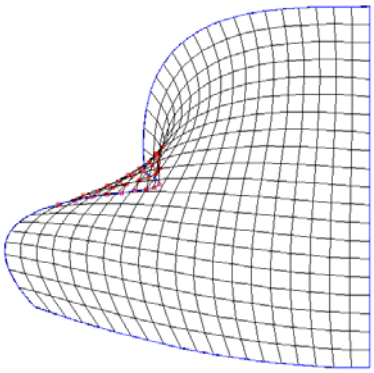
(a) grid with the init HIM



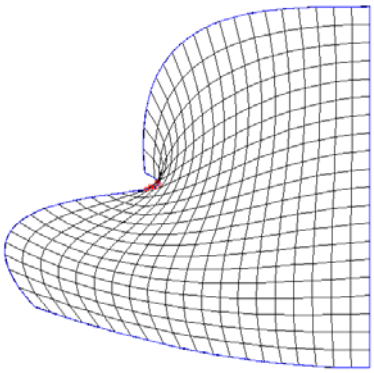
(b) uniform distributed sample points



(c) Gaussian distributed sample points



(d) the grid with the HIM after one iteration step with uniform distributed sample points

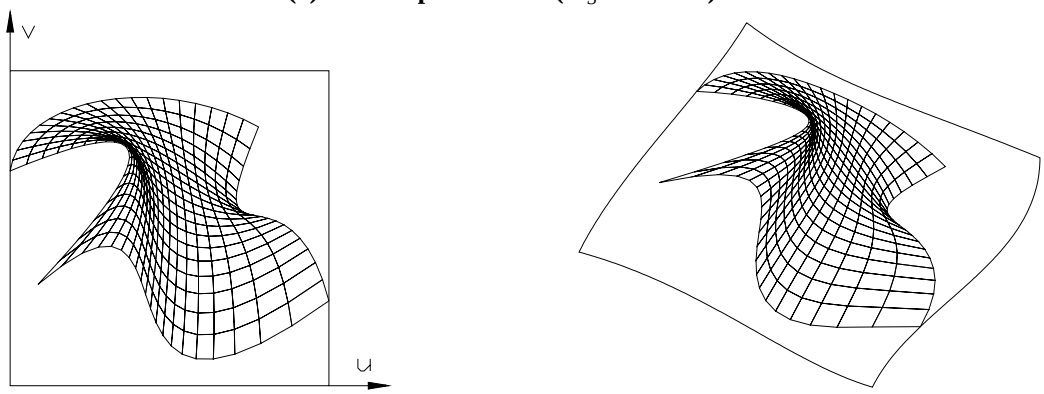


(e) the grid with the HIM after one iteration step with Gaussian distributed sample points

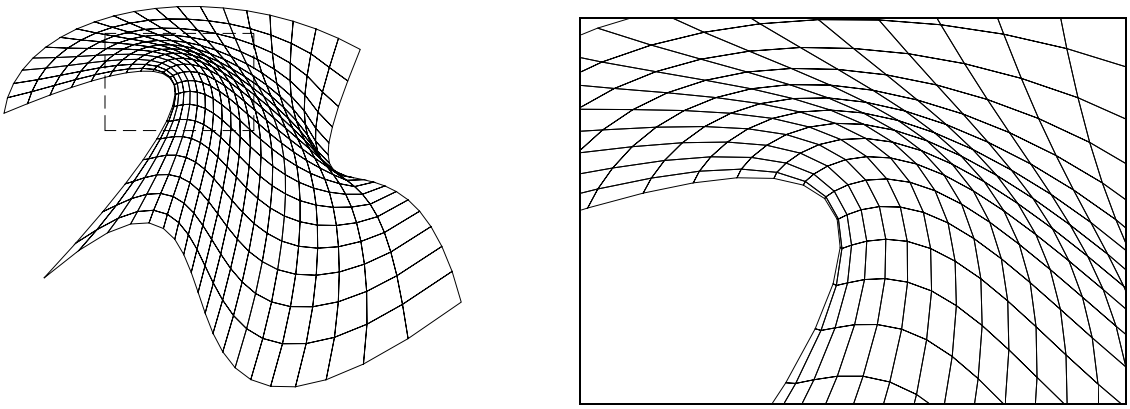
Fig. 5 Uniform-random vs. Gaussian-random distribution of sampling points

To illustrate our point, an example is given in Fig. 5. Fig. 5(a) is the mesh generated by the initial HIM. The sample points with uniform random distribution are displayed in Fig. 5(b), and the sample points with Gaussian distribution centered at the mean point of the self-overlapping region are shown in Fig. 5(c). With *one* iteration step, the HIM optimized meshes based respectively on the uniform random and Gaussian distributions give out the grids as shown in Fig. 5(d) and 5(e) respectively. Obviously as seen from the figure, the latter mesh (Fig. 5(e)) has eliminated more self-overlapping than the former one (Fig. 5(d)).

(a) before optimization ($E_S = 0.6481$ **)**



(b) after optimization ($E_S = 0.6525$ **)**



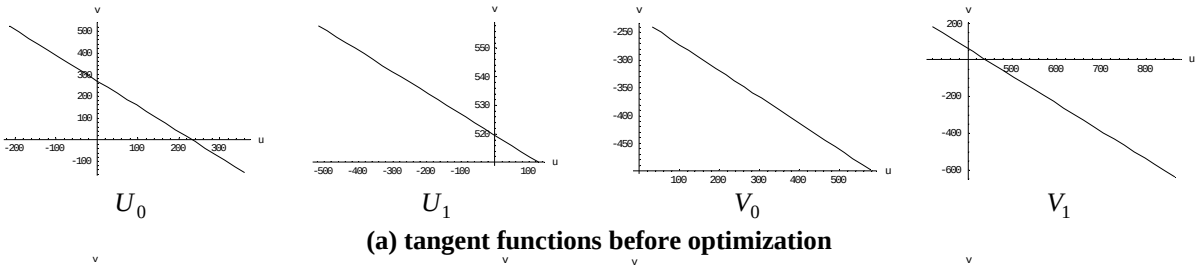
(c) the mesh after Winslow smoothing

Fig. 6 Example III – a trimmed parametric surface

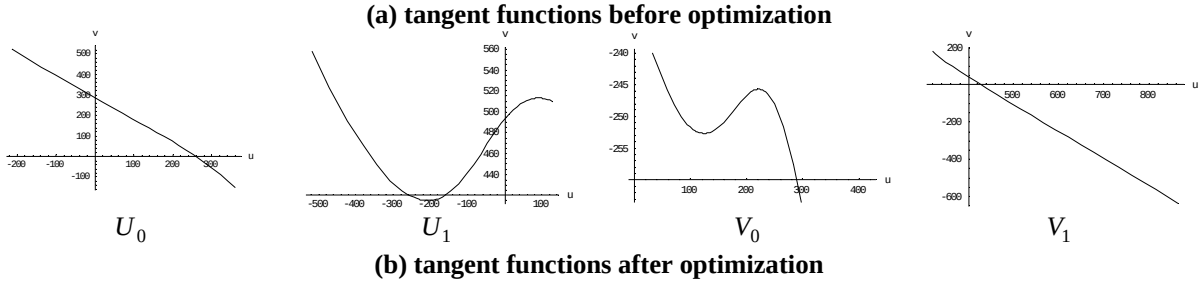
7. Experimental Results and Discussion

Fig. 6 shows the third example, example III, of the experimental results, the structured grid of a trimmed surface. The initial meshes in the parametric space and on the surface (based on the initial setting of Eq. (21)) are displayed in Fig. 6(a), where $E_S = 0.6481$. After the constrained optimization, the resulting meshes, as shown in Fig. 6(b), are free of self-overlapping, where $E_S = 0.6525$. In our five testing examples, this is the only example with quality factor increased after optimization. The major reason for the increase of E_S is that the optimization is constrained by the non-self-overlapping condition, so in order to fully satisfy this constraint the computation sometime needs to release the requirement of element quality. Fig. 6(c) exhibits the resultant mesh after applying the Winslow smoothing [21] to the grids of the optimized HIM; if applying such a smoothing algorithm directly to the initial mesh (Fig. 6(a)), the overlap will still be there after smoothing. This is because the basic idea of mesh smoothing algorithms is to relocate a node based on the positions of its adjacent nodes. Thus, if some of the adjacent nodes are flipped, the smoothing algorithm can hardly flip them back.

For a more poignant comparison for example III, Fig. 7(a) and 7(b) draw the four tangent functions (which are Bézier curves in the *u*-*v* plane) before and after the optimization, respectively. The final four twist vectors after the optimization are t_{00} = (-13.1, -9.4), t_{01} = (-65.4, 98.4), t_{10} = (44.5, -50.8), and t_{11} = (-2.7, 6.8).



(a) tangent functions before optimization

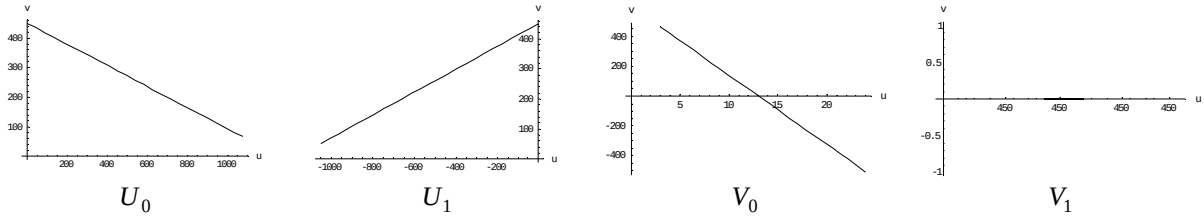


(b) tangent functions after optimization

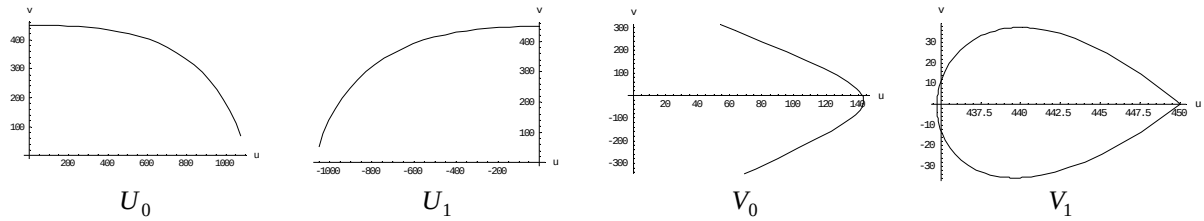
Fig. 7 Example III – comparison of tangent functions

Fig. 8 depicts the tangent functions of the optimization result of the first example, Example I, which has been previously visited in Fig. 3. The final four twist vectors are t_{00} = (-54.5, -85.1), t_{01} = (60.9, -82.6), t_{10} = (7.1, 41.7), and t_{11} = (-6.5, 42.7). This example demonstrates that our algorithm has reasonably small condition number, e.g., the mesh will reserve the symmetry of the region, as manifested by the symmetry among the

finally optimized tangent functions. The global grids quality measurement factor E_S decreases from 0.3328 to 0.3166 by the optimization.

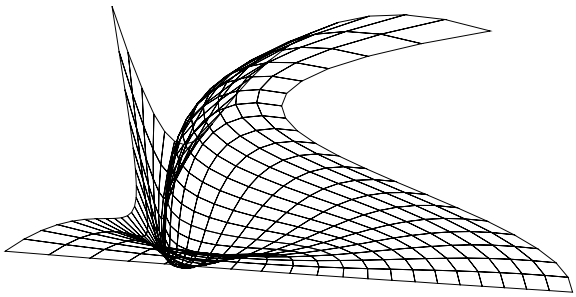


(a) tangent functions before optimization ($E_S = 0.3328$)

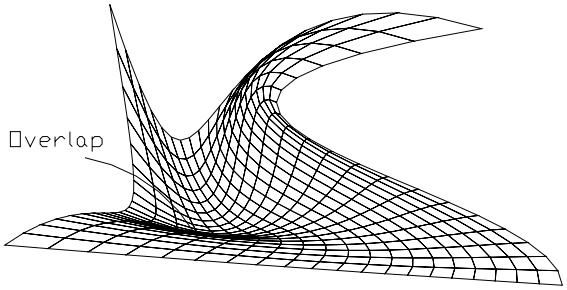


(b) tangent functions after optimization ($E_S = 0.3166$)

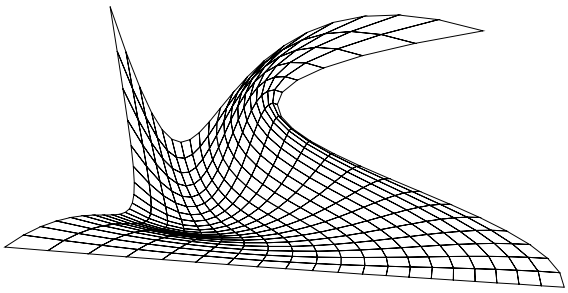
Fig. 8 Example I – comparison of tangent functions



(a) initial mesh ($E_S = 0.7313$)



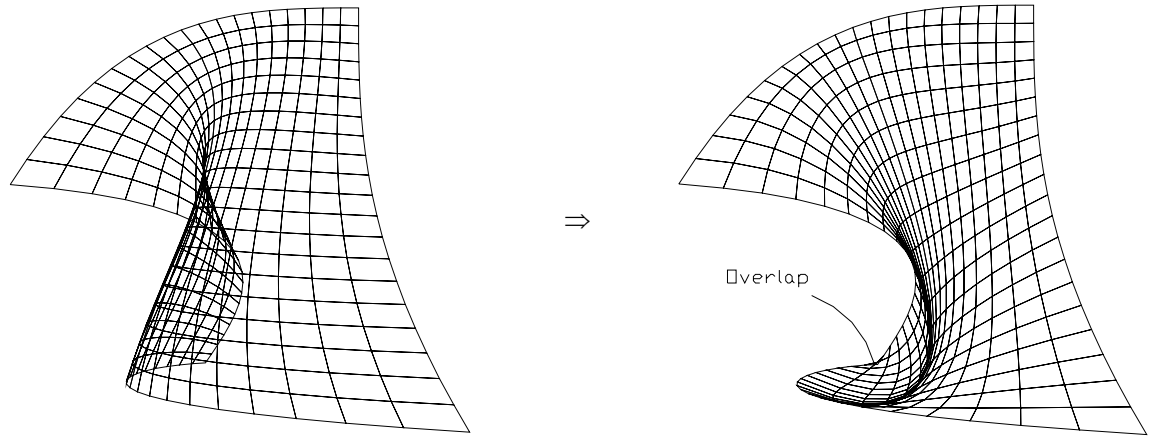
(b) optimization result with degree-two tangent functions after 100 iterations ($E_S = 0.6415$)



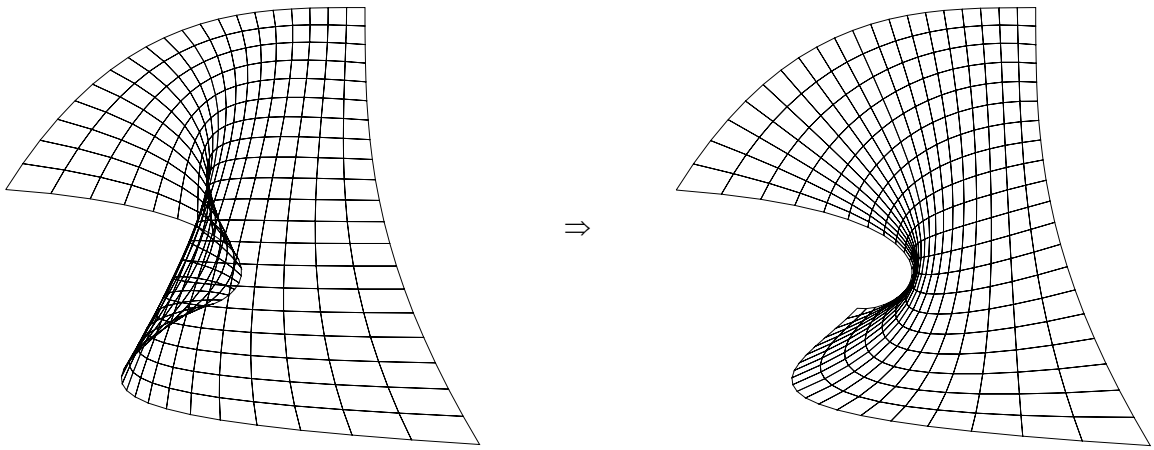
(c) optimization result with degree-three tangent functions after 50 iterations ($E_S = 0.6005$)

Fig. 9 Example IV – choose the proper degree of tangent vector functions

When applying the functional optimization to determine the optimized HIM, the degrees of the tangent vector functions – m and n as given in Eq. (10) – ought to be chosen carefully. We have observed in our experiments that, if the degree of a tangent function is lower than the degrees of the given four boundary curves, the functional optimization might not converge. As an illustration, the four boundary curves in the example shown in Fig. 9 are of degree-3. When setting $m = n = 2$ for the four tangent vector functions, the self-overlapping still remains even after 100 iterations (Fig. 9(b)). By increasing the degree of the tangent functions just by one from 2 to 3, we successfully obtain a non-self-overlapping HIM after just 50 iterations (Fig. 9(c)). On the other hand, however, higher order polynomials tend to make the final shape of the tangent functions oscillate violently which would adversarially affect the overall quality of the mesh, in addition to the extra computational cost (time and memory requirement). Thus, we usually set the degree of tangent functions to be the same as that of the boundary curves or just one degree higher.



(a) irremovable self-overlapping due to negative Jacobian at a corner point ($E_S = 0.4786$ **)**



(b) optimization with all positive Jacobian corner points ($E_S = 0.3673$ **)**

Fig. 10 Example V – the influence of the Jacobian at the corner points

Another issue worth mentioning is the *corner condition* of the given boundary curves. The Jacobians at the corner points of the four given boundary curves are completely determined by these curves (see Eqs. (1) through (8)) and are independent of the mapping $X(\xi,\eta)$. Therefore, a negative Jacobian corner point will always remain negative regardless of the optimization. Because Hermite mapping $X(\xi,\eta)$ is C^1 continuous, for a corner, say (0, 0), with negative Jacobian, there must exist a neighborhood (0, ε)×(0, ε) for some ε>0 in which the Jacobian is negative. Thus, any mesh will incur self-overlapping regardless of the optimization (see Fig. 10(a)). As a comparison, Fig. 10(b) shows a region similar to that in Fig. 10a, but without the negative Jacobian corner point; its non-self-overlapping mesh is easily constructed after just 50 iterations.

The last example is the application of the proposed optimization algorithm in CAD. Very often when a conceptual part is initially designed, the designer first sketches a skeleton of space curves over which a "skin" surface is then delineated to define the shape of the part. Coons patch or Hermite interpolation patch are the two most popular types of "skin" surfaces for filling the "holes" in the skeleton. Fig. 11(b) gives the skeleton of a cell-phone cover shown in Fig. 11(a), and Fig. 11(c) displays the corresponding "skin" surface made of HIM using linear tangent functions and zero twists. The undesired swerving at the bottom of the U-shape boundary is clearly seen, which is due to the strong non-convexity[*] of the boundary curves. As a remedy, we first project the four skeleton edges of the cover onto the keyboard plane of the cell-phone; the proposed HIM optimization algorithm is then applied to the four projected curves in this plane, generating four tangent functions U_0, U_1, V_0, and V_1, as well as the four twist values t_{00}, t_{01}, t_{10}, and t_{11}. These are then used to define a Hermite interpolation directly on the four original (3D) edges. Fig. 11(d) depicts the final part thus obtained, where the improvement in the quality of the surface – in terms of both visual appealing and surface curvature smoothness – is evidently manifested (note the circled place in Fig. 11(c) and Fig. 11(d)).

Finally, Table 2 lists the computing statistics of the examples – Example I through VI. All the tests are performed on a PIII 900 PC with the program written in Java. Usually, several minutes are needed including the displaying time. In the current prototypical implementation, we use rather primitive numerical methods to compute the functional optimization. It is believed that with more efficient functional optimization algorithms, the running time can be significantly shortened.

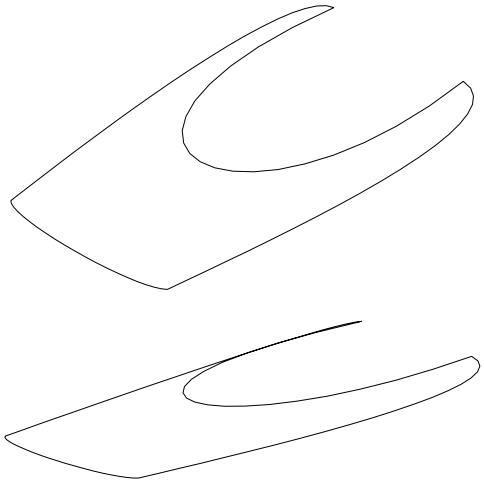
[*] Strictly speaking, the term "convexity" applies only to oriented 2D curves. Here it is used loosely for a 3D curve that is near planar.

Table 2 Running time cost of the examples

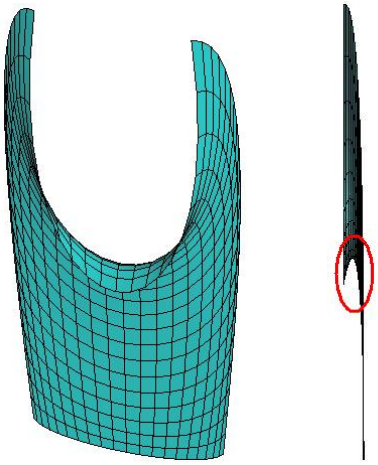
Example	Result figures	Degree of tangent functions	Running time
I	Fig. 8	3	5.3 min.
II	Fig. 4	3	2.3 min.
III	Fig. 6 and 7	3	2.9 min.
IV	Fig. 9(b)	2	3.1 min.
	Fig. 9(c)	3	1.8 min.
V	Fig. 10(a)	3	5.5 min.
	Fig. 10(b)	3	4.6 min.
VI	Fig. 11(d)	3	3.1 min.

* The sample size for the numerical integration is 400.

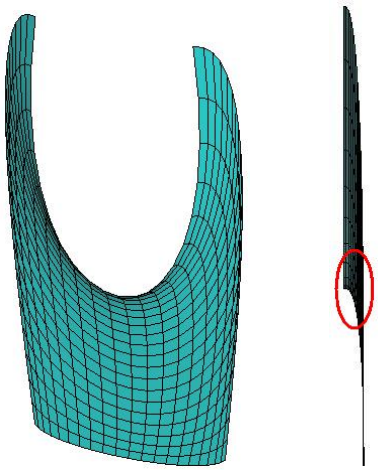
(a) the cell-phone to be modeled



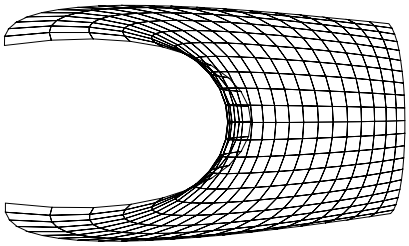
(b) skeleton of the cover



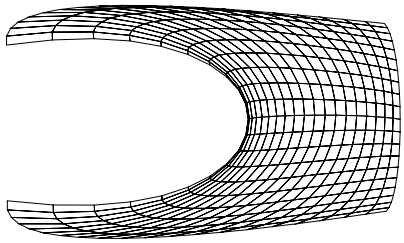
(c) the skinned surface by linear HIM patch



(d) the skinned surface by our optimized HIM



(e) the projected HIM before optimization



(f) the projected HIM after optimization

Fig. 11 Example VI – an application in CAD

8. Conclusion

This paper proposes using Hermite interpolation mapping (HIM) for generating structured quadrilateral grid in a given 2D region defined by four boundary curves. As self-overlapping in the grid may occur under an HIM when the boundary curves are concave and convoluted, we present a numerical solution for adaptively "designing" the tangent functions in the HIM which will guarantee to be free of self-overlapping. In details, the four tangent fields on the four boundary curves are formulated as Bézier curves. The control vectors of these four curves are designed so that the resulting mesh will be free of self-overlapping. Other design vectors that are utilized to control the mesh quality are the twist vectors of the HIM. The "assignment" of these design vectors is then modeled as a constrained functional optimization problem, where the optimization is carried out on the shape control energy that measures the overall mesh quality of the underlying HIM while the self-overlapping is strongly prevented in the form of constraints to the optimization. To solve this constrained functional optimization problem, a penalty function based scheme is utilized that converts the constrained optimization into an unconstrained one which is then solved with a conjugate-gradient method. During the numerical computation, in order to archive a faster convergence, the polar form of the Box-Muller transformation is adopted to transform the uniform distributed sample points into a Gaussian distribution with center located at the mean point of the self-overlapped region. Testing examples show that our approach can generate non-self-overlapping quadrilateral grids with good element shape in a 4-sided region, where the boundaries are very concave and convoluted.

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 our HIM approach. It averts the singularity of elliptic PDE methods when only a C^1 continuous boundary is available; also, the four generic parametric C^1 boundary curves are not required to be converted into a specified representation form which might generate conversion errors.

The main drawback of our HIM method, at least of the current implementation, is the computation time. The time cost of examples given in this paper is shown in Table 2. The same as other iteration-based methods, the computing usually takes several minutes. However, since rather primitive numerical methods are adopted in our current implementation for the functional optimization, the speed is respected to be improved when using more efficient optimization algorithms. Moreover, the following two topics or improvements are worth future research:

- In our current element shape control, the shapes of elements are controlled in the $u-v$ plane; when the shape of grid elements are to be controlled on the given parametric surface M (i.e., in the spatial domain but not the parametric domain), the shape control energy term needs to be redefined – this time, we should try to make $M_\xi \cdot M_\eta \to 0$ allover the patch.
- 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 Hermite solid. Then, the governing equation may come from the similar idea in this paper by detecting certain intrinsic properties of the Hermite solid.

9. Reference

[1] George PL. *Automatic mesh generation: application to finite element methods*. J. Wiley; Paris: Masson, 1991.

[2] Knupp P., Steinberg S. *Fundamentals of grid generation*. CRC Press: Boca Raton, 1994.

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

[4] Chew LP. "Constrained Delaunay triangulations". *Algorithmica*, vol.4, no.1, 1989, pp.97-108.

[5] de Floriani L, Puppo E. "An online algorithm for constrained delaunay triangulation", *CVGIP-Graphical Models & Image Processing*, vol.54, no.4, July 1992, pp.290-300.

[6] Fang TP, Piegl LA. "Algorithm for constrained Delaunay triangulation", *Visual Computer*, vol.10, no.5, 1994, pp.255-65.

[7] Anglada MV. "An improved incremental algorithm for constructing restricted Delaunay triangulations", *Computers & Graphics*, vol.21, no.2, 1997, pp.215-23.

[8] Chin F, Wang CA. "Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time", *SIAM Journal on Computing*, vol.28, no.2, 1998, pp.471-86.

[9] Thompson JF. "A survey of dynamically-adaptive grids in the numerical solution of partial differential equations", *Applied Numerical Mathematics*, vol.1, 1985, pp.3-27.

[10] Soni BK. "Elliptic grid generation system: control functions revisited. I". *Applied Mathematics & Computation*, vol.59, 1993, pp.151-163.

[11] Khamayseh A, Hamann B. "Elliptic grid generation using NURBS surfaces", *Computer Aided Geometric Design*, vol.13, 1996, pp.369-386.

[12] Knupp PM. “Jacobian-weighted elliptic grid generation”, *SIAM Journal on Scientific Computing*, vol.17, 1996, pp.1475-1490.

[13] Kim S. “Control functions and grid qualities measurements in the elliptic grid generation around arbitrary surfaces”, *International Journal for Numerical Methods in Fluids*, vol.33, 2000, pp.81-88.

[14] Smith RE, Eriksson LE. “Algebraic grid generation”, *Computer Methods in Applied Mechanics & Engineering*, vol.64, 1987, pp.285-300.

[15] Shih TIP, Bailey RT, Nguyen HL, Roelke RJ. “Algebraic grid generation for complex geometries”, *International Journal for Numerical Methods in Fluids*, vol.13, 1991, pp.1-31.

[16] Brakhage KH, Muller S. “Algebraic-hyperbolic grid generation with precise control of intersection of angles,” *International Journal for Numerical Methods in Fluids*, vol.33, 2000, pp.89-123.

[17] Yang DCH, Chuang J-J, OuLee TH. “Boundary-conformed toolpath generation for trimmed free-form surfaces”, *Computer-Aided Design*, vol.35, no.2, 2003, pp.127-39.

[18] Aiffa M., Flaherty JE. “A geometrical approach to mesh smoothing”, *Computer Methods in Applied Mechanics & Engineering*, vol.192, no.39-40, pp.4497-4514, 2003.

[19] Hermansson J., Hansbo P. “A variable diffusion method for mesh smoothing”, *Communications in Numerical Methods in Engineering*, vol.19, no.11, pp.897-908, 2003.

[20] Lee YK, Lee CK. “A new indirect anisotropic quadrilateral mesh generation scheme with enhanced local mesh smoothing procedures”, *International Journal for Numerical Methods in Engineering*, vol.58, no.2, 2003, pp.277-300.

[21] Knupp PM. “Winslow smoothing on two-dimensional unstructured meshes”, *Engineering with Computers*, vol.15, no.3, pp.263-268, 1999.

[22] Amenta N, Bern M, Eppstein D. “Optimal point placement for mesh smoothing”, *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM. 1997, pp.528-37.

[23] Charakhch'yam AA, Ivanenko SA. “A variational form of the Winslow grid generator”, *Journal of Computational Physics*, vol.136, no.2, pp.385-398, 1997.

[24] Parthasarathy VN, Kodiyalam S. “A constrained optimization approach to finite element mesh smoothing”, *Finite Elements in Analysis & Design*, vol.9, no.4, Sept. 1991, pp.309-20.

[25] Foley TA. “Scattered data interpolation and approximation with error bounds”, *Computer Aided Geometric Design*, vol.3, no.3, 1986, pp.163-77.

[26] Foley TA. “Weighted bicubic spline interpolation to rapidly varying data”, *ACM Transactions on Graphics*, vol.6, no.1, Jan. 1987, pp.1-18.

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

[28] do Carmo MP. *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, N.J.: Prentice-Hall, 1976.

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

[30] Belegundu AD, and Chandrupatla TR. *Optimization Concepts and Applications in Engineering*, Upper Saddle River, N.J.: Prentice Hall, 1999.

[31] Box GEP, and Muller ME. “A note on the generation of random normal deviates”, *Annals Math. Stat*, vol.29, 1958, pp.610-611.

[32] Carter E. “Generating Gaussian random numbers”, (http://www.taygeta.com/random/gaussian.html).