Using Chebyshev Polynomial Numerical Method to Rebuild Object Surface

Chen-San Chen
Department of Mechanical Engineering,Taipei City University of Science and Technology

Abstract

In reverse engineering, rebuilding a surface more precisely requires more data to be handled. Such a large amount of data cannot be treated using a CAD/CAM system. This study applies orthogonal polynomial functions to rebuild a surface. Fewer data are used to rebuild a surface and the performance of smooth error is compared to that of the NURBS algorithm. The proposed numerical method efficiently derives the coefficients. It requires only specified data sets to calculate the coefficients of the orthogonal function. The desired data set is determined by Lagrange interpolation method and a least square method is presented to determine the exact coefficients from linear simultaneous equations. The results show that the approximation error obtained using orthogonal polynomial functions to rebuild a continuous free surface are lower than obtained when the NURBS method is used.

Keywords: NURBS, orthogonal function, reverse engineering, surface rebuild

Ⅰ. Introduction

Conventionally, the manufacturing of products begins with an engineering drawing and ends with the manufacturing of the product according to the specifications in that drawing. In reverse engineering, computer-controlled measuring machines or high-resolution laser scanning machines are used by designers to create and/or modify the design of an existing object. Reverse engineering includes four stages (1) extracting surface data; (2) processing extracted data processing; (3) reconstructing the surface, and (4) generating a CNC part program. The non-uniform rational B-spline surface algorithm (NURBS) is the most popular and widely accepted method for rebuilding free-form surface. The many disadvantages of a surface reconstruction system include having to handle too many data, difficulties in handling common geometric and difficulties in treating complicated surfaces. This study uses orthogonal polynomial functions to reduce the amount of data to rebuild the free-form surface more quickly and precisely.

Over recent years, several researchers have focused on particular points and parameters that affect the rebuilding of a surface.[1-3] Pahk et al.[4] developed an interactive system for inspecting form molds with sculptured surfaces. The measured data were adjusted using a calculated form error, and the evaluated errors were displayed graphically. All of these studies involved many data to construct the surfaces. This study considers the use of orthogonal functions to construct free-form surfaces. Orthogonal functions have been proven to be able to approach any functions.[5] However, their coefficients are difficult to be determined. Orthogonal functions have recently been used to build neural networks that can learn mappings between input and output data. The defined training algorithm will learn the exact coefficients (or weights of the neural network). Huang and Cheng[5] applied the approximation theory of orthogonal functions to construct a neural network. The training algorithm derived from the least-squared algorithm converged faster than traditional backpropagation methods. Yang and Tseng[6,7] developed a single-layer orthogonal neural network(ONN) whose processing elements were based on orthogonal functions. This network did not suffer from the problems associated with traditional feed-forward neural networks, including the need to determine the initial weights and the number of layers and processing elements. Mai et al.[8] applied an orthogonal neural network to build a learning controller for balancing the ball beam system. Sher et al.[9] compared five existing orthogonal functions, and found that Legendre polynomials and Chebyshev polynomials of the first kind were suitable for generating the neural network, because of their properties of recursion and completeness.

This article presents a numerical method for determining the coefficients of the orthogonal function used to rebuild free-from surfaces. The performance of this method is

compared to that of the NURBS algorithm. The proposed numerical method efficiently derives the coefficients. It requires only *n+1* specified data sets to calculate the *n* coefficients of the orthogonal function with *n* terms. The desired data set is determined by Lagrange interpolation method and a least square method is presented to determine the exact coefficients from linear simultaneous equations.

An illustrative example of the SURFACER software is selected to confirm the performance of the proposed method in rebuilding free-form surfaces. The results indicate that the approximation error associated with using orthogonal polynomial functions to rebuild a free surface is less than that obtained using the NURBS method.

Ⅱ. Theory of Orthogonal Functions

According to the theory of orthogonal functions,[10,11] an arbitrary function $f(x)$, *f*:[a, b] $\rightarrow \Re$, will have an orthogonal polynomial

$$F_n(x) \cong \omega_1\phi_1(x) + \omega_2\phi_2(x) + \omega_3\phi_3(x) + \cdots + \omega_n\phi_n(x) \tag{1}$$

Such that $\lim_{n\to\infty}\int_a^b (f(x) - F_n(x))^2 dx = 0$ (2)

where $\int_a^b \phi_i(x)\phi_j(x)dx = \begin{cases} 0 & i \neq j \\ A_i & i = j \end{cases}$

$$\omega_i = \int_a^b f(x)\phi_i(x)dx / A_i \quad i = 1,2,\cdots \tag{3}$$

$\{\phi_1(x), \phi_2(x), \cdots\}$ is an orthogonal set.

The above equations show that a single-variable function *f(x)* can be approximated by an orthogonal function set and the coefficient, ω_i, is unique. Equation (3) shows that the coefficient, ω_i, will not be available from the integration if function $f(x)$ is unknown. For approximating multi-variable functions, a multi-variable orthogonal function set can be generated by integrating single-variable orthogonal function sets. For example, the corresponding *r*-variable orthogonal function set for a function with *r* variables will be $\{\Phi_1(X), \Phi_2(X), \Phi_3(X), ...\}$. Each of the orthogonal functions is defined as

$$\Phi_i(X) = \phi_{1i}(x_1)\phi_{2i}(x_2)\cdots\phi_{ri}(x_r) \tag{4}$$

As described above, an arbitrary multiple - variable function $y = F(X)$, $X = [x_1 x_2 \ldots x_m]^T$, can also be approximated by an orthogonal function set as shown below.

$$y = F(X) = \sum_{i=1}^{n} \omega_i \Phi_i(X) + R(X,n) = W^T \Phi(X) + R(X,n) \tag{5}$$

where ω_i is the coefficient of the orthogonal function $\Phi_i(X)$. $R(X,n)$ is the expansion error of the orthogonal functions. $\{\Phi_1(X),\Phi_2(X),\ldots\Phi_n(X)\}$ is a set of orthogonal functions.

Supposed that there are total m pairs of datum (x_1,y_1), (x_2,y_2),..., (x_m,y_m), then Eq. (1) becomes

$$Y=\omega_0\phi_0(x)+\omega_1\phi_1(x)+\cdots\cdots+\omega_n\phi_n(x) \tag{6}$$

We will find the value of ω_i, the coefficient of the orthogonal function $\phi_i(x)$, so that Eq. (6) will be able to approach all the given data. Here we apply the least-square approximation method to find the value of ω_i. Let *S* represent the square error.

$$S=\sum_{i=1}^{m}(y_i-Y)^2 \tag{7}$$

The extreme value of *S* can be obtained by letting the first-order derivation of *S* equal to zero. That is

$$\begin{cases} 0=\dfrac{\partial S}{\partial\omega_0}=\displaystyle\sum_{i=1}^{m}2(y_i-\omega_0\phi_0(x_i)-\omega_1\phi_1(x_i)\cdots\cdots-\omega_n\phi_n(x_i))(-\phi_0(x_i)) \\ 0=\dfrac{\partial S}{\partial\omega_1}=\displaystyle\sum_{i=1}^{m}2(y_i-\omega_0\phi_0(x_i)-\omega_1\phi_1(x_i)\cdots\cdots-\omega_n\phi_n(x_i))(-\phi_1(x_i)) \\ \vdots \\ \vdots \\ 0=\dfrac{\partial S}{\partial\omega_n}=\displaystyle\sum_{i=1}^{m}2(y_i-\omega_0\phi_0(x_i)-\omega_1\phi_1(x_i)\cdots\cdots-\omega_n\phi_n(x_i))(-\phi_n(x_i)) \end{cases} \tag{8}$$

Rewrite the above equation and we get

$$\begin{cases} \omega_0\displaystyle\sum_{i=1}^{m}\phi_0(x_i)\phi_0(x_i)+\omega_1\sum_{i=1}^{m}\phi_0(x_i)\phi_1(x_i)+\cdots+\omega_n\sum_{i=1}^{m}\phi_0(x_i)\phi_n(x_i))=\sum_{i=1}^{m}y_i\phi_0(x_i) \\ \omega_0\displaystyle\sum_{i=1}^{m}\phi_1(x_i)\phi_0(x_i)+\omega_1\sum_{i=1}^{m}\phi_1(x_i)\phi_1(x_i)+\cdots+\omega_n\sum_{i=1}^{m}\phi_1(x_i)\phi_n(x_i))=\sum_{i=1}^{m}y_i\phi_1(x_i) \\ \vdots \\ \vdots \\ \omega_0\displaystyle\sum_{i=1}^{m}\phi_n(x_i)\phi_0(x_i)+\omega_2\sum_{i=1}^{m}\phi_n(x_i)\phi_1(x_i)+\cdots+\omega_n\sum_{i=1}^{m}\phi_n(x_i)\phi_n(x_i))=\sum_{i=1}^{m}y_i\phi_n(x_i) \end{cases} \tag{9}$$

The above equation can be written in a matrix form. That is

$$\begin{bmatrix} \displaystyle\sum_{i=1}^{m}\phi_0(x_i)\phi_0(x_i) & \cdots & \cdots & \displaystyle\sum_{i=1}^{m}\phi_0(x_i)\phi_n(x_i) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \displaystyle\sum_{i=1}^{m}\phi_n(x_i)\phi_0(x_i) & \cdots & \cdots & \displaystyle\sum_{i=1}^{m}\phi_n(x_i)\phi_n(x_i) \end{bmatrix}\begin{bmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \vdots \\ \omega_n \end{bmatrix}=\begin{bmatrix} \displaystyle\sum_{i=1}^{m}\phi_0(x_i)y_i \\ \displaystyle\sum_{i=1}^{m}\phi_1(x_i)y_i \\ \vdots \\ \vdots \\ \displaystyle\sum_{i=1}^{m}\phi_n(x_i)y_i \end{bmatrix} \tag{10}$$

In order to get the coefficients, $\omega_0,\omega_1,\omega_2,\cdots,\omega_n$, the linear simultaneous equations of Eq.

(10) have to be solved, which will take a lot of computing time.

There are several orthogonal functions such as Fourier Series, Bessel Function, Legendre Polynomials, and Chebyshev Polynomials. Each of them has its application domain in function approximation. For example, Fourier series is suited to approximate sine/cosine functions. However, it does not have the property of completeness. In other words, there is no guarantee that the approximation of function will be converged. Here Chebyshev polynomials of the first kind is chosen as the orthogonal functions because they have recursive property and completeness property at the boundary points of their definition intervals. The recursive property results from an expansion term being determined by its previous two expansion terms and it is desired to generate the structure of the ONN with the same processing elements.[11]

According to the theory of orthogonal functions[7,12], Chebyshev polynomials of the form $\tilde{\Phi}_n(x)$, $n \geq 1$, have the property that

$$\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} \left|\tilde{\Phi}_n(x)\right| \leq \max_{x\in[-1,1]} \left|P_n(x)\right|, \text{ for all } P_n \in \tilde{\prod}_n \tag{11}$$

where $$P_n(x) = \sum_{k=0}^{n} f(x_k) \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

denotes the Lagrange interpolating polynomial. Equation (11) is used to determine the interpolating nodes to minimize the error in Lagrange interpolation. In Lagrange interpolation, suppose that $x_0, x_1, \ldots, x_n$ are distinct numbers in the interval [-1,1] and function $f(x)$ has n+1 continuous derivatives in the same interval. Then, for each $x \in$ [-1,1], a number $\xi(x)$ exists in (-1,1), which satisfies the following equation.

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots\cdots(x - x_n) \tag{12}$$

There is no control over $\xi(x)$. To minimize the expected error by shrewd placement of the nodes $x_0, x_1, \ldots, x_n$ would be equivalent to finding the values of $x_0, x_1, \ldots, x_n$ to minimize the magnitude of $\left|(x - x_0)(x - x_1)\cdots\cdots(x - x_n)\right|$.

Since $(x - x_0)(x - x_1)\cdots\cdots(x - x_n)$ is a monic polynomial of degree (n+1), the minimum value is obtained if and only if

$$(x - x_0)(x - x_1)\cdots\cdots(x - x_n) = \tilde{\Phi}_{n+1}(x) \tag{13}$$

where x_k, k = 0, 1, ..., n, is chosen to be the (k+1)$^{\text{th}}$ zero of $\tilde{\Phi}_{n+1}(x)$. That is, the value of Eq. (13) will be minimized if x_k is chosen to be

$$x_{k+1} = \cos\left(\frac{2k+1}{2(n+1)}\pi\right) \tag{14}$$

in the interval [-1,1] will satisfy

$$\frac{1}{2^n} = \max_{x\in[-1,1]}\left|(x-\bar{x}_1)\cdots(x-\bar{x}_{n+1})\right| \le \max_{x\in[-1,1]}\left|(x-x_0)\cdots(x-x_n)\right| \tag{15}$$

Therefore, Eq. (10) is simplified as shown below.

$$\begin{bmatrix} \sum_{i=1}^{m}\phi_0^2(x_i) & 0 & \cdots & 0 \\ 0 & \sum_{i=1}^{m}\phi_1^2(x_i) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sum_{i=1}^{m}\phi_n^2(x_i) \end{bmatrix} \begin{bmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m}\phi_0(x_i)y_i \\ \sum_{i=1}^{m}\phi_1(x_i)y_i \\ \vdots \\ \vdots \\ \sum_{i=1}^{m}\phi_n(x_i)y_i \end{bmatrix} \tag{16}$$

Solve Eq. (16) and the coefficients of the Chebyshev polynomial will be

$$\omega_p = \frac{\sum_{i=1}^{m}\phi_p(x_i)y_i}{\sum_{i=1}^{m}\phi_p^2(x_i)} \quad , p=0,1,2\ldots\ldots n \tag{17}$$

If each variable is extended to *n* order of Chebyshev polynomials, then the input data for each variable is the *n*+1 roots of the *n* order Chebyshev polynomials. Therefore, there are $(n+1)^r$ input data to solve the $(n+1)^r$ coefficients of the *r*-variables of Chebyshev polynomials. Then the coefficients ω of each Chebyshev polynomials term can express as

$$\omega_{ij\cdots r} = \frac{\sum_{1i=1}^{m}\sum_{2i=1}^{m}\cdots\sum_{ri=1}^{m}\phi_i^2(x_{1i})\phi_j^2(x_{2i})\cdots\phi_r^2(x_{ri})\cdot y_{1i2i\cdots ri}}{\sum_{1i=1}^{m}\sum_{2i=1}^{m}\cdots\sum_{ri=1}^{m}\phi_i\phi_j\cdots\phi_r} \tag{18}$$

where *i*=1…*n*, *j*=1…*n*, *r*=1…*n*, and m is the number of pair data.

Ⅲ. Approximation of Functions

In this section, we will try to solve the coefficients of Chebyshev polynomials for function approximation and compare the result with that of NURBS method. For Chebyshev polynomials, the independent variable is defined in the interval [-1, 1]. However, the function to be approximated is usually not defined in the same interval. Here by Lagrange's interpolation method [12,13] is used to map the range of the input data to the interval [-1, 1].

First we arrange the input data from small to big. If the data are defined in the interval [a, b], they are mapped to the interval [-1, 1] by the following equation.

$$x_i^{'} = \frac{2(x_i - a)}{b - a}, i=1,2\ldots\ldots m \tag{19}$$

According to Eq. (15), we choose the Chebyshev polynomial independent variable $\bar{x}_i$ to

$$\bar{x}_i = \cos\left(\frac{2i+1}{2(n+1)}\pi\right), \ i=0,1,2,\ldots\ldots,n \tag{20}$$

By Lagrange's interpolation method, $\bar{y}_i$ of independent variable $\bar{x}_i$ can be derived from the selected data (x_1', y_1), (x_2', y_2), (x_3', y_3), …, (x_m', y_m). They are

$$\bar{y}_i = \sum_{k=1}^{m}\left(\prod_{\substack{j=1 \\ j\neq k}}^{m}\frac{\bar{x}_i - x_j^{'}}{x_k^{'} - x_j^{'}}\right)y_k, \quad i=0,1,2\ldots\ldots n \tag{21}$$

The data $(\bar{x}_i, \bar{y}_i)$ can be brought into Eq. (17), to get the Chebyshev polynomial, which is

$$Y(\bar{x}) = \omega_0\Phi_0(\bar{x}) + \omega_1\Phi_1(\bar{x}) + \cdots\cdots + \omega_n\Phi_n(\bar{x}) \tag{22}$$

where $\omega_0 = \frac{1}{n+1}\sum_{k=0}^{n}\bar{y}_k$ (23)

$$\omega_i = \frac{2}{n+1}\sum_{k=0}^{n}\Phi_i(\bar{x}_k)\cdot\bar{y}_k, \quad i=1,2\ldots\ldots n \tag{24}$$

By Eq. (19), we change the function $Y(\bar{x})$ to the original polynomial *Y(x)*

$$Y(x) = Y\left[\frac{2(x-a)}{b-a} - 1\right] \tag{25}$$

Ⅳ. Example of Surface Rebuilding

In design and manufacturing, surface modeling is frequently employed to describe objects precisely and accurately. The generation of a surface usually depends on quantitative data, such as a set of points with known coordinates. In formulating a surface, the designer must have the flexibility to use data a simple form. The choice of the surface form depends on the application. The numerical method and the NURBS surface are implemented herein.

Fig. 1 presents the data that describe the surface of a “human face” formed from

100x100 points with known x, y and z coordinates. The real curve shown in Fig. 2 is the curve x=73.5 in Fig. 1. Ten control points obtained using Eq. (20) are considered to determine the coefficients of the orthogonal function with order ten and thus approximate the curve. The dotted line in Fig. 2 is generated using the orthogonal function. The small circle represents the y coordinates often control points. Clearly, dot line does not approximate the real line closely, because an orthogonal function must be of high order to approximate the real curve. The fewer points chosen are can be reduced and calculated time, but the more errors caused will be. The dotted line in Fig. 3 is generated from an orthogonal function of order 20 and twenty control points by trial and error method. It is almost exactly coincides with the real line.

These results indicate that an orthogonal function determined by 10x10 control points in x and y coordinates cannot build a surface that closely matches the original surface in Fig. 1. Figure 4 shows the surface generated by the orthogonal function determined by 10X10 control points. Figure 5 presents the normal deviation between Figs. 1 and 4. The errors associated with most points are between 0.3 and -0.3 except associated with those points near boundary. Figure 6 shows the surface generated using the orthogonal function determined by 20X20 control points. The surface in Fig. 6 is much closer than that in Fig. 4 to that in Fig. 1. Figure 7 is the normal deviation between figure1 and figure 6. Most normal error are between 0.1 and -0.1 except points near boundary effect by the discontinuous function

Figure 8 shows the NURBS surface determined by the same 100X100 points as in Fig. 1; the orders of u and v are both four. Table 1 presents the mean of the absolute errors and root-mean-square normalized errors among Figs. 5, 7 and 9. The root-mean-square normalized errors are defined as:

$$\mathrm{E}_{\mathrm{ms}} = \frac{1}{\mathrm{N}}\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{N}}(p_{im} - p_{iB})^{2}} \tag{26}$$

where N denotes the number of original measured data points. P_{im} is the originally measured data point. P_{iB} is the relative data point of P_{im} on the generalized.

The use of the orthogonal polynomial function to rebuild a free form surface requires fewer points and yields smaller error. Comparing the performance of the orthogonal function and of NURBS reveals that the NURBS requires more points to build the surface and is not affected by the boundary in Fig.1.

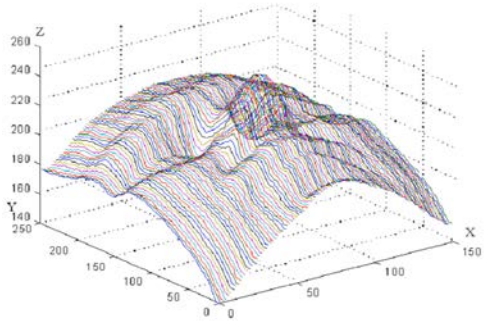

Fig. 1. The surface formed by 100X100 points

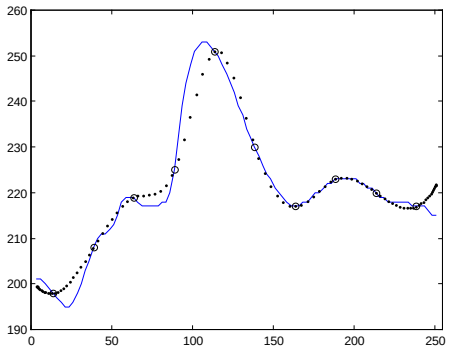

Fig. 2. Dotted line is the orthogonal function of order 10 , circle is the control points, the line curve is at x=73.5 of Fig. 1

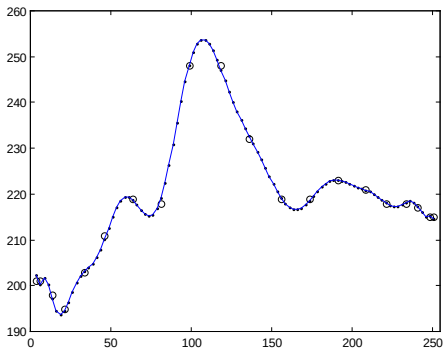

Fig. 3. The orthogonal function of order 20, dotted line has already overlapped with the curve

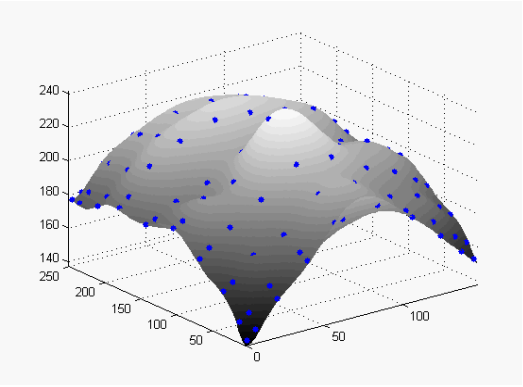

Fig. 4. The surface built by 10X10 control points

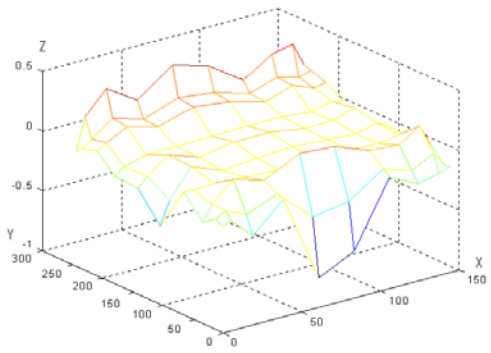

Fig. 5. The approximation error of 10X10 points

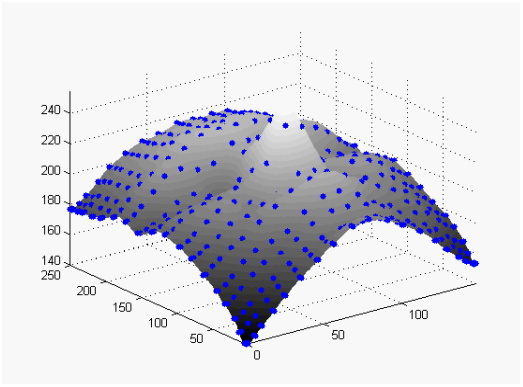

Fig. 6. The surface built by 20X20 control points

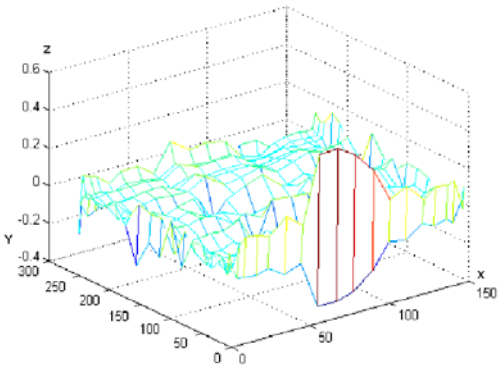

Fig. 7. The approximation error of 20X20 points

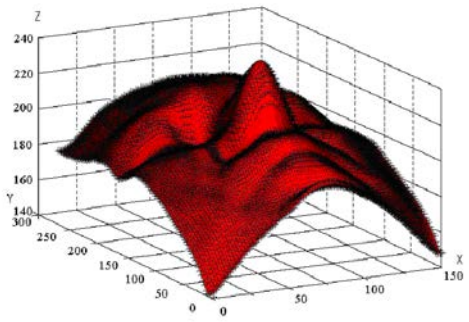

Fig. 8. The 100X100 points built surface by NURBS

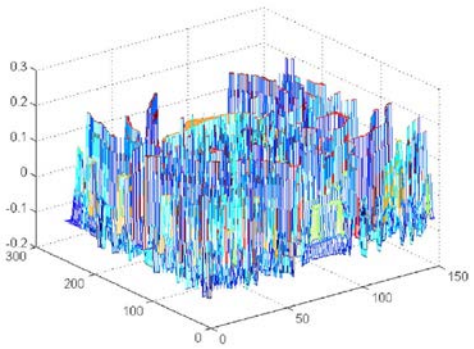

Fig. 9. The approximation error of 100X100 points

Table 1. Error comparisons of Figs. 5, 7 and 9.

	10X10 elements	20X20 elements	*NURBS*
The mean of the absolute value error	0.24183	0.10492	*0.11366*
Root-mean-square normalized error	*0.25614*	*0.09548*	*0.09134*

Ⅴ. Conclusion

This study presented a numerical method for determine the coefficients of an orthogonal function, which is applied to approximate high-order curves and surfaces. The proposed numerical method efficiently derives the coefficients. It requires only specified data sets to calculate the coefficients of the orthogonal function. The desired data set is determined by Lagrange interpolation method and a least square method is presented to determine the exact coefficients from linear simultaneous equations. This study uses orthogonal polynomial functions to reduce the amount of data to rebuild the free-form surface more quickly and precisely. The orthogonal polynomial function appears to have advantages over the NURBS method, as it requires fewer control points and generates surfaces more efficiently. However, its approximation error is larger than that of the NURBS method on the boundary of the production, because function is discontinuous on the boundary. Using more control points to determine the orthogonal function can reduce the approximation error.

Ⅵ.Reference

1. J. Barhak and A. Fischer, “Adaptive Reconstruction of Freeform Objects with 3D SOM Neural Network Grids” ,*Computer and Graphics*, No. 26, pp.745-751, 2002.
2. L. Dang, F. Kong, “Facial Feature Point Extraction Using A New Improved Active Shape Model, ” in Proc. IEEE International Congress on Image and Signal Processing , CISP, Vol. 2, pp. 944-948,2010.
3. O. Ayhan, B. Abaci and T. Akgul; “Improved Active Shape Model for Variable Illumination Conditions, ” in Proc. IEEE International Workshop on Multimedia Signal Processing, MMSP, pp. 322-327,2013.
4. H.J. Pahk, Y.H. Kim, Y.S. Hong and S.G. Kim, “Development of Computer aided Inspection System with CMM for Integrated Mold Manufacturing,” *Annals of the CIRP*, Vol.42, No.1, PP.557-560, 2003.
5. H.P. Huang and R.R. Chans, “Unstable Back- propagation Method in Neural Networks A Remedy,” *The Second International Conference on Automation Technology*, Vol.1, pp.249-255, 1992.
6. S.S. Yang and C.S. Tseng, “An Orthogonal Neural Network for Function Approximation,” *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol.26, No.5, pp.779-785, 1996.
7. C.S. Tseng and S.S. Yang, “A New Orthogonal Neural Network,” *IEEE Conf. on Neural Network*, Perth, West Austral, Vol.1, pp.296-299, 1995.
8. C.C. Mai, C.S. Tseng and C.S. Chen, “An Orthogonal Neural Controller with the Application In A Ball Beam System Control,” *International Journal of Intelligent Control and Systems*, Vol.3, No.2, pp.223-236,1999.
9. C.F. Sher, C.S. Tseng and C.S. Chen, “On the Properties and Performance of Orthogonal Neural Network in Function Approximation,” *International Journal of Intelligent Systems*, Vol.16, No.12, pp.1377-1392, 2001.
10. R. Courant and D. Hilbert, *Methods of Mathematical Physics*, 3rd.ed. The Syndics of The Cambridge University Press, 1956.
11. F.B.Hildebrand, *Advance Calculus for Applications*, Englewood Cliffs, NJ, 1976.
12. R.L.Burden and J.D.Faires, *Numerical Analysis*, PWS publishing Company, 1993.
13.E.Isaacson and H.B.Keller, *Analysis of Numerical Methods*, John Wiley & Sons,1966.
14.S. S. Yeh and H. C. Su, "Implementation of online NURBS curve fitting process on CNC machine," International Journal Advanced Manufacturing Technology, vol. 40, pp. 531-540, 2009.
15.D. S. Du, Y. D. Liu, C. L. Yan and C. G. Li, "An accurate adaptive parametric curve interpolator for NURBS curve interpolation," The International Journal of Advanced Manufacturing Technology, vol. 32, pp.999-1008,2007.
16.C. S. Chen, C. W. Chou, Y. S. Lai, “Real-time coplanar NURBS curves fitting in motion controller,” Advanced Materials Research, Vol. Materials, Mechantronics and Automation

II, pp. 105-111, 2013 .

利用柴比雪夫多項式數值法建構物體曲面

陳振山
臺北城市科技大學機械系

摘 要

在逆向工程中,重建精確的曲面需處理大量的數據。 如此過多的數據無法應用於CAD/CAM系統,中。本研究利用正交多項式函數來重建曲面，利用較少的數據來快速重建曲面，表面誤差結果並與利用非均勻比例 B-函數曲線平面（NURBS）法則進行比較。所提出的數值法僅需特定資料點就可有效的導出正交多項式函數的係數，利用拉格南吉內插法可得特定資料點，經由最小平方法求解線性聯立方程式獲得正確係數值。結果顯示利用正交多項式函數由少數資料點重建的連續函數曲面，並不亞於經由非均勻比例 B-函數曲線平面（NURBS）法則，運算龐大資料點所作的結果。

關鍵字：NURBS，正交函數，逆向工程，曲面重建