

# Chapter 2

## A Second-Order Algorithm for Curve Parallel Projection on Parametric Surfaces

Xiongbing Fang and Hai-Yin Xu

**Abstract** A second-order algorithm is presented to calculate the parallel projection of a parametric curve onto a parametric surface in this chapter. The essence of our approach is to transform the problem of computing parallel projection curve on the parametric surface into that of computing parametric projection curve in the two-dimensional parametric domain of the surface. First- and second-order differential geometric characteristics of the parametric projection curve in the parametric domain of the surface are firstly analyzed. A marching method based on second-order Taylor Approximation is formulated to calculate the parametric projection curve. A first-order correction technique is developed to depress the error caused by the truncated higher order terms in the marching method. Several examples are finally implemented to demonstrate the effectiveness of the proposed scheme. Experimental results indicate that both the computational efficiency and accuracy of the presented method have dominant performance as compared with the first-order differential equation method.

### 2.1 Introduction

Curves on a surface have a wide range of applications in the fields of Computer Graphics, Computer-Aided Geometric Design, Computer Animation, CNC, etc. For instance, curves on a surface can be used for surface trimming [1], surface blending [2], NC tool path generation [3, 4], and so on. According to the designing manner, curves on a surface can be the intersection curve of two surfaces [4],

---

X. Fang (✉)

China Ship Development and Design Center, Wuhan 430064, China  
e-mail: [fangxb2013@sina.cn](mailto:fangxb2013@sina.cn)

H.-Y. Xu

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

the offset of a given curve on a surface [3], the projection curve of a spatial curve onto a surface [5–9], the image of a curve in the parametric domain of a parametric surface [1, 2], or the fitting curve of a sequence of points lying on a surface [10].

In this chapter, we focus on computation of parallel projection of a parametric curve onto a parametric surface. Presently, there are two ways to do this problem. One is the first-order differential equation method [6] and the other is discrete method.

For the problem of calculating the parallel projection of parametric curves onto parametric surfaces, Wang, et al. transformed the condition of parallel projection into a system of differential equations and then formulated the problem as a first-order initial value problem. Numerical methods such as Runge–Kutta and Adams–Bashforth can be utilized to solve the initial value problem to generate a sequence of points. Under this transformation, difficulties lie in the choice of an accurate initial value and the stability of the adopted numerical method. For the discrete method, the parametric curve should first be discretized into a series of points. Parallel projections of these separated points can be calculated through the technique of intersection of a line with a surface. Usually, the computational efficiency of the discrete method depends on that of the intersection algorithm. Though the current projected point can be taken as the initial value of the next iteration, efficiency of the discrete method is generally slow as it does not fully utilize the differential geometric properties of both the parametric curve and the projection curve. The projection curve can be constructed by fitting the projected points generated by the two aforementioned types of approaches.

A second-order algorithm is put forward for tracing the parallel projection of a parametric curve onto a parametric surface. Experimental results show that the proposed scheme has dominant performance in both efficiency and computational accuracy as compared with Wang’s approach [6]. The rest of the chapter is organized as follows. An overview for our approach is presented in the next section. A second-order technique with error adjustment for tracing the projection curve is given in Sect. 2.3. Implementation and experimental results of our approach are carried out in Sect. 2.4 and we conclude the chapter in Sect. 2.5.

## 2.2 Overview

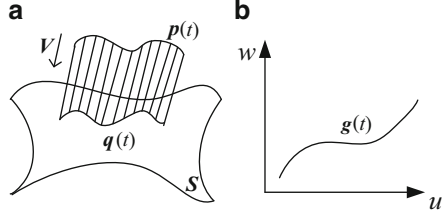
A 3D parametric curve  $\mathbf{p}(t)$  and a parametric surface  $S(u, w)$  given in Fig. 2.1a are represented as

$$\mathbf{p}(t) = [x(t) \ y(t) \ z(t)] \quad (2.1)$$

and

$$S(u, w) = [x(u, w) \ y(u, w) \ z(u, w)] \quad (2.2)$$

**Fig. 2.1** Parallel projection of 3D curve onto surface



respectively. Suppose  $p$  is a point on the curve  $p(t)$  and the  $q$  is the corresponding point generated by projecting the point  $p$  along with the direction  $V$  onto the surface  $S$ . While the point  $p$  moving along the curve  $p(t)$ , its parallel projection point  $q$  also moves along a curve on the surface, which is termed as the parallel projection curve of  $p(t)$  along with the direction  $V$  onto the surface  $S$ . According to the definition of parallel projection, one has

$$(q(t) - p(t)) \times V = 0 \quad (2.3)$$

where “ $\times$ ” denotes the cross product of two vectors. As the point lies on the parametric surface  $S(u, w)$ , the curve  $q(t)$  can be represented as  $q(t) = [x(u(t), w(t)), y(u(t), w(t)), z(u(t), w(t))]$ .

From the above analysis, there is a curve  $g(t) = [u(t), w(t)]$  in the parametric domain of the surface (please see Fig. 2.1b), which has a one-to-one corresponding relationship with the parallel projection curve on the surface. Thus a one-to-one corresponding relationship exists between the 3D curve  $p(t)$  and the 2D curve  $g(t)$ . For the convenience of description, we call  $g(t)$  as parametric projection curve in the remainder.

In this chapter, we transform the problem of computing parallel projection curve on parametric surface into the one of tracing parametric projection curve in 2D  $u$ – $w$  parametric domain. In Sect. 2.3.1, the first- and second-order differential quantities of the parametric projection curve are analyzed. A second-order iteration method for marching a series of points on the parametric projection curve based on Taylor Approximation is established and two methods for choosing the iterative step size are developed in Sect. 2.3.2. Considering the error caused by truncated higher order terms, a simple and efficient way to decrease the error is put forward in Sect. 2.3.3.

## 2.3 Parallel Projection Marching

### 2.3.1 The First- and Second-Order Differential Quantities of the Parametric Projection Curve

In order to calculate the first-order differential quantities, i.e.,  $u_t$  and  $w_t$ , of the parameters  $u$  and  $w$  with respect to the parameter  $t$  of the 3D curve, differentiating Eq. (2.3) with respect to  $t$  produces

$$(\mathbf{S}_u \times \mathbf{V})u_t + (\mathbf{S}_w \times \mathbf{V})w_t = \mathbf{p}_t \times \mathbf{V} \quad (2.4)$$

Taking the cross product of the  $\mathbf{S}_w$  and both sides of the Eq. (2.4), one has

$$([\mathbf{S}_u \times \mathbf{V}] \cdot \mathbf{S}_w)u_t = [\mathbf{p}_t \times \mathbf{V}] \cdot \mathbf{S}_w \quad (2.5)$$

Substituting  $[\mathbf{S}_u \times \mathbf{V}] \cdot \mathbf{S}_w = -(\mathbf{S}_u \times \mathbf{S}_w) \cdot \mathbf{V}$  into Eq. (2.5), we have

$$u_t = \frac{-\mathbf{L} \cdot \mathbf{S}_w}{[\mathbf{S}_u \times \mathbf{S}_w] \cdot \mathbf{V}} \quad (2.6)$$

where  $\mathbf{L} = \mathbf{p}_t \times \mathbf{V}$ . Similarly, dot-multiplying Eq. (2.4) by  $\mathbf{S}_u$  gives

$$w_t = \frac{\mathbf{L} \cdot \mathbf{S}_u}{[\mathbf{S}_u \times \mathbf{S}_w] \cdot \mathbf{V}} \quad (2.7)$$

One can continue to differentiate Eq. (2.3) with respect to the parameter  $t$

$$\left(\frac{d\mathbf{S}_u}{dt} \times \mathbf{V}\right)u_t + (\mathbf{S}_u \times \mathbf{V})u_{tt} + \left(\frac{d\mathbf{S}_w}{dt} \times \mathbf{V}\right)w_t + (\mathbf{S}_w \times \mathbf{V})w_{tt} = \mathbf{p}_{tt} \times \mathbf{V} \quad (2.8)$$

Since  $\frac{d\mathbf{S}_u}{dt} = \mathbf{S}_{uu}u_t + \mathbf{S}_{uw}w_t$  and  $\frac{d\mathbf{S}_w}{dt} = \mathbf{S}_{uw}u_t + \mathbf{S}_{ww}w_t$ , Eq. (2.8) can be rewritten as

$$\begin{aligned} & \left[ \mathbf{S}_{uu}(u_t)^2 + 2\mathbf{S}_{uw}u_tw_t + \mathbf{S}_{ww}(w_t)^2 \right] \times \mathbf{V} + (\mathbf{S}_u \times \mathbf{V})u_{tt} + (\mathbf{S}_w \times \mathbf{V})w_{tt} \\ & = \mathbf{p}_{tt} \times \mathbf{V} \end{aligned} \quad (2.9)$$

Dot-multiplying Eq. (2.9) by  $\mathbf{S}_w$  and  $\mathbf{S}_u$ , respectively, gives

$$\begin{cases} u_{tt} = \frac{-\mathbf{J} \cdot \mathbf{S}_w}{(\mathbf{S}_u \times \mathbf{S}_w) \cdot \mathbf{V}} \\ w_{tt} = \frac{\mathbf{J} \cdot \mathbf{S}_u}{(\mathbf{S}_u \times \mathbf{S}_w) \cdot \mathbf{V}} \end{cases} \quad (2.10)$$

where  $\mathbf{J} = [\mathbf{p}_{tt} - \mathbf{S}_{uu}(u_t)^2 - 2\mathbf{S}_{uw}u_tw_t - \mathbf{S}_{ww}(w_t)^2] \times \mathbf{V}$ , and  $u_t, w_t$  are computed by Eqs. (2.6) and (2.7), respectively.

### 2.3.2 Parametric Projection Curve Marching

In this section, we will use a second-order Taylor Approximation method to trace the parametric projection curve  $\mathbf{g}(t)$  to generate a series of points.

Let  $\mathbf{g}_i = (u_i, w_i)$  and  $\mathbf{g}_{i+1} = (u_{i+1}, w_{i+1})$  be the current and the next parametric projected point in the  $u$ - $w$  domain respectively. Then the position of point  $\mathbf{g}_{i+1}$  can be iteratively calculated through the current projected point and the first- and second-order differential quantities  $u_t, w_t, u_{tt}, w_{tt}$  at the point as follows

$$[u_{i+1}, w_{i+1}] = [u_i, w_i] + [u_t, w_t][k + 1/2]u_{tt}, w_{tt}k^2 \quad (2.11)$$

where  $k$  is a constant iteration step size.

If  $k = \Delta t$  is constant, one can utilize the Eq. (2.11) to get a sequence of points  $(u_i, w_i)$  on the parametric projection curve  $\mathbf{g}(t)$ , where  $i = 1, 2, \dots$ . However, the distribution of the projected points may not be well-proportioned. In order to fully use the differential geometric characteristics of  $\mathbf{g}(t)$  and  $\mathbf{p}(t)$ , we will give two ways to choose the iteration step size.

One can consider marching along the parametric projection curve based on a constant step size  $v_p$  along the 3D parametric curve. Suppose  $r$  is the arc-length parameter of the 3D parametric curve, one has

$$dr = \|x_t \ y_t \ z_t\|dt = \|\mathbf{p}_t\|dt \quad (2.12)$$

Marching along the 3D parametric curve uses

$$t_{i+1} = t_i + k \text{ with } k = \frac{1}{\|\mathbf{p}_t\|} v_p \quad (2.13)$$

in addition to using Eqs. (2.6), (2.7), (2.10) and (2.11).

For marching along the parametric projection curve based on constant step size  $v_q$  along the projection curve, one can write

$$ds = \|\mathbf{q}_t\|dt \quad (2.14)$$

where  $s$  is the arc-length parameter of the parallel projection curve and  $\mathbf{q}_t = \mathbf{S}_u u_t + \mathbf{S}_w w_t$ . Marching along the parallel projection curve uses

$$t_{i+1} = t_i + k \text{ with } k = \frac{1}{\|\mathbf{S}_u u_t + \mathbf{S}_w w_t\|} v_q \quad (2.15)$$

in addition to using Eqs. (2.6), (2.7), (2.10) and (2.11).

### 2.3.3 Error Adjustment

Considering the truncated higher order terms, the position of point computed through Eq. (2.11) may depart from the parametric projection curve  $\mathbf{g}(t)$  in  $u$ - $w$  domain. Hence, a first-order adjustment technique is presented to reduce the error.

Suppose  $\hat{\mathbf{g}}_{i+1} = (\hat{u}_{i+1}, \hat{w}_{i+1})$  be the parametric projection point computed by Eq. (2.11)

$$[\hat{u}_{i+1}, \hat{w}_{i+1}] = [u_i, w_i] + [u_t, w_t]k + 1/2[u_{tt}, w_{tt}]k^2$$

As the above iteration formula omits higher order terms, the image point of  $\hat{\mathbf{g}}_{i+1}$  may deviate from the parallel projection curve on the parametric surface  $\mathcal{S}$ , i.e.,

$$(\hat{\mathbf{q}}_{i+1} - \mathbf{p}_{i+1}) \times \mathbf{V} = \boldsymbol{\varepsilon} \neq \mathbf{0} \quad (2.16)$$

where  $\hat{\mathbf{q}}_{i+1} = \mathcal{S}(\hat{u}_{i+1}, \hat{w}_{i+1})$ ,  $\mathbf{p}_{i+1} = \mathbf{p}(t_{i+1})$ .

Let  $[\Delta u, \Delta w]$  be the correction vector, the corrected parametric projection point and parallel projection point be  $\mathbf{g}_{i+1} = (u_{i+1}, w_{i+1}) = (\hat{u}_{i+1} + \Delta u, \hat{w}_{i+1} + \Delta w)$  and  $\mathbf{q}_{i+1} = \mathcal{S}(u_{i+1}, w_{i+1})$ , respectively. According to the definition of parallel projection, one has

$$(\mathbf{q}_{i+1} - \mathbf{p}_{i+1}) \times \mathbf{V} = \mathbf{0} \quad (2.17)$$

In order to calculate the error adjustment vector  $[\Delta u, \Delta w]$ , we use a first-order Taylor formula to expand the point  $\mathbf{q}_{i+1}$  at  $\hat{\mathbf{g}}_{i+1}$ , i.e.,  $\mathbf{q}_{i+1} = \mathcal{S}(\hat{u}_{i+1}, \hat{w}_{i+1}) + \mathbf{S}_u \Delta u + \mathbf{S}_w \Delta w$ . Substituting  $\mathbf{q}_{i+1}$  into Eq. (2.17) produces

$$(\mathbf{S}_u \Delta u + \mathbf{S}_w \Delta w) \times \mathbf{V} = -\boldsymbol{\varepsilon} \quad (2.18)$$

Dot-multiplying Eq. (2.18) by  $\mathbf{S}_w$  and  $\mathbf{S}_u$  respectively, gives

$$\begin{cases} \Delta u = \frac{\boldsymbol{\varepsilon} \cdot \mathbf{S}_w}{(\mathbf{S}_u \times \mathbf{S}_w) \cdot \mathbf{V}} \\ \Delta w = \frac{-\boldsymbol{\varepsilon} \cdot \mathbf{S}_u}{(\mathbf{S}_u \times \mathbf{S}_w) \cdot \mathbf{V}} \end{cases} \quad (2.19)$$

Note that all variables in Eq. (2.19) are evaluated at point  $\hat{\mathbf{g}}_{i+1}$  and  $t_{i+1}$ . When the deviation  $\|\boldsymbol{\varepsilon}\|$  calculated by Eq. (2.16) is greater than a given threshold  $\alpha$ , i.e.,  $\|\boldsymbol{\varepsilon}\| > \alpha$ , one can adopt the error adjustment vector calculated by Eq. (2.19) to adjust the parametric projection point  $\hat{\mathbf{g}}_{i+1}$ . In the following, the quantities  $\|\boldsymbol{\varepsilon}\|$  and  $\alpha$  are called as projection error and projection error threshold, respectively.

Suppose  $\mathbf{g}_i$  and  $\mathbf{g}_{i+1}$  are two adjacent points after error correction lying on the parametric projection curve  $\mathbf{g}(t) = [u(t) \ w(t)]$ . One can use a line segment to connect the two points  $\mathbf{g}_i$  and  $\mathbf{g}_{i+1}$  as follows:

$$\begin{cases} u = t \ (u_i \leq t \leq u_{i+1}) \\ w = w_i + l(t - u_i) \end{cases} \quad (2.20)$$

where  $l = (w_{i+1} - w_i)/(u_{i+1} - u_i)$ . A  $G^0$  continuous parallel projection curve is acquired by substituting Eq. (2.20) into Eq. (2.2). One can also use the  $G^1$  approximation method [10] to fit the sequence of points  $S(u_i, w_i)$ ,  $i = 1, 2, \dots$ , to get a parallel projection curve with  $G^1$  or higher continuity.

## 2.4 Demonstrations

Our parallel projection algorithm framework is outlined in Sect. 2.4.1 and a number of examples are given to demonstrate the validity of our proposed method in Sect. 2.4.2. The examples make use of cubic NURBS curves and bicubic NURBS surfaces. In this context, these are viewed as parametric curves and surfaces, respectively.

### 2.4.1 Outline of Our Parallel Projection Algorithm

Given the control points and knot vectors for a NURBS curve and a NURBS surface, we need to compute a series of parallel projection points on the surface.

---

#### Algorithm 1. Parallel projection of parametric curves onto parametric surfaces

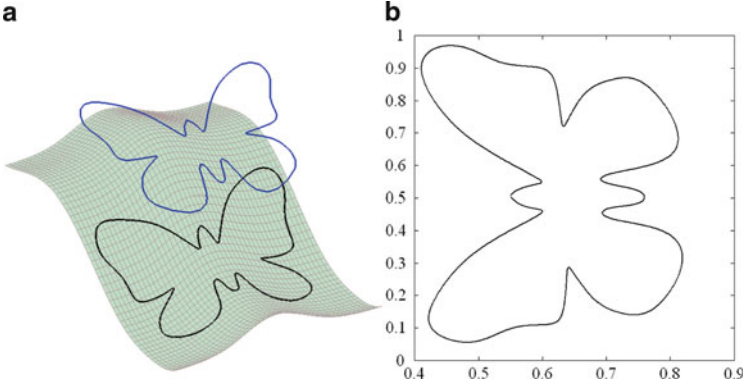
---

**Input:** Parametric curve  $p(t)$ , parametric surface  $S(u, w)$ , the initial projection point  $p_0 = p(t_0)$  and its parallel projection point  $q_0 = S(u_0, w_0)$  on surface  $S$ , projection error threshold  $\alpha$

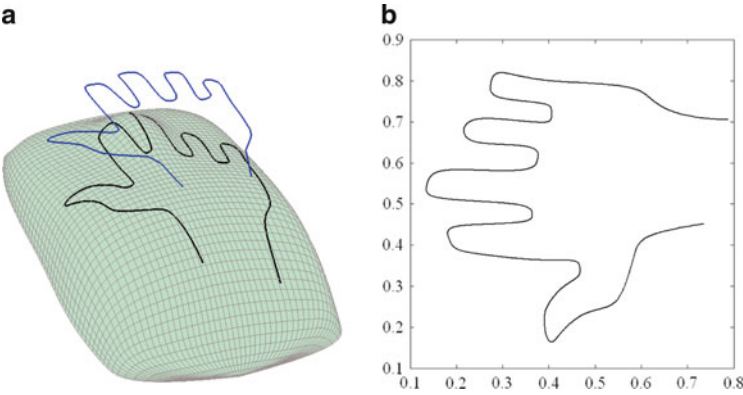
**Output:** A series of parallel projection points  $q_i = S(u_i, w_i)$  ( $i=1, 2, \dots$ ) on surface  $S$ .

**Algorithm Description:**

1.  $t_i = t_0$ ,  $(u_{i+1}, w_{i+1}) = (u_0, w_0)$ ,  $p_i = p_0$ ,  $q_i = q_0$ ;
  2. **do** {
  3.   Compute step size  $k$ ,  $t_{i+1} = t_i + k$  and  $p(t_{i+1})$ ;
  4.   Compute  $(u_{i+1}, w_{i+1})$  and point  $q_{i+1} = S(u_{i+1}, w_{i+1})$ ;
  5.   Compute the projection error  $\varepsilon$ ;
  6.   **while**(  $\|\varepsilon\| > \alpha$  ) **do**{
  7.     Compute the adjustment vector  $[\Delta u, \Delta w]$  with equation (19);
  8.     Compute  $(\hat{u}_{i+1}, \hat{w}_{i+1}) = (u_{i+1}, w_{i+1}) + (\Delta u, \Delta w)$  and  $\hat{q}_{i+1} = S(\hat{u}_{i+1}, \hat{w}_{i+1})$ ;
  9.     Renew  $(u_{i+1}, w_{i+1})$ :  $(u_{i+1}, w_{i+1}) = (\hat{u}_{i+1}, \hat{w}_{i+1})$  and  $q_{i+1} = \hat{q}_{i+1}$ ;
  10.    Recalculate the projection error  $\varepsilon$  with new parameters  $(u_{i+1}, w_{i+1})$  and  $q_{i+1}$ ;
  11.   **end while**
  12.   Renew  $t$ :  $t_i = t_{i+1}$ ,  $(u_i, w_i) = (u_{i+1}, w_{i+1})$  and  $q_i = q_{i+1}$ ;
  13.   Output the parallel projection point  $q_i$ ;
  14. } **while**( $p(t_i)$  is at the end of the 3D parametric curve  $p(t)$ )
-



**Fig. 2.2** Parallel projection example 1. (a) A butterfly curve onto an undeformed surface, and (b) parametric projection curve in  $u$ - $w$  domain



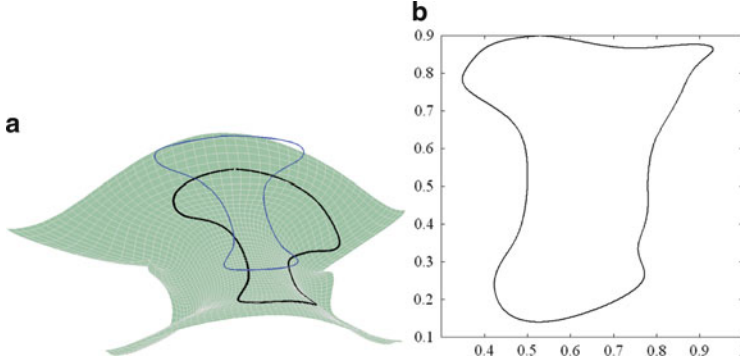
**Fig. 2.3** Parallel projection example 2. (a) A hand-form curve onto a mouse surface, and (b) parametric projection curve in  $u$ - $w$  domain

### 2.4.2 Examples

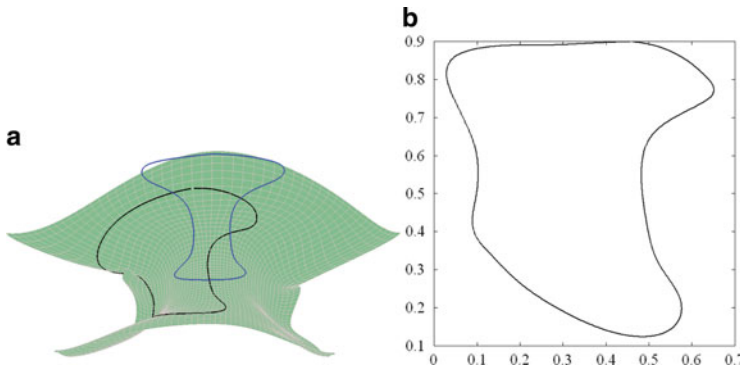
In this section we firstly give four examples (please see Figs. 2.2, 2.3, 2.4, and 2.5) obtained by the **Algorithm 1**. Further detailed comparisons of our scheme with Wang's first-order differential equation method [6] are carried out to testify the computational efficiency and accuracy of our methods (please see Tables 2.1, 2.2, 2.3, and 2.4). All the examples are implemented with Matlab, and run on a PC with 2.80 GHz CPU and 1 GB memory.

Figure 2.2a shows an example of projecting a closed butterfly curve onto an undeformed surface. The corresponding parametric projection curve in the  $u$ - $w$  parametric domain is shown in Fig. 2.2b. The second example shows the projection of a small hand-form curve onto a mouse surface. The hand-form curve and its parallel





**Fig. 2.4** Parallel projection example 3. (a) A closed NURBS curve onto a ridge surface, and (b) parametric projection curve in  $u$ - $w$  domain



**Fig. 2.5** Parallel projection example 4. (a) A closed NURBS curve onto a ridge surface, and (b) parametric projection curve in  $u$ - $w$  domain

projection curve on the NURBS surface is shown in Fig. 2.3a. Figure 2.3b denotes the parametric projection curve in the  $u$ - $w$  parametric domain.

Figures 2.3a and 2.4a are examples of projecting a same closed NURBS curve onto a same complex ridge NURBS surface along two different directions. The corresponding parametric projection curves in the  $u$ - $w$  parametric domain are shown in Figs. 2.3b and 2.4b.

For the parallel projection examples 1 and 2, we compared our method with Wang's first-order differential equation method [6] focusing on computational accuracy and efficiency (please see Tables 2.1, 2.2, 2.3, and 2.4). For the sake of fairness, the step size is specified by constant parametric increment  $\Delta t$  and the tolerance  $\alpha$  of the parallel projection error  $\epsilon$  is set as constant ( $\alpha$  is set as  $1.0e-009$  for all the comparisons). Moreover, the initial values for the two methods are the same. As the parallel projection error of Wang's method may overrun the error threshold  $\alpha$ , it is solved with the classical fourth-order Runge-Kutta method and our error adjustment technique deduced in Sect. 2.3.3.

**Table 2.1** Comparison of accuracy: projection of a butterfly curve onto an undee surface

Step size, $\Delta t$	APE of Wang's method (no error adjustment)	APE of our method	
		(no error adjustment)	(error adjustment)
0.05	2.2598e-009	1.3133e-004	1.8662e-012
0.04	4.7598e-010	8.3957e-005	4.8905e-013
0.03	2.0282e-007	4.9389e-005	8.6769e-014
0.02	6.7689e-011	2.0942e-005	7.6519e-015

**Table 2.2** Comparison of accuracy: projection of a hand-form curve onto a mouse surface

Step size, $\Delta t$	APE of Wang's method (no error adjustment)	APE of our method	
		(no error adjustment)	(error adjustment)
0.05	2.8146e-004	4.1585e-002	7.7697e-011
0.04	1.2407e-004	2.4439e-002	2.6262e-011
0.03	4.1559e-005	1.2371e-002	5.4997e-012
0.02	8.4125e-006	4.8313e-003	1.4454e-012
0.01	5.0854e-007	1.0423e-003	4.5111e-013

**Table 2.3** Comparison of efficiency: projection of a butterfly curve onto an undee surface

Step size, $\Delta t$	Wang's method (error adjustment)		Our method (error adjustment)	
	APE	CPU time (s)	APE	CPU time (s)
0.05	3.0445e-010	30.1094	1.8662e-012	27.2031
0.04	2.1330e-010	37.2500	4.8905e-013	33.0781
0.02	6.7689e-011	73.0781	7.6519e-015	66.1406

For comparison of accuracy of the two methods, the first-order method is realized just by using the classical four-order Runge–Kutta method, while our method is implemented in two ways that one is carried out with error adjustment and the other is without correction. From the results in Tables 2.1 and 2.2 (Note that the symbol “APE” in Tables 2.1, 2.2, 2.3, and 2.4 denotes average projection error), we can see that our method is much lower than Wang’s in precision when implemented without error correction. After the error adjustment, the accuracy of our method is much finer than Wang’s. From the results of large numbers of experiments, we found that the CPU time of our method with error adjustment is less than that of Wang’s as projecting the same number of points onto a surface, while the precision of ours still has predominant performance.

Tables 2.3 and 2.4 are the comparison results of accuracy of our method with Wang’s, which are gained under the conditions of same initial values, iteration step size, and error threshold. Both of the two methods adopted the error correction technique given in Sect. 2.3.3. From the column of CPU time in Tables 2.3 and 2.4, we can see that the efficiency of our method is about 1.1 times of Wang’s. On the

**Table 2.4** Comparison of efficiency: projection of a hand-form curve onto a mouse surface

Step size, $\Delta t$	Wang's method (error adjustment)		Our method (error adjustment)	
	APE	CPU time (s)	APE	CPU time (s)
0.05	4.5873e-010	30.5469	7.7697e-011	26.9219
0.03	1.7461e-010	49.7031	5.4997e-012	43.7813
0.01	3.9048e-010	145.3438	4.5111e-013	131.0313

other hand, from the comparisons of Tables 2.1, 2.2 and Tables 2.3, 2.4, one can also find that our first-order error adjustment approach could efficiently improve the computational precision of the Wang's method.

## 2.5 Conclusion

A second-order algorithm based on Taylor Approximation is proposed to compute the parallel projection of a parametric curve onto a parametric surface. Several examples are presented to demonstrate the effectiveness of the presented approach. Experimental results and comparisons indicate that the computational efficiency of our method is about 1.1 times of that of the first-order differential equation method and our method has superior performance in the computational accuracy.

## References

1. Sederberg, T. W., Finnigan, G. T., Li, X., Lin, H. W., & Ipson, H. (2008). Watertight trimmed NURBS. *ACM Transaction on Graphics*, 27(3), 79:1–79:8.
2. Chuang, J. H., Lin, C. H., & Hwang, W. C. (1995). Variable-radius blending of parametric surfaces. *The Visual Computer*, 11(10), 513–525.
3. Tam, H.-Y., Law, H. W., & Xu, H.-Y. (2004). A geometric approach to the offsetting of profiles on the three-dimensional surfaces. *Computer-Aided Design*, 36(10), 887–902.
4. Xu, H.-Y., Tam, H.-Y., Fang, X., & Hu, L. (2009). Quart-parametric interpolations for intersecting paths. *Computer-Aided Design*, 41(6), 432–440.
5. Pegna, J., & Wolter, F. E. (1996). Surface curve design by orthogonal projection of space curves onto free-form surface. *Journal of Mechanical Design*, 118(1), 45–52.
6. Wang, X., Wei, W., & Zhang, W.-Z. (2010). Projecting curves onto free-form surfaces. *International Journal of Computer Applications in Technology*, 37(2), 153–159.
7. Wang, X., An, L. L., Zhou L. S., & Zhang, L. (2010). Constructing  $G^2$  continuous curve on freeform surface with normal projection. *Chinese Journal of Aeronautics*, 23(1), 137–144.
8. Xu, H.-Y., Fang, X., Hu, L., Xiao F., & Li, D. (2010). An algorithm for curve orthogonal projections onto implicit surfaces. *Journal of Computer-Aided Design and Computer Graphics*, 22(12), 2103–2110.
9. Xu, H.-Y., Fang, X., Tam, H.-Y., Wu, X., & Hu, L. (2012). A second-order algorithm for curve orthogonal projection onto parametric surface. *International Journal of Computer Mathematics*, 89(1), 98–111.
10. Yang, Y.-J., Zeng, W., Yang, C.-L., Meng, X.-X., Yong J.-H., & Deng, B. (2012).  $G^1$  continuous approximate curves on NURBS surfaces. *Computer-Aided Design*, 44(9), 824–834.

Computer Engineering and Networking

Proceedings of the 2013 International Conference on

Computer Engineering and Network (CENet2013)

Wong, W.E.; Zhu, T. (Eds.)

2014, XXII, 1426 p. 627 illus. In 2 volumes, not available  
separately., Hardcover

ISBN: 978-3-319-01765-5