mathematics

Article

Point Orthogonal Projection onto a Spatial Algebraic Curve

Taixia Cheng [1,†], Zhinan Wu [2,†], Xiaowu Li [3,*,†] and Chan Wang [4,*,†]

1 Graduate School, Guizhou Minzu University, Guiyang 550025, China; chengtaixia@163.com
2 School of Mathematics and Computer Science, Yichun University, Yichun 336000, China; zhi_nan_7@163.com
3 College of Data Science and Information Engineering, Guizhou Minzu University, Guiyang 550025, China
4 School of Finance, Central University of Finance and Economics, Beijing 100081, China
* Correspondence: lixiaowu002@126.com (X.L.); wangchanist@126.com (C.W.); Tel.: +86-187-8613-2431 (X.L.); +86-136-8157-3387 (C.W.)
† These authors contributed equally to this work.

Received: 4 January 2020; Accepted: 25 February 2020; Published: 1 March 2020

Abstract: Point orthogonal projection onto a spatial algebraic curve plays an important role in computer graphics, computer-aided geometric design, etc. We propose an algorithm for point orthogonal projection onto a spatial algebraic curve based on Newton's steepest gradient descent method and geometric correction method. The purpose of Algorithm 1 in the first step of Algorithm 4 is to let the initial iteration point fall on the spatial algebraic curve completely and successfully. On the basis of ensuring that the iteration point fallen on the spatial algebraic curve, the purpose of the intermediate **for** loop body including Step 2 and Step 3 is to let the iteration point gradually approach the orthogonal projection point (the closest point) such that the distance between them is very small. Algorithm 3 in the fourth step plays an important double acceleration and orthogonalization role. Numerical example shows that our algorithm is very robust and efficient which it achieves the expected and ideal result.

Keywords: point orthogonal projection; the closest point; intersection; spatial algebraic curve; Newton's steepest gradient descent method; foot-point; geometric correction method

1. Introduction

Orthogonal projection is widely used and plays an important role in geometric modeling, computer graphics and computer aided geometric design. Pegna et al. [1] first proposed the concept of orthogonal projection, and discussed the calculation projecting problem of spatial parametric curve orthogonal projection onto a parametric surface and algebraic surface. The so-called orthogonal projection is to find a point on the curve such that the line segment connected to this point and the given point is perpendicular to the tangent line of the curve at this point. There is a close relationship between the orthogonal projection problem and distance projection problem [1]; therefore, the study of the distance projection problem is helpful for the study of the orthogonal projection problem to some extent.

Many scholars have done a lot of work in these two aspects. Hartmann [2] proposed a first-order tangent line method to calculate point orthogonal projection onto parametric curve and surface. For a few cases of non-convergence and as supplement and perfection of the first-order tangent line method [2], Liang et al. [3] and Li et al. [4] presented hybrid second-order method for orthogonal projection onto parametric curve and surface, respectively. Hu et al. [5] proposed a second-order geometric iterative algorithm with curvature information to approximate the orthogonal projection point of the given point on parametric curves and surfaces. Based on their work [5], Li et al. [6]

provided an improved method for point orthogonal projection onto general parametric surface such that the efficiency in [6] is greatly improved compared with the existing methods. Ma et al. [7] studied the problem for point orthogonal projection onto NURBS curves and surfaces for which they adopted four-step technique: subdividing curve or surface into curve segments or surface patches, finding out the relationship between the control polygon of curve segment or surface patch and the test point, candidate curve segment or surface patch and approximate projection points being found out by comparison and the final projective point being obtained by comparing the distance between the test point and these approximate projective points. Since the minimum distance between two geometries occurred between a pair of special points, they studied the minimum distance between various specific geometry by using the property and obtained some satisfactory results [8–10].

Since the algebraic curve and surface are difficult to control accurately, it is difficult to find the orthogonal projection point on the algebraic curve and surface. At the same time, geometric modeling, computer graphics, computer aided geometric design and many other problems can come down to point orthogonal projection onto curve and surface problems. Therefore, it is of great significance to study point orthogonal projection onto algebraic curves and surfaces.

Up to the present day, the study of point orthogonal projection onto algebraic curves and surfaces is mainly divided into three types: orthogonal projection onto planar algebraic curve, spatial algebraic curve and algebraic surface. Since algebraic curves and algebraic surfaces have no parametric control, it is not easy to solve such problems directly. However, many scholars turned such problems into root-finding problems. There are three types for point orthogonal projection onto planar algebraic curve: local method, global method and compromise method between these two methods. According to the most basic geometric characteristics, the problem of point orthogonal projection onto planar algebraic curve can be understood as solving the equation which the cross product of gradient $\nabla f(\mathbf{X})$ of the planar algebraic curve $f(\mathbf{X})$ at point $\mathbf{X}$ and the vector $\overrightarrow{\mathbf{PX}}$ is zero. The equation can be specifically expressed as the following,

$$\nabla f(\mathbf{X}) \times (\mathbf{P} - \mathbf{X}) = 0 \tag{1}$$

In fact, the geometric meaning is that the inner product of the tangent vector of the planar algebraic curve $f(\mathbf{X})$ at point $\mathbf{X}$ and the vector $\overrightarrow{\mathbf{PX}}$ is zero. Equation (1) can be transformed into the corresponding iterative formula of Newton's steepest gradient descent method.

$$\mathbf{X}_{n+1} = \mathbf{X}_n - (f(\mathbf{X}_n)/\langle \nabla f(\mathbf{X}_n), \nabla f(\mathbf{X}_n)\rangle)\nabla f(\mathbf{X}_n). \tag{2}$$

William et al. [11] used composed method including Lagrange multiplier and Newton's steepest gradient descent method to compute the orthogonal projection point on the planar algebraic curve for each test point. The characteristic of the method [11] is that it is fast but local and depends on the initial iterative point.

In addition to the Newton's local iterative method to solve the equation for point orthogonal projection onto the planar algebraic curve, the first global method converting into solving system of nonlinear equations is the Homotopy continuous method [12,13]. They used the most classical Homotopy transformation method,

$$\mathbf{H}(\mathbf{X}, t) = (1-t)\mathbf{P}(\mathbf{X}) + t\mathbf{Q}(\mathbf{X}), \qquad t \in [0,1], \tag{3}$$

where t is a continuous transformational parameter from 0 to 1, $\mathbf{P}(\mathbf{X}) = 0$ and $\mathbf{Q}(\mathbf{X}) = 0$ are original system of nonlinear equations to be solved and objective solution system of nonlinear equations, respectively. All isolated solutions of original system of nonlinear equations $\mathbf{P}(\mathbf{X}) = 0$ can be obtained by the numerical continuous Homotopy method [12,13], where all the isolated solutions of $\mathbf{P}(\mathbf{X}) = 0$ are exactly the same as the target solutions of system of nonlinear equations $\mathbf{Q}(\mathbf{X}) = 0$. The robustness of the numerical continuous Homotopy methods [12,13] with global convergence is expounded and verified by [14] and their high time consumption property is summarized in [15]. The major difficulty of Homotopy method is how to find a very satisfactory objective solutions of system of nonlinear equations $\mathbf{Q}(\mathbf{X}) = 0$.

Turning the projection problem into a resultant problem and then solving the resultants [16–19] is called the second global resultant method. Using the classical elimination theory, a system of two nonlinear equations with two variables can be transformed into a resultant polynomial with one variable being equivalent to the two original equations. The Sylvester's resultant and Cayley's statement of Bézout's method are the most classical resultant methods [16–19]. If the degree of the planar algebraic curve is no more than 5, all roots of the nonlinear polynomial equation formed by the resultant methods can be solved. However, if the degree of the planar algebraic curve is more than 5, it is very difficult to directly solve all roots of two-polynomial system with the resultant method.

The third global method is Bézier clipping technique [20–22]. In the first step, the nonlinear system of Equation (1) is turned into Bézier form with convex hull property. The rest of the treatment is exactly the same as that of Ma et al. [7] so we do not give a detailed description here. The superiority of global Bézier clipping method is that all roots can be solved, or all orthogonal projection points can be obtained this way. There are two disadvantages of the global clipping method: the first disadvantage is that it needs a lot of time to subdivide, seek, judge and compare; the second disadvantage is that Equation (1) is difficult to directly convert into Bernstein-Bézier form for a small number of planar algebraic curves. Of course, an indirect solution to the difficulty which the idea is come from reviewer's suggestion is that one can use a multivariate solver for this problem. Equation (7) gives a 3×3 algebraic system and its solution gives the stationary (candidate) points.

The first compromise method that is between local and global methods is proposed by Hartmann [2,23] which is composed of the geometric tangent property for computing orthogonal projection point. Continue to execute the iterative formula (2) until the iterative point falls on the planar algebra curve. Taking the iterative point fallen on the planar algebraic curve as being the initial iterative point, compute the foot point once by the iterative formula (4),

$$\mathbf{Q} = \mathbf{P} - \left(\langle \mathbf{P} - \mathbf{Y}_n, \nabla f(\mathbf{Y}_n) \rangle \;/\; \langle \nabla f(\mathbf{Y}_n), \nabla f(\mathbf{Y}_n) \rangle\right) \nabla f(\mathbf{Y}_n). \tag{4}$$

Repeat the above two steps until the foot point Q and the orthogonal projection point completely overlap. Unfortunately, for a small part of the planar algebraic curve, it is invalid to approach the orthogonal projective point with progressive tangent foot point. Based on Hartmann's work in [2,23], Li et al. [24] proposed an improved method for calculating footpoint which the effect is good if the test point is not far away, but the effect is not ideal if the test point is particularly far away. In addition, on the important roles and applications of the algebraic curves and algebraic surfaces, some introduction and explanation are presented in these papers [2,9,10,23,24]. Here we will not explain and elaborate upon them in detail.

The second compromise method is proposed by Redding [25] who used the osculating circle technique to accomplish point orthogonal projection onto the planar algebraic curve. The osculating circle technique mainly includes three steps: (1) Compute curvature at point on the planar algebraic curve and the corresponding radius and center of the curvature circle determined by curvature. (2) Get the footpoint $\mathbf{Q}$ intersected with the line segment formed by the test point and the center of the curvature circle and curvature circle. (3) Take the footpoint $\mathbf{Q}$ approximately as the point fallen on the planar algebraic curve. Repeatedly execute the above three steps until the foorpoint $\mathbf{Q}$ and the orthogonal projection point $\mathbf{P}_\Gamma$ completely coincide. Since there is a certain deviation in the third step of the osculating circle technique for footpoint $\mathbf{Q}$ approximating to the orthogonal projection point on the planar algebraic curve and the planar algebraic curve has no parametric control like parametric curve, the robustness of the osculating circle technique cannot be achieved very well. Greatly inspired by these works [2,23,25], Wu et al. [26] proposed an improved curvature circle algorithm for orthogonal projection onto a planar algebraic curve. Not only in robustness but also in efficiency, the Second Remedial Algorithm in [26] is very satisfactory and ideal.

The third compromise method is the circle shrinking method [27]. Repeatedly iterate the iterative formula (2) such that the iterative point can fall on the planar algebraic curve as much as possible. Draw a circle where the center and radius are the test point $\mathbf{P}$ and $\|\mathbf{P}-\mathbf{P}_\mathbf{I}\|$, respectively. Identify a point $\mathbf{P}^+$ on the circle by means of the mean value theorem and find the intersection named the current iterative point $\mathbf{P}_\mathbf{I}$ which is the intersection between the line segment $\overline{\mathbf{PP}^+}$ and the circle. Repeatedly run the above behavior until the distance between the current point $\mathbf{P}_\mathbf{I}$ and the previous point $\mathbf{P}_\mathbf{I}$ is almost zero. The circle shrinking technique [27] consumes more time to find point $\mathbf{P}^+$ each time and it is difficult to directly compute the intersection $\mathbf{P}_\mathbf{I}$ intersected of line segment $\overline{\mathbf{PP}^+}$ and the planar algebraic curve if the degree of the planar algebraic curve is more than 5.

The fourth compromise method with circle double-and-bisect algorithm is proposed by Hu et al. [28]. Starting with a very small circle with center being test point $\mathbf{P}$ and radius r_1 being arbitrarily small, draw a new circle with the same center being test point $\mathbf{P}$ and radius $r_2 = 2r_1$ (After that, the center of all circles is the test point $\mathbf{P}$). If the new circle does not intersect the planar algebraic curve, draw a new circle with radius twice of r_2. Keep repeating the above action, until the latest circle can intersect with the planar algebraic curve. The previous circle and the latest circle are tagged as the interior circle and the exterior circle, respectively. Furthermore, implement the rest of the process by adopting the bisecting technology. Continue to draw a new circle with new radius $r = (r_1 + r_2)/2$. If the current circle with radius r intersects with the planar algebraic curve, substitute r for r_2, else for r_1. Repeatedly execute the above progress until the difference between the current radius and the previous radius is almost 0. However, with this method it is not easy to judge whether the exterior circle intersects the planar algebraic curve or not [28]. Moreover, the circle double-and-bisect algorithm [28] consumes more time to seek the intersection between the exterior circle and the planar algebraic curve.

On the problem of point orthogonal projection onto curve and surface, there exists a classical method which is transforming the problem of point orthogonal projection onto curve and surface into a specific solver problem. Bartoň M. [29] proposed two blending schemes solvers for solving the problem of point orthogonal projection onto curve and surface. As a system of nonlinear equations, throwing away no-root domain can be realized by a simple linear combination of functions and then determine all control points for its Bernstein-Bézier basis having the same sign, which must be in accord with the seeking function. It can continuously obtain these types of functions to get rid of the no-root domain through the continuous subdivision process. The efficiency of the geometric-based algorithm [29] is higher than that of the existing subdivision mode based GPU. Therefore, two blending schemes in [29] can efficiently reduce the number of subdivisions by using continuous eliminating no-root domains. Result from the consequence in [29], van Sosin and Elber [30] constructed a variety of complex piecewise polynomial systems having zero or inequality constraints in zero-dimensional or one-dimensional solution spaces. Finding out all the root solutions is the advantage of these methods [29,30]; however, more computation consume-time and the need for many subdivision steps are their disadvantages.

2. Algorithm Realization for Point Orthogonal Projection onto a Spatial Algebraic Curve

2.1. Newton's Steepest Gradient Descent Method

Let us elaborate on the general idea. Let $\mathbf{P}$ be a test point. There is a spatial algebraic curve $\mathbf{C}$ which can be written as,

$$\begin{cases} f_1(\mathbf{X}) = 0, \\ f_2(\mathbf{X}) = 0. \end{cases} \tag{5}$$

where $\mathbf{X}$ is equal to (x, y, z), and $f_1(\mathbf{X}) = 0$ and $f_2(\mathbf{X}) = 0$ are two algebraic surfaces, respectively. Their intersection set constitutes a spatial algebra curve $\mathbf{C}$. The aim of this paper is to seek a spatial point $\mathbf{P}_\Gamma$ on the spatial algebraic curve $\mathbf{C}$ to satisfy the basic relationship (See Figure 1),

$$\|\mathbf{P} - \mathbf{P}_\Gamma\| = \min_{\mathbf{X}\in\Gamma} \|\mathbf{P} - \mathbf{X}\|. \tag{6}$$

On the other hand, the vector $\overrightarrow{\mathbf{PX}}$ determined by the test point $\mathbf{P}$ and point $\mathbf{X}$ on the spatial algebraic curve $\mathbf{C}$ is orthogonal to the tangent vector $\mathbf{T}$, where $\mathbf{T} = \nabla f_1(\mathbf{X}) \times \nabla f_2(\mathbf{X})$. The corresponding equation formed by the inner product of the vector $\overrightarrow{\mathbf{PX}}$ and vector $\mathbf{T}$ can be expressed as,

$$\mathbf{F}(\mathbf{X}) = \langle \mathbf{T}, \mathbf{P} - \mathbf{X}\rangle = 0. \tag{7}$$

The above two questions can be summed up as,

$$\begin{cases} f_1(\mathbf{P}_\Gamma) = 0, \\ f_2(\mathbf{P}_\Gamma) = 0, \\ \mathbf{F}(\mathbf{P}_\Gamma) = 0, \\ \|\mathbf{P} - \mathbf{P}_\Gamma\| = \min\limits_{\mathbf{X}\in\Gamma} \|\mathbf{P} - \mathbf{X}\|, \end{cases} \tag{8}$$

where $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]$ is Hamiltonian operator, symbol $\times$ is cross product and symbol $\langle , \rangle$ is inner product. We take s as the arc length parameter of the spatial algebraic curve $\mathbf{C}$ and Let $\mathbf{T} = \left[\frac{dx}{ds}, \frac{dy}{ds}, \frac{dz}{ds}\right]$ be tangent vector along the spatial algebraic curve $\mathbf{C}$. There are many ways to construct the Newton's steepest descent methods of the spatial algebraic curve with Equation (5). After many tests and comparisons, we find that the following model of Newton's steepest descent method with Equation (9) is the best choice for the Newton's steepest descent methods of the spatial algebraic curve.

$$\begin{cases} \mathbf{Y}_n = \mathbf{X}_n - (f_1(\mathbf{X}_n)/\langle \nabla f_1(\mathbf{X}_n), \nabla f_1(\mathbf{X}_n)\rangle)/\nabla f_1(\mathbf{X}_n), \\ \mathbf{X}_{n+1} = \mathbf{Y}_n - (f_2(\mathbf{Y}_n)/\langle \nabla f_2(\mathbf{Y}_n), \nabla f_2(\mathbf{Y}_n)\rangle)/\nabla f_2(\mathbf{Y}_n). \end{cases} \tag{9}$$

Repeatedly run Equation (9), called Newton's steepest gradient descent method, until the iterative termination criteria are satisfied $|f_1(\mathbf{X}_{n+1})| < \varepsilon$ and $|f_2(\mathbf{X}_{n+1})| < \varepsilon$, where the initial iterative point is $\mathbf{X}_0 = \mathbf{P} - (0.1, 0.1, 0.1)$ and the iterative point $\mathbf{X}_{n+1}$ fallen on the spatial algebraic curve $\mathbf{C}$ is called $\mathbf{P}_\mathrm{I}$. The Newton's steepest gradient descent method (Algorithm 1) can be specifically described as (See Figure 2).

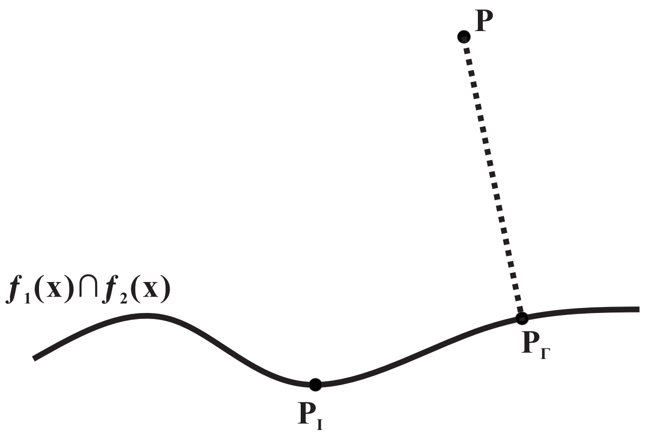

Figure 1. Test point $\mathbf{P}$ orthogonal projection onto a spatial algebraic curve $\mathbf{C}$.

Algorithm 1: The Newton's steepest gradient descent method

```
Input: The test point P and the spatial algebraic curve C.
Output: The iterative point $\mathbf{P}_{\mathrm{I}}$ fallen on the spatial algebraic curve C.
Description:
Step 1:
    $\mathbf{X}_{n+1} = \mathbf{P} - (0.1, 0.1, 0.1)$;
    Do {
        $\mathbf{X}_n = \mathbf{X}_{n+1}$ ;
        Update $\mathbf{X}_{n+1}$ according to the iterative Equation (9);
    }while ($|f_1(\mathbf{X}_{n+1})| > \varepsilon \&\& |f_2(\mathbf{X}_{n+1})| > \varepsilon \&\& \|\mathbf{X}_{n+1} - \mathbf{X}_n\| > \varepsilon$);
Step 2:
    $\mathbf{P}_{\mathrm{I}} = \mathbf{X}_{n+1}$ ;
    Return $\mathbf{P}_{\mathrm{I}}$;
```

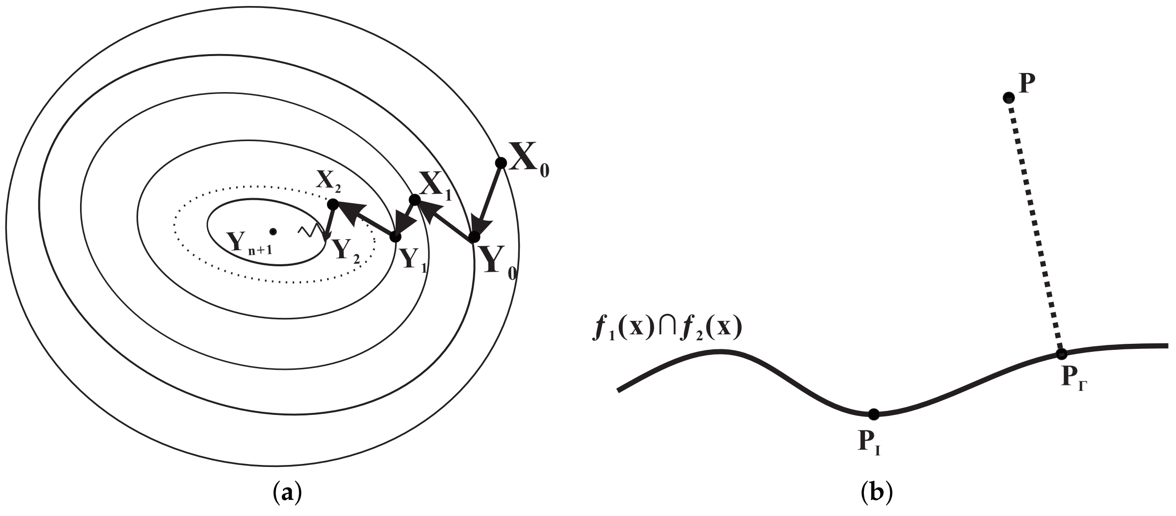

Figure 2. The entire graphic demonstration of Algorithm 1. (**a**) The whole iterative process of the Newton's steepest gradient descent method; (**b**) The last step of the iterative point $\mathbf{P}_{\mathrm{I}}$ fallen on the spatial algebraic curve **C** via the Newton's steepest gradient descent method.

Remark 1. *Starting from the initial iterative point $\boldsymbol{X}_0$, the iterative point of the first equation of the Newton's steepest gradient descent method with Equation (9) causes the iterative point to be closer to the first algebraic surface $f_1(\boldsymbol{X})$. Then through the iterating of the second equation with Equation (9), the iterative point of the second equation with Equation (9) causes the iterative point to be closer to the second algebraic surface $f_2(\boldsymbol{X})$. In this way, Equation (9) is iterated repeatedly such that the distance between the finally iterative point and every algebraic surface is almost zero. Namely, through alternating iteration of two sub-equations with Equation (9), the distance between the finally iterative point and the spatial algebraic curve determined by the intersection set of two algebraic surfaces is almost zero. The Newton's steepest gradient descent method (9) has two advantages. The first advantage of the Newton's steepest gradient descent method (9) is to promote the iterative point to fall on the spatial algebraic curve **C** as much as possible. The second advantage of the Newton's steepest gradient descent method (9) is that the iteration point fallen on the spatial algebraic curve is closer to the orthogonal projection point $\boldsymbol{P}_{\Gamma}$, it is very convenient for implementation of the subsequent sub-algorithms.*

2.2. Computing Foot-Point Q

Although the iterative point $\mathbf{P}_{\mathrm{I}}$ falls on the spatial algebraic curve, there is still a certain small distance between the iterative point $\mathbf{P}_{\mathrm{I}}$ and the orthogonal projection point $\mathbf{P}_{\Gamma}$. There is still a small certain gap between this and our ideal and goal. Our idea is to find a way to gradually reduce the distance between the iteration point $\mathbf{P}_{\mathrm{I}}$ and the orthogonal projection point $\mathbf{P}_{\Gamma}$ to zero. In order to promote the iteration point $\mathbf{P}_\mathrm{I}$ and the orthogonal projection point $\mathbf{P}_\Gamma$ closer, we use the tangent line perpendicular foot method to promote the iteration point $\mathbf{P}_\mathrm{I}$ and the orthogonal projection point $\mathbf{P}_\Gamma$ being closer. The basic idea of this method can be realized as the following form. Draw a tangent line $\mathbf{L}$ of the spatial algebraic curve $\mathbf{C}$ at the iterative point $\mathbf{P}_\mathrm{I}$ fallen the the spatial algebraic curve $\mathbf{C}$. This tangent line $\mathbf{L}$ can be written as,

$$\mathbf{X} = t \times \mathbf{T} + \mathbf{P}_\mathrm{I}, \tag{10}$$

where t is a tangent line parameter to be determined, and $\mathbf{T}$ is the tangent vector of the spatial algebraic curve $\mathbf{C}$ at the iterative point $\mathbf{P}_\mathrm{I}$. Through the test point $\mathbf{P}$, let us construct a plane such that the tangent line $\mathbf{L}$ is perpendicular to the plane. From primary geometric intuitiveness, it is not difficult to find that the intersection $\mathbf{Q}_0$ named the foot point intersected with the tangent line $\mathbf{L}$ and the plane can be expressed as

$$\mathbf{Q}_0 = t_0 \times \mathbf{T} \ + \mathbf{P}_\mathbf{I}, \tag{11}$$

where $t_0 = \langle \mathbf{P} - \mathbf{P}_\mathrm{I}, \mathbf{T} \rangle \ / \ \langle \mathbf{T}, \mathbf{T} \rangle$. In this way, the position of the foot point $\mathbf{Q}_0$ is between the iteration point $\mathbf{P}_\mathrm{I}$ and the orthogonal projection $\mathbf{P}_\Gamma$. At this time, taking foot point $\mathbf{Q}_0$ as the initial iterative point of Algorithm 1, we obtain the new iterative point $\mathbf{P}_\mathrm{I}$ fallen on the spatial algebraic curve $\mathbf{C}$ which the new iterative point $\mathbf{P}_\mathrm{I}$ is called the current iterative point $\mathbf{P}_\mathrm{I}$. According to our observations and the basic geometry characteristic, we can also choose the foot point closer to the spatial algebraic curve. Let us choose a point $\mathbf{Q}$ on the line segment $\overline{\mathbf{Q}_0\mathbf{P}}$ such that $\left\|\overrightarrow{\mathbf{Q}_0\mathbf{Q}}\right\| = \frac{1}{2} \times \left\|\overrightarrow{\mathbf{Q}_0\mathbf{P}_\mathrm{I}}\right\|$. Inspired by this, it is not difficult for us to establish the equation about foot-point $\mathbf{Q}$.

$$\frac{\overrightarrow{\mathbf{Q}_0\mathbf{Q}}}{\overrightarrow{\mathbf{Q}_0\mathbf{P}}} = d, \tag{12}$$

where $d = \frac{d_1}{d_2}, d_1 = \frac{\left\|\overrightarrow{\mathbf{P}_\mathrm{I}\mathbf{Q}_0}\right\|}{2}, d_2 = \left\|\overrightarrow{\mathbf{Q}_0\mathbf{P}}\right\|$. By solving Equation (12), foot-point $\mathbf{Q}$ can be specifically expressed as

$$\mathbf{Q} = d \times (\mathbf{P} - \mathbf{Q}_0) + \mathbf{Q}_0 \tag{13}$$

The computation of the foot-point $\mathbf{Q}$ can be achieved via Algorithm 2 (See Figure 3).

```
Algorithm 2: Computing foot-point Q
  Input: The test point P, the spatial algebraic curve C and the iterative point P_I on the spatial
    algebraic curve C.
  Output: The foot-point Q.
  Description:
  Step 1:
      Compute the foot-point Q by the formula (13).
      Return Q;
```

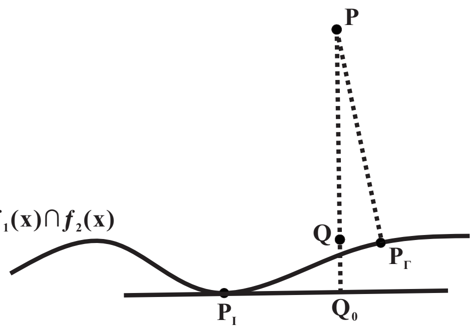

Figure 3. Graphic demonstration for Algorithm 2.

2.3. Accelerating and Correcting Method

By Algorithm 1, the initial iteration point $\mathbf{X}_0$ can fall on the spatial algebraic curve $\mathbf{C}$ and a final iterative point $\mathbf{P}_\mathrm{I}$ is generated. Then the foot-point Q is obtained by Algorithm 2 derived from the final iterative point $\mathbf{P}_\mathrm{I}$. So the foot-point $\mathbf{Q}$ is taken as the initial iterative point of Algorithm 1, and a new current final iterative point $\mathbf{P}_\mathrm{I}$ fallen on the spatial algebraic curve $\mathbf{C}$ is generated. Run Algorithm 1 and Algorithm 2 alternately and repeatedly, the current iterative point $\mathbf{P}_\mathrm{I}$ is getting closer to the orthogonal projection point $\mathbf{P}_\Gamma$. However, the rate of the iterative point $\mathbf{P}_\mathrm{I}$ approaching the orthogonal projection point $\mathbf{P}_\Gamma$ becomes slower and slower. And orthogonalization is hard to achieve. In other words, the first three sub-formulas of Formula (8) are difficult to satisfy which this process is not up to the ideal.

In order to overcome these two disadvantages, we add accelerating and correcting method after Algorithm 2 such that the first three sub-formulas of formula (8) can be satisfied as soon as possible. Since $\mathbf{T} = \nabla f_1 \times \nabla f_2$ is the tangent vector of the spatial algebraic curve $\mathbf{C}$ at point $\mathbf{X}$, so the relationship between the tangent vector $\mathbf{T} = \nabla f_1 \times \nabla f_2$ and the spatial algebraic curve $\mathbf{C}$ can be written as,

$$\begin{cases} \langle \mathbf{T}, \nabla f_1 \rangle = 0, \\ \langle \mathbf{T}, \nabla f_2 \rangle = 0. \end{cases} \tag{14}$$

Furthermore, let $\mathbf{c} = \left[\frac{d^2x}{dt^2}, \frac{d^2y}{dt^2}, \frac{d^2z}{dt^2}\right]$ be the curvature vector of the spatial algebraic curve $\mathbf{C}$ at the point $\mathbf{X}$. Taking the first-order derivative of Equation (14) with respect to arc length parameter s, get the corresponding equation about curvature vector,

$$\begin{cases} \mathbf{T}F_2\mathbf{T}^\mathrm{T} + \nabla f_1 \mathbf{c}^\mathrm{T} = 0, \\ \mathbf{T}G_2\mathbf{T}^\mathrm{T} + \nabla f_2 \mathbf{c}^\mathrm{T} = 0, \\ \mathbf{T}\mathbf{c}^\mathrm{T} = 0, \end{cases} \tag{15}$$

where $F_2 = \begin{pmatrix} f_{1xx} & f_{1xy} & f_{1xz} \\ f_{1yx} & f_{1yy} & f_{1yz} \\ f_{1zx} & f_{1zy} & f_{1zz} \end{pmatrix}$, $G_2 = \begin{pmatrix} f_{2xx} & f_{2xy} & f_{2xz} \\ f_{2yx} & f_{2yy} & f_{2yz} \\ f_{2zx} & f_{2zy} & f_{2zz} \end{pmatrix}$, $f_{1xx} = \frac{\partial f_1^2(X)}{\partial x^2}$, $f_{1xy} = \frac{\partial f_1^2(X)}{\partial x \partial y}$, $f_{2xx} = \frac{\partial f_2^2(X)}{\partial x^2}$, $f_{2xy} = \frac{\partial f_2^2(X)}{\partial x \partial y}$, etc. Solving Equation (15), the specific curvature vector $\mathbf{c}$ can be expressed as

$$\mathbf{c} = \left[-\mathbf{T}F_2\mathbf{T}^\mathrm{T}, -\mathbf{T}G_2\mathbf{T}^\mathrm{T}, 0\right] W_2^\mathrm{T}, \tag{16}$$

where $W_2 = \begin{pmatrix} f_{1x} & f_{1y} & f_{1z} \\ f_{2x} & f_{2y} & f_{2z} \\ x_s & y_s & z_s \end{pmatrix}$, $f_{1x} = \frac{\partial f_1(X)}{\partial x}$, $f_{2x} = \frac{\partial f_2(X)}{\partial x}$, $x_s = \frac{dx}{ds}$, etc. For the case of the spatial algebraic curve, the iterative point $\mathbf{P}_\mathrm{I}$ may deviate from the spatial algebraic curve $\mathbf{C}$. That is, the iterative point $\mathbf{P}_\mathrm{I}$ is not on two algebraic surfaces, $f_1(\mathbf{P}_\mathrm{I}) = err1 \neq 0$, $f_2(\mathbf{P}_\mathrm{I}) = err2 \neq 0$. The basic idea of error correcting can be described as follows. When the iterative point $\mathbf{P}_\mathrm{I}$ deviates from two algebraic surfaces, use correction vector $\delta\mathbf{V} = [\delta x, \delta y, \delta z]$ to correct iteration point $\mathbf{P}_\mathrm{I}$ such that $|f_1(\mathbf{P}_\mathrm{I}| < Le$ and $|f_2(\mathbf{P}_\mathrm{I})| < Le$ (Le is a particularly small value of the allowable error range). The corrected iteration point $\mathbf{P}_\mathrm{I}$ is guaranteed to be on the spatial algebraic curve and the deviations between the corrected iterative point $\mathbf{P}_\mathrm{I}$ and two algebraic surfaces are almost zero. $\Delta\mathbf{V}$ is on the plane formed by the tangent vector $\mathbf{T}$ and curvature vector $\mathbf{c}$ derived from the iteration point $\mathbf{P}_\mathrm{I}$ before correction and is perpendicular to the vector $\mathbf{T} \times \mathbf{c}$. Let $\delta\mathbf{V}$ be on the plane spanned by the vector

$\Delta\mathbf{V}$ and $\mathbf{T} \times \mathbf{c}$, and $\delta\mathbf{V}$ points to two algebraic surfaces. From the above description, we can get the equation about the correction vector $\delta\mathbf{V}$,

$$\begin{cases} \langle \nabla f_1, \delta\mathbf{V} \rangle = 0, \\ \langle \nabla f_2, \delta\mathbf{V} \rangle = 0, \\ \langle [(\mathbf{T} \times \mathbf{c}) \times \Delta\mathbf{V}], \delta\mathbf{V} \rangle = 0. \end{cases} \tag{17}$$

From Equation (17), it is not difficult to get the expression of the correction vector $\delta\mathbf{V}$,

$$\delta\mathbf{V} = [-err1, -err2, 0][(\nabla f_1)^{\mathrm{T}}, (\nabla f_2)^{\mathrm{T}}, [(\mathbf{T} \times \mathbf{c}) \times \Delta\mathbf{V}]^{\mathrm{T}}]^{-1}, \tag{18}$$

where $err1 = f_1(\mathbf{P}_{\mathrm{I}})$, $err2 = f_2(\mathbf{P}_{\mathrm{I}})$, at this moment, the iteration point $\mathbf{P}_{\mathrm{I}}$ is the iteration point before correction, $\Delta\mathbf{V}$ is the difference vector of the corrected iteration point $\mathbf{P}_{\mathrm{I}}$ subtracting the iteration point $\mathbf{P}_{\mathrm{I}}$ before correction. After a lot of testing, we have obtained a preliminary result. When the correction vector $\delta\mathbf{V}$ is added to the iteration point $\mathbf{P}_{\mathrm{I}}$ after iterating, it can promote the robustness and high efficiency of convergence from iteration point $\mathbf{P}_{\mathrm{I}}$ to orthogonal projection point $\mathbf{P}_{\Gamma}$. However, the rate which the iteration point $\mathbf{P}_{\mathrm{I}}$ converges to the orthogonal projection point $\mathbf{P}_{\Gamma}$ is still very slow, at the same time, the rate of the orthogonalization of the tangent vector of the spatial algebraic curve at the iteration point $\mathbf{P}_{\mathrm{I}}$ and the vector connected to the test point $\mathbf{P}$ and the iteration point $\mathbf{P}_{\mathrm{I}}$ is also very slow. There is still a big gap between this situation and our expected goal. In order to accelerate the convergence of the iteration point to the orthogonal projection point $\mathbf{P}_{\Gamma}$ and speed up the orthogonalization, we add three iterative formulas related to Newton's steepest gradient descent method in front of the correction vector $\delta\mathbf{V}$. Therefore, the corresponding iterative formula for accelerating and orthogonalizing is as follows,

$$\begin{cases} \mathbf{Y}_n = \mathbf{X}_n - (f_1(\mathbf{X}_n) / \langle \nabla f_1(\mathbf{X}_n), \nabla f_1(\mathbf{X}_n) \rangle) / \nabla f_1(\mathbf{X}_n), \\ \mathbf{Z}_n = \mathbf{Y}_n - (f_2(\mathbf{Y}_n) / \langle \nabla f_2(\mathbf{Y}_n), \nabla f_2(\mathbf{Y}_n) \rangle) / \nabla f_2(\mathbf{Y}_n), \\ \mathbf{U}_n = \mathbf{Z}_n - (F(\mathbf{Z}_n) / \langle \nabla F(\mathbf{Z}_n), \nabla F(\mathbf{Z}_n) \rangle) / \nabla F(\mathbf{Z}_n), \\ \mathbf{X}_{n+1} = \mathbf{U}_n + \delta\mathbf{V}, \end{cases} \tag{19}$$

where $err1 = f_1(\mathbf{Z}_n)$, $err2 = f_2(\mathbf{Z}_n)$, $\nabla f_1, \nabla f_2, \mathbf{T}, \mathbf{c}$ assigned by $\mathbf{Z}_n$, $\Delta\mathbf{V} = \mathbf{U}_n - \mathbf{Z}_n$.

Remark 2. *We present a complete geometric interpretation for Equation (19). The first two formulas in Equation (19) that are not related to the test point* $\boldsymbol{P}$ *are Newton's steepest gradient descent method. The ultimate goal is to make the initial iteration point fall on the spatial algebraic curve* $\boldsymbol{C}$*. Please refer to Remark 1 for the specific explanation of geometric meaning. The third formula of Equation (19) is the Newton's steepest gradient descent method related to the test point* $\boldsymbol{P}$*. This iterative formula is the iterative formula of Newton's steepest gradient descent method formed by the direct deformation of the formula (7), which the formula (7) is the inner product of the vector* $\overrightarrow{\boldsymbol{PX}}$ *formed by the test point* $\boldsymbol{P}$ *and the point* $\boldsymbol{X}$ *on the spatial algebra curve* $\boldsymbol{C}$ *with the tangent vector* $\boldsymbol{T}$ *at that point* $\boldsymbol{X}$ *is zero. Therefore, the third iteration formula of Equation (19) is mainly used to accelerate the orthogonalization of the vector* $\overrightarrow{\boldsymbol{PX}}$ *and the tangent vector* $\boldsymbol{T}$*. The last formula of Equation (19) is to use the geometric correction vector* $\delta\boldsymbol{V}$ *to promote the iteration point fall on the spatial algebraic curve* $\boldsymbol{C}$ *as much as possible once again. Therefore, Equation (19) plays a dual roles of accelerating the iteration point to fall on the spatial algebraic curve* $\boldsymbol{C}$ *and orthogonalization of the vector* $\boldsymbol{T}$ *and the vector* $\overrightarrow{\boldsymbol{PP}_{\Gamma}}$*. Then accelerating and correcting the method with Equation (19) makes the first three formulas of the formula (8) achieves very satisfactory result. This technique not only accelerates the convergence rate of the whole iteration process but also improves the iteration robustness. Furthermore, the accuracy of the whole iteration process can be improved by the accelerating and correcting method with Equation (19).*

Based on the above illustration, we obtain the following Algorithm 3 (See Figure 4).

Algorithm 3: Accelerating and geometric correcting method for orthogonal projection onto a spatial algebraic curve

```
Input: The iterative point $\mathbf{P}_{\mathrm{I}}$ fallen on the spatial algebraic curve $\mathbf{C}$ and the spatial algebraic curve $\mathbf{C}$.
Output: The final orthogonal projection point $\mathbf{P}_{\Gamma}$ fallen on the spatial algebraic curve $\mathbf{C}$.
Description:
Step 1:
    $\mathbf{x}_{n+1} = \mathbf{P}_{\mathrm{I}}$;
    Do {
        $\mathbf{X}_n = \mathbf{X}_{n+1}$;
        Compute $\mathbf{X}_{n+1}$ according to Equation (19);
    }while ($\|\mathbf{X}_{n+1} - \mathbf{X}_n\|^2 > \varepsilon \&\& |f_1(\mathbf{X}_{n+1})| > \varepsilon \&\& |f_2(\mathbf{X}_{n+1})| > \varepsilon$);
Step 2:
    $\mathbf{P}_{\Gamma} = \mathbf{X}_{n+1}$;
    Return $\mathbf{P}_{\Gamma}$;
```

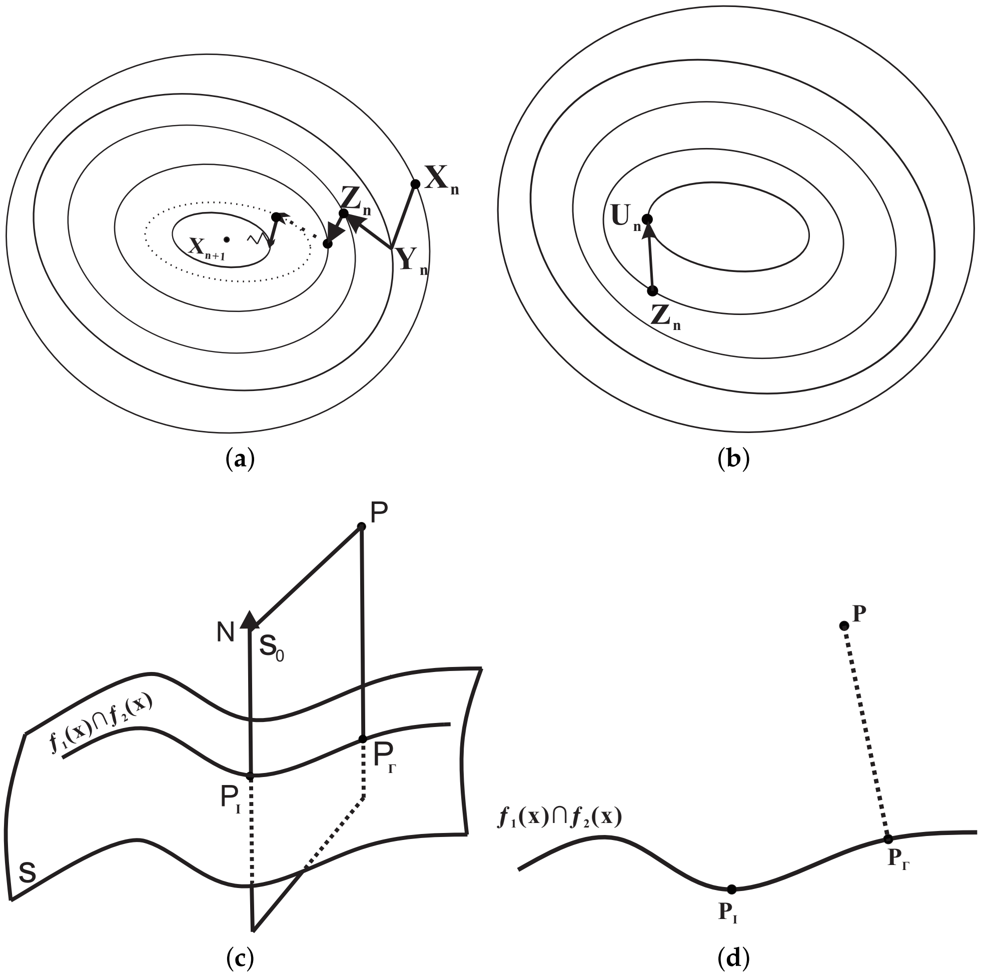

Figure 4. Graphic interpretation of the whole iteration process in Algorithm 3. (**a**) Newton's steepest gradient descent method in the first step and in the second step; (**b**) The Newton's iteration method related to the test point in the third step; (**c**) Correcting the previous iteration point to improve the robustness and high efficiency of iteration in the last step; (**d**) The iterative point $\mathbf{P}_{\mathrm{I}}$ finally iterates to the orthogonal projection point $\mathbf{P}_{\Gamma}$.

Based on the above description, we give the complete algorithm for point orthogonal projection onto a spatial algebraic curve (Algorithm 4) (See Figure 5).

Algorithm 4: A complete algorithm for point orthogonal projection onto a spatial algebraic curve

```
Input: Test point P and the spatial algebraic curve C.
Output: The final orthogonal projection point $P_\Gamma$ of the test point P.
Description:
Step 1: Starting from the neighbor point of the test point P, calculate the iterative point $P_I$
  fallen on the spatial algebraic C via Algorithm 1.
for(i = 0;i < 5;i + +) {
    Step 2: Compute the foot-point Q via Algorithm 2.
    Step 3: Project foot-point Q onto the spatial algebraic curve C via Algorithm 1, then get
  new point $P_I$ fallen on the spatial algebraic curve C.
    }
Step 4: The iterative point $P_I$ fallen on the spatial algebraic curve C being as the initial iterative
  point, calculate the finally orthogonal projection point $P_\Gamma$ fallen on the spatial algebraic C via
  Algorithm 3.
Return $P_\Gamma$;
```

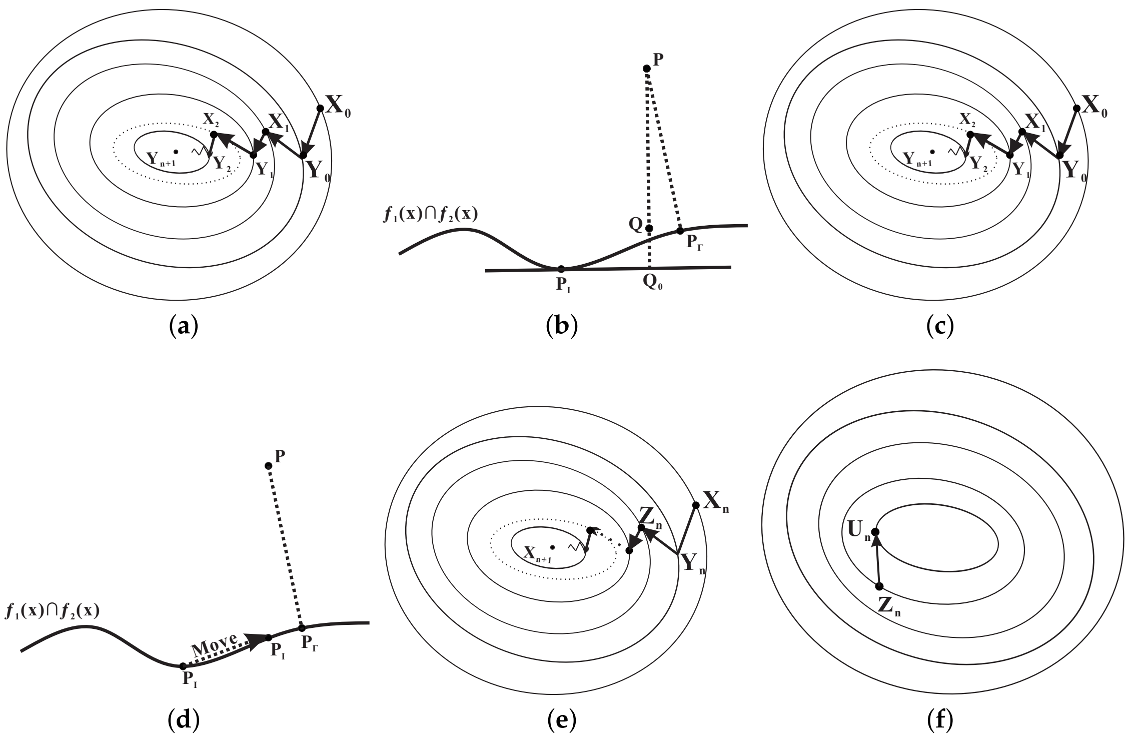

Figure 5. *Cont.*

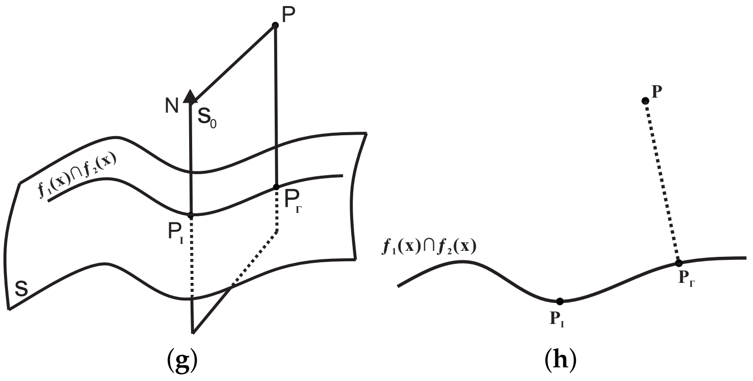

Figure 5. Graphic interpretation of the whole iteration process in Algorithm 3. (**a**) Newton's steepest gradient descent method in the first step; (**b**) Computing the foot-point **Q** in the second step; (**c**) Newton's steepest gradient descent method in the third step; (**d**) Moving the iteration point $\mathbf{P}_I$ to the position close to the orthogonal projection point $\mathbf{P}_\Gamma$ through joint implementation of the second and third steps 5 times; (**e**) The Newton's steepest gradient descent method of the first two formulas in formula (19) of Algorithm 3; (**f**) The Newton's steepest gradient descent method related to the test point of the third formula in formula (19) of Algorithm 3; (**g**) Accelerating and correcting method of the fourth formula in formula (19) of Algorithm 3; (**h**) The iterative point $\mathbf{P}_I$ finally iterates to the orthogonal projection point $\mathbf{P}_\Gamma$.

Remark 3. *In this remark, we present a geometric interpretation for Algorithm 4. The purpose of the first step is to promote the initial iterative point completely iterate to the spatial algebraic curve. In Step 2, the foot-point* **Q** *is derived from the iterative point* $\boldsymbol{P}_I$ *fallen on the spatial algebraic curve of the first step. Now, the position of foot-point* **Q** *is between the iteration point* $\boldsymbol{P}_I$ *and the orthogonal projection point* **P**. *In other words, the foot-point* **Q** *is closer to the orthogonal projection point than the test point* **P**. *Furthermore, by using Algorithm 1 of the third step, the foot-point* **Q** *can be completely iterated to the spatial algebra curve, and the point at which this iteration point falls on the spatial algebraic curve is called the current iteration point and the last iteration point to fall on the spatial algebra curve is named the previous iteration point. It is obvious that the current iteration point is closer to the orthogonal projection point than the previous iterative point. After joint running Step 2 and Step 3 five times in this way, the distance between the latest current iterative point and the orthogonal projection point is very small. If we keep running in this way, the rate of the iteration point getting closer to the orthogonal projection point becomes slower and slower. Moreover, orthogonalization of the tangent vector* $\boldsymbol{T}$ *and the vector* $\overrightarrow{\boldsymbol{P}\boldsymbol{P}_\Gamma}$ *is also difficult to realize. However, Algorithm 3 in the fourth step can make up for these two deficiencies. At this time, running Algorithm 3 plays double important roles for accelerating convergence and orthogonalization. Thus, Algorithm 4 is a robust and efficient algorithm which it achieves a very satisfactory result for point orthogonal projection onto the spatial algebraic curves. In addition, the five cycles of the intermediate* ***for*** *loop body are an empirical value. When the distance between the test point and the spatial algebra curve is not very far, the intermediate* ***for*** *loop body only needs 5 cycles. However, when the distance between the test point and the spatial algebraic curve is very large, the number of cycles of the* ***for*** *loop body needs to be increased. The purpose is to promote the iterative point after the loop of* ***for*** *loop body closer to the orthogonal projection point. Therefore, Algorithm 3 of the fourth step can conver the orthogonal projection point* $\boldsymbol{P}_\Gamma$ *successfully and completely.*

We have done a lot of tests for Algorithm 4. Whether the test point **P** is close to or far from the spatial algebra curve **C**, the test point can be orthogonally projected to the spatial algebraic curve in a very robust and efficient way. We have another surprise and wonderful discovery. When the test point is not far from the spatial algebraic curve, Algorithm 4 can be expressed in simplified version form, where the **for** loop body in the middle can be omitted and the fourth formula with geometric correction method of Equation (19) can also be deleted. So Equation (19) naturally becomes Equation (20).

$$\begin{cases} \mathbf{Y}_n = \mathbf{X}_n - (f_1(\mathbf{X}_n)/\langle \nabla f_1(\mathbf{X}_n), \nabla f_1(\mathbf{X}_n)\rangle)/\nabla f_1(\mathbf{X}_n), \\ \mathbf{Z}_n = \mathbf{Y}_n - (f_2(\mathbf{Y}_n)/\langle \nabla f_2(\mathbf{Y}_n), \nabla f_2(\mathbf{Y}_n)\rangle)/\nabla f_2(\mathbf{Y}_n), \\ \mathbf{X}_{n+1} = \mathbf{Z}_n - (F(\mathbf{Z}_n)/\langle \nabla F(\mathbf{Z}_n), \nabla F(\mathbf{Z}_n)\rangle)/\nabla F(\mathbf{Z}_n), \end{cases} \tag{20}$$

The simplification version of Algorithm 4 can be described as the following Algorithm 5, where Algorithm 5 maintains the same robustness as Algorithm 4, and the time running efficiency of Algorithm 5 is much better than that of Algorithm 4.

Algorithm 5: A simplified version algorithm for point orthogonal projection onto a spatial algebraic curve

```
Input: Test point $\mathbf{P}$ and the spatial algebraic curve $\mathbf{C}$.
Output: Orthogonal projection point $\mathbf{P}_\Gamma$ of the test point $\mathbf{P}$ which orthogonally projects the test
  point $\mathbf{P}$ onto the spatial algebraic curve $\mathbf{C}$.
Description:
Step1: Calculate the iterative point $\mathbf{P}_\mathrm{I}$ fallen on the the spatial algebraic curve $\mathbf{C}$ via
  Algorithm 1.
Step 2:
      $\mathbf{X}_{n+1} = \mathbf{P}_\mathrm{I}$;
      Do {
            $\mathbf{X}_n = \mathbf{X}_{n+1}$;
            Compute $\mathbf{X}_{n+1}$ according to Equation (20);
      }while ($\|\mathbf{X}_{n+1} - \mathbf{X}_n\|^2 > \varepsilon \&\& |f_1(\mathbf{X}_{n+1})| > \varepsilon \&\& |f_2(\mathbf{X}_{n+1})| > \varepsilon$);
Step 3: $\mathbf{P}_\Gamma$=$\mathbf{X}_{n+1}$;
      Return $\mathbf{P}_\Gamma$;
```

Remark 4. *In the actual algorithms implementation process, we take two optimized techniques. Firstly, if the test point $\boldsymbol{P}$ is very far away from the spatial algebraic curve $\boldsymbol{C}$, the initial iterative $\boldsymbol{X}_0$ of Algorithm 1 is a very small proportion number of the test point $\boldsymbol{P}$ which ensures that the initial iterative point $\boldsymbol{X}_0$ converges to the spatial algebraic curve $\boldsymbol{C}$ very robustly. However, the initial iterative point $\boldsymbol{X}_0$ of Algorithm 1 in Step 3 of Algorithm 4 remains unchanged($\boldsymbol{X}_0$=$\boldsymbol{Q}$). Secondly, due to the spatial algebraic curve being composed of the intersection of two algebraic surfaces, the singularity of any algebraic surface may lead to the singularity of spatial algebraic curve. That is to say, the singularity of a spatial algebraic curve will cause the denominators of all formulas and iterative formulas to be zero. In order to avoid the degradation of all iterative formulas with zero denominators, we add a small perturbation number ε to the denominator of each iteration formula such that every iterative formula will not degenerate. Thus, every algorithm can run and iterate normally. Furthermore, no matter how far away the test point is from the spatial algebraic curve and whether there are singular points or not in the spatial algebra curve, Algorithm 4 can run very robustly and efficiently which it meets our expectations and ideals.*

2.4. Finding out Initial Iterative Candidate Points

This subsection comes from the reviewer's previous suggestion. In the field of practical computer graphics, computer-aided geometric design and other applications, it is necessary to solve not only a single orthogonal projection point but also all orthogonal projection points as much as possible. Our preliminary idea is to outline the spatial algebraic curve which the specific expression is presented by the formula (5). We try to find out several 3D bounding boxes within the specified region of the spatial algebraic curve such that every curve segment of the spatial algebraic curve is contained in one 3D bounding box. Randomly select a point in each 3D bounding box as the initial iteration point of Algorithm 4 or Algorithm 5. In this way, we can yield as many orthogonal projection points as possible by using Algorithm 4 or Algorithm 5. Then, through computing the distance between the test point **P** and each orthogonal projection point and by comparing with each distance one by one, the orthogonal projection point corresponding to the shortest distance is what we are looking for. However, it is very difficult to express the spatial algebraic curve directly. On the other hand, the geometric essence of point orthogonal projection onto the spatial algebraic curve is to find a point **X** on the spatial algebraic curve such that the vector $\overrightarrow{\mathbf{PX}}$ and the tangent vector $\mathbf{T}(\mathbf{T} = \nabla f_1(\mathbf{X}) \times \nabla f_2(\mathbf{X}))$ on the spatial algebraic curve at point **X** are perpendicular to each other. Namely, the vector $\overrightarrow{\mathbf{PX}}$ is orthogonal to the tangent vector **T**. The corresponding equation formed by the inner product of the vector $\overrightarrow{\mathbf{PX}}$ and the vector **T** is expressed as the form of Equation (7). The geometric meaning of Equation (7) is algebraic surface. So, outlining the spatial algebraic curve with Equation (5) is transforming into outlining the algebraic surface with Equation (7). We are gong to find several 3D bounding boxes within the specified region of the algebraic surface with Equation (7) such that every surface patch of the algebraic surface is contained in one 3D bounding box. Randomly select a point in each 3D bounding box as the initial iteration point of Algorithm 4 or Algorithm 5. In this way, we can yield as many orthogonal projection points as possible by using Algorithm 4 or Algorithm 5. Then through computing the distance between the test point **P** and each orthogonal projection point and by comparing with each distance one by one, the orthogonal projection point (the closest point) corresponding to the shortest distance is what we are looking for.Therefore, the orthogonal projection point (the closest point) satisfied the shortest distance completely conforms to every formula of Equation (8).

Let us suppose that the region Ω of the spatial algebraic curve with Equation (5) is $\Omega = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$. We may decide that the region of the algebraic surface is the same as that of the spatial algebraic curve. So the region of the algebraic surface is also Ω. We adopt adaptive affine arithmetic method [31–33] to identify a series of 3D bounding boxes such that every algebraic surface patch is contained in every 3D bounding box. We orthogonally project the algebraic surface $\mathbf{F}(\mathbf{X}) = 0$ onto $y - z$ plane, the $x - z$ plane and the $x - y$ plane at the point (x_0, y_0, z_0), respectively, and we obtain the planar algebraic curves $\mathbf{F}(x_0, y, z) = 0$, $\mathbf{F}(x, y_0, z) = 0$ and $\mathbf{F}(x, y, z0) = 0$. respectively. For the sake of convenience, the planar algebraic curves $\mathbf{F}(x_0, y, z) = 0$, $\mathbf{F}(x, y_0, z) = 0$ and $\mathbf{F}(x, y, z0) = 0$ can be written as $\mathbf{F}_1(y, z) = 0$, $\mathbf{F}_2(x, z) = 0$, $\mathbf{F}_3(x, y) = 0$, respectively. We set up a crucial judgment function for the planar algebraic curve $\mathbf{F}_1(y, z) = 0$,

$$\Psi_{yz} = \frac{2\left|\mathbf{F}_{1yz}\right|}{\sqrt{(\frac{\mathbf{F}_{1y}}{y_2})^2 + (\frac{\mathbf{F}_{1z}}{z_3})^2}}, \tag{21}$$

Similarly, we also set up two other crucial judgment functions for the planar algebraic curve $\mathbf{F}_2(x, z) = 0$ and the planar algebraic curve $\mathbf{F}_3(x, y) = 0$,

$$\Psi_{xz} = \frac{2\left|\mathbf{F}_{2xz}\right|}{\sqrt{(\frac{\mathbf{F}_{2x}}{x_1})^2 + (\frac{\mathbf{F}_{2z}}{z_3})^2}}, \tag{22}$$

and

$$\Psi_{xy} = \frac{2\left|\mathbf{F}_{3xy}\right|}{\sqrt{(\frac{\mathbf{F}_{3x}}{x_1})^2 + (\frac{\mathbf{F}_{3y}}{y_2})^2}}, \tag{23}$$

where $\mathbf{F}_{1y} = \frac{\partial \mathbf{F}_1(y,z)}{\partial y}$, $\mathbf{F}_{1z} = \frac{\partial \mathbf{F}_1(y,z)}{\partial z}$, $\mathbf{F}_{2x} = \frac{\partial \mathbf{F}_2(x,z)}{\partial x}$, $\mathbf{F}_{2z} = \frac{\partial \mathbf{F}_2(x,z)}{\partial z}$, $\mathbf{F}_{3x} = \frac{\partial \mathbf{F}_3(x,y)}{\partial x}$, $\mathbf{F}_{3y} = \frac{\partial \mathbf{F}_3(x,y)}{\partial y}$, $\mathbf{F}_{1yz} = \frac{\partial^2 \mathbf{F}_1(y,z)}{\partial y \partial z}$, $\mathbf{F}_{2xz} = \frac{\partial^2 \mathbf{F}_2(x,z)}{\partial x \partial z}$, $\mathbf{F}_{3xy} = \frac{\partial^2 \mathbf{F}_3(x,y)}{\partial x \partial y}$, $x_0 = \frac{a_1+a_2}{2}$, $x_1 = \frac{a_2-a_1}{2}$, $y_0 = \frac{b_1+b_2}{2}$, $y_2 = \frac{b_2-b_1}{2}$, $z_0 = \frac{c_1+c_2}{2}$, $z_3 = \frac{c_2-c_1}{2}$. The each substitution value of variable **X** for nine partial derivative functions $\mathbf{F}_{1y}$, $\mathbf{F}_{1z}$, $\mathbf{F}_{2x}$, $\mathbf{F}_{2z}$, $\mathbf{F}_{3x}$, $\mathbf{F}_{3y}$, $\mathbf{F}_{1yz}$, $\mathbf{F}_{2xz}$, $\mathbf{F}_{3xy}$ is (x_0, y_0, z_0). Afterwards, we will call this point (x_0, y_0, z_0)

as the center point of the 3D bounding box. Combining with Equation (21), Equation (22) and Equation (23), we get the crucial judgment function Ψ with Equation (24),

$$\Psi = Max\{\Psi_{xy}, \Psi_{yz}, \Psi_{xz}\}. \tag{24}$$

We specify a threshold for the crucial judgment function Ψ with Equation (24). The adaptive approach method which we have adopted for solving a series of 3D bounding boxes of the algebraic surface has absorbed the idea of the papers [31–33], which is for solving a series of 2D bounding boxes of the planar algebraic curve. The detailed explanation of Equation (24) is exactly similar to the explanation in [34]. On affine arithmetic, we mainly absorbed the ideas of this paper. As for the more complex topological structure of the algebraic surface, we have to absorb the idea of the paper [35] to solve a series of 3D bounding boxes of the algebraic surface. The specific description for solving a series of 3D bounding boxes of the algebraic surface can be expressed as Algorithm 6.

Algorithm 6: Finding out a series of 3D bounding boxes of the algebraic surface $\mathbf{F}(\mathbf{X}) = 0$

```
Input: The algebraic surface $\mathbf{F}(\mathbf{X}) = 0$ and the 3D bounding box containing or intersecting
  with the algebraic surface $\mathbf{F}(\mathbf{X}) = 0$.
Output: A series of 3D bounding boxes of the algebraic surface $\mathbf{F}(\mathbf{X}) = 0$.
Description:
Step 1: Subdivide this 3D bounding box into 8 3D sub-bounding boxes by dividing 2 on each
  axis.
Step 2: Evaluate the value $\Psi$ of each 3D sub-bounding box according to Equation (24).
Step 3: if($\Psi$ <= threshold and recursion times<10){
            Run Algorithm 6 with the 3D sub-bounding box.
        }
        if($\Psi$ <= threshold and recursion times==10){
            Store the sub-box to a set.
        }
End Algorithm.
```

3. Convergence Analysis

This section proves that Algorithm 4 does not depend on the initial iteration point.

Theorem 1. *Algorithm 4 is independent of the initial iterative point.*

Proof. According to Remark 4, if the test point $\mathbf{P}$ is very far away from the spatial algebraic curve $\mathbf{C}$, the initial iterative $\mathbf{X}_0$ of Algorithm 1 is a very small proportion number of the test point P. That is to say, as long as the distance between the initial iteration point $\mathbf{X}_0$ and the spatial algebraic curve is relatively near, the initial iteration point can converge to the spatial algebraic curve by using Algorithm 1, or the initial iteration point can successfully and completely iterate and fall on the spatial algebraic curve. Based on the iterative result of Algorithm 1, the general idea of the intermediate **for** loop body is to ensure that the iteration point falls on the spatial algebraic curve under the premise such that the iterative point $\mathbf{P}_\Gamma$ can gradually move to a very close position of the orthogonal projection point. According to Remark 2, Algorithm 3 with Equation (19) plays a dual roles of accelerating the iteration point to fall on the spatial algebraic curve $\mathbf{C}$ and orthogonalization of the vector $\mathbf{T}$ and the vector $\overrightarrow{\mathbf{PP}_\Gamma}$. Then accelerating and correcting method with Equation (19) makes the first three formulas of the formula (8) achieve a very satisfactory result. This technique not only accelerates the convergence rate of the whole iteration process but also improves the iteration robustness. Furthermore, the accuracy of the whole iteration process can be improved by the accelerating and correcting the method with Equation (19). Therefore, for the initial iteration point $\mathbf{X}_0$ at any position, by using Algorithm 4, it can be finally iterated to spatial algebraic curve such that the finally iterative point fallen on the spatial algebraic curve is the orthogonal projection point. Namely, the final iteration point $\mathbf{X}_{n+1}$ that falls on the spatial algebraic curve satisfies the first three terms of the Formula (8). Specially, the test point P and the final iteration point $\mathbf{X}_{n+1}$ satisfy the relationship $\left\langle \overrightarrow{\mathbf{PX}_{n+1}}, \mathbf{T} \right\rangle = 0$ or $\left\langle \overrightarrow{\mathbf{PP}_{\Gamma}}, \mathbf{T} \right\rangle = 0$. It is possible that the orthogonal projection points obtained by Algorithm 4 may be different for different initial iteration points. Once the initial iteration point is determined, the orthogonal projection point is also uniquely determined accordingly. It can converge to the orthogonal projection point robustly and efficiently. It is demonstrated that Algorithm 4 is independent of the initial iterative point. □

In a similar way to the proof of Theorem 1, we can state the following result.

Theorem 2. *Algorithm 5 is independent of the initial iterative point.*

4. Numerical Results

In this section, we present two examples to explain that Algorithm 4 is robust and efficient. All calculation were implemented by using Maple 18 environment with $\varepsilon = 10^{-20}$. In Tables 1 and 2, the five symbols $\mathbf{P}$, $\mathbf{P}_{\Gamma}$, $|f_1(\mathbf{P}_{\Gamma})|$, $|f_2(\mathbf{P}_{\Gamma})|$, $|\mathbf{F}(\mathbf{P}_{\Gamma})|$ are the test point, the orthogonal projection point of the test point, the absolute value of the deviation degree of the orthogonal projection point on the first algebraic surface, the absolute value of the deviation degree of the orthogonal projection point on the second algebraic surface and the absolute value of the expression of Equation (7), respectively.

Example 1. *Assume that there is a spatial algebraic curve* $\begin{cases} f_1(x,y,z) = \frac{x^2}{9} + \frac{y^2}{16} + \frac{z^2}{9} - 1 \\ f_2(x,y,z) = x^2 + y^2 - 4z \end{cases}$ *(See Figure 6). In this area [−300, 300] × [−300, 300] × [−300, 300]. We randomly select a large number of test points, the probability of convergence can achieve 100% by Algorithm 4 which is verified to be very high robustness and efficiency for remote test points. In addition, in each of the eight quadrants, we randomly select two different test points. We calculate the corresponding orthogonal projection point, the absolute value of the deviation degree of the orthogonal projection point on the first algebraic surface, the absolute value of the deviation degree of the orthogonal projection point on the second algebraic surface and the absolute value of the expression of Equation (7) for each test point via Algorithm 4. The specific results are shown in Table 1.*

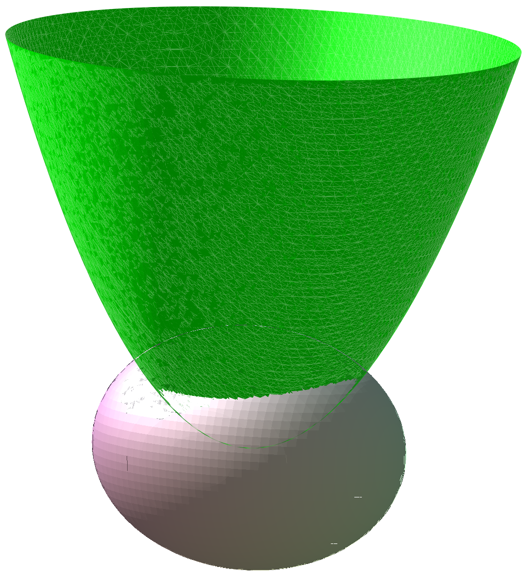

Figure 6. Graphical representation of the spatial algebraic curve for Example 1.

Table 1. Test results of Algorithm 4 for Example 1.

$\mathbf{P}$	$(217,273,129)$	$(357, 263, 289)$	$(374,236,-389)$	$(274,336,-297)$	$(254,-356,287)$
$\mathbf{P}_\Gamma$	$(1.347, 2.448,1.952)$	$(1.749, 2.095, 1.863)$	$(2.241, 1.357,1.715)$	$(-1.274,-2.498,1.966)$	$(-1.574,2.268,1.906)$
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	0.0	0.0	0.0	0.0	2.0×10^{-20}
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	2.0×10^{-19}	0.0	2.0×10^{-19}	0.0	0.0
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	2.9×10^{-17}	1.0×10^{-17}	1.0×10^{-17}	4.0×10^{-17}	2.0×10^{-17}
$\mathbf{P}$	$(354,-376,481)$	$(259,-276,-441)$	$(249,-386,-221)$	$(-369, 286, 317)$	$(-359, 686, 377)$
$\mathbf{P}_\Gamma$	$(1.338, -2.454, 1.954)$	$(-1.266, 2.503, 1.967)$	$(-1.119, 2.591, 1.992)$	$(-1.697, 2.150, 1.875)$	$(-0.946, 2.678, 2.017)$
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	4.0×10^{-20}	0.0	3.0×10^{-20}	3.0×10^{-20}	0.0
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	3.0×10^{-19}	1.0×10^{-19}	1.0×10^{-19}	1.0×10^{-19}	1.0×10^{-19}
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	0.0	2.0×10^{-17}	4.0×10^{-17}	1.0×10^{-17}	0.0
$\mathbf{P}$	$(-348, 677,-397)$	$(-338, 357,-597)$	$(-368,-374, 497)$	$(-348,-324, 457)$	$(-385,-347,-467)$
$\mathbf{P}_\Gamma$	$(-1.217, 2.534, 1.976)$	$(1.262,-2.505, 1.968)$	$(-1.373,-2.429, 1.947)$	$(-1.447,-2.373, 1.932)$	$(-2.031,-1.734, 1.783)$
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	0.0	1.0×10^{-20}	3.0×10^{-20}	0.0	0.0
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	1.0×10^{-19}	1.0×10^{-19}	1.0×10^{-19}	1.0×10^{-19}	0.0
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	1.0×10^{-17}	2.0×10^{-17}	3.0×10^{-17}	3.0×10^{-17}	8.0×10^{-17}
$\mathbf{P}$	$(-355,-397,-492)$				
$\mathbf{P}_\Gamma$	$(1.290,2.487, 1.963)$				
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	4.0×10^{-20}				
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	0.0				
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	5.0×10^{-17}				

Example 2. *Assume that there is a spatial algebraic curve* $\begin{cases} f_1(x,y,z) = \frac{x^2}{9} + \frac{y^2}{16} + \frac{z^2}{9} + 4xy - 1 \\ f_2(x,y,z) = x^2 + y^2 + x - 4z \end{cases}$ *(See Figure 7). In this area [−300, 300] × [−300, 300] × [−300, 300]. We randomly select a large number of test points, the probability of convergence can achieve 100% by Algorithm 4 which is verified to be very high robustness and efficiency for remote test points. In addition, in each of the eight quadrants, randomly select two different test points. We calculate the corresponding orthogonal projection point, the absolute value of the deviation degree of the orthogonal projection point on the first algebraic surface, the absolute value of the deviation degree of the orthogonal projection point on the second algebraic surface and the absolute value of the expression of Equation (7) for each test point via Algorithm 4. The specific results are shown in Table 2.*

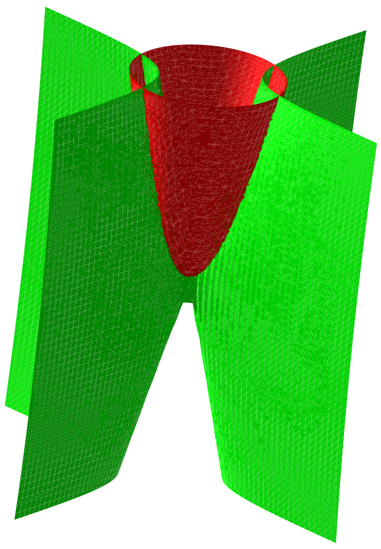
Figure 7. Graphical representation of the spatial algebraic curve for Example 2.

Table 2. Test results of Algorithm 4 for Example 2.

$\mathbf{P}$	$(217,273,129)$	$(357, 263, 289)$	$(374,236,-289)$	$(254,310,-157)$	$(254,-356,287)$
$\mathbf{P}_\Gamma$	$(0.515,0.461,0.248)$	$(0.410,0.580,0.229)$	$(0.360,0.661,0.231)$	$(0.675, 0.344,0.312)$	$(0.014,2.715, 1.847)$
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	0.0	0.0	4.8×10^{-15}	3.7×10^{-11}	1.9×10^{-11}
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	2.7×10^{-13}	7.4×10^{-12}	6.1×10^{-13}	2.5×10^{-13}	3.5×10^{-14}
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	3.5×10^{-16}	2.2×10^{-15}	4.2×10^{-16}	3.7×10^{-15}	7.3×10^{-15}
$\mathbf{P}$	$(354,-376,481)$	$(259,-276,-441)$	$(249,-386,-221)$	$(-369, 286, 317)$	$(-359, 686, 377)$
$\mathbf{P}_\Gamma$	$(0.099, 1.799,0.836)$	$(1.053,0.199,0.550)$	$(2.132,0.014, 1.664)$	$(2.082,0.029,1.607)$	$(1.911,0.053,1.394)$
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	4.6×10^{-16}	3.5×10^{-17}	2.7×10^{-17}	7.6×10^{-18}	5.7×10^{-15}
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	4.2×10^{-16}	2.9×10^{-15}	1.3×10^{-15}	4.7×10^{-16}	3.7×10^{-15}
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	3.2×10^{-16}	4.7×10^{-15}	9.2×10^{-15}	2.3×10^{-16}	4.3×10^{-17}
$\mathbf{P}$	$(-348, 477, -397)$	$(-338, 357, -597)$	$(-368, -374, 497)$	$(-348, -324, 457)$	$(-385, -347, -467)$
$\mathbf{P}_\Gamma$	$(0.043, -0.042, 0.004)$	$(-1.771,-0.09,0.343)$	$(-0.559, -0.425,-0.016)$	$(-0.543, -0.439, -0.013)$	$(-0.255,-0.867,0.139)$
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	4.7×10^{-16}	5.3×10^{-18}	5.8×10^{-17}	2.3×10^{-18}	8.2×10^{-17}
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	3.4×10^{-18}	2.6×10^{-17}	3.5×10^{-18}	3.6×10^{-16}	4.8×10^{-17}
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	3.2×10^{-16}	2.3×10^{-15}	3.4×10^{-16}	6.3×10^{-15}	7.3×10^{-16}
$\mathbf{P}$	$(-355, -397,-492)$				
$\mathbf{P}_\Gamma$	$(-0.388, -0.616,0.035)$				
$\lvert f_1(\mathbf{P}_\Gamma)\rvert$	7.5×10^{-18}				
$\lvert f_2(\mathbf{P}_\Gamma)\rvert$	5.6×10^{-17}				
$\lvert \mathbf{F}(\mathbf{P}_\Gamma)\rvert$	6.4×10^{-18}				

Remark 5. *If the test point is farther away from the spatial algebraic curve or if the spatial algebraic curve is smooth or if there are fewer singularities in the spatial algebraic curve or if the curvature of the spatial algebraic curve is locally non-high curvature, the algorithm provided by us has certain advantages. However, if the spatial algebraic curve has one of the following cases, any case will hinder and affect the normal operation of Algorithm 4. The robustness and the efficiency of Algorithm 4 are greatly reduced.*

(1) *The test point is particularly farther away from the spatial algebraic curve.*
(2) *The spatial algebraic curve is not smooth.*
(3) *There are many singularities in the spatial algebraic curve.*
(4) *The curvature of the spatial algebraic curve is locally high.*
(5) *There are many symmetries in spatial algebraic curves.*
(6) *The spatial algebraic curve are extremely or particularly oscillatory.*

For example, there is a planar curve $f(x) = \cos^4(\frac{1}{x}) + \sin^{19}(x) + \sin^8(\frac{1}{x^2}), x \in [-4, 4]$ *(See Figure 8). From the graphic display results, this curve is particularly oscillatory. If the initial iteration point is slightly further away from the curve, it is difficult to implement by using Algorithm 1 because the inherent essence of the iterative formula (9) for Algorithm 1 is exactly the same as the classical Newton's iterative method. In view of these non-convergent cases, we are going to take three approaches. The first approach is transforming the test point orthogonal projection onto the spatial algebraic curve problem into the root-finding problem [29,30,36,37]. The second approach is to explore a strategy such that the initial iteration point should be as close as possible to the spatial algebraic curve which the goal is to minimize the sensitivity of the initial iteration point. Absorb the very robust methods in the each part of the algorithm that we already have. Then optimize them to the maximum. The third approach is to explore a one or more brand new algorithms such that every algorithm could solve all or part of the above difficult situations. Of course, it is necessary to study the singularity, non-smoothness, local high curvature and oscillation of spatial algebraic curve. Then find out how they relate to each other. Finally, some ideal algorithms that can meet the requirements of robustness and high efficiency are explored based on their intrinsic relevance.*

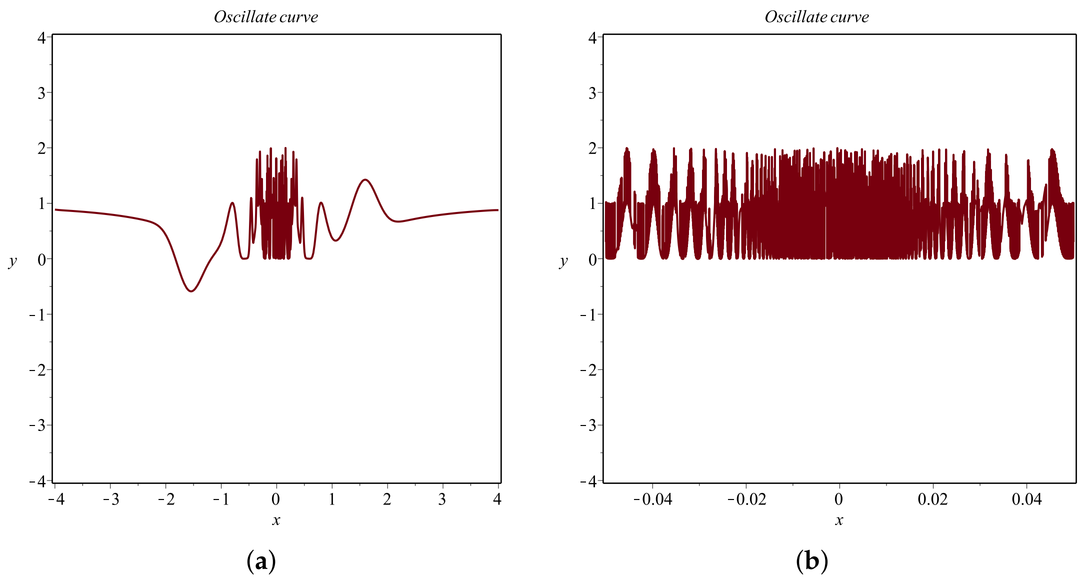

Figure 8. Oscillatory graphic display of the oscillate curve. (**a**) The entire graphic display of the oscillatory curve at the x-coordinate in the interval $[-4, 4]$; (**b**) The graphic zoom display of the oscillatory curve at the x-coordinate in the interval $[-0.05, -0.05]$.

5. Conclusions and Future Work

In this paper, we have developed an algorithm for point orthogonal projection onto a spatial algebraic curve. Algorithm 4 mainly includes three core technologies: Newton's steepest gradient descent method, computing foot-point method and geometric correction method. Algorithm 1 impels the iterative point to fall on the spatial algebraic curve. The middle **for** loop body is to let the iteration point move to the position closed to the orthogonal projection point. The algorithm 3 of the fourth step plays an important double accelerating and orthogonalization role. Numerical examples show that our algorithm is robust and efficient and it achieves our expected result. In future work, we will try to study the singularity, non-smoothness, local high curvature and oscillation of spatial algebraic curve. Then we will find out how they relate to each other. Finally, some ideal algorithms that can meet the requirements of robustness and high efficiency are explored based on their intrinsic relevance. We also try to extract and purify the idea of Algorithm 4. Moreover, the idea is applied to point orthogonal projecting onto algebraic surface.

Author Contributions: The contribution of all the authors is the same. All of the authors team up to develop the current draft. T.C. is responsible for investigating, providing resources and methodology, the original draft, writing, reviewing, validation, editing and supervision of this work. Z.W. is responsible for software, algorithm, program implementation, visualization. C.W. is responsible for writing, reviewing, editing and supervision of this work. X.L. is responsible for algorithm, program implementation, formal analysis of this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China Grant No. 61263034, the Feature Key Laboratory for Regular Institutions of Higher Education of Guizhou Province Grant No.KY[2016]003, the Key Laboratory of Advanced Manufacturing Technology, Ministry of Education, Guizhou University Grant No.KY[2018]479, National Social Science Fund Grant No. 14XMZ001, Fund of the Chinese Ministry of Education Grant No. 15JZD034 and General project of National Social Science Foundation Grant No. 18BYY195.

Acknowledgments: We take the opportunity to thank the anonymous reviewers for their thoughtful and meaningful comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pegna, J.; Wolter, F.E. Surface curve design by orthogonal projection of space curves onto free-form surfaces. *J. Mech. Des.* **1996**, *118*, 45–52. [CrossRef]
2. Hartmann, E. The normal form of a planar curve and its application to curve design. In *Mathematical Methods for Curves and Surfaces II*; Vanderbilt University Press: Nashville, TN, USA, 1997; pp. 237–244.
3. Liang, J.; Hou, L.K.; Li, X.W.; Pan, F.; Cheng, T.X.; Wang, L. Hybrid second order method for orthogonal projection onto parametric curve in n-Dimensional Euclidean space. *Mathematics* **2018**, *6*, 306. [CrossRef]
4. Li, X.W.; Wang; Wu, Z.N.; Hou, L.K.; Liang, J.; Li, Q.Y. Hybrid second-order iterative algorithm for orthogonal projection onto a parametric surface. *Symmetry* **2017**, *9*, 146. [CrossRef]
5. Hu, S.-M.; Wallner, J. A second order algorithm for orthogonal projection onto curves and surfaces. *Comput. Aided Geometr. Des.* **2005**, *22*, 251–260. [CrossRef]
6. Li, X.W.; Wu, Z.N.; Pan, F.; Liang, J.; Zhang, J.F.; Hou, L.K. A gometric strategy algorithm for orthogonal projection onto a parametric surface. *J. Comput. Sci. Technol.* **2019**, *34*, 1279–1293. [CrossRef]
7. Ma, Y.L.; Hewitt, W.T. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Comput. Aided Geom. Des.* **2003**, *20*, 79–99. [CrossRef]
8. Chen, X.-D.; Yong, J.-H.; Zheng, G.-Q. Computing Minimum Distance between Two Implicit Algebraic Surfaces. *Comput.-Aided Des.* **2006**, *38*, 1053–1061. [CrossRef]
9. Kim, K.-J. Minimum Distance between A Canal Surface and A Simple Surface. *Comput.-Aided Des.* **2003**, *35*, 871–879. [CrossRef]
10. Lee, K.; Seong, J.K.; Kim, K.J.; Hong, S.J. Minimum distance between two sphere-swept surfaces. *Comput.-Aided Des.* **2007**, *39*, 452–459. [CrossRef]

11. William, H.P.; Brian, P.F.; Teukolsky, S.A.; William, T.V. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed.; Cambridge University Press: Cambridge, UK, 1992.
12. Morgan, A.P. Polynomial continuation and its relationship to the symbolic reduction of polynomial systems. In *Symbolic and Numerical Computation for Artificial Intelligence*; Academic Press: Cambridge, MA, USA, 1992; pp. 23–45.
13. Layne, T.W.; Billups, S.C.; Morgan, A.P. Algorithm 652: HOMPACK: A suite of codes for globally convergent homotopy algorithms. *ACM Trans. Math. Softw.* **1987**, *13*, 281–310.
14. Berthold, K.P.H. Relative orientation revisited. *J. Opt. Soc. Am. A* **1991**, *8*, 1630–1638.
15. Dinesh, M.; Krishnan, S. Solving algebraic systems using matrix computations. *ACM Sigsam Bull.* **1996**, *30*, 4–21
16. Chionh, E.-W. Base Points, Resultants, and the Implicit Representation of Rational Surfaces. Ph.D. Thesis, University of Waterloo, Waterloo, ON, Canada, 1990.
17. De Montaudouin, Y.; Tiller, W. The Cayley method in computer aided geometric design. *Comput. Aided Geom. Des.* **1984**, *1*, 309–326. [CrossRef]
18. Albert, A.A. *Modern Higher Algebra*; D.C. Heath and Company: New York, NY, USA, 1933.
19. Thomas, W.; David, S.; Anderson, C.; Goldman, R.N. Implicit representation of parametric curves and surfaces. *Comput. Vis. Graph. Image Proc.* **1984**, *28*, 72–84.
20. Nishita, T.; Sederberg, T.W.; Kakimoto, M. Ray tracing trimmed rational surface patches. *ACM SIGGRAPH Comput. Graph.* **1990**, *24*, 337–345. [CrossRef]
21. Elber, G.; Kim, M.-S. Geometric Constraint Solver Using Multivariate Rational Spline Functions. In Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, Ann Arbor, MI, USA, 4–8 June 2001; pp. 1–10.
22. Sherbrooke, E.C.; Patrikalakis, N.M. Computation of the solutions of nonlinear polynomial systems. *Comput. Aided Geom. Des.* **1993**, *10*, 379–405. [CrossRef]
23. Hartmann, E. On the curvature of curves and surfaces defined by normal forms. Comput. *Aided Geom. Des.* **1999**, *16*, 355–376. [CrossRef]
24. Li, X.W.; Pan, F.; Cheng, T.-X.; Wu, Z.N.; Liang, J.; Hou, L.K. Integrated hybrid second order algorithm for orthogonal projection onto planar implicit curve. *Symmetry* **2018**, *10*, 164. [CrossRef]
25. Nicholas, J.R. Implicit polynomials, orthogonal distance regression, and the closest point on a curve. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 191–199.
26. Wu, Z.N.; Li, X.W. An improved curvature circle algorithm for orthogonal projection onto a planar algebraic curve. *Mathematics* **2019**, *7*, 912. [CrossRef]
27. Aigner, M.; Jüttler, B. Robust computation of foot points on implicitly defined curves. In *Mathematical Methods for Curves and Surfaces*; Troms, Nashboro Press: Brentwood, TN, USA, 2004; pp. 1–10.
28. Hu, M.; Zhou, Y.; Li, X. Robust and accurate computation of geometric distance for Lipschitz continuous implicit curves. *Vis. Comput.* **2017**, *33*, 937–947. [CrossRef]
29. Bartoň, M. Solving polynomial systems using no-root elim- ination blending schemes. *Comput.-Aided Des.* **2011**, *43*, 1870–1878.
30. Van Sosin, B.; Elber, G. Solving piecewise polynomial constraint systems with decomposition and a subdivision-based solver. *Comput.-Aided Des.* **2017**, *90*, 37–47. [CrossRef]
31. Lopes, H.; Oliveira, J.B.; de Figueiredo, L.H. Robust adaptive polygonal approximation of implicit curves. *Comput. Graph.* **2002**, *26*, 841–852. [CrossRef]
32. Paiva, A.; de Carvalho Nascimento, F.; de Figueiredo, L.H.; Stolfi, J. Approximating implicit curves on triangulations with affine arithmetic. In Proceedings of the SIBGRAPI 2012, Ouro Preto, Brazil, 22–25 August 2012; pp. 94–101.
33. De Carvalho Nascimento, F.; Paiva, A.; De Figueiredo, L.H.; Stolfi, J. Approximating implicit curves on plane and surface triangulations with affine arithmetic. *Comput. Graph.* **2014**, *40*, 36–48. [CrossRef]
34. De Figueiredo, L.H.; Stolfi, J. Affine arithmetic: Concepts and applications. *Numer. Algorithms* **2004**, *37*, 147–158. [CrossRef]
35. Paiva, A.; Lopes, H.; Lewiner, T.; de Figueiredo, L.H. Robust adaptive meshes for implicit surfaces. In Proceedings of the SIBGRAPI 2006, Manaus, Brazil, 8–11 October 2006; pp. 205–212.

36. Park, C.H.; Elber, G.; Kim, K.J.; Kim, G.Y.; Seong, J.K. A hybrid parallel solver for systems of multivariate polynomials using CPUs and GPUs. *Comput.-Aided Des.* **2011**, *43*, 1360–1369. [CrossRef]
37. Bartoň, M.; Elber, G.; Hanniel, I. Topologically guaranteed univariate solutions of underconstrained polynomial systems via no-loop and single-component tests. *Comput.-Aided Des.* **2011**, *43*, 1035–1044.