axioms

MDPI

Article

Application of Orthogonal Polynomial in Orthogonal Projection of Algebraic Surface

Xudong Wang [1,†], Xiaowu Li [2,†] and Yuxia Lyu [3,*,†]

[1] School of Economics, Capital University of Economics and Business, Beijing 100070, China
[2] College of Data Science and Information Engineering, Guizhou Minzu University, Guiyang 550025, China
[3] School of Economics and Management, Shandong Youth University of Political Science, Jinan 250100, China
* Correspondence: yuxialv@126.com; Tel.: +86-135-0640-1186
† These authors contributed equally to this work.

Abstract: Point orthogonal projection onto an algebraic surface is a very important topic in computer-aided geometric design and other fields. However, implementing this method is currently extremely challenging and difficult because it is difficult to achieve to desired degree of robustness. Therefore, we construct an orthogonal polynomial, which is the ninth formula, after the inner product of the eighth formula itself. Additionally, we use the Newton iterative method for the iteration. In order to ensure maximum convergence, two techniques are used before the Newton iteration: (1) Newton's gradient descent method, which is used to make the initial iteration point fall on the algebraic surface, and (2) computation of the foot-point and moving the iterative point to the close position of the orthogonal projection point of the algebraic surface. Theoretical analysis and experimental results show that the proposed algorithm can accurately, efficiently, and robustly converge to the orthogonal projection point for test points in different spatial positions.

Keywords: point orthogonal projection; algebraic surface; orthogonal polynomial; Newton's gradient descent method; hybrid geometric accelerating orthogonal method

MSC: 39-XX; 65-XX; 68U05

Citation: Wang, X.; Lyu, Y.; Li, X. Application of Orthogonal Polynomial in Orthogonal Projection of Algebraic Surface. *Axioms* **2022**, *11*, 544. https://doi.org/10.3390/axioms11100544

Academic Editor: Serkan Araci

Received: 16 August 2022
Accepted: 29 August 2022
Published: 11 October 2022

1. Introduction

Orthogonal projection is an important topic in geometric modeling and computer-aided geometric design, etc. The concept of orthogonal projection and how to orthogonally project a spatial parametric curve onto a parametric surface and algebraic surface was first proposed by Pegna et al. [1]. The orthogonal projection involves finding a point on the curve or surface such that the line segment connected by this objective point and the given point is perpendicular to the tangent line or the tangent plane of the curve or the surface at this objective point. Since the distance projection is the extended form of the orthogonal projection, the study of this problem will greatly promote the study of orthogonal projections [1]. The study in [1] also presented many applications of orthogonal projections. In design, for the cutting, patching, and welding of free-form shell structures, such as in naval and aeronautical architecture, or in car body design, surfaces have to be cut along pre-defined trimming lines before their assembly. Such a trimming line is usually defined in parametric space for parametric surfaces and in three-dimensional space for implicit surfaces.

The orthogonal projection problem has been widely studied by many experts. A first-order tangent line perpendicular method for a point orthogonal projection onto a parametric curve and surface was proposed by Hartmann [2]. For the supplementation and the improvement of the first-order tangent line perpendicular method [2], Liang et al. [3] and Li et al. [4] proposed two hybrid second-order methods for the two same topics, respectively. Hu et al. [5] proposed a second-order geometric curvature information mode

for approximating the orthogonal projection problem of parametric curves and surfaces. Based on their work [5], Li et al. [6] proposed an improved method for orthogonal projection onto a general parametric surface, such that the efficiency in [6] was improved, compared with the existing methods. Ma et al. [7] proposed the topic for point orthogonal projection onto NURBS curves and surfaces, including a four-step technique: subdividing the curve or surface into curve segments or surface patches, finding out the corresponding control polygon of the curve segments or surface patches, identifying the candidate curve segment or surface patch, and confirming some candidate projection points by comparison, with the final projective point being obtained by comparing the distance between the test point and these candidate projection points. Due to the minimum distance between two geometry objects being generated between a pair of special points, researchers [8–10] have studied the minimum distance problem between some specific geometric objects using the specific geometric property, and reached some satisfactory results.

Since the algebraic curve and algebraic surface do not have any parameter control form, finding out the orthogonal projection point on the algebraic curve and surface is very difficult. However, there are many fields such as geometric modeling, computer graphics, computer-aided geometric design, etc., that need to be addressed with point orthogonal projection onto algebraic curves and surfaces problems, which makes this a particularly important topic.

Presently, the research into point orthogonal projections onto algebraic curves and surfaces mainly involves point orthogonal projections onto planar algebraic curves. However, there is little research on point orthogonal projection onto an algebraic surface. There are more than 10 papers on point orthogonal projections onto planar algebraic curves, and they are divided into three types: local method, global method, and compromise method between the two methods. It can be seen from the basic geometric characteristic that the problem of point orthogonal projections onto planar algebraic curves can be transformed into solving the equation where the cross product of gradient $\nabla f_0(\mathbf{X})$ of the planar algebraic curve $f_0(\mathbf{X})$ and the vector $\overrightarrow{\mathbf{PX}}$ is zero at point $\mathbf{X}$. The specific equation can be expressed in the following form:

$$\nabla f_0(\mathbf{X}) \times (\mathbf{P} - \mathbf{X}) = 0 \tag{1}$$

The corresponding Newton's gradient descent iterative formula of Equation (1) can be expressed as the formula (2),

$$\mathbf{X}_{n+1} = \mathbf{X}_n - (f_0(\mathbf{X}_n) / \langle \nabla f_0(\mathbf{X}_n), \nabla f_0(\mathbf{X}_n) \rangle) \nabla f_0(\mathbf{X}_n). \tag{2}$$

The second local iteration method is combined with the Lagrange multiplier and Newton's gradient descent method for computing the orthogonal projection point on the planar algebraic curve, as in William et al. [11]. Of course, the combined method [11] is fast but local, and dependent on the initial point.

As for the Newton's gradient descent method (2) for solving Equation (1) with an orthogonal projection problem of the planar algebraic curve, the first global approach for solving the system of nonlinear equations is the homotopy continuous method [12,13]. The most classical homotopy transformation technique adopted for solving the orthogonal projection problems of planar algebraic curves is the following:

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

where t is a continuous parameter from 0 to 1, and $\mathbf{P}(\mathbf{X}) = 0$ and $\mathbf{Q}(\mathbf{X}) = 0$ are the original function to be solved and the objective solution function, respectively. All isolated solutions of the original function system $\mathbf{P}(\mathbf{X}) = 0$ can be obtained using the global homotopy method [12,13], where all the isolated solutions of $\mathbf{P}(\mathbf{X}) = 0$ are exactly the same as the objective solution function $\mathbf{Q}(\mathbf{X}) = 0$. The robustness of the global continuous homotopy methods [12,13] with convergence is confirmed by [14], and their low efficiency is shown

in [15]. The greatest difficulty of the homotopy method is in seeking out a very satisfactory and correct objective function $\mathbf{Q}(\mathbf{X}) = 0$.

The second global method for point orthogonal projection onto a planar algebraic curve problem is the global resultants method [16–19], which transformed the projection problem into a results system. By using the elimination theory, a nonlinear system of equations with two variables can be turned into a resultant polynomial with one variable, where the characteristic of the solutions is equivalent to the original function system with two equations. The most important and classical resultant methods [16–19] are Sylvester's resultant and Cayley's resultant formed by Bézout's method. The advantage of the global resultant methods [16–19] is that the degree of the planar algebraic curve must be less than 5. Therefore, all the roots of the univariate nonlinear polynomial equation yielded using the resultant methods can be completely solved. However, the resultant methods [16–19] fail to solve all roots of the nonlinear polynomial equation with a degree of 5 or more.

The third global technique is the Bézier clipping method [20–22]. Turning Equation (1) into the Bézier form with a convex hull property is the first step of the Bézier clipping technique. The remaining processing steps are completely equivalent to those of Ma et al. [7], the detailed description of which is omitted here. The advantage of the global Bézier clipping method is that all roots can be obtained, or all orthogonal projection points can be yielded using this technique. There are two disadvantages of the global clipping method: the first is that it takes a lot of time to subdivide, to seek, to judge, and to compare; the second is that transforming Equation (1) into the Bernstein–Bézier form for a small part of the planar algebraic curves is very difficult or even impossible.

The fourth global technique is to solve all the roots of the equations and systems of equations [23–26]. That is to say, the point orthogonal projection algebraic surface problem can be transformed into solving all the roots of the equations and systems of equations [23–26]. Bartoň M. [23] proposed two blending scheme solvers for the problem of finding all the roots. As a system of nonlinear equations, a simple linear combination of functions is realized for eliminating the no-root domain, and then all control points for its Bernstein–Bézier basis can be determined, having the same sign, which must be in accord with the seeking function. Then, through the continuous subdivision process, these types of functions are obtained to eliminate the no-root domains. Therefore, two blending schemes in [23] can efficiently reduce the number of subdivisions. van Sosin B. and Elber G. [24] constructed a variety of complex piecewise polynomial systems with zero or inequality constraints in zero-dimensional or one-dimensional solution spaces. To overcome the time cost of subdivision, Park, C.H. et al. [25] presented a hybrid parallel algorithm for solving systems of multivariate constraints by exploiting both the CPU and the GPU multicore architectures. This was achieved by decomposing the constraint solving technique into two different components, hierarchy traversal and polynomial subdivision, each of which is more suitable to CPUs and GPUs, respectively, whose solver can fully exploit the availability of the hybrid, multicore architectures of CPUs and GPUs. The proposed parallel method improved the performance compared to the state-of-the-art subdivision-based CPU solver. To further facilitate solving for all of the roots, Bartoň M. et al. [26] proposed a subdivision model of topological guarantee, whose core technique is to project the unknown multivariable region of the high-dimensional space to the two-dimensional plane according to the known region of all bounds of the univariate. The advantage of these four classical methods [23–26] is that the robustness of solving all the roots is very good, but the time consumption is relatively large.

In addition to the local and global methods, the third type of approach is the compromise method. The first compromise method was proposed by Hartmann [2,27] and it includes the geometric tangent perpendicular property for solving the orthogonal projection problem. The iterative formula (2) is repeatedly run until the iterative point iterates to the planar algebra curve. The iterative point on the planar algebraic curve is used as the initial point, and the foot-point is once again calculated using Equation (4),

$$\mathbf{Q} = \mathbf{P} - (\langle \mathbf{P} - \mathbf{Y}_n, \nabla f_0(\mathbf{Y}_n) \rangle / \langle \nabla f_0(\mathbf{Y}_n), \nabla f_0(\mathbf{Y}_n) \rangle) \nabla f_0(\mathbf{Y}_n). \tag{4}$$

The foot-point $\mathbf{Q}$ is used again as the initial iteration point of Equation (2), and the above two iterative formulas are repeatedly run until the foot-point $\mathbf{Q}$ completely collides with the orthogonal projection point. Unfortunately, if the test point is far away from a plane algebraic curve or algebraic surface, the foot-point $\mathbf{Q}$ being the next iterative point will cause more errors, and finally lead to non-convergence for a small part of the planar algebraic curves.

Redding [28] presented the second compromise method, which adopted the osculating circle technique for computing the orthogonal projection point of the planar algebraic curve. The osculating circle technique mainly consists of three important steps: (1) Computing the curvature and the corresponding radius and center of the curvature circle of the planar algebraic curve at the point. (2) Computing the line segment determined by the test point and the center of the curvature circle, and identifying the foot-point $\mathbf{Q}$ intersected with the line segment and the curvature circle. (3) Approximately taking the foot-point $\mathbf{Q}$ as the point being on the planar algebraic curve. The above three steps are run repeatedly until the foot-point $\mathbf{Q}$ is the same as the orthogonal projection point $\mathbf{P}_\Gamma$. Since the perpendicular foot-point regarded as the point on the planar algebraic curve will result in more errors and deviations in the third step of the osculating circle technique, and it is not representative of the original parameter, the convergence robustness of the osculating circle technique is sometimes not guaranteed.

The third compromise method is the circle shrinking technique [29]. Equation (2) is run repeatedly such that the iterative point $\mathbf{P}_\mathtt{I}$ can iterate to the planar algebraic curve maximally. Construct a circle whose the center and radius are the test point $\mathbf{P}$ and $\|\mathbf{P} - \mathbf{P}_\mathtt{I}\|$, respectively. Mark a point $\mathbf{P}^+$ on the circle by using the mean value theorem, and find out the intersection point between the line segment $\overline{\mathbf{P}\mathbf{P}^+}$ and the circle. We name this intersection point the current iterative point $\mathbf{P}_\mathtt{I}$. Repeatedly iterate the above behavior until the current point $\mathbf{P}_\mathtt{I}$ and the previous point $\mathbf{P}_\mathtt{I}$ are completely overlapping. It takes more time to find point $\mathbf{P}^+$ each time for the circle shrinking technique [29]. At the same time, and if the degree of the planar algebraic curve is more than 5, it is not easy to directly solve the intersection $\mathbf{P}_\mathtt{I}$ of line segment $\overline{\mathbf{P}\mathbf{P}^+}$ and the planar algebraic curve using the circle shrinking technique [29].

The fourth compromise method is associated with the circle double-and-bisect algorithm presented by Hu et al. [30]. Draw an initial small circle with the test point $\mathbf{P}$ as the center, and an arbitrarily small length as the radius r_1. A new circle with the same center $\mathbf{P}$ and radius $r_2 = 2r_1$(after that, the center of all circles is the test point $\mathbf{P}$) is drawn once again. If the second new circle and the planar algebraic curve do not intersect, redraw a new circle, such that the radius of the new circle is twice that of r_2. Repeat the above behavior until the latest circle and the planar algebraic curve are intersected. The previous circle and the latest circle are named as the interior circle and the exterior circle, respectively. Once the latest circle intersects with the planar algebraic curve, the remaining processing technique adopts the bisecting technology. A new circle with new radius $r = (r_1 + r_2)/2$ is continuously drawn. If the current circle with radius r and the planar algebraic curve intersect, substitute r for r_2, or else, for r_1. Repeatedly run the above action until the interior circle and the exterior circle are completely coincident. However, it is difficult to determine using this method whether the exterior circle intersects with the curve or not [30], if the degree of the planar algebraic curve is more than 5. Additionally, it takes more time to find the intersection between the exterior circle and the planar algebraic curve in the double-and-bisect algorithm [30]. In addition, in the third compromise method [29] and the fourth compromise method [30], they have a common processing technology that needs to judge the sign of the function $f_0(\mathbf{Q})$. If the topological structure of the planar algebraic curve is complex, or if there are many branches in the planar algebraic curve, it is not easy for the two compromise methods [29,30] to implement this technical link.

The fifth compromise method was proposed by Cheng T. et al. [31], and it is a point orthogonal projection onto a spatial algebraic curve. Its shortcoming is that the effect of

the correction method of the third algorithm is not ideal, which leads to a reduction in efficiency.

Orthogonal polynomials not only play an important role in point orthogonal projection onto an algebraic surface, but they also have many important theoretical and application values in other aspects. Cesarano C. [32] proved the existence and uniqueness of the extremal node in the polynomial system for any fixed system of multiplicities. From the standard definitions of the incomplete two-variable Hermite polynomials, Cesarano C. et al. [33] proposed a non-trivial generalization polynomial with the Bessel-type functions as the Humbert functions and a non-trivial generalization Lagrange polynomial. Dattoli G. et al. [34] discussed the theory of Lagrange polynomials associated with generalized forms. They adopted two different approaches based on the integral transform method and the Umbral Calculus.

In a word, from the above literature description and analysis, the robustness of the point orthogonal projection onto an algebraic surface is still a very difficult issue to overcome. In order to improve the robustness and efficiency, we construct an orthogonal polynomial (Equation (9)) and use the Newton iterative method for iteration.

The proposed algorithm mainly contains three sub-algorithms: Algorithms 1–3. Equation (11) causes the initial point to be on the algebraic surface as much as possible, according to Newton's gradient descent property. Then, the iteration point falls on the algebraic surface completely. After Step 2 and Step 3 of Algorithm 2 are jointly implemented five times, the first iteration point falling on the algebraic surface is gradually moved very close to the position of the orthogonal projection point. Additionally, the final iteration point conforms to Newton's local iterative convergence condition of the two sub-equations of Equation (15). In this way, after repeatedly running Equation (15) of Algorithm 3, the iterative point converges to the objective point (the orthogonal projection point) quickly and robustly. The proposed algorithm mainly captures three important geometric features. First, it maximizes the effect of Newton's gradient descent method; that is, when each sub-algorithm is implemented, the iteration point can always fall on the algebraic surface, which is a particularly important action for improving the robustness. Second, Algorithm 2 ensures that on the basis of the iteration point on the algebraic surface, the iteration point moves very close to the position of the orthogonal projection point. Third, the final iteration point of Algorithm 2 satisfies Newton's local convergence condition of Algorithm 3. Algorithm 3 can quickly iterate the iteration point to the algebraic surface, and also speed up the orthogonalization (the final iteration point coincides with the orthogonal projection point).

Algorithm 1: Newton's gradient descent method.

Input: The test point $\mathbf{P}$ and the algebraic surface $f(\mathbf{X})$
Output: The iterative point $\mathbf{P}_{\mathtt{I}}$ on the algebraic surface $f(\mathbf{X})$
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 Equation (11);
}while $(|f(\mathbf{X}_{n+1})| > \varepsilon \&\& \|\mathbf{X}_{n+1} - \mathbf{X}_n\| > \varepsilon)$;
Step 2:
$\mathbf{P}_{\mathtt{I}} = \mathbf{X}_{n+1}$;
Return $\mathbf{P}_{\mathtt{I}}$;

Algorithm 2: Computing the foot-point **Q** and moving the iterative point $\mathbf{P}_{\mathrm{I}}$ to the close position of the orthogonal projection point $\mathbf{P}_{\Gamma}$.

Input: The test point **P**, the algebraic surface **S**, and the iterative point $\mathbf{P}_{\mathrm{I}}$.
Output: The current iterative point $\mathbf{P}_{\mathrm{I}}$ to the close position of the orthogonal projection point $\mathbf{P}_{\Gamma}$.
Description:
Step 1: With the neighbor point of the test point **P** as the initial point, obtain the iterative point $\mathbf{P}_{\mathrm{I}}$ on the algebraic surface **S** via Algorithm 1.
for(i = 0; i<5; i++) {
Step 2: Obtain the foot-point **Q** via Equation (14).
Step 3: With the foot-point **Q** as the initial point of Equation (11), compute the iterative point $\mathbf{P}_{\mathrm{I}}$ on the algebraic surface **S** via Algorithm 1.
}
Step 4: Return $\mathbf{P}_{\mathrm{I}}$;

Algorithm 3: Hybrid geometric accelerated orthogonal method.

Input: The current iterative point $\mathbf{P}_{\mathrm{I}}$ on the algebraic surface $f(\mathbf{X})$ and the algebraic surface $f(\mathbf{X})$.
Output: The corresponding orthogonal projection point $\mathbf{P}_{\Gamma}$ of the test point **P**.
Description:
Step 1:
$\mathbf{X}_{n+1} = \mathbf{P}_{\mathrm{I}}$;
Do {
$\mathbf{X}_{n} = \mathbf{X}_{n+1}$;
Compute $\mathbf{X}_{n+1}$ by using the iterative formula (15);
}while ($\|\mathbf{X}_{n+1} - \mathbf{X}_{n}\|^2 > \varepsilon \&\& |f(\mathbf{X}_{n+1})| > \varepsilon$)
Step 2:
$\mathbf{P}_{\Gamma} = \mathbf{X}_{n+1}$;
Return $\mathbf{P}_{\Gamma}$;

2. Implementation of the Hybrid Geometry Strategy Algorithm

Let us elaborate on the general idea. There is an algebraic surface **S**, where the equation of the algebraic surface is,

$$f(\mathbf{X}) = 0, \tag{5}$$

where $\mathbf{X} = (x, y, z)$. Our aim is to find a point $\mathbf{P}_{\Gamma}$ on the algebraic surface **S** via a spatial test point **P**, such that the relationship could be satisfied (see Figure 1),

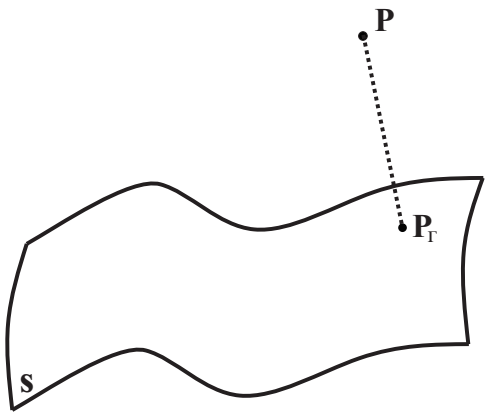

Figure 1. Point orthogonal projection onto an algebraic surface.

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

The above problem can be written as

$$\begin{cases} f(\mathbf{P}_\Gamma) = 0, \\ \|\mathbf{P}-\mathbf{P}_\Gamma\| = \min_{\mathbf{X}\in\Gamma}\|\mathbf{P}-\mathbf{X}\|, \end{cases} \tag{7}$$

where the symbol $\|\cdot\|$ is norm. Formula (7), related to the orthogonal projection point $\mathbf{P}_\Gamma$, provides two important criteria where the orthogonal projection point $\mathbf{P}_\Gamma$ should fall on the algebraic surface $\mathbf{S}$, and the distance $\|\mathbf{P}-\mathbf{P}_\Gamma\|$ is the shortest between the test point $\mathbf{P}$ and point $\mathbf{X}$ on the algebraic surface $\mathbf{S}$. From the second formula of Formula (7), the third important implied and potential geometric meaning that can also be indicated is that the vector $\overrightarrow{\mathbf{PP}_\Gamma}$ is perpendicular to the tangent plane vector of the algebraic surface $\mathbf{S}$ at the orthogonal projection point $\mathbf{P}_\Gamma$, or that the cross product of the vector $\overrightarrow{\mathbf{PP}_\Gamma}$ and the gradient vector $\nabla f(\mathbf{P}_\Gamma)$ is zero. Namely, we seek that the orthogonal projection point $\mathbf{P}_\Gamma$ (the objective point) should be satisfied with the relationship where the cross product of the vector $\overrightarrow{\mathbf{PX}}$ and the gradient vector $\nabla f(\mathbf{X})$ is zero,

$$F_0(\mathbf{X}) = \nabla f(\mathbf{X}) \times (\mathbf{P}-\mathbf{X}) = 0, \tag{8}$$

where $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]$ and symbol $\times$ are the Hamiltonian operator and the cross product, respectively. Since Equation (8) is a vector equation, not a scalar equation, it is not easy to solve Equation (8). Taking the inner product of the vector itself $\nabla f(\mathbf{X}) \times (\mathbf{P}-\mathbf{X})$ of Equation (8), it is transformed into the following corresponding equation, which is easy to solve,

$$F(\mathbf{X}) = \langle \nabla f(\mathbf{X}) \times (\mathbf{P}-\mathbf{X}), \nabla f(\mathbf{X}) \times (\mathbf{P}-\mathbf{X}) \rangle = 0, \tag{9}$$

where the symbol $\langle \cdot, \cdot \rangle$ is the inner product. The orthogonal projection $\mathbf{P}_\Gamma$ of the test point $\mathbf{P}$ orthogonally projecting onto the algebraic surface $\mathbf{S}$ should have three important geometric properties: (1) The orthogonal projection point $\mathbf{P}_\Gamma$ should fall onto the algebraic surface $\mathbf{S}$; (2) the distance $\|\mathbf{P}-\mathbf{P}_\Gamma\|$ is the shortest between the test point $\mathbf{P}$ and point $\mathbf{X}$ on the algebraic surface S; (3) the inner product of the vector $\overrightarrow{\mathbf{PP}_\Gamma}$ and the tangent plane vector of the algebraic surface $\mathbf{S}$ at the orthogonal projection point $\mathbf{P}_\Gamma$ is zero, or the cross product of the vector $\overrightarrow{\mathbf{PP}_\Gamma}$ and the gradient vector $\nabla f(\mathbf{P}_\Gamma)$ is zero. These three important geometric properties can be expressed specifically as,

$$\begin{cases} f(\mathbf{P}_\Gamma) = 0, \\ F(\mathbf{P}_\Gamma) = \langle \nabla f(\mathbf{P}_\Gamma) \times (\mathbf{P}-\mathbf{P}_\Gamma), \nabla f(\mathbf{P}_\Gamma) \times (\mathbf{P}-\mathbf{P}_\Gamma) \rangle = 0, \\ \|\mathbf{P}-\mathbf{P}_\Gamma\| = \min_{\mathbf{X}\in\Gamma}\|\mathbf{P}-\mathbf{X}\|, \end{cases} \tag{10}$$

where $F(\mathbf{X}) = \langle \nabla f(\mathbf{X}) \times (\mathbf{P}-\mathbf{X}), \nabla f(\mathbf{X}) \times (\mathbf{P}-\mathbf{X}) \rangle$.

2.1. Newton's Gradient Descent Method

The corresponding Newton's gradient descent iterative formula related to Equation (5) is as follows,

$$\mathbf{X}_{n+1} = \mathbf{X}_n - (f(\mathbf{X}_n)/\langle \nabla f(\mathbf{X}_n), \nabla f(\mathbf{X}_n) \rangle) \nabla f(\mathbf{X}_n). \tag{11}$$

The purpose of this iterative Formula (11) is to prompt the iterative point to iterate to the algebraic surface according to the Newton's gradient descent property. A detailed description of the idea can be concretely expressed as Algorithm 1 (see Figure 2).

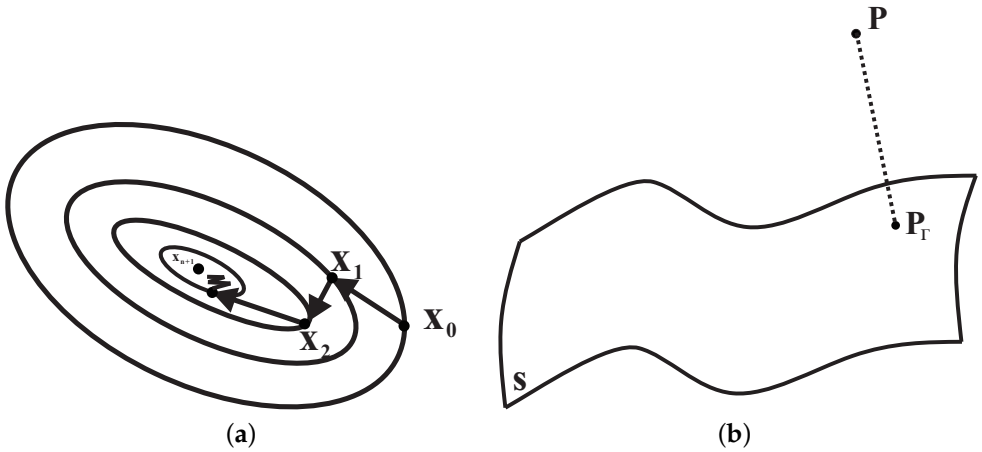

Figure 2. The complete schematic of Algorithm 1. (**a**) Newton's gradient descent method; (**b**) Point orthogonal projection onto algebraic surface. All the curves of figures (**a**) denote contour surfaces and not contour curves.

Remark 1. *We present a geometric interpretation for Algorithm 1. In Figure 2, each closed loop is actually represented as a contour surface. The outermost layer of the contour surface is called the first contour surface, the second layer of the contour surface is called the second contour surface, followed by the third contour surface, the fourth contour surface, and so on. The initial iteration point of Equation (11)* $\boldsymbol{X}_0$ *upon the first contour surface has Newton's gradient descent property. Using the first iteration of Equation (11), the first iteration point* $\boldsymbol{X}_1$ *falls on the second contour surface, and the vector* $\overrightarrow{\boldsymbol{X}_0\boldsymbol{X}_1}$ *is determined by the gradient vector* $\nabla f(\boldsymbol{X}_0)$*, which is tangential to the second contour surface. Immediately, after the second iteration of Equation (11),* $\boldsymbol{X}_1$ *is an initial iterative point, and the gradient vector* $\nabla f(\boldsymbol{X}_1)$ *is perpendicular to the gradient vector* $\nabla f(\boldsymbol{X}_0)$*. The second iteration point* $\boldsymbol{X}_2$ *is fallen onto the third contour surface, and the vector* $\overrightarrow{\boldsymbol{X}_1\boldsymbol{X}_2}$ *is determined using the gradient vector* $\nabla f(\boldsymbol{X}_1)$*, which is tangential to the third contour surface. Equation (11) is iterated repeatedly in this way; the iteration point* $\boldsymbol{X}_{n+1}$ *finally falls on the innermost contour surface. Of course, the innermost contour surface almost becomes a point that has fallen on the algebraic surface* $\boldsymbol{S}$*. The first property of Newton's gradient descent indicates that the absolute value* $|f(\boldsymbol{X})|$ *of the algebraic surface* $f(\boldsymbol{X})$ *decreases fastest along the opposite direction to the gradient vector* $\nabla f(\boldsymbol{X})$*. That is, for every iteration, the absolute value of the algebraic surface is rapidly smaller than the absolute value before the iteration* ($|f(\boldsymbol{X}_{k+1})| < |f(\boldsymbol{X}_k)|$)*, until the absolute value of the algebraic surface is almost zero* ($|f(\boldsymbol{X})| \approx 0$)*, and the iteration is terminated. The second property of Newton's gradient descent method is to select a direction with the largest slope from the current position to perform the next step. In the current position, the third property of Newton's gradient descent method is to fit the algebraic surface* $\boldsymbol{S}$ *using the quadric as the local surface. Then, the path chosen by the Newton's gradient descent method will be more consistent with the real optimal descent path. Therefore, Equation (11) causes the initial point* $\boldsymbol{X}_0$ *to be on the algebraic surface as much as possible according to the Newton's gradient descent property. Finally, the iteration point falls on the algebraic surface completely.*

2.2. Moving the Iterative Point $\boldsymbol{P}_I$ *to the Close Position of the Orthogonal Projection Point* $\boldsymbol{P}_\Gamma$

From Algorithm 1, Equation (11) of Algorithm 1, characterized by the Newton's gradient descent property, is to prompt the initialization point $\mathbf{X}_0$ to fall on the algebraic surface $\mathbf{S}$ maximally. The iterative point on the algebraic surface is named as the point $\mathbf{P}_\mathrm{I}$. After many tests, we find that the point $\mathbf{P}_\mathrm{I}$ is not far from the orthogonal projection point $\mathbf{P}_\Gamma$, but there is still a certain distance. Our idea is to make the iterative point $\mathbf{P}_\mathrm{I}$ as close to the orthogonal projection point $\mathbf{P}_\Gamma$ as much as possible. That is, to let the point $\mathbf{P}_\mathrm{I}$ gradually move to the position near the orthogonal projection point $\mathbf{P}_\Gamma$. This serves to lay a good foundation for the subsequent orthogonalization. In order to bring the iteration point $\mathbf{P}_\mathrm{I}$ closer to the orthogonal projection point $\mathbf{P}_\Gamma$, we adopt the tangent plane vertical foot

technique to achieve this goal. The basic goal of this technique can be brought about by the following mode. We draw a vertical foot-point $\mathbf{Q}_0$ of the tangent plane derived from the iterative point $\mathbf{P}_\mathtt{I}$ via the test point $\mathbf{P}$. Its expression is the following,

$$\mathbf{Q}_0 = \mathbf{P} - (\langle \mathbf{P} - \mathbf{P}_\mathtt{I}, \nabla f(\mathbf{P}_\mathtt{I}) \rangle / \langle \nabla f(\mathbf{P}_\mathtt{I}), \nabla f(\mathbf{P}_\mathtt{I}) \rangle) \nabla f(\mathbf{P}_\mathtt{I}). \tag{12}$$

At this moment, taking the foot-point $\mathbf{Q}_0$ as the initial point of Algorithm 1, we obtain the new iterative point $\mathbf{P}_\mathtt{I}$ on the algebraic surface $\mathbf{S}$, where the new iterative point $\mathbf{P}_\mathtt{I}$ is named as the current iterative point $\mathbf{P}_\mathtt{I}$. Based on our understanding and basic geometry properties, let us take the point closer to the algebraic surface as the foot-point. In order to make the foot-point $\mathbf{Q}$ close to the algebraic surface and the orthogonal projection point $\mathbf{P}_\Gamma$ at the same time, combining with geometric intuition, we select a foot-point $\mathbf{Q}$ on the line segment $\overline{\mathbf{P}\mathbf{Q}_0}$ to satisfy the relationship

$$\frac{\overrightarrow{\mathbf{Q}_0\mathbf{Q}}}{\overrightarrow{\mathbf{Q}_0\mathbf{P}}} = d, \tag{13}$$

where $d = \dfrac{d_1}{d_2}$, $d_1 = 0.5 \cdot \left\| \overrightarrow{\mathbf{P}_\mathtt{I}\mathbf{Q}_0} \right\|$, $d_2 = \left\| \overrightarrow{\mathbf{Q}_0\mathbf{P}} \right\|$. According to Equation (13), it is easy to obtain the following foot-point form $\mathbf{Q}$ (see Figure 3a),

$$\mathbf{Q} = d \cdot (\mathbf{P} - \mathbf{Q}_0) + \mathbf{Q}_0. \tag{14}$$

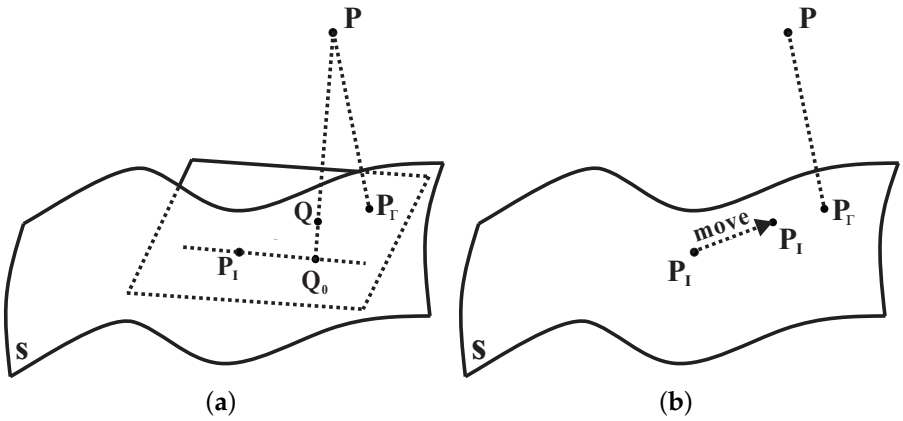

Figure 3. The complete graphic demonstration of Algorithm 2. (**a**) Computing the foot-point $\mathbf{Q}$ of Equation (14); (**b**) Moving the iterative point $\mathbf{P}_\mathtt{I}$ to the close position of the orthogonal projection point $\mathbf{P}_\Gamma$.

Now, running Equation (14) and Algorithm 1 repeatedly, the iterative point $\mathbf{P}_\mathtt{I}$ can converge and move to the close position of the orthogonal projection point $\mathbf{P}_\Gamma$. The detailed description can be expressed as Algorithm 2 (see Figure 3b).

Remark 2. *We present a geometric interpretation for Algorithm 2. From Remark 1, the purpose of Algorithm 1 is to let the initial iteration point be on the algebraic surface maximally. Starting from the iterative point $\boldsymbol{P}_I$, the foot-point $\boldsymbol{Q}$ is derived by Formula (14). According to our geometric intuition, the vertical point $\boldsymbol{Q}_0$ is located between the iterative point $\boldsymbol{P}_I$ and the orthogonal projection point $\boldsymbol{P}_\Gamma$. There are four advantages to choosing point $\boldsymbol{Q}$ instead of point $\boldsymbol{Q}_0$ as the foot-point. Not only is the foot-point $\boldsymbol{Q}$ located between the iterative point $\boldsymbol{P}_I$ and the orthogonal projection point $\boldsymbol{P}_\Gamma$, but the distance between the foot-point $\boldsymbol{Q}$ and the algebraic surface $\boldsymbol{S}$ is also shorter that the distance between the vertical point $\boldsymbol{Q}_0$ and the algebraic surface $\boldsymbol{S}$. Consequently, the iteration time with Algorithm 1 from the foot-point $\boldsymbol{Q}$ being used as the initialization point to iterate to the algebraic surface is less than that from the perpendicular point $\boldsymbol{Q}_0$ being used as the initialization point to iterate to the algebraic surface. The fourth advantage is that the closer the foot-point $\boldsymbol{Q}$ is to*

the algebraic surface, the higher the stability of the iterative point $\boldsymbol{P}_I$ obtained through Algorithm 1. This means that the distance between the iterative point $\boldsymbol{P}_I$ fallen on the algebraic surface $\boldsymbol{S}$ caused by Algorithm 1 with the foot-point $\boldsymbol{Q}$ as the initial iterative point and the orthogonal projection point $\boldsymbol{P}_\Gamma$ is not much longer than the distance between the iterative point $\boldsymbol{P}_I$ fallen on the algebraic surface $\boldsymbol{S}$ caused by Algorithm 1 with the perpendicular point $\boldsymbol{Q}_0$ as the initial iterative point and the orthogonal projection point $\boldsymbol{P}_\Gamma$. After Step 2 and Step 3 are jointly implemented five times, the first iteration point $\boldsymbol{P}_I$ fallen on the algebraic surface $\boldsymbol{S}$ is gradually moved very close to the position of the orthogonal projection point $\boldsymbol{P}_\Gamma$.

2.3. Hybrid Geometric Accelerating Orthogonal Method

From Section 2.2, it is not difficult to know that the current iterative point $\mathbf{P}_\mathrm{I}$ is not only fallen on the algebraic surface, but is also very close to the orthogonal projection point $\mathbf{P}_\Gamma$. If we convert the **for** loop body in Algorithm 2 into the **do** ... **while** loop body and the termination criteria are that the distance between the previous iterative point $\mathbf{P}_\mathrm{I}$ and the current iterative point $\mathbf{P}_\mathrm{I}$ is zero and the absolute value of the function $f(\mathbf{P}_\mathrm{I})$ is almost zero, the slightly improved version of Algorithm 2 is intrinsically and completely equivalent to the foot-point algorithm for an implicit surface in [27]. The slightly improved version of Algorithm 2 is robust and efficient for less partial algebraic surfaces; however, it cannot ensure convergence for total algebraic surfaces. Even if the slightly improved version of Algorithm 2 converges, the moving speed of the iteration point $\mathbf{P}_\mathrm{I}$ to the orthogonal projection point $\mathbf{P}_\Gamma$ is occurring very slowly, and the speed of the cross product being zero is determined by the vector $\overrightarrow{\mathbf{PP}_\mathrm{I}}$, and the vector $\nabla f(\mathbf{P}_\mathrm{I})$ is also very slow.

In order to improve the convergence rate, and to accelerate the satisfaction of the first two formulas in Equation (10),

$$\begin{cases} \mathbf{Y}_n = \mathbf{X}_n - (\langle f(\mathbf{X}_n)\rangle / \langle \nabla f(\mathbf{X}_n), \nabla f(\mathbf{X}_n)\rangle)\nabla f(\mathbf{X}_n), \\ \mathbf{X}_{n+1} = \mathbf{Y}_n - (\langle F(\mathbf{Y}_n)\rangle / \langle \nabla F(\mathbf{Y}_n), \nabla F(\mathbf{Y}_n)\rangle)\nabla F(\mathbf{Y}_n), \end{cases} \tag{15}$$

where $\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}]|_{\mathbf{X}_n}$, $\nabla F = [\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}]|_{\mathbf{Y}_n}$. The algorithm formed by this iteration (15) for accelerating orthogonality and falling on the algebraic surface can be described by Algorithm 3 (see Figure 4).

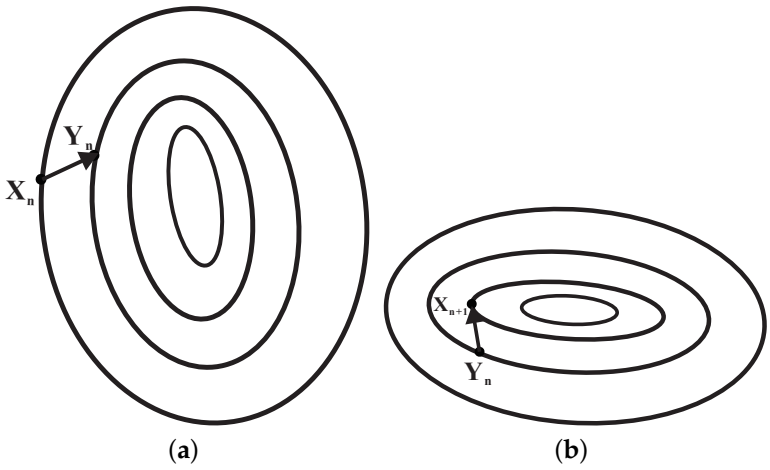

Figure 4. The whole graphic demonstration of Algorithm 3. (**a**) Newton's gradient descent method unrelated to the test point of the first formula in Equation (15); (**b**) Newton's gradient descent method associated with the test point of the second formula in Equation (15); All the curves of figures (**a**,**b**) denote contour surfaces and not contour curves.

Remark 3. *We present a geometric description for Algorithm 3. After performing Algorithm 2, the current iterative point $\boldsymbol{P}_I$ is not only fallen on the algebraic surface, but the distance between the current iteration point $\boldsymbol{P}_I$ and the orthogonal projection point $\boldsymbol{P}_\Gamma$ is also significantly smaller than*

the distance between the previous iteration point $\boldsymbol{P}_I$ and the orthogonal projection point $\boldsymbol{P}_\Gamma$. That is, the current iteration point $\boldsymbol{P}_I$ is closer to the orthogonal projection point $\boldsymbol{P}_\Gamma$, and after several iterations, the distance between the current iteration point $\boldsymbol{P}_I$ and the orthogonal projection point $\boldsymbol{P}_\Gamma$ is very small. In this case, the current iteration point $\boldsymbol{P}_I$ accords with the local convergence condition of the Newtonian type of Equation (15), which ensures the successful iterative convergence of Equation (15).

The purpose of the first formula of Equation (15) is to ensure that the iterative point can iterate to the algebraic surface maximally, according to the geometrical property of Newton's gradient descent method. The prototype formula of the second formula of Equation (15) is Formula (9), the geometric essence of which is to seek out a point $\mathbf{X}$ on the algebraic surface. Therefore, we make the vector $\overrightarrow{\mathbf{PX}}$ be perpendicular to the tangent plane of the algebraic surface at the point $\mathbf{X}$. That is to say, we use Formula (9), where we seek the point $\mathbf{X}$ on the algebraic surface, as it plays an important role in accelerating orthogonalization. Namely, every iteration of the second formula of the iterative Formula (15) corresponding to Formula (9) is to ensure that the absolute value of Equation (9) with the expression $F(\mathbf{Y}_n)$ becomes smaller or even zero under the condition that the initial iterative point falls on the algebraic surface. Although the second iterative formula of Equation (15) is a locally convergent Newton-type iterative formula, it can be seen from the final iteration point $\mathbf{P}_\mathtt{I}$ of Algorithm 2 that the final iteration point $\mathbf{P}_\mathtt{I}$ conforms to the Newton's local iterative convergence condition of the two sub-equations of Equation (15), so that the iteration of Equation (15) can converge successfully. In this way, we repeatedly run Equation (15), and the iterative point converges to the objective point (the orthogonal projection point $\mathbf{P}_\Gamma$) quickly and robustly.

Through the above comprehensive analysis, we obtain Algorithm 4, which is the complete algorithm on the point orthogonal projection onto the algebraic surface (see Figure 5).

Algorithm 4: The complete hybrid geometry strategy algorithm for point orthogonal projection onto an algebraic surface.

Input: Test point $\mathbf{P}$ and the algebraic surface $f(\mathbf{X}) = 0$.
Output: Final orthogonal projection point $\mathbf{P}_\Gamma$ of the test point $\mathbf{P}$.
Description:
Step 1: Starting from the adjacent point of test point $\mathbf{P}$, calculate the iterative point $\mathbf{P}_\mathtt{I}$ of the algebraic surface via Algorithm 1.
Step 2: Starting from the iteration point $\mathbf{P}_\mathtt{I}$, the new iteration point $\mathbf{P}_\mathtt{I}$ fallen on the algebraic surface $f(\mathbf{X}) = 0$ close to the orthogonal projection $\mathbf{P}_\Gamma$ is calculated using Algorithm 2.
Step 3: Compute the orthogonal projection point $\mathbf{P}_\Gamma$ via Algorithm 3.
Return $\mathbf{P}_\Gamma$;

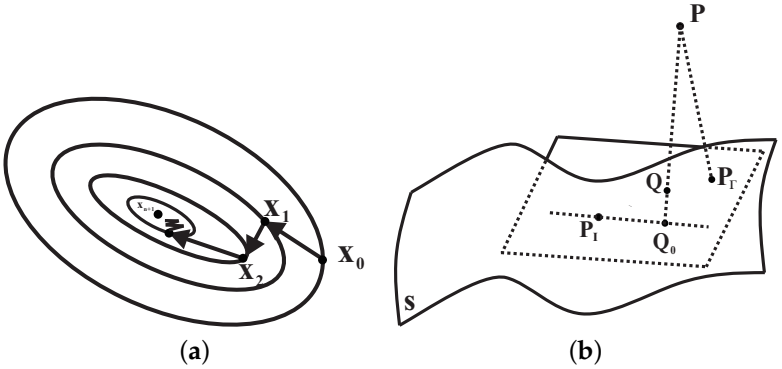

Figure 5. *Cont.*

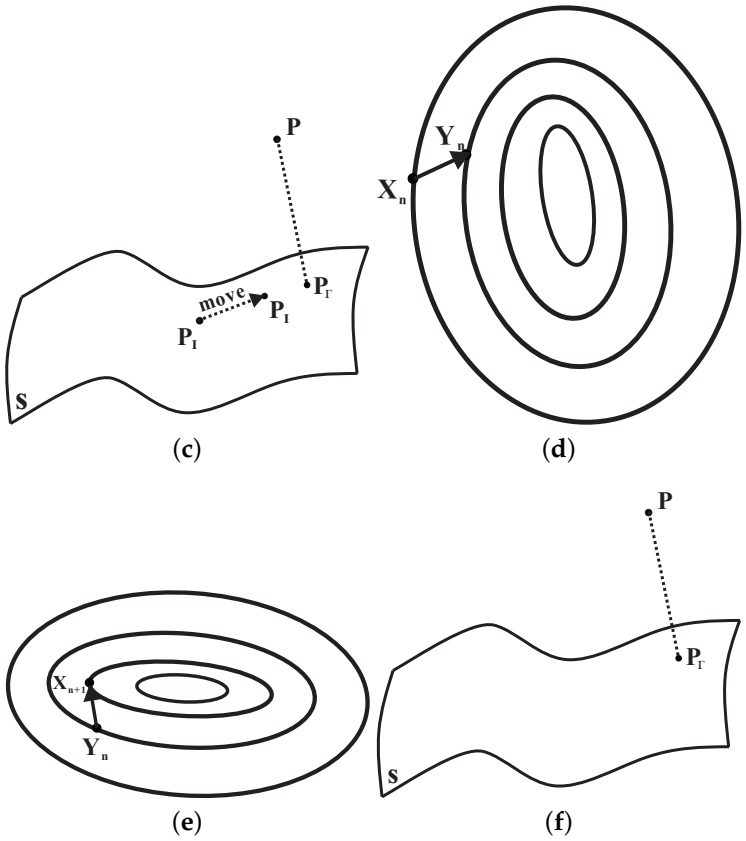

Figure 5. The complete graphic demonstration of Algorithm 4. (**a**) Newton's gradient descent method unrelated to test point; (**b**) Computing foot-point **Q** of Equation (14); (**c**) Moving the iteration point $\mathbf{P}_{\mathrm{I}}$ to a position near the orthogonal projection point $\mathbf{P}_{\Gamma}$ in **for** loop body of Algorithm 2; (**d**) Newton's gradient descent method, unrelated to test point of the first formula in Equation (15); (**e**) Newton's gradient descent method, associated with test point of the second formula in Equation (15); (**f**) Point orthogonal projection onto algebraic surface. All the curves of figures (**a**,**d**,**e**) are contour surfaces and not contour curves.

Remark 4. *In the actual programming implementation of Algorithm 4, we adopt three optimized techniques. Firstly, if the test point* $\boldsymbol{P}$ *is a long distance from its orthogonal projection on the algebraic surface, the initialization point* $\boldsymbol{X}_0$ *of Algorithm 1 is changed to a very small percentage of the test point. However, the initial iterative point* $\boldsymbol{X}_0$ *of Algorithm 1 in Step 3 of Algorithm 2 is still the foot-point* $\boldsymbol{Q}$ *computed using Step 2 of Algorithm 2. Secondly, in order to avoid degenerative situations (the denominators of the iterative Formulas (8), (9), (11), (12), and (15) are 0), we add up a very small perturbation positive number* ε *to the denominator of each iterative formula, such that Algorithm 4 and other algorithms can run and iterate normally. Thirdly, we have a wonderful discovery. In Algorithm 5, if the test point* $\boldsymbol{P}$ *is relatively close to the algebraic surface, or if the iteration point fallen on the algebraic surface of the test point* $\boldsymbol{P}$ *and the orthogonal projection point are close to each other, this indicates that the iterative point results of Algorithm 1 with Newton's local convergence condition of Algorithm 3 have been satisfied. Algorithm 2 with for loop body can be omitted; Algorithm 4 only includes Algorithms 1 and 3. Therefore, the simplified Algorithm 4 can run more efficiently. However, if the test point* $\boldsymbol{P}$ *is far away from the algebraic surface, all steps of Algorithm 4 must be fully run, such that Algorithm 4 is very robust.*

The simplified and efficient version of Algorithm 4 is represented as Algorithm 5 (see Figure 6).

Algorithm 5: The simplified version hybrid geometry strategy algorithm for point orthogonal projection onto an algebraic surface.

Input: Test point $\mathbf{P}$ and the algebraic surface $f(\mathbf{X})$.
Output: Final orthogonal projection point $\mathbf{P}_\Gamma$ of the test point $\mathbf{P}$.
Description:
Step 1: Calculate the iterative point $\mathbf{P}_\mathrm{I}$ fallen on the $f(\mathbf{X})$ via Algorithm 1.
Step 2: Compute the orthogonal projection point $\mathbf{P}_\Gamma$ via Algorithm 3.
Return $\mathbf{P}_\Gamma$;

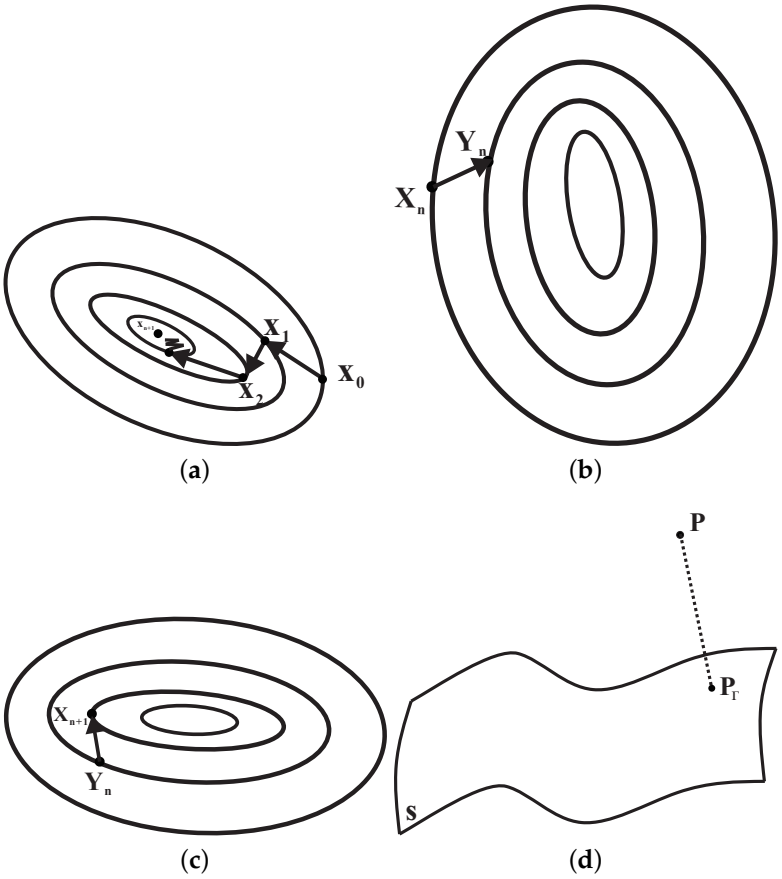

Figure 6. The complete graphic demonstration of Algorithm 5. (**a**) Algorithm 1 related to Newton's gradient descent method; (**b**) Newton's gradient descent method unrelated to test point of the first formula in Equation (15); (**c**) Newton's gradient descent method associated with test point of the second formula in Equation (15); (**d**) Point orthogonal projection onto algebraic surface. All the curves of figures (**a**–**c**) are contour surfaces and not contour curves.

2.4. Treatment of Multiple Solutions

In practical computer graphics and computer-aided geometric design and other applications, we are going to calculate not just the single orthogonal projector, but sometimes even all the orthogonal projectors. If the topological structure of the algebraic surface is simple, where its genus is zero, and if the algebraic surface is smooth, we present a simple solving method. For a given test point $\mathbf{P} = (p_1, p_2, p_3)$, we assign seven other coordinate symbols to the test point $\mathbf{P}$, respectively. In this way, the changed coordinate symbols of the test point are $(p_1, p_2, -p_3)$, $(p_1, -p_2, p_3)$, $(p_1, -p_2, -p_3)$, $(-p_1, p_2, p_3)$, $(-p_1, p_2, -p_3)$, $(-p_1, -p_2, p_3)$, and $(-p_1, -p_2, -p_3)$, respectively. For each of the eight points, we present a certain percentage reduction, such as one-hundredth of every point, etc. Of course, if the distance between the test point and the corresponding orthogonal projection point of the algebraic surface is very large, one-hundredth of the proportion can be reduced to less. The

eight points after scaling down are the initialization point of Algorithm 1, correspondingly. In this way, we want to move the initialization point of each quadrant of the 3D coordinates closer to the algebraic surface. Thus, in each quadrant of 3D coordinates, the corresponding orthogonal projection point can be obtained by using Algorithm 4 as much as possible.

If the topological structure of the algebraic surface is not simple, with its genus not being less than 1, or if the algebraic surface contains multiple branches, the simple method of solving is not fit for dealing with a complicated algebraic surface. Our preliminary idea is to outline the algebraic surface. For this reason, we try to identify a second method for computing several 3D bounding boxes within the prescribed region of the algebraic surface, where every algebraic surface patch is enclosed within one 3D bounding box. We randomly choose a point in each 3D bounding box as the initial point of Algorithm 4, and a corresponding orthogonal projection point is generated. Then, by calculating the distance between the test point $\mathbf{P}$ and each orthogonal projection point, and by finding out the minimum distance for all distances, the orthogonal projection point of the corresponding shortest distance can be found. According to the elementary knowledge of differential geometry, seeking out an orthogonal projection point is to seek out a point $\mathbf{X}$ on the algebraic surface where the cross product between the vector $\overrightarrow{\mathbf{PX}}$ and the normal vector $\mathbf{N}(\mathbf{N} = \nabla f(\mathbf{X}))$ is zero. Namely, the vector $\overrightarrow{\mathbf{PX}}$ is orthogonal to the tangent plane of the algebraic surface at the point $\mathbf{X}$, where the corresponding expression determined by the geometric property is Equation (8). Since Equation (8) is a vector equation and not a scalar equation, it is not easy to solve. Taking the inner product of the vector itself $\nabla f(\mathbf{X}) \times (\mathbf{P} - \mathbf{X})$ of Equation (8), it naturally becomes the scalar equation with Equation (9). Of course, in essence, the geometry of Equation (9) is an algebraic surface, since the algebraic surface with Equation (9) can better embody the essential geometric property of an orthogonal projection than the algebraic surface with Equation (5). Therefore, in the actual selection of algebraic surface patches, we use an algebraic surface with Equation (9) to concretely realize the search for all orthogonal projection points.

Let us assume that the region Ω of the algebraic surface with Equation (9) is $\Omega = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$. We employ the adaptive affine arithmetic method [35–37] to mark a series of 3D bounding boxes where every algebraic surface patch is enclosed in every 3D bounding box. The algebraic surface $F(\mathbf{X}) = 0$ is orthogonally projected onto the $y - z$ plane, the $x - z$ plane, and the $x - y$ plane at the point (x_0, y_0, z_0), respectively. Thus, we obtain three planar algebraic curves $F(x_0, y, z) = 0$, $F(x, y_0, z) = 0$, and $F(x, y, z0) = 0$ on three planes that are perpendicular to each other. To simplify the following expression, the planar algebraic curves $F(x_0, y, z) = 0$, $F(x, y_0, z) = 0$, and $F(x, y, z0) = 0$ can be expressed as $F_1(y, z) = 0$, $F_2(x, z) = 0$, and $F_3(x, y) = 0$, respectively. We construct an important judging function with the planar algebraic curve $F_1(y, z) = 0$,

$$\Psi_{yz} = \frac{2|F_{1yz}|}{\sqrt{(\frac{F_{1y}}{y_2})^2 + (\frac{F_{1z}}{z_3})^2}}. \tag{16}$$

Analogously, we also construct an important judging function with the planar algebraic curve $F_2(x, z) = 0$ and the planar algebraic curve $F_3(x, y) = 0$,

$$\Psi_{xz} = \frac{2|F_{2xz}|}{\sqrt{(\frac{F_{2x}}{x_1})^2 + (\frac{F_{2z}}{z_3})^2}}, \tag{17}$$

and

$$\Psi_{xy} = \frac{2|F_{3xy}|}{\sqrt{(\frac{F_{3x}}{x_1})^2 + (\frac{F_{3y}}{y_2})^2}}, \tag{18}$$

where $F_{1y} = \frac{\partial F_1(y,z)}{\partial y}$, $F_{1z} = \frac{\partial F_1(y,z)}{\partial z}$, $F_{2x} = \frac{\partial F_2(x,z)}{\partial x}$, $F_{2z} = \frac{\partial F_2(x,z)}{\partial z}$, $F_{3x} = \frac{\partial F_3(x,y)}{\partial x}$, $F_{3y} = \frac{\partial F_3(x,y)}{\partial y}$, $F_{1yz} = \frac{\partial^2 F_1(y,z)}{\partial y \partial z}$, $F_{2xz} = \frac{\partial^2 F_2(x,z)}{\partial x \partial z}$, $F_{3xy} = \frac{\partial^2 F_3(x,y)}{\partial x \partial y}$, $x_0 = \frac{a_1+a_2}{2}$, $x_1 = \frac{a_2-a_1}{2}$, $y_0 = \frac{b_1+b_2}{2}$, $y_2 = \frac{b_2-b_1}{2}$, $z_0 = \frac{c_1+c_2}{2}$, and $z_3 = \frac{c_2-c_1}{2}$. The unknown variable $\mathbf{X}$ in each of the nine partial derivative functions F_{1y}, F_{1z}, F_{2x}, F_{2z}, F_{3x}, F_{3y}, F_{1yz}, F_{2xz}, F_{3xy} is replaced by a point value (x_0, y_0, z_0), and afterwards, this point is named (x_0, y_0, z_0) as the center point of the 3D bounding box. By combining three Formulas (16)–(18), we obtain the crucial judging function Ψ with Equation (19),

$$\Psi = Max\{\Psi_{xy}, \Psi_{yz}, \Psi_{xz}\}. \tag{19}$$

We assign a critical value for the crucial judging function Ψ with Equation (19). The adaptive approach method [38,39] of solving a series of 2D bounding boxes of the planar algebraic curve is adopted to solve a series of 3D bounding boxes of the algebraic surface. The exact interpretation of Equation (19) is completely the same as the interpretation in [38]. On an affine arithmetic, we mainly assimilate the idea of the work in [38]. However, we have to assimilate the idea of the paper [39] to investigate a series of 3D bounding boxes with the more complicated topological structure of the algebraic surface. The detailed description for solving a series of 3D bounding boxes of the algebraic surface can be described as Algorithm 6.

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

Input: The algebraic surface $F(\mathbf{X}) = 0$ and the initial 3D bounding box including or intersecting with the algebraic surface $F(\mathbf{X}) = 0$.
Output: A number of 3D bounding boxes satisfied with certain conditions.
Description:
Step 1: Subdivide this 3D bounding box into 8 3D sub-bounding boxes by dividing by 2 on each axis.
Step 2: Compute the critical value Ψ of each 3D sub-bounding box through Equation (19).
Step 3: if (Ψ <= the critical value and recursion times< 10){
Execute Algorithm 6 with the 3D sub-bounding box.
}
if (Ψ <= critical value and recursion times == 10){
Store all 3D bounding sub-boxes in one set.
}
End Algorithm.

In short, three important techniques and schemes are adopted in the process of realizing a point orthogonal projection onto an algebraic surface. In the first step, the Newton gradient descent method of Algorithm 1 is used to ensure that the initial iteration can iterate and fall on the algebraic surface. In the second step, Algorithm 2 is used to gradually move the iterative point fallen on the algebraic surface to the orthogonal projection point at a very close position, such that the local convergence condition of the last step is satisfied. In the third step, Algorithm 3 is adopted to accelerate the iteration of the iteration point to the algebraic surface and orthogonalization by using the Newton gradient descent method and the second-order Newton iteration method under the condition of local convergence condition. Thus, Algorithm 4 is guaranteed to be robust and efficient. In the latter part of Section 2, a simplified state and a multi-solution state are also discussed and analyzed.

3. Convergence Analysis

Lemma 1. *Suppose that $F \in C^2(D)$, where this function F is Equation (9). Assume that the function $F : D \subseteq R \to R$ has a simple root $\boldsymbol{X}^* \in D$, where D is an open interval. If the initial*

iterative point $\boldsymbol{X}_0$ is sufficiently close to $\boldsymbol{X}^*$, then the method defined by the second step of Equation (15) has second-order convergence.

Proof. This lemma is completely similar to the fundamental local convergence theorem of Newton's iterative method. We are not going to prove it. □

Theorem 1. *Algorithm 4 is able to converge successfully, and the order of convergence of Algorithm 4 is no more than 2.*

Proof. Part One: Algorithm 4 is able to converge successfully.

Algorithm 4 mainly contains three important components: Algorithm 1 (Newton's gradient descent method), Algorithm 2 (computing the foot-point **Q** and moving the iterative point $\mathbf{P}_{\mathrm{I}}$ to the close position of the orthogonal projection point $\mathbf{P}_{\Gamma}$), and Algorithm 3 (the hybrid geometric accelerated orthogonal method).

From Remark 1, the function of Equation (11) is to cause the initial point $\mathbf{X}_0$ to be on the algebraic surface as much as possible, according to the Newton's gradient descent property. Consequently, the initial point $\mathbf{X}_0$ can be realized to be fallen on the algebraic surface.

From Remark 2, the essential geometric feature of Algorithm 2 is to make the iteration point $\mathbf{P}_{\mathrm{I}}$ of the first fallen on the algebraic surface realized by Algorithm 1 move five times, and to let the iteration point $\mathbf{P}_{\mathrm{I}}$ be gradually moved to the position that is particularly close to the orthogonal projection point $\mathbf{P}_{\Gamma}$. Thus, the final iteration point $\mathbf{P}_{\mathrm{I}}$ of Algorithm 2 satisfies the Newtonian's local convergence condition of Algorithm 3.

From Remark 3, the geometric essence of Algorithm 3 is a Newton-type iteration. The final iteration point of Algorithm 2 as the initial iteration point of Algorithm 3 can satisfy the local convergence condition of Algorithm 3, or the initial iteration point of Algorithm 3 satisfies the convergence condition of iteration Formula (15); Algorithm 3 can converge quickly and successfully.

In short, from the above analysis, we can show that Algorithm 4 can be convergent.

Part two: The order of convergence of Algorithm 4 is no more than 2.

In this part, a numerical analysis method is used to prove the order of convergence of Algorithm 4. Algorithm 4 mainly includes three sub-algorithms: Algorithm 1 that adopts Formula (11), Algorithm 2 that adopts Formula (14), and Algorithm 3 that adopts Formula (15).

Firstly, the order of convergence of the iterative formula (11) is proven to be 2. Without a loss of generality, it is assumed that the algebraic surface $f(\mathbf{X}) = 0$ can be expressed in parameterized form, where the parameter $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T$ is the corresponding parameter of the orthogonal projection point $\mathbf{P}_{\Gamma}$ of the test point **P** on the algebraic surface after parameterization. It is not difficult to tell that the corresponding parameterized Newton's iteration of iteration (11) can be expressed as

$$\mathbf{U}_{n+1} = \mathbf{U}_n - G_1(\mathbf{U}_n)F_1(\mathbf{U}_n), \tag{20}$$

where $G_1(\mathbf{U}) = F_1'^{-1}(\mathbf{U})$ is inverse matrix of the Jacobian matrix of the equation $F_1(\mathbf{U}) = 0$. Taylor's expansion is performed near the parameter root $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T$ of the equation $F_1(\mathbf{U}) = 0$, then we have

$$F_1(\mathbf{U}_n) = C_0 + C_1\mathbf{e}_n + C_2\mathbf{e}_n^2 + o\left(\mathbf{e}_n^3\right), \tag{21}$$

where $\mathbf{e}_n = \mathbf{U}_n - \boldsymbol{\alpha}, C_i = (1/i!)\left(F_1^{(j)}(\boldsymbol{\alpha})\right), i = 0, 1, 2$. From Equation (21), we can easily obtain the following two formulas,

$$F_1'(\mathbf{U}_n) = C_1 + 2C_2\mathbf{e}_n + o\left(\mathbf{e}_n^2\right) \tag{22}$$

and

$$F_1''(\mathbf{U}_n) = 2C_2 + o(\mathbf{e}_n) \tag{23}$$

Thus, according to Equations (21)–(23), it can be concluded that the iterative error of Equation (20) is

$$\mathbf{e}_{n+1} = C_1^{-1}C_2\mathbf{e}_n^2 + o\left(\mathbf{e}_n^3\right) \tag{24}$$

Equation (24) shows that iterative Equation (11) converges to order 2.

Secondly, it is deduced that the order of foot-point $\mathbf{Q}$ in Equation (14) is no more than 2. Since $\mathbf{Q}_0$ is the vertical foot derived from the tangent plane, it is clear that the order of foot-point $\mathbf{Q}$ in Equation (14) is 1. Therefore, the order of convergence of the foot-point $\mathbf{Q}$ is 1.

Thirdly, from Lemma 1, the order of convergence of the second equation of Equation (15) is 2, and the order of convergence of the first equation of Equation (15) is also 2. Then, the order of convergence of Equation (15) is 2.

Based on the above three parts, it can be seen that the order of convergence of Algorithm 4 is no more than 2. □

4. Experimental Results

Example 1. *Suppose that an algebraic surface* $f(x,y,z) = \frac{x^2}{4} + xy + \frac{y^2}{13} + z - 1 = 0$ *(see Figure 7), in the region* $[-200, 200] \times [-200, 200] \times [-200, 200]$*. All the computations were performed using the mathematics and engineering computing software Maple 18, with* $\varepsilon = 10^{-20}$ *via Algorithm 4. In Table 1, the four symbols* $\boldsymbol{P}$, $\boldsymbol{P}_\Gamma$, $|f(\boldsymbol{P}_\Gamma)|$, *and* $F(\boldsymbol{P}_\Gamma)$ *are the test point, the orthogonal projection point of the test point, the deviation degree of the orthogonal projection point on the algebraic surface, and the expression* $\langle \nabla f(\boldsymbol{P}_\Gamma) \times (\boldsymbol{P} - \boldsymbol{P}_\Gamma), \nabla f(\boldsymbol{P}_\Gamma) \times (\boldsymbol{P} - \boldsymbol{P}_\Gamma) \rangle$ *for the second formula of Equation (10), respectively. In the specified region, we randomly select a mass of 3D points as test points. The probability of non-convergence using Algorithm 4 with these test points is extremely low, which is detected to have very high robustness and efficiency. Furthermore, in each quadrant of eight quadrants, we arbitrarily choose two different test points. The corresponding orthogonal projection point for each test point is computed using Algorithm 4. The concrete computed values are displayed in Table 1, where the digital values of the orthogonal projective point we present are abbreviated. For example, if test point* $\boldsymbol{P}$ *is (320,490,730), the actual values* $\boldsymbol{P}_\Gamma$, $|f(\boldsymbol{P}_\Gamma)|$ *and* $F(\boldsymbol{P}_\Gamma)$ *are (0.65483015050952098661, 0.11053569270327875241, 0.81947740733240397739), 1.0* $\times$ 10^{-20}, *and 3.0900825879965*$\times$ 10^{-21}, *respectively. In Table 1, the corresponding values of the other test points are completely the same. Limited by Table 1, we only present four digits after the decimal point.*

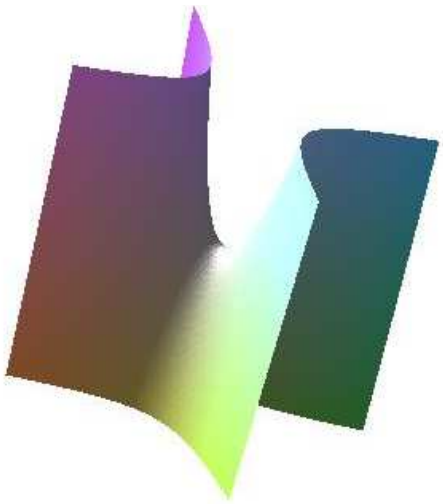

Figure 7. Graphic demonstration for Example 1.

Table 1. The obtained running results of Algorithm 4 through Example 1.

$\mathbf{P}$	(320, 490, 730)	(376, 949, 738)	(276, 806, −528)
$\mathbf{P}_\Gamma$	(0.6548, 0.1105, 0.8194)	(1.3100, −0.1467, 0.7615)	(−1.5632, 0.2567, 0.7853)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	1.0×10^{-20}	5.0×10^{-20}	2.0×10^{-20}
$F(\mathbf{P}_\Gamma)$	3.1×10^{-21}	1.3×10^{-21}	3.3×10^{-22}
$\mathbf{P}$	(476, −606, 238)	(882, −206, 215)	(582, −406, −715)
$\mathbf{P}_\Gamma$	(−3.2380, 3.7149, 9.3464)	(−1.8122, 5.1651, 7.4873)	(0.74713, −1.1846, 1.6375)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	2.0×10^{-19}	2.0×10^{-20}	0.0
$F(\mathbf{P}_\Gamma)$	6.5×10^{-12}	9.0×10^{-21}	4.9×10^{-22}
$\mathbf{P}$	(−622, 316, 213)	(−307, 416, 213)	(−207, 273, −376)
$\mathbf{P}_\Gamma$	(2.1944, −4.1355, 7.5558)	(2.4351, −2.7091, 5.5502)	(−0.8714, 0.9816, 1.5914)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	1.0×10^{-19}	5.9×10^{-18}	4.0×10^{-20}
$F(\mathbf{P}_\Gamma)$	2.5×10^{-14}	1.5×10^{-13}	5.6×10^{-22}
$\mathbf{P}$	(−359, −233, 246)	(−259, −333, 246)	(−249, −233, −556)
$\mathbf{P}_\Gamma$	(−0.7784, −1.0665, −0.0692)	(−1.2903, −0.4026, 0.0517)	(0.3791, 0.2582, 0.8610)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	1.0×10^{-20}	3.0×10^{-20}	1.0×10^{-20}
$F(\mathbf{P}_\Gamma)$	4.3×10^{-17}	3.1×10^{-22}	1.0×10^{-18}
$\mathbf{P}$	(476, 616, −5238)	(522, −606, −215)	(−237, 393, −256)
$\mathbf{P}_\Gamma$	(−0.1122, −0.0347, 0.9928)	(3.2821, −3.9458, 10.060)	(−1.7864, 1.8008, 3.1697)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	2.1×10^{-18}	3.8×10^{-19}	2.0×10^{-19}
$F(\mathbf{P}_\Gamma)$	8.9×10^{-22}	9.8×10^{-14}	1.9×10^{-14}
$\mathbf{P}$	(−249, −633, −256)		
$\mathbf{P}_\Gamma$	(2.5127, −0.2743, 0.1050)		
$\lvert f(\mathbf{P}_\Gamma)\rvert$	0.0		
$F(\mathbf{P}_\Gamma)$	6.7×10^{-17}		

Example 2. *Suppose an algebraic surface* $f(x,y,z) = x^3 + y^3 + z^3 + 1 - (x+y+z+1)^4 = 0$ *(see Figure 8), in the region* $[-200, 200] \times [-200, 200] \times [-200, 200]$. *All computations were performed using the Maple 18 environment with* $\varepsilon = 10^{-20}$ *via Algorithm 4. In Table 2, the four symbols* $\boldsymbol{P}$, $\boldsymbol{P}_\Gamma$, $|f(\boldsymbol{P}_\Gamma)|$, *and* $F(\boldsymbol{P}_\Gamma)$ *are the test point, the orthogonal projection point of the test point, the deviation degree of the orthogonal projection point on the algebraic surface, and the expression* $\langle \nabla f(\boldsymbol{P}_\Gamma) \times (\boldsymbol{P} - \boldsymbol{P}_\Gamma), \nabla f(\boldsymbol{P}_\Gamma) \times (\boldsymbol{P} - \boldsymbol{P}_\Gamma) \rangle$ *for the second formula of Equation (10), respectively. In the specified region, we randomly select a mass of 3D points as test points; the probability of non-convergence using Algorithm 4 with these test points is extremely low, and it is detected with very high robustness and efficiency. Furthermore, in each quadrant of eight quadrants, we arbitrarily choose two different test points. The corresponding orthogonal projection point for each test point is computed using Algorithm 4. The concrete computed values are displayed in Table 2, where the digital values of the orthogonal projective point we present are abbreviated. For example, if test point* $\boldsymbol{P}$ *is (320,490,530), the actual values* $\boldsymbol{P}_\Gamma$, $|f(\boldsymbol{P}_\Gamma)|$ *and* $F(\boldsymbol{P}_\Gamma)$ *are (0.89782830182962418807,* -0.47457258895408409955, -0.30037859976898496332), $9.418433921 \times 10^{-10}$, *and* $1.6452196947017076704 \times 10^{-12}$, *respectively. In Table 2, the corresponding values of the other test points are completely the same. Limited by Table 2, we only present 10 digits after the decimal point.*

Example 3. *Now, we consider a self-intersecting quasi-algebraic surface* $f(x,y,z) = x^2 - y^2 + z^3 = 2$, *where* $y = exp(-xz)$ *(see Figure 9). Since the surface is not a complete algebraic surface in the true sense, some parts of the surface are singular regions, which may not converge for Algorithm 4, but for non-singular regions, Algorithm 4 can still converge. In the specified region, we arbitrarily choose two different 3D test points. The implementation requirements and environment are exactly the same as those of Examples 1 and 2. The corresponding orthogonal projection point for each test point was computed using Algorithm 4. The concrete computed values are displayed in Table 3, and the digital values of the orthogonal projective point we present are abbreviated. From Table 3, the probability of convergence using Algorithm 4 with these test points is high, which is detected to be a sign of robustness and efficiency.*

Table 2. The obtained running results of Algorithm 4 through Example 2.

$\mathbf{P}$	(320,490,530)	(326,449,278)
$\mathbf{P}_\Gamma$	(0.8978283018, −0.4745725889, −0.3003785997)	(0.5805473488, 0.0686066996, −0.6802898709)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	9.4×10^{-10}	6.1×10^{-10}
$F(\mathbf{P}_\Gamma)$	1.6×10^{-12}	5.4×10^{-13}
$\mathbf{P}$	(276, 306, −328)	(376, 316, −238)
$\mathbf{P}_\Gamma$	(0.3499239738, 0.2912835991, −0.9361716812)	(0.2734464553, 0.3875179615, −0.9183591981)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	2.6×10^{-10}	3.7×10^{-10}
$F(\mathbf{P}_\Gamma)$	1.8×10^{-11}	4.0×10^{-11}
$\mathbf{P}$	(276, −309, 238)	(382, −206,215)
$\mathbf{P}_\Gamma$	(0.2785766037, −0.9391397370, 0.3606532988)	(0.1720828409, −0.9348105519, 0.5199384062)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	5.2×10^{-10}	6.7×10^{-10}
$F(\mathbf{P}_\Gamma)$	2.6×10^{-11}	2.0×10^{-11}
$\mathbf{P}$	(382, −287, −409)	(422, −306, −217)
$\mathbf{P}_\Gamma$	(0.2468466777, −0.3925225425, −0.4136091918)	(0.23453300365, −0.4031775077, −0.3865167494)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	8.4×10^{-10}	8.5×10^{-10}
$F(\mathbf{P}_\Gamma)$	1.3×10^{-11}	1.1×10^{-11}
$\mathbf{P}$	(−422, 316, 213)	(−307, 216, 293)
$\mathbf{P}_\Gamma$	(−0.2590172850, −0.1971877899, −0.2069292587)	(−0.2565723559, −0.2062237616, −0.1977314391)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	9.5×10^{-10}	7.8×10^{-10}
$F(\mathbf{P}_\Gamma)$	5.8×10^{-11}	8.8×10^{-11}
$\mathbf{P}$	(−207, 273, −376)	(−237, 393, −256)
$\mathbf{P}_\Gamma$	(−0.3472321429, 0.1281172458, −0.3965695863)	(−0.3615862188, 0.1322536132, −0.36628133060)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	8.7×10^{-10}	7.7×10^{-10}
$F(\mathbf{P}_\Gamma)$	4.0×10^{-12}	2.3×10^{-12}
$\mathbf{P}$	(−359, −233, 246)	(−259,−333,246)
$\mathbf{P}_\Gamma$	(−0.3904801634, −0.3514720192, 0.1165325261)	(−0.3594935775, −0.3823713615, 0.1161247111)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	8.3×10^{-10}	7.2×10^{-10}
$F(\mathbf{P}_\Gamma)$	2.4×10^{-12}	4.7×10^{-11}
$\mathbf{P}$	(−249, −233, −556)	(−249, −633, −256)
$\mathbf{P}_\Gamma$	(0.6777775475, −0.6538305244, −1.0105058748)	(0.62667855421, −0.9971463160, −0.6338419899)
$\lvert f(\mathbf{P}_\Gamma)\rvert$	2.8×10^{-10}	2.0×10^{-10}
$F(\mathbf{P}_\Gamma)$	3.1×10^{-11}	2.2×10^{-11}

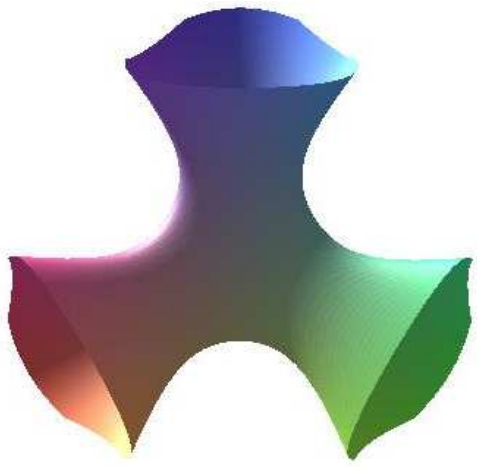

Figure 8. Graphic demonstration for Example 2.

Remark 5. *In this remark, outside the regions* $[-200, 200] \times [-200, 200] \times [-200, 200]$ *specified in Examples 1 and 2, we randomly selected two different test points in each quadrant that were far from the algebraic surface. Through Algorithm 4, each test point can be orthogonally projected onto the corresponding orthogonal projection point. The existing algorithms cannot converge to the corresponding orthogonal projection points because the test points are far from the algebraic surface, or the initialization points are not properly selected, etc. Table 4 shows whether the various*

algorithms converge and the reasons for their convergence. Once again, it shows that Algorithm 4 can converge to the corresponding orthogonal projection point quickly, accurately, and efficiently.

Table 3. The obtained running results of Algorithm 4 through Example 3.

$\mathbf{P}$	$(6, 7, 8)$	$(8, 6, 5)$
$\mathbf{P}_\Gamma$	$(0.9064333791, 6.9928001253, 1.1419222319)$	$(1.2015703426, 5.9813549954, 0.8314404941)$
$\lvert f(\mathbf{P}_\Gamma)\rvert$	5.5×10^{-7}	5.3×10^{-6}
$F(\mathbf{P}_\Gamma)$	4.6×10^{-5}	6.0×10^{-4}
$\mathbf{P}$	$(6, -6, 6)$	$(7, -6, 8)$
$\mathbf{P}_\Gamma$	$(1.0295698711, -6.0000061990, 1.0295698711)$	$(0.9723667468, -6.0029643353, 1.0843126066)$
$\lvert f(\mathbf{P}_\Gamma)\rvert$	3.0×10^{-10}	1.6×10^{-5}
$F(\mathbf{P}_\Gamma)$	1.0×10^{-9}	9.1×10^{-6}
$\mathbf{P}$	$(-4, 4, -5)$	$(-4, 5, -6)$
$\mathbf{P}_\Gamma$	$(-0.9304838087, 3.9918808576, -1.1213468254)$	$(-0.8542470910, 4.9870922392, -1.1851067449)$
$\lvert f(\mathbf{P}_\Gamma)\rvert$	8.6×10^{-5}	2.1×10^{-5}
$F(\mathbf{P}_\Gamma)$	9.4×10^{-5}	5.9×10^{-5}
$\mathbf{P}$	$(-7, -6, -8)$	$(-6, -8, -9)$
$\mathbf{P}_\Gamma$	$(-0.9724553347, -6.0060342760, -1.0842198022023942600)$	$(-0.8575669009, -8.0072947909, -1.1814724407)$
$\lvert f(\mathbf{P}_\Gamma)\rvert$	1.9×10^{-5}	5.1×10^{-5}
$F(\mathbf{P}_\Gamma)$	2.9×10^{-5}	4.5×10^{-5}

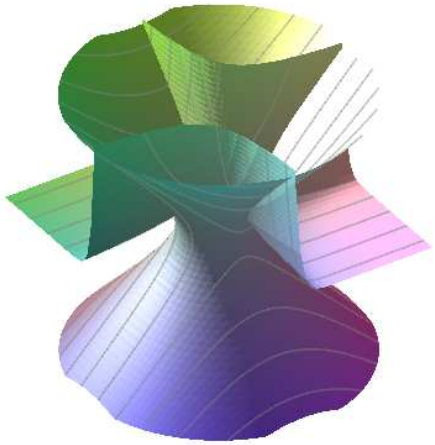

Figure 9. Graphic demonstration of Example 3.

Table 4. Comparison of convergence results of various algorithms and their reason analysis with Examples 1 and 2.

Algorithms	Convergence or Not	Reasons
Our Algorithm 4	Being convergent	Being independent of the initial point.
Newton's method	Being divergent	Being dependent on the initial point.
Algorithm [1]	Being divergent	Being dependent on the initial point.
Homotopy method [12,13]	Being divergent	Difficulty in finding an objective solving system of nonlinear equations of homotopy method.
Resultant method [16–19]	Being divergent	Difficulty in directly solving all solutions of nonlinear polynomial system with the resultant method, due to the degree of the algebraic surface being more than 4.
Bézier clipping technique [20–22]	Being divergent	Equation (9) being difficult or even impossible to transform into Bernstein–Bézier form for a fraction of the algebraic surfaces.
Algorithm [27]	Being divergent	The vertical foot-point being regarded as the next iteration point resulting from the test point being far from the algebraic surface.

5. Conclusions and Future Work

In this paper, we discuss and analyze a topic associated with point orthogonal projection onto the algebraic surface. The presented key and core algorithm is involved in constructing an orthogonal polynomial and using the Newton iterative method for iteration. In order to ensure maximum robustness, two techniques were adopted before the Newton iteration: (1) Newton's gradient descent method, which is used to make the initial iteration point fall onto the algebraic surface; and (2) computing the foot-point and moving the iterative point to the close position of the orthogonal projection point. Theoretical analysis and experiences show that the proposed algorithm could accurately and efficiently converge to the orthogonal projection point for test points in different spatial positions.

In the future, we will try to study and explore some more efficient and robust algorithms for calculating the minimum distance between a point and an algebraic surface, or the shortest distance between two algebraic surfaces. For any unrestricted initial iteration point and test point for any position in three-dimensional space, or for an irregular algebraic surface, future work will involve constructing a brand new algorithm to satisfy the following conditions: the convergence of a new algorithm should be robust and efficient, and the convergent orthogonal projection point should simultaneously fit for three relationships of Equation (9). It will undoubtedly be a huge future challenge to develop and explore such satisfactory algorithms.

Author Contributions: The contributions of all the authors are the same. All of the authors teamed up to develop the current draft. X.W. is responsible for investigating, providing resources and methodology, the original draft, writing, reviewing, validation, and editing and supervision of this work. Y.L. is responsible for software, algorithm, program implementation, and visualization. X.W. is responsible for writing, reviewing, and editing and supervision of this work. X.L. is responsible for algorithm, program implementation, and 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, Shandong Youth University of Political Science Doctor Starting Project No. XXPY20050(700212), and the Natural Science Research Project of Guizhou Minzu University No. GZMUZK[2021]YB20.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We take the opportunity to thank the anonymous reviewers for their thoughtful and meaningful comments. Many thanks to the editors for their great help.

Conflicts of Interest: The authors declare no conflicts 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.; Wang, L.; Wu, Z.; Hou, L.K.; Liang, J.; Li, Q. 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 Geom. 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. Bartoň, M. Solving polynomial systems using no-root elimination blending schemes. *Comput.-Aided Des.* **2011**, *43*, 1870–1878. [CrossRef]
24. 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]
25. 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]
26. 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. [CrossRef]
27. Hartmann, E. On the curvature of curves and surfaces defined by normal forms. *Comput. Aided Geom. Des.* **1999**, *16*, 355–376. [CrossRef]
28. 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.
29. Martin, A.; 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.
30. 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]
31. Cheng, T.; Wu, Z.; Li, X.; Wang, C. Point Orthogonal Projection onto a Spatial Algebraic Curve. *Mathematics* **2020**, *8*, 317. [CrossRef]
32. Cesarano, C. Generalized Chebyshev polynomials. *Hacet. J. Math. Stat.* **2014**, *43*, 731–740. [CrossRef]
33. Cesarano, C.; Cennamo, G.; Placidi, L. Humbert Polynomials and Functions in Terms of Hermite Polynomials towards Applications to Wave Propagation. *Wseas Trans. Math.* **2014**, *13*, 595–602. [CrossRef]
34. Dattoli, G.; Ricci, P.E.; Cesaranoc, C. The Lagrange Polynomials, the Associated Generalizations and the Umbral Calculus. *Integral Transform. Spec. Funct.* **2003**, *14*, 181–186.
35. Lopes, H.; Oliveira, J.B.; de Figueiredo, L.H. Robust adaptive polygonal approximation of implicit curves. *Comput. Graph.* **2002**, *26*, 841–852.
36. 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: 25th SIBGRAPI Conference on Graphics, Patterns and Images, Ouro Preto, Brazil, 22–25 August 2011; IEEE Press: Piscataway, NJ, USA, 2012; pp. 94–101. [CrossRef]
37. 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]
38. de Figueiredo, L.H.; Stolfi, J. Affine arithmetic: Concepts and applications. *Numer. Algorithms* **2004**, *37*, 147–158.
39. Paiva, A.; Lopes, H.; Lewiner, T.; de Figueiredo, L.H. Robust adaptive meshes for implicit surfaces. In Proceedings of the SIBGRAPI 2006: XIX Brazilian Symposium on Computer Graphics and Image Processing, Manaus, AM, Brazil, 8–11 October 2006; IEEE Press: Piscataway, NJ, USA, 2006; pp. 205–212. [CrossRef]