algorithms

MDPI

Article

Algebraic Point Projection for Immersed Boundary Analysis on Low Degree NURBS Curves and Surfaces

Huanyu Liao, Pavan Kumar Vaitheeswaran, Tao Song and Ganesh Subbarayan *

School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA; liao85@purdue.edu (H.L.); pvaithee@purdue.edu (P.K.V.); songtao.thu@gmail.com (T.S.)

* Correspondence: ganeshs@purdue.edu

Received: 30 December 2019; Accepted: 26 March 2020; Published: 31 March 2020

Abstract: Point projection is an important geometric need when boundaries described by parametric curves and surfaces are immersed in domains. In problems where an immersed parametric boundary evolves with time as in solidification or fracture analysis, the projection from a point in the domain to the boundary is necessary to determine the interaction of the moving boundary with the underlying domain approximation. Furthermore, during analysis, since the driving force behind interface evolution depends on locally computed curvatures and normals, it is ideal if the parametric entity is not approximated as piecewise-linear. To address this challenge, we present in this paper an algebraic procedure to project a point on to Non-uniform rational B-spline (NURBS) curves and surfaces. The developed technique utilizes the resultant theory to construct implicit forms of parametric Bézier patches, level sets of which are termed algebraic level sets (ALS). Boolean compositions of the algebraic level sets are carried out using the theory of R-functions. The algebraic level sets and their gradients at a given point on the domain are then used to project the point onto the immersed boundary. Beginning with a first-order algorithm, sequentially refined procedures culminating in a second-order projection algorithm are described for NURBS curves and surfaces. Examples are presented to illustrate the efficiency and robustness of the developed method. More importantly, the method is shown to be robust and able to generate valid solutions even for curves and surfaces with high local curvature or $\mathscr{G}_0$ continuity—problems where the Newton–Raphson method fails due to discontinuity in the projected points or because the numerical iterations fail to converge to a solution, respectively.

Keywords: NURBS; implicit representation; resultant; algebraic level sets; point projection and inversion

1. Introduction

Given a test point and a parametric entity (curve or surface), the generalized point projection problem is to find the closest point (footpoint) on the entity as well as the corresponding parameter value. Since the footpoint is the closest point on the curve or surface, the line connecting the test point to the footpoint is normal to the curve or the surface [1]:

$$g(u) = \mathbf{C}'(u) \cdot (\mathbf{C}(u) - \mathbf{P}) = 0 \tag{1}$$

Given a parametric curve or surface entity $\mathbf{C}(u) \in \mathbb{R}^n$ (u is treated as a vector when the entity is a surface), the Euclidean distance function $d_E(\mathbf{x})$ is defined as the shortest distance from physical test point $\mathbf{x}$ to $\mathbf{C}(u)$ given by:

$$d_E(\mathbf{x}) = \inf_u \|\mathbf{x} - \mathbf{C}(u)\| \tag{2}$$

where $\mathbf{C}(u)$ is a physical point on the curve or surface of interest. The distance function $d_E(\mathbf{x})$ is continuous for all $\mathbf{x} \in \mathbb{R}^n$ and differentiable almost everywhere. The "footpoint" of projection in parametric space u_f is defined as:

$$u_f(\mathbf{x}) = \underset{u \in [a,b]}{\arg\min} \|\mathbf{x} - \mathbf{C}(u)\| \tag{3}$$

where $[a, b]$ is the parameter range. In general, $u_f(\mathbf{x})$ may be non-unique, discontinuous, or non-existent, as illustrated in Figure 1. The footpoint of a test point near the curve or surface segment with high local curvature can be non-unique, leading to discontinuity of point projection process as illustrated in Figure 1a. The non-existence is illustrated in Figure 1b, and occurs around points where $\mathbf{C}(u)$ has only $\mathscr{G}^0$ continuity.

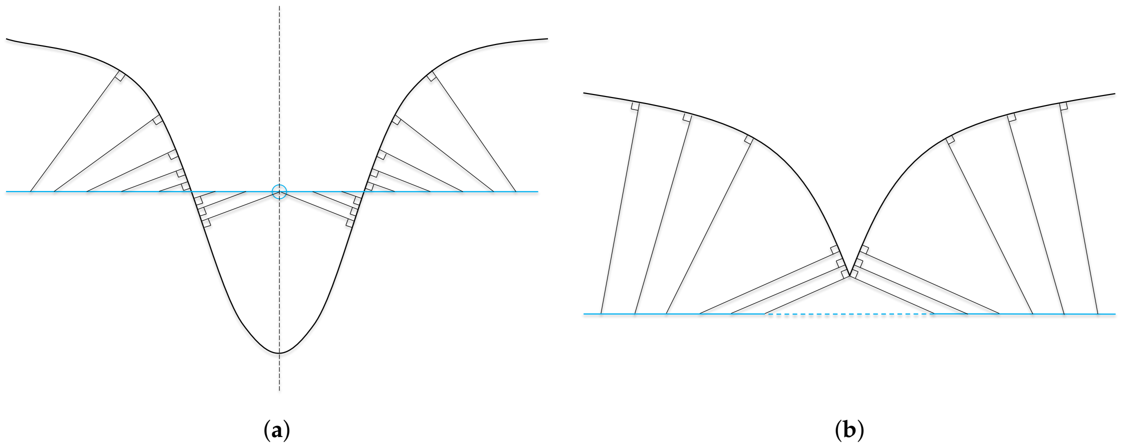

(**a**) (**b**)

Figure 1. Special cases encountered during point projection: (**a**) Exact projection from points on a straight line to a bell-shaped curve. Discontinuity occurs at the circled central point due to non-uniqueness of point projection. (**b**) Projection from points on a straight line to a cone-shaped curve. The dashed line segment has no footpoint on the curve.

This problem is of importance in geometric modeling. For instance, while fitting a curve or surface to sampled data, one may need to compute corresponding parameter values and errors at data points since the error is the distance between the data point and the fitting curve or surface [2].

Point projection also plays an important role in computer aided engineering (CAE), especially when boundaries are immersed into the domain and evolved. Such immersed boundary analysis [3] uses a non-conforming mesh to significantly reduce computational cost required for mesh generation as the boundaries evolve. Often, the immersed boundaries are represented as parametric splines, and, more recently, in isogeometric analysis, the underlying domain is also approximated by parametric splines. Isogeometric analysis (IGA) [4,5] is aimed at building behavioral approximations isoparametrically on a spline geometry, most commonly on NURBS curves and surfaces. The goal is to eliminate the need to mesh the geometry for analysis and to ensure the exactness of geometry to the CAD model during analysis. Tambat and Subbarayan [6] developed an Enriched Isogeometric Analysis (EIGA) for immersed boundary problems in which both the domain as well as the enrichments are described by NURBS entities, which are then blended to describe the enriched approximation.

In any immersed boundary problem solution, capturing the interaction of the field approximation defined on the immersed (explicitly defined) boundary with the approximations on the enriched domain requires one to determine the nearest point on the boundary from any given point in the underlying domain. This projection from the spatial point to the boundary is necessary to compute the influence of the domain approximation on those approximations defined on the boundary (see Figure 2). For example, in solutions to mechanical contact problems [7,8], point projection is needed to define the normal gap and tangential slip between two bodies. In fluid–structure interaction (FSI) problems, point projection is required to transfer kinematic and traction data between non-matching fluid–structure interface [9]. In the enriched isogeometric analysis mentioned above, point projection is used to enrich the base approximations with those on lower-dimensional geometrical features such as crack surfaces and phase boundaries, enabling simulations of fracture propagation [6,10] and solidification [11]. A fast and robust point projection method is critical to efficiently solving these problems.

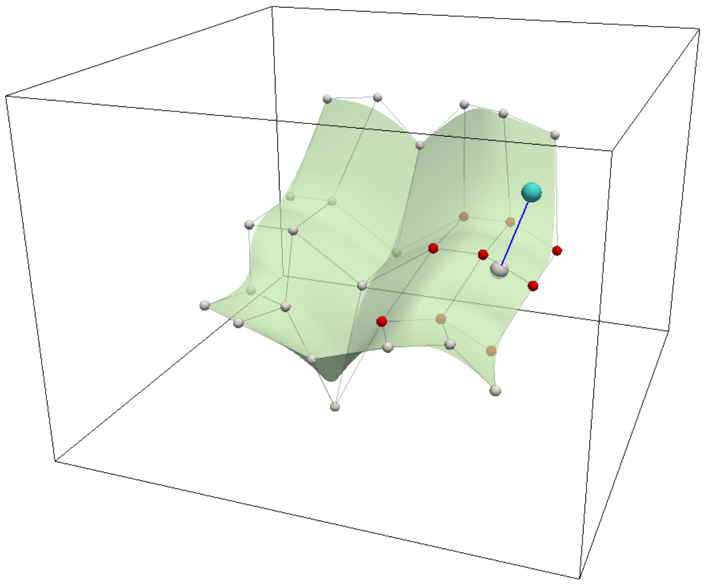


Figure 2. Behavioral analysis in the presence of complex free form embedded surface. The spatial point may only influence a local region of the surface with the highlighted control points, which can be identified by point projection.

The rest of the paper is organized as follows. In Section 2, a review of the literature pertaining to the point projection problem is carried out. In Section 3, the algebraic estimation of distance from a low-degree NURBS curve or surface is reviewed. In Section 4.1, a detailed algorithm for two-dimensional algebraic point projection is presented followed in Section 4.2 by its extension to three-dimensional NURBS surfaces. Several examples are provided in Section 5 to validate the developed algorithm. The paper is concluded with remarks in Section 6.

2. Literature Review

The use of Newton–Raphson (NR) iterations for solving Equation (1) is well established at this time. These iterative methods mainly consist of two steps:

1. Seek an initial point or segment.
2. Iterate by Newton–Raphson scheme until convergence.

The robustness and the efficiency of Newton–Raphson scheme depends significantly on the initial guess. Therefore, to assure convergence of the second step, careful selection of initial guess is needed. In addition, if the NURBS entity has only $\mathscr{G}_0$ continuity at some local point, the derivative based Newton–Raphson scheme would fail to converge to such a point.

To assure robustness of the iterations, a significant focus of the existing literature is on eliminating portions of the curve or surface where the solution cannot lie. Piegl and Tiller [2] developed a non-iterative, heuristic algorithm where a NURBS surface was decomposed into quadrilaterals and test points were projected onto the closest quadrilateral. Ma and Hewitt [12] described a search for the initial guess of the footpoint by recursively sub-dividing the Bézier segment associated with a valid control polygon. However, using the control polygon to eliminate segments of the curve may exclude the correct solution [13]. Selimovic [14] improved Ma and Hewitt's method using selective sub-division of the NURBS curve (surface) based on distance to the end (corner) point of the entity. Chen et al. [15] pointed out the need for all control points to lie on different sides of a hyperplane in Selimovic's algorithm and proposed a circular clipping method with a sufficiency condition for a curve to lie outside an elimination circle. Other iterative methods in the physical space have also been proposed for point projection including the geometric first-order iterative [16–18] and geometric second-order iterative [19–21] methods.

In this paper, a robust and efficient point projection technique for low degree two-dimensional (2D) NURBS curves and three-dimensional (3D) NURBS surfaces is developed. The proposed technique preserves and operates directly on the parametric description of the NURBS curve or surface. Therefore, the technique gives a projected point directly on the curve or surface when query points lie on the parametric curve or surface. In addition, the technique is robust for curves and surfaces with high local curvature or $\mathscr{G}_0$ continuity compared to techniques that rely on derivatives. A detailed comparison of existing methods in the literature relative to the proposed method is listed in Table 1.

Table 1. Comparison of methods used for point projection in literature

Description	Reference	Algebraic	Initial Guess Dependent?	Efficiency	Accuracy	Smoothness
Subdivision method	[22]	Yes	No	High	Medium	No
Subdivision method + Newton–Raphson method	[12,14,15]	No	Yes	High	High	No
Geometric iteration method	[16–21]	No	Yes	High	High	No
Proposed method		Yes	No	High	Medium	Yes

3. Background on Algebraic Level Sets

In addition to point projection, blending behaviors on immersed boundaries with domain also requires distance estimates from the boundary to a point in the domain. This is because the behavioral influence of the immersed boundaries decays monotonically with distance. While traditionally distances are estimated using Newton–Raphson iterations, recently, Upreti et al. [23,24] developed algebraic procedures for efficient computation of distance estimates from curves and surfaces. The method developed by Upreti et al. relies on converting the parametric NURBS geometry to its implicit form using the Dixon resultant, and constructing level sets on the implicit form of the geometry. Upreti et al. termed these level sets as algebraic level sets. The construction of the algebraic level sets requires one to decompose the NURBS entity into constituent Bézier patches and later to blend the level sets constructed on Bézier patches using R-functions. The algebraic level sets provide monotonic measures of distance that are accurate to exact distance near the boundary. While the algebraic level sets are approximate measures of distance, they are sufficient to capture the interaction of domain approximations with those on the immersed boundary during analysis. The algebraic level sets have the following properties:

1. Accurate measure of distance locally near the curve/surface
2. Monotonic function of Euclidean (exact) distance
3. Sufficiently smooth for engineering applications
4. Efficiently obtained without numerical iterations for points close to the curve/surface

For the sake of completeness, we briefly review the computation of algebraic level sets and illustrate the procedure through simple examples.

3.1. Implicitization of a Parametric Curve

Given a rational parametric curve $\mathbf{C}(X(u), Y(u), W(u))$ of degree p with $x = \frac{X(u)}{W(u)}, y = \frac{Y(u)}{W(u)}$, one can construct two auxiliary polynomials:

$$g_1(x,u) = W(u)x - X(u) = 0 \tag{4a}$$
$$g_2(y,u) = W(u)y - Y(u) = 0 \tag{4b}$$

The above polynomial equations can be rearranged in descending power of u as follows:

$$g_1(u) = a_p u^p + a_{p-1}u^{p-1} + \cdots + a_1 u + a_0 \tag{5a}$$
$$g_2(u) = b_p u^p + b_{p-1}u^{p-1} + \cdots + b_1 u + b_0 \tag{5b}$$

From the above, the following resultant system may be obtained through algebraic manipulations [25]:

$$\begin{bmatrix} (a_p b_{p-1}) & \cdots & (a_p b_0) \\ \vdots & \ddots & \vdots \\ (a_p b_0) & \cdots & (a_1 b_0) \end{bmatrix} \begin{pmatrix} u^{p-1} \\ u^{p-2} \\ \vdots \\ 1 \end{pmatrix} = \left[\mathbf{M}^B\right]_{p\times p} \begin{pmatrix} u^{p-1} \\ u^{p-2} \\ \vdots \\ 1 \end{pmatrix} = 0 \tag{6}$$

where $(a_i b_j) = a_i b_j - a_j b_i$, $\mathbf{M}^B$ is Bezout matrix and is a function of x and y having the following important property:

$$\mathbf{M}^B(x,y) = \mathbf{M}^B_x x + \mathbf{M}^B_y y + \mathbf{M}^B_w \tag{7}$$

where $\mathbf{M}^B_x, \mathbf{M}^B_y$, and $\mathbf{M}^B_w$ depend on control point coordinates and weights. Therefore, these matrices can be pre-computed for a given rational parametric curve and re-used given any new physical point $\mathbf{x}$. The determinant, $\det(\mathbf{M}^B(\mathbf{x}))$, is defined as the Bezout resultant. Since all the allowable parameter values u for curve $\mathbf{C}(X(u), Y(u), W(u))$ are roots of Equation (6), $\det(\mathbf{M}^B(\mathbf{x})) = 0$ gives the equation of the implicitized curve. Thus, the algebraic level sets corresponding to a rational parametric curve (e.g., Bézier curve) are given by:

$$\Gamma(\mathbf{x}) = \det(\mathbf{M}^B(\mathbf{x})) \tag{8}$$

An example of algebraic level sets is shown in Figure 3.

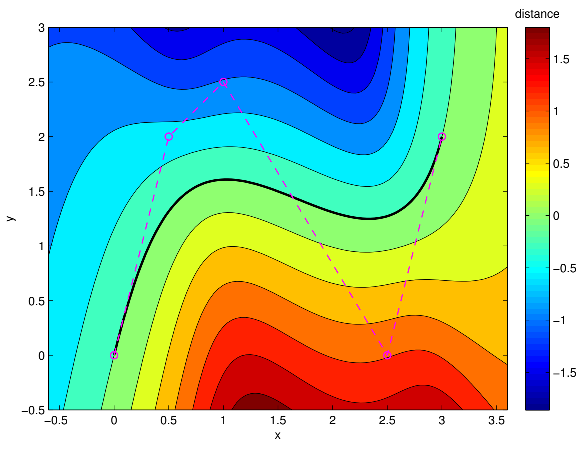


Figure 3. Implicitization of a quartic Bézier curve. Level set $\Gamma(\mathbf{x}) = \det(\mathbf{M}(\mathbf{x}))$ can be used as a measure of distance.

3.2. Boolean Operations by R-Functions

As observed in Figure 3, the direct implicitization extends the parametric curve beyond its end points, and yields an invalid distance measure in the extended region. Therefore, it is desirable to trim the curve $\mathbf{C}(X(u), Y(u), W(u))$ within its parameter range $u \in [a, b]$. In related prior work, Biswas and Shapiro [26] constructed an approximate distance from a line segment as:

$$g = \sqrt{\Gamma^2 + \frac{(|\phi| - \phi)^2}{4}} \tag{9}$$

with Γ being the distance in the normal direction from a piecewise linear approximation of the curve, and ϕ being distance to the region formed by a circle circumscribing the line that is positive inside and negative outside. This form yields a smooth distance function across the boundary $\phi = 0$. Upreti et al. [23] extended the above idea by carrying out Boolean operations on fields obtained on (individual curve segments of) an arbitrarily shaped parametric curve and an enclosing convex region using R-functions (Figure 4). The R-functions [27,28] enable a smooth and purely algebraic Boolean operation, and result in a continuous distance measure. Two specific R-functions used in this study are:

1. R-conjunction, equivalent to Boolean intersection:

$$g_1 \wedge g_2 = g_1 + g_2 - \sqrt{g_1^2 + g_2^2} \tag{10}$$

2. R-disjunction, equivalent to Boolean union:

$$g_1 \vee g_2 = g_1 + g_2 + \sqrt{g_1^2 + g_2^2} \tag{11}$$

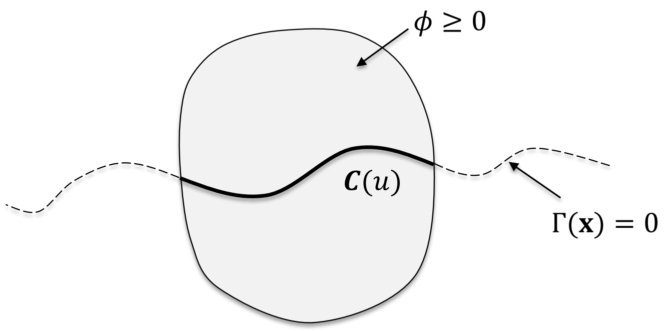


Figure 4. A convex region $\phi \geq 0$ is used to trim the implicitized curve $\Gamma(\mathbf{x}) = 0$ constructed from a parametric curve $\mathbf{C}(u)$.

A well known property of Bézier and NURBS geometry is the convex hull property, which assures that the curve/surface is contained within its convex hull constructed using the control points. Upreti et al. [23] used the convex hull property of Bézier and NURBS curves to provide a natural convex region bounded by control points for curve trimming. Assume that the ith hyper-plane of the convex hull is expressed as:

$$h_i(\mathbf{x}) = \mathbf{n}_i \cdot \mathbf{x} + b_i = 0 \tag{12}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a spatial point, $\mathbf{n}_i \in \mathbb{R}^n$ is inner normal, and $b_i \in \mathbb{R}$ is offset. Thus, the exact distance from any point $\mathbf{x}$ to the hyper-plane $h_i(\mathbf{x}) = 0$ is $h_i(\mathbf{x})$. The function ϕ can be obtained by applying R-conjunction operation of Equation (10) to all $h_i(\mathbf{x}), i = 1, 2, ..., n$. An example of a cubic Bézier curve is shown in Figure 5.

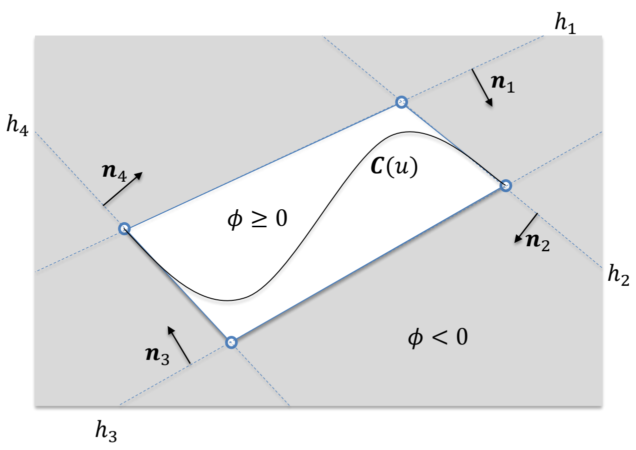


Figure 5. The control points of a cubic Bézier curve $\mathbf{C}(u)$ form a convex hull consisting of four hyper-planes h_1, h_2, h_3 and h_4 with inner normals $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$, and $\mathbf{n}_4$, respectively. Boolean intersection of the four hyper-planes using R-functions yields a trimming region $\phi \geq 0$.

3.3. Normalization and Composition of Algebraic Level Sets

The aforementioned procedure generates a monotonic and continuous distance measure for a basic parametric curve such as a Bézier curve. Piecewise polynomial curves such as NURBS curves, on the other hand, require decomposition to Bézier segments and composition of algebraic level sets of the obtained segments. Further, normalization for individual level sets is desired to yield a monotonically varying composed field. Considering a physical footpoint $\mathbf{x}_f$, one can approximate $\Gamma(\mathbf{x}_f)$ to a first order using Taylor expansion:

$$\Gamma(\mathbf{x}_f) = \Gamma(\mathbf{x}) - \frac{\partial \Gamma(\mathbf{x})}{\partial \mathbf{n}} d \tag{13}$$

Since the resultant has exact zero set on a parametric curve, i.e., $\Gamma(\mathbf{x}_f) = 0$, one can derive a distance in the normal direction as follows:

$$d = \frac{\Gamma(\mathbf{x})}{\frac{\partial \Gamma(\mathbf{x})}{\partial \mathbf{n}}} = \frac{\Gamma}{\|\nabla \Gamma\|} \tag{14}$$

After obtaining normalized algebraic level sets for each decomposed Bézier segment, one can compose them using R-conjunction operation (Equation (10)) and thereby generate the desired algebraic level sets. As demonstrated in [23], the R-conjunction operation preserves the normalization of individual Bézier segments. However, an implicitized curve obtained from a Bézier curve of degree p may have as many as $\frac{1}{2}(p-1)(p-2)$ self-intersections or double points [29]. Any double points inside the convex hull will affect the algebraic level sets construction, and therefore need to be moved out by sub-divisions of the Bézier curve. The algorithm to carryout this process is discussed in Reference [23]. Thus, for practical reasons of avoiding more than one double point while enabling sufficient generality in modeling complex geometries, the methodology is restricted to low degree NURBS curves $(p \leq 3)$. Figure 6 shows an example of algebraic level sets of an open curve containing two points with $\mathscr{G}^0$ continuity.

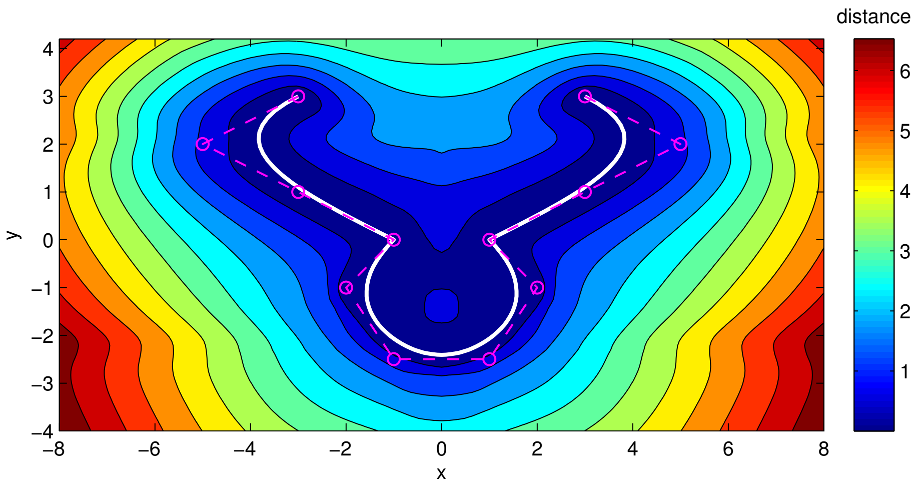


Figure 6. Algebraic level sets of a symmetric cubic spline. $\mathscr{G}^0$ continuity is present at $(x, y) = (-1, 0)$ and $(1, 0)$. The generated algebraic level sets retain the symmetry while ensuring the smoothness of the field.

3.4. Extension to NURBS Surface

The algebraic level sets construction can be extended to three-dimensional NURBS surfaces in a straightforward manner by implicitizing the rational parametric surface with the Dixon resultant [25]. Given a rational parametric surface $\mathbf{S}(X(u,v), Y(u,v), Z(u,v), W(u,v))$ of degree $p \times q$ with $x = \frac{X(u,v)}{W(u,v)}, y = \frac{Y(u,v)}{W(u,v)}$ and $z = \frac{Z(u,v)}{W(u,v)}$, three auxiliary polynomials can be formed as follows:

$$g_1(x, u, v) = W(u, v)x - X(u, v) = 0 \tag{15a}$$
$$g_2(y, u, v) = W(u, v)y - Y(u, v) = 0 \tag{15b}$$
$$g_3(z, u, v) = W(u, v)z - X(u, v) = 0 \tag{15c}$$

As before, using algebraic elimination theory, one can derive the corresponding resultant system for surface $\mathbf{S}$:

$$\left[\mathbf{M}^D\right]_{2pq \times 2pq} \begin{pmatrix} 1 & u & \cdots & u^{p-1} & \cdots & v^{2q-1} & uv^{2q-1} & \cdots & u^{p-1}v^{2q-1} \end{pmatrix}^T = 0 \tag{16}$$

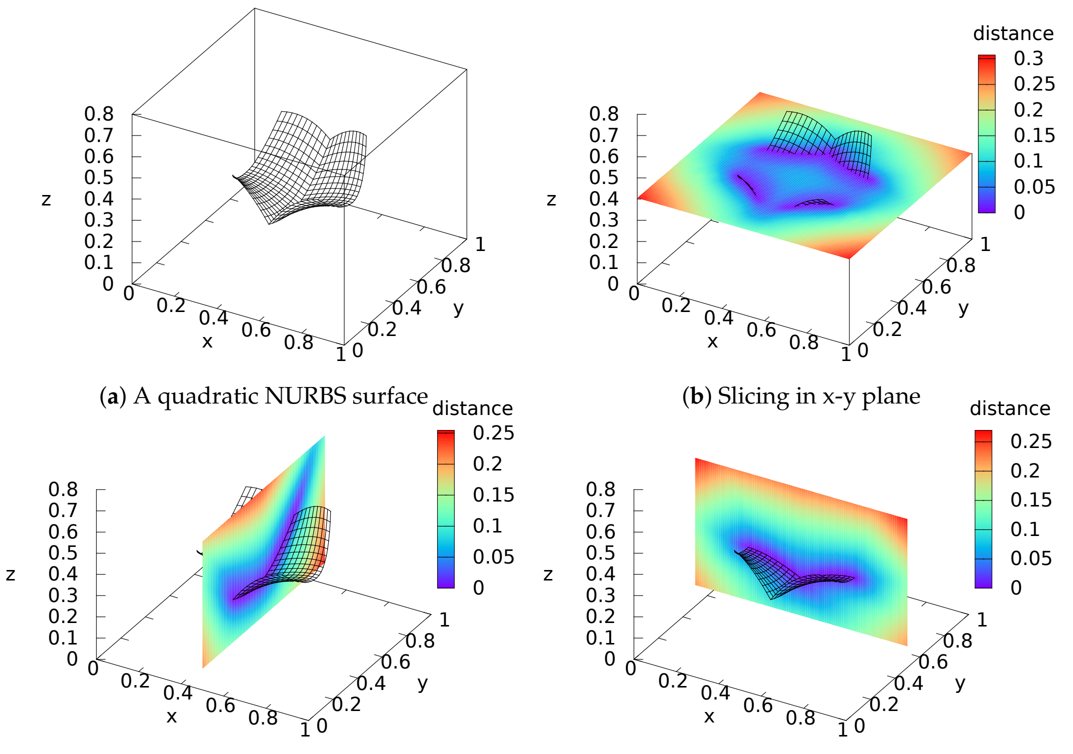
where the vector is indexed lexicographically. $\mathbf{M}^D$ is the Dixon matrix, which also possesses a property analogous to Equation (7) of linearity with respect to x, y, and z:

$$\mathbf{M}^D(\mathbf{x}) = \mathbf{M}^D_x x + \mathbf{M}^D_y y + \mathbf{M}^D_z z + \mathbf{M}^D_w \tag{17}$$

where, as before, $\mathbf{M}^D_x, \mathbf{M}^D_y, \mathbf{M}^D_z$, and $\mathbf{M}^D_w$ depend on control point coordinates and weights. The determinant of the Dixon matrix is the Dixon resultant:

$$\Gamma(\mathbf{x}) = \det(\mathbf{M}^D(\mathbf{x})) \tag{18}$$

An example of the algebraic level sets from a free surface is illustrated in Figure 7. The pseudo-code in Algorithm 1 shows the generic steps in algebraic level sets computation for NURBS curves and surfaces. Both NURBS curves and surfaces are denoted by $\mathbf{C}(\mathbf{u})$ here for notational convenience, with the implicit understanding that $\mathbf{u} = (u)$ for curves and $\mathbf{u} = (u, v)$ for surfaces.



(**a**) A quadratic NURBS surface (**b**) Slicing in x-y plane

(**c**) Slicing in y-z plane (**d**) Slicing in x-z plane

Figure 7. Algebraic level sets from a symmetric quadratic NURBS surface. (**a**) The valley of the surface contains only a $\mathscr{G}^0$ continuity across the plane of symmetry. The level sets are plotted over three principal planes slicing the surface: (**b**) x-y plane; (**c**) y-z plane; and (**d**) x-z plane.

Algorithm 1 Algebraic level sets algorithm.

Input: NURBS curve or surface $\mathbf{C}(\mathbf{u})$ and given test point $\mathbf{x}$
Output: Algebraic distance measure d from $\mathbf{x}$ to the NURBS entity $\mathbf{C}(\mathbf{u})$

```
1:  function ALGEBRAIC_DISTANCE(C, x)
2:      $\mathscr{B}(\mathbf{C}) \leftarrow$ Split NURBS entity C into a Bézier set with segments $B_i, i = 1, 2, \cdots, n$
3:      for $i \leftarrow 1, n$ do                      ▷ Loops are independent and parallelizable
4:          $h_i \leftarrow$ Create convex hull for $B_i \in \mathscr{B}(\mathbf{C})$
5:          $\Gamma_i \leftarrow$ Compute the Bezout or Dixon resultant using Equation (8) or Equation (18),
    respectively
6:          $\Gamma_i \leftarrow \frac{\Gamma_i}{\|\nabla\Gamma_i\|}$      ▷ Normalization of the resultant using Equation (14)
7:          $d_i \leftarrow$ Carryout Boolean union of distance fields of $h_i$ obtained using Equation (9) with $\Gamma_i$
8:      end for
9:      $d \leftarrow$ Carryout Boolean intersection of individual level sets $d_i, i = 1, 2, \cdots, n$ using Equation
    (10)
10: end function
```

3.5. Time Complexity of the Algebraic Level Sets Algorithm

The algorithmic complexity of Algorithm 1 is given in [23] and reproduced here. Consider a NURBS curve of degree p. Further, assume that the NURBS curve decomposes into n Bézier curves. This decomposition process can be carried out in $O(n)$ time. For each step in constructing the algebraic level sets of the Bézier curve, the time complexity is a function of its degree p and is given in Table 2.

Since each Bézier curve comprises of $p+1$ control points, the construction of its convex hull requires $O(p\log p)$ time. Since the convex hull contains at most $p+1$ edges, computing the normalized distance field for the hull requires $O(p)$ time. Finding the Bezout/Dixon resultant involves evaluating the determinant of the Bezout matrix of size $p \times p$ and costs $O(p^3)$ time. Normalization of this resultant requires the gradient of the resultant, which in turn requires solving a linear system (see Equation (21)) and costs $O(p^3)$ time. The trimming operation only evaluates Equation 9 and can be carried out in $O(1)$ time. Thus, constructing algebraic level sets for a Bézier curve segment requires $O(p^3)$ time. These level sets constructed on Bézier curve segments can then be combined to construct level sets for the NURBS curve in $O(np^3)$ time. The computational time complexity for a NURBS surface of degree $p \times q$ may be obtained analogously. For a Bézier segment of such a NURBS surface, the convex hull contains at most $(p+1)(q+1)$ points and edges, and the Dixon matrix used for the resultant is of size $2pq \times 2pq$. From the step-by-step time complexities listed in Table 2, construction of algebraic level sets for a Bézier surface segment requires $O(p^3q^3)$ time, while that of the NURBS surface requires $O(np^3q^3)$ time.

Table 2. Time complexity of each step in Algorithm 1 for computing the algebraic level sets for a Bézier segment. Time complexities are listed for Bézier curves of degree p and Bézier surfaces of degree $p \times q$.

Step	**Time Complexity**	
	Curve	**Surface**
Convex hull construction	$O(p\log(p))$	$O(pq\log(pq))$
Distance field of convex hull	$O(p)$	$O(pq)$
Computing Dixon resultant	$O(p^3)$	$O(p^3q^3)$
Normalization of resultant	$O(p^3)$	$O(p^3q^3)$
Trimming operation	$O(1)$	$O(1)$

4. Methodology of Algebraic Point Projection

In this section, we describe the developed procedure for algebraic point projection. The development of the algorithmic procedure is initially motivated using one-parameter NURBS curves and later extended to two-parameter NURBS surfaces.

4.1. Algebraic Point Projection for a NURBS Curve

As illustrated using Figure 1, iterative numerical solution for point projection may lead to a discontinuity in the projected point or may miss the correct solution. Hence, we develop in this section a purely algebraic point projection algorithm with the following properties:

1. Exact at any point on the curve or surface, i.e., exact point inversion
2. Controllable accuracy when projected from points near the curve or surface
3. Efficient, non-iterative, and non-recursive solution procedure
4. Footpoints are continuous even near curve segments with high curvature
5. Valid solutions even when projected onto curves with only $\mathscr{G}^0$ continuity

The present method consists of two steps: estimation of the footpoint in physical space and parametric inversion using the resultant matrix.

4.1.1. First Order Algebraic Point Projection in Physical Space

From Equation (14), the gradient of the normalized approximate distance function to a Bézier segment is derived as:

$$\nabla d = \left(\mathbf{I} - \frac{\Gamma}{\|\nabla\Gamma\|^2}\mathbf{H} \right) \frac{\nabla\Gamma}{\|\nabla\Gamma\|} \tag{19}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{H}$ is the Hessian of function $\Gamma(\mathbf{x})$. Using the above gradient, the physical footpoint $\mathbf{x}_f$ can now be approximately located as:

$$\mathbf{x}_f = \mathbf{x} - d\frac{\nabla d}{\|\nabla d\|} \tag{20}$$

To calculate d and ∇d using Equations (14) and (19), it would appear at first glance as though Γ, $\nabla\Gamma$, and $\mathbf{H}$ need to be evaluated for every test point. However, the following derivation as well as procedural detail show that d and ∇d can be computed efficiently without explicitly calculating Γ, $\nabla\Gamma$ or $\mathbf{H}$. One can express $\nabla\Gamma$ and $\mathbf{H}$ in terms of Bezout matrix $\mathbf{M}^B$ and its components $\mathbf{M}_x^B$ and $\mathbf{M}_y^B$ (the superscript B is dropped in the following for ease of reading):

$$\nabla\Gamma = |\mathbf{M}|\begin{bmatrix}\mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial x}\mathbf{M}\right)\\ \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial y}\mathbf{M}\right)\end{bmatrix} = |\mathbf{M}|\begin{bmatrix}\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\end{bmatrix} = \Gamma\tilde{\mathbf{g}} \tag{21}$$

$$\mathbf{H} = |\mathbf{M}|\begin{bmatrix}\begin{array}{c}\mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)^2\right)\end{array} & \begin{array}{c}\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\right)\end{array}\\ \begin{array}{c}\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\right)\end{array} & \begin{array}{c}\mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)^2\right)\end{array}\end{bmatrix} = \Gamma\tilde{\mathbf{H}} \tag{22}$$

where $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{H}}$ are the vector/matrix multiplying $|\mathbf{M}|$ in the above equations, respectively. Substituting Equations (21) and (22) back into Equations (14) and (19), one obtains:

$$d = \frac{1}{\|\tilde{\mathbf{g}}\|} \tag{23}$$

$$\nabla d = \frac{\tilde{\mathbf{g}}}{\|\tilde{\mathbf{g}}\|} - \frac{\tilde{\mathbf{H}}\tilde{\mathbf{g}}}{\|\tilde{\mathbf{g}}\|^3} \tag{24}$$

The efficiency of algebraic point projection in two-dimensional physical space is summarized as follows:

1. Component matrices $\mathbf{M}_x$, $\mathbf{M}_y$, and $\mathbf{M}_w$ are constant for a given rational parametric curve. Therefore, they can be pre-computed and repeatedly used at a point $\mathbf{x}$.
2. Only matrix $\mathbf{M}$ needs to be factorized, and the procedure extensively reuses the products $\mathbf{M}^{-1}\mathbf{M}_x$ and $\mathbf{M}^{-1}\mathbf{M}_y$ when computing $\tilde{\mathbf{H}}$ and $\tilde{\mathbf{g}}$.
3. For a Bezout matrix $\mathbf{M}$ of size $p \times p$, where p is the degree of the rational parametric curve, the typical computational cost is low since p is usually small in engineering applications.

4.1.2. Second Order Algebraic Point Projection in Physical Space

For test points near the curve, the first-order algebraic point projection method described in Section 4.1.1 performs well; however, as shown through the example in Figure 8, when points are farther away, the projection is not accurate. The example is of a quadratic Bézier curve with test points on a horizontal line. The failure of first-order algebraic point projection is because the distance function derived in Equation (14) is essentially a first order approximation using Taylor expansion. To improve accuracy and make algebraic point projection possible for points farther from the curve, we present next a second-order algebraic point projection algorithm in the following. While the second-order algorithm is also approximate, in practice, it is sufficient for enriched immersed boundary analysis since only quadrature points near the immersed boundary needs to be projected on to the surface.

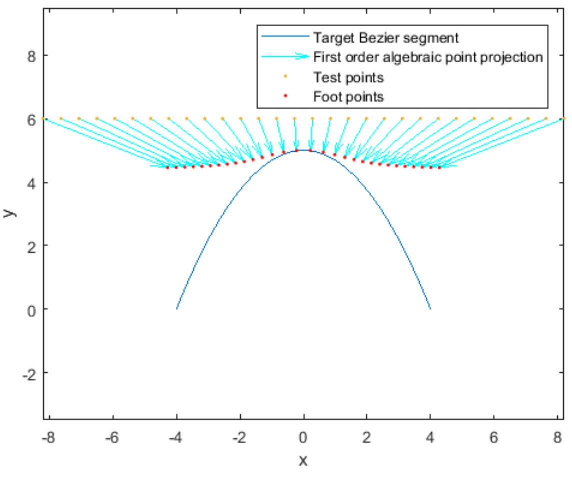


Figure 8. First-order algebraic point projection fails to reach footpoints on the curve for test points not close to quadratic Bézier curve.

The second-order approximation of resultant $\Gamma(\mathbf{x}_f)$ at footpoint $\mathbf{x}_f$ can be written as:

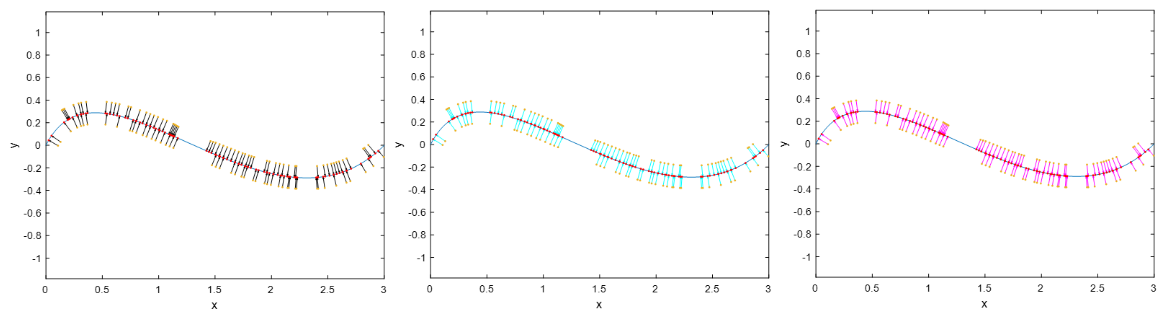
$$\Gamma(\mathbf{x}_f) = \Gamma(\mathbf{x}) - \frac{\partial\Gamma(\mathbf{x})}{\partial\mathbf{n}} d_{quad} + \frac{\partial^2\Gamma(\mathbf{x})}{\partial\mathbf{n}^2} d_{quad}^2 \tag{25}$$

Since the resultant at footpoint is zero, i.e., $\Gamma(\mathbf{x}_f) = 0$, Equation (25) can be rearranged as:

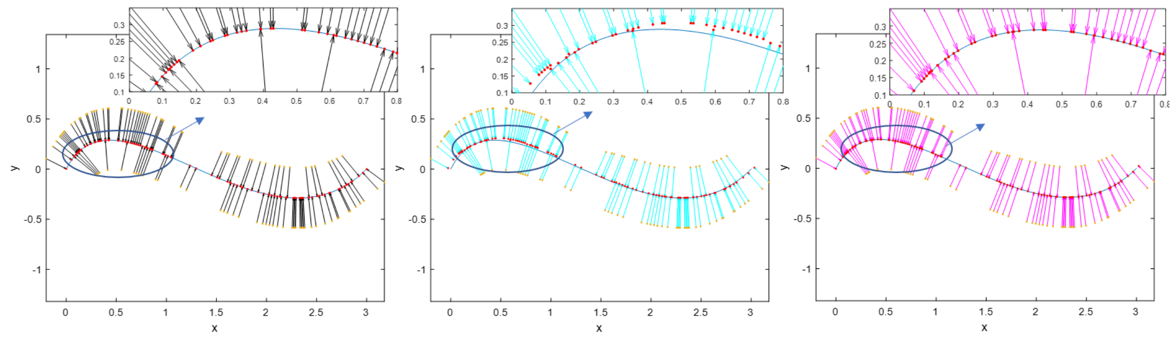
$$d_{quad} = \frac{1}{\mathbf{n}\cdot\mathbf{H}\cdot\mathbf{n}}(\nabla\Gamma\cdot\mathbf{n} \pm \sqrt{(\nabla\Gamma\cdot\mathbf{n})^2 - 2\Gamma\mathbf{n}\cdot\mathbf{H}\cdot\mathbf{n}}) \tag{26}$$

where d_{quad} is the distance estimate between the test point and footpoint based on the second-order approximation. $\nabla\Gamma$ and $\mathbf{H}$ are the gradient and Hessian of algebraic level sets, which are calculated as described in Section 4.1.1. For test points near the curve, a good approximation of normal vector $\mathbf{n}$ could be $\mathbf{n} = \frac{\nabla\Gamma}{\|\nabla\Gamma\|}$ or $\mathbf{n} = \frac{\nabla d}{\|\nabla d\|}$.

Algebraic point projection in two-dimensional physical space is validated in Figure 9, where the second-order point projection result is compared against first-order point projection as well as Newton–Raphson iterations for various test distances. Both algebraic point projection methods yield the reference solution obtained using the Newton–Raphson method in the limit when the query point is on the curve/surface, and when test distance is small ($\bar{d} = 0.1$). The second-order point projection, however, converges to the solution when distances are larger ($\bar{d} = 0.3$), where the first-order point projection fails.



(**a**) Test distance $\bar{d} = 0.1$



(**b**) Test distance $\bar{d} = 0.3$

Figure 9. Point projection for cubic Bézier curve in two-dimensional physical space using the developed algebraic method as well as Newton–Raphson iterations for: (**a**) test distance $\bar{d} = 0.1$; and (**b**) test distance $\bar{d} = 0.3$. From left to right, Newton–Raphson method, first-order algebraic point projection, and second-order algebraic point projection are shown. The inset image shows that first-order point projection fails to converge onto footpoints on the curve when distances are larger.

4.1.3. Improvement to First Order Algebraic Point Projection

As illustrated in Figures 8 and 9, when the test point is far from the curve, first-order algebraic point projection is inaccurate. In immersed boundary analysis, point projection is only required at quadrature points near the boundary since the physical influence of the boundary on the underlying domain is local. For quadrature points that are far from the boundary, a recursive first-order algebraic point projection must be adopted for accuracy. The main idea is to treat the footpoint estimate of the previous step of first-order algebraic point projection as the starting point for the next step, and to recursively proceed until convergence. If a point is far from the boundary, we combine the distance to the curve d in Equation (14) and distance to convex hull of Bézier segment ϕ to get the composed distance $\tilde{d}$ and its gradient $\nabla\tilde{d}$ as follows:

$$\tilde{d} = \sqrt{d^2 + \frac{(|\phi| - \phi)^2}{4}} \tag{27}$$

$$\nabla\tilde{d} = \begin{cases} \nabla d, & \phi \geq 0 \\ \dfrac{2\nabla d + 2\nabla\phi}{\sqrt{d^2 + \phi^2}}, & \phi < 0 \end{cases} \tag{28}$$

The composed distance together with its gradient is then used for estimating the footpoint in Equation (20). The result after recursive execution is illustrated in Figure 10. Comparing against original first-order algebraic point projection in Figure 9b, one can observe that the improved first order procedure is better.

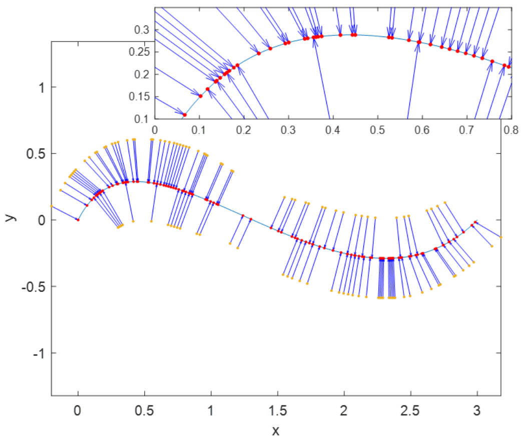
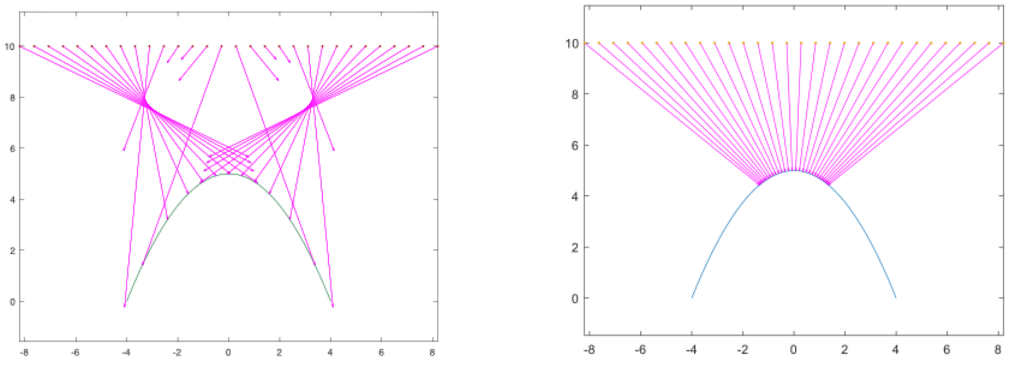


Figure 10. Recursive first-order algebraic point projection at test distance $\bar{d} = 0.3$.

4.1.4. Improvement to Second Order Algebraic Point Projection

While second-order algebraic point projection is more accurate, as should be expected, relative to first-order algebraic point projection, it could be further improved, as discussed below. Since the $\mathbf{n}$ computed locally at a point in the domain is relatively inaccurate for approximating the direction of the vector $\mathbf{x} - \mathbf{x}_f$ at greater distances, Equation (26) will fail to provide the correct solution as distance from the curve increases. As illustrated in Figure 11a, at test points along the line where $\mathbf{n}$ does not intersect with the Bézier curve, no valid solution is obtained. This is because the normal vector $\mathbf{n}$ generated using algebraic level sets usually does not coincide with normal vector $\mathbf{n}_f$ at footpoint, especially when test point is far away. One solution to this issue is to construct a new vector $\mathbf{n}_{corr}$, which is guaranteed to have positive discriminant in Equation (26). A trial $\mathbf{n}_{corr}$ that points to the centroid of control polygon achieves the desired result. This choice of the normal direction at points far away from the curve is simple and effective as shown in Figure 11b. While this correction is not general, it is likely to provide sufficient accuracy in practice for quadrature points where the behavior of the immersed boundary is blended with that of the underlying domain.



(**a**) Before correction (**b**) After correction

Figure 11. Second-order algebraic point projection with test points at a far distance: (**a**) using non-corrected normal vector; and (**b**) using corrected normal vector.

4.1.5. Inversion to Parametric Space

Given a footpoint $\mathbf{x}_f$ in physical space, finding a corresponding parameter u_f such that $\mathbf{C}(u_f) = \mathbf{x}_f$ is the point inversion problem. The direct approach to carrying out point inversion is by solving a system of polynomial equations, which may not be easy to do since there is no analytical solution for high-order polynomials of $deg > 4$, and since $\mathbf{x}_f$ may not lie exactly on curve $\mathbf{C}(u)$ [22].

This drawback can be overcome by using the Bezout matrix [25], as shown in the following. Evaluate $\mathbf{M}^B(\mathbf{x}_f)$ with $\mathbf{x}_f = (x_f, y_f)$:

$$\mathbf{M}^B(\mathbf{x}_f) = \mathbf{M}_x^B x_f + \mathbf{M}_y^B y_f + \mathbf{M}_w^B = [m_{ij}]_{p\times p} \tag{29}$$

Then, Equation (6) can be rewritten as the following over-constrained linear system:

$$A\tilde{\mathbf{u}} \equiv \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1(p-1)} \\ m_{21} & m_{22} & \cdots & m_{2(p-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{p1} & m_{p2} & \cdots & m_{p(p-1)} \end{bmatrix} \begin{pmatrix} u^{p-1} \\ u^{p-2} \\ \vdots \\ u \end{pmatrix} = - \begin{pmatrix} m_{1p} \\ m_{2p} \\ \vdots \\ m_{pp} \end{pmatrix} \tag{30}$$

Matrix A is full ranked if Equation (5) has only one common root, i.e., if $\mathbf{x}_f$ is not a double point [30]. Thus, u_f can be obtained by solving a linear least square problem resulting from Equation (30), which requires bounded computational cost unlike numerical iterations using the Newton–Raphson method. In addition, if a physical test point $\mathbf{x}$ is initially on the curve, then $\mathbf{x}_f = \mathbf{x}$, and the point inversion can be directly applied, without needing to find the footpoint. Alternative to solving the over-constrained system, one can discard any one row of $\mathbf{A}$ in Equation (30), which will make it full rank and the point inversion solution can be obtained by solving linear system of equations.

To compute u_f on a NURBS curve, which is piece-wise polynomial, projection onto the appropriate Bézier segment is required. For this purpose, one can identify the closest Bézier segment using individual algebraic level sets, and apply algebraic point projection on the closest Bézier segment. Denoting the computed parameter on the closest Bézier curve as u_f^B, the corresponding parameter on the original NURBS curve u_f^N can be obtained by purely linear scaling and offset. Unlike iterative or recursive schemes, the algebraic method guarantees the existence of a definite footpoint without needing to manipulate user-controlled parameters such as stop criterion or recursion limit in an effort to coax a solution. If a test point is close to the connection node of two adjacent Bézier segments, a result of $u_f^B < 0$ or $u_f^B > 1$ may be obtained. In this case, higher projection precision can be achieved when a second point projection to the adjacent Bézier segment is attempted (see Figure 12).

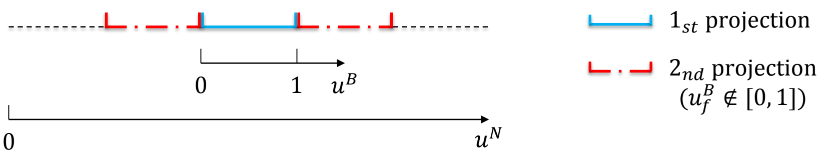


Figure 12. Illustration of the second projection onto an adjacent Bézier curve segment if the first projection yields an out-of-span solution.

4.2. Extension to NURBS Surfaces

Analogous to the algebraic level sets in three-dimensions, the algebraic point projection can be naturally extended to three-dimensional Bézier and NURBS surfaces by replacing Bezout matrix $\mathbf{M}^B$ with Dixon matrix $\mathbf{M}^D$. Since the Dixon matrix also has the linearity property, as given in Equation (17), the basic procedure for point projection remains the same, as outlined in Section 4.1.

4.2.1. Projection in Physical Space

Utilizing the linearity property (Equation (17)), one can rewrite $\nabla\Gamma$ and $\mathbf{H}$ in Equation (19) as follows (as before, superscript D is dropped for ease of reading):

$$\nabla\Gamma = |\mathbf{M}| \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial x}\mathbf{M}\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial y}\mathbf{M}\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\frac{\partial}{\partial z}\mathbf{M}\right) \end{bmatrix} = |\mathbf{M}| \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \\ \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \end{bmatrix} = \Gamma\tilde{\mathbf{g}} \tag{31}$$

$$\mathbf{H} = |\mathbf{M}| \begin{bmatrix} \begin{matrix} \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_x\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)^2\right) \end{matrix} & \begin{matrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\right) \end{matrix} & \begin{matrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\right) \end{matrix} \\ \begin{matrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\right) \end{matrix} & \begin{matrix} \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)^2\right) \end{matrix} & \begin{matrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\right) \end{matrix} \\ \begin{matrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_x\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\left(\mathbf{M}^{-1}\mathbf{M}_x\right)\right) \end{matrix} & \begin{matrix} \mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\mathrm{tr}\left(\mathbf{M}^{-1}\mathbf{M}_y\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_z\right)\left(\mathbf{M}^{-1}\mathbf{M}_y\right)\right) \end{matrix} & \begin{matrix} \mathrm{tr}^2\left(\mathbf{M}^{-1}\mathbf{M}_z\right) \\ -\mathrm{tr}\left(\left(\mathbf{M}^{-1}\mathbf{M}_z\right)^2\right) \end{matrix} \end{bmatrix}$$
$$= \Gamma\tilde{\mathbf{H}} \tag{32}$$

Next, as before, Equations (20), (23) and (24) can be exploited to obtain the physical footpoint $\mathbf{x}_f$ in three-dimensional space. Earlier statements on efficiency of the algebraic point projection for rational parametric curves also apply to rational parametric surfaces except that the size of the Dixon matrix $\mathbf{M}^D$ is $2pq \times 2pq$, where p and q are the degrees of the rational parametric surface in each dimension. Algebraic point projection in three-dimensional physical space is demonstrated in Figure 13, where test points are projected onto a target Bézier surface using the proposed second-order algebraic point projection method and the Newton–Raphson iterations. Again, one can observe that the proposed method leads to accurate solutions for test points closer to the surface.

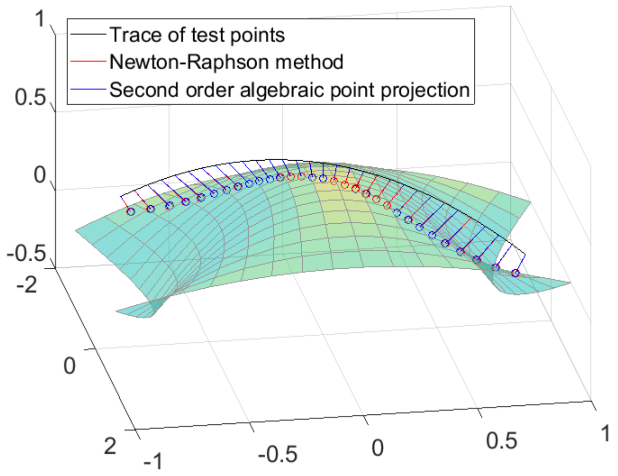


Figure 13. Point projection in 3D physical space using the proposed algebraic method and Newton–Raphson iterations.

4.2.2. Inversion to Parametric Space

The point inversion for rational parametric surfaces can be carried out using Dixon matrix as well. Substituting the physical coordinates of the footpoint $\mathbf{x}_f = (x_f, y_f, z_f)$ in $\mathbf{M}^D$, we get:

$$\mathbf{M}^D(\mathbf{x}_f) = \mathbf{M}_x^D x_f + \mathbf{M}_y^D y_f + \mathbf{M}_z^D z_f + \mathbf{M}_w^D = [m_{ij}]_{2pq\times 2pq} \tag{33}$$

Thus, as before, the homogeneous system in Equation (16) can be converted into an over-constrained non-homogeneous system as follows:

$$\begin{bmatrix} m_{12} & \cdots & m_{1(1+i+jp)} & \cdots & m_{1(2pq)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{(2pq)2} & \cdots & m_{(2pq)(1+i+jp)} & \cdots & m_{(2pq)(2pq)} \end{bmatrix} \begin{pmatrix} u^{p-1}v^{2q-1} \\ \vdots \\ u^i v^j \\ \vdots \\ u \end{pmatrix} = -\begin{pmatrix} m_{11} \\ \vdots \\ m_{(2pq)1} \end{pmatrix} \tag{34}$$

Generally, the parameters (u_f, v_f) of the footpoint $\mathbf{x}_f$ can be computed by solving a least square problem or discarding any one row and then solving a linear system of equations as mentioned earlier. As before, the computation of parameters (u_f^N, v_f^N) on a NURBS surface requires sub-division of the surface into a set of Bézier segments. One can apply algebraic point projection to the Bézier segment with smallest algebraic level set measure, and acquire (u_f^N, v_f^N) from the Bézier parameters (u_f^B, v_f^B) by simple linear scaling and offset. As illustrated in Figure 14, a second projection may be necessary when u_f^B or v_f^B is outside the range $[0, 1]$.

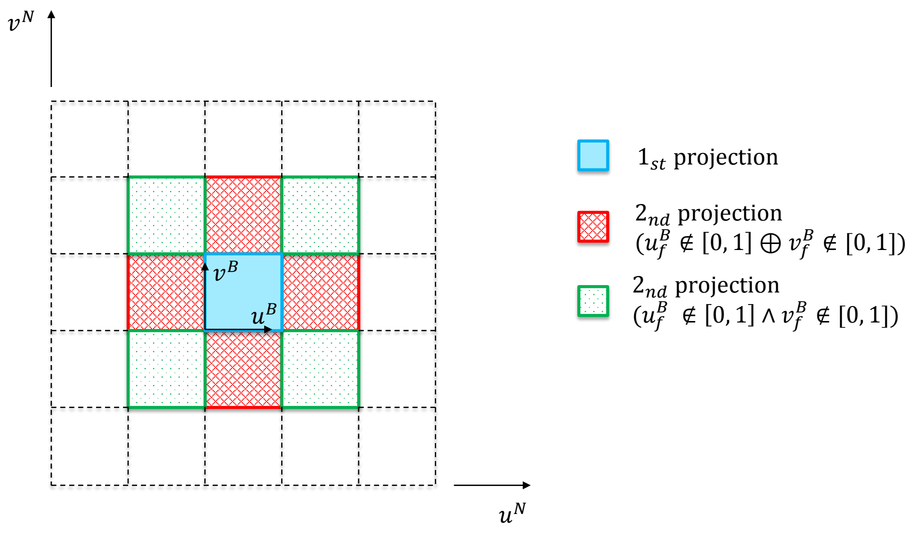

v^N
v^B
u^B
u^N
1_{st} projection
2_{nd} projection
$(u_f^B \notin [0,1] \oplus v_f^B \notin [0,1])$
2_{nd} projection
$(u_f^B \notin [0,1] \wedge v_f^B \notin [0,1])$

Figure 14. Illustration of the second projection onto an adjacent Bézier surface segment if the first projection yields an out-of-span solution.

4.3. Time Complexity of the Algebraic Point Projection Algorithm

The generic pseudo-code of algebraic point projection for both NURBS curves and surfaces is listed in Algorithm 2. As can be seen in Algorithms 1 and 2, algebraic level sets (ALS) and algebraic point projection (APP) are closely connected. Algebraic level sets provide the closest Bézier segment for the first point projection, but also restrict the target curve or surface to be low degree ($p, q \leq 3$) so as to avoid double points.

The time complexity for Algorithm 2 is now presented for both curves and surfaces. Consider a NURBS curve of degree p or a NURBS surface of degree $p \times q$, decomposing into n Bézier segments. For each step in finding the point projection, the time complexity is a function of the degree and is given in Table 3. As shown in Section 3.5, computing the algebraic level set at a point costs $O(np^3)$ time for a curve and $O(np^3q^3)$ time for a surface. Choosing the closest Bézier segment requires finding the minimum of the level sets of n Bézier segments and costs $O(n)$ time. Projection in the physical space to the closest Bézier segment involves finding the gradient and Hessian of the Bezout/Dixon matrix, which requires the solution to a linear system (see Equations (21) and (22)). This costs $O(p^3)$ time for curves and $O(p^3q^3)$ time for surfaces. Point inversion to parametric coordinates requires solving a resultant system (Equations (30) and (34)), which also requires $O(p^3)$ time for curves and $O(p^3q^3)$ time for surfaces. Finally, the parametric coordinates of the Bézier segment are scaled and offset to obtain the parametric coordinates of the NURBS curve/surface. This can be done in $O(1)$ time. Thus, the total time complexity of algebraic point projection is the same as that of computing the algebraic level sets, i.e., $O(np^3)$ for NURBS curves and $O(np^3q^3)$ for NURBS surfaces.

Algorithm 2 Algebraic point projection algorithm.

Input: NURBS curve or surface $\mathbf{C}(\mathbf{u})$ and given point $\mathbf{x}$
Output: Parameter $\mathbf{u}_f^N$ of footpoint on NURBS entity $\mathbf{C}$.

```
1:  function ALGEBRAIC_POINT_PROJECTION(C, x)
2:      $\mathscr{B}(\mathbf{C}) \leftarrow$ Split NURBS entity C into a Bézier set with segments $B_i, i = 1, 2, \cdots, n$
3:      for $i \leftarrow 1, n$ do                    ▷ Loops are independent, parallelization is possible
4:          $d_i \leftarrow$ ALGEBRAIC_DISTANCE($B_i$, x)
5:      end for
6:      $\mathbf{x}_f \leftarrow \mathbf{x} - d_{B_j}(\mathbf{x})\mathbf{n}_{B_j(\mathbf{x})}$ ▷ $j = \arg\min_{i\in[1,n]} d_i$, and $d_{B_j}$ is the distance to $j$th Bezier segemnt using
    1st or 2nd order approximation, $\mathbf{n}_{B_j(\mathbf{x})}$ is normal vector of $d_{B_j}$
7:      $\mathbf{u}_f^B \leftarrow$ Solve Equation (30) with $\mathbf{M}^B(\mathbf{x}_f)$ or Equation (34) with $\mathbf{M}^D(\mathbf{x}_f)$
8:      if $\mathbf{u}_f^B$ is out of span then
9:          $\mathbf{u}_f^B \leftarrow$ Carryout the second projection based on Figure 12 or Figure 14
10:         if Second projection is infeasible or $\mathbf{u}_f^B$ is still out of span then
11:             $\mathbf{u}_f^B \leftarrow$ Compute corresponding parameter value on $B_j$ boundary
12:         end if
13:     end if
14:     $\mathbf{u}_f^N \leftarrow$ Scale and offset $\mathbf{u}_f^B$ based on knot span of C
15: end function
```

Table 3. Time complexity of each step in Algorithm 2 for algebraic point projection. Time complexities are listed for NURBS curves of degree p and NURBS surfaces of degree $p \times q$.

Step	Time Complexity	
	Curve	**Surface**
Computing algebraic level set	$O(np^3)$	$O(np^3q^3)$
Determining the closest Bézier segment	$O(n)$	$O(n)$
Projection in physical space	$O(p^3)$	$O(p^3q^3)$
Point inversion to parametric space	$O(p^3)$	$O(p^3q^3)$
Scaling and offset	$O(1)$	$O(1)$

5. Results and Discussion

Four numerical examples are presented to demonstrate the algebraic point projection methodology of Sections 4.1.1 and 4.1.2. Two of the examples are projections to curves and the remaining two to surfaces. The curve examples show the performance of algebraic point projection in simple tests to achieve the accuracy of the Newton–Raphson method. The examples also reveal the superiority of second-order point projection over first-order point projection. The surface examples further demonstrate the cheaper computational cost of second-order algebraic point projection when compared against the Newton–Raphson method.

The procedure for algebraic point projection implemented in the examples is summarized as follows and illustrated through a flowchart in Figure 15:

1. Decompose NURBS curve or surface and get the corresponding implicit form for each Bézier entity.
2. Construct algebraic level sets for the decomposed Bézier entities.
3. Compose level sets on Bézier patches using R-functions to construct algebraic level sets on the NURBS entity.
4. Find nearest Bézier entity for a given quadrature point in the domain.
5. Project test point on the nearest Bézier entity to get foot point in the physical space.
6. Apply point inversion and decide if recursive projection is needed.
7. Obtain global parametric solution by scaling the parameter value of the Bézier entity to that of the NURBS entity.

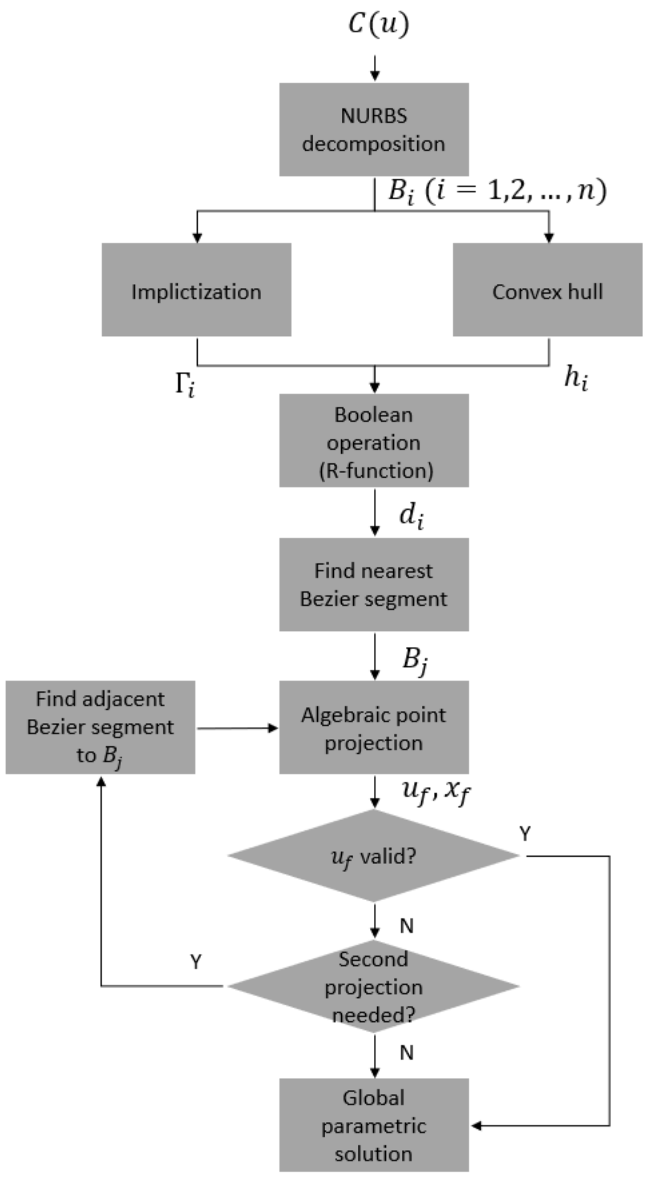
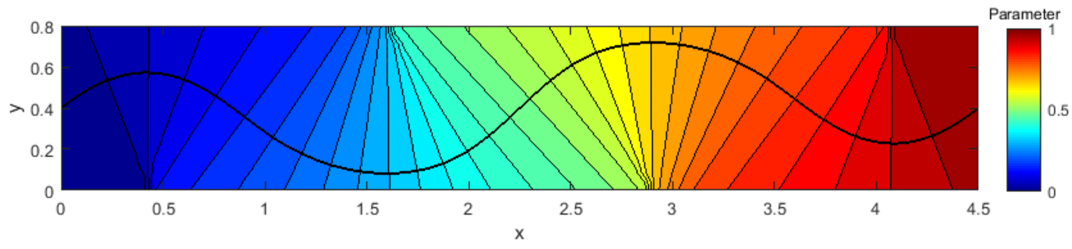

$C(u)$
NURBS decomposition
$B_i\ (i = 1,2,\ldots,n)$
Implictization
Convex hull
Γ_i
h_i
Boolean operation (R-function)
d_i
Find nearest Bezier segment
B_j
Find adjacent Bezier segment to B_j
Algebraic point projection
u_f, x_f
u_f valid?
Y
N
Second projection needed?
Y
N
Global parametric solution

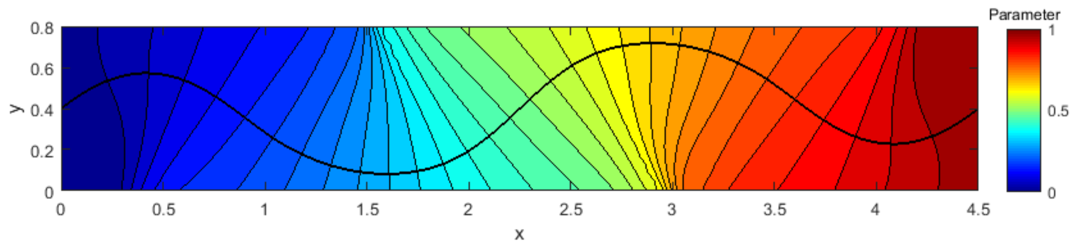
Figure 15. Flowchart for execution of algebraic point projection.

5.1. Curve Tests

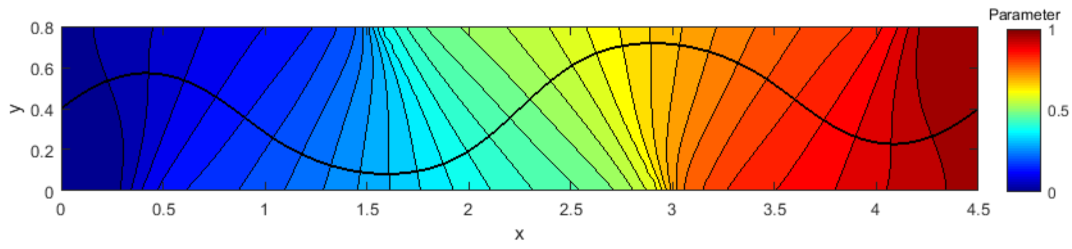
The first curve example is illustrated in Figure 16, where physical points in the underlying domain are projected onto a given NURBS curve using both Newton–Raphson iterations as well as the algebraic point projection. Contour levels indicate the value of parameter u_f^N of the predicted footpoint.



(**a**) Parameter values of the footpoints obtained through Newton–Raphson iterations



(**b**) Parameter values of the footpoints obtained through first-order algebraic point projection



(**c**) Parameter values of the footpoints obtained through second-order algebraic point projection

Figure 16. Parameter values of footpoints obtained using: (**a**) Newton–Raphson method; (**b**) first-order algebraic point projection; and (**c**) second-order algebraic point projection. Parameter range of NURBS curve is $[0, 1]$.

As can be observed in Figure 16, both first- and second-order algebraic point projection provide exact on-curve solutions and good approximate solutions to the parameter values from points near the curve. In addition, one may observe that there are two regions, at the bottom-left and upper-right of Figure 16a, respectively, where due to high curvature of nearby curve segments, a jump in parameter value occurs when Newton–Raphson iterations are used. Such discontinuities disappear when algebraic point projection is used. The relationship between distance from the curve and relative error is obtained for both first- and second-order algebraic point projection methods using the Newton–Raphson method as a reference (Figure 17). The test points are picked along an arbitrary vertical trace line in Figure 16. The parameter range of the NURBS curve is $[0, 1]$ in the example. As the test point moves closer to the target curve, the projection error decreases quadratically (first-order point projection) and cubically (second-order point projection).

The second curve example shown in Figure 18 demonstrates the robustness of the algebraic point projection when the footpoint is either discontinuous or non-existent. One can observe in Figure 18b that, although Points A and B are continuous in terms of the parametric value on the tracing curve, the Newton–Raphson method results in a large jump in parametric solution A_f^N and B_f^N, whereas the algebraic method yields a continuous parameter value. In general, the algebraic projected solution is smoother and matches the Newton–Raphson solution well, and second-order point projection is more accurate than first-order point projection.

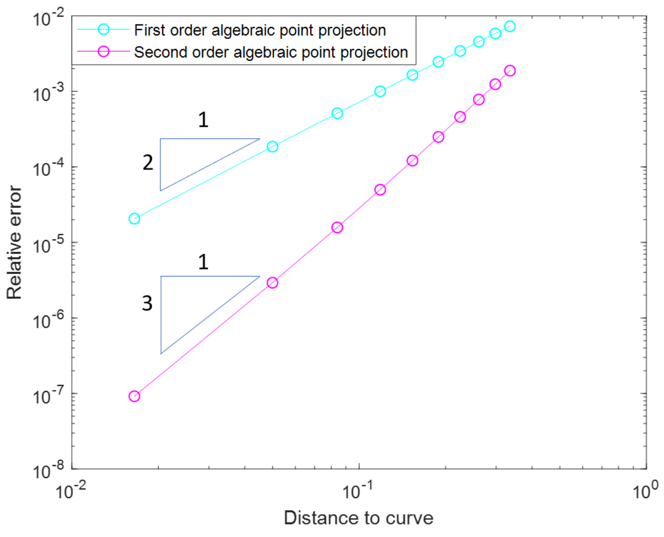
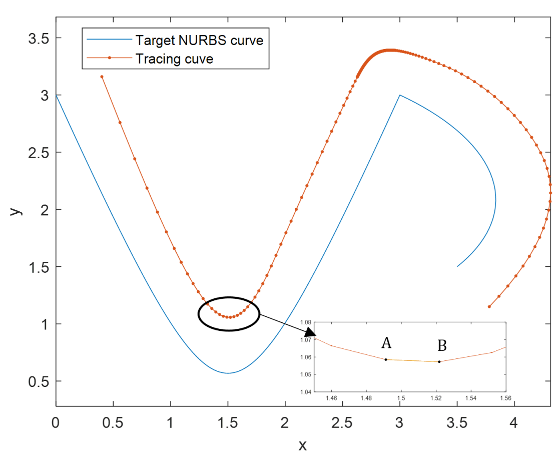
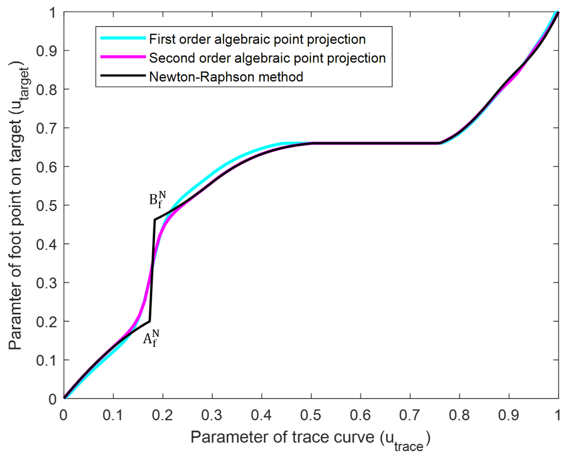


Figure 17. Relative error $\frac{|u_f^{NR} - u_f^{APP}|}{b-a}$ vs. distance of test points $d(\mathbf{x})$, where u_f^{NR} and u_f^{APP} are parameter values of footpoints obtained using the Newton–Raphson method and the algebraic point projection respectively.



(**a**) Trace of test points and target curve. The trace was roughly offset relative to the target.

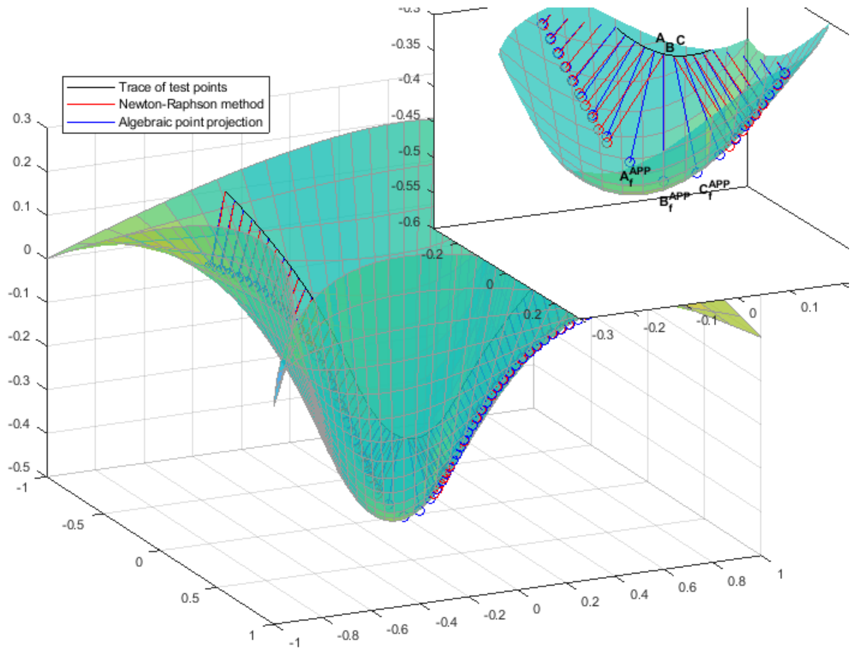


(**b**) Parameter values of footpoints vs. those of the trace curve

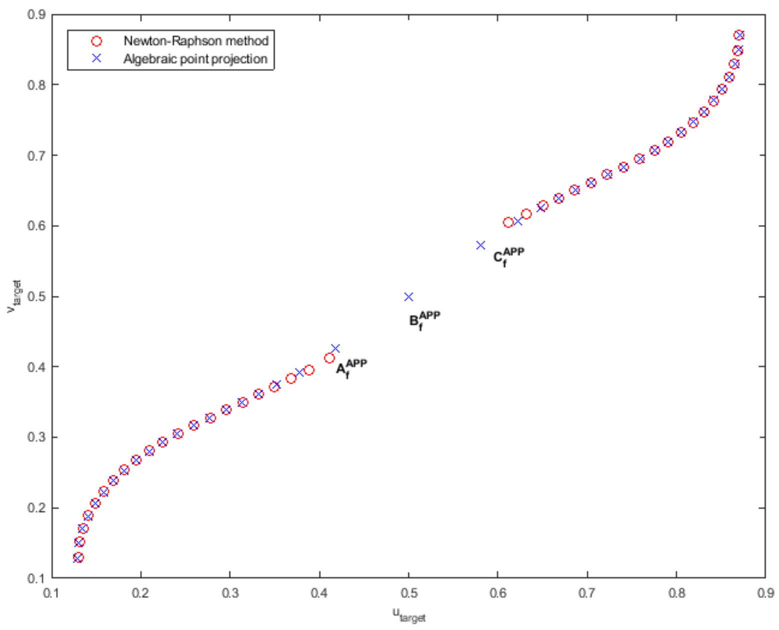
Figure 18. Illustration of the robustness of the 2D algebraic point projection for the NURBS curve. (**a**) Trace of points that were projected onto target curve. (**b**) Solution parameter of footpoints on target curve vs. parameter of trace curve for the two methods. Parameter discontinuity in Newton–Raphson solution occurs due to non-uniqueness of the footpoint near the local minimum at $u_{trace} \approx 0.176$

5.2. Surface Tests

The first and second surface examples demonstrate the robustness of the second-order algebraic point projection for NURBS surfaces involving discontinuous and non-existent footpoints, respectively. In the first example (Figure 19), the discontinuous projection occurs again when the Newton–Raphson method is applied on a surface segment with high mean curvature. In the second example (Figure 20), the Newton–Raphson method failed in regions where the mathematical footpoints do not exist. Not only does the algebraic point projection overcome those problems, but it also produces an accurate and efficient solution. As listed in Table 4, the computational cost per point of the proposed method is only 26% and 14% of that of the Newton–Raphson method in the first and second surface examples, respectively.

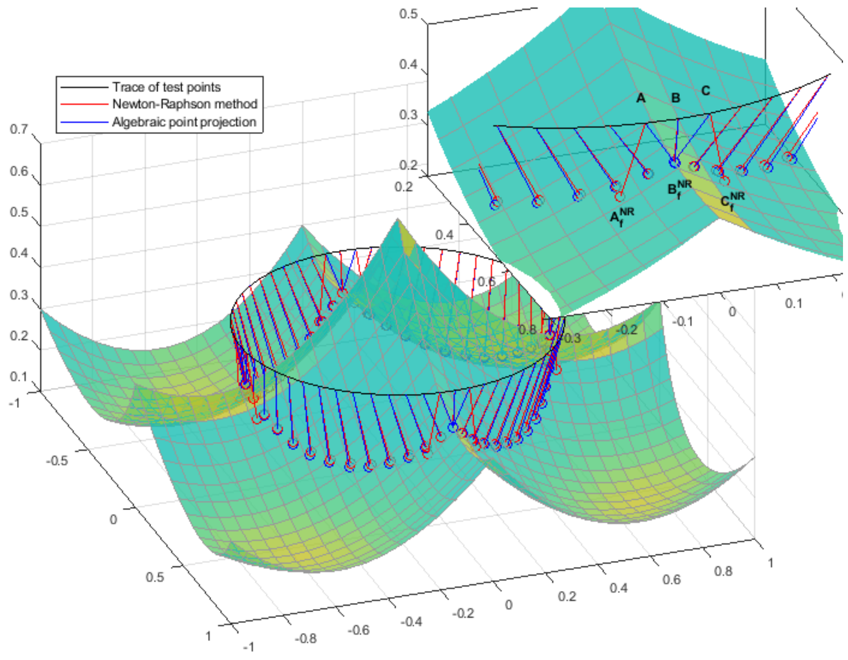


(**a**) Trace of test points, bowl-shaped target surface and the identified footpoints.

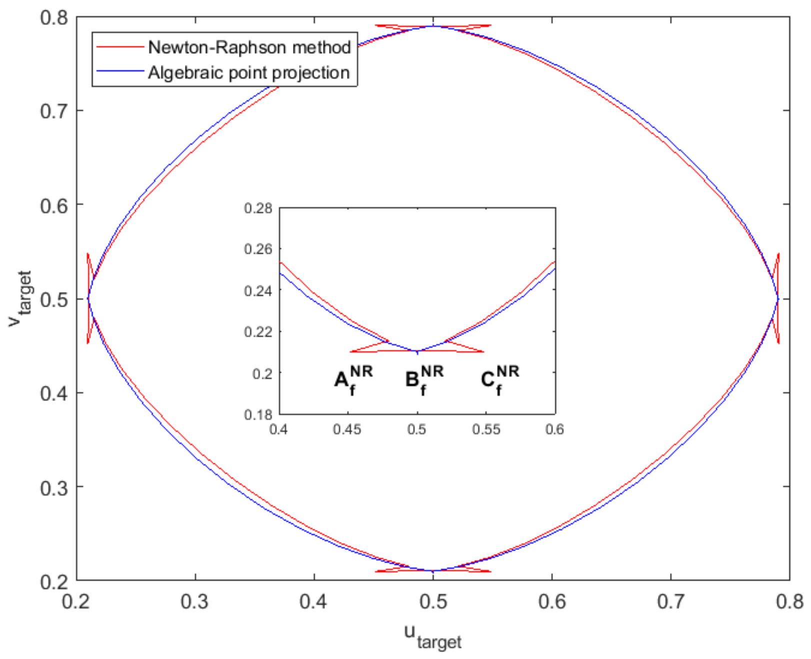


(**b**) Parameter values (u_{target}, v_{target}) of footpoints shown in (**a**).

Figure 19. Illustration of the robustness of the 3D algebraic point projection algorithm involving discontinuous footpoints. (**a**) Trace of points that were projected onto bowl-shaped target surface using the proposed algebraic method and the Netwon-Raphson method. (**b**) Parameters of footpoints on the target obtained by both the methods. Discontinuity occurs due to non-unique footpoints for test points near the bottom of the surface.



(**a**) Trace of test points, mountain-shaped target surface and consequent footpoints.



(**b**) Parameter values (u_{target}, v_{target}) of footpoints.

Figure 20. Illustration of the robustness of the 3D algebraic point projection algorithm involving test points whose mathematical footpoints do not exist. (**a**) Trace of points which are projected onto mountain-shaped target surface using both methods. (**b**) Parameters of footpoints on target. The solution does not exist near the four mountain ridges of $\mathscr{G}^0$ continuity as shown in the four corner regions of (**b**).

Table 4. The results of point projection for NURBS surfaces. The tolerance in Newton–Raphson iterations was chosen as $\epsilon = 10^{-8}$. Note that the time of finding an initial point is excluded in the time per iteration.

Example Surface	Newton–Raphson Iterations			2nd order APP	
	Time per Point (µs)	Average Number of Iterations	Time per Iteration (µs)	Time per Point (µs)	Time per Point for Algorithm 1 (µs)
#1 (Figure 19)	211.91	5.00	36.03	55.28	14.84
#2 (Figure 20)	769.73	10.85	50.47	107.88	16.92

6. Conclusions

In this paper, novel first- and second order-algebraic point projection methods for low degree two-dimensional NURBS curves and three-dimensional NURBS surfaces are proposed. The procedure utilizes the recently developed algebraic level sets. As a first step, the differential property of the resultant matrix is used to obtain the footpoint in the physical space. Next, the parameter value of the footpoint is computed by solving the over-constrained resultant system. Algebraic point projection eliminates inefficient iterative computations and the need for a good initial guess by providing an exact on-curve solution and good approximate solution for points near the curve. Through numerical examples, the algebraic method, especially the second-order point projection is demonstrated to be faster and more smooth than Newton–Raphson iterations.

The proposed algebraic point projection technique possesses two limitations. As illustrated in Section 4, the proposed method is inaccurate or will fail when query points are far away from the curve/surface due to the inaccuracy of the algebraic distance measure. Algebraic point projection will also fail at points where the gradient of the algebraic level-sets is not defined. Such a point occurs, for example, at the center of a circle or sphere. At these locations, a correction to the gradient or a perturbed point is needed. Although algebraic point projection is an approximation, particularly for test points far away from the curve/surface, it has utility for isogeometric analysis, where small inaccuracy in point inversion solution may be acceptable as a trade-off against smooth, robust, and efficient projection. Such a projection is critical given the large number of quadrature points at which point projection is necessary during analysis.

Author Contributions: Conceptualization, T.S. and G.S.; Methodology, H.L., P.K.V. and T.S.; Software, H.L., P.K.V. and T.S.; Validation, H.L., P.K.V. ,T.S. and G.S.; Formal analysis, P.K.V.; Investigation, H.L., P.K.V., T.S. and G.S.; Resources, G.S.; Data curation, H.L., P.K.V., and T.S.; Writing—original draft preparation, H.L., P.K.V., T.S. and G.S.; Writing—review and editing, G.S.; Visualization, H.L., P.K.V. and T.S.; Supervision, G.S.; Project administration, G.S.; Funding acquisition, G.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors wish to thank the reviewers for their suggestions that improved the manuscript.

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

References

1. Mortenson, M. *Geometric Modeling*; John Wiley & Sons: New York, NY, USA, 1985.
2. Piegl, L.; Tiller, W. Parametrization for surface fitting in reverse engineering. *Comput. Aided Des.* **2001**, *33*, 593–603. [CrossRef]
3. Peskin, C.S. Flow patterns around heart valves: a numerical method. *J. Comput. Phys.* **1972**, *10*, 252–271. [CrossRef]
4. Natekar, D.; Zhang, X.; Subbarayan, G. Constructive solid analysis: a hierarchical, geometry-based meshless analysis procedure for integrated design and analysis. *Comput. Aided Des.* **2004**, *36*, 473–486. [CrossRef]
5. Hughes, T.J.; Cottrell, J.A.; Bazilevs, Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 4135–4195. [CrossRef]
6. Tambat, A.; Subbarayan, G. Isogeometric enriched field approximations. *Comput. Methods Appl. Mech. Eng.* **2012**, *245-246*, 1–21. [CrossRef]

7. Wriggers, P.; Zavarise, G. *Computational Contact Mechanics*; John Wiley & Sons: New York, NY, USA, 2004.
8. De Lorenzis, L.; Temizer, I.; Wriggers, P.; Zavarise, G. A large deformation frictional contact formulation using NURBS-based isogeometric analysis. *Int. J. Numer. Methods. Eng.* **2011**, *87*, 1278–1300. [CrossRef]
9. Bazilevs, Y.; Calo, V.M.; Zhang, Y.; Hughes, T.J. Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Comput. Mech.* **2006**, *38*, 310–322. [CrossRef]
10. Tambat, A.; Subbarayan, G. Simulations of arbitrary crack path deflection at a material interface in layered structures. *Eng. Fract. Mech.* **2015**, *141*, 124–139. [CrossRef]
11. Song, T.; Upreti, K.; Subbarayan, G. A sharp interface isogeometric solution to the Stefan problem. *Comput. Methods Appl. Mech. Eng.* **2015**, *284*, 556–582. [CrossRef]
12. Ma, Y.L.; Hewitt, W. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Comput. Aided Geom. Des.* **2003**, *20*, 79–99. [CrossRef]
13. Chen, X.D.; Su, H.; Yong, J.H.; Paul, J.C.; Sun, J.G. A counterexample on point inversion and projection for NURBS curve. *Comput. Aided Geom. Des.* **2007**, *24*, 302. [CrossRef]
14. Selimovic, I. Improved algorithms for the projection of points on NURBS curves and surfaces. *Comput. Aided Geom. Des.* **2006**, *23*, 439–445. [CrossRef]
15. Chen, X.D.; Yong, J.H.; Wang, G.; Paul, J.C.; Xu, G. Computing the minimum distance between a point and a NURBS curve. *Comput. Aided Des.* **2008**, *40*, 1051–1054. [CrossRef]
16. Hoschek, J.; Lasser, D.; Schumaker, L.L. *Fundamentals of Computer Aided Geometric Design*; AK Peters: Natick, MA, USA, 1993.
17. Hartmann, E. On the curvature of curves and surfaces defined by normalforms. *Comput. Aided Geom. Des.* **1999**, *16*, 355–376. [CrossRef]
18. Hu, S.; Sun, J.; Jin, T.; Wang, G. Computing the parameters of points on NURBS curves and surfaces via moving affine frame method. *J. Softw.* **2000**, *11*, 49–53.
19. 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]
20. Liu, X.M.; Yang, L.; Yong, J.H.; Gu, H.J.; Sun, J.G. A torus patch approximation approach for point projection on surfaces. *Comput. Aided Geom. Des.* **2009**, *26*, 593–598. [CrossRef]
21. Li, X.; Wu, Z.; Pan, F.; Liang, J.; Zhang, J.; Hou, L. A Geometric Strategy Algorithm for Orthogonal Projection onto a Parametric Surface. *J. Comput. Sci. Technol.* **2019**, *34*, 1279–1293. [CrossRef]
22. Piegl, L.; Tiller, W. *The NURBS Book*; Springer Science & Business Media: New York, NY, USA, 1997.
23. Upreti, K.; Song, T.; Tambat, A.; Subbarayan, G. Algebraic distance estimations for enriched isogeometric analysis. *Comput. Methods Appl. Mech. Eng.* **2014**, *280*, 28–56. [CrossRef]
24. Upreti, K.; Subbarayan, G. Signed algebraic level sets on NURBS surfaces and implicit Boolean compositions for isogeometric CAD–CAE integration. *Comput. Aided Des.* **2017**, *82*, 112–126. [CrossRef]
25. Sederberg, T. Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design. Ph.D. Thesis, Purdue University, West Lafayette, Indiana, 1983.
26. Biswas, A.; Shapiro, V. Approximate distance fields with non-vanishing gradients. *Graph. Models* **2004**, *66*, 133–159. [CrossRef]
27. Rvachev, V. On the analytical description of some geometric objects. *Rep. Ukr. Acad. Sci.* **1963**, *153*, 765–767.
28. Shapiro, V. *Theory of R-Functions and Applications: A Primer*; Technical Report TR91-1219; Cornell University: Ithaca, NY, USA, 1991.
29. Abhyankar, S. *Algebraic Geometry for Scientists and Engineers*; American Mathematical Society: Providence, RI, USA, 1990; Volume 35.
30. Goldman, R.; Sederberg, T.; Anderson, D. Vector elimination: A technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves. *Comput. Aided Geom. Des.* **1984**, *1*, 327–356. [CrossRef]