

Article

Adaptive Approximate Implicitization of Planar Parametric Curves via Asymmetric Gradient Constraints

Minghao Guo ^{1,*} and Yan Gao ²¹ School of Mathematics and Statistics, Changchun University of Technology, Changchun 130000, China² Beihu Mingda Experimental School, Northeast Normal University, Changchun 130000, China

* Correspondence: mhgao@ccut.edu.cn

Abstract: Converting a parametric curve into the implicit form, which is called implicitization, has always been a popular but challenging problem in geometric modeling and related applications. However, existing methods mostly suffer from the problems of maintaining geometric features and choosing a reasonable implicit degree. The present paper has two contributions. We first introduce a new regularization constraint (called the asymmetric gradient constraint) for both polynomial and non-polynomial curves, which efficiently possesses shape-preserving. We then propose two adaptive algorithms of approximate implicitization for polynomial and non-polynomial curves respectively, which find the “optimal” implicit degree based on the behavior of the asymmetric gradient constraint. More precisely, the idea is gradually increasing the implicit degree, until there is no obvious improvement in the asymmetric gradient loss of the outputs. Experimental results have shown the effectiveness and high quality of our proposed methods.

Keywords: approximate implicitization; parametric curves; machine learning; implicit curve fitting; asymmetric gradient constraint



Citation: Guo, M.; Gao, Y. Adaptive Approximate Implicitization of Planar Parametric Curves via Asymmetric Gradient Constraints. *Symmetry* **2023**, *15*, 1738. <https://doi.org/10.3390/sym15091738>

Academic Editors: José Carlos R. Alcántud, Zhixun Su and Sergei D. Odintsov

Received: 24 May 2023

Revised: 8 August 2023

Accepted: 7 September 2023

Published: 11 September 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In geometric modeling and computer-aided design, the implicit and the parametric form are the two main representations of curves and surfaces. The parametric representations provide a simple way of generating points for displaying curves and surfaces. However, it has to introduce a parametrization of the geometry, which is always a challenging problem. Without requiring any parametrization, the implicit representations offer several advantages, such as the closeness under certain geometric operations (like union, intersection, and blending), the representation ability for describing an object with complicated geometry, and so on. Thus in this paper, we discuss the problem of converting the parametric form of curves into the implicit form, which is called implicitization. Implicitization is also of independent interest, since certain questions in areas as diverse as robotics [1,2]. For instance, in [3], they compute the implicit polynomial equations that describe the constraints of a mechanism by a linear algorithm.

1.1. Related Work

Implicitization has been receiving increased attention in the past few years. Traditional implicitization approaches are based on the elimination theory (such as μ -basis [4], Gröbner bases [5,6], resultants [7] and moving curves and surfaces [8,9]), in which the implicitization problem is solved by the elimination of the parametric variables. However, high polynomial degrees of their outputs not only make this form computationally expensive and numerically unstable, but also cause self-intersections and unwanted branches in most cases.

To alleviate this problem, many approximate implicitization techniques have been proposed, and we follow this line of work. Most of these methods fix the degree of the

objective implicit form, and thus the implicitization problem converts to find a solution in a finite vector space. We group these methods into two categories:

Methods in the first category minimize the algebraic distance from the input parametric curve/surface to the output implicit curve/surface, with a chosen implicit degree. One of the first methods that use this idea is [10], in which the main approximation tool is the singular value decomposition. Later, Ref. [11] discussed theoretical and practical aspects of Dokken's method in [10] under different polynomial basis functions, and proposed a new method for the least squares approach to approximate implicitization using orthogonal polynomials; see Section 2.2 for details. Furthermore, piecewise approximate implicitization with prescribed interpolating conditions using tensor-product B-splines was studied by [12]. Recently, Ref. [13] proposed a method to determine primitive shapes of geometric models by combining clustering analysis with Dokken's method. In that paper, the implicit degree of curve/surface patches was determined by checking whether the smallest singular value is less than a certain threshold, where the threshold was inferred using a straightforward statistical approach.

The second category is the fitting-based methods, which is also called discrete approximate implicitization. These methods minimize the squared algebraic distances of a set of points sampled from the given parametric curves and surfaces. Ref. [14] proposed an approach, that is, to approximate the set of sampling points with MQ quasi-interpolation in order to possess shape preserving and then to approximate the error function by using RBF networks. Ref. [15] developed an autoencoder-based fitting method, which is to put sampling points into an encoder to obtain polynomial coefficients and then put them into a decoder to output the predicted function value. Ref. [16] described an algorithm for approximating sampling points and associated normal vectors simultaneously, in the context of fitting with implicitly defined algebraic spline curves. In that paper, the coefficients of the output function f are obtained as the minimum of

$$\sum_{j=1}^N (f(\mathbf{p}_j))^2 + \lambda \sum_{j=1}^N \|\nabla f(\mathbf{p}_j) - \mathbf{n}_j\|^2 + \text{"tension"},$$

where $\{\mathbf{p}_j\}_{j=1}^N$ are sampled point data, $\{\mathbf{n}_j\}_{j=1}^N$ are unit normals at these points, and λ is the regulator gain. The first term represents the (algebraic) distance of the point data from the implicit curve $f = 0$. The second term, which is called the **symmetric gradient constraint** by us in this paper, controls the influence of the normal vectors $\mathbf{n}_j$ to the resulting curve. However, the symmetric gradient constraint is too strict to find output curves of low degrees, since it requires that the gradient vector $\nabla f(\mathbf{p}_j)$ and the normal vector $\mathbf{n}_j$ have the same direction and magnitude simultaneously in any point $\mathbf{p}_j$. The third term "tension" is added to pull the approximating curve towards a simpler shape. Afterward, the idea of [16] has been generalized to the algebraic spline surface case in [17], the rotational surface case in [18], and the space curve case in [19].

A related approach has been proposed by [20], where an alternative gradient constraint

$$1 - \frac{1}{N} \sum_{j=1}^N \frac{\nabla f(\mathbf{p}_j)}{\|\nabla f(\mathbf{p}_j)\|} \cdot \mathbf{n}_j$$

is defined to make the normalized gradient vector $\frac{\nabla f(\mathbf{p}_j)}{\|\nabla f(\mathbf{p}_j)\|}$ close to the unit normal vector $\mathbf{n}_j$ in any point $\mathbf{p}_j$. More precisely, this gradient constraint requires that the mean of angles of $\frac{\nabla f(\mathbf{p}_j)}{\|\nabla f(\mathbf{p}_j)\|}$ and $\mathbf{n}_j$ equals to zero. This constraint is used subsequently in their algorithm to find the "optimal" degree of the implicit polynomial needed for the representation of the data set. However, the normalization of both gradient and normal vectors leads to multimodal functional dependencies of the solution from data, and computationally

expensive metaheuristic algorithms have to be used to realize a better exploration of the search space.

1.2. Contributions of the Paper

In this paper, we attempt to recover the approximate implicitization of the planar parametric curve adaptively. The contributions of our work are summarized as follows:

- As we know, for a parametric polynomial (or rational) curve of degree n , the exact implicit representation has a total degree less than or equal to n . Thus, if we approximate with degrees higher than n , some constraints have to be added to avoid bad behavior such as extra branches or near-singular areas.
To tackle these challenges, we introduce the so-called **asymmetric gradient constraint**, which bends the direction of the implicit curve closer to that of the parametric curve, even for the cases of implicit degrees higher than n . Moreover, compared to the symmetric gradient constraint in [16], our new regularization constraint enlarges the solution space.
- We perform our objective function of approximate implicitization into the quadratic form, so that the eigenvalue/eigenvector method can be used to find the minimum more rapidly than the PSO (particle swarm optimization) algorithm in [20].
- We develop an adaptive implicitization algorithm, which is to find an implicit polynomial that produces a compact and smooth representation of the input curve with the lowest degree as possible, and at the same time minimizes the implicitization error.

The remainder of the paper is organized as follows. We state the problem and present a synopsis of Dokken's method for approximate implicitization in Section 2. Section 3 introduces our implicitization method (AGM) for polynomial curves and shows our numerical results. In Section 4, the AGM is extended to non-polynomial curves. Section 5 finalizes the paper with a conclusion and some possible directions for future work.

2. Background

2.1. Problem Formulation

A parametric polynomial curve of degree m in $\mathbb{R}^2$ is given by

$$\mathbf{p}(t) = \begin{pmatrix} p_1(t) \\ p_2(t) \end{pmatrix}, t \in [a, b],$$

where p_1 and p_2 are polynomials in t . An implicit (algebraic) curve of degree n in $\mathbb{R}^2$, is defined by the zero contours of a bivariate polynomial

$$\begin{aligned} f_{\mathbf{b}}(x, y) &= \sum_{i=1}^k b_i \phi_i(x, y) \\ &= (\phi_1(x, y), \phi_2(x, y), \dots, \phi_k(x, y)) \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix}, \end{aligned}$$

where $\{\phi_i(x, y)\}_{i=1}^k$ generates a basis for bivariate polynomials of total degree n , $k = \binom{n+2}{2}$ denotes the number of basis functions, $\mathbf{b} = (b_1, b_2, \dots, b_k)^\top$ is the coefficient vector of $f_{\mathbf{b}}(x, y)$.

An exact implicitization of $\mathbf{p}(t)$ is a non-zero $f_{\mathbf{b}}(x, y)$, such that the squared algebraic distance (AD for short) from $\mathbf{p}(t)$ to the implicit curve $f_{\mathbf{b}}(x, y) = 0$ equals to zero, i.e.,

$$L_{AD} = \int_a^b [f_{\mathbf{b}}(\mathbf{p}(t))]^2 dt = 0.$$

However, as stated in the introduction, in practice, one may prefer to work with lower degrees. Thus in this paper, we consider the approximate implicitization problem, which is to seek the “optimal” $f_{\mathbf{b}}(x, y)$ with a lower degree n , that minimizes the squared AD constraint L_{AD} under some additional criterion to be specified.

2.2. Dokken’s (Weak) Method

In this subsection, we give a brief description of Dokken’s (weak) method (DM for short). Notice that the expression $f_{\mathbf{b}}(\mathbf{p}(t))$ is a univariate polynomial of degree mn in t , Dokken finds that $f_{\mathbf{b}}(\mathbf{p}(t))$ can be factorized as

$$f_{\mathbf{b}}(\mathbf{p}(t)) = (\alpha(t))^{\top} D_1 \mathbf{b},$$

where

- $\mathbf{b}$ is the unknown coefficient vector of $f_{\mathbf{b}}(x, y)$;
- $\alpha(t) = (\alpha_1(t), \alpha_2(t), \dots, \alpha_{mn+1}(t))^{\top}$ is the basis of the space of univariate polynomials of degree mn ; and
- D_1 is the collocation matrix whose columns are the coefficients of $\phi_i(\mathbf{p}(t))$ expressed in the $\alpha(t)$ -basis.

Lemma 1 ([11]). *Let*

$$G_{\alpha} = \int_a^b \alpha(t)(\alpha(t))^{\top} dt \quad (1)$$

denote the Gram matrix of the basis $\alpha(t)$. Then, the squared AD of $\mathbf{p}(t)$ from $f_{\mathbf{b}}(x, y) = 0$ can be written as

$$L_{AD} = \int_a^b [f_{\mathbf{b}}(\mathbf{p}(t))]^2 dt = \mathbf{b}^{\top} A_1 \mathbf{b}, \quad (2)$$

where

$$A_1 = D_1^{\top} G_{\alpha} D_1$$

is a positive semidefinite matrix.

Lemma 1 shows that L_{AD} is a homogeneous quadratic form of $\mathbf{b}$. In order to avoid the null vector $\mathbf{b} = 0$, Dokken introduces the normalization $\|\mathbf{b}\| = 1$. Denote by $\mathbf{b}_{DM}$ the unit eigenvector corresponding to the smallest eigenvalue of A_1 , then $\mathbf{b}_{DM}$ is the solution of Dokken’s method for minimizing L_{AD} subject to $\|\mathbf{b}\| = 1$.

3. Methodology for Polynomial Curves

In this section, we provide the approximate implicitization methodology for polynomial curves. First, we propose the asymmetric gradient constraint to keep the gradient vector of $f_{\mathbf{b}}(x, y) = 0$ and the tangent vector of $\mathbf{p}(t)$ being perpendicular. Then, we represent the objective function in the matrix form. Finally, we propose the adaptive implicitization algorithm to compute the “optimal” implicitization $f_{\mathbf{b}}(x, y)$ and perform some experiments to show the validity of the algorithm.

3.1. Distance Constraint

We use the squared AD constraint (AD loss for short) in Equation (2):

$$L_{AD} = \mathbf{b}^{\top} A_1 \mathbf{b}.$$

3.2. Asymmetric Gradient Constraint

To obtain a non-trivial solution, the implicitization problem must be regularized by restricting $f_{\mathbf{b}}$ to some specified class of functions. One reasonable approach is to require that this be the class of “shape-preserving” functions. We present the so-called asymmetric gradient constraint (AG loss for short):

$$L_{AG} = \|\nabla f_{\mathbf{b}} \cdot \mathbf{p}'\| = \int_a^b [\nabla f_{\mathbf{b}}(\mathbf{p}(t)) \cdot \mathbf{p}'(t)]^2 dt, \quad (3)$$

where

- $\nabla f_{\mathbf{b}}(\mathbf{p}(t))$ is the gradient vector of the implicit curve at the point $\mathbf{p}(t)$;
- $\mathbf{p}'(t)$ is the tangent vector of the parametric curve at the point $\mathbf{p}(t)$; and
- The inner product is

$$\nabla f_{\mathbf{b}}(\mathbf{p}(t)) \cdot \mathbf{p}'(t) = \|\nabla f_{\mathbf{b}}(\mathbf{p}(t))\| \|\mathbf{p}'(t)\| \cos \theta,$$

where θ denotes the angle of $\nabla f_{\mathbf{b}}(\mathbf{p}(t))$ and $\mathbf{p}'(t)$ at the point $\mathbf{p}(t)$.

Compared to the symmetric gradient constraint in [16], Our AG constraint only requires that the gradient vector of the implicit curve and the normal vector of the parametric curve have the same direction at any point. Intuitively speaking, the AG constraint bends the direction of the implicit curve closer to the parametric curve's direction. If the inner product $\nabla f_{\mathbf{b}}(\mathbf{p}(t)) \cdot \mathbf{p}'(t)$ equals 0, then the tangents of the implicit and parametric curve are exactly parallel. The smaller the inner product is, the more similar are the appearance of them.

Theorem 1. *The AG constraint L_{AG} in Equation (3) can be written in a homogeneous quadratic form of $\mathbf{b}$ using the basis $\alpha(t)$.*

Proof. We can perform the AG constraint as follows. Since the expressions $\mathbf{p}'(t)$ and $\nabla f_{\mathbf{b}}(\mathbf{p}(t))$ are polynomial vectors of degree $m-1$ and $(n-1)m$ in t , respectively, their inner product $\nabla f_{\mathbf{b}}(\mathbf{p}(t)) \cdot \mathbf{p}'(t)$ is a polynomial of degree $nm-1$ in t . Thus, it also can be written as a linear combination of $\alpha(t)$, which is the basis for univariate polynomials of degree mn . Every coefficient of this linear combination is a linear expression of $\mathbf{b}$. As a result, the inner product can be factored into

$$\begin{aligned} \nabla f_{\mathbf{b}}(\mathbf{p}(t)) \cdot \mathbf{p}'(t) &= (\alpha(t))^\top \begin{pmatrix} \mathbf{b}'\text{'s linear expression} \\ \vdots \\ \mathbf{b}'\text{'s linear expression} \end{pmatrix} \\ &= (\alpha(t))^\top D_2 \mathbf{b}, \end{aligned}$$

where D_2 is the collocation matrix whose rows are the coefficients of “ $\mathbf{b}$'s linear expressions” expressed in $\mathbf{b}$. Finally, the AG constraint can be written as

$$L_{AG} = \int_a^b [\nabla f_{\mathbf{b}}(\mathbf{p}(t)) \cdot \mathbf{p}'(t)]^2 dt = \mathbf{b}^\top A_2 \mathbf{b}, \quad (4)$$

where

$$A_2 = D_2^\top G_\alpha D_2$$

is a positive semidefinite matrix and G_α is the Gram matrix of the basis $\alpha(t)$ in Equation (1). $\square$

3.3. Putting Things Together

Summing up, due to Equations (2) and (4), the approximate implicitization is found by minimizing the positive semidefinite quadratic objective function

$$L_{\lambda,n}(\mathbf{b}) = L_{AD} + \lambda L_{AG} = \mathbf{b}^\top (A_1 + \lambda A_2) \mathbf{b} \quad (5)$$

over the coefficients $\mathbf{b}$ of $f_{\mathbf{b}}(x, y)$, while keeping the degree n of $f_{\mathbf{b}}(x, y)$ fixed. The first term in Equation (5) measures the fidelity of the implicit curve to the given parametric curve, and the second term in Equation (5) tries to maintain geometric features that the implicit curve must have. The trade-off between these requirements is controlled by $\lambda > 0$, called the regulator gain.

Similar to Dokken's method, denote by $\mathbf{b}_{AGM}$ the unit eigenvector corresponding to the smallest eigenvalue of the symmetric matrix

$$A = A_1 + \lambda A_2,$$

then $\mathbf{b}_{AGM}$ is the solution for minimizing Equation (5) subject to $\|\mathbf{b}\| = 1$.

3.4. Adaptive Implicitization Algorithm

The adaptive implicitization is to obtain the "optimal" degree n_{op} for the implicit polynomial $f_{\mathbf{b}}$, where $1 \leq n_{op} \leq n_{\max}$. We estimate n_{op} via the behavior of the AG constraint as the implicit degree n increases. We have performed lots of experiments on examining the change trend of the AG loss in Equation (4), and have found that the change usually goes through three stages as n increases:

- First, the AG loss drops significantly (i.e., under-fitting).
- Second, the AG loss reaches the minimum, and then changes very slightly (i.e., good fitting).
- Third, the AG loss increases conversely (i.e., over-fitting).

Thus, we introduce two thresholds for the stopping criterion:

- ϵ_{AD} : to examine whether the AD loss in Equation (2) satisfies our default precision;
- ϵ_{AG} : to check the monotonicity of the AG loss in Equation (4) to avoid over-fitting.

The adaptive implicitization methodology for polynomial curves, called the asymmetric gradient method (AGM for short), is summarized in Algorithm 1. The inputs of Algorithm 1 are the parametric curve $\mathbf{p}(t)$, the maximum implicit degree $n_{\max}$, and thresholds $\epsilon_{AD}, \epsilon_{AG}$. Line 1 is employed to initialize the implicit degree n . In the whole loop (line 2 to line 20), the matrix $A^{(n)}$ in the objective function $L_{\lambda,n}(\mathbf{b})$ is computed first using the collocation matrix $D_1^{(n)}, D_2^{(n)}$ and the Gram matrix $G_{\alpha}^{(n)}$; then for the n th (current) cycle, the "optimal" coefficient vector $b^{(n)}$ is found, the AD loss $e_1^{(n)}$ and the AG loss $e_2^{(n)}$ are computed subsequently; if $n = n_{\max}$, which means that the coefficient vectors $b^{(1)}, \dots, b^{(n-1)}$ obtained in previous cycles are unacceptable, then $b^{(n)}$ is treated as the final result and the algorithm is terminated under this circumstance (line 10 to line 12); if $n < n_{\max}$ and $e_1^{(n)}, e_2^{(n)}$ satisfy the stopping criteria (line 14), then return $b^{(n)}$ as the final result and terminate the algorithm; if none of the aforesaid If statements holds, then let n increase by one and go to the next cycle.

3.5. Experiments

Additional branches (i.e., the extra zero contour) generated in the implicitization procedure make the resulting curves challenging to be interpreted, and the elimination of additional branches is the main problem in designing the implicitization methods. Ref. [21] addresses this problem by combining two or more eigenvectors (associated with small eigenvalues), which leads to a gradual decline in accuracy. With the AGM developed in this paper, additional branches can be avoided as much as possible in the implicitization procedure.

We choose the basis $\alpha(t)$ to be the univariate Bernstein polynomial basis of degree $p = mn + 1$, i.e.,

$$\alpha_i(t) = B_{i,p}(t) = \frac{p!}{i!(p-i)!} t^i (1-t)^{p-i}, i = 0, 1, \dots, p,$$

and the basis $\{\phi_i(x, y)\}_{i=1}^k$ to be the bivariate monomial basis of total degree n . We set the maximum implicit degree $n_{\max} = 7$, the regulator gain $\lambda = 0.1$, and the thresholds $\epsilon_{AD} = 10^{-4}$ and $\epsilon_{AG} = 10^{-3}$.

Algorithm 1: AGM for polynomial curves.

Data: The polynomial curve $\mathbf{p}(t)$; the maximum implicit degree $n_{\max}$; the stopping thresholds $\epsilon_{AD}, \epsilon_{AG}$.

Result: The coefficient vector $\mathbf{b}_{AGM}$ of the implicit polynomial $f_{\mathbf{b}}$.

```

1  $n \leftarrow 1$ ;
2 while  $n \leq n_{\max}$  do
3   Construct the collocation matrix  $D_1^{(n)}$  of the AD constraint;
4   Construct the collocation matrix  $D_2^{(n)}$  of the AG constraint;
5   Compute the Gram matrix  $G_{\alpha}^{(n)}$ ;
6    $A_1^{(n)}, A_2^{(n)} \leftarrow (D_1^{(n)})^{\top} G_{\alpha}^{(n)} D_1^{(n)}, (D_2^{(n)})^{\top} G_{\alpha}^{(n)} D_2^{(n)}$ ;
7    $A^{(n)} \leftarrow A_1^{(n)} + \lambda A_2^{(n)}$ ;
8    $\mathbf{b}^{(n)} \leftarrow$  the unit eigenvector corresponding to the smallest eigenvalue of  $A^{(n)}$ ;
9    $e_1^{(n)}, e_2^{(n)} \leftarrow (\mathbf{b}^{(n)})^{\top} A_1^{(n)} \mathbf{b}^{(n)}, (\mathbf{b}^{(n)})^{\top} A_2^{(n)} \mathbf{b}^{(n)}$ ;
10  if  $n = n_{\max}$  then
11     $\mathbf{b}_{AGM} \leftarrow \mathbf{b}^{(n)}$ ;
12    return
13  else
14    if  $e_1^{(n)} \leq \epsilon_{AD}$  and  $|e_2^{(n)} - e_2^{(n-1)}| \leq \epsilon_{AG}$  then
15       $\mathbf{b}_{AGM} \leftarrow \mathbf{b}^{(n)}$ ;
16      return
17    end
18  end
19   $n \leftarrow n + 1$ ;
20 end

```

Example 1. Consider the polynomial parametric curve

$$C_1(t) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} B_{0,3}(t) + \begin{pmatrix} 2 \\ 1 \end{pmatrix} B_{1,3}(t) + \begin{pmatrix} 0 \\ 2 \end{pmatrix} B_{2,3}(t) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} B_{3,3}(t),$$

where the parameter of $C_1(t)$ takes value in $[0, 1]$, and $\{B_{i,3}(t)\}_{i=0}^3$ is the Bernstein polynomial basis of degree 3. $C_1(t)$ is shown in Figure 1.

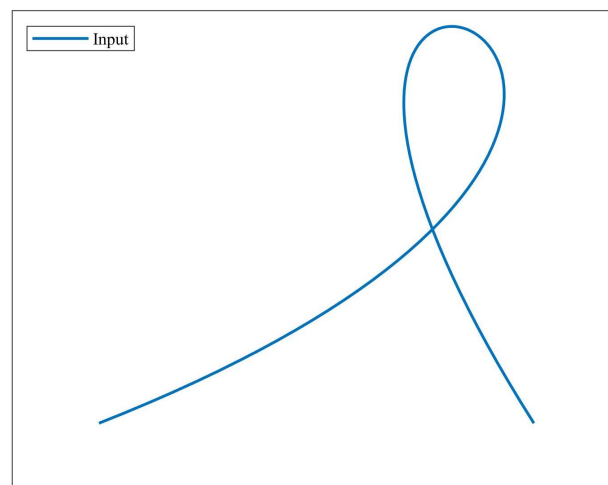


Figure 1. The input curve $C_1(t)$.

The first row and second row of Figure 2 show the adaptive implicitization process of $C_1(t)$ by AGM and DM, respectively. We notice that the second-order ($n = 2$) outputs of AGM and DM both give rather poor fits to the input curve $C_1(t)$. The third-order ($n = 3$) outputs of the two methods both seem to give the best fit. When we go to a much higher-order polynomial ($n = 5$), our AGM obtains an excellent fit, which refrains from additional branches as much as possible, see (c) vs (f) in Figure 2.

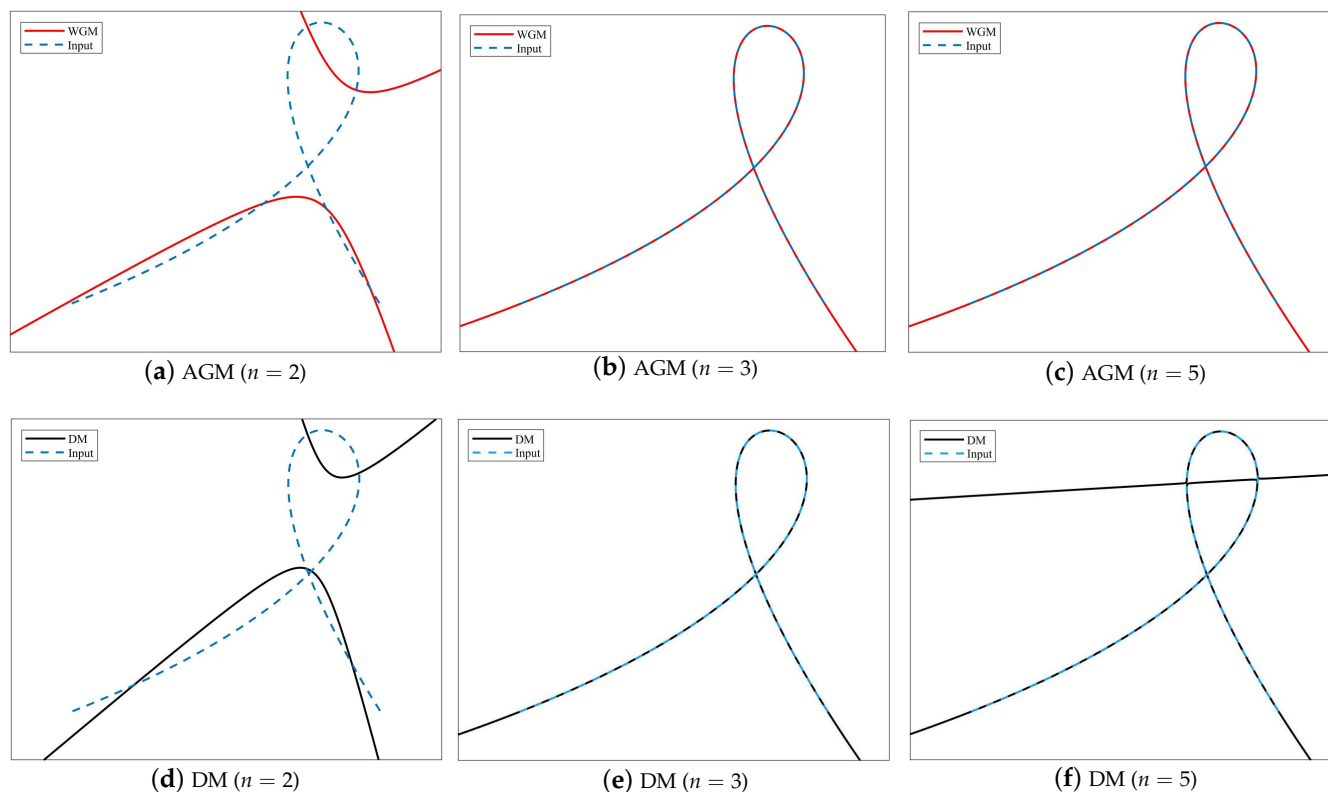


Figure 2. Adaptive implicitization of $C_1(t)$. The blue dash line in (a–f) is the input curve, the red line in (a–c) is the output curve by our method, and the black line in (d–f) is the output curve by DM. From left to right: the implicit degree $n = 2, 3, 5$.

Figure 3 shows the statistic of our method on changes of the AD and AG loss for $C_1(t)$, when the implicit degree n is increasing. We note that the second order output ($n = 2$) gives relatively large values of the AD and AG loss, and this can be attributed to the fact that the corresponding output is rather inflexible and is incapable of capturing the oscillations in the input curve $C_1(t)$. The third and fourth orders ($n = 3, 4$) both give small values for the above two losses, and these also give reasonable implicit representations, as can be seen from Figure 2. the case of $n = 5$ is an over-fitting since the AD and AG losses become relatively large again, and it should be avoided in practice.

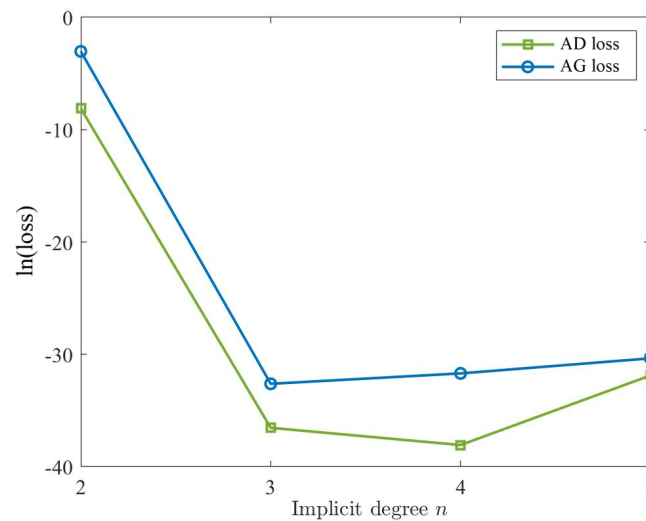


Figure 3. Statistics of our method on changes of the AD and AG loss for $C_1(t)$, as the implicit degree n increases.

Example 2. Consider the polynomial parametric curve

$$C_2(t) = \binom{1}{5} B_{0,4}(t) + \binom{-3}{-15} B_{1,4}(t) + \binom{2}{20} B_{2,4}(t) + \binom{11}{-5} B_{3,4}(t) + \binom{1}{5} B_{4,4}(t),$$

where the parameter of $C_2(t)$ take values in $[0, 1]$, and $\{B_{i,4}(t)\}_{i=0}^4$ is the Bernstein polynomial basis of degree 4. $C_2(t)$ is shown in Figure 4.

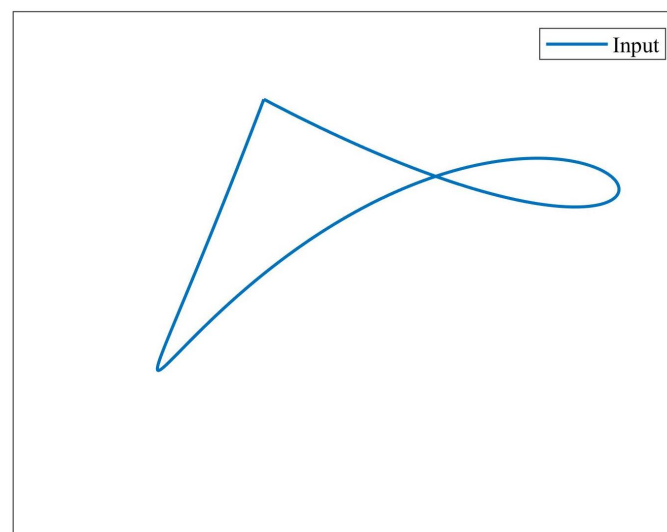


Figure 4. The input curve $C_2(t)$.

The first row and second row of Figure 5 show the adaptive implicitization process of $C_2(t)$ by AGM and DM, respectively. Generally speaking, these two methods have similar outputs as the implicit degree n varies. We notice that both outputs for $n = 2, 3$ give rather poor fits to the input curve $C_2(t)$, and the ones for $n = 4, 5$ seems to give the “nearly” best fit. When we go to a much higher-order polynomial ($n = 6$), both outputs produce unwanted branches. However, in the viewpoint of “shape-preserving”, we can see from Figure 5 that for every iteration of the implicit degree n , the shapes of AGM’s outputs approach $C_2(t)$ much closer than that of DM’s.

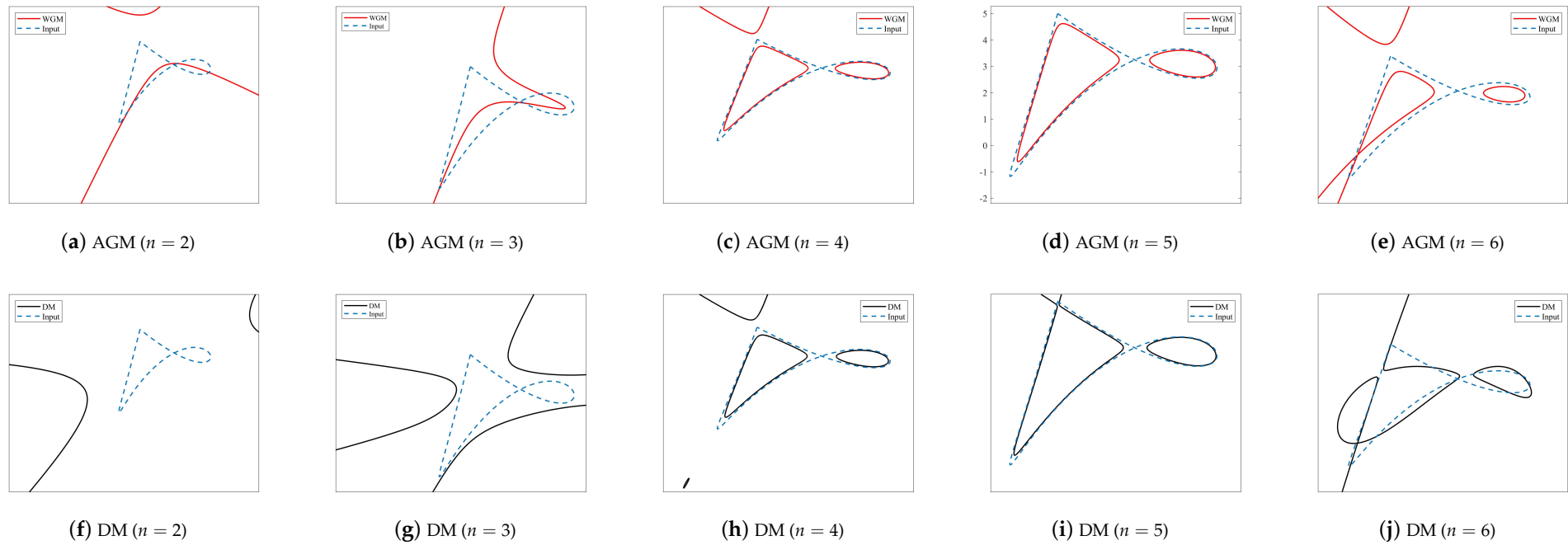


Figure 5. Adaptive implicitization of $C_2(t)$. The blue dash line in (a–j) is the input curve, the red line in (a–e) is the output curve by AGM, and the black line in (f–i) is the output curve by DM. From left to right: the implicit degree $n = 2, 3, 4, 5, 6$.

Figure 6 shows the statistic of our method on changes of the AD and AG loss for $C_2(t)$, when the implicit degree n is increasing. We note that the second and third orders ($n = 2, 3$) give relatively large values of the AD and AG loss, and this can be attributed to the fact that the corresponding output is rather inflexible and is incapable of capturing the oscillations in the input curve $C_2(t)$. The fourth and fifth orders ($n = 4, 5$) both give small values for the above two losses, and these also give reasonable implicit representations, as can be seen from Figure 5. The case of $n = 6$ is an over-fitting since the AD and AG losses become relatively large again.

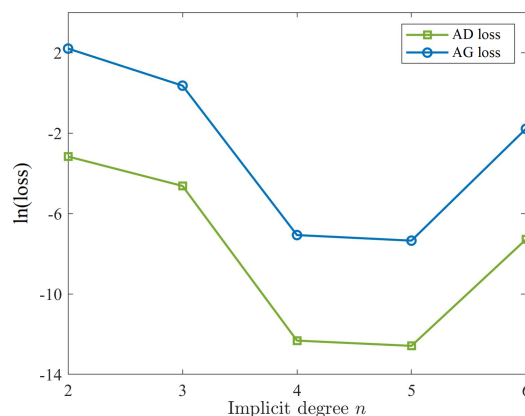


Figure 6. Statistics of our method on changes of the AD and AG loss for $C_2(t)$, as the implicit degree n increases.

Finally, the evaluation results of three methods (AGM, DM, and the method in [20]) on the input curves $C_1(t)$, $C_2(t)$ are presented in Table 1. Table 1 shows that the proposed AGM achieved enough precision. In addition, AGM has the second-lowest inference time after DM, and is significantly faster than the method in [20].

Table 1. Performance of AGM on $C_1(t)$, $C_2(t)$ (all timings are measured in seconds).

Input	AD Loss	AG Loss	AGM Time	DM Time	Method in [20] Time
$C_1(t)$	2.312×10^{-16}	1.266×10^{-14}	0.0019	0.0016	48.71
$C_2(t)$	3.727×10^{-6}	9.119×10^{-4}	0.0030	0.0023	128.81

4. Methodology for Non-Polynomial Curves

To deal with non-polynomial curves, our approach in this section is to sample several of points with associated oriented tangent vectors, and then convert them into implicit polynomials by discrete approximate implicitization (i.e., implicit fitting).

4.1. Curve Sampling

Here, we employ the uniform sampling, which makes the sample points distribute uniformly in the parametric space. All the sampling points and their associated tangent vectors are, respectively, denoted by $\{\mathbf{p}_j\}_{j=1}^N$ and $\{\mathbf{T}_j\}_{j=1}^N$.

4.2. Discrete Approximate Implicitization

Discrete approximate implicitization, also called implicit fitting, is to retrieve the implicit polynomial

$$f_{\mathbf{b}}(x, y) = \sum_{i=1}^k b_i \phi_i(x, y)$$

from the sampling points $\{\mathbf{p}_i\}_{i=1}^N$ by minimizing the sum of the squared algebraic distances (AD loss for short):

$$L_{AD} = \sum_{j=1}^N (f_{\mathbf{b}}(\mathbf{p}_j))^2 = \left\| (f_{\mathbf{b}}(\mathbf{p}_1), f_{\mathbf{b}}(\mathbf{p}_2), \dots, f_{\mathbf{b}}(\mathbf{p}_N))^{\top} \right\|^2. \quad (6)$$

We simplify L_{AD} into the matrix form. While

$$f_{\mathbf{b}}(\mathbf{p}_j) = (\phi_1(\mathbf{p}_j), \phi_2(\mathbf{p}_j), \dots, \phi_k(\mathbf{p}_j)) \mathbf{b}, \quad j = 1, 2, \dots, N,$$

we have

$$(f_{\mathbf{b}}(\mathbf{p}_1), f_{\mathbf{b}}(\mathbf{p}_2), \dots, f_{\mathbf{b}}(\mathbf{p}_N))^{\top} = D_1 \mathbf{b},$$

where $D_1 = (\phi_i(\mathbf{p}_j))_{N \times k}$. Then, L_{AD} in Equation (6) can be written as

$$L_{AD} = \mathbf{b}^{\top} A_1 \mathbf{b}, \quad (7)$$

where

$$A_1 = D_1^{\top} D_1.$$

To avoid the trivial $\mathbf{b} = 0$ for L_{AD} 's minimization, we introduce the asymmetric gradient constraint (AG loss for short) for the discrete case:

$$\begin{aligned} L_{AG} &= \sum_{j=1}^N (\nabla f_{\mathbf{b}}(\mathbf{p}_j) \cdot \mathbf{T}_j)^2 \\ &= \left\| (\nabla f_{\mathbf{b}}(\mathbf{p}_1) \cdot \mathbf{T}_1, \nabla f_{\mathbf{b}}(\mathbf{p}_2) \cdot \mathbf{T}_2, \dots, \nabla f_{\mathbf{b}}(\mathbf{p}_N) \cdot \mathbf{T}_N)^{\top} \right\|^2. \end{aligned}$$

where $\nabla f_{\mathbf{b}}(\mathbf{p}_j)$ is the gradient vector of $f_{\mathbf{b}}(x, y)$ in any point $\mathbf{p}_j$. The role of L_{AG} is to keep $\mathbf{p}_j$'s tangent vector and the gradient vector of the implicit polynomial $f_{\mathbf{b}}$ being perpendicular.

Theorem 2. The asymmetric gradient constraint L_{AG} can be written in a homogeneous quadratic form of $\mathbf{b}$.

Proof. Notice that each inner product $\nabla f_{\mathbf{b}}(\mathbf{p}_j) \cdot \mathbf{T}_j$ can be represent as a linear combination of $\mathbf{b}$, $j = 1, 2, \dots, N$. Then

$$(\nabla f_{\mathbf{b}}(\mathbf{p}_1) \cdot \mathbf{T}_1, \nabla f_{\mathbf{b}}(\mathbf{p}_2) \cdot \mathbf{T}_2, \dots, \nabla f_{\mathbf{b}}(\mathbf{p}_N) \cdot \mathbf{T}_N)^{\top} = D_2 \mathbf{b},$$

where D_2 is the collocation matrix, whose rows are the coefficients of $\nabla f_{\mathbf{b}}(\mathbf{p}_j) \cdot \mathbf{T}_j$'s linear expressions. Afterwards, L_{AG} can be written in the matrix notation as

$$L_{AG} = \mathbf{b}^{\top} A_2 \mathbf{b}, \quad (8)$$

where

$$A_2 = D_2^{\top} D_2.$$

□

Finally, due to Equations (7) and (8), the discrete approximate implicitization is found by minimizing the positive semidefinite quadratic objective function

$$L_{\lambda, n}(\mathbf{b}) = L_{AD} + \lambda L_{AG} = \mathbf{b}^{\top} (A_1 + \lambda A_2) \mathbf{b} \quad (9)$$

over the coefficients $\mathbf{b}$ of $f_{\mathbf{b}}(x, y)$, while keeping the degree n of $f_{\mathbf{b}}(x, y)$ fixed. Denote by $\mathbf{b}_{AGM}$ the unit eigenvector corresponding to the smallest eigenvalue of the symmetric matrix

$$A = A_1 + \lambda A_2,$$

then $\mathbf{b}_{AGM}$ is the solution for minimizing $L_{\lambda,n}(\mathbf{b})$ subject to $\|\mathbf{b}\| = 1$.

4.3. Adaptive Implicitization Algorithm

The asymmetric gradient method (AGM for short) of adaptive implicitization for non-polynomial curves is summarized in Algorithm 2. Algorithm 2 is almost identical to Algorithm 1, with three changes:

- The curve sampling is employed in the first place (line 1);
- The AD and AG matrices $A_1^{(n)}, A_2^{(n)}$ are constructed without the Gram matrix G_{α} (line 6);
- Since the changing trend of the AG's loss for non-polynomial curves is more subtle (see the last Figure for example), the stopping criterion of ϵ_{AG} is replaced by $e_2^{(n)} \leq \epsilon_{AG}$ (line 14), to check whether the AG loss satisfies our default precision.

Algorithm 2: AGM for non-polynomial curves.

Data: The parametric curve $\mathbf{p}(t)$; the maximum implicit degree $n_{\max}$; the stopping thresholds $\epsilon_{AD}, \epsilon_{AG}$.
Result: The coefficient vector $\mathbf{b}_{AGM}$ of the implicit polynomial $f_{\mathbf{b}}$.

```

1 Uniform sampling on  $\mathbf{p}(t)$ ;
2  $n \leftarrow 1$ ;
3 while  $n \leq n_{\max}$  do
4   Construct the collocation matrix  $D_1^{(n)}$  of the AD constraint;
5   Construct the collocation matrix  $D_2^{(n)}$  of the AG constraint;
6    $A_1^{(n)}, A_2^{(n)} \leftarrow (D_1^{(n)})^{\top} D_1^{(n)}, (D_2^{(n)})^{\top} D_2^{(n)}$ ;
7    $A^{(n)} \leftarrow A_1^{(n)} + \lambda A_2^{(n)}$ ;
8    $\mathbf{b}^{(n)} \leftarrow$  the unit eigenvector corresponding to the smallest eigenvalue of  $A^{(n)}$ ;
9    $e_1^{(n)}, e_2^{(n)} \leftarrow (\mathbf{b}^{(n)})^{\top} A_1^{(n)} \mathbf{b}^{(n)}, (\mathbf{b}^{(n)})^{\top} A_2^{(n)} \mathbf{b}^{(n)}$ ;
10  if  $n = n_{\max}$  then
11     $\mathbf{b}_{AGM} \leftarrow \mathbf{b}^{(n)}$ ;
12    return
13  else
14    if  $e_1^{(n)} \leq \epsilon_{AD}$  and  $e_2^{(n)} \leq \epsilon_{AG}$  then
15       $\mathbf{b}_{AGM} \leftarrow \mathbf{b}^{(n)}$ ;
16      return
17    end
18  end
19   $n \leftarrow n + 1$ ;
20 end
```

4.4. Experiments

We choose the basis $\{\phi_i(x, y)\}_{i=1}^k$ to be the bivariate monomial basis of total degree n . We set the maximum implicit degree $n_{\max} = 7$, the regulator gain $\lambda = 0.01$, and the thresholds $\epsilon_{AD} = 10^{-2}$ and $\epsilon_{AG} = 10^{-1}$.

Example 3. Consider the non-polynomial parametric curve

$$C_3(t) = \begin{pmatrix} 2(1 + \cos t) \cos t \\ 2(1 + \cos t) \sin t \end{pmatrix},$$

where the parameter of $C_3(t)$ take values in $[0, 10]$. $C_3(t)$ is shown in Figure 7.

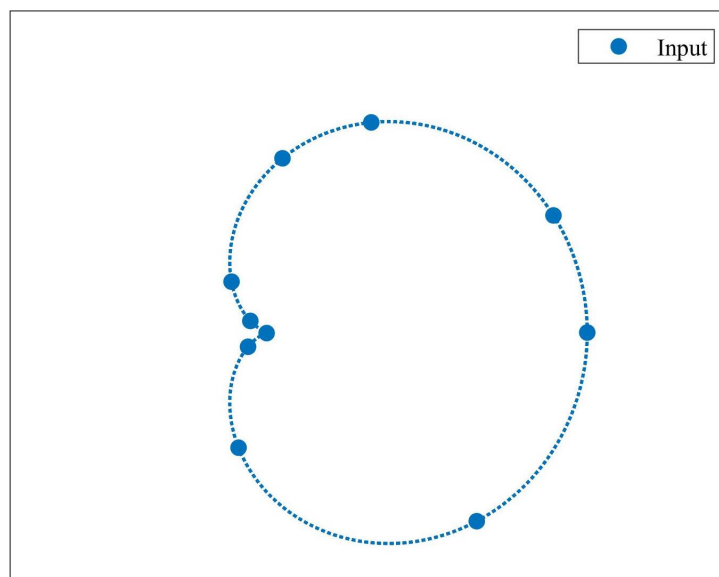


Figure 7. The input curve $C_3(t)$ and 10 points are sampled uniformly from $C_3(t)$.

The first row and second row of Figure 8 show the adaptive implicitization process of $C_3(t)$ by AGM and DM, respectively. We notice that the third-order ($n = 3$) outputs of AGM and DM both give rather poor fits to the input curve $C_3(t)$. When we go to a much higher-order polynomial ($n = 4, 5$), our AGM obtains a much more excellent fit than DM, which refrains from additional branches as much as possible.

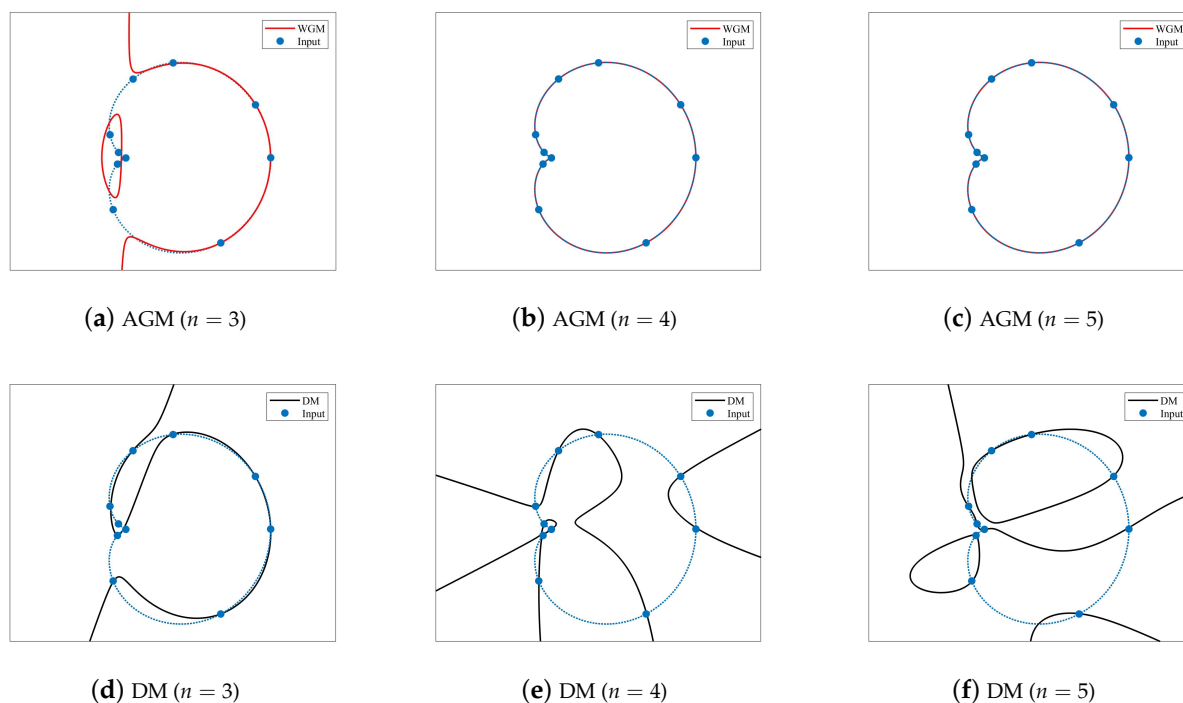


Figure 8. Adaptive implicitization of $C_3(t)$. The blue dash line in (a–f) is the input curve, the red line in (a–c) is the output curve by AGM, and the black line in (d–f) is the output curve by DM. From left to right: the implicit degree $n = 3, 4, 5$.

Figure 9 shows the statistic of our method on changes of the AD and AG loss for $C_3(t)$, when the implicit degree n is increasing. We note that the third order output ($n = 3$) gives relatively

large values of the AD and AG loss, and this can be attributed to the fact that the corresponding output is rather inflexible and is incapable of capturing the oscillations in the input curve $C_3(t)$. The third and fourth order ($n = 4, 5$) both give small values for the above two losses, and these also give reasonable implicit representations, as can be seen from Figure 8.

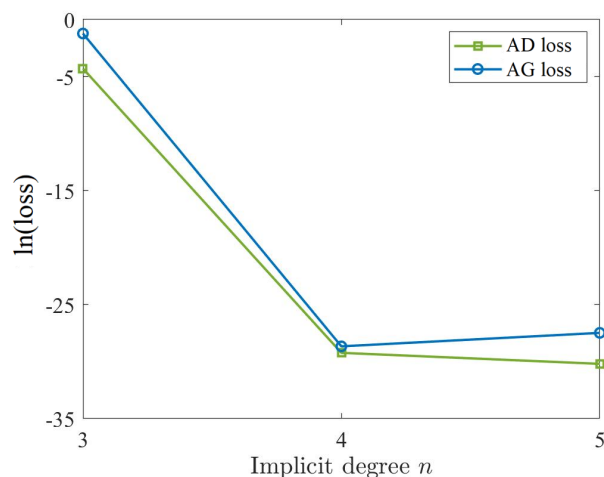


Figure 9. Statistics of our method on changes of the AD and AG loss for $C_3(t)$, as the implicit degree n increases.

Example 4. Consider the non-polynomial parametric curve

$$C_4(t) = \begin{pmatrix} t \cos t \\ t \sin t \end{pmatrix},$$

where the parameters of $C_4(t)$ take values in $[0, 14]$. $C_4(t)$ is shown in Figure 10.

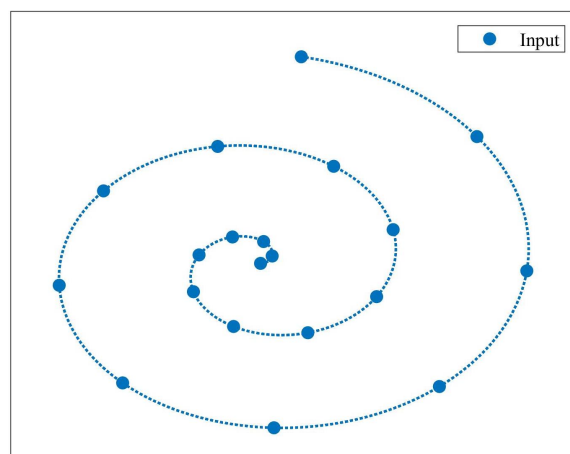


Figure 10. The input curve $C_4(t)$ and 20 points are sampled uniformly from $C_4(t)$.

The first row and second row of Figure 11 show the adaptive implicitization process of $C_4(t)$ by AGM and DM, respectively. We notice that the third- and fourth-order ($n = 3, 4$) outputs of AGM and DM both give rather poor fits to the input curve $C_4(t)$. When we go to higher-order polynomials ($n = 5, 6, 7$), our AGM still outputs “shape-preserving” fits, which refrain from additional branches as much as possible.

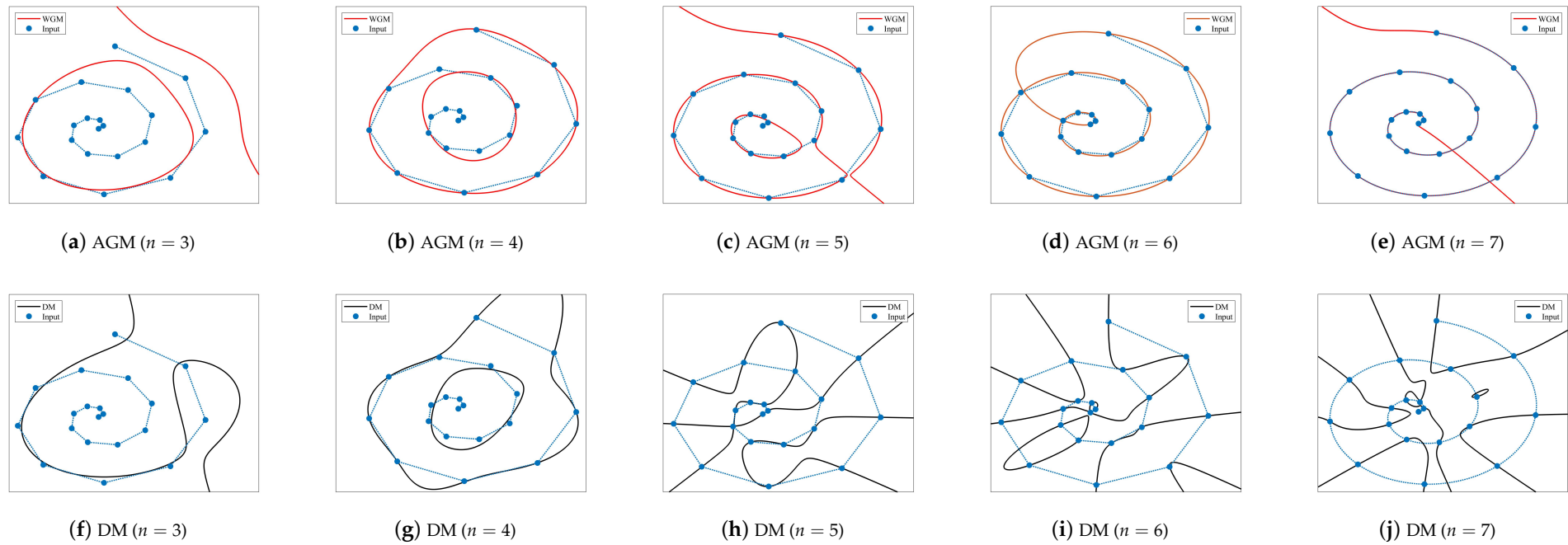


Figure 11. Adaptive implicitization of $C_4(t)$. The blue dash line in (a–j) is the input curve, the red line in (a–c,g,h) is the output curve by AGM, and the black line in (d–f,i,j) is the output curve by DM. From left to right: the implicit degree $n = 3, 4, 5$.

Figure 12 shows the statistic of our method on changes of the AD and AG loss for $C_4(t)$, when the implicit degree n is increasing. We note that the low orders ($n = 3, 4, 5$) give relatively large values of the AD and AG loss, and this can be attributed to the fact that the corresponding output is rather inflexible and is incapable of capturing the oscillations in the input curve $C_4(t)$. The seventh order ($n = 7$) gives small values for the above two losses, and this gives a reasonable implicit representation.

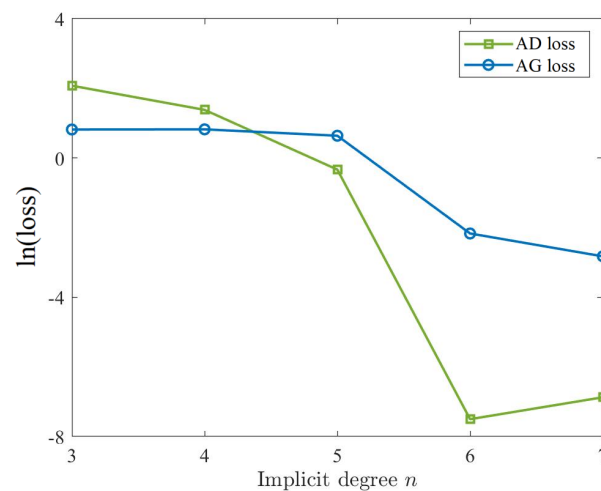


Figure 12. Statistics of our method on changes of the AD and AG loss for $C_4(t)$, as the implicit degree n increases.

Finally, the evaluation results of three methods (AGM, DM, and the method in [20]) on the input curves $C_3(t)$, $C_4(t)$ are presented in Table 2. Table 2 shows that the proposed AGM achieved enough precision. In addition, AGM has the second lowest inference time after DM, and is significantly faster than the method in [20].

Table 2. Performance of AGM on $C_3(t)$, $C_4(t)$ (all timings are measured in seconds).

Input	AD Loss	AG Loss	AGM Time	DM Time	Method in [20] Time
$C_3(t)$	2.011×10^{-13}	3.558×10^{-13}	0.0039	0.0027	59.49
$C_4(t)$	1.114×10^{-3}	6.721×10^{-2}	0.0051	0.0039	166.58

5. Conclusions

In this paper, we proposed a novel approach for the adaptive implicitization of parametric curves based on the so-called asymmetric geometric constraint, named AGM. AGM solves the implicitization problem with regularization terms naturally with very little extra computation effort. Thus, it not only avoids additional branches but also reduces the computational cost-effectively. Several experiments presented demonstrate that AGM produces high-quality implicitization results. In future work, we plan to generalize the proposed method to the cases of parametric surfaces and space parametric curves.

Author Contributions: Writing—original draft preparation, Y.G. and M.G.; writing—review and editing, Y.G.; visualization, Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are included within the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

AD	Algebraic distance
AG	Asymmetric gradient
DM	Dokken's method
AGM	Asymmetric gradient method

References

1. Raj, J.; Raghuwaiya, K.; Sharma, B.; Vanualailai, J. Motion Control of a Flock of 1-Trailer Robots with Swarm Avoidance. *Robotica* **2021**, *39*, 1926–1951. [\[CrossRef\]](#)
2. Raj, J.; Raghuwaiya, K.; Havea, R.; Vanualailai, J. Autonomous control of multiple quadrotors for collision-free navigation. *IET Control Theory Appl.* **2023**, *17*, 868–895. [\[CrossRef\]](#)
3. Walter, D.R.; Husty, M.L. On Implicitization of Kinematic Constraint Equations. *Mach. Des. Res.* **2010**, *26*, 218–226.
4. Perez-Diaz, S.; Shen, L.Y. Inversion, degree, reparametrization and implicitization of improperly parametrized planar curves using μ -basis. *Comput. Aided Geom. Des.* **2021**, *84*, 101957. [\[CrossRef\]](#)
5. Tran, Q.N. Efficient Gröbner walk conversion for implicitization of geometric objects. *Comput. Aided Geom. Des.* **2004**, *21*, 837–857. [\[CrossRef\]](#)
6. Anwar, Y.; Tasman, H.; Hariadi, N. Determining implicit equation of conic section from quadratic rational Bézier curve using Gröbner basis. *Proc. J. Phys. Conf. Ser.* **2021**, *2106*, 12–17. [\[CrossRef\]](#)
7. Pérez-Díaz, S.; Sendra, J.R. A univariate resultant-based implicitization algorithm for surfaces. *J. Symb. Comput.* **2008**, *43*, 118–139. [\[CrossRef\]](#)
8. Busé, L.; Laroche, C.; Yıldırım, F. Implicitizing rational curves by the method of moving quadrics. *Comput.-Aided Des.* **2019**, *114*, 101–111. [\[CrossRef\]](#)
9. Lai, Y.; Chen, F.; Shi, X. Implicitizing rational surfaces without base points by moving planes and moving quadrics. *Comput. Aided Geom. Des.* **2019**, *70*, 1–15. [\[CrossRef\]](#)
10. Dokken, T. Approximate implicitization. *Math. Methods Curves Surf.* **2001**, 81–102.
11. Barrowclough, O.J.; Dokken, T. Approximate implicitization using linear algebra. *J. Appl. Math.* **2012**, *2012*, 293746. [\[CrossRef\]](#)
12. Raffo, A.; Dokken, T. Piecewise approximate implicitization with prescribed conditions using tensor-product B-splines. In Proceedings of the International Congress on Industrial and Applied Mathematics, Valencia, Spain, 15–19 July 2019. [\[CrossRef\]](#)
13. Raffo, A.; Barrowclough, O.J.; Muntingh, G. Reverse engineering of CAD models via clustering and approximate implicitization. *Comput. Aided Geom. Des.* **2020**, *80*, 101876. [\[CrossRef\]](#)
14. Wang, R.; Wu, J. Approximate implicitization based on RBF networks and MQ quasi-interpolation. *J. Comput. Math.* **2007**, *25*, 97–103.
15. Wang, G.; Li, W.; Zhang, L.; Sun, L.; Chen, P.; Yu, L.; Ning, X. Encoder-X: Solving unknown coefficients automatically in polynomial fitting by using an autoencoder. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 3264–3276. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Jüttler, B. Least-Squares Fitting of Algebraic Spline Curves via Normal Vector Estimation. In *The Mathematics of Surfaces IX*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 263–280.
17. Jüttler, B.; Felis, A. Least-Squares Fitting of Algebraic Spline Surfaces. *Adv. Comput. Math.* **2002**, *17*, 135–152. [\[CrossRef\]](#)
18. Shalaby, M.; Jüttler, B. Approximate implicitization of space curves and of surfaces of revolution. In *Geometric Modeling and Algebraic Geometry*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 215–227.
19. Aigner, M.; Jüttler, B.; Poteaux, A. Approximate implicitization of space curves. In *Numerical and Symbolic Scientific Computing*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–19.
20. Interian, R.; Otero, J.M.; Ribeiro, C.C.; Montenegro, A.A. Curve and surface fitting by implicit polynomials: Optimum degree finding and heuristic refinement. *Comput. Graph.* **2017**, *67*, 14–23. [\[CrossRef\]](#)
21. Jüttler, B.; Chalmoviansky, P.; Shalaby, M.; Wurm, E. Approximate algebraic methods for curves and surfaces and their applications. In Proceedings of the 21st Spring Conference on Computer Graphics, Budmerice, Slovakia, 12–14 May 2005; pp. 13–18.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.