NUMERICAL SIMULATIONS OF A CURVE FLOW RELATED TO CENTRO-AFFINE INVARIANTS

YANHUA YU\(^1\) and FAN XIA\(^1\)

\(^1\)Northeastern University Department of Mathematics

October 19, 2023

Abstract

In this paper, we propose a novel numerical implementation for visualizing the evolutionary process of a curve flow. This flow is an invariant second-order flow in centro-affine geometry. The key of the scheme proposed in this essay is to indirectly obtain the position vector by solving the evolution equation of the support function. Additionally, to improve efficiency, we choose a specific tangential velocity which can eliminate the necessity to update normal angles at each iteration. The reason is that the normal angles do not change over time under this tangential velocity. Finally, we conduct multiple experiments to demonstrate the feasibility and efficiency of our proposed method.
NUMERICAL SIMULATIONS OF A CURVE FLOW RELATED TO CENTRO-AFFINE INVARIANTS

FAN XIA AND YANHUA YU*

ABSTRACT. In this paper, we propose a novel numerical implementation for visualizing the evolutionary process of a curve flow. This flow is an invariant second-order flow in centro-affine geometry. The key of the scheme proposed in this essay is to indirectly obtain the position vector by solving the evolution equation of the support function. Additionally, to improve efficiency, we choose a specific tangential velocity which can eliminate the necessity to update normal angles at each iteration. The reason is that the normal angles do not change over time under this tangential velocity. Finally, we conduct multiple experiments to demonstrate the feasibility and efficiency of our proposed method.

Key words- centro-affine geometry; crystalline tangential velocity; curve evolution; numerical simulation

1. INTRODUCTION

Over the past several decades, significant research has been dedicated to invariant geometric flows for curves and surfaces in Euclidean and affine geometries. Here is a concise expression to describe the geometric heat flow that remains invariant under group operations:

\[
X_t = X_{ss}.
\]  

(1.1)

In the case where the group \( G \) is the Euclidean group, the flow (1.1) can be identified as curve shortening flow (CSF) [12]. This leads to a second-order nonlinear parabolic equation involving the Euclidean curvature \( k \) and the Euclidean arc-length \( s \):

\[
k_t = k_{ss} + k^3.
\]

There is an extensive literature on the CSF published since the 1980s by several authors. Firstly, Dziuk proposed a finite element method for numerical simulation of the evolution of closed curves in \( \mathbb{R}^2 \) [5]. Then, Kimura [16, 17] investigated a tangential velocity \( \alpha \) to ensure that the curve maintains a constant relative local length \( r \) during the evolution. Furthermore, the authors in [6] introduced a variable time scale for the harmonic map heat flow to obtain numerical schemes for CSF. Recently, Elliott and Fritz proposed a finite element method based on the DeTurk trick [11], which is applicable to CSF and mean curvature flow (MCF).

* Corresponding author: yyh_start@126.com
When the group $G$ is the special affine group, the flow (1.1) is referred to as affine curve shortening flow (ACSF). In contrast to the CSF, the ACSF remains invariant under affine transformations. According to [1, 25], this flow gives rise to a second-order nonlinear parabolic equation for the equiaffine curvature $k$ and equiaffine arc-length $s$

$$k_t = \frac{4}{3} k_{ss} + \frac{1}{3} k^3.$$ 

A geometric method is proposed in [3, 4] for numerical approximation of ACSF in the plane. This method uses common invariants to approximate the difference invariants of the Lie group action and then uses discrete finite difference to estimate the affine curvature. Besides, the authors established a connection between grid peeling and ACSF [9, 29], which makes the numerical computation more efficient.

There are two important areas of investigation within affine differential geometry: centro-equiaffine differential geometry and centro-affine differential geometry. The former is determined by the special linear group $SL(n, R)$, which represents a set of volume-preserving linear transformations [30]. And the flow (1.1) induces a second-order nonlinear parabolic equation for the centro-equiaffine curvature $k$ and centro-equiaffine arc-length $s$

$$k_t = k_{ss} + 4k^3.$$ 

The latter consists of the subgroup of the affine transformation group that fixes the origin, identified by the general linear group [23]. In this case, the flow (1.1) leads to the well-known inviscid Burgers equation related to the centro-affine curvature $k$ and centro-affine arc-length $s$

$$k_t = kk_s.$$ 

In [15], XinJie Jiang et al. discovered an invariant flow in centro-affine differential geometry, which is governed by a second-order nonlinear parabolic equation for the centro-affine curvature. In the centro-affine setting, this flow is characterized as

$$X_t (p, t) = \left( \lambda + \int_0^{s(p)} kds \right) X + \frac{k}{2} X_s,$$

where $p$ is the arbitrary curve parameter and $\lambda$ is a constant. However, in the Euclidean setting, the flow (1.2) is equivalent to

$$X_t = -\frac{1}{2} \log \left( \frac{k}{h^3} \right) X = \beta N,$$

where the normal velocity

$$\beta = \frac{h}{2} \log \left( \frac{k}{h^3} \right).$$

Here $k$ denotes Euclidean curvature, $N$ represents the normal vector and $h = -\langle X, N \rangle$ is the support function of the curve.
This paper primarily focuses on the numerical implementation of the flow under the normal velocity (1.4). The numerical implementation of geometric flows has attracted significant attention (see, for example, [7, 8]) due to the difficulty in obtaining explicit solutions. There are numerous strategies for dealing with the numerical evolution of geometric flows, which can be categorized into two types: the embedded methods and the interface tracking methods. This manuscript primarily utilizes the interface tracking technique to describe the evolving curve as a sequence of discrete nodes. In conventional methods, we directly evolve the position vectors of these nodes to obtain their positions in the subsequent time steps. However, we adopt an indirect approach by evolving the support function of the nodes to indirectly derive the position vectors.

Only moving grid nodes along the normal direction may lead to undesired concentration [13, 18, 26, 31]. Therefore, to ensure a high-quality grid, we introduce the tangential velocity $\alpha$, so that the grid nodes evolve according to the following equation:

\begin{equation}
X_t = \beta N + \alpha T.
\end{equation}

This approach is effective because the tangential velocity $\alpha$ has no impact on the shape of the evolving curve [10]. There are also several well-known strategies for choosing the tangential velocity, such as preserving relative local length [13, 17, 19], asymptotically uniform redistribution [20, 21, 22], crystalline curvature redistribution [31] and curvature adjusted redistribution [2, 27]. In our numerical implementation, we adopt the crystalline tangential velocity $\alpha = -k^{-1}\beta_s$. Under this tangential velocity, the grid points are distributed densely (sparsely) in those part of the curve where the absolute value of the curvature is large (small). This choice takes into account the non-uniform shape of the curve. In addition, the second advantage of this choice is that it saves time in updating the normal angles. This is mainly for during the evolution process according to equation (1.5), the normal angle of the grid nodes remains unchanged.

The remainder of this paper is organized as follows: in Section 2, we present a system of governing equations for describing the evolution of plane curves. To obtain a numerical solution for the flow (1.4), we state two numerical approximation schemes in Section 3. In Section 3.1, a numerical scheme is provided for directly evolving the position vector. On the other hand, in Section 3.2, a numerical scheme is investigated that aims to indirectly obtain the position vector by evolving the support function. In Section 4, we conduct numerical experiments on a variety of closed convex curves using the scheme constructed in Section 3. The results demonstrate the feasibility of the proposed novel scheme. Then we observe its time-saving benefits upon comparison and analysis of the experimental results. In Section 5 we perform experiments to verify self-similar solutions, further demonstrating the feasibility of our proposed method. Finally conclusions and future research lines are outlined in Section 6.
2. Governing equations

This paper primarily focuses on the evolution of closed smooth plane curves $X$ under the flow (1.4). These curves are parameterized by $X = \{X(u), u \in [0,1]\}$. The arc-length parameter is denoted by $s$. For the unit tangent vector we have $T = X_s$ and we can choose the unit inward normal vector $N$ such that $\det(T, N) = 1$. Furthermore, we always choose the normal angle as $\theta$. The problem of evolution of curves is stated as follows: For a given initial curve $X^0 = \{X(u, 0)\}$, find a family of planar curves $X^t = \{X(u, t)\}, u \in [0,1], t \in [0, +\infty)$. Using the Frenet formula $X_{ss} = T_s$, the equation governing the position vector $X$ takes the following form:

\[ X_t = \alpha X_s + w X_{ss}, \quad w = \frac{\beta}{k} = \frac{h}{2k} \log \left( \frac{k}{h^3} \right), \]

where $X_s = g^{-1}X_u$. The symbol $g = |X_u|$ stands for the local length of a curve that is parameterized by $X = (x, y)_u$. According to [28], the tangential velocity in equation (2.1) can be regarded as a solution to:

\[ (\varphi(k) \alpha)_s = f - \frac{\langle f \rangle}{\langle \varphi(k) \rangle} \varphi(k) + \omega(t) \left( \frac{L^t}{g(u, t)} \langle \varphi(k) \rangle - \varphi(k) \right), \]

where $f = \varphi(k) k\beta - \varphi_k(k) (\beta_{ss} + k^2 \beta)$ and the shape function $\varphi(k)$ is a function related to the modules of curvature. Here the bracket $\langle F \rangle$ denotes the average of any function $F(u, t)$ on the curve $X^t$:

\[ \langle F \rangle = \frac{1}{L^t} \int_{X^t} F \, ds, \]

$L^t$ signifies the total length of a curve: $L^t = \int_{X^t} ds = \int_0^1 g(u, t) \, du$. In order to ensure the uniqueness of the solution $\alpha$ for equation (2.2), the authors of [28] assume the following renormalization condition:

\[ \langle \varphi(k) \alpha \rangle = 0. \]

The strategies for the selection of tangential velocity can be categorized into the following cases:

**Case 1.** (uniform redistribution) Consider the case of the shape function $\varphi(k) = 1$. The relaxation function $\omega = \omega(t)$ determines the rate at which redistribution becomes uniform. According to the cited in [26], when $\omega = 0$, the solution $\alpha$ preserves the relative local length $r$. When $\omega \neq 0$, asymptotically uniform redistribution (AUTR) is introduced in [20, 21, 22].

**Case 2.** (curvature adjusted redistribution) Consider the case which has following smoothed shape function:

\[ \varphi(k) = 1 - \varepsilon + \varepsilon \sqrt{1 - \varepsilon + \varepsilon k^2}, \quad \varepsilon \in (0, 1). \]

As $\varepsilon \to 1$, the grid nodes tend to distribute more towards regions with larger curvature modulus. In addition, using a suitable shape function allows for
a more accurate fitting of the actual curve and minimizes the discrepancies in area or length introduced by discretization. The shape function \([28]\) for the length and area optimization is given as:

\[
\varphi (k) = |k|^{1/3} \text{ for the length discrepancy,}
\]

\[
\varphi (k) = |k|^{2/3} \text{ for the area discrepancy.}
\]

In Figure 1 we plot an example of distribution of grid points along the ellipse at different values of \(\varepsilon\). It demonstrates a strong correlation between the value \(\varepsilon\) and the point distribution.

**Case 3.** (crystalline curvature redistribution (CCR)) Consider the case where the shape function \(\varphi (k) = |k|\) and \(\omega (t) = 0\). By solving (2.2) and (2.3), we end up with the tangential velocity \(\alpha = -k^{-1}\beta_s\).

In order to obtain the position vector at subsequent time, Ševčovič et al. discretize the position vector equation (2.1) directly in [28]. However, in this paper, a more accurate numerical scheme is proposed by us. Instead of directly solving the position vector equation (2.1), we indirectly obtain the position vector \(X\) by solving the evolution equation of the support function. According to Eq. (2.1), we have the normal vector equation as follows:

\[
\theta_t = \beta_s + \alpha k.
\]

Subsequently, the evolution equation for the support function \(h\) can be derived from Eq. (2.1) and (2.5) as

\[
h_t = -\beta + \theta_t h_\theta.
\]

It is natural to solve (2.6), we must get \(\theta\) from (2.5). Considering the computational time complexity, we adopt the crystalline tangential velocity \(\alpha = -k^{-1}\beta_s\) that makes \(\theta\) satisfy \(\theta_t = 0\). Thus the support function Eq. (2.6) can be rewrite as:

\[
h_t = -\beta = \frac{h}{2} \log h^3 (h + h_{\theta \theta}).
\]

Notably, there is no necessity for updating \(\theta\) at each iteration. Therefore discretization of (2.7) yields satisfactory numerical results. In addition, the position vector is gained indirectly as

\[
X(\theta) = \begin{pmatrix} -h \cos \theta + h_\theta \sin \theta \\
-h_\theta \cos \theta - h \sin \theta \end{pmatrix}.
\]

3. **Numerical approximation**

The numerical approximation of a moving curve \(X^t\) is achieved based on the polygonal representation of curves. The curve \(X\) can be approximated as an \(m\)-sided polygonal curve \(P = \sum_{i=1}^m S_i\), where the vertices are labeled as \(X_i\) in counterclockwise order and the edges are depicted as \(S_i = [X_{i-1}, X_i]\), with \(i = 1, 2, ..., m\). We introduce the flowing dual volume \(S^*_i = [X^*_i, X_i] \cup [X_i, X^*_{i+1}]\) of \(S_i\), where \(X^*_i\) signifies \(\frac{X_i + X_{i-1}}{2}\). Through the following steps,
Figure 1. (a), (c), (e) are evolving ellipses starting from the same ellipse with semi-axes $a_0 = 0.5$ and $b_0 = 1$, and (b), (d), (f) are the curves when $t = 0.1$, respectively.

We can compute several quantities on $\mathcal{P}$ from $\{X_i\}_{i=1}^m$. It is important to note that any quantities defined for $m$-sided polygons below adhere to the periodic numbering convention, i.e., $F_0 = F_m$ and $F_{m+1} = F_1$. 
Step 1  The length on the edge $S_i$ is signified as

$$r_i = |X_i - X_{i-1}|,$$

and the quantity at the point $X_i$ is signified as $r_i^*$ which are defined by an average of $r_i$ and $r_{i+1}$.

Step 2  The total length of the curve is approximated by the length $L$ of the polygon,

$$L = \sum_{i=1}^{m} r_i = \sum_{i=1}^{m} r_i^*.$$

Step 3  The $i$-th unit normal angle $\theta_i$ is derived from the $i$-th outward unit normal vector $N_i = (\cos \theta_i, \sin \theta_i)^\top$ on $S_i$. The approximation of $N$ can be expressed as $N_i = -T^\perp_i$, $T_i = (X_i - X_{i-1}) / r_i$.

Note that $T_i$ stands for the $i$-th unit tangent vector on $S_i$. Firstly, from $N_1 = (N_{11}, N_{12})^\top$, we obtain $\theta_1 = 2\pi - \arccos(N_{11})$ if $N_{12} < 0$ and $\theta_1 = \arccos(N_{11})$ if $N_{12} \geq 0$. Secondly, we can iteratively compute $\theta_{i+1}$ from $\theta_i$ for $i = 1, 2, ..., m$.

$$\theta_{i+1} = \begin{cases} 
\theta_i + \arcsin(D) & \text{if } I > 0, \\
\theta_i + \arcsin(I) & \text{if } D > 0, \\
\theta_i - \arcsin(I) & \text{otherwise.}
\end{cases}$$

Here, $I = N_i \cdot N_{i+1}$ symbolizes the Euclidean inner product of vectors $N_i$ and $N_{i+1}$. On the other hand, $D = \det(N_i, N_{i+1})$ demonstrates the determinant of the matrix formed by vectors $N_i$ and $N_{i+1}$. Finally, we obtain $\theta_0 = \theta_1 - (\theta_{m+1} - \theta_m)$ and $\theta_{m+2} = \theta_{m+1} + (\theta_2 - \theta_1)$. Similarly, the normal angle at the point $X_i$ can be represented as $\theta_i^* = \frac{\theta_i + \theta_{i+1}}{2}$.

Step 4  The $i$-th support function at the point $X_i$ is devoted by

$$h_i = -x_i \cos \theta_i^* - y_i \sin \theta_i^*.$$

Because $k = \frac{1}{h_i + ddh_i}$, the $i$-th curvature $k_i$ at the point $X_i$ can be illustrated as

$$k_i = \frac{1}{h_i + ddh_i}, \quad ddh_i = 2 \frac{h_{i+1} - h_i}{\theta_{i+1} - \theta_i} - \frac{h_i - h_{i-1}}{\theta_{i} - \theta_{i-1}},$$

where $ddh_i$ is the approximation of $h_{\theta \theta}$.

We now turn our attention to give direct and indirect schemes for numerical discretization. Section 3.1 investigates the direct scheme, while Section 3.2 presents the indirect scheme.

3.1. The discretization of the evolution equation of the position vector.

To begin with, we employ a numerical semi-implicit scheme in time to approximate the position vector $X$ at discrete time $t_j = j\tau (j = 0, 1, 2, \ldots)$.
Here $\tau$ is a fixed positive time increment. The temporal discretization can be expressed as

$$\frac{X^j - X^{j-1}}{\tau} = w^{j-1} (X^j)_{ss} + \alpha^{j-1} (X^j)_s.$$  

Next we integrate (2.1) over the flowing dual volume $S^*_i$ for spatial discretization. We can show the spacial discretization that

$$\frac{(X_i)_t}{r_i^s} = \frac{w_i}{r_i^s} \left( \frac{X_{i+1} - X_i}{r_{i+1}} - \frac{X_i - X_{i-1}}{r_i} \right) + \frac{\alpha_i}{2r_i^s} (X_{i+1} - X_{i-1}),$$

where $w_i = w (X_i, \theta_i^*, k_i^*)$. Finally, combining (3.1) and (3.2), we distinguish the fully-discrete linear system. The unknowns $X^j_i$ provide the numerical solution, given by the following equation:

$$\frac{X^j_i - X^{j-1}_i}{\tau} = a_- X^j_{i-1} + a_0 X^j_i + a_+ X^j_{i+1},$$

$a_- = \frac{b}{r_{i+1}^s} - a$, $a_0 = -(a_+ + a)$, $a_+ = \frac{b}{r_{i+1}^s} + a$, $a = \frac{\alpha_i^{j-1}}{2r_i^s}$, $b = \frac{w_i^{j-1}}{r_i^s}$, where $X^j_i$ is the $i$-th position vector of an $m$-sided polygonal curve $\mathcal{P}$ at time $t = j\tau$ ($i = 1, 2, ..., m$). In other words, we end up with a tridiagonal system for updating the discrete position vector at the new $j$ time level:

$$-a_- \tau X^j_{i-1} + (1 - a_0 \tau) X^j_i - a_+ \tau X^j_{i+1} = X^j_{i-1}$$

subject to periodic boundary conditions $X^j_0 = X^j_m, X^j_{m+1} = X^j_1$. Each tridiagonal system is solved using the Gauss-Seidel iteration, and the iteration is stopped when the maximum norm of the difference between subsequent Gauss-Seidel iterations is less than the prescribed tolerance, eg., $TOL = 10^{-10}$. Note that the tridiagonal matrix is strictly diagonally dominant as long as $\tau$ is small enough. The time step $\tau$ can be adaptively chosen using the following expression [28]:

$$\tau_j = \frac{r_j^2}{4 (1 + \lambda)} \left( \frac{w_{max}^j}{r_j^2_{min}} + \frac{\max |\alpha^j|}{2} \right)^{-1},$$

where $\lambda > 0$, $F_{min} = \min_{1 \leq i \leq m} F_i$, $F_{max} = \max_{1 \leq i \leq m} F_i$, and $|F|_{max} = \max_{1 \leq i \leq m} |F_i|$. 

Next, we discretize the tangential velocity equation (2.2) and the renormalization constraint (2.3) to obtain an approximate discrete representation of the tangential velocity $\alpha$. The discretization of the tangential velocity is

$$\alpha_1 = -\frac{1}{\mathcal{L}_i\varphi_1^i} \sum_{i=2}^{m} \Psi_i^* r_i^s, \alpha_i = \frac{1}{\varphi_1^i} \left( \varphi_1^* \alpha_1 + \Psi_i \right), \; i = 2, 3, ..., m.$$ 

The detailed process of discretization can be seen in [28]. In the above expression, $\varphi_i$ and $\varphi_1^*$ denote the approximate values of the shape function $\varphi(k)$ defined on $\mathcal{S}_i$ and at the point $X_i$, respectively. $\Psi_i$ is the partial sum
of \(\{\psi_i\}\), which is defined as \(\Psi_i = \sum_{l=2}^{\psi_i} \psi_l\), for \(i = 2, 3, \ldots, m\), with \(\Psi_1 = 0\). 

\(\{\psi_i\}\) is considered as follows:

\[
(3.5) \quad \psi_i = \frac{\langle f \rangle}{\langle \varphi \rangle} \varphi_i r_i - f_i r_i + \left( \mathcal{L} \langle \varphi \rangle \frac{1}{N} - \varphi_i r_i \right) \omega,
\]

\[
f_i = (\beta_{s^*})_i + k_i \beta_i \varphi_k (k_i) - k_i \beta_i \varphi_i.
\]

What we shall concentrate on here is that \(\beta_i = \beta (X^*_i, \theta, k_i)\) is a constant on \(S_i\), and \((\beta_{s^*})_i\) can be expressed as:

\[
(\beta_{s^*})_i = \frac{(\beta_{s^*})_i - (\beta_{s^*})_{i-1}}{r_i},
\]

where \((\beta_{s^*})_i = \frac{\beta_{i+1} - \beta_i}{r_i^*}\). Throughout the article, we always take for granted that the approximation of \(\langle F \rangle\) for any function \(F\):

\[
\langle F \rangle = \frac{1}{L} \sum_{i=1}^{m} F_i r_i.
\]

It can be observed from (3.4) that the tangential velocity is closely related to the shape function. In Section 4, we perform numerical simulations using the scheme (3.3) under different shape functions.

### 3.2. The discretization of the evolution equation of the support function.

Our main goal is to acquire an approximate support function by discretizing the evolution equation (2.7) in this section. Subsequently, we derive the position vector indirectly by utilizing the approximate support function.

Firstly, the discretization in time of the evolution equation (2.7) is given as follows:

\[
(3.6) \quad \frac{h^j - h^{j-1}}{\tau} = -\frac{h^{j-1}}{2} \log \left( (h^{j-1})^3 \left( h^{j-1} + h_{\theta \theta}^{j-1} \right) \right).
\]

Next, we discretize the equation (2.7) in space using finite difference methods. Since the normal angle \(\theta\) satisfies \(\theta_t = 0\), it is sufficient to determine the initial normal angles. The determination of the initial normal angles can be divided in two ways: uniform initial distribution (UI distribution) and non-uniform initial distribution (Non-UI distribution). For Non-UI distribution, we can illustrate the approximate values of \(h_{\theta \theta}\) and \(h_{\theta}\) as \(ddh\) and \(dh\) respectively, in the following manner:

\[
\begin{align*}
ddh_i &= \frac{h_{i+1} - h_i}{\theta_{i+1}^* - \theta_i^*}, \\
dh_i &= \frac{2h_{i+1} - h_{i-1}}{\theta_{i+1}^* - \theta_{i-1}^*},
\end{align*}
\]

where \(i = 1, 2, \ldots, m\). Alternatively, in the case of UI distribution, the approximation for \(h_{\theta \theta}\) and \(h_{\theta}\) can be seen in Algorithm 1. The fully discretized
formulation is obtained by combining equation (3.6) with the approximation for \( h_{\theta \theta_i} \), resulting in the following expression:

\[
(3.7) \quad \frac{h_i^j - h_i^{j-1}}{\tau} = -\frac{h_i^{j-1}}{2} \log \left( \left( h_i^{j-1} \right)^3 \left( h_i^{j-1} + dhh_i^{j-1} \right) \right),
\]

where \( h_i^j \) is the \( i \)-th support function of an \( m \)-sided polygonal curve \( P \) at time \( t = j \tau \) (\( i = 1, 2, ..., m \)). The complete numerical approximation for Non-UI distribution is presented in Algorithm 1. The advantage of UI distribution over Non-UI distribution lies in its more accurate representation of the curve. This is because UI distribution satisfies \( \theta_s = k \), which leads to the curvature satisfying \( k_i r_i = k_j r_j \), where \( i, j = 1, ..., m \). As a result, there are more (fewer) vertices in regions with larger (smaller) curvature.

Faster generation of numerical results can be acquired by using this numerical scheme instead of the numerical scheme in Section 3.1. This is verified through numerical experiments conducted in Section 4.

4. Discussion on numerical experiments

The purpose of this section is to study the curve flow driven by the normal velocity (1.4). When visualizing the numerical results, we represent the curves with solid lines and indicate the motion of selected mesh points with the points.

Firstly, we present the numerical results attained from simulating the evolution of ellipses using the scheme (3.7). The initial position vectors are determined by uniformly distributing the normal angles. Here is the specific method:

\[
x_i = \begin{cases} 
  a_0^2 \sin \theta_i \sqrt{\frac{1}{b_0^2 \cos^2 \theta_i + a_0^2 \sin^2 \theta_i}} & \text{if } \theta_i \leq \pi, \\
  -a_0^2 \sin \theta_i \sqrt{\frac{1}{b_0^2 \cos^2 \theta_i + a_0^2 \sin^2 \theta_i}} & \text{if } \theta_i > \pi,
\end{cases}
\]

\[
y_i = \begin{cases} 
  \frac{b_0^2 \sin \theta_i \cos \theta_i}{a_0^3 \sin \theta_i} & \text{if } \theta_i \neq \pi, \\
  \frac{b_0 \sqrt{1 - \tan^2 \theta_i}}{a_0} & \text{if } \theta_i = \pi.
\end{cases}
\]

The evolving results of the experiment are visualized in Figure 2. It is evident that when the initial curve \( C_0 \) is a circle with radius \( r_0 = 1.2 > 1 \), the curve expands outward. Conversely, if the radius is \( r_0 = 0.8 < 1 \), the curve shrinks inward. In the case where the initial curve \( C_0 \) is an ellipse, the observed results indicate that the curve expands outward if the initial semi-axes are \( a_0 = 1.25 \) and \( b_0 = 1 \) (\( a_0 b_0 > 1 \)), but shrinks inward if they are \( a_0 = 0.75 \) and \( b_0 = 1 \) (\( a_0 b_0 < 1 \)). Remarkably, these results are consistent with the conclusions presented in the paper [24].
Algorithm 1 Numerical approximation for UI distribution.

**Input:** Number of grid points \( m \); Parameter equations \( x(v), y(v) \); Number of time steps \( M \); Time step size \( \tau \);

**Output:** The position vector \( X \) at time \( t = M\tau \);

1: Compute uniform increment of normal angle \( \Delta \theta = \frac{2\pi}{m} \);
2: Initialize normal angle \( \theta_i^* = i\Delta \theta, i = 1, 2, ..., m \);
3: Compute the derivative of \( y \) to \( x \):
   \[
   \dot{y}_i = -\frac{\cos \theta_i^*}{\sin \theta_i^*},
   \]
   where \( \cdot \) stands for taking the derivative of \( x \);
4: Compute parameter \( v_i \) by solving \( \dot{y}_i = y_v(v_i) \);
5: Compute initial position vector \( x_i = x(v_i), y_i = y(v_i) \);
6: Compute initial support function \( h_i = -x_i \cos \theta_i^* - y_i \sin \theta_i^* \);
7: for \( j = 1 : M \) do
8: Compute approximation \( dh \) of \( h_{\theta \theta} \) at \( j - 1 \) moment,
   \[
   ddh_i = \frac{h_{i+1} - 2h_i + h_{i-1}}{(\Delta \theta)^2};
   \]
9: Compute \( h_i \) at \( j \) moment,
   \[
   h_i = h_i - \frac{\tau h_i}{2} \log \left( h_i^3 (h_i + ddh_i) \right);
   \]
10: end for
11: Compute approximation \( dh \) of \( h_{\theta} \) at time \( t = M\tau \),
   \[
   dh_i = \frac{h_{i+1} - h_{i-1}}{2\Delta \theta};
   \]
12: Compute the position vector \( X \) at time \( t = M\tau \),
   \[
   x_i = -h_i \cos \theta_i^* + dh_i \sin \theta_i^*,
   y_i = -dh_i \cos \theta_i^* - h_i \sin \theta_i^*.
   \]

Next we analyze the evolution of four convex closed curves under the flow (1.4), parameterized respectively as
\[
\rho = (\sin 2v + 7)/8,
\]
\[
\rho = (\sin 3v + 15)/16,
\]
\[
\rho = (\sin 5v + 39)/40,
\]
\[
\rho = (\sin 5v + 59)/60.
\]
The coordinates \((x, y)^\top\) of any curve are equal to \((\rho \cos v, \rho \sin v)^\top\), where the parameter \( \rho \) symbolizes the polar radius and \( v \) signifies the polar angle.
Figure 2. Evolution of initial curve which is the circle (a), (b) and ellipse (c), (d), respectively. The blue dashed lines embody the initial curves, and the red lines portray the curves when $t = 1$. Numerical parameters: $m = 64$, $\tau = 0.001$.

From Figures 3(a)-(d), it can be detected that at time $t = 1$, the initial convex closed curve (4.1)-(4.4) tends to approach an elliptical shape. Further analysis of the shape feature can be conducted by comparing the normalized centro-equiaffine curvature. We shall employ the definition of normalized centro-equiaffine curvature

$$H_i^j = \frac{k_i^j}{h_i^j}, \quad i = 1, \ldots, m,$$

where $k_i^j$ and $h_i^j$ respectively signify curvature, and support function of the $i$-th point at time $t_j = j\tau$. Applying normalization techniques to map the
data onto the interval $[0,1]$, we can compare and analyze data with varying scales.

In Figure 4, different colored lines are used to distinguish curves at different time points. Figure 4 (left) illustrates the evolution of the curve parameterized as (4.1). It can be perceived in Figure 4 (right) that the normalized centro-equiaffine curvature at different points gradually converges to a consistent value over time. Additionally, we calculate the ratio of the minimum to maximum normalized centro-equiaffine curvature in Table 1, and the results showed that regardless of the initial shape of the curve, this ratio tends to approach 1. This further demonstrates that as time increases, the centro-equiaffine curvature $\frac{k}{\kappa^3}$ tends to converge to a consistent value, resulting in the shape of the curve approaching an ellipse over time (as shown in Figure 4). This finding is consistent with Theorem 1.1 proposed in [15].

**Figure 3.** The evolution of different initial convex closed curves. The blue lines embody the initial curve, and the red lines portray the curve when $t = 1$. Numerical parameters: $m = 64$, $\tau = 0.001$. 

(a) $\rho = \frac{\sin 3v + 15}{16}$  
(b) $\rho = \frac{\sin 2v + 7}{8}$  
(c) $\rho = \frac{\sin 5v + 39}{40}$  
(d) $\rho = \frac{\sin 5v + 59}{60}$
Table 1. The ratio of the minimum to maximum centro-equiaffine curvature.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$\rho = (\sin 2v + 7)/8$</th>
<th>$\rho = (\sin 3v + 15)/16$</th>
<th>$\rho = (\sin 5v + 39)/40$</th>
<th>$\rho = (\sin 5v + 59)/60$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.5297</td>
<td>0.3982</td>
<td>0.2744</td>
<td>0.4708</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8975</td>
<td>0.6748</td>
<td>0.8750</td>
<td>0.9149</td>
</tr>
<tr>
<td>0.4</td>
<td>0.9651</td>
<td>0.7807</td>
<td>0.9825</td>
<td>0.9885</td>
</tr>
<tr>
<td>0.6</td>
<td>0.9870</td>
<td>0.8638</td>
<td>0.9977</td>
<td>0.9985</td>
</tr>
<tr>
<td>0.8</td>
<td>0.9934</td>
<td>0.9143</td>
<td>0.9997</td>
<td>0.9998</td>
</tr>
<tr>
<td>1.0</td>
<td>0.9942</td>
<td>0.9466</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Figure 4. Evolution for the interface (left) and its sum normalized centro-equiaffine curvature (right) with the curve $\rho = (\sin 2v + 7)/8$ as initial configuration. Numerical parameters: $m = 64$, $\tau = 0.001$.

Figures 5(a)-(f) illustrate the evolution process with the curve (4.1) as the initial configuration. In Figures 5(a)-(c), we employ UI distribution. It can be observed that more points are used to describe the curve in regions where the modulus of curvature is larger. However, in Figures 5(d)-(f), the Non-UI distribution is used, and it can be seen that there is rapid merging in certain regions and poor discrete representation in other parts of the curve. This merging leads to the occurrence of the swallowtail phenomenon, as shown in Figure 5(f). Clearly, the evolving curves generated through the UI distribution exhibit a superior effect compared to those produced with the Non-UI distribution.

In addition, we analyze the comparison of discrepancy in area and length. If the initial curve is an ellipse, we get the position vector of the moving curve through $X(u,t) = \eta(t)z(u)$, where the scaling function $\eta(t) =$
NUMERICAL SIMULATIONS OF A SECOND-ORDER CURVE FLOW

\[
a_0 b_0 \left( \exp(2t) - 1 \right)^{1/2} \quad \text{and} \quad z(u) = X(u, 0) = (a_0 \cos(2\pi u), b_0 \sin(2\pi u)). \]

The following formula is derived to introduce the numerical discrepancy between the curve \( X^t \) and the polygonal curve \( P^t \) at time \( t = t_j \):

\[
\Delta t_j^L = \left| 1 - \frac{L^t_j}{L^j} \right|, \quad \Delta t_j^A = \left| 1 - \frac{A^t_j}{A^j} \right|.
\]

Here, \( L^t = \eta(t)L^0 \) and \( A^t = \eta(t)^2A^0 \) illustrate the length and area of the curve \( X^t \), respectively, while \( L^t \) and \( A^t \) exemplify the length and area of the polygonal curve \( P^t \). Table 2 shows the numerical discrepancies attained by performing the numerical scheme (3.3) at time \( t = 0.1 \). Table 3 corresponds to another numerical scheme (3.7). By considering the results from both Table 2 and Table 3, we conclude that using the numerical scheme (3.7) with the UI distribution can most effectively optimize the length and area.

An important tool for testing numerical algorithms is the so-called experimental order of convergence (EOC). The idea behind the definition of EOC is straightforward. For the order of convergence \( \alpha \) with respect to a spatial discretization parameter \( h = 1/m \), the error \( E_h(m) \) (calculated in the prescribed \( L^p(0, 1) \) norm) between the numerical solution and the exact

\[\text{Figure 5. Initial curves together with evolving curves in time steps } t = 0.15, 0.45. \] The discrete representations with UI distribution (a)-(c) and Non-UI distribution (d)-(f). The blue lines embody the initial curves, the green lines portray the curves when \( t = 0.15 \) and the red depict when \( t = 0.45 \).
Table 2. The values of length and area discrepancy using numerical scheme (3.3). Numerical parameters: \( \tau = 0.1/m^2 \), \( t = M\tau = 0.1, M = m^2 \).

<table>
<thead>
<tr>
<th>Discrepancy</th>
<th>( \varphi_0 (k) \equiv 1 )</th>
<th>( \varphi_{0.1} (k) )</th>
<th>( \varphi_{0.5} (k) )</th>
<th>( \varphi_{0.9} (k) )</th>
<th>( \varphi_1 (k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta L</td>
<td>_{m=16} )</td>
<td>0.009509</td>
<td>0.009509</td>
<td>0.009332</td>
<td>0.008196</td>
</tr>
<tr>
<td>( \Delta L</td>
<td>_{m=32} )</td>
<td>0.002392</td>
<td>0.002391</td>
<td>0.002344</td>
<td>0.002065</td>
</tr>
<tr>
<td>( \Delta L</td>
<td>_{m=64} )</td>
<td>0.000599</td>
<td>0.000599</td>
<td>0.000587</td>
<td>0.000517</td>
</tr>
<tr>
<td>( \Delta L</td>
<td>_{m=128} )</td>
<td>0.000150</td>
<td>0.000150</td>
<td>0.000147</td>
<td>0.000129</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=16} )</td>
<td>0.032059</td>
<td>0.032049</td>
<td>0.031853</td>
<td>0.031496</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=32} )</td>
<td>0.008129</td>
<td>0.008125</td>
<td>0.008061</td>
<td>0.007954</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=64} )</td>
<td>0.002040</td>
<td>0.002039</td>
<td>0.002022</td>
<td>0.001994</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=128} )</td>
<td>0.000510</td>
<td>0.000510</td>
<td>0.000506</td>
<td>0.000499</td>
</tr>
</tbody>
</table>

Table 3. The values of length and area discrepancy using numerical scheme (3.7). Numerical parameters: \( \tau = 0.1/m^2 \), \( t = M\tau = 0.1, M = m^2 \).

<table>
<thead>
<tr>
<th>Discrepancy</th>
<th>UI distribution</th>
<th>Non-UI distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta L</td>
<td>_{m=16} )</td>
<td>0.005628</td>
</tr>
<tr>
<td>( \Delta L</td>
<td>_{m=32} )</td>
<td>0.001469</td>
</tr>
<tr>
<td>( \Delta L</td>
<td>_{m=64} )</td>
<td>0.000372</td>
</tr>
<tr>
<td>( \Delta L</td>
<td>_{m=128} )</td>
<td>0.000093</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=16} )</td>
<td>0.026760</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=32} )</td>
<td>0.007042</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=64} )</td>
<td>0.001786</td>
</tr>
<tr>
<td>( \Delta A</td>
<td>_{m=128} )</td>
<td>0.000448</td>
</tr>
</tbody>
</table>

The solution satisfies \( E_p (m) \approx m^{-\alpha} \). Numerical error at \( t = t_j \) with the number of points \( m \) can be defined as

\[
E_p^j (m) = \begin{cases} 
\max_{1 \leq i \leq m} \left| \frac{(x^j_i)^2}{(a\eta(t_j))^2} + \frac{(y^j_i)^2}{(b\eta(t_j))^2} - 1 \right| & \text{if } p = \infty, \\
\left( \frac{1}{m} \sum_{1 \leq i \leq m} \left| \frac{(x^j_i)^2}{(a\eta(t_j))^2} + \frac{(y^j_i)^2}{(b\eta(t_j))^2} - 1 \right| \right)^{1/p} & \text{if } 1 \leq p < \infty,
\end{cases}
\]
where $X^j_i = (x^j_i, y^j_i)^\top$ is the $i$-th grid point. When we halve the discretization step $h$ the so-called EOC can be determined easily as follows:

$$\alpha_p(m) \approx \log_2 \left( \frac{E^j_p(m/2)}{E^j_p(m)} \right).$$

Now test the EOC for the example which initial curve is an ellipse with long semi-axis $a_0 = 0.6$, short semi-axis $b_0 = 0.3$. We begin with conduct the experiments using the numerical scheme (3.3). In this scheme, the shape function is chosen as $\varphi(k) = 1 - \varepsilon + \varepsilon \sqrt{1 - \varepsilon + \varepsilon k^2}$, where $\varepsilon = 0, 0.1, 0.5, 1$. From Table 4, we conclude that regardless of the value of $\varepsilon$, $\alpha \approx 2$. Next, the numerical scheme (3.7) is employed. Table 5 shows that the EOC obtained from the scheme (3.7) also satisfies $\alpha \approx 2$. This means that these two different numerical schemes seem to have the same advantage in terms of EOC.

### Table 4. Error and EOC using different $\varepsilon$ for numerical scheme (1.3).

<table>
<thead>
<tr>
<th>$m$</th>
<th>$\varepsilon$</th>
<th>$E_1(m)$</th>
<th>$\alpha_1(m)$</th>
<th>$E_2(m)$</th>
<th>$\alpha_2(m)$</th>
<th>$E_\infty(m)$</th>
<th>$\alpha_\infty(m)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0</td>
<td>0.000903</td>
<td>1.8394</td>
<td>0.000994</td>
<td>1.8137</td>
<td>0.001636</td>
<td>1.6820</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.001005</td>
<td>1.8316</td>
<td>0.001106</td>
<td>1.8037</td>
<td>0.001908</td>
<td>1.6815</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.001652</td>
<td>1.6871</td>
<td>0.001802</td>
<td>1.6668</td>
<td>0.003265</td>
<td>1.5489</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.001005</td>
<td>1.8316</td>
<td>0.001106</td>
<td>1.8037</td>
<td>0.001908</td>
<td>1.6815</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0.000824</td>
<td>1.9521</td>
<td>0.000920</td>
<td>1.9594</td>
<td>0.001561</td>
<td>2.0010</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.000952</td>
<td>1.9494</td>
<td>0.001065</td>
<td>1.9482</td>
<td>0.001730</td>
<td>1.9588</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.001468</td>
<td>1.9589</td>
<td>0.001655</td>
<td>1.9603</td>
<td>0.002653</td>
<td>1.9612</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.001468</td>
<td>1.9589</td>
<td>0.001655</td>
<td>1.9603</td>
<td>0.002653</td>
<td>1.9612</td>
</tr>
<tr>
<td>64</td>
<td>0</td>
<td>0.001201</td>
<td>1.9434</td>
<td>0.001406</td>
<td>1.9381</td>
<td>0.002827</td>
<td>1.9366</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.001492</td>
<td>1.9362</td>
<td>0.001749</td>
<td>1.9343</td>
<td>0.003336</td>
<td>1.9337</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.002509</td>
<td>1.9278</td>
<td>0.002933</td>
<td>1.9283</td>
<td>0.005326</td>
<td>1.9278</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.001468</td>
<td>1.9589</td>
<td>0.001655</td>
<td>1.9603</td>
<td>0.002653</td>
<td>1.9612</td>
</tr>
<tr>
<td>128</td>
<td>0</td>
<td>0.001340</td>
<td>1.9924</td>
<td>0.001544</td>
<td>1.9769</td>
<td>0.003214</td>
<td>1.9480</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.001653</td>
<td>1.9778</td>
<td>0.001962</td>
<td>1.9663</td>
<td>0.004133</td>
<td>1.9587</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.002738</td>
<td>1.9614</td>
<td>0.003306</td>
<td>1.9548</td>
<td>0.006887</td>
<td>1.9528</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.001468</td>
<td>1.9589</td>
<td>0.001655</td>
<td>1.9603</td>
<td>0.002653</td>
<td>1.9612</td>
</tr>
</tbody>
</table>

Finally, we compare the numerical results acquired from two different schemes, (3.3) and (3.7). According to calculations, each time step takes
Table 5. Error and EOC for UI distribution.

<table>
<thead>
<tr>
<th>m</th>
<th>$E_1 (m)$</th>
<th>$\alpha_1 (m)$</th>
<th>$E_2 (m)$</th>
<th>$\alpha_2 (m)$</th>
<th>$E_\infty (m)$</th>
<th>$\alpha_\infty (m)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.009059</td>
<td>0.009405</td>
<td></td>
<td></td>
<td></td>
<td>0.010494</td>
</tr>
<tr>
<td>32</td>
<td>0.001826</td>
<td>2.3107</td>
<td>0.001910</td>
<td>2.2999</td>
<td></td>
<td>1.9342</td>
</tr>
<tr>
<td>64</td>
<td>0.000433</td>
<td>2.0768</td>
<td>0.000459</td>
<td>2.0567</td>
<td></td>
<td>1.9879</td>
</tr>
<tr>
<td>128</td>
<td>0.000108</td>
<td>2.0061</td>
<td>0.000113</td>
<td>2.0164</td>
<td>0.000173</td>
<td>1.9979</td>
</tr>
</tbody>
</table>

Table 6. The CPU times for different methods when $t = 0.1$.

<table>
<thead>
<tr>
<th>m</th>
<th>scheme(3.6)</th>
<th>scheme(3.4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UI distribution</td>
<td>Non-UI distribution</td>
</tr>
<tr>
<td>16</td>
<td>0.03546</td>
<td>0.06491</td>
</tr>
<tr>
<td>32</td>
<td>0.05263</td>
<td>0.07511</td>
</tr>
<tr>
<td>64</td>
<td>0.16577</td>
<td>0.20118</td>
</tr>
<tr>
<td>128</td>
<td>0.39278</td>
<td>0.47270</td>
</tr>
</tbody>
</table>

$0.1/m^2$ seconds, where $m$ denotes the number of grid points. The computation times for each scheme are recorded in Table 6. It is clear from Table 6 that employing scheme (3.3) leads to slower CPU time compared to using scheme (3.7). The main reason is that scheme (3.3) requires considering the position vector, support function, and curvature at each iteration, while scheme (3.7) only requires knowing the support function and curvature at each step, and then calculating the position vector in the final step. Moreover, scheme (3.7) satisfies the relation $\alpha = -k^{-1}\beta$. Furthermore, when implementing scheme (3.3), we also choose the same tangential velocity $\alpha = -k^{-1}\beta$, which means considering CCR. By comparing the numerical results of CCR and AUTR in Table 6, it becomes evident that including the tangential velocity $\alpha = -k^{-1}\beta$ significantly improves efficiency. And when implementing scheme (3.7), we determine the initial positions based on a uniform or non-uniform distribution of the normal angles. We observe from Table 6 that UI distribution has slightly shorter CPU time compared to Non-UI distribution. This is because the UI distribution is more stable and has a more reasonable grid distribution.
5. The self-similar solution

Any self-similar solution under the flow (1.4) satisfying the equation [14, 28]

\[ A_1 - \frac{A_2}{2} \frac{h_\theta}{h} = \frac{1}{2} \log \frac{k}{h^3}. \]

The system has two parameters, \( A_1 \) and \( A_2 \), which determine the type of the self-similar motion. All possible values of \( A_1 \) and \( A_2 \) are divided into five cases which yield curves with significantly different properties and behaviour. Those are:

- \( A_1 > 0 \) and \( A_2 \neq 0 \): Curves which rotate and expand.
- \( A_1 < 0 \) and \( A_2 \neq 0 \): Curves which rotate and shrink.
- \( A_1 = 0 \) and \( A_2 \neq 0 \): Curves which only rotate.
- \( A_1 > 0 \) and \( A_2 = 0 \): Expanding curves.
- \( A_1 < 0 \) and \( A_2 = 0 \): Shrinking curves.

We do not consider the case \( A_1 = A_2 = 0 \) because in this case the self-similar solution is a straight line. In the case of \( A_2 = 0 \), we find that the centro-affine curvature \( \frac{k}{h^3} \) is constant, which implies the curve is an ellipse. As depicted in Figure 6, the evolving ellipse under the normal velocity (1.4) exhibits only scaling behavior. However, for the case \( A_2 \neq 0 \), the self-similar solution not only scales with time but also involves rotation around the origin. Those self-similar solutions with rotational behavior rotate forever at a constant angular speed \( -\frac{A_2}{2} \). To numerically implement the evolving process of the self-similar solution with rotation, we introduce a specific tangential velocity that aligns with

\[ \theta_t = -\frac{A_2}{2}. \]

Hence, combining (2.6) and (5.1), we easily derive the evolution equation of the support function as

\[ h_t = \frac{h}{2} \log h^3 (h + h_{\theta\theta}) - \frac{A_2}{2} h_{\theta}. \]

By taking \( V = \theta + \frac{A_2}{2} t \), we derive \( V_t = 0 \) and

\[ h_t = \frac{h}{2} \log h^3 (h + h_{VV}) - \frac{A_2}{2} h_V. \]

One obtains the fully discretized formulation as follows:

\[ \frac{h^j - h^{j-1}}{\tau} = -\frac{h^{j-1}}{2} \log \left( (h^{j-1})^3 (h^{j-1} + dh^{j-1}) \right) - \frac{A_2}{2} dh^{j-1}, \]

The tangential velocity takes the following form:

\[ \alpha = -\frac{A_2}{2k} - \frac{\beta_s}{k}. \]

In the subsequent numerical experiments, we employ the tangential velocity (5.5) and validate its effectiveness in the case when \( A_2 \neq 0 \). As shown in Figure 7, when \( A_1 = 0 \), the self-similar solution rotates around the origin.
Furthermore, Figure 8 and Figure 9 demonstrate that over time, the self-similar solution not only rotates around the origin but also expands outward \((A_1 < 0)\) or contracts inward \((A_1 > 0)\). For examining the self-similarity,

![Figure 6. The evolving ellipse when \(t = M\tau\), where \(M\) stands for the number of time steps. Numerical parameters: \(m = 64, \tau = 0.0001\).](image)

Table 7. The ratio of the minimum to maximum \(K_t^i\) for ellipse.

<table>
<thead>
<tr>
<th>(A_1)</th>
<th>(A_2)</th>
<th>(t = 0.1)</th>
<th>(t = 0.2)</th>
<th>(t = 0.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0</td>
<td>0.99023454667799</td>
<td>0.99201425503936</td>
<td>0.993737561177378</td>
</tr>
<tr>
<td>-0.5</td>
<td>0</td>
<td>0.9902348301491</td>
<td>0.99201484780883</td>
<td>0.99373853828743</td>
</tr>
</tbody>
</table>

we introduce the expression of \(i\)-th point at time \(t = t_j\):

\[
K_i^j = \frac{k_i^j}{k_i^0}, \quad i = 1, ..., m.
\]

As long as the shape of the curve remains, whether the curve is scaling or rotating, the value \(K_i^j\) at any time \(t\) is constant. From Table 7 and Table 8, it is evident that the ratio of the minimum to maximum \(K_i^t\) at time \(t\) remains close to 1 regardless of the values of \(A_1\) and \(A_2\). Obviously, the approach using the tangential velocity (5.5) is effective because of its self-similarity.

6. Conclusion

The purpose of this article is to propose a novel method that can visualize the evolutionary process of curve flow at a faster rate compared to other methods. To begin with, we demonstrate the feasibility of this proposed method through experimental results. For this aim, it is natural to
Figure 7. The evolving curve of self-similar solution when $t = M\tau$ in the case $A_1 = 0, A_2 = -5$. Numerical parameters: $m = 64, \tau = 0.005$.

Table 8. The ratio of the minimum to maximum $K^t_i$ for the self-similar solution which has rotating behavior.

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$t = 0.5$</th>
<th>$t = 1.0$</th>
<th>$t = 1.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-5</td>
<td>0.99985575120724</td>
<td>0.99971149926942</td>
<td>0.99956724419073</td>
</tr>
<tr>
<td>-0.5</td>
<td>-5</td>
<td>0.99610165628884</td>
<td>0.99218588426807</td>
<td>0.98825284337907</td>
</tr>
<tr>
<td>0.5</td>
<td>-5</td>
<td>0.98916411309595</td>
<td>0.97852921607608</td>
<td>0.96809071642409</td>
</tr>
</tbody>
</table>
Figure 8. The evolving curve of self-similar solution when $t = M\tau$ in the case $A_1 = 0.5$, $A_2 = -5$. Numerical parameters: $M$ which signifies the number of time steps, $m = 64$ which signifies the number of grid points, $\tau = 0.005$ which signifies the time step.

experimentally verify the conclusion that a curve approaches an ellipse under the flow (1.4) regardless of its initial closed convex shape. In addition, the self-similarity of the self-similar solutions is validated by simulating the evolutionary process. In order to prove the advantages of our method, we split the experiments into two cases: experiments done by scheme (3.6) and scheme (3.4), respectively. And some details can be seen in Table 6.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under grant-11631007 and grant-11971251, and by the Fundamental Research Funds for the Central Universities under grantN2104007.
Figure 9. The evolving curve of self-similar solution when $t = M \tau$ in the case $A_1 = -0.5$, $A_2 = -5$. Numerical parameters: $m = 64$, $\tau = 0.005$.

References


Fan Xia  
Department of Mathematics, Northeastern University, Shenyang, 110819, P.R. China  
Email address: 2100139@stu.neu.edu.cn

YanHua Yu  
Department of Mathematics, Northeastern University, Shenyang, 110819, P.R. China  
Email address: yyh_start@126.com