Advances in Computational Mathematics (2024) 50:74
https://doi.org/10.1007/s10444-024-10144-5

A continuation method for fitting a bandlimited curve to points in the plane

Mohan Zhao[1] · **Kirill Serkh**[2]

Received: 15 June 2023 / Accepted: 18 April 2024 / Published online: 16 July 2024

Abstract

In this paper, we describe an algorithm for fitting an analytic and bandlimited closed or open curve to interpolate an arbitrary collection of points in $\mathbb{R}^2$. The main idea is to smooth the parametrization of the curve by iteratively filtering the Fourier or Chebyshev coefficients of both the derivative of the arc-length function and the tangential angle of the curve and applying smooth perturbations, after each filtering step, until the curve is represented by a reasonably small number of coefficients. The algorithm produces a curve passing through the set of points to an accuracy of machine precision, after a limited number of iterations. It costs $O(N \log N)$ operations at each iteration, provided that the number of discretization nodes is N. The resulting curves are smooth, affine invariant, and visually appealing and do not exhibit any ringing artifacts. The bandwidths of the constructed curves are much smaller than those of curves constructed by previous methods. We demonstrate the performance of our algorithm with several numerical experiments.

Keywords Parametrization · Bandlimited functions · C^∞ functions · Approximation theory · Filtering · Bézier splines · Smooth interpolation

Mathematics Subject Classification (2010) 65D10 · 65D18 · 68U05

Communicated by: Zydrunas Gimbutas

✉ Mohan Zhao
mohan.zhao@mail.utoronto.ca

Kirill Serkh
kserkh@math.toronto.edu

[1] Department of Computer Science, University of Toronto, 40 St George St, M5S 2E4 Toronto, Ontario, Canada

[2] Department of Mathematics and Computer Science, University of Toronto, 40 St George St, M5S 2E4 Toronto, Ontario, Canada

1 Introduction

The construction of smooth curves passing through data points has uses in many areas of applied science, including boundary integral equation methods, computer graphics, and geometric modeling. While much of the time, C^k continuity is sufficient, there are certain applications for which C^∞ continuity is essential. One such example is the high-accuracy solution of partial differential equations on general geometries. In CAD/CAM systems, C^∞ smooth curves can be used as primitives to construct arbitrary smooth objects. Solving partial differential equations on these smooth objects prevents the loss of accuracy due to the imperfect smoothness of C^k shapes.

Countless methods have been proposed for fitting a spline or a C^k curve to a given set of data points. Most interpolation techniques use piecewise polynomials and impose constraints to ensure global C^k smoothness of the curve (see, for example, [1], [2], [3], [4]). In CAD/CAM systems, non-uniform rational B-splines (NURBS) are commonly used to construct a curve which approximates a set of control points, by defining the curve as a linear combination of the control points multiplied by C^k and compactly supported B-spline basis functions. The contribution of each control point to the overall curve is determined by the corresponding weight, and the B-spline basis functions are normalized to ensure that the approximating curve remains affine invariant (see, for example, [5]). A generalization of NURBS, called partition of unity parametrics (PUPs), was first introduced in Runions and Samavati [6]. The PUP curves are constructed by replacing the weighted B-spline basis functions with arbitrary normalized weight functions (WFs), so that the resulting curves still exhibit the desired properties, including compact support and C^k smoothness. In [6], the authors specifically discuss uniform B-spline WFs, to illustrate that each WF can be adjusted independently to fine-tune various shape parameters of the curve. Additionally, they observe that it is possible to choose the WFs to generate a PUP curve that interpolates the control points without solving a system of equations.

Another method proposed by Zhang and Ma [7] employs products of the sinc function and Gaussian functions as basis functions for constructing C^∞ interpolating curves that pass through all the given data points exactly. The resulting curves are almost affine invariant and almost compactly supported, and their shapes can be adjusted locally by directly adding or moving control points. Subsequently, Runions and Samavati [8] designed CINPACT-splines, by employing C^∞ and compactly supported bump functions as the WFs in a PUP curve, optionally multiplied by the normalized sinc function. When the WFs are chosen to be products of bump functions and the normalized sinc function, the resulting C^∞ curve interpolates the control points exactly, and when the WFs are bump functions, the resulting C^∞ curve approximates a uniform B-spline with the given control points. In addition to the properties inherited from PUP curves, CINPACT-splines possess C^∞ smoothness and the ability to specify tangents at control points. To increase the accuracy of the approximation to uniform B-splines, Akram et al. [9] further proposed CINAPACT-splines, by successively convolving a CINPACT-spline with B-splines of order one, ensuring any

finite order of approximation to uniform B-splines, as well as to other compactly supported kernels with maximal order and minimal support ([10]), while preserving C^∞ smoothness and compact support. Zhu [11] proposed curves that share similarities with CINPACT-splines in terms of affine invariance, compact support, and C^∞ smoothness. In [11], a class of non-negative blending functions is constructed by designing basis functions which combine bump functions with the sinc function. The resulting interpolating curves are defined by three local shape parameters, with one of the parameters determining whether the curve approximates or interpolates the given control points.

One notable distinction of the approach of [7] from the other methods we have discussed is that, since Gaussian functions are utilized in the basis functions, the interpolating curves produced by [7] are not only C^∞ smooth, but are also analytic. This paper mainly compares our method with [7], as the interpolating curves in [7] have a smaller bandwidth, compared to methods based on C^∞ compactly supported bump functions. The approach in [7] (as well as [6], [8], [9], [11]) necessitates a more specially chosen distribution of data points to achieve a visually smooth curve, as it only guarantees smoothness in the curve parameter, which does not necessarily correspond to the smoothness of the curve in $\mathbb{R}^2$. However, our method directly smooths the tangential angle of the curve and the first derivative of the arc-length function, yielding a significantly smoother curve which is also more visually appealing.

Among all the methods for constructing a C^∞ interpolating curve, the algorithm described by Beylkin and Rokhlin [12] bears the closest resemblance to our method, generating a bandlimited closed curve through a set of data points. The bandlimited curve is constructed by filtering the Fourier coefficients of the tangential angle of the curve, parametrized by the arc-length. However, the number of coefficients required to represent the curve can be large, which appears to be a major drawback of the algorithm in practical applications.

In this paper, we describe an algorithm for fitting a bandlimited closed or open curve to pass through a collection of points. The main idea is to iteratively filter the tangential angle and the first derivative of the arc-length function of the curve and apply small corrections after each filtering step, until the desired bandwidth of the curve is reached, to the required precision. Our algorithm produces an analytic and affine invariant curve with far fewer coefficients, and the curve is visually appealing and free of ringing artifacts.

The structure of this paper is as follows. Section 2 describes the mathematical preliminaries. Section 3 describes the algorithm to construct the bandlimited approximation to a closed curve, and to an open curve. Finally, Sect. 4 presents several numerical examples to demonstrate the performance of our algorithm, as well as some comparisons between our algorithm and the methods proposed in [7] and [12].

2 Preliminaries

In this section, we describe the mathematical and numerical preliminaries.

2.1 Geometric properties of a curve

Let $\gamma : [a, b] \to \mathbb{R}^2$ be a smooth curve parametrized by the curve parameter t, such that

$$\gamma(t) = (x(t), y(t)), \qquad t \in [a, b], \tag{1}$$

where $x(t)$ and $y(t)$ are the x and y coordinates.

Assuming $\gamma \in C^1([a, b])$, we define the tangent vector $T(t)$,

$$T(t) = (x'(t), y'(t)), \qquad t \in [a, b], \tag{2}$$

and the arc-length $s(t)$, which is the length of the curve from the point $(x(a), y(a))$ to the point $(x(t), y(t))$,

$$s(t) = \int_a^t \|T(\tau)\| d\tau, \qquad t \in [a, b]. \tag{3}$$

It is obvious that

$$s'(t) = \|T(t)\|, \qquad t \in [a, b]. \tag{4}$$

Thus, we have

$$s'(b) = s'(a) \tag{5}$$

when the curve is closed. The tangential angle $\theta(t)$ of the curve at the point $(x(t), y(t))$ measures the angle between the tangent vector $T(t)$ at that point and the x-axis, defined by the formula

$$\theta(t) = \operatorname{atan2}(y'(t), x'(t)), \qquad t \in [a, b], \tag{6}$$

where $\operatorname{atan2}: \mathbb{R}^2 \to (-\pi, \pi]$ is the arctangent at the point $(x(t), y(t))$. As a result, $\theta(t) \in (-\pi, \pi]$. Since the function atan2 has a branch cut at $\theta = -\pi$, it is possible for $\theta(t)$ to have ω jump discontinuities of size 2π, where $\omega \in \mathbb{Z}$ is the winding number.

The curve $(x(t), y(t))$ can be constructed from $\theta(t)$ and $s'(t)$ by the formulas

$$x(t) = \int_a^t s'(\tau) \cos\theta(\tau)\, d\tau + x(a), \qquad t \in [a, b], \tag{7}$$

$$y(t) = \int_a^t s'(\tau) \sin\theta(\tau)\, d\tau + y(a), \qquad t \in [a, b], \tag{8}$$

and $(x(a), y(a)) = \gamma(a)$. If the curve is closed, we require $x(a) = x(b)$ and $y(a) = y(b)$, which means that

$$\int_a^b s'(\tau) \cos\theta(\tau)\, d\tau = 0 \tag{9}$$

and that

$$\int_a^b s'(\tau)\sin\theta(\tau)\,d\tau = 0. \tag{10}$$

2.2 Cubic Bézier interpolation

A Bézier curve is a function $\mathbf{B}\colon [0,1] \to \mathbb{R}^2$ defined by a set of control points $\mathbf{P}_0$, …, $\mathbf{P}_m \in \mathbb{R}^2$. The Bézier curve is designed to go through the first and the last control points $\mathbf{P}_0$ and $\mathbf{P}_m$, and the shape of the curve is determined by the intermediate control points $\mathbf{P}_1$, …, $\mathbf{P}_{m-1}$. A mth order Bézier curve is a polynomial of degree m, defined by

$$\begin{aligned}\mathbf{B}(t) &= \sum_{i=0}^{m}\binom{m}{i}(1-t)^{m-i}t^i\mathbf{P}_i,\\ &= (1-t)^m\mathbf{P}_0 + \binom{m}{1}(1-t)^{m-1}t\mathbf{P}_1 + \ldots\\ &\quad + \binom{m}{m-1}(1-t)t^{m-1}\mathbf{P}_{m-1} + t^m\mathbf{P}_m,\end{aligned} \tag{11}$$

where $t \in [0,1]$.

A continuous Bézier spline connecting all the given points $\mathbf{C}_0$, …, $\mathbf{C}_n$ can be constructed by combining n cubic Bézier curves

$$\mathbf{B}_i(t) = (1-t)^3\mathbf{P}_{i0} + 3(1-t)^2t\mathbf{P}_{i1} + 3(1-t)t^2\mathbf{P}_{i2} + t^3\mathbf{P}_{i3}, \qquad i = 1,\ldots,n, \tag{12}$$

where $t \in [0,1]$ and $\mathbf{B}_i(t)$ is the ith Bézier curve, with controls points

$$\mathbf{P}_{i0} = \mathbf{C}_{i-1}, \tag{13}$$

$$\mathbf{P}_{i3} = \mathbf{C}_i, \tag{14}$$

for $i = 1, \ldots, n$. We define the spline $\mathbf{S}\colon [0,n] \to \mathbb{R}^2$ from the cubic Bézier curves $\mathbf{B}_i$ by letting $\mathbf{S}(t) = \mathbf{B}_i(t-i+1)$ for $t \in [i-1, i]$, for $i = 1, \ldots, n$. It is easy to see that $\mathbf{S} \in C^0([0,n])$, however, in general, $\mathbf{S} \notin C^1([0,n])$. It is possible to ensure $\mathbf{S} \in C^2([0,n])$ by imposing additional conditions on the intermediate control points, which we derive as follows. Note that the following derivation is similar to the one presented in Joost [13]. First, we observe that the first and second derivatives of a cubic Bézier curve are

$$\mathbf{B}_i'(t) = -3(1-t)^2\mathbf{P}_{i0} + 3(3t^2-4t+1)\mathbf{P}_{i1} + 3t(2-3t)\mathbf{P}_{i2} + 3t^2\mathbf{P}_{i3}, \tag{15}$$

$$\mathbf{B}_i''(t) = 6(1-t)\mathbf{P}_{i0} + 6(3t-2)\mathbf{P}_{i1} + 6(1-3t)\mathbf{P}_{i2} + 6t\mathbf{P}_{i3}, \tag{16}$$

for $i = 1, \ldots, n$. In order for $\mathbf{S} \in C^2([0,n])$, we require that

$$\mathbf{B}_{i-1}'(1) = \mathbf{B}_i'(0), \qquad i = 1,\ldots,n, \tag{17}$$

$$\mathbf{B}''_{i-1}(1) = \mathbf{B}''_i(0), \qquad i = 1, \ldots, n. \tag{18}$$

Then, Eq. 17 implies that

$$\mathbf{P}_{(i-1)2} = 2\mathbf{C}_{i-1} - \mathbf{P}_{i1}, \qquad i = 1, \ldots, n. \tag{19}$$

Likewise, it is possible to show that Eq. 18 implies that

$$\mathbf{P}_{(i-1)1} + 2\mathbf{P}_{i1} = \mathbf{P}_{i2} + 2\mathbf{P}_{(i-1)2}, \qquad i = 1, \ldots, n. \tag{20}$$

Substituting Eq. 19 into Eq. 20, we get

$$\mathbf{P}_{(i-1)1} + 4\mathbf{P}_{i1} + \mathbf{P}_{(i+1)1} = 2\mathbf{C}_i + 4\mathbf{C}_{i-1}, \qquad i = 1, \ldots, n. \tag{21}$$

2.2.1 Solving for the control points for an open curve

When the curve is open, we have that Eq. 21 must hold for $i = 2, \ldots, n-1$, and we need two boundary conditions in order to solve a linear system of n equations for the values of $\mathbf{P}_{11}, \ldots, \mathbf{P}_{n1}$. Assuming that the slopes at the two endpoints of the curve, c_{left} and c_{right}, are given, we have

$$\mathbf{B}'_1(0) = c_{\text{left}}, \tag{22}$$

and

$$\mathbf{B}'_n(1) = c_{\text{right}}. \tag{23}$$

It is possible to show that Eq. 22 implies that

$$\mathbf{P}_{11} = \frac{c_{\text{left}} + 3\mathbf{C}_0}{3} \tag{24}$$

and Eq. 23 implies that

$$\mathbf{P}_{n2} = \frac{3\mathbf{C}_n - c_{\text{right}}}{3}. \tag{25}$$

Substituting Eqs. 19 and 25 into Eq. 20, we get

$$\mathbf{P}_{(n-1)1} + 4\mathbf{P}_{n1} = 4\mathbf{C}_{n-1} + \mathbf{C}_n - \frac{c_{\text{right}}}{3}. \tag{26}$$

With Eqs. 21, 24, and 26, we build a system of n equations to calculate $\mathbf{P}_{11}, \ldots, \mathbf{P}_{n1}$ and use Eq. 19, Eq. 25 and the values of $\mathbf{P}_{11}, \ldots, \mathbf{P}_{n1}$ to calculate $\mathbf{P}_{12}, \ldots, \mathbf{P}_{n2}$. This system of equations is tridiagonal, and so can be solved in $O(n)$ operations.

2.2.2 Solving for the control points for a closed curve

When the curve is closed, we require $n+1$ cubic Bézier curves instead of n cubic Bézier curves to connect the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$, where the $(n+1)$th curve connects the points $\mathbf{C}_n$ and $\mathbf{C}_0$. We have that the conditions Eq. 21 must hold for $i = 2, \ldots, n$, and we need the following two boundary conditions,

$$\mathbf{B}_1'(0) = \mathbf{B}_{n+1}'(1), \tag{27}$$
$$\mathbf{B}_1''(0) = \mathbf{B}_{n+1}''(1), \tag{28}$$

to solve a linear system of $(n+1)$ equations for the values of $\mathbf{P}_{11}, \ldots, \mathbf{P}_{(n+1)1}$.

It is possible to show that Eq. 27 implies that

$$\mathbf{P}_{11} + \mathbf{P}_{(n+1)2} = 2\mathbf{C}_0 \tag{29}$$

and Eq. 28 implies that

$$-2\mathbf{P}_{11} + \mathbf{P}_{12} = \mathbf{P}_{(n+1)1} - 2\mathbf{P}_{(n+1)2}. \tag{30}$$

Substituting Eqs. 19 and 20 into Eq. 29, we get

$$\mathbf{P}_{11} + \mathbf{P}_{n1} + 4\mathbf{P}_{(n+1)1} = 2\mathbf{C}_0 + 4\mathbf{C}_n. \tag{31}$$

Substituting Eqs. 19 and 20 into Eq. 30, we get

$$-2\mathbf{P}_{11} - \mathbf{P}_{21} + 2\mathbf{P}_{n1} + 7\mathbf{P}_{(n+1)1} = -2\mathbf{C}_1 + 8\mathbf{C}_n. \tag{32}$$

Similarly, with Eqs. 21, 31 and 32, we build a system of $(n+1)$ equations to calculate $\mathbf{P}_{11}, \ldots, \mathbf{P}_{(n+1)1}$ and use Eq. 19, Eq. 29 and the values of $\mathbf{P}_{11}, \ldots, \mathbf{P}_{(n+1)1}$ to calculate $\mathbf{P}_{12}, \ldots, \mathbf{P}_{(n+1)2}$. This system of equations is cyclic tridiagonal, and thus, we can solve it in $O(n)$ operations.

2.3 Chebyshev polynomial interpolation

A smooth function $f(x)$ on the interval $[-1, 1]$ can be approximated by a $(n-1)$th order Chebyshev expansion with the formula

$$f(x) \approx \sum_{k=0}^{n-1} \hat{f}_k T_k(x), \tag{33}$$

where $T_k(x)$ is the Chebyshev polynomial of the first kind of degree k, defined by

$$T_k(x) = \cos(k \arccos x), \qquad x \in [-1, 1]. \tag{34}$$

It is known that the Chebyshev coefficients $\{\hat{f}_k\}$ decay like $O(n^{-m+\frac{1}{2}})$ when $f \in C^m([-1, 1])$ and the coefficients $\hat{f}_k$ are chosen to satisfy the n collocation equations

$$f(x_i) = \sum_{k=0}^{n-1} \hat{f}_k T_k(x_i), \qquad i = 0, \ldots, n-1, \tag{35}$$

for the practical Chebyshev nodes $\{x_i\}$,

$$x_i = -\cos\Big(\frac{i\pi}{n-1}\Big), \qquad i = 0, \ldots, n-1. \tag{36}$$

Alternatively, one can compute $\hat{f}_k$ for $k = 0, \ldots, n-1$ using the discrete Chebyshev transform,

$$\hat{f}_0 = \frac{1}{n-1}\Big(\frac{1}{2}(f(x_0) + f(x_{n-1})) + \sum_{i=1}^{n-2} f(x_i)T_0(x_i)\Big), \tag{37}$$

and

$$\hat{f}_k = \frac{2}{n-1}\Big(\frac{1}{2}(f(x_0)(-1)^k + f(x_{n-1})) + \sum_{i=1}^{n-2} f(x_i)T_k(x_i)\Big), \tag{38}$$

for $k = 1, \ldots, n-1$. Once the coefficients $\{\hat{f}_k\}$ are computed, we can use the expansion $\sum_{k=0}^{n-1} \hat{f}_k T_k(x)$ to evaluate $f(x)$ everywhere on the interval $[-1, 1]$.

2.3.1 Spectral differentiation and integration

Assuming that $k \geq 1$ is an integer, the formula

$$2T_k(x) = \frac{T'_{k+1}(x)}{k+1} - \frac{T'_{k-1}(x)}{k-1}, \tag{39}$$

can be used to spectrally differentiate the Chebyshev expansion of $f(x)$, as follows. Suppose that

$$f(x) \approx \sum_{k=0}^{n-1} \hat{f}_k T_k(x) \tag{40}$$

and that

$$f'(x) \approx \sum_{k=0}^{n-1} \hat{f}'_k T_k(x). \tag{41}$$

The coefficients $\hat{f}'_k$ can be computed from $\hat{f}_k$ by iterating from $k = n-1, n-2, \ldots, 2$ and, at each iteration, assigning $\hat{f}'_{k-1}$ the value $2k\hat{f}_k$, and assigning $\hat{f}_{k-2}$ the value $\frac{k}{k-2}\hat{f}_k + \hat{f}_{k-2}$.

Similarly, the formula

$$2\int_{-1}^{t} T_k(x)\,dx = \frac{T_{k+1}(t)}{k+1} - \frac{T_{k-1}(t)}{k-1} - \frac{(-1)^{k+1}}{k+1} + \frac{(-1)^{k-1}}{k-1} \tag{42}$$

can be used to spectrally integrate the Chebyshev expansion of $f(x)$. Suppose that

$$\int_{-1}^{t} f(x)\,dx \approx \sum_{k=0}^{n} \hat{\hat{f}}_k T_k(t). \tag{43}$$

Since

$$\begin{aligned}\int_{-1}^{t} f(x)dx &\approx \sum_{k=0}^{n-1} \hat{f}_k \int_{-1}^{t} T_k(x)\,dx \\ &= \sum_{k=1}^{n-1} \hat{f}_k \frac{1}{2}\Big(\frac{T_{k+1}(t)}{k+1} - \frac{T_{k-1}(t)}{k-1} - \frac{(-1)^{k+1}}{k+1} + \frac{(-1)^{k-1}}{k-1}\Big) \\ &\quad + \hat{f}_0(t+1),\end{aligned} \tag{44}$$

one can compute the coefficients $\hat{\hat{f}}_k$ from $\hat{f}_k$ by first assigning $\hat{f}_1$ the value $\hat{f}_1 + \hat{f}_0$, then iterating from $k = n-1, \ldots, 1$ and, at each iteration, assigning $\hat{f}_{k+1}$ the value $\hat{f}_{k+1} + \frac{\hat{f}_k}{2(k+1)}$, assigning $\hat{f}_{k-1}$ the value $\hat{f}_{k-1} - \frac{\hat{f}_k}{2(k-1)}$, and assigning $\hat{f}_0$ the value $\hat{f}_0 - \hat{f}_k(\frac{(-1)^{k+1}}{2(k+1)} - \frac{(-1)^{k-1}}{2(k-1)})$. Finally, $\hat{\hat{f}}_k$ takes the value $\hat{f}_k$, for $k = n, \ldots, 0$.

2.4 The discrete Fourier transform (DFT)

A periodic and smooth function $f(x)$ on the interval $[0, 1]$ can be approximated by a n-term Fourier series using the discrete Fourier transform. The discrete Fourier transform defines a transform from a sequence of n complex numbers $f_0, \ldots, f_{n-1}$ to another sequence of n complex numbers $\hat{f}_0, \ldots, \hat{f}_{n-1}$, by the formula

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-\frac{2\pi i}{n}kj}, \qquad k = 0, \ldots, n-1, \tag{45}$$

The sequence $\{\hat{f}_k\}$ consists of the Fourier coefficients of $\{f_k\}$.

The inverse discrete Fourier transform (IDFT) is given by

$$f_j = \frac{1}{n}\sum_{k=0}^{n-1} \hat{f}_k e^{\frac{2\pi i}{n}kj}, \qquad j = 0, \ldots, n-1. \tag{46}$$

Another representation of the DFT which is usually used in applications is given by a shift in the index k, and a change in the placement of the scaling by $\frac{1}{n}$,

$$\hat{f}_k = \frac{1}{n}\sum_{j=0}^{n-1} f_j e^{-\frac{2\pi i}{n}kj}, \qquad k = -\frac{n}{2}, \ldots, \frac{n}{2} - 1. \tag{47}$$

Thus, the corresponding IDFT is

$$f_j = \sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} \hat{f}_k e^{\frac{2\pi i}{n}kj}, \qquad j = 0, \ldots, n-1. \tag{48}$$

Suppose that $f : [0, 1] \to \mathbb{C}$ is a smooth and periodic function, and that $f_j = f(t_j)$ for $j = 0, \ldots, n-1$, where $\{t_j\}$ are the equispaced points on $[0, 1]$. Observing that

$$\begin{aligned} \hat{f}_k &= \frac{1}{n}\sum_{j=0}^{n-1} f_j e^{-\frac{2\pi i}{n}kj}, \\ &\approx \int_0^1 f(x) e^{-2\pi i k x}\, dx, \end{aligned} \tag{49}$$

for $k = -\frac{n}{2}, \ldots, \frac{n}{2} - 1$, we obtain the approximation to $f(x)$ by a truncated Fourier series,

$$f(x) \approx \sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} \hat{f}_k e^{2\pi i k x}, \qquad x \in [0, 1]. \tag{50}$$

It is known that the Fourier coefficients $\{\hat{f}_k\}$ decay like $O(n^{-m+\frac{1}{2}})$ when $f \in C^m(S^1)$, where $S^1 = [0, 1]$ is the circle.

2.4.1 Spectral differentiation and integration

A truncated Fourier series approximation to $f(x)$ on $[0, 1]$ can be spectrally differentiated as follows. Suppose that $f(x)$ is given by Eq. 50 and that

$$f'(x) \approx \frac{1}{n}\sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} \hat{f}'_k e^{2\pi i k x}. \tag{51}$$

Since

$$f'(x) \approx \frac{1}{n}\sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} \hat{f}_k e^{2\pi i k x} \cdot 2\pi i k, \tag{52}$$

the coefficients $\hat{f}'_k$ can be computed from $\hat{f}_k$ by assigning $\hat{f}'_k$ the value $\hat{f}_k \cdot 2\pi ik$ for $k = -\frac{n}{2}, \ldots, \frac{n}{2} - 1$.

Similarly, a truncated Fourier series approximation to $f(x)$ can be spectrally integrated as follows. Suppose that

$$\int_0^t f(x)dx \approx \frac{1}{n} \sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} \hat{\hat{f}}_k e^{2\pi ikt}. \tag{53}$$

Since

$$\int_0^t f(x)dx \approx \frac{1}{n} \sum_{k\neq 0} \frac{\hat{f}_k}{2\pi ik} e^{2\pi ikt} - \frac{1}{n} \sum_{k\neq 0} \frac{\hat{f}_k}{2\pi ik} + \frac{1}{n} \hat{f}_0 t, \tag{54}$$

it is easy to see that, for $\int_0^t f(x)dx$ to be periodic, it must be the case that $\hat{f}_0 = 0$. Then, we have

$$\int_0^t f(x)dx \approx \frac{1}{n} \sum_{k\neq 0} \frac{\hat{f}_k}{2\pi ik} e^{2\pi ikt} - \frac{1}{n} \sum_{k\neq 0} \frac{\hat{f}_k}{2\pi ik}. \tag{55}$$

We can compute the Fourier coefficients $\hat{\hat{f}}_k$ from $\hat{f}_k$ by assigning $\hat{\hat{f}}_k$ the value $\frac{\hat{f}_k}{2\pi ik}$ for $k = -\frac{n}{2}, \ldots, \frac{n}{2} - 1, k \neq 0$ and assigning $\hat{\hat{f}}_0$ the value $-\sum_{k\neq 0} \frac{\hat{f}_k}{2\pi ik}$.

2.5 Gaussian filter

A low-pass filter is commonly used in signal processing to construct a bandlimited function. In this paper, we use the Gaussian filter, which is a popular low-pass filter whose impulse response is a Gaussian function,

$$g(x) = ae^{-\pi a^2 x^2}, \tag{56}$$

where a determines the bandwidth of $g(x)$.

The Gaussian filter $g_0, \ldots, g_{n-1}$ is defined to be the IDFT of the sequence

$$\hat{g}_k = e^{-\pi \frac{k^2}{a^2}}, \qquad k = -\frac{n}{2}, \ldots, \frac{n}{2} - 1, \tag{57}$$

and coincides with the discrete values of $g(x)$ at the equispaced nodes $x_j = \frac{j}{n}$, $j = 0, \ldots, n-1$.

To filter the Fourier coefficients $\hat{f}_0, \ldots, \hat{f}_{n-1}$ in Eq. 47, we take the product

$$\hat{h}_k = \hat{g}_k \hat{f}_k, \qquad k = -\frac{n}{2}, \ldots, \frac{n}{2} - 1. \tag{58}$$

It is easily to obtain $h_0, \ldots, h_{n-1}$ by the IDFT,

$$h_j = \frac{1}{n} \sum_{k=0}^{j} g_k f_{j-k}, \qquad j = 0, \ldots, n-1. \tag{59}$$

This can be considered to be a smoothing of $f(x)$ by a convolution with the Gaussian function $g(x)$.

Filtering the Chebyshev coefficients $\hat{f}_0, \ldots, \hat{f}_{n-1}$ defined in Eq. 35 is very similar to filtering the Fourier coefficients, and proceeds as follows. Substituting $x = \cos(\theta)$, where $x \in [-1, 1]$, into Eq. 33, we have

$$\begin{aligned} f(\cos(\theta)) &\approx \sum_{k=0}^{n-1} \hat{f}_k T_k(\cos(\theta)) \\ &= \sum_{k=0}^{n-1} \hat{f}_k \cos(k\theta), \end{aligned} \tag{60}$$

where $\theta \in [-\pi, \pi]$. Letting $\hat{f}_{-k} = \hat{f}_k$, $k = 1, \ldots, n-1$, we have

$$f(\cos(\theta)) \approx \frac{1}{2} \sum_{k=-n+1}^{n-1} \hat{f}_k e^{ik\theta} + \frac{1}{2} \hat{f}_0, \qquad \theta \in [-\pi, \pi]. \tag{61}$$

Hence, by defining ϕ by the formula $\theta = 2\pi\phi$,

$$f(\cos(2\pi\phi)) \approx \frac{1}{2} \sum_{k=-n+1}^{n-1} \hat{f}_k e^{2\pi ik\phi} + \frac{1}{2} \hat{f}_0, \qquad \phi \in [-\tfrac{1}{2}, \tfrac{1}{2}]. \tag{62}$$

Since Eq. 62 can be viewed as a Fourier transform in ϕ with the Fourier coefficients $\{\hat{f}_k\}$, we follow Eq. 58 to filter $\{\hat{f}_k\}$, and apply the IDFT to obtain the filtered values of $\{f_j\}$. Therefore, $f(x)$ is smoothed by a convolution with the Gaussian function $g(\phi)$ in the ϕ-domain, where $x = \cos(2\pi\phi)$.

Alternatively, there are other low-pass filters that can be used, such as the Butterworth filter (see, for example, Chapter 14 of [14]) which resembles the Gaussian filter but is flatter in the passband. The brick-wall filter also preserves signals with lower frequencies and excludes signals with higher frequencies. However, after applying the brick-wall filter, the resulting functions tend to oscillate at the cutoff frequency (this phenomenon is known as ringing).

3 The algorithm

In this section, we give an overview of our algorithm for fitting a C^∞ curve to pass through a collection of points $\mathbf{C}_0, \ldots, \mathbf{C}_n$. We begin with a C^2 cubic Bézier spline

connecting the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$. Given that the curve is at least C^2, we interpolate the tangential angle $\theta(t)$ and the first derivative of the arc-length function $s'(t)$, which are both C^1, using Chebyshev expansions when the curve is open, or using truncated Fourier series when the curve is closed. We then iteratively filter the coefficients of $\theta(t)$ and $s'(t)$ by applying a Gaussian filter, whose bandwidth decreases with each iteration. If the curve is closed before filtering, we impose constraints on $\theta(t)$ and $s'(t)$ to ensure that the curve remains closed. We then reconstruct the curve with the filtered values of $\theta(t)$ and $s'(t)$ at discretization nodes.

While filtering leads to small discrepancies between the reconstructed curve and the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$, it also improves the bandwidth of the curve. To fix the discrepancies after each filtering step, we rotate and rescale the curve to minimize the total distance between the curve and the points and add small, smooth perturbations, which do not negatively affect the smoothness of the curve. We stop filtering when the desired bandwidths of the Chebyshev or Fourier approximations to $\theta(t)$ and $s'(t)$ are achieved. This algorithm gives us a C^∞ smooth curve that can be represented by a reasonably small number of coefficients. The resulting curve also maintains affine invariance.

This iterative filtering procedure can be viewed as a continuation method, producing a sequence of curves along a continuous path from large bandwidth to small bandwidth. Suppose that $F\big((\theta(t), s'(t)), \tilde{\lambda}\big)$ measures the extent to which $(\theta(t), s'(t))$ satisfies some bandlimit $\tilde{\lambda}$, and the extent to which the curve represented by $(\theta(t), s'(t))$ interpolates the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$. The goal of our algorithm is to find $(\theta_\lambda(t), s'_\lambda(t))$, such that $F\big((\theta_\lambda(t), s'_\lambda(t)), \lambda\big) = 0$, for some desired bandwidth λ. One approach is to start with an initial curve represented by $(\theta_{\lambda_0}(t), s'_{\lambda_0}(t))$, such that $F\big((\theta_{\lambda_0}(t), s'_{\lambda_0}(t)), \lambda_0\big) = 0$, where the bandwidth λ_0 can be much larger than the desired bandwidth λ, even ∞. If λ_0 is much larger than λ, applying a single step of filtering to reduce the bandwidth to λ will lead to large discrepancies between the reconstructed curve and the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$, and adding large perturbations will reduce the overall smoothness of the curve. Since a small filtering step results in only a small change in the curve, and adding small perturbations does not reduce the bandwidth of the curve, we filter the curve iteratively by slightly reducing the bandwidth of $(\theta_{\lambda_{i-1}}(t), s'_{\lambda_{i-1}}(t))$ from λ_{i-1} to λ_i at the ith iteration, so that we can ensure $F\big((\theta_{\lambda_i}(t), s'_{\lambda_i}(t)), \lambda_i\big) = 0$. Our method can be interpreted as a continuation method, following a continuous solution path $\big\{(\theta_{\tilde{\lambda}}(t), s'_{\tilde{\lambda}}(t))\big\}_{\tilde{\lambda}=\lambda_0}^{\lambda}$ under the constraint $F\big((\theta_{\tilde{\lambda}}(t), s'_{\tilde{\lambda}}(t)), \tilde{\lambda}\big) = 0$.

3.1 Initial approximation

To initialize our algorithm, we require a C^2 curve, the reasons for which are described in Sect. 3.3.

Given a set of data points $\mathbf{C}_0, \ldots, \mathbf{C}_n \in \mathbb{R}^2$, we fit a cubic Bézier spline by solving for the intermediate control points $\{\mathbf{P}_{i1}\}$ and $\{\mathbf{P}_{i2}\}$, as described in Sect. 2.2.1 for an open curve and in Sect. 2.2.2 for a closed curve. We define the Bézier spline $S\colon [0, L] \to \mathbb{R}^2$ connecting all the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$ by

$$\mathbf{S}(t) = \mathbf{B}_i(t - i + 1), \qquad t \in [0, n] \text{ and } i = 1, \ldots, n, \tag{63}$$

if the curve is open, or

$$\mathbf{S}(t) = \mathbf{B}_i(t - i + 1), \qquad t \in [0, n+1] \text{ and } i = 1, \ldots, n+1, \tag{64}$$

if the curve is closed.

3.2 Representation of the curve

In this section, we denote the curve by

$$\gamma(t) = (x(t), y(t)), \tag{65}$$

where $\gamma : [0, L] \to \mathbb{R}^2$ is at least C^2.

3.2.1 Representation of an open curve

When the curve is open, we discretize $x(t)$ and $y(t)$ at $N \gg n$ practical Chebyshev nodes $\{t_j\}$ on the interval $[0, L]$ (see Eq. 36) to obtain $\{x_j\}$ and $\{y_j\}$, where $x_j = x(t_j)$ and $y_j = y(t_j)$. We use $(N-1)$th order Chebyshev expansions to approximate $x(t)$ and $y(t)$, constructing the coefficients from $\{x_j\}$ and $\{y_j\}$ using the discrete Chebyshev transform, and then spectrally differentiate $x(t)$ and $y(t)$ to derive the Chebyshev expansions approximating $x'(t)$ and $y'(t)$. By Eqs. 4 and 6, we can compute the values of $s'(t)$ and $\theta(t)$ sampled at nodes $\{t_j\}$, and then construct the corresponding Chebyshev expansions, again using the discrete Chebyshev transform. However, performing the Chebyshev transform on $\theta(t)$ requires $\theta(t)$ to be continuous, and as discussed in Sect. 2.1, $\theta(t)$ can have jump discontinuities of size 2π. These can be fixed by adding or subtracting multiples of 2π to $\theta(t)$ wherever a discontinuity is detected.

3.2.2 Representation of a closed curve

When the curve is closed, we discretize $x(t)$ and $y(t)$ at $N \gg n$ equispaced nodes $\{t_j\}$ on the interval $[0, L]$, where

$$t_j = \frac{j}{N} L, \qquad j = 0, \ldots, N-1, \tag{66}$$

to obtain $\{x_j\}$ and $\{y_j\}$ by $x_j = x(t_j)$ and $y_j = y(t_j)$. We then approximate $x(t)$ and $y(t)$ by N-term Fourier series, separately, and spectrally differentiate $x(t)$ and $y(t)$ to approximate $x'(t)$ and $y'(t)$. Following the same procedure described in Sect. 3.2.1, we ensure that $\theta(t)$ is continuous and approximate $s'(t)$ by a truncated Fourier series. Recall that, in order to approximate functions by their Fourier series, the functions must be both smooth and periodic. The sequence $\{\theta_j\}$, which are the discrete values of $\theta(t)$ at $\{t_j\}$, is not periodic after being shifted by multiples of 2π to remove the discontinuities. Defining c by

$$c = \theta(n+1) - \theta(0), \tag{67}$$

we let

$$\tilde{\theta}_j = \theta_j - \frac{c}{L} t_j, \qquad t_j \in [0, L]. \tag{68}$$

This transforms $\{\theta_j\}$ into a periodic sequence $\{\tilde{\theta}_j\}$ on the interval $[0, L]$, which can be approximated by a truncated Fourier series. To recover the true values of $\{\theta_j\}$ after filtering, we can add $\frac{c}{L} t_j$ to $\tilde{\theta}_j$. In an abuse of notation, we denote $\{\tilde{\theta}_j\}$ by $\{\theta_j\}$ wherever the meaning is clear.

3.3 Filtering the curve

In this section, we describe the process of iteratively filtering $\theta(t)$ and $s'(t)$ using a Gaussian filter. Given $\gamma(t) \in C^2$, we have $\theta(t) \in C^1$ and $s'(t) \in C^1$. It is known that the decay rate of the Chebyshev coefficients or the Fourier coefficients of a C^1 function is $O(N^{-\frac{1}{2}})$, where N is the order of the expansion. By iteratively decreasing the bandwidth of the Gaussian filter, we construct a sequence of bandlimited representations of $\theta(t)$ and $s'(t)$. The decay rate of the Fourier coefficients or the Chebyshev coefficients in the expansions of $\theta(t)$ and $s'(t)$ increases with each iteration. This filtering process smooths both the curve itself and the parameterization of the curve.

3.3.1 Filtering the open curve

Let $\{t_j\}$ denote the practical Chebyshev nodes translated to the interval $[0, L]$ (see Eq. 36). Using the Chebyshev expansions of $\theta(t)$ and $s'(t)$ computed in Sect. 3.2.1, we discretize $\theta(t)$ and $s'(t)$ at the points $\{t_j\}$ to obtain the sequences $\{\theta_j\}$ and $\{s'_j\}$, where $\theta_j = \theta(t_j)$ and $s'_j = s'(t_j)$. We filter the Chebyshev coefficients $\{\hat{\theta}_k\}$ of $\theta(t)$ and $\{\hat{s'}_k\}$ of $s'(t)$ using the Gaussian filter in Eq. 57, and obtain the filtered coefficients $\{\hat{\theta}_k^{(f)}\}$ and $\{\hat{s}_k^{(f)}\}$,

$$\hat{\theta}_k^{(f)} = e^{-\pi \frac{k^2}{a^2}} \hat{\theta}_k, \qquad k = 0, \ldots, N-1, \tag{69}$$

and

$$\hat{s}_k^{(f)} = e^{-\pi \frac{k^2}{a^2}} \hat{s'}_k, \qquad k = 0, \ldots, N-1. \tag{70}$$

Applying the IDFT to $\{\hat{\theta}_k^{(f)}\}$ and $\{\hat{s}_k^{(f)}\}$, we obtain

$$\theta_j^{(f)} = \sum_{k=0}^{N-1} \hat{\theta}_k^{(f)} T_k(\bar{t}_j), \qquad \bar{t}_j \in [-1, 1], \tag{71}$$

where $\bar{t}_j = \frac{2}{L} t_j - 1$, $t_j \in [0, L]$, and

$$s_j'^{(f)} = \sum_{k=0}^{N-1} \hat{s}_k^{(f)} T_k(\bar{t}_j). \tag{72}$$

We can then use the values of $\{\theta_j^{(f)}\}$ and $\{s_j'^{(f)}\}$ to recover $\{x_j^{(f)}\}$ and $\{y_j^{(f)}\}$ using Eqs. 7 and 8.

In practice, we set $a = \beta N$ at the first iteration of the filtering process, for some $\beta > 0$, so that the sizes of $\hat{\theta}_k^{(f)}$ and $\hat{s}_k'^{(f)}$ become small for values of k close to $N-1$. Subsequently, we decrease a slightly at each iteration to reduce the bandwidth of the Gaussian filter.

3.3.2 Filtering the closed curve

Assume that $\{\theta_j\}$ and $\{s_j'\}$ are the values of $\theta(t)$ and $s'(t)$ discretized at the equispaced nodes $\{t_j\}$ in Eq. 66, where $\theta_j = \theta(t_j)$ and $s_j' = s'(t_j)$. We apply the DFT to derive the Fourier coefficients $\{\hat{\theta}_k\}$ of $\theta(t)$ and $\{\hat{s}'_k\}$ of $s'(t)$. Using the Gaussian filter, we filter the Fourier coefficients $\{\hat{\theta}_k\}$ and $\{\hat{s}'_k\}$ to obtain the filtered Fourier coeffcients $\{\hat{\theta}_k^{(f)}\}$ and $\{\hat{s}_k'^{(f)}\}$,

$$\hat{\theta}_k^{(f)} = e^{-\pi \frac{k^2}{a^2}} \hat{\theta}_k, \qquad k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1, \tag{73}$$

and

$$\hat{s}_k'^{(f)} = e^{-\pi \frac{k^2}{a^2}} \hat{s}'_k, \qquad k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1. \tag{74}$$

We recover the filtered sequences $\{\theta_j^{(f)}\}$ and $\{s_j'^{(f)}\}$ by applying the IDFT to the filtered Fourier coefficients $\{\hat{\theta}_k^{(f)}\}$ and $\{\hat{s}_k'^{(f)}\}$,

$$\theta_j^{(f)} = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{\theta}_k^{(f)} e^{\frac{2\pi i}{N} kj} + \frac{c}{L} t_j, \qquad j = 0, \ldots, N-1, \tag{75}$$

and

$$s_j'^{(f)} = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{s}_k'^{(f)} e^{\frac{2\pi i}{N} kj}, \qquad j = 0, \ldots, N-1. \tag{76}$$

Similarly, the curve can be reconstructed from $\{\theta_j^{(f)}\}$ and $\{s_j'^{(f)}\}$, using Eqs. 7 and 8.

When the curve is closed, we set $a = \beta \frac{N}{2}$ at the first iteration of the filtering process, for some $\beta > 0$, so that the sizes of $\hat{\theta}_k^{(f)}$ and $\hat{s}_k'^{(f)}$ become small for values of k close to $-\frac{N}{2}$ or $\frac{N}{2} - 1$. Similarly, we decrease a at each subsequent iteration to reduce the bandwidth of the Gaussian filter.

3.4 Closing the curve

Applying a filter to $\theta(t)$ and $s'(t)$ of a closed curve, in general, makes the curve become open. To close the curve, we require that

$$\int_0^L s'(t)\cos\theta(t)\,dt = 0, \tag{77}$$

and

$$\int_0^L s'(t)\sin\theta(t)\,dt = 0. \tag{78}$$

We orthogonalize $s'(t)$ to $\cos\theta(t)$ and $\sin\theta(t)$ using the trapezoidal rule, as follows. Suppose we have the values $\{s'_j\}$, $\{\cos\theta_j\}$ and $\{\sin\theta_j\}$ of $s'(t)$, $\cos\theta(t)$ and $\sin\theta(t)$ sampled at the points $\{t_j\}$ defined in Eq. 66. We ensure that $\{s'_j\}$ is orthogonal to $\cos\theta_j$ by setting $\{s'_j\}$ to the values

$$s'_j - \cos\theta_j \frac{\frac{1}{N}\sum_{j=0}^{N-1} s'_j\cos\theta_j}{\frac{1}{N}\sum_{j=0}^{N-1}\cos^2\theta_j}, \qquad j = 0,\ldots,N-1. \tag{79}$$

We let $\{\lambda_j\}$ be the vector defined by the formula

$$\lambda_j = \sin\theta_j - \cos\theta_j \frac{\frac{1}{N}\sum_{j=0}^{N-1}\sin\theta_j\cos\theta_j}{\frac{1}{N}\sum_{j=0}^{N-1}\cos^2\theta_j}, \qquad j = 0,\ldots,N-1. \tag{80}$$

Finally, we orthogonalize $\{s'_j\}$ to $\{\lambda_j\}$ by setting $\{s'_j\}$ to the values

$$s'_j - \lambda_j \frac{\frac{1}{N}\sum_{j=0}^{N-1} s'_j\lambda_j}{\frac{1}{N}\sum_{j=0}^{N-1}\lambda_j^2}, \qquad j = 0,\ldots,N-1. \tag{81}$$

The sequence $\{s'_j\}$ is now orthogonal to both $\{\cos\theta_j\}$ and $\{\sin\theta_j\}$. Thus, the conditions Eqs. 77 and 78 are satisfied to within the accuracy of the trapezoidal rule.

3.5 Repositioning the curve

In general, the curve will not pass through the original data points after filtering. Moreover, filtering $\theta(t)$ and $s'(t)$ changes the tangent vector $T(t)$, which results in changes in the orientation and position of the curve. In this section, we describe how to rotate the reconstructed curve so that the sum of squares of the distances between the curve and the original data points is minimized.

Given the original data points $\{\mathbf{C}_i\}$, where $\mathbf{C}_i = (\mathbf{C}_{ix}, \mathbf{C}_{iy})$, $i = 0,\ldots,n$, we find $\tilde{t}_0,\ldots,\tilde{t}_n \in [0, L]$ such that if $(\tilde{x}_i, \tilde{y}_i) = (x(\tilde{t}_i), y(\tilde{t}_i))$, then $(\tilde{x}_i, \tilde{y}_i)$ is the closest point

on the curve to $(\mathbf{C}_{ix}, \mathbf{C}_{iy})$, for $i = 0, \ldots, n$. We determine $\tilde{t}_0, \ldots, \tilde{t}_n$ only once, as described in Remark 2. Let $(\bar{x}, \bar{y})$ denote the centroid of the closest points $\{(\tilde{x}_i, \tilde{y}_i)\}$. Suppose that $\{\phi_i\}$ are the values of the angle between $\{(\tilde{x}_i, \tilde{y}_i)\}$ and $(\bar{x}, \bar{y})$, and that $\{r_i\}$ are the distances between $\{(\tilde{x}_i, \tilde{y}_i)\}$ and $(\bar{x}, \bar{y})$. We shift the centroid of all the closest points $\{(\tilde{x}_i, \tilde{y}_i)\}$ by $(\Delta x, \Delta y)$, and rotate the curve by an angle of ψ around the centroid. Observe that the sum of squares of the distances between the closest points and the original data points is given by

$$f(\psi, \Delta x, \Delta y) = \sum_{i=0}^{n} (\bar{x} + \Delta x + r_i \cos(\phi_i + \psi) - \mathbf{C}_{ix})^2 + (\bar{y} + \Delta y + r_i \sin(\phi_i + \psi) - \mathbf{C}_{iy})^2. \tag{82}$$

We use Newton's method to obtain the values of ψ, Δx, and Δy which minimize $f(\psi, \Delta x, \Delta y)$.

Remark 1 One might also think to rescale the curve by multiplying $\{r_i\}$ by a constant c, since filtering $s'(t)$ changes the length of the curve. However, rescaling the curve distorts the structure of the closest points $(\tilde{x}_i, \tilde{y}_i)$ on the curve. Large perturbations, as described in Sect. 3.6, are sometimes needed as a result, and therefore, the smoothness of the curve after adding perturbations can be reduced.

3.6 Adding perturbations to the curve

Since the curve does not pass through the original data points $\{\mathbf{C}_i\}$ after filtering $\theta(t)$ and $s'(t)$, we introduce a set of Gaussian functions, $\{g_i(t)\}$, which we use as smooth perturbations that can be added to the curve to ensure that the curve passes through the points $\{\mathbf{C}_i\}$. We define $g_i(t)$ by

$$g_i(t) = e^{-\sigma_i \left(\frac{t-\tilde{t}_i}{L}\right)^2}, \qquad i = 0, \ldots, n, \tag{83}$$

for $t \in [0, L]$, where $\tilde{t}_i$ is the curve parameter of the closest point $(\tilde{x}_i, \tilde{y}_i) = (x(\tilde{t}_i), y(\tilde{t}_i))$ to C_i, and σ_i determines the bandwidth of the perturbation. When the curve is closed, $g_i(t)$ is modified to be a periodic function with period L, given by the formula

$$g_i(t) = \sum_{k=-\infty}^{\infty} e^{-\sigma_i \left(\frac{t-\tilde{t}_i}{L} + k\right)^2}, \qquad i = 0, \ldots, n. \tag{84}$$

It is obvious that $g_i(t) = g_i(t + L)$. We construct $\{(\bar{x}_j, \bar{y}_j)\}$ from $\{(x_j, y_j)\}$ by adding $g_i(t)$ at the discretized points $\{t_j\}$,

$$\bar{x}_j = x_j + \sum_{i=0}^{n} c_{ix} g_i(t_j), \qquad j = 0, \ldots, N-1, \tag{85}$$

and

$$\bar{y}_j = y_j + \sum_{i=0}^{n} c_{iy} g_i(t_j), \qquad j = 0, \ldots, N-1, \tag{86}$$

where $\{c_{ix}\}$ and $\{c_{iy}\}$ are the coefficients of perturbations in x and y, separately, which are reasonably small since the curve is filtered slightly at each iteration. Let $\{(\bar{\bar{x}}_j, \bar{\bar{y}}_j)\}$ denote the points on the perturbed curve corresponding to $\tilde{t}_0, \ldots, \tilde{t}_n$. We require

$$\bar{\bar{x}}_i = \mathbf{C}_{ix}, \qquad i = 0, \ldots, n, \tag{87}$$

and

$$\bar{\bar{y}}_i = \mathbf{C}_{iy}, \qquad i = 0, \ldots, n, \tag{88}$$

and solve two linear systems of $n+1$ equations to compute the values of $\{c_{ix}\}$ and $\{c_{iy}\}$. We observe that, since the perturbations $g_i(t)$ are Gaussians, they are each, to finite precision, compactly supported. Thus, the linear systems that we solve are effectively banded, and the number of bands is determined by $\min_i \sigma_i$. An $O(n+1)$ solver can be used to speed up the computations.

Remark 2 We only calculate $\{\tilde{t}_i\}$ once, at the first iteration before filtering, and use the same set of $\{\tilde{t}_i\}$ at each iteration. Although it seems more natural to recalculate $\{\tilde{t}_i\}$ at each iteration, so that the discrepancies are fixed by smaller perturbations, the resulting perturbations are always orthogonal to the curve. Changes in the length of the curve due to filtering cannot be eliminated by adding such perturbations, with the effect that the length of the curve grows if the points $\{\tilde{t}_i\}$ are recalculated at each iteration. By using the same set of closest points for all iterations, the perturbations can be oblique, which results in a control over the total length of the curve during the filtering process.

3.7 The termination criterion of the algorithm

Since the bandwidths of the coefficients of $\theta(t)$ and $s'(t)$ are reduced at each iteration, and adding small, smooth perturbations has a negligible effect on the bandwidth of the curve, one can expect to achieve the desired bandwidth of the representations of $\theta(t)$ and $s'(t)$ by iteratively filtering the coefficients. However, we note that there is a minimum number of coefficients that are necessary to represent a curve, as determined by the sample data points. When fewer than this number of coefficients are used, the curve reconstructed by these overfiltered coefficients may deviate drastically from the sample data points. The resulting large perturbations required to fix the discrepancies can harm the smoothness of the curve. The purpose of this section is to set up a termination criterion, so that the algorithm will terminate if the coefficients of $\theta(t)$ and $s'(t)$, beyond a user-specified number of terms, are filtered to zero, to the requested accuracy.

We denote the desired accuracy of the approximation by ϵ, which is often set to be machine precision, and the desired number of coefficients representing the curve that are larger than ϵ by n_{coefs}. Due to the potentially large condition number of spectral differentiation, some accuracy is lost when computing the coefficients of $x'(t)$ and $y'(t)$, and thus $\theta(t)$ and $s'(t)$, at each iteration. Thus, we determine thresholds for the coefficients of $\theta(t)$ and $s'(t)$, representing the smallest values we can expect to measure, and denote them by δ_θ and $\delta_{s'}$. We consider first the open curve case. Since the condition number of the Chebyshev differentiation matrix is bounded by approximately $N^{\frac{3}{2}}$, where N is the number of coefficients, the error induced by differentiating $x(t)$ and $y(t)$ is approximately

$$\epsilon N^{\frac{3}{2}} \sqrt{\|x(t)\|^2_{L^2[0,L]} + \|y(t)\|^2_{L^2[0,L]}} \tag{89}$$

$$\approx \epsilon N^{\frac{3}{2}} \sqrt{\sum_j x_j^2 w_j + \sum_j y_j^2 w_j}, \tag{90}$$

where $\{w_j\}$ denotes the Chebyshev weights on $[0, L]$. Considering how $\theta(t)$ is calculated, the error in $\theta(t)$ is proportional to the error in $x'(t)$ and $y'(t)$, divided by the norm of the tangent vector $(x'(t), y'(t))$. Thus, we set

$$\begin{aligned} \delta_\theta &= \epsilon N^{\frac{3}{2}} \sqrt{\|x(t)\|^2_{L^2[0,L]} + \|y(t)\|^2_{L^2[0,L]}} \cdot \left\| \frac{1}{\sqrt{x'(t)^2 + y'(t)^2}} \right\|_{L^\infty[0,L]} \\ &\approx \frac{\epsilon N^{\frac{3}{2}} \sqrt{\sum_j x_j^2 w_j + \sum_j y_j^2 w_j}}{\min \sqrt{x_j'^2 w_j + y_j'^2 w_j}}, \end{aligned} \tag{91}$$

where x_i' and y_i' are the discretized values of $x'(t)$ and $y'(t)$. Similarly, the error in $s'(t)$ is proportional to the error in $x'(t)$ and $y'(t)$. Thus, we set

$$\begin{aligned} \delta_{s'} &= \epsilon N^{\frac{3}{2}} \sqrt{\|x(t)\|^2_{L^2[0,L]} + \|y(t)\|^2_{L^2[0,L]}}, \\ &\approx \epsilon N^{\frac{3}{2}} \sqrt{\sum_j x_j^2 w_j + \sum_j y_j^2 w_j}. \end{aligned} \tag{92}$$

The thresholds δ_θ and $\delta_{s'}$ for the closed curve case are almost identical, except that the condition number of spectral differentiation matrix is approximately N, where N is the number of coefficients, from which it follows that $N^{\frac{3}{2}}$ is replaced by N, and the weights w_j are replaced by $\frac{L}{N}$.

Suppose that we have the desired accuracy of the approximation, ϵ, the threshold, δ_θ, and the desired number of coefficients larger than ϵ, n_{coefs}. We consider first the coefficients of $\theta(t)$. Our goal is to determine the number of coefficients, $n_{\text{coefs}}^{\delta_\theta}$, that we expect to be larger than δ_θ, when there are only n_{coefs} terms larger than $\|\hat{\theta}\|_\infty \epsilon$. In order to approximate $n_{\text{coefs}}^{\delta_\theta}$, we assume that the coefficients $\{\hat{\theta}_k\}$ decay exponentially, like

$\|\hat{\theta}\|_\infty e^{-Ck}$, from the maximum value $\|\hat{\theta}\|_\infty$ to $\|\hat{\theta}\|_\infty \epsilon$. This implies that $C = \frac{\log(1/\epsilon)}{n_{\text{coefs}}}$. Thus,

$$e^{-\log(1/\epsilon)\frac{n_{\text{coefs}}^{\delta_\theta}}{n_{\text{coefs}}}} = \delta_\theta, \tag{93}$$

so,

$$n_{\text{coefs}}^{\delta_\theta} = n_{\text{coefs}} \frac{\log(1/\delta_\theta)}{\log(1/\epsilon)}. \tag{94}$$

We compute $n_{\text{coefs}}^{\delta_{s'}}$ in exactly the same way. At each iteration, if only $n_{\text{coefs}}^{\delta_\theta}$ and $n_{\text{coefs}}^{\delta_{s'}}$ numbers of terms are larger than δ_θ and $\delta_{s'}$, respectively, then the algorithm terminates. Eventually, n_{coefs} coefficients are returned to the user to represent the curve, up to the precision ϵ.

Remark 3 Since the values of δ_θ, $\delta_{s'}$, $n_{\text{coefs}}^{\delta_\theta}$ and $n_{\text{coefs}}^{\delta_{s'}}$ are fairly consistent in each iteration, we only calculate these values once, at the first iteration.

3.8 Summary and cost of the algorithm

The algorithm can be summarized as follows:

1. Given $n+1$ points $\mathbf{C}_0, \ldots, \mathbf{C}_n$, fit a C^2 Bézier spline to connect the points.
2. Discretize the curve at $N \gg n+1$ Chebyshev nodes if the curve is open, or $N \gg n+1$ equispaced nodes if the curve is closed, and compute $\{\theta_j\}$ and $\{s'_j\}$.

Repeat the steps 3, …, 9 until a C^∞ smooth curve can be represented by the requested number of coefficients, n_{coefs}, letting i denote the iteration number:

3. Obtain the Chebyshev coefficients or the Fourier coefficients of $\{\theta_j\}$ and $\{s'_j\}$.
4. Determine the number of coefficients of $\{\theta_j\}$ and $\{s'_j\}$ larger than δ_θ and $\delta_{s'}$. If there are fewer than $n_{\text{coefs}}^{\delta_\theta}$ and $n_{\text{coefs}}^{\delta_{s'}}$, respectively, then return the first n_{coefs} coefficients of $x(t)$ and $y(t)$.
5. Apply the filter in Eq. 57, with parameter a_i, to the coefficients of $\{\theta_j\}$ and $\{s'_j\}$ to compute the filtered values of $\{\theta_j\}$ and $\{s'_j\}$.
6. In the case of a closed curve, modify $\{s'_j\}$ to satisfy the constraints Eqs. 77 and 78 in order to close the curve after filtering.
7. Reconstruct the curve from $\{\theta_j\}$ and $\{s'_j\}$ by Eqs. 7 and 8.
8. Rotate the curve to minimize the sum of squares of the distances between the curve and the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$.
9. Add smooth Gaussian perturbations to make the curve pass through the points $\mathbf{C}_0, \ldots, \mathbf{C}_n$.
10. Compute $a_{i+1} = c \cdot a_i$, for some $0 < c < 1$.

Solving for the control points of the Bézier spline in step 1 costs $O(n+1)$ operations, and discretizing the spline at N points in step 2 costs $O(N)$ operations. Step 3 involves

spectral differentiation and the discrete Chebyshev transform in the open curve case, or the DFT in the closed curve case, where the discrete Chebyshev transform can be replaced by the fast Chebyshev transform and the DFT can be replaced by the FFT. The cost of step 3 is thus reduced to $O(N \log N)$. Checking the termination condition in step 4 costs approximately $O(N)$ operations. Applying the filter and reconstructing $\{\theta_j\}$ and $\{s'_j\}$ in step 5 has the same cost as applying the inverse fast Chebyshev transform or the IFFT, which costs $O(N \log N)$ operations. If the curve is closed, we must modify $\{s'_j\}$ so that the curve remains closed. The cost of closing the curve by looping through $\{s'_j\}$ in step 6 is $O(N)$. Step 7 involves spectral integration, and the inverse fast Chebyshev transform in the open curve case, or the IFFT in the closed curve case, which has the same $O(N \log N)$ cost as step 3. The cost of using Newton's method to rotate the curve in step 8 is $O(n+1)$, and the cost of solving for the coefficients of the smooth perturbations added to the curve in step 9 is $O(n+1)$. The total cost is thus $O(N \log N)$ per iteration.

3.9 Affine invariance of the curve

Recall that the primary steps of our algorithm involve filtering $\theta(t)$ and $s'(t)$, closing the curve if needed, repositioning the curve, and adding perturbations, as described in Sects. 3.3, 3.4, 3.5, and 3.6, respectively. We aim to demonstrate the affine invariance of the curve produced by our algorithm, by examining each of these steps under affine transformations. Note that the main types of affine transformations include translation, rotation, and scaling, which result in either an addition of an angle to $\theta(t)$ or a scaling in $s'(t)$. Although there exist other types of affine transformations, such as reflection and shear mapping, we omit them from this discussion as their analysis follows the same reasoning.

- Since the filtering process does not involve any low-frequency coefficients, the addition of a constant to $\theta(t)$, which is equivalent to an addition to the zero-frequency coefficient, is unaffected by the filtering process. Scaling $s'(t)$ is clearly also unaffected by the filtering process.
- Closing the curve requires the orthogonality of $s'(t)$ to both $\cos(\theta(t))$ and $\sin(\theta(t))$, as specified in Eqs. 77 and 78. Recall that $\cos(\theta(t)+c) = \cos(\theta(t))\cos(c) - \sin(\theta(t))\sin(c)$ and $\sin(\theta(t)+c) = \sin(\theta(t))\cos(c) + \cos(\theta(t))\sin(c)$. Thus, the curve remains closed after the addition of a constant to $\theta(t)$. Scaling $s'(t)$ also preserves its orthogonality to $\sin(\theta(t))$ and $\cos(\theta(t))$.
- The repositioning step minimizes the distance between the sample data points and certain points on the curve, and thus, the final position of the curve relative to the sample data points is invariant under translation, rotation, and scaling.
- Since perturbations are added to eliminate the discrepancies between the curve and the sample data points, and the perturbations are determined by the positions of certain points on the curve relative to the sample data points, the step of adding perturbations is also invariant under translation, rotation, and scaling.

Considering that the curve preserves affine invariance during each of these steps, the resulting curve produced by our algorithm is affine invariant.

3.10 Comparison with the method of Beylkin and Rokhlin

As our method is closely related to the method of Beylkin and Rokhlin [12], in this section, we provide an analysis of the similarities and differences between these two methods.

Both methods ensure the global smoothness of the curve, with the primary steps involving filtering the coefficients of the representations of the curve to attain smaller bandwidths and adding smooth perturbations to reconnect the curve with the sample data points without reducing the smoothness of the curve.

The method of [12] filters the tangential angle of the curve, $\theta(s)$, which is parametrized by the arc-length parameter s. This parametrization may lead to dramatic changes in $\theta(s)$ around some high-curvature features on the curve, requiring a large number of coefficients to represent the curve. Our method, on the other hand, uses the curve parameter t for the parametrization of the curve and filters both the tangential angle $\theta(t)$ and the first derivative of the arc-length function $s'(t)$. This choice of parameterization allows $s'(t)$ to become smaller near the high-curvature areas, resulting in smaller bandwidths for both $\theta(t)$ and $s'(t)$. Additionally, it avoids the computation of the inverse function of $s(t)$, required by the method of [12]. The procedure of closing the curve can be achieved by directly orthogonalizing $s'(t)$ to $\cos(\theta(t))$ and $\sin(\theta(t))$ using the trapezoidal rule, without employing nonlinear searches, as required by the method of [12].

The next key difference is that we use a continuation method rather than a single step of filtering, by slightly filtering the representations of the curve at each iteration and adding small perturbations to the reconstructed curve. This enables the representations of the curve to have much smaller final bandwidths after a reasonable number of iterations, provided that a small amount of filtering is applied at each iteration.

The final key difference is that our method is applicable not only to closed curves, but also to open curves.

4 Numerical experiments

In this section, we demonstrate the performance of our algorithm with several numerical examples, and compare our method to several other methods for filtering bandlimited curves through user-specified points. We implemented our algorithm in Fortran 77 and compiled it using the GFortran Compiler, version 9.4.0, with -O3 flag. All experiments were conducted on a laptop with 16 GB of RAM and an Intel 11th Gen Core i7-1185G7 CPU. Furthermore, we use FFTW library (see [15]) for the implementations of the FFT and the fast cosine transform. The latter is used to implement the fast Chebyshev transform. An implementation of our algorithm is provided in https://doi.org/10.5281/zenodo.7742917.

The following notation appears in this section:

- N: the number of discretization nodes.
- n: the number of sample data points.
- n_{iters}: the maximum number of iterations.

- n_{stop}: the number of iterations needed for the algorithm to terminate.
- a_1: the parameter defined in Eq. 57, which determines the bandwidth of the Gaussian filter at the first iteration. We set $a_1 = \beta N$ for open curves and $a_1 = \beta \frac{N}{2}$ for closed curves. Throughout this section, we choose $\beta = \frac{\sqrt{2\pi}}{5}$.
- h_{filter}: the amount of filtering at each iteration, as expressed in the formula $a_i = a_1(1 - h_{\text{filter}})^{i-1}$, where a_i determines the bandwidth of the Gaussian filter at the ith iteration.
- ϵ: the desired accuracy of the approximation to the curve. As ϵ is dependent on the size of the curve, for consistency, we scale the sample data points, so that either the width or height of the collection of data points, whichever is closer to 1, is set equal to 1.
- n_{coefs}: the requested number of the coefficients representing the curve to precision ϵ.
- n_{bands}: the bandwidth of the matrix describing the effect of the Gaussian perturbations centered at each closest point.
- x'_{left}, y'_{left}: the derivative of the initial curve specified at the left endpoint, in the x coordinate and y coordinate separately. This notation only exists in the open curve case. Notice that the filtering process can potentially alter the derivative of the curve at the left endpoint.
- x'_{right}, y'_{right}: the derivative of the initial curve specified at the right endpoint, in the x coordinate and y coordinate separately. This notation only exists in the open curve case. Notice that the filtering process can potentially alter the derivative of the curve at the right endpoints.
- E_{samp}: the maximum l_2 norm of the distance between the curve, defined by n_{coefs} Chebyshev or Fourier coefficients, and the sample data points.

While there is no strict rule on how to choose the input parameters, we assume that the users pick a reasonable combination of inputs, so that the algorithm terminates before reaching the maximum number of iterations, n_{iters}.

4.1 Performance of our algorithm

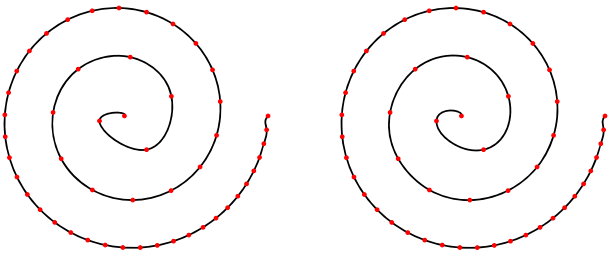
In this section, we demonstrate the performance of our algorithm on several examples, for both open and closed curves.

4.1.1 Open curve examples

For our first example, we sample some points from a spiral shape with the polar representation $(r(t)\cos\varphi(t), r(t)\sin\varphi(t))$, where

$$\begin{aligned} \varphi(t) &= \frac{6\pi}{\log 2}\log t, \\ r(t) &= \varphi(t), \end{aligned} \tag{95}$$

with $t \in [1, 2]$, and construct the initial Bézier spline passing through the data points, as shown in Fig. 1a. The sample data points are scaled so that their width is 1. We set



(a) Initial curve (b) Final curve

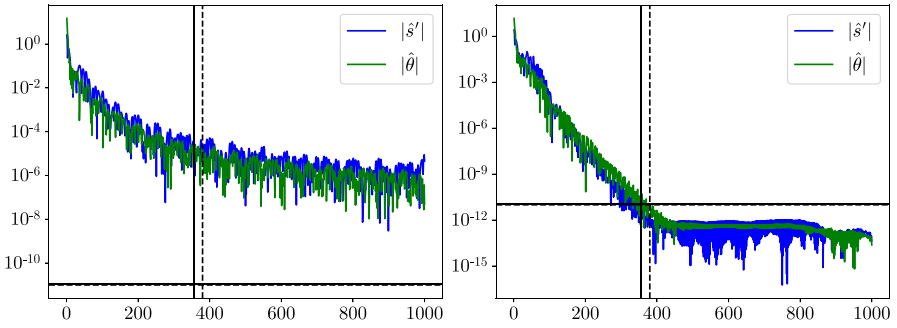
Fig. 1 The result of our algorithm applied to Eq. 95. The red dots mark the sample data points

$N = 1000, n = 50, x'_{\text{left}} = 0.05, y'_{\text{left}} = 0.05, x'_{\text{right}} = 0.05, y'_{\text{right}} = 0.05, n_{\text{iters}} = 60,$ $h_{\text{filter}} = \frac{1}{25}, \epsilon = 10^{-16}, n_{\text{coefs}} = 500, n_{\text{bands}} = 8$. After $n_{\text{stop}} = 16$ iterations, the algorithm terminates and returns a curve represented by only 500 Chebyshev coefficients. The magnitudes of the Chebyshev coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, are displayed in Fig. 2. We display the Chebyshev coefficients that are necessary to represent both the initial and final curves in Fig. 3. We can see that the shape of the final curve in Fig. 1b is smoother, especially at the center of the spiral. Moreover, the final curve passes through the sample data points with an error of $E_{\text{samp}} = 0.11548 \cdot 10^{-13}$.

Another example, depicted in Fig. 4a, is obtained by sampling from the curve

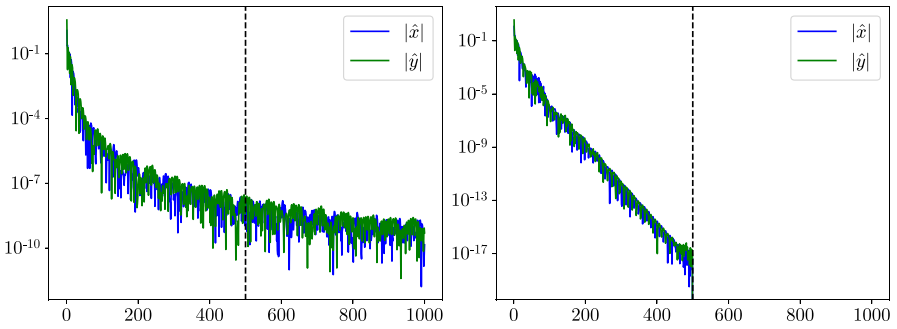
$$\gamma(t) = \left(5t, 3\cos\left(10t\pi\right)^3\right), \qquad t \in [0, 1]. \tag{96}$$

The sample data points are scaled so that their height is 1. We run the algorithm by choosing $n = 70, N = 4500, x'_{\text{left}} = 0.25, y'_{\text{left}} = 0.25, x'_{\text{right}} = 0.25, y'_{\text{right}} = 0.25,$



(a) Before smoothing (b) After smoothing

Fig. 2 of $s'(t)$ and $\theta(t)$ corresponding to Fig. 1. The value of $\delta_{s'}$ is indicated by a horizontal solid line and the value of δ_θ is indicated by a horizontal dashed line. The 355th coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 380th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line

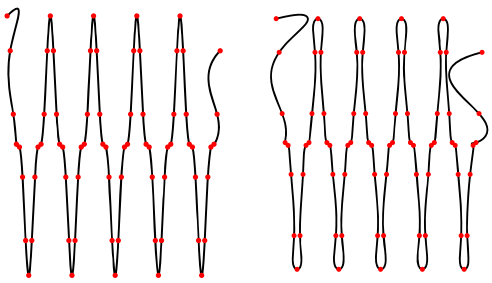


(a) Before smoothing (b) After smoothing

Fig. 3 Chebyshev coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 1. The value of n_{coefs} is indicated by a vertical dashed line

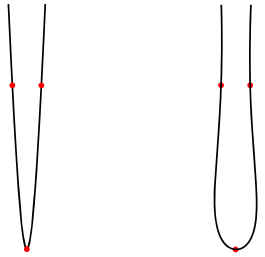
$n_{\text{iters}} = 70$, $h_{\text{filter}} = \frac{1}{45}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 3620$, $n_{\text{bands}} = 6$. The initial curve is observed to bend unnaturally when zooming in on some details, for example, those shown in Fig. 5a. Thus, a reasonably large number of Chebyshev coefficients are required to represent $s'(t)$ and $\theta(t)$, as shown in Fig. 6. By looking at Figs. 4b and 5b, the final curve appears more like a manually drawn smooth curve after $n_{\text{stop}} = 60$ iterations. The coefficients returned by the algorithm represent a curve passing through the sample data points to within an error of $E_{\text{samp}} = 0.16875 \cdot 10^{-13}$. We display the magnitudes of the Chebyshev coefficients of both the initial and final curves in Fig. 7.

Figure 8a shows a roughly sketched shape resembling a snake. We scale the sample data points so that their height is 1 and run the algorithm by choosing $N = 4000$, $n = 44$, $x'_{\text{left}} = 0.05$, $y'_{\text{left}} = -0.02$, $x'_{\text{right}} = -0.06$, $y'_{\text{right}} = 0.02$, $n_{\text{iters}} = 80$, $h_{\text{filter}} = \frac{1}{50}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 1780$, $n_{\text{bands}} = 6$. The algorithm terminates at the $n_{\text{stop}} = 71$st iteration, and the final curve passes through the sample data points to within an error of $E_{\text{samp}} = 0.35056 \cdot 10^{-14}$. We present the magnitudes of the coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, in Fig. 9, and the magnitudes of the coefficients of $x(t)$ and $y(t)$ of both the initial and final curves in Fig. 10.



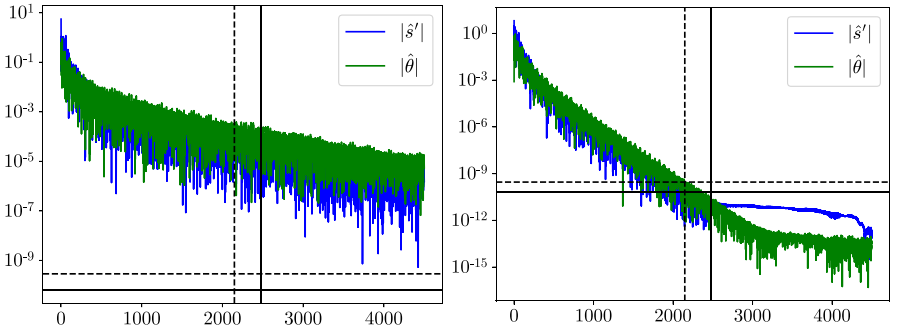
(a) Initial curve (b) Final curve

Fig. 4 The result of our algorithm applied to Eq. 96. The red dots mark the sample data points



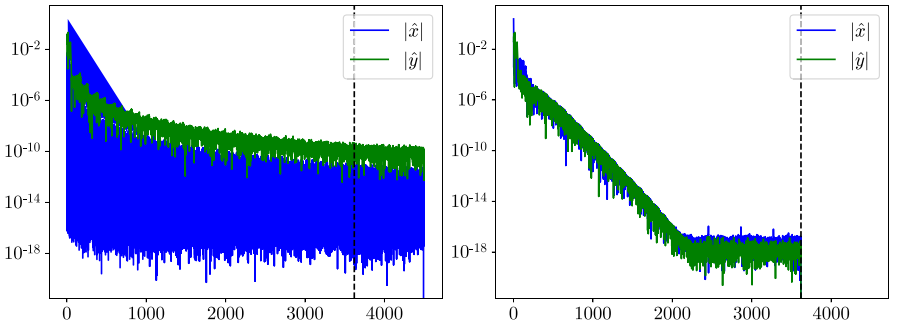
(a) Initial curve (b) Final curve

Fig. 5 A zoom-in detail of Fig. 4



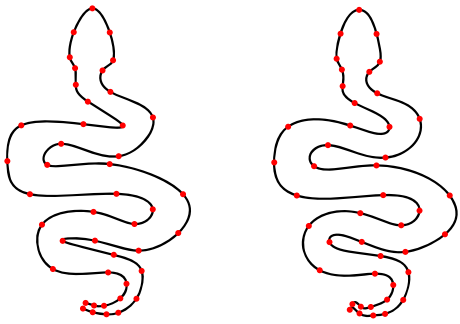
(a) Before smoothing (b) After smoothing

Fig. 6 Chebyshev coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 4. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 2472nd coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 2148th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



(a) Before smoothing (b) After smoothing

Fig. 7 Chebyshev coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 4. The value of n_{coefs} is indicated by a vertical dashed line

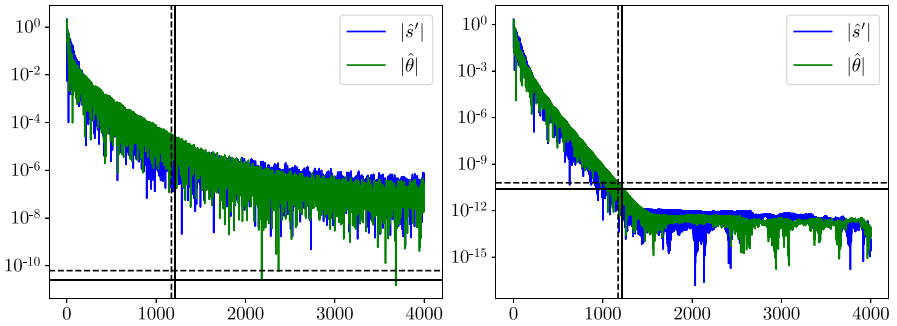


(a) Initial curve (b) Final curve

Fig. 8 The result of our algorithm applied to a hand-drawn depiction of a snake. The red dots mark the sample data points

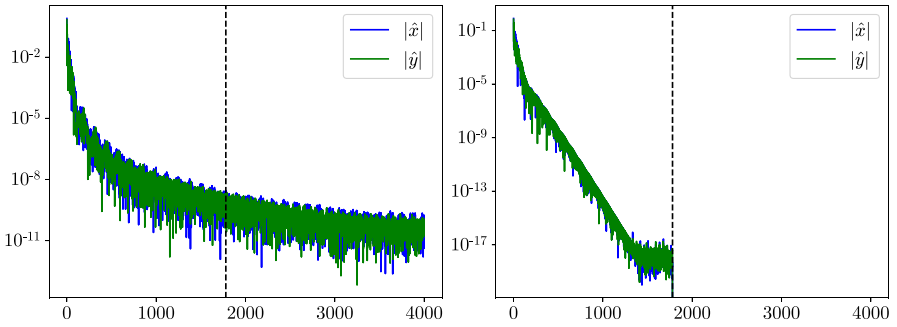
We then apply our algorithm to an arbitrarily oscillating shape, as displayed in Fig. 11a. The sample data points are scaled so that their width is 1. The initial curve oscillates unnaturally and has some sharp features. We set $N = 4500$, $n = 40$, $x'_{\text{left}} = 0.20$, $y'_{\text{left}} = -0.20$, $x'_{\text{right}} = -0.20$, $y'_{\text{right}} = 0.40$, $n_{\text{iters}} = 70$, $h_{\text{filter}} = \frac{1}{40}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 1830$, $n_{\text{bands}} = 8$. After $n_{\text{stop}} = 69$ iterations, the algorithm terminates and returns a curve passing through the sample data points to within an error of $E_{\text{samp}} = 0.22649 \cdot 10^{-13}$. The final curve in Fig. 11b resembles a curve drawn by hand, with a completely smooth shape that bends naturally to pass through all the sample data points to exhibit these oscillations. We present the magnitudes of the coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, in Fig. 12, and the magnitudes of the coefficients of $x(t)$ and $y(t)$ of both the initial and final curves in Fig. 13.

The runtimes per iteration for the first open curve example are displayed in Table 1. Since we use the library [15] for the implementation of the FFT, and the speed of the



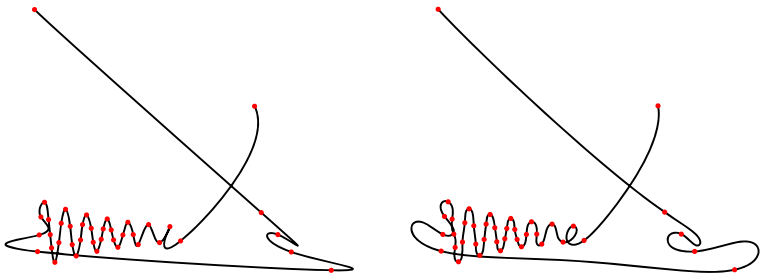
(a) Before smoothing (b) After smoothing

Fig. 9 Chebyshev coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 8. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 1214th coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 1171st coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



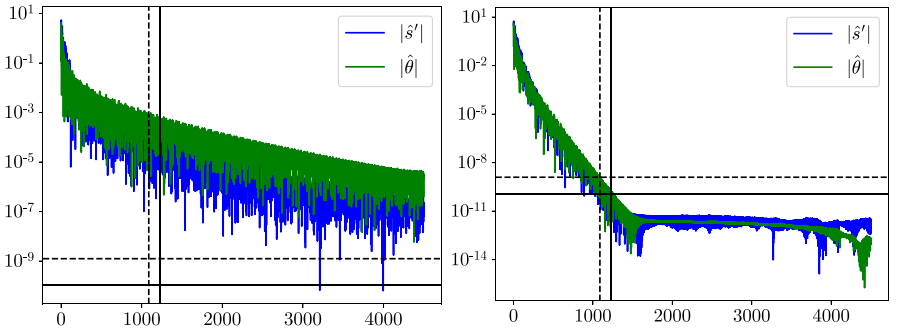
(a) Before smoothing (b) After smoothing

Fig. 10 Chebyshev coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 8. The value of n_{coefs} is indicated by a vertical dashed line



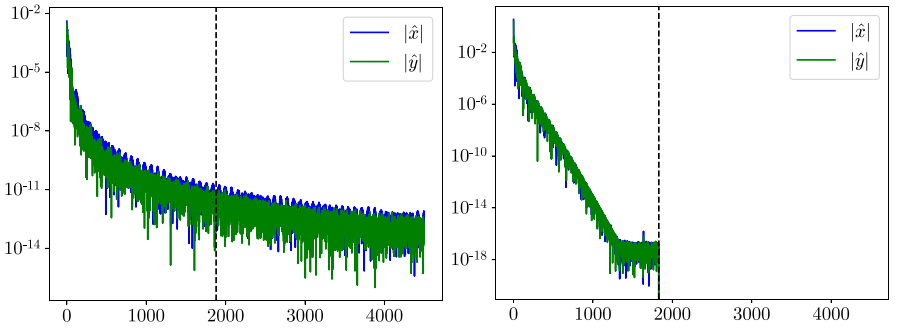
(a) Initial curve (b) Final curve

Fig. 11 The result of our algorithm applied to an arbitrarily oscillating shape. The red dots mark the sample data points



(a) Before smoothing (b) After smoothing

Fig. 12 Chebyshev coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 11. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 1222nd coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 1088th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



(a) Before smoothing (b) After smoothing

Fig. 13 Chebyshev coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 11. The value of n_{coefs} is indicated by a vertical dashed line

FFT routines in the library depends in a complicated way on the input size, we observe that the runtimes in Table 1 are not strictly proportional to the number of discretization points, N.

4.1.2 Closed curve examples

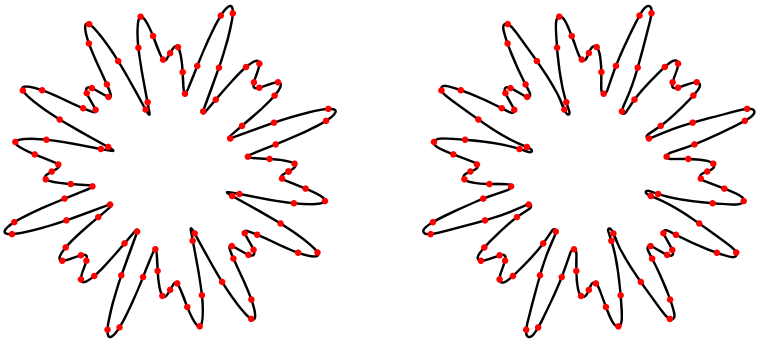
The first closed curve example is obtained by sampling from the polar representation $(r(t)\cos\varphi(t), r(t)\sin\varphi(t))$, where

$$\begin{aligned} \varphi(t) &= 2\pi t, \\ r(t) &= (1 + \frac{1}{\alpha}\cos(18\varphi(t))\sin(4\varphi(t)), \end{aligned} \tag{97}$$

with $t \in [0, 1]$, and α is a tuning parameter. The sample data points are scaled so that both their width and height are 1. We sample the curve Eq. 97 with $\alpha = 2$, $N = 8000$ and $n = 100$ to obtain the initial curve in Fig. 14a. Applying the algorithm with $n_{\text{iters}} = 70$, $h_{\text{filter}} = \frac{1}{40}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 5200$, $n_{\text{bands}} = 12$, we obtain the final curve, which appears much smoother visually when zoomed in on some details, as shown in Fig. 15. The filtered coefficients of $\theta(t)$ and $s'(t)$ are displayed in Fig. 16b. We find that, after $n_{\text{stop}} = 46$ iterations, 5200 coefficients of $x(t)$ and $y(t)$ are necessary to represent the final curve displayed in Fig. 14b, to within an error of $E_{\text{samp}} = 0.20644 \cdot 10^{-14}$. The magnitudes of the coefficients of both the initial and final curves are displayed in Fig. 17.

Table 1 Average runtime per iteration of our algorithm, for the first open curve example, calculated by determining the total runtime for 100 iterations and dividing by the number of iterations

Case	$N = 1025$	$N = 2049$	$N = 4097$	$N = 8193$
Fig. 1	$0.14308 \cdot 10^{-02}$	$0.20470 \cdot 10^{-02}$	$0.29810 \cdot 10^{-02}$	$0.51040 \cdot 10^{-02}$



(a) Initial curve (b) Final curve

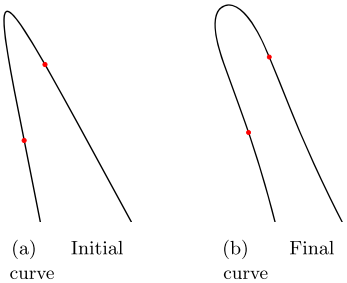
Fig. 14 The result of our algorithm applied to Eq. 97 with $\alpha = 2$. The red dots mark the sample data points

Remark 4 Note that, since the DFT, $X_{-\frac{N}{2}}, \ldots, X_{\frac{N}{2}-1}$, of a real sequence, x_0, …, x_{N-1}, satisfies the relation:

$$X_k = \overline{X}_{-k}, \qquad k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1, \tag{98}$$

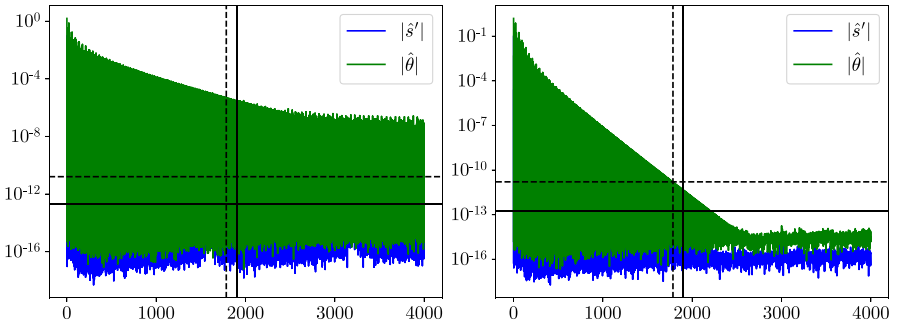
we only show the magnitudes of the coefficients for $k = 0, \ldots, \frac{N}{2} - 1$.

Another example is provided in Fig. 18, sampling the curve Eq. 97 with $\alpha = 8$, $N = 2000$ and $n = 60$. The sample data points are scaled so that both their width and height are 1. This curve has a smaller curvature than the previous curve. In this case, we set $n_{\text{iters}} = 60$, $h_{\text{filter}} = \frac{1}{40}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 1560$, $n_{\text{bands}} = 8$. The algorithm terminates after $n_{\text{stop}} = 25$ iterations, and the error between the final curve and the sample data points is $E_{\text{samp}} = 0.10240 \cdot 10^{-14}$. The magnitudes of the coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, are displayed in Fig. 19, and the magnitudes of the coefficients of both the initial and final curves are displayed in Fig. 20. While the shapes of the curves appear similar before and after smoothing, the coefficients change dramatically.



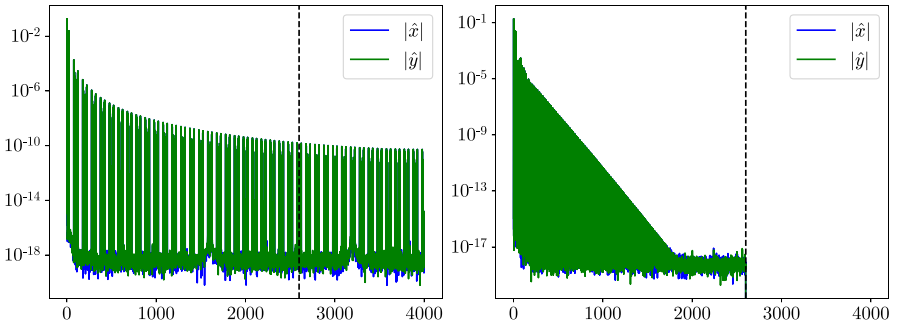
(a) Initial curve (b) Final curve

Fig. 15 A zoom-in detail of Fig. 14



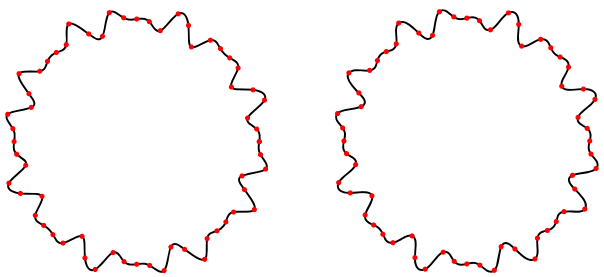
(a) Before smoothing (b) After smoothing

Fig. 16 Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 14. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 1901st coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 1785th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



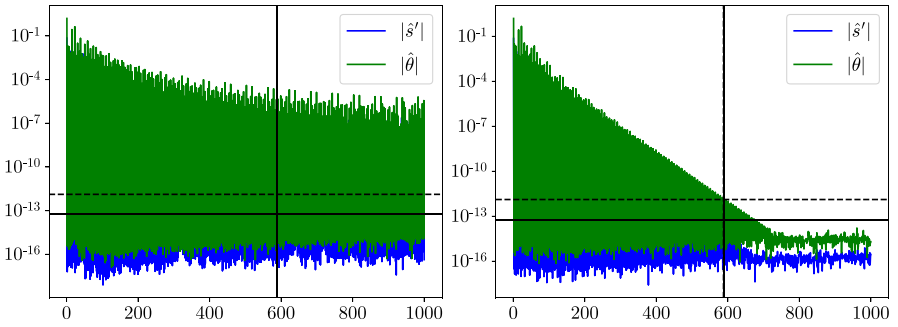
(a) Before smoothing (b) After smoothing

Fig. 17 Fourier coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 14. The value of n_{coefs} is indicated by a vertical dashed line



(a) Initial curve (b) Final curve

Fig. 18 The result of our algorithm applied to Eq. 97 with $\alpha = 8$. The red dots mark the sample data points. Although the two curves are visually indistinguishable, the curve on the right can be represented with fewer coefficients

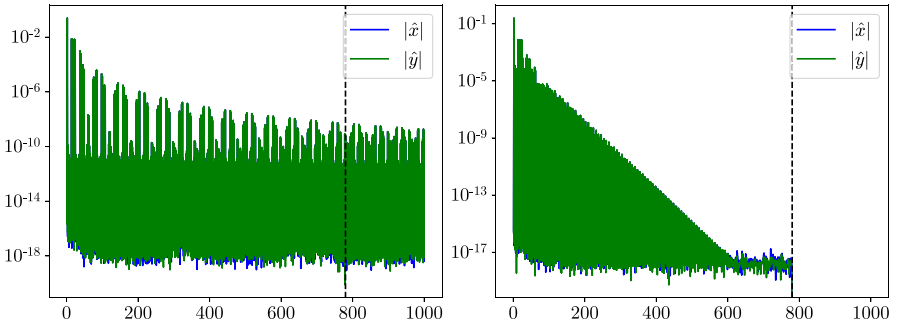


(a) Before smoothing (b) After smoothing

Fig. 19 Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 18. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 588th coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 588th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line

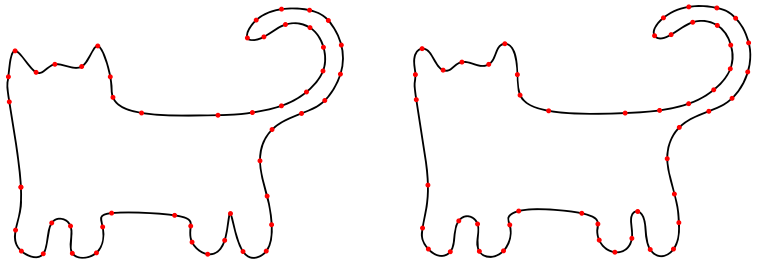
The example shown in Fig. 21 has the shape of a cat, corresponding to $n = 50$ sample data points. We scale the sample data points so that their height is 1, and discretize the curve with $N = 4000$, and set $n_{\text{iters}} = 100$, $h_{\text{filter}} = \frac{1}{50}$, $\epsilon = 10^{-15}$, $n_{\text{coefs}} = 1360$, $n_{\text{bands}} = 4$. After $n_{\text{stop}} = 70$ iterations, 1360 coefficients are sufficient to represent the curve, and the error between the final curve and the sample data points is $E_{\text{samp}} = 0.10214 \cdot 10^{-13}$. The magnitudes of the coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, are displayed in Fig. 22, and the magnitudes of the coefficients of both the initial and final curves are displayed in Fig. 23. We observe that the sharp corners on the curve in Fig. 21a become more rounded, and the final curve in Fig. 21b more closely resembles the shape of a cat.

The runtimes per iteration for the first two closed curve examples are displayed in Table 2. We observe that, as in the open curve example, the runtimes are not strictly proportional to N.



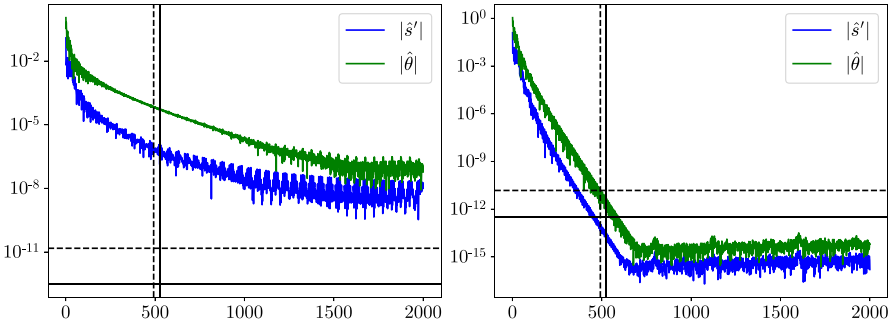
(a) Before smoothing (b) After smoothing

Fig. 20 Fourier coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 18. The value of n_{coefs} is indicated by a vertical dashed line



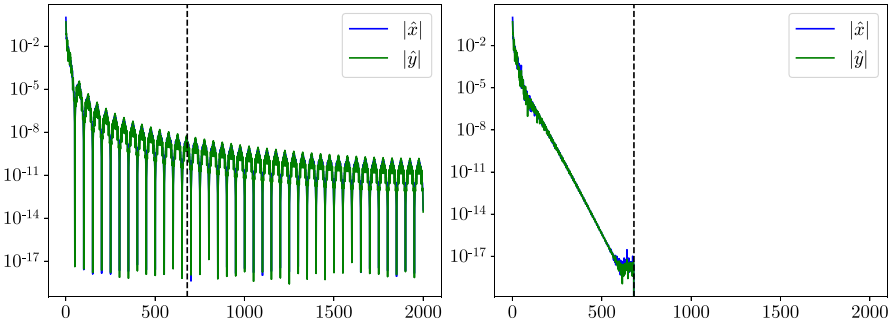
(a) Initial curve (b) Final curve

Fig. 21 The result of our algorithm applied to a hand-drawn depiction of a cat. The red dots mark the sample data points



(a) Before smoothing (b) After smoothing

Fig. 22 Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 21. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 525th coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 492nd coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



(a) Before smoothing (b) After smoothing

Fig. 23 Fourier coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 21. The value of n_{coefs} is indicated by a vertical dashed line

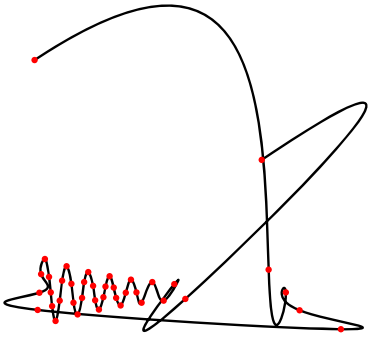
Table 2 Average runtime per iteration, for the first two closed curve examples, calculated by determining the total runtime for 250 iterations and dividing by the number of iterations

Case	$N = 1024$	$N = 2048$	$N = 4096$	$N = 8192$
Fig. 14	$0.44050 \cdot 10^{-03}$	$0.72667 \cdot 10^{-03}$	$0.13412 \cdot 10^{-02}$	$0.26530 \cdot 10^{-02}$
Fig. 18	$0.36162 \cdot 10^{-03}$	$0.61837 \cdot 10^{-03}$	$0.11924 \cdot 10^{-02}$	$0.24492 \cdot 10^{-02}$

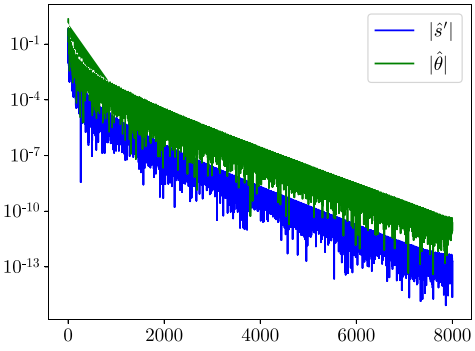
4.2 Comparison with the algorithm in Zhang and Ma

We first compare the performance of our algorithm with that of the algorithm presented in Zhang and Ma [7], using the set of sample data points that define the arbitrarily oscillating shape in Fig. 11. We set the parameter a in [7] to $a = 0.2$, which produces the smoothest shape of the curve, as displayed in Fig. 24a. The magnitudes of the coefficients of $s'(t)$ and $\theta(t)$ of the resulting curve are shown in Fig. 24b. Although the curve in Fig. 24a requires fewer coefficients to represent $x(t)$ and $y(t)$ compared to the curve in Fig. 11b, it requires over 8000 coefficients to represent $s'(t)$ and $\theta(t)$, as opposed to the 1830 coefficients for the curve produced by our method, as shown in Fig. 12b. This results in a curve with areas of high curvature. With our algorithm, any high-curvature areas are effectively smoothed, yielding a visually appealing curve and requiring much fewer coefficients to represent $s'(t)$ and $\theta(t)$.

We also present a comparison between our algorithm and the algorithm of [7] for a closed curve example. Figure 25a displays the result of the algorithm in [7] applied to the set of sample data points defining the cat in Fig. 21a, with $a = 0.4$. The number of coefficients required to represent $s'(t)$ and $\theta(t)$ is approximately $2 \cdot 4000 = 8000$, as displayed in Fig. 25b, whereas with our method, only 1360 coefficients are necessary to represent the curve in Fig. 21b. As we can observe from Fig. 25a, their curve contains some sharp features, and its visual smoothness is similar to the initial curve as displayed in Fig. 21a, prior to applying our algorithm. In contrast, our algorithm eliminates high-curvature areas, resulting in a much smoother appearance.

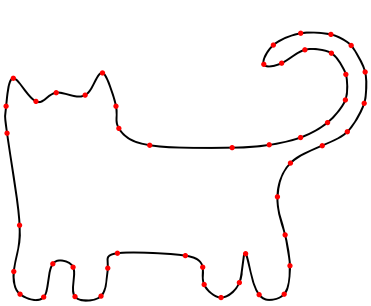


(a) The interpolating curve. The red dots mark the sample data points.

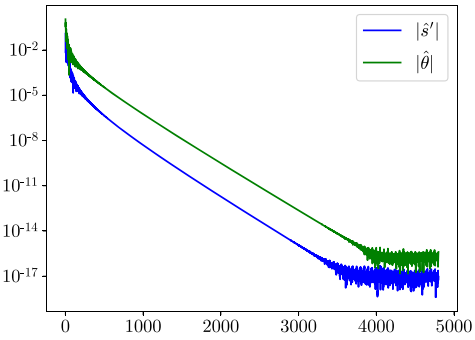


(b) Chebyshev coefficients of $s'(t)$ and $\theta(t)$ corresponding to fig. 24a.

Fig. 24 The result of the algorithm in [7] applied to the same data points as in Fig. 11



(a) The interpolating curve. The red dots mark the sample data points.



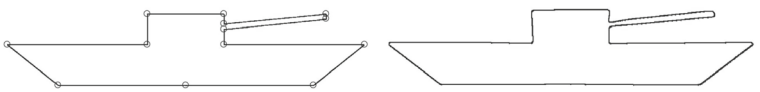
(b) Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to fig. 25a.

Fig. 25 The result of the algorithm in [7] applied to the same data points as in Fig. 21a

4.3 Comparison with the algorithm in Beylkin and Rokhlin

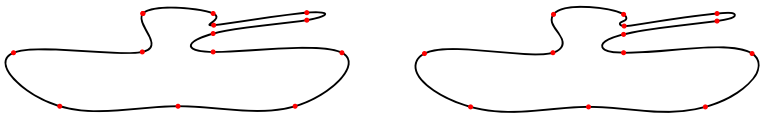
Inspired by Figs. 3 and 5 in Beylkin and Rokhlin [12], we apply our algorithm to the same sample data points to show that our algorithm produces a smoother curve and represents the curve with fewer coefficients. We start with the example in Fig. 5, reproduced in Fig. 26a. The sample data points are scaled so that their width is 1. With the parameters $N = 1600$, $n = 13$, $n_{\text{iters}} = 80$, $h_{\text{filter}} = \frac{1}{40}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 840$, and $n_{\text{bands}} = 4$, our algorithm terminates after $n_{\text{stop}} = 40$ iterations, and the final curve in Fig. 27b passes through the sample data points within an error of $E_{\text{samp}} = 0.19666 \cdot 10^{-13}$. We display the magnitudes of the coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, in Fig. 28, and the magnitudes of the coefficients of both the initial and final curves in Fig. 29.

Since the initial curve is constructed by using smooth splines to connect the sample data points, and our algorithm filters the curve further during the filtering process, the shape of the final curve deviates from that of Fig. 5 in [12]. In order to preserve the shape of the curve in Fig. 5, we increase the number of sample data points in a non-uniform way, so that more data points are sampled near the sharp features on the curve. The sample data points are scaled so that their width is 1. Applying our algorithm with $N = 4000$, $n = 59$, $n_{\text{iters}} = 80$, $h_{\text{filter}} = \frac{1}{50}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 1700$, $n_{\text{bands}} = 4$, we obtain $E_{\text{samp}} = 0.70031 \cdot 10^{-13}$ at the $n_{\text{stop}} = 47$th iteration. The initial curve is shown in Fig. 30a, and the final curve is shown in Fig. 30b. The magnitudes of the coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, are displayed in Fig. 31, and the magnitudes of the coefficients of both the initial and final curves are



(a) Figure 5 in [12] (b) Figure 6 in [12]

Fig. 26 The result of the algorithm in [12] applied to Figure 5 in [12]



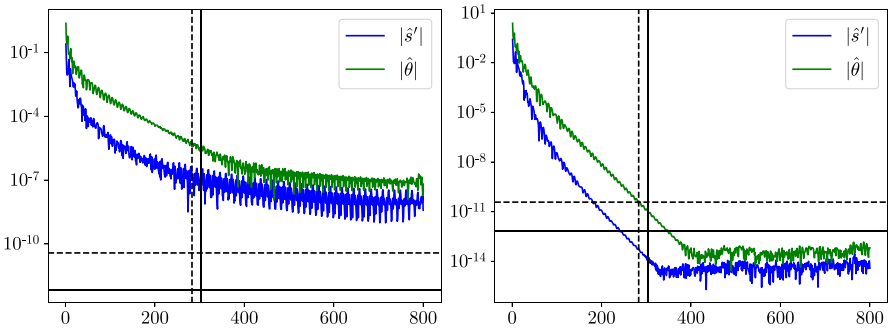
(a) Initial curve (b) Final curve

Fig. 27 The result of our algorithm applied to Figure 5 in [12]. The red dots mark the sample data points. Although the two curves are visually indistinguishable, the curve on the right can be represented with fewer coefficients

displayed in Fig. 32. It is noteworthy that the shape of the curve has been preserved, and our algorithm produces a curve represented by only 1700 coefficients, while the algorithm in [12] produces a curve in Fig. 6 represented by $2 \cdot 2300 = 4600$ coefficients. Moreover, the number of coefficients required to represent the over-sampled curve in Fig. 30b increases only moderately when compared to those of the curve in Fig. 27b.

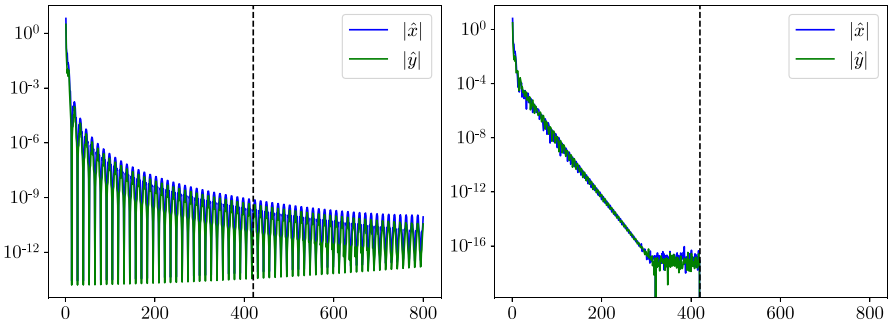
For Figure 3 in [12], reproduced in Fig. 33, we scale the sample data points so that their height is 1 and apply our algorithm to the initial curve in Fig. 34a, with $N = 2000$, $n = 40$, $n_{\text{iters}} = 70$, $h_{\text{filter}} = \frac{1}{40}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 690$, $n_{\text{bands}} = 4$. Although the difference can not be distinguished visually, after $n_{\text{stop}} = 45$ iterations, 690 coefficients are necessary to represent the curve in Fig. 34b, to within an error of $E_{\text{samp}} = 0.27535 \cdot 10^{-13}$. The magnitudes of the coefficients of $s'(t)$ and $\theta(t)$, for both the initial and final curves, are displayed in Fig. 35, and the magnitudes of the coefficients of both the initial and final curves are displayed in Fig. 36. Note that the algorithm in [12] requires approximately $2 \cdot 7000 = 14,000$ coefficients to fit a curve passing through the same sample data points.

We also present the following examples to demonstrate the differences between our method and the method of [12], as described in Sect. 3.10.



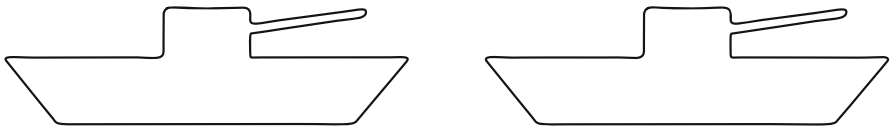
(a) Before smoothing (b) After smoothing

Fig. 28 Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 27. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 303rd coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 283rd coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



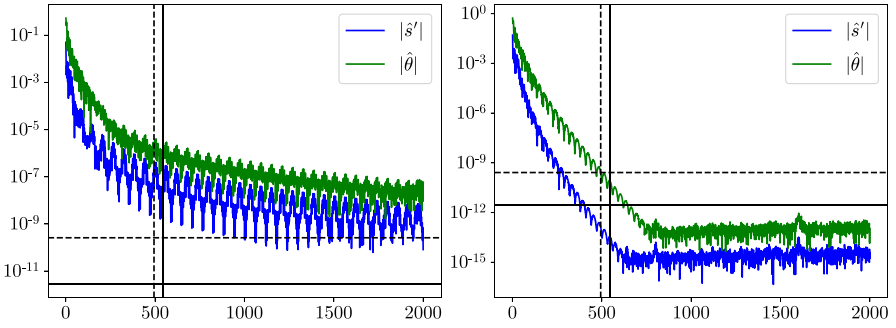
(a) Before smoothing (b) After smoothing

Fig. 29 Fourier coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 27. The value of n_{coefs} is indicated by a vertical dashed line



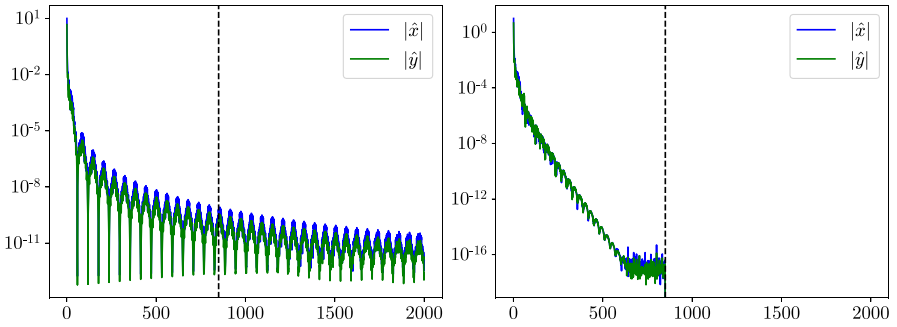
(a) Initial curve (b) Final curve

Fig. 30 The result of our algorithm applied to Figure 5 in [12], with more sample data points. Due to the large quantity and non-uniform distribution of the sample data points, we choose not to display them in the plot. Although the two curves are visually indistinguishable, the curve on the right can be represented with fewer coefficientsption



(a) Before smoothing (b) After smoothing

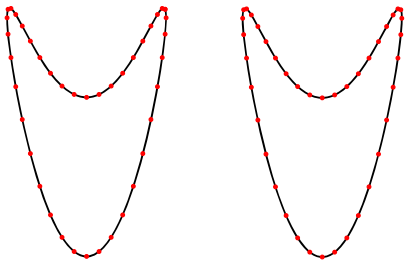
Fig. 31 Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 30. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 545th coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 494th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



(a) Before smoothing (b) After smoothing

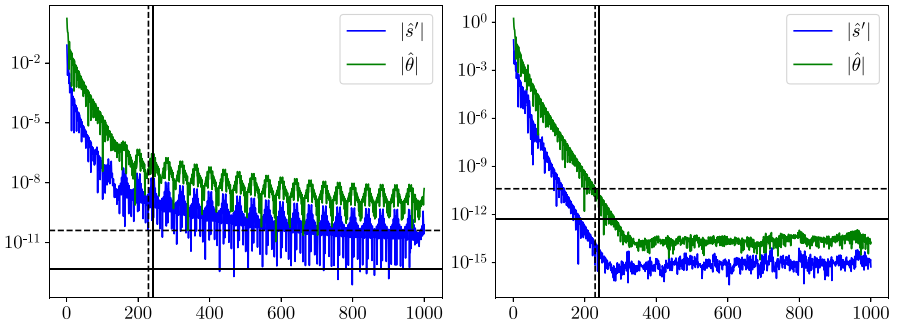
Fig. 32 Fourier coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 30. The value of n_{coefs} is indicated by a vertical dashed line

Fig. 33 Figure 3 in [12]



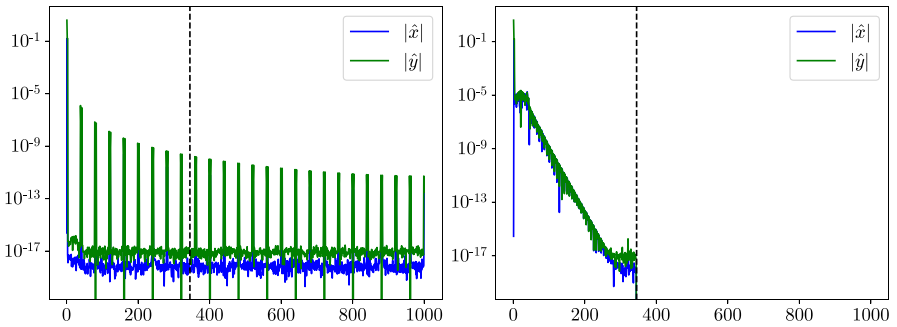
(a) Initial curve (b) Final curve

Fig. 34 The result of our algorithm applied to Figure 3 in [12]. The red dots mark the sample data points. Although the two curves are visually indistinguishable, the curve on the right can be represented with fewer coefficients



(a) Before smoothing (b) After smoothing

Fig. 35 Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 34. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 241st coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 229th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



(a) Before smoothing (b) After smoothing

Fig. 36 Fourier coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 34. The value of n_{coefs} is indicated by a vertical dashed line

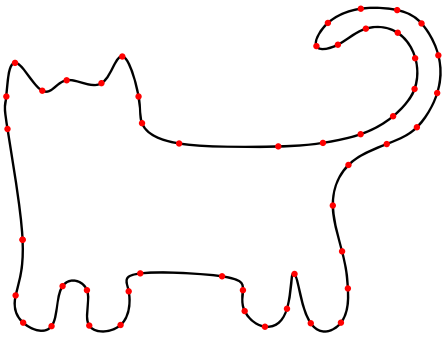
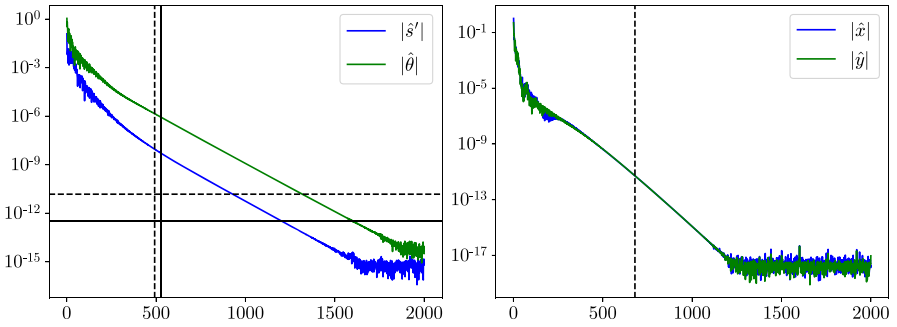
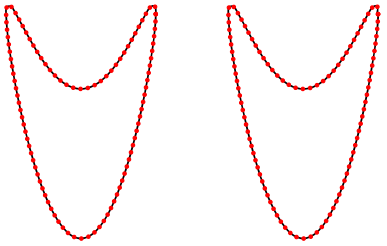


Fig. 37 The result of our algorithm with a single step of filtering, applied to the same data points as in Fig. 21a



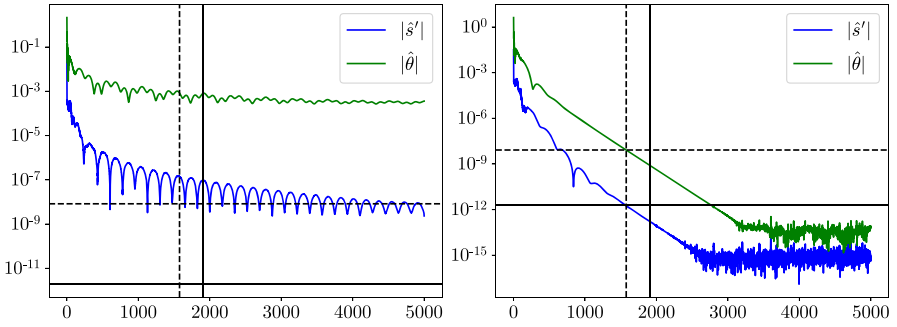
(a) Fourier coefficients of $s'(t)$ and $\theta(t)$ (b) Fourier coefficients of $x(t)$ and $y(t)$

Fig. 38 Fourier coefficients of the resulting curve corresponding to Fig. 37



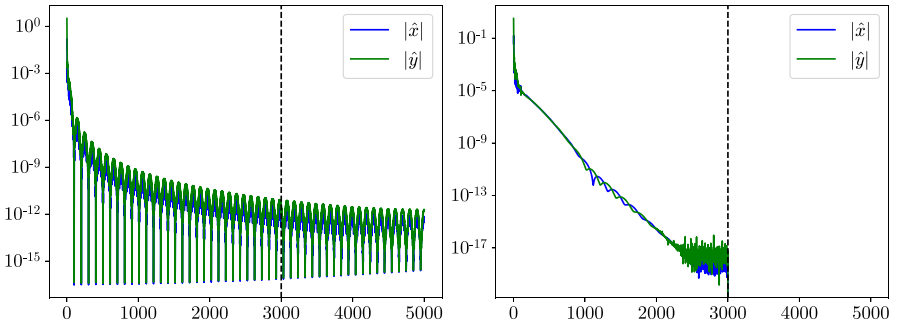
(a) Initial curve (b) Final curve

Fig. 39 The result of our algorithm applied to equispaced sample data points in arc-length on the curve in Fig. 34a. The red dots mark the sample data points. Although the two curves are visually indistinguishable, the curve on the right can be represented with fewer coefficients



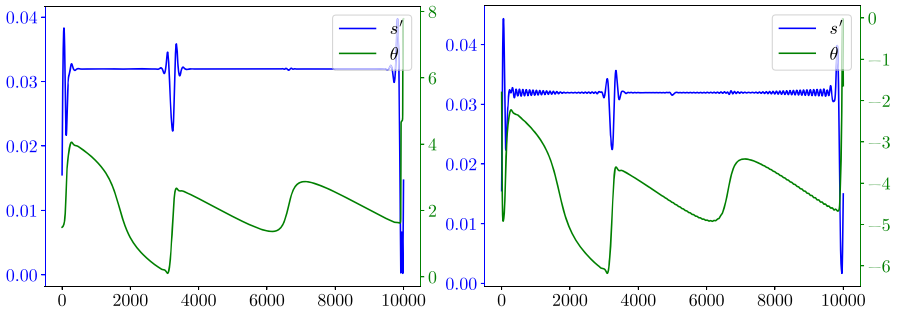
(a) Before smoothing (b) After smoothing

Fig. 40 Fourier coefficients of $s'(t)$ and $\theta(t)$ corresponding to Fig. 39. The value of $\delta_{s'}$ is indicated by a horizontal solid line, and the value of δ_θ is indicated by a horizontal dashed line. The 1912nd coefficient of $s'(t)$ decays to $\delta_{s'}$, indicated by a vertical solid line. The 1577th coefficient of $\theta(t)$ decays to δ_θ, indicated by a vertical dashed line



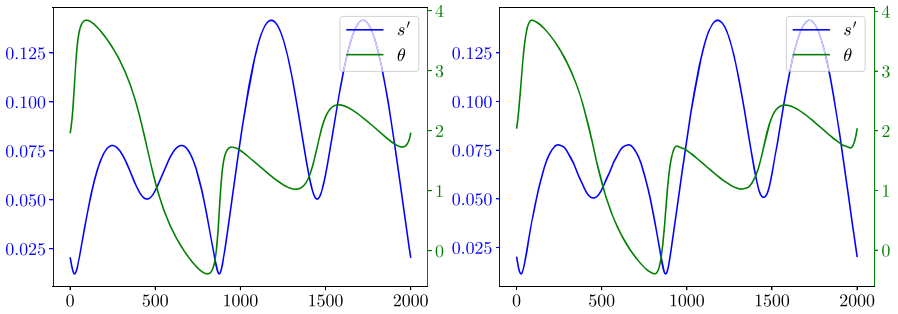
(a) Before smoothing (b) After smoothing

Fig. 41 Fourier coefficients of $x(t)$ and $y(t)$ corresponding to Fig. 39. The value of n_{coefs} is indicated by a vertical dashed line



(a) Before smoothing (b) After smoothing

Fig. 42 $s'(t)$ and $\theta(t)$ corresponding to Fig. 39



(a) Before smoothing (b) After smoothing

Fig. 43 $s'(t)$ and $\theta(t)$ corresponding to Fig. 34

Recalling the discussion in Sect. 3.10, a key difference between our method and the method of [12] is our use of iterative filtering. Suppose that, instead of iteratively filtering the curve, we use a single step of filtering to directly filter the coefficients of the representations of the initial curve to the desired bandwidth. This is similar to the method of [12]. We apply our algorithm to the same set of sample data points defining the cat, as depicted in Fig. 21a, with the same settings: $n = 50$, $N = 4000$, $\epsilon = 10^{-15}$, $n_{\text{coefs}} = 1360$, $n_{\text{bands}} = 4$. However, we replace a_1, which determines the bandwidth of the Gaussian filter at the first iteration, by $a_1' = a_1(1 - h_{\text{filter}})^{n_{\text{stop}}}$, where $h_{\text{filter}} = 1/50$ and $n_{\text{stop}} = 70$ are the same as in the iterative filtering case. This ensures that the total amount of filtering remains the same, so that, before adding perturbations, the coefficients of the curve are filtered to have the desired bandwidth. The resulting curve is displayed in Fig. 37, and the magnitudes of the Fourier coefficients used to represent the curve are shown in Fig. 38. The curve appears less smooth than the one shown in Fig. 21b, and the coefficients are not filtered to zero after $n_{\text{coefs}} = 1360$ terms. Since substantial perturbations are added to the reconstructed curve after a large of amount of filtering, the bandwidth of the curve is destroyed. This effect is demonstrated by both the visual appearance of the curve and the large bandwidths of the representations of the curve.

Another key difference between our method and the method of [12] is the use of different parametrizations of the curve. In our method, we allow non-arc-length parametrizations. In order to imitate the arc-length parametrization used in [12] for the purpose of comparison, we employ equispaced sample data points in arc-length on the curve in Fig. 34a. To preserve the original shape of the curve, especially around areas with high curvature, we choose a large number of sample data points, $n = 101$, and scale the points so that their height is 1. We apply our algorithm to the curve in Fig. 39a, with $N = 10,000$, $n_{\text{iters}} = 60$, $h_{\text{filter}} = \frac{1}{40}$, $\epsilon = 10^{-16}$, $n_{\text{coefs}} = 6000$, $n_{\text{bands}} = 4$. After $n_{\text{stop}} = 35$ iterations, 6000 coefficients are sufficient to represent the curve in Fig. 39b, and the error between the final curve and the sample data points is $E_{\text{samp}} = 0.33319 \cdot 10^{-13}$. The magnitudes of the coefficients of $s'(t)$ and $\theta(t)$ are displayed in Fig. 40, and the magnitudes of the coefficients of both the initial and final curves are displayed in Fig. 41. In Fig. 42, we plot $s'(t)$ and $\theta(t)$ for both the initial and final curves. Additionally, we plot $s'(t)$ and $\theta(t)$ for the curves in Fig. 34 in Fig. 43. It can be seen from the plots that the use of non-uniform sample data points in Fig. 34a, together with the non-arc-length parametrization in our algorithm, results in smoother $s'(t)$ and $\theta(t)$. All these observations confirm that arc-length parametrizations can lead to significantly larger bandwidths for the representations of the curve.

5 Conclusion

Our algorithm produces a bandlimited curve passing through a set of points, up to an accuracy of machine precision. It first constructs a C^2 Bézier spline passing through the points, and then recursively applies a Gaussian filter to both the derivative of the arc-length function and the tangential angle of the curve, to control the bandwidth of their Fourier or Chebyshev coefficients, followed by smooth corrections. The resulting curve can be represented by a small number of coefficients and resem-

bles a smooth curve drawn naturally by hand, free of ringing artifacts. The algorithm costs $O(N \log N)$ operations at each iteration, where N is the number of discretization nodes, and the cost can be further reduced by calling the FFT in the FFTW library [15], in which the speed of the FFT routines is optimized for inputs of certain sizes.

One possible extension of this paper is to design an algorithm for curves and surfaces in $\mathbb{R}^3$. The main methodology is still applicable, if we parametrize a curve in $\mathbb{R}^3$ by a function $\gamma(t)\colon I \to \mathbb{R}^3$, where $I \subset \mathbb{R}$, in terms of the same parameter t as in this paper, and a surface in $\mathbb{R}^3$ by a function $\gamma(t_1, t_2)\colon I_1 \times I_2 \to \mathbb{R}$, where $I_1, I_2 \subset \mathbb{R}$, in terms of both t_1 and t_2. We can apply the Chebyshev or the Fourier approximation in each parameter, depending on whether the curve or surface is periodic in that parameter, filter the coefficients, and add smooth perturbations in a similar way. Another application is to implement the algorithm of this paper as a geometric primitive in CAD/CAM systems. Since primitives are generally defined as level sets of polynomials (see Chapter 2 of [16]), the techniques in this paper could be used for the constructions of more general C^∞ shapes in CAD/CAM systems.

Funding This work was supported in part by the NSERC Discovery Grants RGPIN-2020-06022 and DGECR-2020-00356.

Data Availability The implementation of our algorithm is provided in https://doi.org/10.5281/zenodo.7742917.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Akima, H.: A new method of interpolation and smooth curve fitting based on local procedures. J. ACM **17**(4), 589–602 (1970). https://doi.org/10.1145/321607.321609
2. Björkenstam, U., Westberg, S.: General cubic curve fitting algorithm using stiffness coefficients. Comput. Aided Des. **19**(2), 58–64 (1987). https://doi.org/10.1016/S0010-4485(87)80046-5
3. Bica, A.M.: Optimizing at the end-points the Akima's interpolation method of smooth curve fitting. Comput. Aided Geom. Des. **31**(5), 245–257 (2014). https://doi.org/10.1016/j.cagd.2014.03.001
4. Knott, G.D.: Interpolating cubic splines. Birkhäuser, Boston (2000). https://doi.org/10.1007/978-1-4612-1320-8
5. Piegl, L., Tiller, W.: The NURBS book. Springer, Berlin (1995). https://doi.org/10.1007/978-3-642-97385-7
6. Runions, A., Samavati, F.F.: Partition of unity parametrics: a framework for meta-modeling. Vis. Comput. **27**(6), 495–505 (2011). https://doi.org/10.1007/s00371-011-0567-x
7. Zhang, R., Ma, W.: An efficient scheme for curve and surface construction based on a set of interpolatory basis functions. ACM Trans. Graph. **30**(2), 1–11 (2011). https://doi.org/10.1145/1944846.1944850
8. Runions, A., Samavati, F.: CINPACT-splines: a class of C^∞ curves with compact support. Curves and Surfaces: 8th International Conference, Paris, France, June 12-18, 2014, Revised Selected Papers (2015). https://doi.org/10.1007/978-3-319-22804-4_27
9. Akram, B., Alim, U.R., Samavati, F.F.: CINAPACT-splines: a family of infinitely smooth, accurate and compactly supported splines. Advances in Visual Computing (2015). https://doi.org/10.1007/978-3-319-27857-5_73
10. Blu, T., Thcvenaz, P., Unser, M.: MOMS: maximal-order interpolation of minimal support. IEEE Trans. Image Process. **10**(7), 1069–1080 (2001). https://doi.org/10.1109/83.931101

11. Zhu, Y.: A class of blending functions with c^∞ smoothness. Numerical Algorithms **88**(2), 555–582 (2021). https://doi.org/10.1007/s11075-020-01049-7
12. Beylkin, D., Rokhlin, V.: Fitting a bandlimited curve to points in a plane. SIAM J. Sci. Comput. **36**(3), 1048–1070 (2014). https://doi.org/10.1137/130932703
13. Joost, M.: Cubic Bézier splines. Available online at (2011). https://www.michael-joost.de/bezierfit.pdf
14. Thompson, M.T.: Intuitive analog circuit design. Newnes, Burlington (2006). https://doi.org/10.1016/B978-075067786-8/50000-8
15. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE **93**(2), 216–231 (2005). Package is available at http://fftw.org/
16. Hoffmann, C.M.: Geometric and solid modeling, (2002). Available online at https://www.cs.purdue.edu/homes/cmh/distribution/books/geo.html