Journal of Engineering Technology and Applied Sciences, 2026 *e-ISSN:* 2548-0391
Received 20 November 2025, Accepted 8 April 2026 *Vol. 11 , 65-90*
Published online 12 September 2026, Research Article *doi: 10.30931/jetas.1817031*

Citation: Kapçak, S., "Fitting a Closed Curve to Unordered Planar Points". Journal of Engineering Technology and Applied Sciences 11 2026 : 65-90.

FITTING A CLOSED CURVE TO UNORDERED PLANAR POINTS

Sinan Kapçak

Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands
s.kapcak@tue.nl

Abstract

Fitting smooth closed curves to unordered planar point clouds arises in various applications, including shape reconstruction, image processing, geometric modelling, and computational geometry. We propose a numerical framework for this problem under an approximate star-shaped assumption. The method combines a deterministic centre-selection strategy with a trigonometric least-squares approximation of the radial function. By minimising a radial ambiguity functional, an interior centre is identified to enable a well-defined polar representation without point ordering or neighbourhood reconstruction. A truncated Fourier series is then fitted to obtain a smooth periodic parametric curve. In contrast to approaches that require point ordering, neighbourhood reconstruction, or nonlinear optimisation, the proposed framework reduces the two-dimensional fitting problem to a one-dimensional least-squares approximation in the radial coordinate, resulting in a smooth analytic representation obtained through a linear and computationally efficient procedure. Accuracy is evaluated using normalised RMSE and maximum point-to-curve deviation. Experiments on a 250-point unstructured dataset and on invariant curves arising from dynamical systems demonstrate accurate approximation with moderate Fourier order. The method is effective for planar point clouds that admit an approximate star-shaped structure.

Keywords: Curve fitting, unordered planar point, closed curve, Neimark-Sacker, discrete dynamical system

1. Introduction

Planar datasets are frequently provided as discrete point samples, while the underlying geometric object is not explicitly given. Depending on the modelling objective and the structure of the data, different problem settings arise.

In many applications, the points are assumed to lie near a one-dimensional curve that represents the object of interest. If the samples are ordered or naturally parameterised, the task is typically interpolation or smooth parametrisation. Spline-based representations such as B-splines and

periodic splines are widely used in this context due to their flexibility and local control properties [1, 2, 3, 9]. Curve fitting plays a central role in geometric modelling, computer-aided design (CAD), computer graphics, and computer-aided manufacturing (CAM) [9, 5]. It is also fundamental in approximation theory, optimisation, statistics, image processing, and data compression [7, 8].

When the data are unordered but still approximate a single closed contour, the problem becomes one of curve fitting without explicit connectivity information. Modern spline-based approaches address this setting using geometric distance minimisation and curvature-based optimisation criteria. For example, Wang, Pottmann, and Liu [4] proposed curvature-based squared distance minimisation for B-spline fitting to point clouds, improving geometric fidelity. More recently, Wang et al. [12] introduced a robust fitting framework based on a nonlinear Gauss–Helmert model, emphasising statistical consistency and noise robustness. Related reconstruction strategies also arise in boundary modelling and numerical applications [6].

In more general point-cloud settings, the objective may be structural reconstruction rather than direct curve fitting. The dataset may contain interior samples together with boundary points, and the goal is to extract an outer contour. Such problems are common in geometric modelling and image-based shape analysis [7]. In these cases, neighbourhood reconstruction and topology recovery become central tasks.

The present work focuses on the intermediate case of unordered planar samples that approximate a single closed curve and admit an approximate star-shaped structure. Rather than reconstructing connectivity or solving nonlinear geometric optimisation problems, we propose a simple and computationally efficient fitting framework based on polar transformation and trigonometric least-squares approximation. The key idea is to select an interior centre, transform the data into angle–radius coordinates, and approximate the radial function by a truncated Fourier series. This produces a smooth, global, and analytically compact representation without requiring point ordering.

We apply the method to two distinct settings. The first considers an unstructured, noisy planar point cloud where no ordering information is available. The second examines an invariant curve generated by a Neimark–Sacker bifurcation, where the boundary is smooth and naturally closed.

The remainder of the paper is organised as follows. Section 2 introduces the fitting framework, including the centre-selection strategy and the trigonometric least-squares formulation. Section 3 presents the two applications. Finally, Section 4 provides discussion and concluding remarks.

2. Method

The method assumes that the input point cloud is approximately star-shaped with respect to a suitable centre. That is, from this centre, each radial direction must intersect the dataset at most once. This condition ensures that the transformation to polar coordinates yields a well-defined angle–radius mapping. The centre is determined automatically using the radial ambiguity minimisation strategy described in Section 2.2.

Let $P = \{(x_1, y_1), \dots, (x_k, y_k)\}$ be an unordered set of planar points approximating a closed shape, and let $c = (c_x, c_y)$ denote the selected centre. We compute polar coordinates relative to c:

$$r_i = \sqrt{(x_i - c_x)^2 + \left(y_i - c_y\right)^2}, \quad t_i = \arctan2\left(y_i - c_y,\, x_i - c_x\right) \tag{1}$$

Here, the function $\arctan2(y, x)$ computes the angle from the positive x-axis to the point (x, y), returning a value in $(-\pi, \pi]$. It is defined as:

$$\arctan2(y,x) = \begin{cases} 2\tan^{-1}\left(\dfrac{y}{x + \sqrt{x^2 + y^2}}\right) & x > 0, \\ 2\tan^{-1}\left(\dfrac{-x + \sqrt{x^2 + y^2}}{y}\right) & x \le 0,\ y \ne 0, \\ \pi & x < 0,\ y = 0, \\ \text{undefined} & x = 0,\ y = 0. \end{cases}$$

This transformation produces a dataset

$$\bar{P} = \{(t_1, r_1), \ldots, (t_k, r_k)\},$$

which we use to construct a smooth, periodic approximation by fitting a trigonometric model of the form:

$$r(t) = \sum_{i=0}^{n} \left(a_i \sin(it) + b_i \cos(it)\right).$$

The coefficients $\{a_i, b_i\}$ are computed using the linear least squares method. Although the model includes sine and cosine terms, which are nonlinear functions of the independent variable t, the dependence on the coefficients is linear. This allows the problem to be formulated as a system of linear equations whose solution yields the best-fitting parameters in the least-squares sense. The resulting model is inherently smooth and 2π-periodic, and provides a compact and analytically convenient representation of the radial structure of the data.

Once the radius function $r(t)$ is fitted, the corresponding closed curve in Cartesian coordinates is recovered as

$$\Gamma(t) = \left(c_x + r(t)\cos t,\ c_y + r(t)\sin t\right), \quad -\pi < t \le \pi.$$

2.1 Algorithm overview

The following steps outline the procedure used to fit a closed curve to a set of unordered planar points using centre selection, polar transformation, and trigonometric approximation:

1. Input: A set of planar data points approximating a closed shape; order n of the trigonometric model.

2. Centre Selection: Determine a suitable centre $c = \left(c_x, c_y\right)$ by minimising the radial ambiguity score

$$S(c) = \sum_{j=1}^{B} \mathrm{IQR}_j ,$$

where the angular domain is partitioned into B bins and IQR_j denotes the interquartile range of radii in bin j. The minimisation is performed using a multi-scale grid search starting from the coordinate-wise median of the data (see Section 2.2).

3. Polar Transformation: For each data point (x, y), compute polar coordinates relative to the selected centre:

$$r = \sqrt{(x - c_x)^2 + \left(y - c_y\right)^2}, \quad t = \arctan2\left(y - c_y,\, x - c_x\right).$$

This yields a set of angle–radius pairs (t, r), provided the star-shaped condition is approximately satisfied.

4. Model Construction: Define a trigonometric model of the form:

$$r(t) = \sum_{i=0}^{n} \left(a_i \sin(it) + b_i \cos(it)\right).$$

5. Coefficient Estimation: Use linear least squares to determine the coefficients $\{a_i, b_i\}$ that best fit the (t, r) data.

6. Curve Reconstruction: Reconstruct the curve in Cartesian coordinates:

$$\Gamma(t) = \left(c_x + r(t)\cos t,\ c_y + r(t)\sin t\right), \quad -\pi < t \leq \pi.$$

7. Output: Display the resulting closed parametric curve $\Gamma(t)$ overlaid on the original dataset.

2.2 Centre selection via radial ambiguity minimisation

A central requirement of the proposed polar fitting framework is that the underlying point cloud be approximately star-shaped with respect to the chosen origin. More precisely, the polar transformation must produce a single-valued mapping $t \mapsto r(t)$, meaning that each angular direction intersects the dataset in at most one radial layer of points. If this condition is violated, multiple data points may share nearly identical angular coordinates while having significantly different radii, leading to ambiguity in the radial representation and instability in the subsequent trigonometric regression.

To address this issue in a systematic and reproducible manner, we introduce a radial ambiguity functional that quantifies the degree to which a candidate centre produces a single-valued radial structure. Let $c = \left(c_x, c_y\right)$ be a candidate centre. For each data point (x_i, y_i), we compute its polar coordinates relative to c:

$$t_i = \arctan2\left(y_i - c_y, x_i - c_x\right), \qquad r_i = \sqrt{(x_i - c_x)^2 + \left(y_i - c_y\right)^2}.$$

The angular domain $(-\pi, \pi]$ is discretised into B uniform bins. Within each angular bin, we collect the associated radial values and measure their dispersion using the interquartile range (IQR), defined as the difference between the third and first quartiles of the radii. The IQR provides a robust estimate of radial spread that is insensitive to isolated outliers. If the dataset is locally single-layered in a given angular sector, the radii in that bin are tightly clustered and the IQR is small. Conversely, if multiple radial layers occur in the same direction—indicating a violation of approximate star-shapedness—the IQR becomes large.

The total radial ambiguity score for the centre c is defined as

$$S(c) = \sum_{j=1}^{B} \mathrm{IQR}_j,$$

where IQR_j denotes the interquartile range in bin j. Optionally, a small penalty term may be added for empty bins to discourage centres that lie outside the effective interior of the dataset. The optimal centre is then selected as

$$c^* = \operatorname*{argmin}_c S(c),$$

that is, the point that minimises overall radial dispersion across angular sectors. Geometrically, this corresponds to finding a location from which the point cloud appears most radially coherent.

Since the ambiguity functional is non-smooth and derivative information is not readily available, we minimise $S(c)$ using a multi-scale grid search. Starting from a robust initial estimate given by the coordinate-wise median of the data, we evaluate $S(c)$ on a finite grid of candidate centres within a bounding box proportional to the dataset diameter. The best candidate is selected, and the search region is iteratively shrunk and refined. This coarse-to-fine strategy provides a deterministic and reproducible optimisation procedure with modest computational cost. For typical datasets considered in this work, the total number of evaluations remains small, and the resulting centre is stable under moderate perturbations.

It is important to note that this strategy does not compute the exact geometric kernel of the point set. Instead, it approximates star-shapedness in a statistical sense by minimising radial dispersion. If no centre yields uniformly small ambiguity—such as in shapes with deep concavities or multiple disconnected components—the method correctly signals that the polar representation is unsuitable. This clarifies the geometric limitation of the approach and explains why strongly non–star-shaped datasets cannot be reliably processed using the proposed framework.

3. Applications

In this section, we apply the proposed method to two representative datasets with different structural properties. The first example is a 250-point dataset, representing an unstructured, noisy, concave cloud where point ordering is unavailable. The second example concerns an invariant curve arising from a Neimark–Sacker bifurcation, illustrating a case from dynamical systems where the boundary is smooth and naturally closed. Together, these applications

illustrate the performance of the method on both unstructured noisy datasets and on datasets sampled from recognisable closed geometric structures.

3.1 Fitting Noisy and Concave Shapes

In this section, we demonstrate the complete fitting pipeline on a concrete unordered and noisy planar dataset consisting of 250 points representing a concave shape. The data are unstructured: no point ordering, neighbourhood relationships, or connectivity information are assumed. Our objective is to (i) select a centre automatically so that the polar representation becomes as close as possible to single-valued, and then (ii) fit a smooth closed curve using the trigonometric regression model. The procedure is implemented in SageMath creating an interactive tool (See Appendix B).

3.1.1 Data set

Let $P = \{(x_i, y_i)\}_{i=1}^{250} \subset \mathbb{R}^2$ be the given point set (listed explicitly in the Appendix A). Figure 1a shows the data in Cartesian coordinates. The points form a noisy closed contour with non-uniform sampling density. Since the data are unordered, a direct periodic spline fit would require an additional ordering or boundary-reconstruction step. Our approach instead constructs a closed curve by fitting a periodic radius function in polar coordinates, provided a suitable centre is chosen.

3.1.2 Automatic centre selection

A key requirement for the polar fitting approach is that the mapping $t \mapsto r(t)$ be approximately single-valued. Geometrically, this corresponds to the point set being approximately star-shaped with respect to the selected centre $c = (c_x, c_y)$: along most angular directions, the data should be concentrated in a single radial layer. If the centre is poorly chosen, multiple radial layers may occur for the same angle, leading to ambiguity in the (t, r) representation and an unstable regression.

To choose c automatically, we minimise a *radial ambiguity score* based on angular binning and the interquartile range (IQR).

Polar coordinates about a candidate centre.

For a candidate centre c, we compute polar coordinates of each point relative to c:

$$t_i = \arctan2(y_i - c_y,\ x_i - c_x), \qquad r_i = \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2}.$$

This produces a set of angle–radius pairs $\bar{P} = \{(t_i, r_i)\}_{i=1}^{250}$.

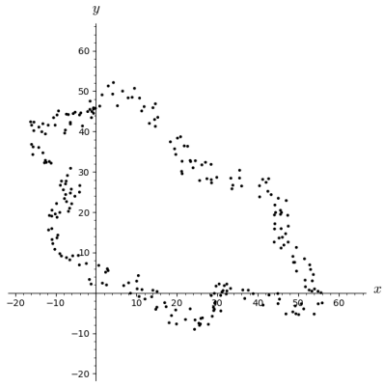
Angular binning and robust radial dispersion.

We discretise the angular interval $(-\pi, \pi]$ into B equal bins (See Figure 1b). Within each bin we collect the radii of all points whose angles fall into that bin. If, for a given direction, the data are concentrated around a single boundary intersection, then the radii within that bin have small spread. If instead two radial layers occur (a typical symptom of non–star-shaped behaviour

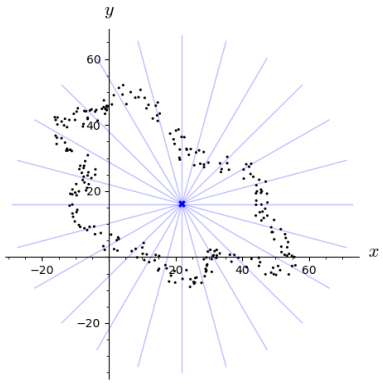
relative to c), the radii in that bin become widely separated. To quantify spread robustly, we use the interquartile range:

$$\mathrm{IQR} = Q_3 - Q_1,$$

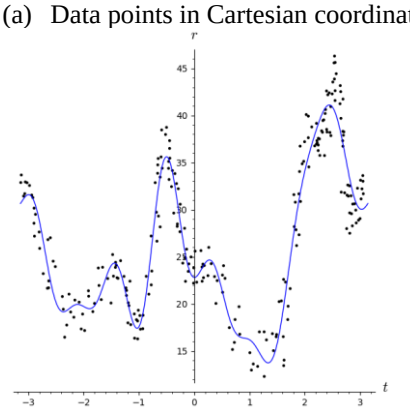
where Q_1 and Q_3 denote the first and third quartiles of the radii in the bin. The IQR measures the width of the middle 50% of the data in that bin and is less sensitive to isolated outliers than the full range or variance.



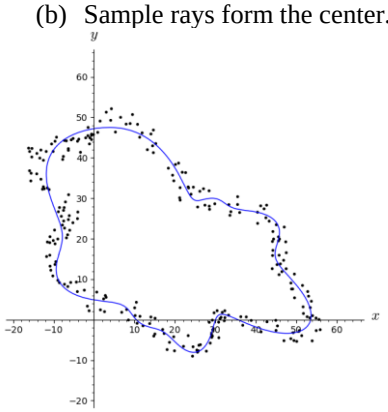
(a) Data points in Cartesian coordinates.



(b) Sample rays form the center.



(c) Fitting curve and raw points in t-r plane.



(d) Final fitted curve overlaid on the original dataset

Figure 1. Application of the method to data given in Appendix A. In (b), sample rays from the automatically selected centre are given. The centre is chosen to minimise radial ambiguity, so that most angular directions correspond to a single dominant radial layer.

Ambiguity score.

The total ambiguity score is defined by

$$S(c) = \sum_{j=1}^{B} \mathrm{IQR}_j,$$

where IQR_j is the interquartile range computed from the radii in the jth angular bin (computed only when a bin contains at least a prescribed number of points, `min_bin_count` in the code in Appendix B). Optionally, an additional penalty term may be added to discourage centres that

create many empty angular bins. In the present experiment we used $B = 180$, `min_bin_count=2`, and zero empty-bin penalty.

Multi-scale grid search minimization.

The functional $S(c)$ is evaluated numerically and is not differentiable due to angular binning. We therefore minimise it by a deterministic multi-scale grid search:

1. Initial centre: start from the coordinate-wise median of the data (robust to outliers).

2. Coarse search: evaluate $S(c)$ on an $N \times N$ grid in a square window around the current best centre. The half-width of this window is set to `init_box_scale` times the data diameter (bounding-box diagonal).

3. Refinement: move to the best grid point, shrink the search window by a factor `shrink`, and repeat for a fixed number of rounds `rounds`.

For this application we used `grid_N=21`, `rounds=4`, `init_box_scale=0.35`, and `shrink=0.35`. Since the optimisation is only two-dimensional, the total number of objective evaluations remains small (here on the order of $\texttt{rounds} \cdot \texttt{grid_N}^2$), making the procedure computationally inexpensive and reproducible.

Selected centre and diagnostics.

For the present data set, the algorithm returned

$c = (21.9493484024403,\ 16.0912763937185),$

with ambiguity score $S(c) = 169.0483933879094$. The optimisation reported 40 empty bins out of 180. Figure 1b shows the selected centre overlaid on the data as well as the sample rays emanating from c; most rays encounter a single outer "band" of points, indicating that the polar representation is sufficiently well-defined for regression.

3.1.3 Polar transformation and trigonometric regression

With the selected centre c fixed, we transform the data to $\bar{P} = \{(t_i, r_i)\}$ using equations in (1). We then fit the trigonometric model

$$r(t) = \sum_{k=0}^{n} \big(a_k \sin(kt) + b_k \cos(kt)\big), \tag{2}$$

by linear least squares (`find_fit` in SageMath), leading to the coefficients $\{a_k, b_k\}$, whose numerical values are presented in Table 1. Figure 1c shows the fitted function $r(t)$ overlaid on the (t, r) point cloud.

In this example, the Fourier order $n = 8$ is chosen as a representative moderate order that provides accurate approximation while avoiding unnecessary model complexity.

3.1.4 Curve reconstruction in cartesian coordinates

Finally, the fitted closed curve is reconstructed in Cartesian coordinates by
$\Gamma(t) = \left(c_x + r(t)\cos t,\ c_y + r(t)\sin t\right), \qquad -\pi < t \leq \pi.$
Figure 1d shows the resulting smooth closed curve together with the original point set. The curve provides a global, smooth approximation of the contour while filtering out local noise and sampling irregularity.

Table 1. Coefficients a_k and b_k for the given noisy and concave dataset.

k	a_k	b_k
0	0.000000000000000	22.812737345241420
1	1.386647856326355	-3.993135129312179
2	-6.969182974196633	4.463359034442801
3	1.115376478760630	2.490857864931326
4	0.008421530035867	-0.890183712837973
5	-2.491541350356785	-1.278817676769828
6	0.537910681589234	-1.376210937619389
7	0.410350619452240	-1.140750206082715
8	1.846284152668821	-0.684939838947184

3.1.5 Quantitative evaluation

To quantify the approximation quality, we compute point-to-curve distances by sampling $\Gamma(t)$ uniformly (here 3000 samples) and, for each data point $p \in P$, taking the minimum Euclidean distance to the sampled curve. From these distances we report:

- **RMSE:** $\sqrt{\frac{1}{|P|}\sum_{p\in P} d\,(p,\Gamma)^2}$, measuring average deviation.
- **Max error:** $\max_{p\in P} d(p,\Gamma)$, a Hausdorff-like one-sided worst-case deviation.

For this data set we obtained

$$\mathrm{RMSE} = 1.9030588647687372, \qquad \mathrm{Max} = 5.633906279728659.$$

To remove scale dependence, we normalise by the data diameter D (bounding-box diagonal), yielding

$$\mathrm{RMSE}/D = 0.02013121308140871, \qquad \mathrm{Max}/D = 0.059597403894118085.$$

Thus, the fitted curve achieves an average deviation of approximately 2.0% of the data diameter, with a worst-case deviation of approximately 6.0%. These values indicate that the method yields a strong global approximation for this unordered and unstructured contour data set.

3.2 Neimark–Sacker invariant curve

The Neimark–Sacker bifurcation is a classical mechanism in discrete-time dynamical systems, leading to the birth of a closed invariant curve when a fixed point loses stability via a pair of complex-conjugate multipliers crossing the unit circle. This bifurcation is the discrete analogue

of the continuous-time Hopf bifurcation and typically results in quasi-periodic behavior on a closed invariant curve [10, 11].

We apply the proposed method to numerically approximate the invariant closed curve arising from a Neimark–Sacker bifurcation. This approximation relies entirely on simulated data and does not make use of asymptotic or symbolic expansions. As a result, it may not capture the detailed analytical structure near the bifurcation point. However, it offers a practical and broadly applicable alternative in cases where such expansions are unavailable, intractable, or unnecessary.

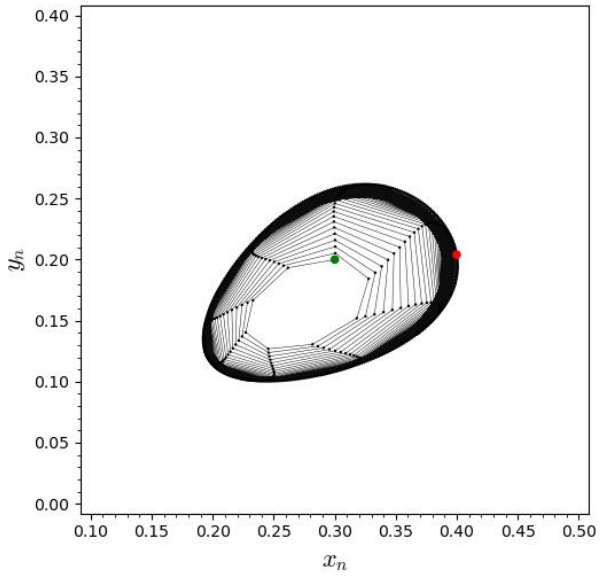


Figure 2. Phase diagram of the nonlinear system (3) for $\phi = 2.45$, $\beta = 0.754$, $c = 9.625$. The orbit starting from the initial condition $(0.3,0.2)$, marked by a green dot, is attracted by a closed invariant curve. The terminal point after 600 iterations is shown by a red dot.

To generate the dataset, we simulate a single orbit that converges to the invariant closed curve. This yields a representative sample of points without requiring explicit knowledge of fixed points.

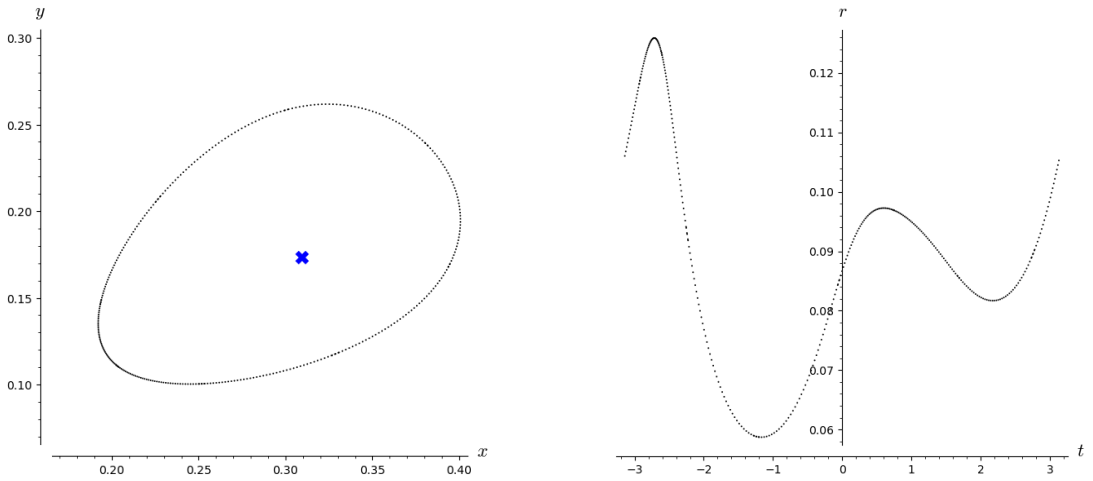
Consider the following nonlinear discrete-time dynamical system describing a predator–prey interaction with a refuge effect [13] :

$$\begin{aligned} x_{n+1} &= \phi(1-\beta)x_n + \phi\beta x_n e^{-cy_n}, \\ y_{n+1} &= \beta x_n(1-e^{-cy_n}), \end{aligned} \tag{3}$$

where ϕ, β, and c are parameters, with $0 < \beta < 1$. It is shown in [13] that, for certain parameter values, system (3) undergoes a Neimark–Sacker bifurcation. Numerical simulations further indicate that trajectories in a neighbourhood of the fixed point form a closed, ring-like structure, which is characteristic of this type of bifurcation. However, the invariant curve associated with this behaviour is neither derived nor approximated analytically in [13]. In this section we will apply our method to obtain a numerical approximation of the closed invariant curve.

Figure 2 illustrates the phase-space dynamics of the system for the initial condition $(0.3,0.2)$. The trajectory appears to converge to a closed invariant curve surrounding the fixed point.

The resulting point cloud is unordered but displays a clearly closed geometry. Using our method, we fit a smooth closed curve that approximates the shape of this invariant set. A total of 1000 iterations were performed, but only the last 350 points—those approaching steady-state behavior—were used for fitting.



(a) Data points in xy-plane. (b) Data points in rt-plane.

Figure 3. (a) Data points obtained by the iteration of system (3) and automatically selected center $c = (0.3096, 0.1733)$ (blue). (b) Data points converted to t and r.

To apply the polar-based fitting procedure, we first determine a suitable centre $c = (c_x, c_y)$ using the radial ambiguity minimisation strategy described in Section 2.2. This automatic centre selection seeks a point relative to which the dataset becomes approximately star-shaped, thereby reducing angular ambiguities in the polar representation.

Once the centre is selected, each data point is expressed in polar coordinates relative to c, yielding radius–angle pairs (t, r). The trigonometric fit is then performed in the t–$r(t)$ domain.

The automatically selected centre (Figure 3a) is used to define the polar transformation, and the resulting t–$r(t)$ representation is shown in Figure 3b. The final smooth curves in the original Cartesian frame, obtained using least-squares trigonometric fits with different orders n, are shown in Figure 4.

Let us take the radius function $r(t)$ to have the following trigonometric form of order ten:

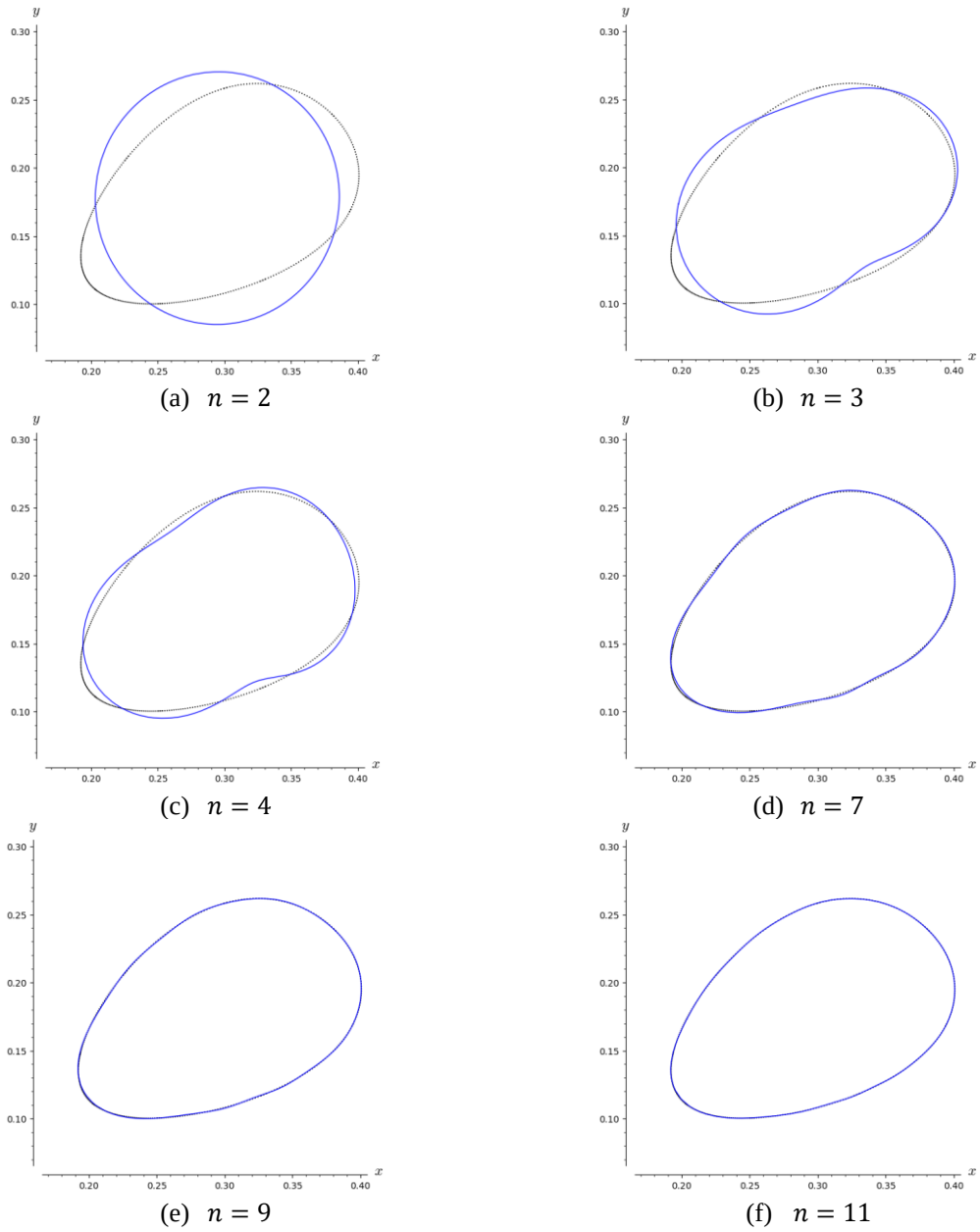
$$r(t) = \sum_{k=0}^{10} \left(a_k \sin(kt) + b_k \cos(kt)\right) \tag{4}$$

where the coefficients $\{a_k\}$ and $\{b_k\}$ were obtained via least-squares regression using the last 350 iterations of the simulated orbit. The specific values are listed in Table 2.

Influence of the Fourier Order.

Table 3 and Figure 4 illustrate the effect of the trigonometric order n on the fitting accuracy. For small values of n, the error decreases rapidly as additional harmonics capture the dominant

geometric features of the dataset. In particular, the normalised RMSE drops from 0.05827 at $n = 2$ to 0.00182 at $n = 8$, representing a substantial improvement in approximation quality.



(a) $n = 2$ (b) $n = 3$ (c) $n = 4$ (d) $n = 7$ (e) $n = 9$ (f) $n = 11$

Figure 4. The effect of varying n on the fitting results for the invariant curve caused by Neimark-Sacker bifurcation.

Table 2. Fourier coefficients a_k and b_k of the fitted radius function $r(t)$ in equation (4), obtained from data generated by the system (3) exhibiting a Neimark–Sacker bifurcation.

k	a_k	b_k
0	0.000000000000000	0.086150171367060
1	0.008598599153637	-0.010469872062964
2	0.014683001449922	0.011895072913250
3	-0.005194740878519	-0.001536001411462
4	0.004192905737671	-0.000257346587825
5	-0.001413937961487	0.001573590512328
6	0.000632278426818	-0.001151094770659

k	a_k	b_k
7	0.000153662756060	0.000806522570745
8	-0.000270792925044	-0.000400896972533
9	0.000334931338401	0.000157465483252
10	-0.000243663018250	0.000026696530005

For intermediate orders ($8 \leq n \leq 12$), the error continues to decrease, but at a slower rate. Beyond approximately $n = 12$, the improvement becomes marginal. For example, increasing the order from $n = 15$ to $n = 20$ yields only a negligible reduction in normalised RMSE, while the maximum deviation does not decrease monotonically. This behaviour indicates diminishing returns in accuracy relative to model complexity and suggests the onset of mild oscillatory effects for higher orders.

These results demonstrate that moderate values of n already provide highly accurate approximations, and that the choice of n may be guided by a balance between model simplicity and desired precision.

Table 3. Normalised RMSE and maximum deviation for different Fourier orders n. Errors are normalised by the dataset diameter.

n	norm. RMSE	norm. Max
2	0.05827	0.09103
3	0.02161	0.03994
4	0.01402	0.02515
5	0.00774	0.01469
6	0.00479	0.00921
7	0.00297	0.00569
8	0.00182	0.00360
9	0.00121	0.00237
10	0.00079	0.00157
12	0.00040	0.00079
15	0.00026	0.00053
20	0.00024	0.00057

3.2.1 Comparison with periodic spline interpolation

To provide a classical smooth-curve baseline, we compare the proposed Fourier-based approximation with a periodic cubic spline interpolation. Since the dataset consists of unordered orbit points, the data were first parameterised by their polar angle relative to the selected centre and sorted accordingly. Separate periodic cubic splines were then fitted to the coordinate functions $x(t)$ and $y(t)$ with periodic boundary conditions to ensure closure of the curve. The resulting spline was sampled uniformly over $(-\pi, \pi]$, and performance was evaluated using the same RMSE and maximum point-to-curve deviation metrics employed for the proposed method.

Table 4. Comparison on the Neimark–Sacker dataset (1000 iterations, last 350 points used). Errors are normalised by the dataset diameter.

Method	norm. RMSE	norm. Max
Proposed ($n = 11$)	0.0005529	0.0010578
Proposed ($n = 20$)	0.0002418	0.0005670
Proposed ($n = 25$)	0.0002417	0.0005564

Method	norm. RMSE	norm. Max
Periodic spline	0.0002416	0.0005543

The periodic spline achieves very small errors on this smooth invariant curve. For moderate Fourier order ($n = 11$), the proposed method yields larger deviations; however, the approximation improves rapidly as the order increases. For $n = 20$ and $n = 25$, the normalised RMSE and maximum deviation are essentially identical to those obtained by the periodic spline. This behaviour reflects the smoothness of the underlying invariant curve, for which both spline interpolation and trigonometric approximation converge to the same geometric representation.

It is important to note that the periodic spline is an interpolatory method and therefore passes exactly through all given data points, whereas the proposed method performs a global least-squares approximation in the radial domain. Despite not interpolating the data, our model achieves comparable accuracy for sufficiently large order while producing a single global trigonometric representation of the curve.

4. Discussion and conclusion

This work introduced a simple and globally analytic method for fitting smooth closed curves to unordered planar point clouds under an approximate star-shaped assumption. The approach is based on three key components: automatic centre selection, polar transformation of the data, and least-squares trigonometric approximation of the radial function. Together, these steps yield a compact Fourier representation of the curve that is smooth, periodic, and computationally efficient to evaluate.

The centre-selection strategy plays a crucial role in the stability of the method. By minimising radial dispersion across angular bins through a multi-scale grid search, the procedure enforces an approximate star-shaped structure without requiring prior ordering or neighbourhood reconstruction. This makes the method applicable to unstructured datasets where connectivity information is unavailable.

Two representative examples were considered. For the first example consisting of a noisy and concave planar point cloud of 250 unordered samples, the method produced a stable global approximation resulting in a smoother and more controlled representation. The fitted curve has an average deviation of about 2.0% of the data diameter and a maximum deviation of about 6.0%, indicating a strong global approximation for this unordered and unstructured contour data set.

For a second example discussing the invariant curve arising from a Neimark–Sacker bifurcation, the proposed approach achieved very small normalised RMSE and maximum deviation values. As the Fourier order increased, the approximation rapidly converged and became essentially indistinguishable from periodic spline interpolation. This confirms the expected convergence behaviour for smooth closed curves.

Note that standard Neimark–Sacker curve approximation methods often rely on analytically known fixed points. However, such expressions are frequently unavailable or transcendental in practice. In contrast, our approach remains applicable even in these cases, as it does not require explicit analytical information.

The method has several advantages. It avoids explicit point ordering, does not require curvature estimation, and produces a single analytic expression for the entire curve. At the same time, it remains computationally lightweight compared to nonlinear optimisation-based spline fitting techniques.

The primary limitation of the approach is the approximate star-shaped requirement. If the dataset contains severe self-intersections or multiple radial crossings with respect to any interior point, the polar transformation becomes ambiguous and the radial model is no longer well defined. Thus, the method is most appropriate for geometries that admit a meaningful central reference point.

In summary, the proposed radial Fourier fitting framework provides a simple yet effective alternative for smooth closed-curve approximation in unordered planar point clouds. It balances geometric interpretability, analytic compactness, and computational simplicity, and performs competitively with classical spline methods for smooth data while remaining stable in moderately noisy settings.

References

[1] Pavlidis, T., “Curve fitting with conic splines”, IEEE Transactions on Pattern Analysis and Machine Intelligence 5 (1983) : 597–606.

[2] Plass, M., Stone, M., “Curve-fitting with piecewise parametric cubics”, Computer Graphics (Proceedings of SIGGRAPH) 17 (1983) : 229–239.

[3] Pratt, V., “Direct least-squares fitting of algebraic surfaces”, Computer Graphics (Proceedings of SIGGRAPH) 21 (1987) : 145–152.

[4] Wang, W., Pottmann, H., Liu, Y., “Fitting B-spline curves to point clouds by curvature-based squared distance minimization”, ACM Transactions on Graphics 25 (2006) : 214–238.

[5] Pérez-Arribas, F., Castañeda-Sabadell, I., “Automatic modelling of airfoil data points”, Aerospace Science and Technology 55 (2016) : 449–457.

[6] Zieniuk, E., Kapturczak, M., “Modeling the shape of boundary using NURBS curves directly in modified boundary integral equations for Laplace’s equation”, Computational and Applied Mathematics 37 (2018) : 4835–4855.

[7] Ebrahimi, A.R., Barid Loghmani, Gh., Sarfraz, M., “Capturing outlines of planar generic images by simultaneous curve fitting and sub-division”, Journal of AI and Data Mining 8 (2020) : 105–118.

[8] Jahanshahloo, A., Ebrahimi, A., “Reconstruction of the initial curve from a two-dimensional shape for the B-spline curve fitting”, The European Physical Journal Plus 137 (2022) : 411.

[9] Salomon, D., Curves and Surfaces for Computer Graphics, Springer, New York (2006).

[10] Kuznetsov, Y.A., Elements of Applied Bifurcation Theory, 3rd ed., Springer (2004).

[11] Guckenheimer, J., Holmes, P., Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields, Springer (1983).

[12] Wang, B., Zhao, Z., Chen, Y., Yu, J., "A novel robust point cloud fitting algorithm based on nonlinear Gauss–Helmert model", IEEE Transactions on Instrumentation and Measurement 72 (2023) : 1–12.

[13] Kapçak, S., Elaydi, S., Ufuktepe, Ü., "Stability of a predator–prey model with refuge effect", Journal of Difference Equations and Applications 22 (2016) : 989–1004.

Appendix A. Dataset Used in Section 3.1

The following list contains the 250 planar data points used in Section 3.1. The dataset is provided in (x, y) format and can be directly used in SageMath.

```
data = [(14.0833704403109, 45.8640216456390), (45.0776617115718, 23.4405471919957),
(50.7575650411329, -2.70669532291200), (25.8131738033750, -7.71173135261880),
(-7.30322199737740, 8.73657205625871), (-14.1610339304113, 36.0191054122393),
(0.928158933443999, 46.2608684338284), (33.3438458653406, 28.4075534508328),
(53.0611543497487, 5.79820603993560), (29.2639864509807, -0.128695870276498),
(2.59813583903540, 1.96797102298470), (-7.54060301511360, 26.9944926253356),
(-3.75641473937790, 44.5811666429496), (23.2870231231354, 32.7864967673419),
(45.2653569621621, 13.5917374690221), (33.2746402011878, 1.09671244628380),
(12.1088798438365, -1.53136610632940), (-6.52224036546240, 20.9623375175046),
(-7.32348479899850, 44.2671427911168), (18.3870693246318, 37.4179789827804),
(45.8685895112713, 19.9095630040816), (48.6734803267470, -4.70633335920610),
(19.7089335691344, -4.23862346682650), (-10.9316872059561, 15.9167906409223),
(-15.9216961304098, 41.5414278694992), (7.94633884181729, 48.3101558345725),
(41.2195894123120, 28.1134339284239), (52.7051591342664, 0.707507044367899),
(27.7857361362989, -7.79049713635140), (-2.39593836977420, 7.27881953178554),
(-11.5467306501614, 32.5350858609895), (-1.01995316182670, 44.9295378682023),
(28.5724481504959, 27.2475763508156), (48.6929016445665, 9.03789302550311),
(30.4517283408389, 2.20133170431630), (8.51681509470418, -0.0962846454843991),
(-7.28593192475440, 22.6099033090221), (-6.83704195731280, 44.1686333644370),
(22.6543767419599, 36.3197445962604), (44.1497172148445, 15.8383995998806),
(41.3135619324578, -3.01831665031630), (14.7864983014366, -4.06469241876000),
(-10.8859544782716, 20.4860330272300), (-13.4010980706476, 39.9100487090700),
(14.6822114789536, 46.8392909049767), (45.6195717619239, 20.4541968797278),
(53.9495581905271, -5.24963345438070), (25.4326886059074, -7.52626465039020),
(-8.55077462682670, 9.12977733498486), (-15.5457890547603, 36.1618611150312),
(3.02755746424840, 51.2512734996069), (33.8375343533432, 26.6629295394860),
(51.7071465787267, 3.15742563175650), (28.4971088733617, -0.660518505542001),
(2.80067296631200, 5.96431660733510), (-8.07545979827300, 25.5290954631139),
(-0.617288087062001, 45.8215415055389), (24.5690613964110, 32.7241664364482),
(46.8886802701620, 14.6568670617229), (32.3655084308339, 0.483147996061600),
(10.0547335047529, 2.91351188596070), (-7.50864555525240, 24.3682768585637),
(-7.74551420023360, 39.5769315060948), (19.4305813359526, 35.6816771782675),
(44.2279667897210, 17.1130448667592), (45.3656119452957, -1.55274439794940),
(18.0809570539812, -7.40190507307810), (-11.7296640495741, 15.5840677886778),
(-13.1320592699057, 41.8990017236045), (9.51457585324754, 50.6679002956663),
(40.4291080490749, 26.5340141310120), (53.3847323434043, 0.436055853945900),
(26.3132043732871, -6.19919139344280), (-4.37252709602570, 9.27227087530646),
(-12.5148258799190, 32.8311426549590), (-1.50423306098290, 45.4472663624204),
(27.4343267067376, 28.2786590076513), (51.5564666690773, 8.44427449518415),
(30.8254782629267, -0.632199891543301), (8.06941049367435, 2.49131082576180),
(-4.08209708892430, 24.9544496333512), (-3.34972860518070, 41.3457091668882),
(21.2850163690683, 32.5986242809797), (44.0657638710493, 12.5828396498529),
(38.8673924432823, -0.196337038345799), (13.7718304862097, -0.967603202570100),
(-8.95715050791950, 19.9176078856140), (-10.4875089909077, 43.3990650590751),
(14.4887252381538, 42.9402647673624), (44.2382984560540, 21.7603531707173),
(50.1328318246946, -5.37991741740290), (24.3625716224001, -9.02752355838560),
(-10.1858755390058, 10.7922228126339), (-15.9400992459121, 36.7830172421700),
(4.12657710073830, 49.2419229508879), (33.6147394283352, 25.7713415466891),
(51.7857995079678, 1.51599847800780), (29.2634532371511, -4.05024456297090),
(1.58367058773620, 2.41473720079940), (-8.53359194998340, 27.6899367904810),
(-0.606885150363700, 45.5318912881851), (23.9997162370306, 30.9856985452373),
(46.4610852479885, 11.8458986122376), (31.7162371572352, 1.79297491889860),
(10.6518721547623, -0.900575027554499), (-7.80252991665100, 23.3609572625232),
```

(-6.34255265853040, 42.2368899673079), (20.1467061691430, 38.3358558069239),
(44.4080511390261, 19.9079805042673), (46.9760336261575, -5.21371420344140),
(19.9023745785739, -7.70825702725700), (-10.8698481550002, 13.2075341300519),
(-15.3528993190762, 42.2915015614923), (11.2923594543718, 45.0645602411032),
(42.6885218214193, 28.2786377585394), (54.9052417505178, 0.394658159968500),
(29.1114536477759, -4.38936201820470), (-4.11809432373130, 6.92566004219137),
(-11.0980877434013, 32.1319295349256), (-1.45872714472700, 47.4905101449408),
(28.4569056328223, 31.8286838642246), (49.5896717408130, 5.38789497726150),
(32.2219053264603, -0.276657077190499), (6.85327275987264, 1.80931518848510),
(-5.22079897363990, 23.9384006231987), (-4.07640186262730, 44.0797756013414),
(23.3974154061994, 32.5414843919978), (47.5173284164052, 12.5747575461713),
(36.7787931874650, -1.39263660649160), (15.1021290976717, 0.343750567759001),
(-10.0766106906335, 19.5397406680656), (-10.1106683546437, 41.5501335571884),
(13.1947111604482, 41.9854264933864), (43.3144802804099, 24.3333796739636),
(49.2617516129128, -3.22976258474440), (24.4919929357867, -6.71082896309380),
(-9.65401140166930, 13.6297580113063), (-15.6271054124203, 34.2678153226602),
(4.30522644572670, 52.0953537206589), (36.0157678490473, 26.4622672189988),
(53.7484166174525, 4.54431617754470), (30.8683024538256, -4.46778172842310),
(0.676620713165100, 6.80644268111991), (-6.28236134321220, 30.8617139965626),
(-2.05791307807430, 44.9139791745892), (26.1721127385870, 31.7084836422031),
(46.0848424122659, 13.7710148302157), (31.1600069822185, 0.252215800467500),
(11.3677232615008, 0.814853237565700), (-6.00904495607740, 22.0807362894449),
(-9.27660227705060, 45.0720936700730), (20.9185605320326, 38.6669382469253),
(47.4137102270479, 19.2174061394871), (45.0136569217522, -2.66862539024350),
(17.3396600537932, -3.42428130400840), (-10.9399201339546, 19.0178922155203),
(-14.1218061425601, 41.0616870041813), (8.93667489314029, 48.4431549565578),
(40.4632354663040, 23.7820031655636), (55.6432208342846, 0.0301050979992006),
(29.0433756763193, -3.56117641333340), (-6.47411705507580, 8.17614504316638),
(-12.7430424269200, 32.2287956195986), (1.50490144110250, 49.0500215663043),
(28.5047612346179, 28.0325589628336), (48.8571186126539, 7.55965847447667),
(29.6904720713539, 1.67958134794050), (2.99325144376810, 5.52266257264040),
(-3.99935966975340, 26.5825141042857), (-5.16528316426960, 44.7558094695407),
(21.1768567743907, 30.1013477968190), (45.7728153077872, 15.9104229437563),
(37.6572865379877, 0.664913344824900), (15.0584172889311, -3.54968820117080),
(-9.75906600511870, 22.1306444535884), (-11.8205676775853, 41.5471276759429),
(15.2751952474775, 43.4885662587228), (45.6242642204564, 19.4561416638932),
(49.4373767424919, -5.17333039762480), (23.2893695033401, -3.95627694319490),
(-10.6827052314930, 12.2037174449048), (-13.1201261310191, 34.6875010401614),
(5.29818596380549, 46.3474282957298), (35.5021825280464, 28.4466995049707),
(54.0331921835049, 0.973545058947099), (29.3906496182609, -2.59741312596710),
(-1.21551137611840, 2.18616294664520), (-7.06600941383080, 29.0798975806310),
(-1.12883246189030, 43.4083939388456), (27.0748668082488, 32.3350321994232),
(49.5526786324633, 11.2532736965816), (32.3425221999083, 2.18127838495760),
(10.1722751550025, 1.00714116723340), (-6.33455770666250, 24.7204572372829),
(-7.57554897896390, 40.3264059593605), (21.8205482278409, 36.4088828107602),
(44.1231346230599, 19.3053979620111), (44.9540550564105, -0.171592800480799),
(18.6617390114253, -5.61408261469120), (-11.5289026631934, 19.1764714120011),
(-16.1754106616359, 42.3724636619283), (11.9070868543127, 46.1375814455028),
(42.0596755828299, 27.3741136950360), (55.9386243865406, -2.49085191945810),
(25.5954492906979, -8.04723100390640), (-5.76256212601030, 9.18151106950166),
(-12.2991227526875, 32.3278618298199), (-0.147434997431100, 45.7227212467389),
(29.8667741150848, 28.6164811541611), (52.8469528151210, 6.93779639725695),
(29.5070381373391, -2.22538711853190), (2.37920806608640, 5.11768718723710),
(-7.38553090103970, 27.6783416090993), (-5.63536234227960, 44.4126105857022),
(21.3030033988914, 29.5239871456256), (45.7579604567479, 11.1043849677154),
(36.3005729970801, 0.716205771571801), (14.0658603845289, 0.650499565394000),
(-6.81179599867030, 20.2011848940733), (-9.63494908463720, 44.0704684567996),
(14.6778147406047, 41.2516034351028), (47.0444448639804, 22.8703424245831),
(51.3624734698702, -3.96265787228030), (22.1281897344613, -6.59860406946240),
(-9.44156853754080, 14.3075356415018), (-15.3857159072104, 40.1805920811891),

(6.57187256901424, 49.9811237264161), (35.5205912568897, 30.3734522823830),
(53.4840135270912, 2.97207466966340), (28.7465119340308, -5.79732876663320),
(-1.63495708857240, 3.31995482585080), (-8.81924611013510, 26.6548964842217),
(-0.615218616584601, 44.4926151560740), (25.6261197065328, 27.8268632056830),
(49.1274361832382, 7.57026020056624), (31.4186807077310, 0.564112370347100),
(10.4731309891711, 4.28441977149410), (-6.11336423372770, 25.6902953197217),
(-6.33557454448200, 41.8337903993249), (19.7322218079573, 34.2932231615998),
(47.5459923164546, 16.5295576543571), (42.7812233758175, -0.539636372398100),
(16.9829876874806, -2.43409827113830), (-9.63140153262630, 18.7294706307919),
(-12.5126490201333, 39.3700884342812), (10.7665816357537, 48.2511651328134),
(42.5068343094912, 25.1640991121605), (54.9516838624218, -2.70808222434180),
(25.9306206873889, -6.33367410474300), (-8.94467406475380, 9.70728116648584)]

Appendix B. SageMath Implementation

The following SageMath script implements the centre-selection procedure and the radial trigonometric least-squares fitting described in Sections 2 and 3.

```
# Ver. centre-aware step-by-step fitter
import math
var('x')

# ----------------------------
# Centre selection code
# ----------------------------
def _median(lst):
    s = sorted(lst)
    n = len(s)
    if n == 0:
        raise ValueError("Empty list")
    if n % 2 == 1:
        return s[n//2]
    return 0.5*(s[n//2 - 1] + s[n//2])

def _iqr(values):
    v = sorted(values)
    n = len(v)
    if n < 4:
        return (v[-1] - v[0]) if n >= 2 else 0.0
    mid = n // 2
    lower = v[:mid]
    upper = v[mid:] if (n % 2 == 0) else v[mid+1:]
    q1 = _median(lower)
    q3 = _median(upper)
    return float(q3 - q1)

def _polar_pairs(points, c):
    cx, cy = c
    t = []
    r = []
    for (x0, y0) in points:
        dx = x0 - cx
        dy = y0 - cy
        t.append(math.atan2(dy, dx))
```

```
        r.append(math.hypot(dx, dy))
    return t, r

def ambiguity_score(points, c, B=180, min_bin_count=2,
empty_bin_penalty=0.0):
    t, r = _polar_pairs(points, c)
    bins = [[] for _ in range(B)]
    two_pi = 2.0 * math.pi

    for ang, rad in zip(t, r):
        u = ang + math.pi
        j = int((u / two_pi) * B)
        if j == B:
            j = B - 1
        bins[j].append(rad)

    score = 0.0
    empty = 0
    for bj in bins:
        if len(bj) >= min_bin_count:
            score += _iqr(bj)
        elif len(bj) == 0:
            empty += 1

    if empty_bin_penalty > 0.0:
        score += empty_bin_penalty * empty
    return float(score), empty

def robust_initial_center(points):
    xs = [p[0] for p in points]
    ys = [p[1] for p in points]
    return (float(_median(xs)), float(_median(ys)))

def select_center(points,
                  B=180,
                  rounds=4,
                  grid_N=21,
                  init_box_scale=0.35,
                  min_bin_count=2,
                  empty_bin_penalty=0.0,
                  shrink=0.35):
    if len(points) < 5:
        raise ValueError("Need at least 5 points for stable centre
selection.")

    xs = [p[0] for p in points]
    ys = [p[1] for p in points]
    dx = max(xs) - min(xs)
    dy = max(ys) - min(ys)
    diameter = math.hypot(dx, dy)

    if diameter == 0:
        c0 = robust_initial_center(points)
```

```
        s0, e0 = ambiguity_score(points, c0, B=B,
                                 min_bin_count=min_bin_count,
                                 empty_bin_penalty=empty_bin_penalty)
        return c0, s0, e0

    best_c = robust_initial_center(points)
    best_s, best_empty = ambiguity_score(points, best_c, B=B,
                                         min_bin_count=min_bin_count,

empty_bin_penalty=empty_bin_penalty)

    if grid_N % 2 == 0:
        grid_N += 1

    half = init_box_scale * diameter

    for _ in range(rounds):
        cx, cy = best_c
        step = (2.0 * half) / (grid_N - 1)

        local_best_c = best_c
        local_best_s = best_s
        local_best_empty = best_empty

        for i in range(grid_N):
            xx = cx - half + i * step
            for j in range(grid_N):
                yy = cy - half + j * step
                s, empty = ambiguity_score(points, (xx, yy), B=B,
                                           min_bin_count=min_bin_count,

empty_bin_penalty=empty_bin_penalty)
                if s < local_best_s:
                    local_best_s = s
                    local_best_c = (xx, yy)
                    local_best_empty = empty

        best_c, best_s, best_empty = local_best_c, local_best_s,
local_best_empty
        half *= shrink

    return best_c, best_s, best_empty


# ----------------------------
# Error metrics (point -> sampled curve)
# ----------------------------
def sample_curve_xy(r_func, c, N=3000):
    cx, cy = c
    pts = []
    for i in range(N):
        t = -math.pi + 2.0*math.pi*i/(N-1)
        rr = float(r_func(t))
```

```
        pts.append((cx + rr*math.cos(t), cy + rr*math.sin(t)))
    return pts

def point_to_curve_distance(p, curve_pts):
    x0, y0 = float(p[0]), float(p[1])
    return min(math.hypot(x-x0, y-y0) for (x,y) in curve_pts)

def rmse_and_max(points, curve_pts):
    ds = [point_to_curve_distance(p, curve_pts) for p in points]
    rmse = math.sqrt(sum(d*d for d in ds)/len(ds))
    mx = max(ds)
    return rmse, mx


# ----------------------------
# Coefficient table helpers (ASCII / LaTeX)
# ----------------------------
def fourier_coeff_rows(r_expr, N):
    """
    Returns rows [(k, cos_coeff, sin_coeff)] for k=0..N
    with the correct constant term at k=0.
    """
    trig_part = sum(
        r_expr.coefficient(cos(k*x))*cos(k*x) +
r_expr.coefficient(sin(k*x))*sin(k*x)
        for k in range(1, N+1)
    )
    c0 = (r_expr - trig_part).simplify_full()

    rows = [(0, float(c0), 0.0)]
    for k in range(1, N+1):
        ck = float(r_expr.coefficient(cos(k*x)))
        sk = float(r_expr.coefficient(sin(k*x)))
        rows.append((k, ck, sk))
    return rows

def print_ascii_coeff_table(rows, digits=15):
    print(f"{'k':>2} | {'Coeff. cos(kx)':>20} | {'Coeff. sin(kx)':>20}")
    print("-"*50)
    for k, ck, sk in rows:
        print(f"{k:>2} | {ck:>20.{digits}f} | {sk:>20.{digits}f}")

def latex_coeff_table(rows, digits=15, booktabs=True, caption="Fourier
coefficients"):
    fmt = f"{{:.{digits}f}}"
    lines = []
    if booktabs:
        lines += [
            r"\begin{table}[h]",
            r"\centering",
            r"\begin{tabular}{r r r}",
            r"\toprule",
            r"$k$ & Coeff.\ $\cos(kx)$ & Coeff.\ $\sin(kx)$ \\",
```

```
            r"\midrule",
        ]
    else:
        lines += [
            r"\begin{table}[h]",
            r"\centering",
            r"\begin{tabular}{r r r}",
            r"\hline",
            r"$k$ & Coeff.\ $\cos(kx)$ & Coeff.\ $\sin(kx)$ \\",
            r"\hline",
        ]

    for k, ck, sk in rows:
        lines.append(f"{k} & {fmt.format(ck)} & {fmt.format(sk)} \\\\")
    lines += [r"\bottomrule" if booktabs else r"\hline", r"\end{tabular}"]
    if caption:
        lines.append(rf"\caption{{{caption}}}")
    lines.append(r"\end{table}")
    return "\n".join(lines)


# ----------------------------
# Interactive fitter (step-by-step)
# ----------------------------
@interact
def _(
    data = input_box(label="Data Points", default=data, width=40),
    intervalx = input_box(label="Interval $x$", default=(-20, 65),
width=40),
    intervaly = input_box(label="Interval $y$", default=(-20, 65),
width=40),
    ord_fourier = slider(1, 20, 1, default=9, label="Order"),
    nok = input_box(label="Number of Points (use last)", default=350,
width=40),

    centre_mode = selector(["manual", "auto"], default="auto",
label="Centre mode"),
    centre_points = selector(["last nok", "all"], default="last nok",
label="Auto uses"),

    move_x = slider(-5, 5, 0.01, default=0, label="Manual centre x"),
    move_y = slider(-5, 5, 0.01, default=-1.5, label="Manual centre y"),

    B = slider(60, 720, 30, default=180, label='Auto: angle bins B'),
    rounds = slider(1, 6, 1, default=4, label='Auto: refinement rounds'),
    grid_N = slider(9, 41, 2, default=21, label='Auto: grid_N'),
    init_box_scale = slider(0.05, 1.00, 0.05, default=0.35, label='Auto:
init scale'),
    shrink = slider(0.20, 0.70, 0.05, default=0.35, label='Auto: shrink'),
    min_bin_count = selector([1,2,3,4], default=2, label='Auto: min per
bin'),
    empty_bin_penalty = slider(0.0, 0.05, 0.001, default=0.0, label='Auto:
empty penalty'),
```

```
    curve_samples = slider(500, 6000, 500, default=3000, label="Curve
samples (error)"),
    point_size = slider(10, 80, 5, default=10, label='Point size'),

    show_tr = checkbox(True, label="Show t-r plot"),
    show_fit_tr = checkbox(True, label="Show fit in t-r plot"),
    show_axes = checkbox(True, label="Show Axes"),
    figsize = slider(4, 12, 1, default=6, label="Figure Size"),

    # --- NEW OPTIONS ---
    show_centre = checkbox(True, label="Show centre marker"),
    print_r_expr = checkbox(False, label="Print r(t) explicitly"),
    show_coeff_table = checkbox(False, label="Show coefficient table"),
    latex_output = checkbox(False, label="LaTeX output (table)"),
    use_booktabs = checkbox(True, label="LaTeX: booktabs"),
    table_digits = slider(0, 18, 1, default=15, label="Table digits"),
):

    pts = list(data)
    if len(pts) == 0:
        print("Empty dataset.")
        return

    nok = int(nok)
    FromTo = pts[-nok:] if nok <= len(pts) else pts

    # choose which points define the centre
    centre_set = FromTo if centre_points == "last nok" else pts

    if centre_mode == "manual":
        c = (float(move_x), float(move_y))
        amb_score = None
        empty_bins = None
    else:
        c, amb_score, empty_bins = select_center(
            centre_set,
            B=int(B), rounds=int(rounds), grid_N=int(grid_N),
            init_box_scale=float(init_box_scale),
            min_bin_count=int(min_bin_count),
            empty_bin_penalty=float(empty_bin_penalty),
            shrink=float(shrink)
        )

    # Plot points (+ optional centre)
    P = points(FromTo, size=point_size, color="black")
    C = point(c, size=140, marker='X')
    base_plot = P + (C if show_centre else Graphics())
    show(base_plot, aspect_ratio=1,
         axes=bool(show_axes), figsize=[int(figsize), int(figsize)],
         xmin=intervalx[0], xmax=intervalx[1], ymin=intervaly[0],
ymax=intervaly[1],
         axes_labels=['$x$','$y$'])
```

```
    if centre_mode == "auto":
        print("Chosen centre c =", c)
        print("Ambiguity score S(c) =", amb_score)
        print("Empty bins =", empty_bins, "out of", int(B))
    else:
        print("Manual centre c =", c)

    cx, cy = c

    # Polar transform around c
    polar_pts = [(atan2(y-cy, x-cx), sqrt((x-cx)^2 + (y-cy)^2)) for (x, y)
in FromTo]

    if show_tr:
        show(points(polar_pts, size=point_size, color="black"),
axes=bool(show_axes),
             figsize=[int(figsize), int(figsize)],
             axes_labels=['$t$','$r$'])

    # Build Fourier model for r(t)
    RRk1 = PolynomialRing(QQ, 'k1', 25)
    RRk2 = PolynomialRing(QQ, 'k2', 25)
    k1 = RRk1.gens()
    k2 = RRk2.gens()
    model(x) = sum(k1[i]*sin(i*x) + k2[i]*cos(i*x) for i in
range(int(ord_fourier)))

    sol = find_fit(polar_pts, model)
    r_expr = model.subs(sol)
    r(x) = r_expr

    # --- NEW: print r(t) explicitly ---
    if print_r_expr:
        print("\nr(t) =")
        show(r_expr)   # pretty Sage output
        # If you prefer raw text instead, replace with: print(r_expr)

    if show_tr and show_fit_tr:
        fit_graf = plot(r_expr, -pi, pi)
        show(points(polar_pts, size=point_size, color="black") + fit_graf,
             axes=bool(show_axes), figsize=[int(figsize), int(figsize)],
             axes_labels=['$t$','$r$'])

    # Plot fitted curve in xy
    PP = parametric_plot((cx + r(x)*cos(x), cy + r(x)*sin(x)), (x, -pi,
pi))
    show(base_plot + PP, aspect_ratio=1,
         axes=bool(show_axes), figsize=[int(figsize), int(figsize)],
         xmin=intervalx[0], xmax=intervalx[1], ymin=intervaly[0],
ymax=intervaly[1],
         axes_labels=['$x$','$y$'])
```

```
    # --- NEW: coefficient table (ASCII or LaTeX) ---
    if show_coeff_table:
        Ntab = int(ord_fourier) - 1   # because your model sums
i=0..ord_fourier-1
        rows = fourier_coeff_rows(r_expr, Ntab)

        print("\nFourier coefficient table (k=0..%d):" % Ntab)
        if latex_output:
            print(latex_coeff_table(
                rows,
                digits=int(table_digits),
                booktabs=bool(use_booktabs),
                caption="Fourier coefficients"
            ))
        else:
            print_ascii_coeff_table(rows, digits=int(table_digits))

    # Compute errors in xy
    curve_pts = sample_curve_xy(r, c, N=int(curve_samples))
    rmse, mx = rmse_and_max(FromTo, curve_pts)

    # Normalise by bbox diameter (nice for paper tables)
    xs = [float(p[0]) for p in FromTo]
    ys = [float(p[1]) for p in FromTo]
    D = math.hypot(max(xs)-min(xs), max(ys)-min(ys))

    if D > 0:
        print("RMSE =", rmse, " | normalised RMSE =", rmse/D)
        print("Max error =", mx, " | normalised Max =", mx/D)
    else:
        print("RMSE =", rmse)
        print("Max error =", mx)
```