UDC 519.85, 004.92

O. V. FROLOV
Simon Kuznets Kharkiv National University of Economics

CIRCLE-INTERSECTION-BASED EQUAL-CHORD PARTITIONING ALGORITHM FOR A PLANAR PARAMETRIC CURVE

This paper addresses the problem of equal-chord partitioning of planar parametric curves, which consists of determining a sequence of points on a curve such that the Euclidean distances (chords) between consecutive points are equal. Equal-chord discretization is important in computer-aided design, CNC machining, robotics, and computer graphics, as it provides stable geometric and technological properties compared to parameter-uniform or arc-length-based sampling.

This work aims to develop efficient algorithms for equal-chord partitioning of planar parametric curves in the classical setting, i.e., with fixed endpoints and a prescribed number of segments, while explicitly accounting for the multivalued nature of solutions arising from curve–circle intersections.

The proposed approach is based on successive intersections of the curve with a moving circle of fixed radius. Starting from one or both endpoints, the circle center is placed at the current partition point, and subsequent points are obtained as intersections between the curve and the circle in a specified direction. To handle non-convex curves and curves with inflection points, all possible intersection solutions are organized into a tree structure, where each branch represents an alternative partitioning trajectory. The optimal partition is selected by minimizing the difference between the residual segment length and the circle radius.

Two strategies for determining the partitioning radius are investigated. The first relies on uniform redistribution of the residual chord-length error among all segments and is effective for convex curves with unique intersection solutions. The second formulates radius selection as a one-dimensional optimization problem and ensures robust convergence for curves of complex geometry.

An experimental evaluation of planar Bézier curves of various orders demonstrates stable convergence of the proposed algorithms across a wide range of segment counts. As the number of segments increases, the multiplicity of curve–circle intersections decreases, and the computational complexity approaches linear behavior in practical scenarios. The proposed method avoids numerical arc-length integration and auxiliary grid-based approximations and can be naturally extended to spatial curves.

Keywords: *pseudocode, iteration, computational complexity, partitioning, chord, parametric planar curve, segment, intersection equation.*

О. В. ФРОЛОВ
Харківський національний економічний університет імені Семена Кузнеця

АЛГОРИТМ РІВНОХОРДОВОГО РОЗБИТТЯ ПЛОСКОЇ ПАРАМЕТРИЧНОЇ КРИВОЇ НА ОСНОВІ ПЕРЕТИНУ З КОЛОМ

У статті розглядається задача рівнохордового розбиття плоских параметричних кривих, яка полягає у визначенні послідовності точок на кривій так, щоб евклідові відстані (хорди) між сусідніми точками були рівними. Рівнохордова дискретизація є важливою для систем автоматизованого проєктування, числового програмного керування, робототехніки та комп'ютерної графіки, оскільки забезпечує стабільні геометричні й технологічні властивості траєкторій порівняно з параметрично рівномірним або дуговим дискретизуванням.

Метою роботи є розроблення ефективних алгоритмів рівнохордового розбиття плоских параметричних кривих у класичній постановці – за фіксованими кінцевими точками та заданою кількістю сегментів – з урахуванням багатозначності розв'язків, що виникає при перетині кривої з колом.

Запропонований підхід ґрунтується на послідовному перетині кривої з рухомим колом фіксованого радіуса. Починаючи з одного або обох кінців кривої, центр кола розміщується в поточній точці розбиття, а наступні точки визначаються як точки перетину кривої з колом у заданому напрямку. Для обробки не опуклих кривих і кривих з точками перегину всі можливі розв'язки рівняння перетину організовуються у вигляді деревоподібної структури, де кожна гілка відповідає альтернативній траєкторії розбиття. Оптимальний варіант визначається шляхом мінімізації різниці між довжиною залишкового сегмента та радіусом кола.

У роботі досліджено два підходи до визначення радіуса кола: метод рівномірного перерозподілу похибки між сегментами та оптимізаційний підхід, у якому радіус є мінімумом одновимірної цільової функції. Експериментальні дослідження на плоских кривих Безьє різного порядку підтверджують стабільну збіжність алгоритмів і майже лінійну обчислювальну складність у практичних сценаріях. Запропонований метод не потребує чисельного

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

інтегрування довжини дуги, легко узагальнюється на просторові криві та є перспективним для подальших досліджень.

Ключові слова: *псевдокод, ітерація, обчислювальна складність, розбиття, хорда, параметрична плоска крива, відрізок, рівняння перетину.*

Problem Statement

The discretization of continuous geometric objects constitutes a fundamental problem in computational geometry and is relevant to many interdisciplinary applications. In particular, numerous fields – including computer vision, robotics, signal processing, curve simplification in computer graphics, geographic information systems, and digital manufacturing – rely on the discretization and segmentation of planar curves, which serve as fundamental geometric primitives. These methods aim to partition a curve into homogeneous segments with identical characteristics or to minimize a predefined error.

The requirement of equal-chord partitioning, where a curve is divided into segments of identical chord length, adds significant value in practical contexts. It simplifies CNC curve reproduction by maintaining a uniform tool feed rate [1] and supports trajectory recovery from video sequences [2].

Therefore, studying equal-chord partitioning methods for curves attracts significant attention due to their potential implementation in computer-aided design and digital manufacturing systems for complex-shaped objects, geographic information systems, computer vision, and robotics.

Analysis of recent studies and publications

Numerous studies have been devoted to curve partitioning into segments, covering various methods, algorithms, and criteria depending on the specific problems being addressed. Uniform arc-length sampling of parametric curves is a common task in practical applications. For polynomial curves, such discretization requires numerical integration [3–5], complicating implementation on specific hardware. A simplified method based on initial random sampling of curve points was proposed in [6].

The sampling of curves, in particular NURBS, based on reparameterization by curvature and on mixed parameterization by arc length and curvature with equal weights was studied in [7]. A similar approach was presented in [8], where a combined criterion based on arc length and bending energy, quadratic of curvature, was used. These methods require numerical integration and the solution of systems of nonlinear equations.

An adaptive sampling method for free-form aero-engine blade surfaces based on discrete curvature was proposed in [9]. By iteratively refining a triangular mesh and inserting points according to local curvature variation, the method achieves high reconstruction accuracy with fewer samples than uniform approaches.

An efficient interpolation strategy for parametric curve machining addressing chord-error constraints was developed in [4]. The method ensures high accuracy and real-time performance by combining curvature-based key point selection, adaptive arc-length computation, and a modified Runge–Kutta scheme.

Optimal partitioning of planar curve segments using a cost functional combining arc length and local geometric properties was examined in [10].

The problem of equal-chord partitioning of curves has been addressed in several studies [11–17]. The existence of equal-chord partitioning was theoretically proven in [11]. A more extensive contribution was presented in [12], where, in addition to theoretical results, an algorithm based on a piecewise-linear approximation of the distance function between curve points was proposed. That work also analyzed chord-length inequality errors, computational complexity, and experimental results.

An extension of equipartition methods to two-dimensional domains was proposed in [13], in which complex planar shapes are divided into equal-area regions while minimizing the internal boundary length.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

An interpolation method for CNC deposition based on equal-chord length determination was proposed in [14]. The algorithm employs sequential determination of the curve parameter using fixed-step partitioning combined with binary search.

The existence of inscribed polygons with prescribed edge ratios for Jordan curves was established in [15], which guarantees the existence of equal-chord partitioning as a special case.

The problem of transforming a discrete polygonal curve with non-uniform segment lengths into an equal-chord representation was studied in [16], where convergence of iterative vertex respacing to an equilateral partition was proven.

Algorithms for equal-chord partitioning of planar parametric curves based on intersections with a moving circle were proposed in [17; 18], assuming a unique intersection point. In this approach, the circle center is sequentially moved to the obtained partition pointx.

The implementation of equal-chord partitioning in computer-aided design systems can be exemplified by Rhinoceros 3D, which provides a curve partitioning functionality via the Divide command [19].

Purpose

The object of research is the process of discretization of planar parametric curves by partitioning them into segments with equal chord length.

Subject of research – algorithms and methods for equal-chord partitioning of planar parametric curves, in particular, an approach based on successive intersections of a curve with a circle of fixed radius, taking into account the multivalued nature of intersection solutions.

Purpose of the study – to develop an efficient algorithm for equal-chord partitioning of planar parametric curves in the classical formulation (with fixed endpoints and a prescribed number of segments), ensuring robustness in the presence of multiple curve–circle intersection solutions.

The novelty of this research lies in applying partitioning approaches that account for multivalued solutions arising from the intersection of the curve and a moving circle.

Presentation of the main research material

The problem of partitioning a curve, given in parametric vector form on the Euclidean plane, into equal chord-length segments has been considered in the “canonical” formulation [11; 12], which can be stated as follows. Let the equation of the curve be given as

$$\mathbf{p} = \mathbf{p}(t). \tag{1}$$

It is required, on the given interval of the curve parameter $t \in [t_0, t_n]$, to determine the intermediate values $t_0 < t_1 < t_2 < \ldots < t_{n-1} < t_n$ such that, when substituted into equation (1), we obtain

$$P_i = \mathbf{p}(t_i), (i = 0,1, \ldots, n). \tag{2}$$

A sequence of points $P\ \{P_0, P_1, P_2, \ldots, P_{n-1}, P_n\}$, for which the condition of equality of the polyline segments is satisfied:

$$d_1 = d_2 = \ldots = d_n, \tag{3}$$

where denotes the Euclidean distance from point P_0 to point P_1, and so on.

Thus, the planar curve on the interval $[t_0, t_n]$ is partitioned into n segments with equal chord lengths. The existence of such an equal-chord partition has been proved in previous research [11; 12; 18].

The partitioning of a curve into segments of equal chord length can be considered a nonlinear optimization problem over the variables that define the positions of the internal partition points on the curve – t_i, $i = 1, \ldots, n - 1$. The problem formulation will be treated as the minimization of the error function representing the inequality of adjacent chord lengths

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

$$\sum_{i=1}^{n-1} e_i \to \min_t. \tag{4}$$

The inequality error for a consecutive pair of chords can be expressed as

$$e_i = \left|d_i - d_{i+1}\right| = \left|\left\|p(t_i) - p(t_{i-1})\right\| - \left\|p(t_i) - p(t_{i+1})\right\|\right|, i = 1,\ldots, n-1. \tag{5}$$

The constraints imposed on the functions and variables can be written as follows:
– non-coincidence of the partition points

$$t_i - t_{i-1} - \chi \geq 0,\ i = 1,\ \ldots,\ n,\ \chi - \text{small conitant}; \tag{6}$$

– the accumulation of errors must not exceed the allowable value ε

$$\frac{\varepsilon}{n-1} - e_i \geq 0,\ \ i = 1,\ldots, n-1, \tag{7}$$

which can also be written as a constraint on the objective function

$$\varepsilon - \sum_{i=1}^{n-1} e_i \geq 0. \tag{8}$$

Let us consider an approach based on the simple idea of partitioning a planar curve by moving an imaginary circle of fixed radius along it. The circle sequentially intersects the curve in the direction of its imagined motion, with the center being relocated to each obtained intersection point. If the initial center of the circle is placed at one of the curve's endpoints, then dividing the curve into n segments is sufficient to perform $n - 1$ such intersections.

Given two endpoints of the curve segment, the general case may be represented by two circles moving towards each other from both ends (t_0 and t_n). The number of steps taken from each endpoint can be arbitrarily adjusted, including the degenerate case of unidirectional motion (i.e., zero steps from one end). If the number of partitioning steps in the direction from t_0 to t_n is denoted by n_1, then in the opposite direction, it is necessary to perform $n - n_1 - 1$ intersections.

As a result, the circle-based equipartition produces $n - 1$ equal arclength segments and one residual segment, whose length, in the general case, differs from the others. The corresponding parameter interval of the curve defines this residual segment – $[t_{n_1}, t_{n_1+1}]$. After completing the equipartition through circle motion, one must decide on the subsequent procedure by comparing the length of the residual segment with the current circle radius.

For the algorithm to operate, an initial radius value for the circle is required. This radius should ensure that the curve is segmented into n parts, with the partition points approximately at the midpoints of the prescribed parameter intervals. At the same time, the interval of the residual segment must not overlap with the intervals of the other segments. In [17, 18], a method was proposed for this purpose, relying on the statistics of initial uniform parameter-based partitioning of the curve into n parts.

The main challenge in circle-based partitioning of a planar curve is the possible presence of multiple intersection points [18]. This situation occurs for curves with inflection points, depending on the number of partition segments n, when the circle radius becomes comparable to the curvature radius of the curve at specific arc locations.

Figure 1 clearly illustrates the dependence of the number of intersection points on the radius of the partitioning circle. Relatively small variations in the circle radius may lead to one intersection (circle with point $P_{i+1}^{(1)}$), two intersections (circle with points $P_{i+1}^{(2,1)}$, $P_{i+1}^{(2,2)}$), three intersections (circle with points $P_{i+1}^{(3,1)}$, $P_{i+1}^{(3,2)}$, $P_{i+1}^{(3,3)}$), or even more intersection points with the curve.

Figure 2 demonstrates how different intersection points may be obtained depending on the direction of the imagined motion, even though the same common chord of the curve is used. Specifically, a circle centered at point P_i yields, in the forward direction (Fig. 2, *a*), three intersection

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

points $P_{i+1}^{(1)}$, $P_{i+1}^{(2)}$, and $P_{i+1}^{(3)}$. If the circle center is then shifted to point $P_j = P_{i+1}^{(3)}$, the same-radius circle, when intersected with the curve in the opposite direction (Fig. 2, *b*), produces, besides the point $P_{j-1}^{(3)}$ coinciding with P_i, two additional points $P_{j-1}^{(1)}$ and $P_{j-1}^{(2)}$, which differ from the previously obtained points $P_{i+1}^{(1)}$ and $P_{i+1}^{(2)}$.

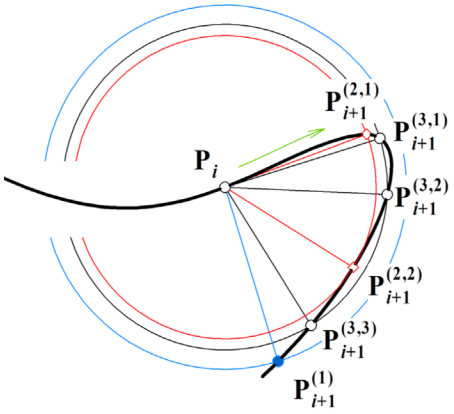

Fig. 1. Relationship between the number of curve–circle intersection points and the radius of the partitioning circle

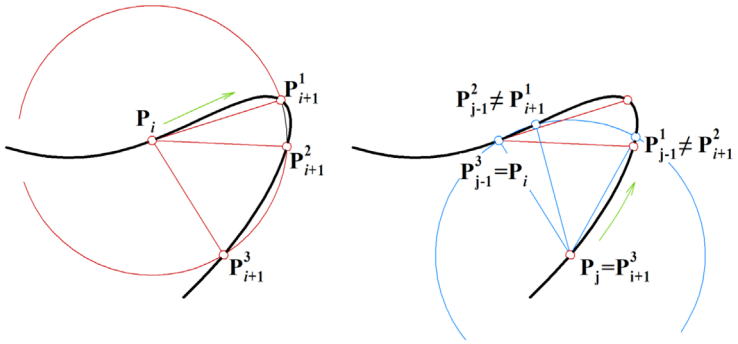

Fig. 2. Dependence of curve–circle intersection positions on motion direction

Thus, the occurrence of multiple intersection points necessitates an analysis to determine which points should be included in the final sequence of partitioning points. To this end, each candidate intersection point is incorporated into a distinct sequence of partitioning points. The residual segment length is evaluated and compared with the circle radius for every such sequence. The optimal solution is then identified as the sequence that minimizes the deviation of the residual segment.

If we represent the set of point sequences obtained by circle-based partitioning in a given direction as a tree, its root will be located at the endpoint of the curve from which the partitioning begins. Accordingly, if the curve–circle intersection produces multiple partition points, several branches will emerge from the node (partition point) corresponding to the current circle center. Each branch will generate a new sequence that includes the path of nodes from the parent to the root, together with the current intersection point.

The tree's height equals the number of partitioning steps from the given endpoint, while its width corresponds to the number of sequences, each representing one possible partitioning configuration. Thus, the tree's width depends on the number of multiple intersections encountered by the moving circle within the specified number of steps and the multiplicity of these intersections. Since there are two possible directions of curve partitioning by a circle, in the general case, two trees of partition sequences will be formed, one for each endpoint of the curve.

Figure 3 shows how multiple intersection points lead to alternative partition paths represented as tree branches.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

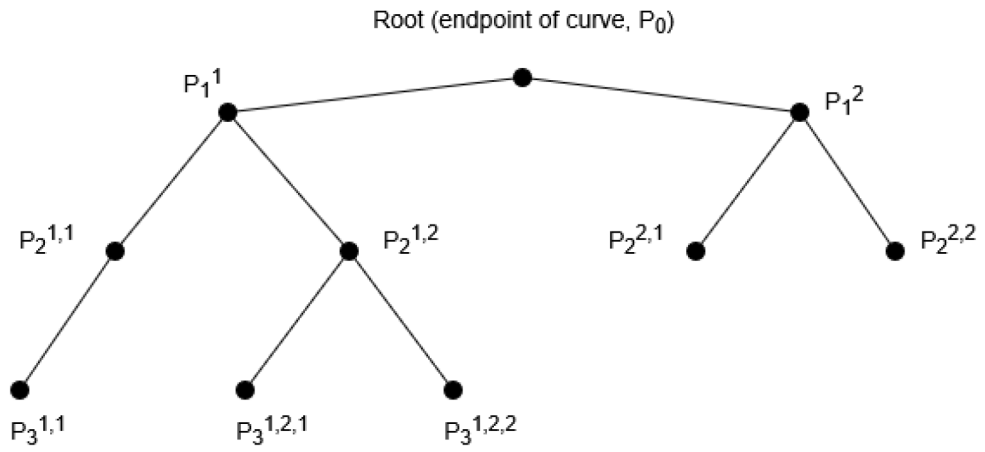

Fig. 3. Example tree showing curve partition sequences generated by successive circle–curve intersections

Explanation of fig. 3:

– Root (P_0) – the endpoint of the curve from which the partitioning process begins.

– P_i^1, P_i^2, P_i^3 – intersection points obtained at the i-th partitioning step; the superscript (¹, ², ³) denotes the ordinal number among several possible points.

Each branch of the tree represents a distinct partition sequence, consisting of the path from the root (P_0) through subsequent nodes (intersection points) to a terminal node (leaf of the tree). Each tree node is defined as a data structure consisting of four elements: the curve parameter value corresponding to the intersection with the circle (splitting point), the coordinates of the associated point on the curve, a reference to the parent node, and an array of references to the child nodes. The entire tree can thus be defined as an array of nodes, accessible via their indices.

The algorithm presented in Algorithm 1 is proposed to implement the partitioning procedure. This algorithm employs the following procedures: FIND-ROOTS – for the numerical solution of the equation describing circle–curve intersections, accounting for possible multiple solutions; ADD-CHILD – for linking child nodes to a parent node; as well as auxiliary routines such as LENGTH (to compute the array length), CLEAR (to reset the array), and PUSH (to append an element to its end).

The FIND-ROOT procedure implements the computation of the intersection point between the circle and the curve. Its implementation depends on the specific equation of the curve and the method used to obtain the roots of the intersection equation. The intersection equation in vector form is expressed as follows:

$$||p(t) - p(t_{ci})|| = r. \tag{9}$$

Here, $\mathbf{p}(t_{ci})$ denotes the current position of the circle center, and r is the value of the partition radius. Using this equation, the FIND-ROOT procedure must determine the root between the values t_{ci} and t_n for the forward pass (or between t_0 and t_{ci} for the backward pass). The obtained parameter value is then used to compute the coordinates of the intersection point using the curve's parametric equations. In the next iteration of the loop, the circle's center is moved to this intersection point.

Since the intersection of a circle and a line can be computed in constant time, the computational complexity of this algorithm depends on the number of partition points n and the number of partition variants associated with multiple intersection points. Let m denote the maximum multiplicity among all partition points generated by the circle–curve intersections in the given direction of motion. In the worst case, the full m-ary partition tree is considered.

The construction of such a tree requires time, which is defined as the sum of n- 1 terms of a geometric progression $O\left(\frac{m(1-m^{n-1})}{1-m}\right)$. In the best case, there is only one partition variant, resulting in complexity.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

Algorithm 1 Planar curve partitioning by moving-circle intersections

Input: Starting point parameter value t_0, second value of parameter t_n for solver's initial conditions interval – $[t_0, t_n]$, n – number of segments in partition, circle radius r for evaluating partition, *direction* is a bool value (TRUE for direction from start point, FALSE – from endpoint), *params* – list of curve's shape parameters

Output: tree T $\{node_0, node_1, node_2, \ldots\}$ of partition points and *partitions* – number of tree leaves (corresponding number of partition versions)

Local: *partitions* – number of partitions formed when a curve intersects a circle, *ancestors* is an array of node indexes, which are ancestors for the current step, *allroots* – number of intersection equation roots at the current step for all partitions, *roots* – array of t – values for current intersection between the curve ana the circle, *nroots* – number of roots for current intersection between the curve ana the circle

```
if direction then // check direction to initiate parameters
  u ← t_n
  T ← Tree(t_0, p(t_0, params))
else
  u ← t_0
  T ← Tree(t_n, p(t_n, params))
end if
partitions ← 1 // initialize number of partitions
ancestors ← {0} // array of ancestor nodes
for i ← 1…n do
  allroots ← 0
  for j ← 0…partitions - 1 do
    id ← ancestors[j]
    P_c ← T[id].point
    t_c ← T[id].t
    if direction then // check direction to find intersections
      roots ← FIND-ROOTS(x_c, y_c, t_c, u, params, r) // solve equation of an
intersection circle and curve
    else
      roots ← FIND-ROOTS(P_c, u, t_c, params, r)
    end if
    nroots ← LENGTH(roots) // number of equation roots
    for k ← 0…nroots - 1 do
      P_j ← p(roots[k], params)
      ADD-CHILD(T, id, roots[k], P_j)
    end for
    allroots ← allroots + nroots
  end for
  if allroots > partitions then // check number of partitions
    partitions ← allroots
  end if
  CLEAR(ancestors)
  for j ← 0…partitions - 1 do
    PUSH(ancestors, LENGTH(T) – partitions + j)
  end for
end for
return T, partitions
```

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

The optimal partition after n – 1 steps is selected as the one minimizing the difference between the residual segment length and the circle radius, subject to the placement of the residual interval. If we denote the number of partition variants for the tree generated in the direction from t_0 to t_n as m_l, and in the opposite direction as m_r, then the criterion can be defined by the following expression:

$$\Delta = \min\left(\left|\operatorname{sgn}(t_{rj} - t_{li})\left\|p(t_{rj}) - p(t_{li})\right\| - r\right|\right), i = 1\ldots m_l,\ j = 1\ldots m_r, \tag{10}$$

where t_{li}, and t_{rj} are the values of the curve parameter for the boundary partition points in the forward and backward directions, respectively.

To obtain a partition of the curve into equal-length chord segments based on the circle displacement method, it is necessary, after completing the partitioning loop, to compare the obtained value Δ with the allowable chord length deviation. Then, a decision is made to either continue the computations with an adjusted radius or stop the process using the obtained equal-chord partition points.

To determine the radius of the circle that yields an equal-chord partition, two approaches were used:

Uniform error distribution among all partition segments:

$$r = \frac{\Delta}{n} + \overline{r}, \tag{11}$$

where Δ is the difference (error) between the chord length of the residual segment and the radius of the partition circle, and $\overline{r}$ is the "previous" radius value. It should be noted that formula (6) provides the same result as averaging the chord length over all segments. The same approach was adopted in [17].

Minimization of the error as the objective function for optimizing the partition radius (Algorithm 2). Zero-order optimization methods can be applied for this purpose. After obtaining the radius value using this approach, the objective function value must be compared with the allowable tolerance. If the criterion is not met, the search continues over a modified radius interval; otherwise, the process terminates. As the initial search interval for the radius, the curve's maximum and minimum segment lengths were used, partitioned uniformly by parameter into *n* parts. After the optimal radius is found within the initial interval at a given optimizer iteration, if it does not satisfy the tolerance, the search continues on a reduced interval, and so on.

The computational complexity of obtaining the radius for equal-chord partitioning using this algorithm can be determined based on the worst-case scenario of the curve partitioning procedure with respect to *n*. Let *k* denote the number of iterations of the partitioning procedure required to achieve the specified accuracy (partition non-uniformity). Then, the desired complexity can be expressed as $O\left(k\frac{m(1-m^{n-1})}{1-m}\right)$. For the case of radius adjustment based on optimization, the total number of partitioning iterations is defined as the product

$$k = si_o, \tag{12}$$

where *s* is the number of calls to the OPTIMIZE procedure, and i_o is the number of iterations within the optimization procedure.

Experiments

Let's present the experimental study, which aims to assess the performance of the proposed algorithm and examine the impact of its parameters and modifications on partitioning time.

The software used in the experiments was implemented in *Julia* [20]. The choice of this language is justified by its modernity, high performance, and orientation towards computational mathematics tasks. The experiments were carried out using double-precision floating-point representation.

For the numerical solution of equations (9), the following Julia packages for nonlinear root-finding were used: *NonlinearSolve* [21], *IntervalRootFinding* [22], and *Roots* [23]. In solving the curve–circle intersection equation, the function *find_zeros* from the *Roots* package was applied to

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

Algorithm 2 Optimization-based search for equal-chord partition radius
Input: Starting point parameter value t_0, finish value of parameter t_n, n – number of segments in partition, n_1 – number of segments in direction from starting point, r_{min}, r_{max} – initial minimal and maximum bounds for radius search, number of iterations for optimizer – i_o, absolute error ε for the difference between segments length, *params* – list of curve's shape parameters
Output: r – value of radius for the equipartition of a plane curve
Local: Current values of the lower bound – a and the upper bound – b, and the stop bool value are used for breaking the iteration.

```
a ← r_min // initiate radius of partitions
b ← r_max
r, δ ← OPTIMIZE(Δ(t_0, t_n, n, n_1, params), a, b, i_o)// optimization for initial interval
while δ > ε do // stop condition
   a_1 ← a // split search interval into two parts
   b_1 ← r //
   a2 ← r //
   b_2 ← b //
   r_1, δ_1 ← OPTIMIZE(Δ(t_0, t_n, n, n_1, params), a_1, b_1, i_o)// optimization for first part of split interval
   r_2, δ_2 ← OPTIMIZE(Δ(t_0, t_n, n, n_1, params), a_2, b_2, i_o)// optimization for first part of split interval
   if r_1 < r_2 then // compare radii and define next search interval
      a ← a_1
      b ← b_1
      r ← r_1
      δ ← δ_1
   else
      a ← a_2
      b ← b_2
      r ← r_2
      δ ← δ_2
   end if
end while
return r // radius of equipartition
```

obtain multiple solutions, or a combination of the function *find_zero* from the same package with the function *roots* from *IntervalRootFinding*, which made it possible first to determine the intervals containing the roots and then to compute the solutions within each interval.

The *solve* function from the *NonlinearSolve* package or the *find_zero* function from Roots was used to obtain unique solutions. The *BenchmarkTools* package [24] was employed to conduct performance measurements. The results were processed and visualized using *MS Excel* and the *Plots* package. All computations were performed on a laptop with a quad-core Intel Core i5-6300HQ processor.

To test various aspects of the proposed partitioning algorithm, two specific planar parametric curves were selected – sixth-order Bézier curves with the standard partition interval [0,1], where the endpoints are the first and last points of their control polygon. Due to the different placements of the control polygon points, one curve (Fig. 4) is convex, while the other (Fig. 5) has two inflection points where the curvature changes sign. This choice of curves provided distinct conditions for applying the equal-chord partitioning algorithms and investigating their effects.

Now, let us examine the experiments related to the circle-displacement-based partitioning algorithm. The objective function is defined as the dependence of the absolute error–given by the difference between the chord length of the residual segment and the partition circle radius–on the radius value itself. When the required number of partition segments is distributed between the left and right trees, the form of the objective function varies depending on the position of the residual partition segment. As an illustration, we present the dependence of the absolute error on the partition radius for

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

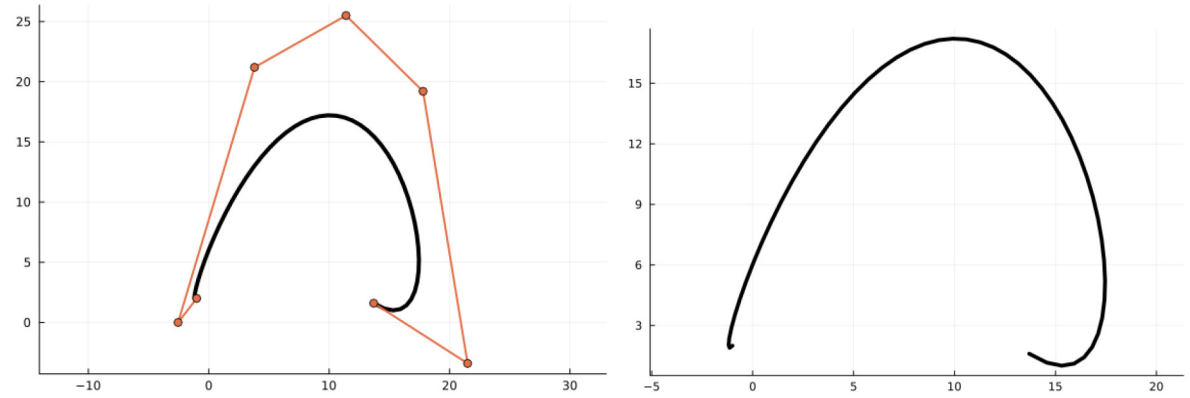

Fig. 4. A planar convex sixth-order Bézier curve to which the partitioning algorithms were applied: left – the curve with its control polygon defined by the points (–1.0; 2.0), (–2.55; 0.0), (3.8; 21.2), (11.4; –25.5), (17.8; 19.2), (21.5; –3.4), (13.7; 1.6); right – the shape of the Bézier curve

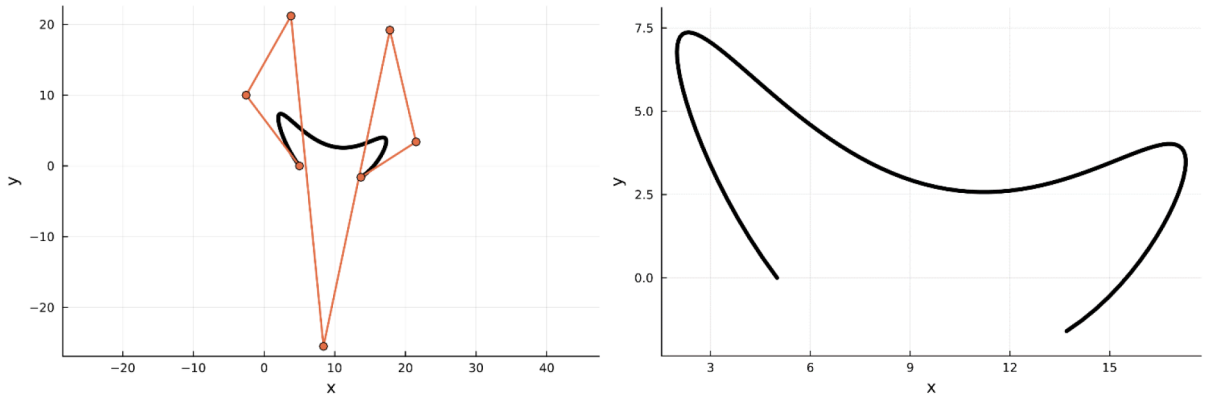

Fig. 5. A non-convex planar sixth-order Bézier curve to which the partitioning algorithms were applied: (a) the curve with its control polygon defined by the points (5.0; 0.0), (–2.55; 10.0), (3.8; 21.2), (8.4; –25.5), (17.8; 19.2), (21.5; 3.4), (13.7; –1.6); (b) the shape of the Bézier curve

a fixed number of partition segments and different $n_1$parameter values ($0 \le n_1 \le n - 1$), which governs the distribution of segments between the left and right trees.

Fig. 6 presents the objective function plots for the convex curve (Fig. 4), whereas Fig. 7 shows the corresponding plots for the non-convex curve (Fig. 5). In the first case (convex curve), the plots exhibit a smooth convergence to the minimum point without the appearance of local extrema, and their overall shapes remain similar across different values of n_1. By contrast, the second case (a non-convex curve) produces step-like plots with local extrema. This behavior is explained by the selection among multiple solutions of the intersection equation. The number and length of the steps in the non-convex case vary with the values of n_1. The number of steps increases for values corresponding to points farther from the endpoints ($n_1 = 4, 5$). This is a consequence of greater solution variability and the resulting increase in the number of branches within the partition trees.

Fig. 8 and 9 show the dependence of the absolute error on the number of partition segments when distributed across the trees according to the expression $n_1 = \lfloor n/2 \rfloor$. In both cases, these figures illustrate the reduction of the search interval with increasing discretization. In the case of the non-convex curve, it can also be observed that the shape of the error-dependence curves becomes smoother. This is because, as the number of segments increases, the variability of the solutions decreases (as does the partition circle radius). At a certain number of segments, the intersection of the circle with the curve will yield a single intersection point for any position of the circle's center along the curve within the partition interval.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

At the next stage, the influence of the curve partitioning degree n on the number of trajectory tree branches m and the number of partitioning iterations k required to achieve the specified chord

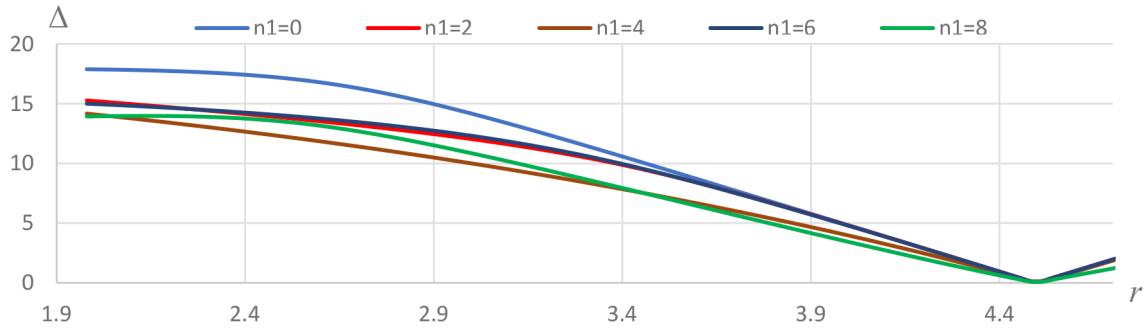

Fig. 6. Plots the dependence of the absolute error on the partition radius for the convex curve (Fig. 4) at $n = 9$ and different values of n_1

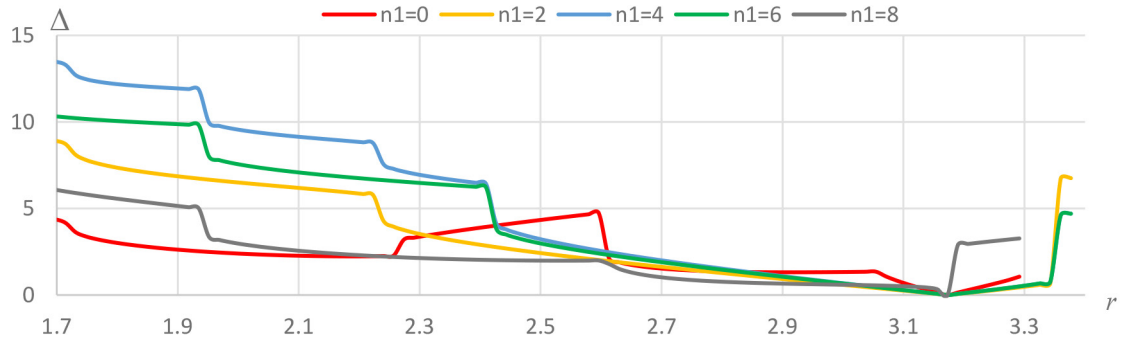

Fig. 7. Plots the dependence of the absolute error as a function of the partition radius for the non-convex curve (Fig. 5) at $n = 9$ and different values of n_1

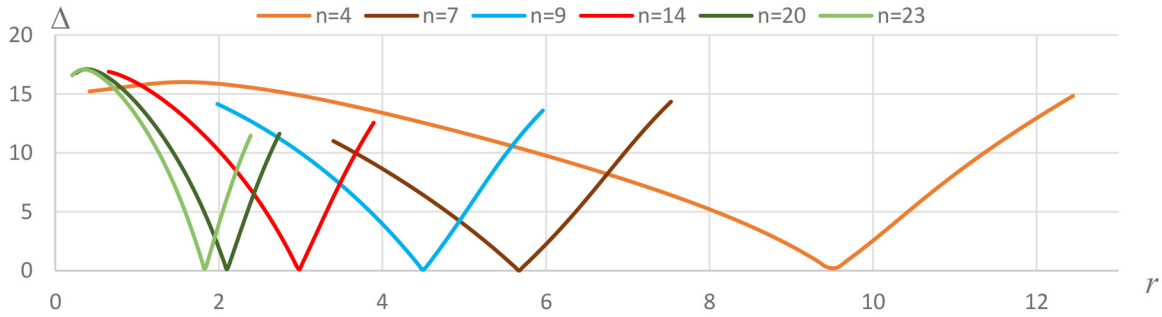

Fig. 8. Plots of the absolute error as a function of the partition radius for the convex curve (Fig. 4) at different values of n

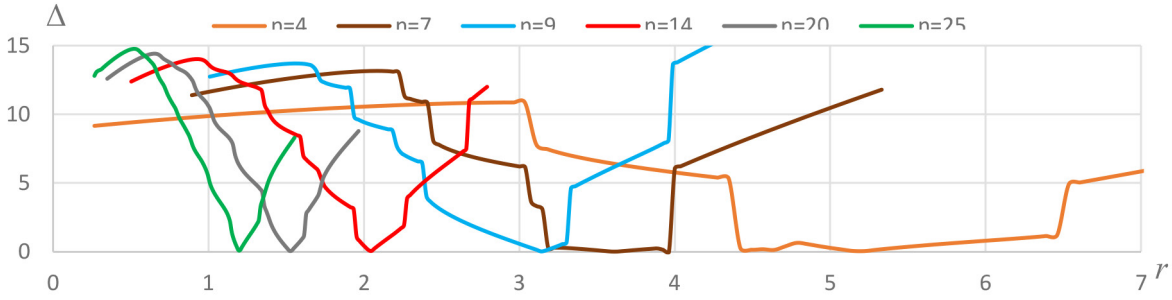

Fig. 9. Plots of the absolute error as a function of the partition radius for the non-convex curve (Fig. 5) at $n_1 = \lfloor n/2 \rfloor$ and different values of n

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

length deviation was investigated. The initial initialization of the partition circle radius, or the interval of its variation during search for the optimal value, was performed using uniform sampling of the curve parameter with the same number of nodes. For the convex curve, within the range of partition radius variation, no multiple solutions of the intersection equation were observed; that is, a single equal-chord partition trajectory was constructed.

First, we examined whether the parameter n_1, which distributes the partition nodes between the left and right trees, influences the number of algorithm iterations at a given error tolerance and fixed n. Fig. 10 shows the experimental results for the convex curve with nine partition segments and an allowable chord-length deviation $\varepsilon = 0.0001$. A single numerical solver for the intersection equation was employed to determine the partition points. The results were obtained for two approaches to setting the radius: (i) adjusting the initial value through error averaging, and (ii) optimization using Brent's method as implemented in Algorithm 2. In both cases, only minor fluctuations in the number of iterations were observed across different values of n_1. However, the number of iterations in the first approach was four times smaller, making it more efficient for convex curves.

Fig. 11 presents the experimental results for the non-convex curve under the same conditions. Since averaging the error for specific values failed to find the optimal value, often getting stuck in local extrema, it was decided to use only Brent's method and the algorithm shown in Algorithm 2. The experiment demonstrated that the number of iterations for values near the edges of the partition interval is 1.3–2 times higher than for values closer to half of, as explained by the shape of the objective function plot (Fig. 9).

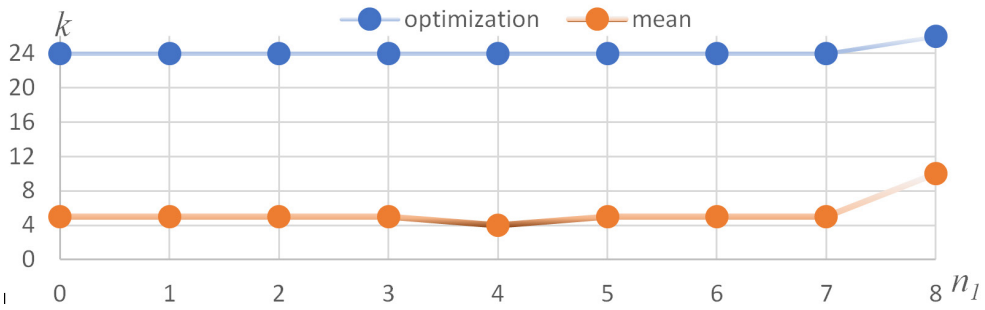

Fig. 10. The number of iterations of the circle-based partitioning algorithm for the curve in Fig. 4 as n_1 varies from 0 to $n - 1$, with $n = 9$ partition segments, allowable chord length deviation $\varepsilon = 0.0001$, and two methods for determining the optimal partition circle radius

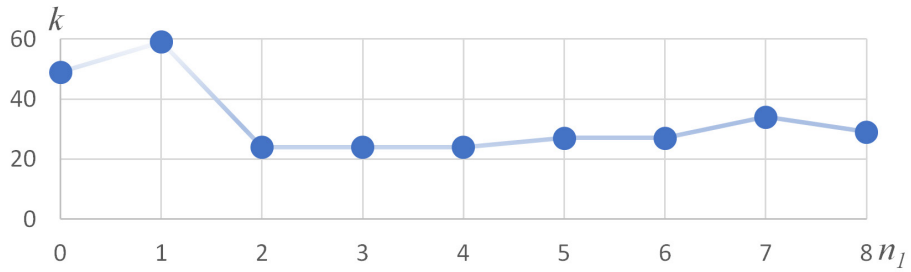

Fig. 11. The number of iterations of the circle-based partitioning algorithm for the curve in Fig. 5 as n_1 varies from 0 to $n - 1$, with $n = 9$ segments, allowable chord length deviation $\varepsilon = 0.0001$, and the optimal partition circle radius determined using Algorithm 2

Therefore, to obtain the partitioning results, the first method of determining the optimal radius value was chosen for the convex curve. In contrast, the second method was applied to the non-convex curve. In both cases, the distribution parameter value was set as half the number of segments according to $n_1 = \lfloor n/2 \rfloor$.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

Fig. 12 shows the results of varying the number of iterations required to obtain an equal-chord partition of the convex curve with a given error tolerance ε = 0.0001. The plot shows a decrease in the number of iterations with increasing partitioning degree, as expected for the averaging-based algorithm.

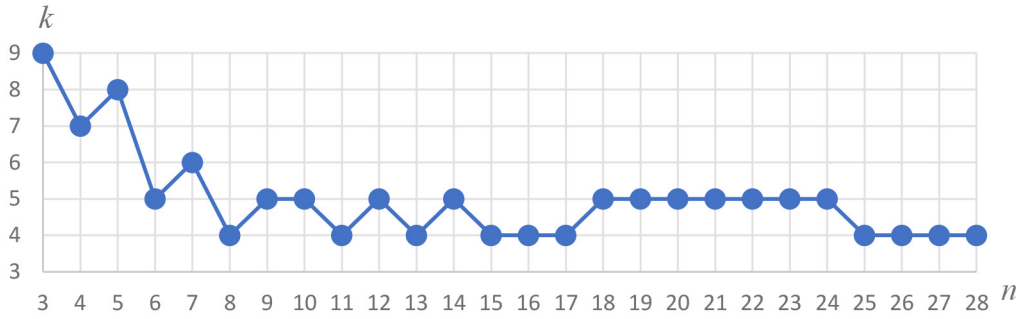

Fig. 12. The number of iterations of the circle-based partitioning algorithm for the curve in Fig. 4 as *n* varies from 3 to 28, with allowable chord length deviation ε = 0.0001 and $n_1 = \lfloor n/2 \rfloor$

In Fig. 13, the number of partition trajectories is reported, and in Fig. 14, the corresponding iteration counts for the non-convex curve (Fig. 5) are plotted against the number of partition segments. The analysis revealed that the number of trajectories arising from multiple intersection roots was reduced from nine to one. However, unlike the convex-curve algorithm with uniform error distribution, Brent's method–based algorithm showed no clear trend between the number of iterations and the partitioning degree.

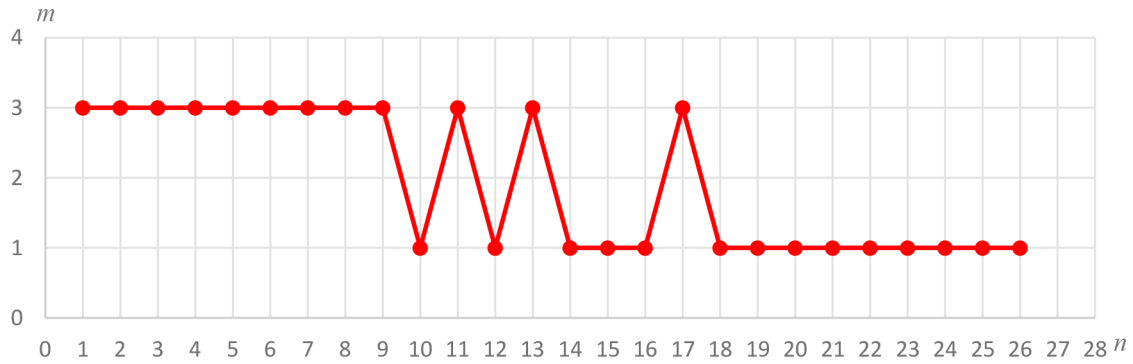

Fig. 13. The number of partition trajectories in the circle-based partitioning algorithm for the curve shown in Fig. 5, as the number of segments varies from 3 to 28, with allowable chord length deviation ε = 0.0001 and $n_1 = \lfloor n/2 \rfloor$

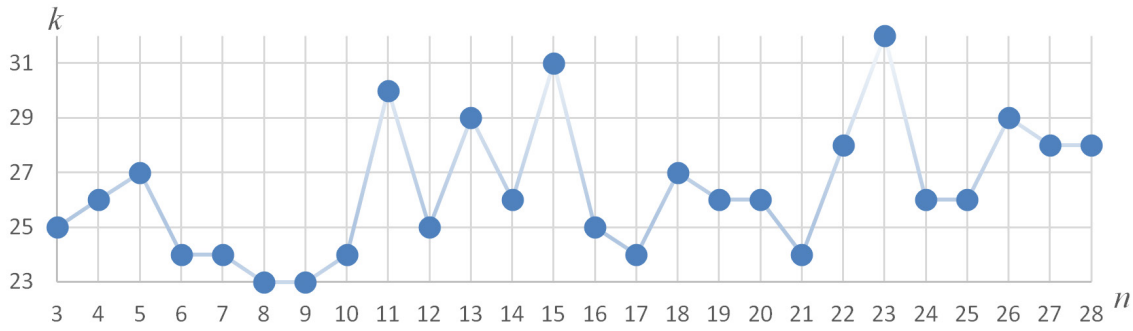

Fig. 14. The number of iterations of the circle-based partitioning algorithm for the curve in Fig. 5 as the number of segments varies from 3 to 28, with allowable chord length deviation ε = 0.0001 and $n_1 = \lfloor n/2 \rfloor$

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

Next, the influence of the allowable error in segment lengths ε on the number of iterations required for partitioning was investigated. For this purpose, experiments analogous to the previous ones were conducted with error values 0.001, 0.0001, 0.00001 and 0.000001. Fig. 15 presents the results for the convex curve, where the method with uniform redistribution of the difference between the chord length of the residual segment and the partition circle radius was used. The growth in iteration count is more significant for small values of n (4–6 iterations), whereas with increasing n, it falls to 1–2 iterations, despite a 1000-fold reduction in the error tolerance. Experiments for the non-convex curve using the optimization-based approach to find the partition radius (Table 1) revealed practically no dependence on accuracy improvement within this range of values. Only in one case, when $n = 23$ the number of iterations increased from 32 to 131, because the required partition accuracy could not be achieved in a single optimization step.

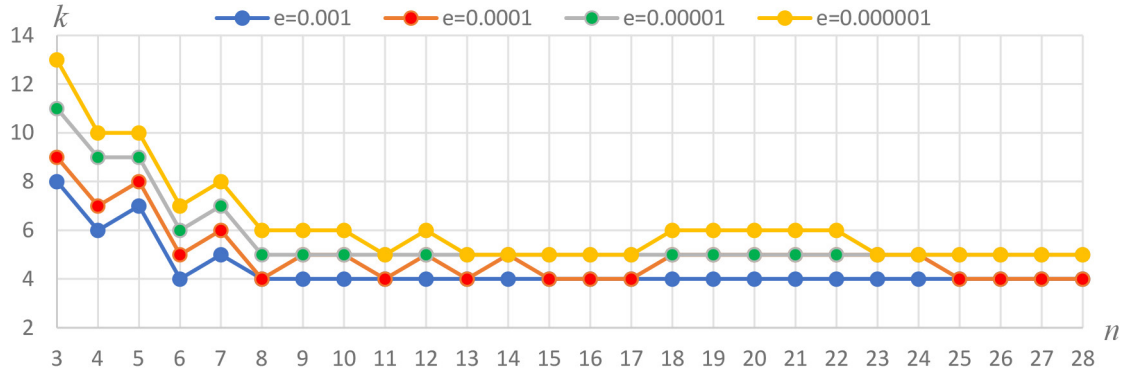

Fig. 15. The number of iterations of the circle-based partitioning algorithm for the curve in Fig. 4 as n varies from 3 to 28, with allowable chord length deviation ε = 0.001, 0.0001, 0.00001, 0.000001 and $n_1 = \lfloor n/2 \rfloor$

At the final stage, equal-chord partitioning experiments were conducted for a more complex curve case. The purpose of these experiments was to demonstrate the applicability of the proposed algorithm and to compare its performance with simpler cases. Fig. 16 shows a fifteenth-degree Bézier curve to which the proposed algorithm was applied.

For this curve, equal-chord partitioning was performed for segment counts n ranging from 3 to 100. The same numerical methods used to solve the intersection equation in the previous experiments were employed. An optimization-based approach using Brent's method was applied to determine the partition radius. The distribution of the segment count was controlled by the parameter $n_1 = \lfloor n/2 \rfloor$. The tolerance for segment inequality was set to ε = 0.0001. Fig. 17 illustrates the maximum number of branches m observed among the partition trees as the number of segments n varies. In contrast to the simpler curve case shown in Fig. 5, the value of m increases substantially and reaches values of up to 19. This growth indicates a significantly higher structural complexity in the partitioning process, arising from increased geometric complexity and a higher degree of the Bézier curve. As a result, the number of alternative intersection configurations increases, leading to deeper, more branched partition trees.

Fig. 18 presents the number of iterations required to achieve the prescribed partitioning accuracy as the number of segments n varies. With the exception of four values that can be considered outliers due to the shape of the objective function, the iteration counts remain fairly stable, ranging from 20 to 30. This level of convergence is comparable to that observed for the simpler curve shown in Fig. 16.

Finally, we compare the average execution time required to partition the curves shown in Fig. 16 and Fig. 5. Fig. 19 illustrates the execution time as a function of the number of segments for both curve variants. As shown, the partitioning time for the curve in Fig. 16 exhibits substantial fluctuations. These variations can be attributed to both the greater number of branches in the partition trees and the increased computational cost of numerically solving higher-order intersection equations. Starting from a segment count of n = 74, where only a single branch remains, the partitioning time stabilizes

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

Table 1

The number of iterations of the circle-based partitioning algorithm for the curve in Fig. 5 as *n* varies from 3 to 28, with allowable chord length deviation ε = 0.001, 0.0001, 0.00001 and $n_1 = \lfloor n/2 \rfloor$

n	***k***			
	ε = 0.001	**ε = 0.0001**	**ε = 0.00001**	**ε = 0.000001**
3	27	27	24	25
4	26	25	27	26
5	27	27	27	27
6	24	24	24	24
7	24	24	24	24
8	23	23	23	23
9	23	23	23	23
10	24	24	24	24
11	30	30	30	30
12	25	25	25	25
13	29	29	29	29
14	26	26	26	26
15	31	31	31	31
16	25	25	25	25
17	24	24	24	24
18	27	27	27	27
19	26	26	26	26
20	26	26	26	26
21	24	24	24	24
22	28	28	28	28
23	32	32	131	131
24	26	26	26	26
25	26	26	26	26
26	29	29	29	29
27	28	28	28	28
28	28	28	28	28

with a significantly reduced fluctuation amplitude and subsequently shows an approximately linear growth with respect to *n*. For the simpler curve shown in Fig. 5, the partitioning time increases almost linearly over the entire range of *n*. When comparing the two curves, a predictably higher execution time is observed for the curve in Fig. 16. However, it should be noted that after the influence of multiple branches in the partition trees disappears (i.e., after $n = 74$) the average partitioning time for the more complex curve exceeds that of the simpler curve by a factor of approximately 2.5.

The results obtained in this study are compared with those of related prior work. In [12], the Iso-Level Algorithm (ILA) was proposed, which partitions curves into iso-chord segments based on a grid approximation of the distance function between two points. The computational complexity of this method is $O(n \cdot m^3)$, where *m* denotes the number of grid lines. Since $m > n$, the effective complexity exceeds $O(n^4)$, resulting in considerable computational inefficiency. Further confirmation of this limitation is provided by the experimental study in [12], which considered only a relatively small number of segments ($n \leq 12$).

In contrast, the algorithms proposed herein apply to substantially larger values of *n*. The empirical analysis indicates that, in practical scenarios, the proposed methods exhibit near-linear growth, representing a significant improvement in computational efficiency while preserving segmentation accuracy for complex curves.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

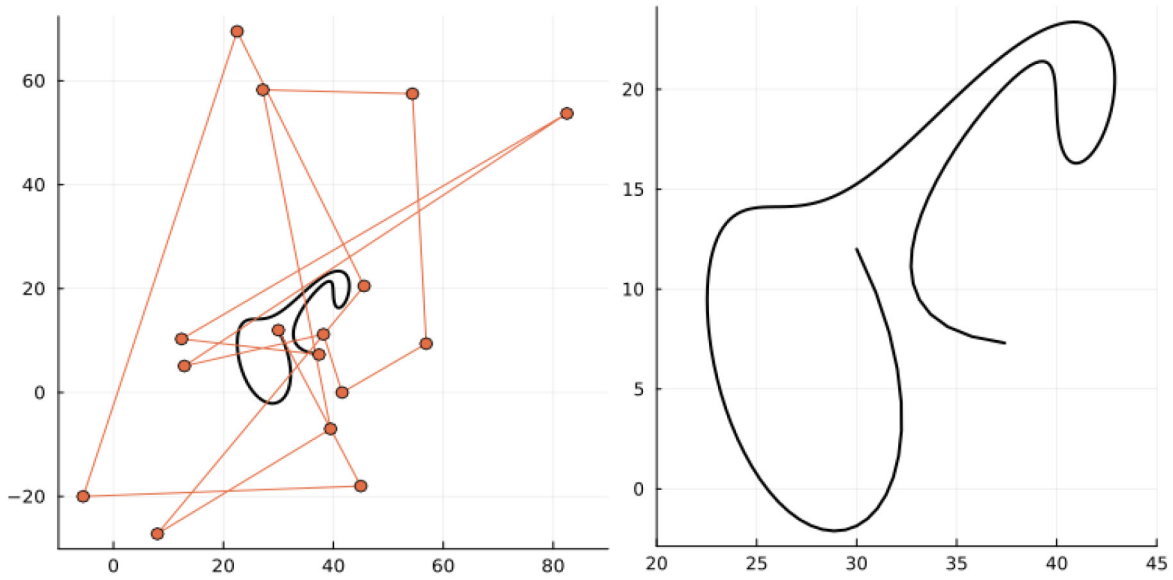

Fig. 16. A non-convex planar fifteen-order Bézier curve to which the partitioning algorithms were applied: (a) the curve with its control polygon defined by the points (30.0; 12.0), (45.0; –18.0), (–5.5; –20.0), (22.5; 69.5), (45.6; 20.5), (8.0; –27.2), (39.5; –7.0), (27.2; 58.25), (54.4; 57.5), (56.9; 9.4), (41.6; 0.0), (38.2; 11.2), (12.9; 5.1), (82.5; 53.7), (12.4; 10.3), (37.4; 7.3); (b) the shape of the Bézier curve

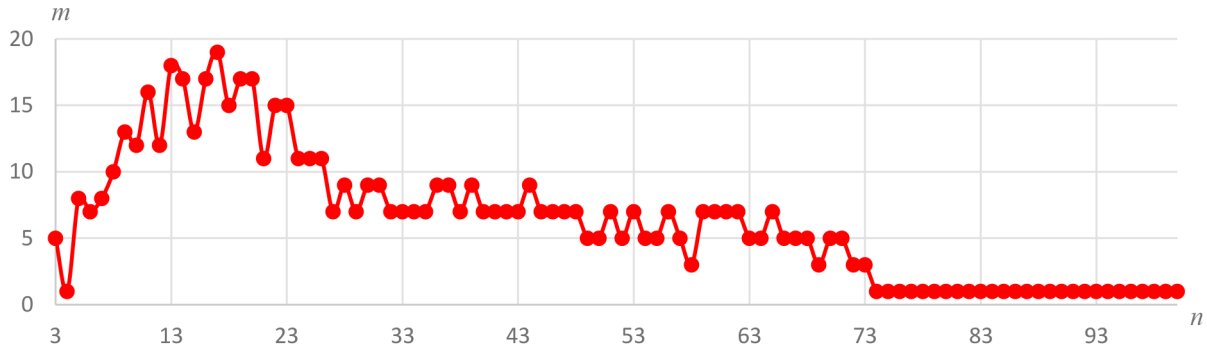

Fig. 17. The number of partition trajectories in the circle-based partitioning algorithm for the curve shown in Fig. 16, as the number of segments varies from 3 to 100, with allowable chord length deviation $\varepsilon = 0.0001$ and $n_1 = \lfloor n/2 \rfloor$

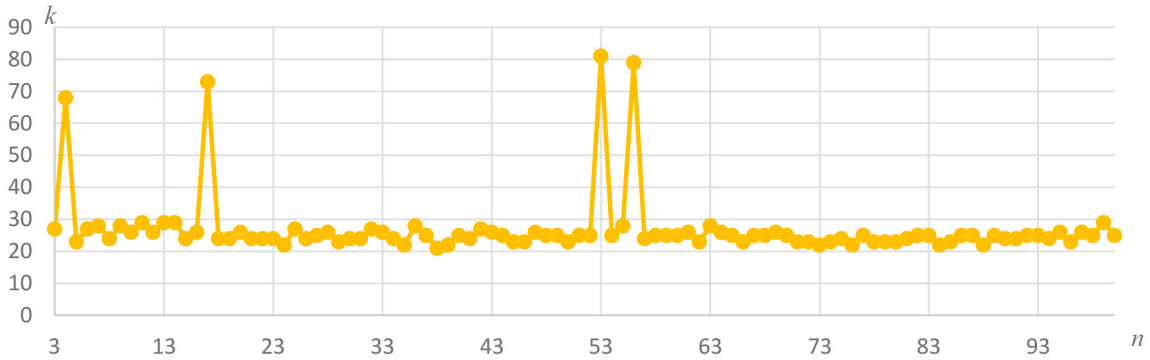

Fig. 18. The number of iterations of the circle-based partitioning algorithm for the curve in Fig. 16 as *n* varies from 3 to 100, with allowable chord length deviation $\varepsilon = 0.0001$ and $n_1 = \lfloor n/2 \rfloor$

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

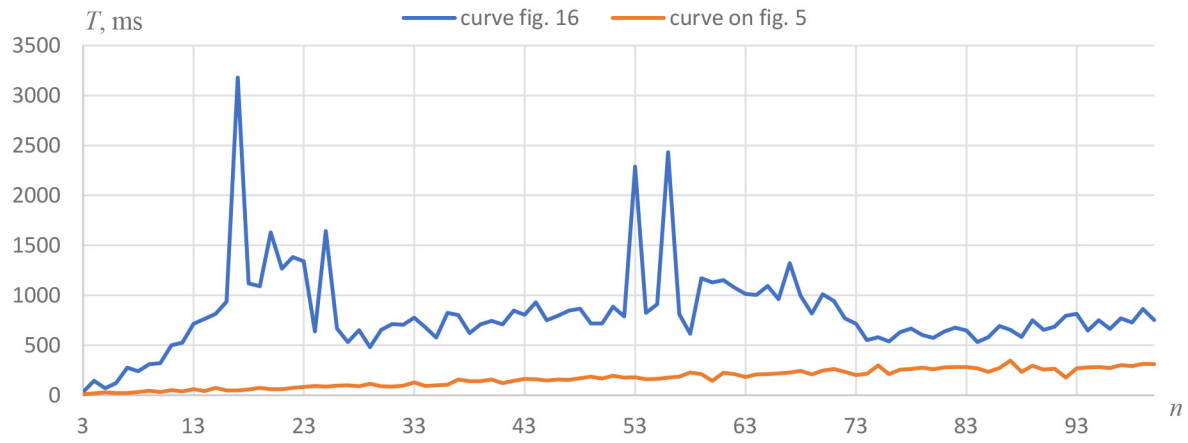

Fig. 19. Execution time for the circle-intersection-based algorithm for the curve in Fig. 5 (red) and the curve in Fig. 16 (blue) with allowable chord length deviation ε = 0.0001

A further advantage of the proposed algorithms is their direct extension to spatial curves. In this variant, the first algorithm replaces circles with partitioning spheres.

The ECLD algorithm, as presented in [14], addresses a slightly different problem: determining partition points on a curve for a predefined chord length. Although the idea of applying a binary search to determine the position of a point on a curve corresponding to a prescribed chord length can be incorporated into the algorithm proposed in this work, this approach is applicable only when there is a unique intersection.

Conclusions

This study addresses the problem of partitioning a planar curve into segments of equal chord length. Proposed was an approach to solving this problem in the “classical” formulation based on intersecting the curve with a circle of fixed radius. Several algorithmic variants were considered for the circle-intersection-based approach, employing different methods for determining the radius value. The algorithm structure was presented, including the curve partitioning procedure and the optimization of the radius parameter.

The experimental section included studies on the dependence of the number of iterations required to achieve the desired partitioning accuracy on the number of segments for different algorithmic variants, as well as the impact of increasing accuracy on the iteration count.

The experiments confirmed that the proposed algorithms are suitable for segmenting planar parametric curves into equal chord-length segments over a wide range of segment counts. For convex curves, the moving-circle algorithm with uniform error redistribution across segments requires fewer iterations because it avoids multiple intersection roots. Overall, the experimental results demonstrate that the proposed algorithm exhibits stable convergence behavior across curves of varying geometric complexity. Despite the increased number of partition branches and local irregularities in the objective function for higher-degree curves, the number of iterations required to reach the prescribed accuracy remains nearly constant. This suggests that the convergence rate is primarily determined by the properties of the optimization procedure rather than by the specific shape or degree of the curve, supporting the robustness and practical applicability of the proposed approach.

Summarizing the conducted research and further developing the proposed approach to equal-chord partitioning of parametric curves, the following promising directions for future work can be identified:

– extending the algorithm to the case of spatial parametric curves;

– examining the efficiency of employing parallel computations in the partitioning procedures.

https://doi.org/10.32782/mathematical-modelling/2026-9-1-39 ISSN 2618-0332 (print), 2618-0340 (online)

Bibliography

1. Shpitalni M., Koren Y., Lo C.C. Real-time curve interpolators. *Computer-Aided Design*. 1995. Vol. 26, № 11. P. 832–838. https://doi.org/10.1016/0010-4485(94)90097-3
2. Panagiotakis C., Tziritas G. Any dimension polygonal approximation based on equal errors principle. *Pattern Recognition Letters*. 2007. Vol. 28. P. 582–591. https://doi.org/10.1016/j.patrec.2006.10.005
3. Zhong W., Luo X., Chang W. A real-time interpolator for parametric curves. International *Journal of Machine Tools and Manufacture*. 2018. Vol. 125. P. 133–145. https://doi.org/10.1016/j.ijmachtools.2017.11.010
4. Wei J., Sun C., Zhang X. J., Wang E. J., Law D. An efficient and accurate interpolation method for parametric curve machining. *Scientific Reports*. 2022. Vol. 12, № 1. Art. 16000. https://doi.org/10.1038/s41598-022-20018-9
5. Han X. T., Zhu K. F., Wang X. B. A hash approach to refine CNC computation of arc length and parameter of NURBS with high efficiency and precision. *International Journal of Precision Engineering and Manufacturing*. 2024. Vol. 25. P. 1243–1256. https://doi.org/10.1007/s12541-024-00976-y
6. Chalkis A., Katsamaki C., Tonelli-Cueto J. On the error of random sampling uniformly distributed random points on parametric curves. *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*, ISSAC ’22 (05 July 2022 New York, United States). 2022. P. 273–282. https://doi.org/10.1145/3476446.3536190
7. Pagani L., Scott P.J. Curvature based sampling of curves and surfaces. *Computer Aided Geometric Design*. 2018. Vol. 59. P. 32–48. https://doi.org/10.1016/j.cagd.2017.11.004
8. Hernández-Mederos V., Estrada-Sarlabous J. Sampling points on regular parametric curves with control of their distribution. *Computer Aided Geometric Design*. 2003. Vol. 20, № 6. P. 363–382. https://doi.org/10.1016/S0167-8396(03)00079-7
9. Huang L., Cao Y., Peng R., Wei G. Adaptive sampling technique for free-form surface of aero-engine blades based on discrete curvature. *International Journal of Precision Engineering and Manufacturing*. 2025. Vol. 26. P. 325–344. https://doi.org/10.1007/s12541-024-01161-x
10. Firsov A. Study of the mathematical models of optimal partitioning for particular cases. *Eastern-European Journal of Enterprise Technologies*. 2018. Vol. 1, № 4(91). P. 69–76. https://doi.org/10.15587/1729-4061.2018.123261
11. Lopez M.A., Reisner S. A note on curves equipartition. *arXiv*. 2008. P. 1–3. https://doi.org/10.48550/arXiv.0707.4298
12. Panagiotakis C., Athanassopoulos K., Tziritas G. The equipartition of curves. *Computational Geometry*. 2009. Vol. 42, № 6–7. P. 677–689. https://doi.org/10.1016/j.comgeo.2009.01.003
13. Panagiotakis C. Fast equipartition of complex 2D shapes with minimal boundaries. *Algorithms*. 2025. Vol. 18, № 5. Art. 277. https://doi.org/10.3390/a18050277
14. Wang X.-R., Wang Z.-Q., He P., Lin T.-S. The high-energy micro-arc spark–computer numerical control deposition of planar NURBS curve coatings. *International Journal of Advanced Manufacturing Technology*. 2016. Vol. 87. P. 3325–3335. https://doi.org/10.1007/s00170-016-8698-x
15. Xu Y., Zhou Z. Polygons inscribed in Jordan curves with prescribed edge ratios. *Topology and Its Applications*. 2024. Vol. 354. Art. 108971. https://doi.org/10.1016/j.topol.2024.108971
16. Manivel M., Silva M., Thompson R. Iterative respacing of polygonal curves. *SN Computer Science*. 2022. Vol. 3. Art. 419. https://doi.org/10.1007/s42979-022-01343-2
17. Frolov O. Equipartition algorithm for a flat parametric curve based on the intersection between it and a moving circle. *Cybernetics and Computer Technologies*. 2025. № 1. P. 12–31. https://doi.org/10.34229/2707-451X.25.1.2

18. Theoretical and applied aspects of software engineering: architectural solutions, algorithmic methods and intelligent systems / I. Ushakova, O. Frolov, L. Znakhur, S. Znakhur, Yu. Chyrva. Kharkiv : S. Kuznets KhNEU, 2026. 223 p.
19. Divide. Rhinoceros 3D. URL: https://docs.mcneel.com/rhino/8/help/en-us/commands/divide.htm
20. The Julia Programming Language: Official site. URL : https://julialang.org/ (дата звернення : 02.03.2026).
21. Getting started with nonlinear root finding in Julia. URL: https://docs.sciml.ai/NonlinearSolve/stable/tutorials/getting_started/ (дата звернення : 28.02.2026).
22. IntervalRootFinding.jl. URL: https://juliaintervals.github.io/IntervalRootFinding.jl/v0.1/index.html (дата звернення : 18.02.2026).
23. Roots.jl. URL: https://juliamath.github.io/Roots.jl/stable/ (дата звернення : 22.02.2026).
24. BenchmarkTools.jl. URL: https://juliaci.github.io/BenchmarkTools.jl/stable/ (дата звернення : 03.03.2026).

References

1. Shpitalni, M., Koren, Y., & Lo, C. C. (1995). Real-time curve interpolators. *Computer-Aided Design*, *26*(11), 832–838. https://doi.org/10.1016/0010-4485(94)90097-3 [in English].
2. Panagiotakis, C., & Tziritas, G. (2007). Any dimension polygonal approximation based on equal errors principle. *Pattern Recognition Letters*, *28*, 582–591. https://doi.org/10.1016/j.patrec.2006.10.005 [in English].
3. Zhong, W., Luo, X., & Chang, W. (2018). A real-time interpolator for parametric curves. *International Journal of Machine Tools and Manufacture*, *125*, 133–145. https://doi.org/10.1016/j.ijmachtools.2017.11.010 [in English].
4. Wei, J., Sun, C., Zhang, X. J., Wang, E. J., & Law, D. (2022). An efficient and accurate interpolation method for parametric curve machining. *Scientific Reports*, *12*(1), Article 16000. https://doi.org/10.1038/s41598-022-20018-9 [in English].
5. Han, X. T., Zhu, K. F., & Wang, X. B. (2024). A hash approach to refine CNC computation of arc length and parameter of NURBS with high efficiency and precision. *International Journal of Precision Engineering and Manufacturing*, *25*, 1243–1256. https://doi.org/10.1007/s12541-024-00976-y [in English].
6. Chalkis, A., Katsamaki, C., & Tonelli-Cueto, J. (2022). On the error of random sampling uniformly distributed random points on parametric curves. *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation (ISSAC '22)* (pp. 273–282). Association for Computing Machinery. https://doi.org/10.1145/3476446.3536190 [in English].
7. Pagani, L., & Scott, P. J. (2018). Curvature based sampling of curves and surfaces. *Computer Aided Geometric Design*, *59*, 32–48. https://doi.org/10.1016/j.cagd.2017.11.004 [in English].
8. Hernández-Mederos, V., & Estrada-Sarlabous, J. (2003). Sampling points on regular parametric curves with control of their distribution. *Computer Aided Geometric Design*, *20*(6), 363–382. https://doi.org/10.1016/S0167-8396(03)00079-7 [in English].
9. Huang, L., Cao, Y., Peng, R., & Wei, G. (2025). Adaptive sampling technique for free-form surface of aero-engine blades based on discrete curvature. *International Journal of Precision Engineering and Manufacturing*, *26*, 325–344. https://doi.org/10.1007/s12541-024-01161-x [in English].
10. Firsov, A. (2018). Study of the mathematical models of optimal partitioning for particular cases. *Eastern-European Journal of Enterprise Technologies*, *1*(4(91)), 69–76. https://doi.org/10.15587/1729-4061.2018.123261 [in English].
11. Lopez, M. A., & Reisner, S. (2008). A note on curves equipartition. *arXiv*. https://doi.org/10.48550/arXiv.0707.4298 [in English].

12. Panagiotakis, C., Athanassopoulos, K., & Tziritas, G. (2009). The equipartition of curves. *Computational Geometry*, 42(6–7), 677–689. https://doi.org/10.1016/j.comgeo.2009.01.003 [in English].
13. Panagiotakis, C. (2025). Fast equipartition of complex 2D shapes with minimal boundaries. *Algorithms*, *18*(5), Article 277. https://doi.org/10.3390/a18050277 [in English].
14. Wang, X.-R., Wang, Z.-Q., He, P., & Lin, T.-S. (2016). The high-energy micro-arc spark–computer numerical control deposition of planar NURBS curve coatings. *The International Journal of Advanced Manufacturing Technology*, 87, 3325–3335. https://doi.org/10.1007/s00170-016-8698-x [in English].
15. Xu, Y., & Zhou, Z. (2024). Polygons inscribed in Jordan curves with prescribed edge ratios. *Topology and Its Applications*, 354, Article 108971. https://doi.org/10.1016/j.topol.2024.108971 [in English].
16. Manivel, M., Silva, M., & Thompson, R. (2022). Iterative respacing of polygonal curves. *SN Computer Science*, 3, Article 419. https://doi.org/10.1007/s42979-022-01343-2 [in English].
17. Frolov, O. (2025). Equipartition algorithm for a flat parametric curve based on the intersection between it and a moving circle. *Cybernetics and Computer Technologies*, 1, 12–31. https://doi.org/10.34229/2707-451X.25.1.2 [in English].
18. Ushakova, I., Frolov, O., Znakhur, L., Znakhur, S., & Chyrva, Y. (2026). Theoretical and applied aspects of software engineering: Architectural solutions, algorithmic methods and intelligent systems. Kharkiv : S. Kuznets KhNEU, 223 p. [in English].
19. McNeel. (2024). Divide. *Rhinoceros 3D documentation*. Retrived from: https://docs.mcneel.com/rhino/8/help/en-us/commands/divide.htm [in English].
20. The Julia Programming Language. (2025). Retrived from: https://julialang.org/ [in English].
21. SciML. (2024). Getting started with nonlinear root finding in Julia. Retrived from: https://docs.sciml.ai/NonlinearSolve/stable/tutorials/getting_started/ [in English].
22. JuliaIntervals. (2024). IntervalRootFinding.jl. https://juliaintervals.github.io/IntervalRootFinding.jl/v0.1/index.html [in English].
23. JuliaMath. (2024). Roots.jl. Retrived from: https://juliamath.github.io/Roots.jl/stable/
24. JuliaCI. (2024). BenchmarkTools.jl. Retrived from: https://juliaci.github.io/BenchmarkTools.jl/stable/ [in English].

Frolov Oleg Vasylovych – Candidate of Technical Sciences, Associate Professor, Associate Professor at the Information Systems Department of the Simon Kuznets Kharkiv National University of Economics. E-mail: frolgx@gmail.com, ORCID: 0000-0002-2250-5857.

Фролов Олег Васильович – к.т.н., доцент, доцент кафедри інформаційних систем Харківського національного економічного університету імені Семена Кузнеця. E-mail: frolgx@gmail.com, ORCID: 0000-0002-2250-5857.

Дата першого надходження статті до видання: 05.03.2026
Дата прийняття статті до друку після рецензування: 24.04.2026
Дата публікації (оприлюднення) статті: 01.07.2026