ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20260502123125

Preliminary communication
Received: 2026-05-02, Accepted: 2026-06-01

Geometric Modeling of 2D Regions in AutoCAD Using the Marching Squares Algorithm with R-Functions

Nuraliev Fakhriddin Murodillayevich, Inoyatov Mirzayor Bakhtiyor ugli*, Ibodullaev Sardor Nasriddin ugli, Umarova Dildora Bakhtiyarovna, Giyosov Ulugbek Eshpulatovich

Abstract: This study presents a methodology for geometric modeling of two-dimensional regions in AutoCAD based on the theory of R-functions by V. L. Rvachev combined with the Marching Squares algorithm. A specialized RFM 2D Plugin was developed in C# for the AutoCAD.NET API, enabling users to define geometric domains through implicit functions of the form $f(x, y) = 0$ and to construct complex composite regions via R-operations (conjunction, disjunction, and negation). Region boundaries are extracted automatically using the Marching Squares algorithm with bisection-based edge interpolation (an accuracy of $\Delta x/1024$) and are rendered as closed AutoCAD Polyline contours filled with Hatch objects and unified into a single editable Group. The method was validated on three classes of planar domains — four simple primitives (circle, ellipse, rectangle, triangle), parabolic and hyperbolic regions, and three complex composite models (a house, a six-pointed star, and a chess-king piece) composed of up to 15 primitive functions. The results show that a single analytical expression replaces the multi-command sequences required by conventional parametric modeling, that the number of contour points grows linearly and the computation time quadratically with grid resolution, and that boundary smoothness can be controlled through a single Rα parameter. The study demonstrates that embedding R-functions theory directly within the AutoCAD .NET environment yields a unified, flexible, and practically applicable framework for modeling irregular and analytically defined planar domains.

Keywords: 2D geometric modeling; AutoCAD; Hatch; implicit domain; Marching Squares; R-functions

1 INTRODUCTION

Computer-Aided Geometric Modeling (CAGM) is an integral part of modern engineering design and is widely used in construction, mechanical engineering, aerospace, and other fields. In professional CAD systems such as AutoCAD, 2D regions are typically created using parametric commands—"Circle", "Ellipse", "Rectangle"—and their associated "Trim", "Fillet", and "Boolean" operations [1, 2]. However, conventional parametric methods exhibit significant limitations and complexity when constructing regions with intricate geometries—such as intersections of multiple curved boundaries, or regions bounded by parabolic or hyperbolic curves. In particular, drawing each boundary curve individually and then manually trimming and joining them is a multi-step process that reduces design efficiency [3, 4].

An effective alternative to this problem is modeling based on implicit functions (i.e., $F(x, y) = 0$). In this approach, a geometric region is defined by $f(x, y) \geq 0$, where $f(x, y) = 0$ denotes the boundary, $f(x, y) > 0$ the interior, and $f(x, y) < 0$ represents the exterior [5, 6, 7]. The mathematical foundation of implicit modeling is the R-functions theory proposed by academician V.L. Rvachev in 1963 [8, 9]. R-function is analytical analogs of logical operations (conjunction $\wedge$, disjunction $\vee$, negation $\neg$), which enable set-theoretic operations on geometric regions using continuous, differentiable functions [10, 11].

To display the boundary of an implicitly defined region on screen or within a CAD system, it must be converted into a sequence of points forming a polyline. For this purpose, the Marching Squares (MS) algorithm is employed—the two-dimensional analogue of the Marching Cubes algorithm proposed by Lorensen and Cline in 1987 [12]. The Marching Squares algorithm examines the sign change of the function at each cell of a 2D planar grid and generates isoline segments based on 16 possible configurations [13, 14]. Maple [13] was the first to investigate this algorithm in the context of 2D geometric design. Newman and Yi [5] presented a comprehensive survey of the Marching Cubes/Squares algorithms.

Shapiro [10] extensively covered the application of R-functions in geometric modeling and computational mechanics, introducing the concept of semi-analytic geometry. Pasko et al. [15] formalized the Function Representation (FRep) concept and developed a methodology for performing set-theoretic operations, blending, offset, and metamorphosis on implicit functions. Ricci [16] proposed a constructive geometry approach for computer graphics, a work that served as an important foundation for practical applications of R-functions theory.

The R-Function Method (RFM) is also widely applied in engineering. Rvachev and Sheiko [17] demonstrated its broad use in boundary value problems in mechanics. Tsukanov and Shapiro [18, 19] developed the SAGE meshfree system and successfully applied the RFM to natural frequency analysis of 2D structures. Kurpa et al. [20] applied the RFM to vibration analysis of porous functionally graded material (FGM) plates, demonstrating its contemporary engineering relevance.

However, there are almost no studies in the existing literature on the direct application of R-functions theory to the AutoCAD environment—that is, the development of a practical software tool that automatically visualizes 2D regions as Polyline and Hatch objects based on user-entered formulas. The aim of the present study is to fill precisely this gap by developing a plugin that integrates R-function with the Marching Squares algorithm and operates via the AutoCAD.NET API, and by evaluating its effectiveness experimentally.

AutoCAD was deliberately selected as the target platform for several complementary reasons. First, it is one of the most widely deployed CAD systems in industry, with a global user base of more than 38.5% according to 6sense

market analytics, which maximizes the practical reach of the developed tool [21]. Second, AutoCAD exposes a mature, well-documented managed programming interface — the AutoCAD.NET API — that provides direct access to the drawing database, to the geometric entities (Polyline, Hatch, Region), and to the command pipeline required for automatic visualization. Third, its native object model maps naturally onto the output of the Marching Squares algorithm: isoline segments become Polylines and filled domains become Hatch objects without any intermediate file-format conversion. Finally, AutoCAD is a de facto industry standard in architecture, civil, and mechanical engineering, so a plugin embedded in this environment can be integrated directly into established professional workflows rather than requiring a separate, stand-alone application.

The scientific novelty of the study consists of: (1) the practical application of R-functions theory and the Marching Squares algorithm integrated with the AutoCAD.NET API; (2) the presentation of numerical results for simple and complex 2D domains and a comparative analysis with conventional AutoCAD commands.

2 LITERATURE REVIEW

The literature analyzed in this section was identified through a systematic search of the following bibliographic databases: Scopus, Web of Science, IEEE Xplore, the ACM Digital Library, and Google Scholar. The search combined the key terms “R-functions”, “implicit modeling” / “implicit surfaces”, “Marching Squares” / “Marching Cubes”, “function representation (FRep)”, “constructive solid geometry”, and “AutoCAD .NET plugin” using Boolean operators, for example: (“R-functions” AND (“Marching Squares” OR “implicit modeling”) AND (“CAD” OR “AutoCAD”)). The search covered publications from 1963 to 2026. Studies were included if they addressed (i) the theoretical foundations of R-functions, (ii) the polygonization or boundary extraction of implicit domains, or (iii) engineering applications of implicit / function-based modeling; studies were excluded if they did not involve implicit or function-based representations, or were not available in English or Russian. After removing duplicates and applying these criteria, 33 core works were retained and are organized below into five thematic subsections.

2.1 Foundations of R-functions Theory

The founder of R-functions theory, academician V. L. Rvachev, proposed in 1963 a new method for the analytical description of geometric objects [8]. In his fundamental monograph published in 1982 [9], Rvachev developed the complete mathematical theory of R-functions and applied it to the solution of boundary value problems. An R-function is a real-valued function whose sign (positive or negative) depends solely on the signs of its arguments. This property allows logical operations to be expressed as continuous, differentiable functions, which is of fundamental importance in geometric modeling.

Shapiro [11] was the first to extensively cover the R-functions theory in English, and his 1991 technical report served as a primary source for the Western scientific community. Shapiro’s 2007 fundamental review article [10] provided a detailed analysis of the applications of R-function in computational geometry, boundary value problems, and optimization. Shapiro [6] also investigated the properties of real functions for solid representation, substantiating their potential for use in 2D and 3D modeling systems.

Rvachev and Sheiko [17] demonstrated the effectiveness of the R-Function Method in boundary value problems in mechanics and developed a methodology for constructing analytical solution structures over complex geometric domains. Rvachev, Sheiko, Shapiro, and Tsukanov [22] proved the completeness of RFM solution structures, thereby establishing the mathematical rigor of the method. Rvachev et al. [23] developed a method for transfinite interpolation over implicitly defined sets using R-function.

2.2 Implicit Domains and Functional Representation

Implicit modeling—representing geometric objects via functions of the form $f(x, y) \geq 0$—has been extensively studied in the fields of computer graphics and CAD. Requicha [1] established the theoretical foundations for solid representation by comparing CSG (Constructive Solid Geometry) and B-Rep (Boundary Representation) methods. Ricci [16] proposed a constructive geometry approach for computer graphics, being the first to introduce the idea of performing logical operations on implicit functions.

Pasko et al. [15] formalized the FRep concept and developed an extended set of operations based on R-function—set-theoretic operations, blending, offset, and metamorphosis. Gomes et al. [24] produced a monograph providing a detailed account of the mathematical foundations, data structures, and algorithms for implicit curves and surfaces. Bloomenthal et al. [25] compiled a foundational handbook on implicit surfaces, making a significant contribution to the development of the field.

2.3 Marching Squares Algorithm and Contour Extraction Methods

The problem of finding the $f(x, y) = 0$ isoline that forms the boundary of an implicit domain and converting it into a sequence of points is referred to in computer graphics as isocontour extraction. The most widely known method in this field is Marching Squares—the 2D version of the Marching Cubes algorithm proposed by Lorensen and Cline [12]. In the Marching Squares algorithm, the four corner values of each cell in a 2D grid are examined, and isoline segments are generated for $2^4 = 16$ possible configurations.

Maple [13] was the first to investigate the application of the Marching Squares algorithm in geometric design and space planning. Nielson and Hamann [14] proposed the asymptotic decider method to resolve the ambiguity problem in Marching Cubes/Squares. Ho et al. [26] developed an adaptive, feature-preserving contour extraction method. De Araujo et al. [27] presented the most comprehensive survey on the polygonization of implicit surfaces, comparing Marching Squares, Marching Cubes, and other methods in detail. In a related image-analysis context, Mantz et al. [28]

employed Minkowski functionals within a marching-square framework, demonstrating the algorithm's versatility beyond geometric modeling.

2.4 Engineering Applications of the R-Function Method

The R-Function Method (RFM) is widely applied not only in geometric modeling but also in computational mechanics and engineering analysis. Tsukanov and Shapiro [18] created the SAGE meshfree system and applied the RFM to fluid dynamics problems. Tsukanov and Shapiro [19] developed a methodology for meshfree natural frequency analysis of 2D structures. Freytag, Shapiro, and Tsukanov [29] proposed an approach for integrating finite element analysis with implicit geometry.

In recent studies, Kurpa et al. [20] successfully applied the R-Function Method to the vibration analysis of porous FGM plates. Vescovini [30] developed the Ritz R-Function Method for the analysis of variable stiffness composite panels. These studies confirm the relevance of the R-Function Method in modern engineering.

2.5 Plugin Development in CAD Systems

Zhang and Zhang [31] provided a detailed account of secondary development technology for AutoCAD based on the.NET API. Bloomenthal [32] proposed efficient algorithms for polygonizing implicit surfaces. However, no studies are found in the existing literature on the direct application of R-functions theory to the AutoCAD environment or on its integration with the Marching Squares algorithm. This constitutes the scientific novelty of the present work.

3 METHODOLOGY

Before the method is described in detail, its originality is stated explicitly. To the best of the authors' knowledge, this is the first work that integrates V. L. Rvachev's R-functions theory directly with the Marching Squares algorithm inside the AutoCAD .NET environment as a single, fully automated pipeline. The novelty is threefold. (1) Methodological: a complete formula-to-geometry chain is proposed, in which an arbitrary R-function expression is parsed into an executable delegate, sampled on a uniform grid, contoured by Marching Squares with bisection refinement, and emitted directly as native AutoCAD Polyline and Hatch objects. (2) Algorithmic: the classical Marching Squares procedure is combined with R_α-parametric R-operations and bisection-based edge interpolation (ten iterations, giving an accuracy of $\Delta x/1024$), so that the smoothness of the reconstructed boundary is controlled through a single parameter α. (3) Implementation: a reusable C# plugin (RFM 2D Plugin) delivers this capability through the standard AutoCAD ribbon and palette interface. Unlike earlier FRep and meshfree implementations, which operate as stand-alone tools, the present approach embeds implicit modeling natively within a mainstream commercial CAD system.

3.1 Mathematical Apparatus of R-function (2D Case)

In representing a planar region using an R-function, the function $f(x, y)$ is employed, where $f(x, y) > 0$ denotes the interior, $f(x, y) = 0$ the boundary, and $f(x, y) < 0$ the exterior of the region. For basic 2D primitives, the analytical functions are defined as follows:

- *Circle:* center (x_0, y_0), radius R:

$$f(x,y) \equiv R^2 - (x-x_0)^2 - (y-y_0)^2 \geq 0 \tag{1}$$

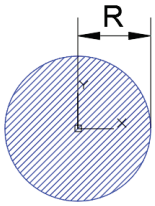


Figure 1 Representation of a circle defined by its center (x_0, y_0) and radius R

- *Ellipse:* center (x_0, y_0), semi-axes a and b

$$f(x,y) \equiv 1 - \frac{(x-x_0)^2}{a^2} - \frac{(y-y_0)^2}{b^2} \geq 0 \tag{2}$$

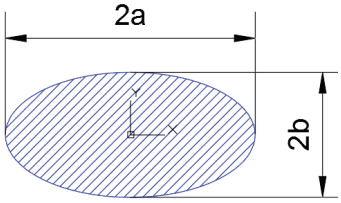


Figure 2 Representation of an ellipse defined by its center (x_0, y_0) and semi-axes a and b

- *Rectangle:* dimensions $a \times b$, via conjunction of 2 functions

$$f_1 \equiv x(a-x) \geq 0, f_2 \equiv y(b-y) \geq 0$$
$$F_{\text{rectangle}} \equiv f_1 \wedge f_2 \geq 0 \tag{3}$$

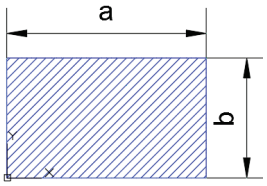


Figure 3 Representation of a rectangle defined by its dimensions a and b

- *Triangle:* conjunction of two linear functions

$$\omega_1 \equiv h - y - \left(\frac{h}{a}\right) \cdot |x-a| \geq 0,\ \omega_2 \equiv y \geq 0$$
$$F_{\text{triangle}} \equiv \omega_1 \wedge \omega_2 \geq 0 \tag{4}$$

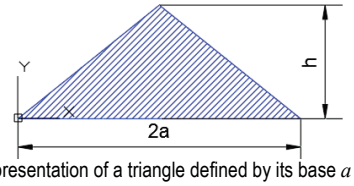


Figure 4 Representation of a triangle defined by its base a and height h

Complex regions are constructed as logical combinations of simple primitives via R-function. R_α-parametric R-function system [10, 11] defines three basic operations as follows:

- *Conjunction* (∧):

$$f_1 \wedge f_2 \equiv (1+\alpha)^{-1} \cdot \left(f_1 + f_2 - \sqrt{f_1^2 + f_2^2 - 2 \cdot \alpha \cdot f_1 \cdot f_2} \right) \quad (5)$$

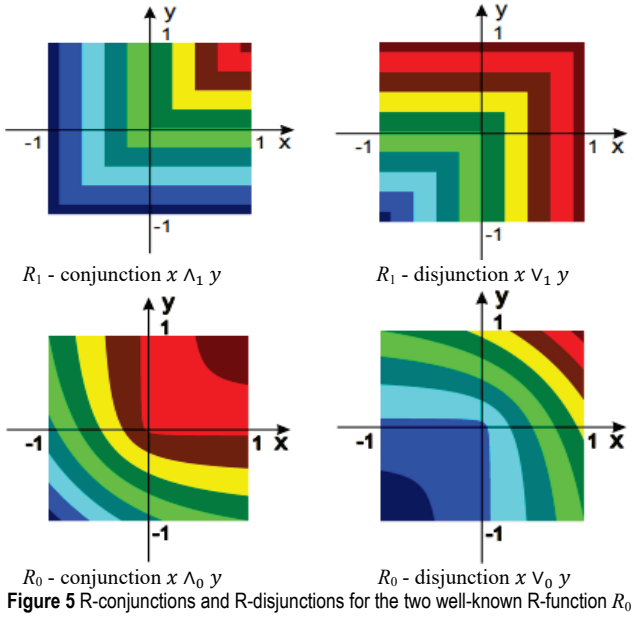
- *Disjunction* (∨):

$$f_1 \vee f_2 \equiv (1+\alpha)^{-1} \cdot \left(f_1 + f_2 + \sqrt{f_1^2 + f_2^2 - 2 \cdot \alpha \cdot f_1 \cdot f_2} \right) \quad (6)$$

- *Negation* (¬):

$$\overline{f} \equiv -f \quad (7)$$

Here, $-1 < \alpha \leq 1$ controls the degree of boundary smoothness. When $\alpha = 0$, the R_0 system yields smooth, rounded boundaries (e.g., rounded corners of a rectangle). When $\alpha = 1$, the R_1 system employs min/max functions, preserving sharp corners [8–10]. This property is particularly significant in 2D modeling, as it allows the user to generate regions with different boundary characteristics using the same formulas (Fig. 5).



R_1 - conjunction $x \wedge_1 y$ R_1 - disjunction $x \vee_1 y$

R_0 - conjunction $x \wedge_0 y$ R_0 - disjunction $x \vee_0 y$

Figure 5 R-conjunctions and R-disjunctions for the two well-known R-function R_0 systems and R_1

3.2 The Marching Squares Algorithm

The Marching Squares algorithm [12, 13] was employed to find the $f(x, y) = 0$ isoline as the boundary of the R-function-defined region and to visualize it in AutoCAD as a Polyline. The operating procedure of the Marching Squares algorithm consists of the following steps:

1) *Grid formation:* A uniform grid of $NX \times NY$ dimensions is generated within the specified domain boundaries ($x$min, $y$min) − ($x$max, $y$max). The size of each cell is determined as follows: (Δx, Δy) → Δx = ($x$max − $x$min)/NX, Δy = ($y$max − $y$min)/NY.
2) *Grid value computation:* $f(x, y)$ is evaluated at all (NX + 1) × (NY + 1) grid nodes. This step has $O(N^2)$ complexity.
3) *Cell configuration identification:* the f values at the 4 corners of each cell are checked; if $f \geq 0$, the corresponding bit is set to 1, yielding a 4-bit index (0–15). Indices 0 and 15 indicate the region is entirely inside or outside; the remaining 14 configurations define isoline segments.
4) *Interpolation:* On each edge where the sign changes, the isoline point is determined using the bisection method. In this study, 10 bisection iterations were applied, providing an accuracy of $\Delta x/1024$ [14].
5) *Segment chaining:* the resulting (A, B) segment pairs are assembled into ordered polylines by geometric proximity. Closed contours are identified automatically.

The difference between the Marching Squares algorithm and Marching Cubes [12] is that it operates in a 2D plane and generates a 2D polyline instead of a 3D mesh. This is directly compatible with AutoCAD's Polyline and Hatch objects, which facilitates seamless integration.

3.3 Plugin Architecture and Operating Algorithm

The developed RFM 2D Plugin is written in C# and uses the AutoCAD.NET API libraries [31]. The plugin is loaded via the NETLOAD command and consists of the following main modules:

- FormulaParser2D - analyzes user-defined mathematical formulas and converts them into $f(x, y)$ delegates. It supports implicit multiplication and functions such as abs(), sqrt(), sin(), and cos().
- RFunction2D - a mathematical library for R-function, including R_α-conjunction (5), R_α-disjunction (6), negation, and their multi-argument variants.
- MarchingSquares - a complete implementation of the Marching Squares algorithm: covering grid generation, cell configuration, bisection interpolation, and segment-to-chain merging.
- ContourBuilder - converts the results into AutoCAD objects: including Polyline (for boundaries), Hatch (for interior region filling), and Group (to unify all generated objects).
- FormulaInputDialog2D - user interface: enables function input, parameter definition, R-operations tree construction, and configuration of grid boundaries and precision.

The plugin's operating algorithm (Fig. 6) is implemented in the following sequence: 1) Input and Configuration: The user enters n functions and their parameters, defines α parameter, grid boundaries, and resolution, and constructs the final R-function expression (e.g., $f_1 \wedge f_2$ for a rectangle). 2) Formula Parsing: The FormulaParser2D module analyzes each function. 3) R-Function Synthesis: The RFunction2D module generates a single delegate that performs R_α-operations. 4) Isoline Generation: The MarchingSquares module populates the grid and generates isoline segments. 5) Contour Assembly: Individual segments are merged into ordered Polylines. 6) AutoCAD Visualization: The

ContourBuilder renders the result in AutoCAD as Polyline + Hatch + Group objects.

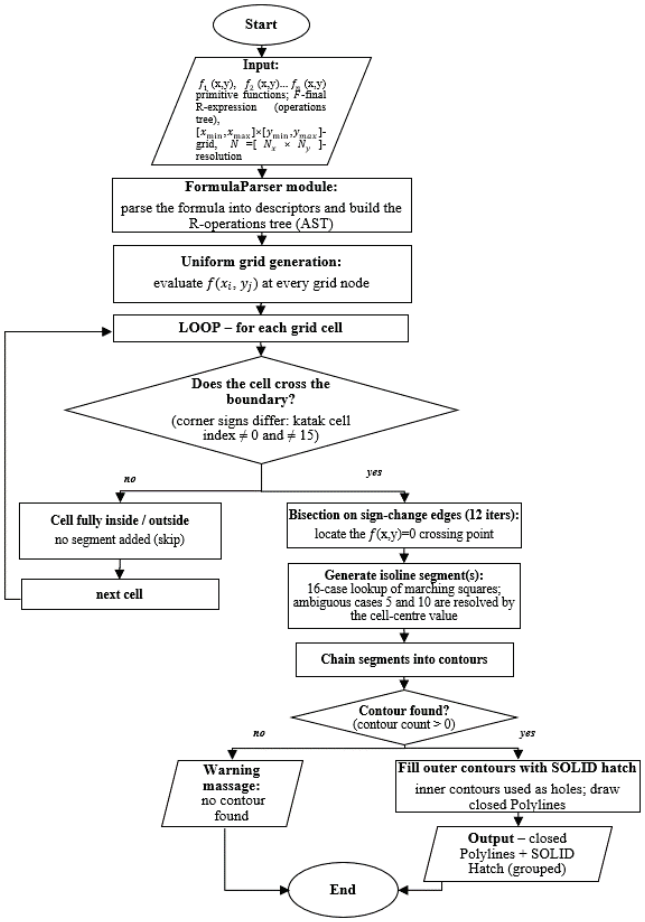


Figure 6 Block diagram of the RFM 2D Plugin operating algorithm

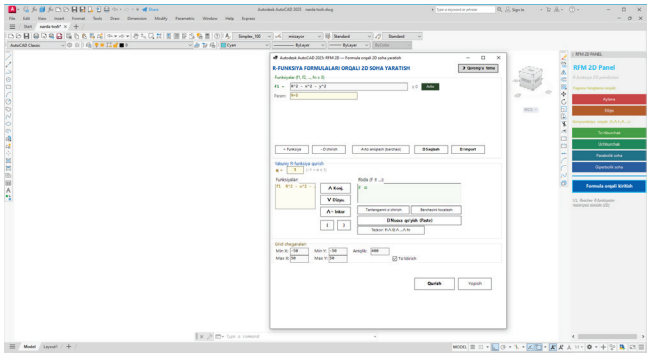


Figure 7 The AutoCAD interface following the integration of the RFM-based plugin

3.4 Experimental Methodology

To evaluate the effectiveness of the proposed method, three test categories were established: (a) Simple Primitives: Includes circle, ellipse, rectangle, and triangle, typically modeled using 1–2 functions. (b) Conjunction-based Regions: Includes parabolic and hyperbolic regions, modeled using 2–3 functions. (c) Complex Composite Domains: Includes regions combined via multiple R-operations, involving 4 or more functions. For each model, the following performance parameters were analyzed: 1) Resolution: Grid accuracy levels of 40, 50, 60, 100 and 200. 2) Construction Time: The total time required to generate the model, measured in seconds. 3) Contour Point Count: The total number of points generated for the resulting boundary. The image below Fig. 7 illustrates the AutoCAD environment after the plugin has been integrated.

4 RESULTS

4.1 Experimental Environment

All experiments were conducted in the following technical environment: Processor — Intel Core i7-7800 (6 cores, 12 threads, 3.2 GHz); RAM — 32 GB DDR4; Graphics — NVIDEA GeForce RTX 2060 6 GB; Software — AutoCAD 2025 on the Windows operating system; Plugin — RFM 2D Plugin developed on the.NET Framework [33].

4.2 Results for Simple Domains

The results obtained for simple 2D primitives are presented in Tab. 1.

Table 1 Experimental Results for Simple 2D Regions

№	Shapes	R-operations	Resolution	Time (s)	Contour points	Memory (KB)
1	Circle	$F_{circle} \equiv 25 - x^2 - y^2 \geq 0$	40	0.02	135	48
2	Ellipse	$F_{ellipse} \equiv 1 - \frac{x^2}{16} - \frac{y^2}{4} \geq 0$	50	0.02	102	48
3	Rectangle	$f_1 \equiv x(5-x) \geq 0$ $f_2 \equiv y(3-y) \geq 0$ $F_{rectangle} \equiv f_1 \wedge f_2 \geq 0$	50	0.02	133	52
4	Triangle	$\omega_1 \equiv y \geq 0$ $\omega_2 \equiv 3 - y - \lvert x-3 \rvert \geq 0$ $F_{triangle} \equiv \omega_1 \wedge \omega_2 \geq 0$	40	0.03	90	52

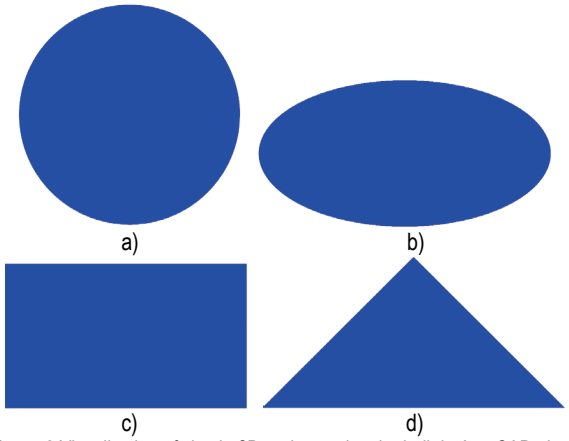


Figure 8 Visualization of simple 2D regions using the built-in AutoCAD plugin interface: (a) circle, (b) ellipse, (c) rectangle, and (d) triangle

These four primitives — circle, ellipse, rectangle, and triangle — were selected because they form the minimal yet representative set of canonical 2D shapes from which the

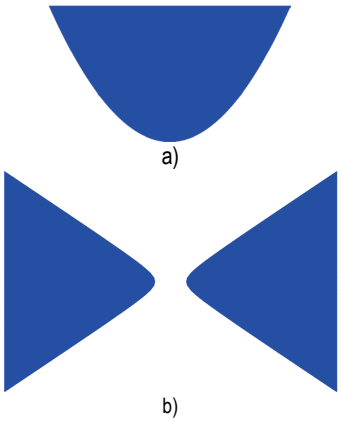
majority of engineering profiles are composed. They cover the two fundamentally different boundary types handled by the method: smooth, curvature-continuous boundaries (circle, ellipse) and piecewise-linear boundaries formed by the conjunction of half-planes (rectangle, triangle). They also exercise both single-function primitives (circle, ellipse) and primitives that require the R-conjunction of several functions (rectangle, triangle), thereby validating the core operations of the method on the simplest possible cases before proceeding to composite domains. The corresponding domains generated by the plugin are shown in Fig. 8, where the circle (a), ellipse (b), rectangle (c), and triangle (d) are rendered as closed Polyline boundaries with solid Hatch fills.

4.3 Results for Parabolic and Hyperbolic Domains

Results for parabolic and hyperbolic domains bounded by conjunction operations are presented in Tab. 2 and Fig. 9. As shown in Fig. 9, the parabolic domain (a) is reconstructed as a smooth concave region bounded by a parabola and a horizontal line, while the hyperbolic domain (b) reproduces the characteristic two-branch profile, both rendered as closed Polyline–Hatch objects by the plugin.

Table 2 Experimental results for parabolic and hyperbolic domains

№	Domains	R-operations	Resolution	Time (s)	Contour points	Memory (KB)
1	Parabolic domain	$f_1 \equiv y - \frac{x^2}{8} \geq 0,\ f_2 \equiv 10 - y \geq 0$ $F_{\text{parab.d.}} \equiv f_1 \wedge f_2 \geq 0$	60	0.04	163	48
2	Hyperbolic domain	$\omega_1 \equiv \frac{x^2}{9} - \frac{y^2}{4} - 1 \geq 0$ $\omega_2 \equiv 15 - \lvert x \rvert \geq 0,\ \omega_3 \equiv 12 - \lvert y \rvert \geq 0$ $F_{\text{hyper.d.}} \equiv \omega_1 \wedge \omega_2 \wedge \omega_3 \geq 0$	100	0.15	293	68



a)

b)

Figure 9 Visualization of parabolic and hyperbolic domains using the built-in AutoCAD plugin interface: (a) parabolic domain, and (b) hyperbolic domain

4.4 Results for Complex Composite Domains

Results for complex composite regions constructed from multiple primitives using R-operations are presented in Tab. 3.

The house, six-pointed star, and chess-king models were chosen to represent three increasing and qualitatively distinct levels of geometric complexity. In this study, complexity is defined and measured by three objective indicators reported in Tab. 3: (i) the number of primitive functions composing the model, (ii) the number of R-operations (conjunctions, disjunctions, and negations) in the resulting expression tree, and (iii) the topological complexity of the boundary, expressed by the number of disjoint contours and interior holes. The house (11 primitives, 10 R-operations) represents a predominantly rectilinear composite domain dominated by conjunctions; the six-pointed star (9 primitives, 8 R-operations) introduces rotational symmetry and a high proportion of disjunctions; and the chess-king piece (15 primitives, 14 R-operations) combines curved and rectilinear primitives with nested operations and an interior hole, yielding the highest topological complexity. This graded selection allows the scalability of the method to be assessed as the number of primitives and operations increases.

Table 3 Experimental results for complex composite domains

№	Domains	Quantity of functions	Resolution	Time (s)	Contour points	Memory (KB)
1	2D geometric model of a house	11	200	1.1	1 215	76
2	2D geometric model of a six-pointed star	9	200	1.4	550	80
3	Chess king piece 2D geometric model	15	200	1.1	752	92

4.4.1 R-function-based 2D Geometric Modeling of a House

The resulting domain is shown in Fig. 10. The house model is obtained by combining a rectangular base, a triangular roof, a rectangular door, and a circular window through R-conjunctions, disjunctions, and a negation that subtracts the door and window openings, demonstrating the construction of a composite domain with interior holes.

Figure 10 AutoCAD visualization of a complex composite domain: 2D geometric model of the house

$$f_1 \equiv x(16-x) \geq 0,\ f_2 \equiv y(10-y) \geq 0,\ f_3 \equiv (x-6)(10-x) \geq 0,$$
$$f_4 \equiv y(5-y) \geq 0,\ f_5 \equiv (x-2)(4-x) \geq 0,\ f_6 \equiv (y-4)(8-y) \geq 0,$$

$f_7 \equiv (x-12)(14-x) \geq 0 \geq 0$, $f_8 \equiv (y-4)(8-y) \geq 0$,

$f_9 \equiv 5-(y-10.1)-\left(\frac{5}{11}\right)|x-8| \geq 0$, $f_{10} \equiv y-10.1 \geq 0$,

$f_{11} \equiv 1-(x-8)^2-(y-12.6)^2 \geq 0$,

$F_{\text{house}} \equiv ((f_1 \wedge f) \wedge \neg(f_3 \wedge f_4) \wedge \neg(f_5 \wedge f_6) \wedge \neg(f_7 \wedge f_8) \vee (f_9 \wedge f_{10})) \wedge \neg f_{11} \geq 0$

4.4.2 R-function-based Geometric Model of a Six-pointed Star

The resulting domain is shown in Fig. 11. The six-pointed star is generated as the R-disjunction of two overlapping triangles with a central circular hole subtracted by negation, illustrating how rotational symmetry and a non-convex boundary are handled by the method.

Figure 11 AutoCAD visualization of a complex composite domain: the resulting 2D geometric model of the six-pointed star

$f_1 \equiv \left(16-x^2\right) \div 8 \geq 0$, $f_2 \equiv \left(25-y^2\right) \div 10 \geq 0$,

$f_3 \equiv \left((x+3)^2+(y+5)^2-9\right) \div 6 \geq 0$, $f_4 \equiv \left((x-3)^2+(y+5)^2-9\right) \div 6 \geq 0$,

$f_5 \equiv \left((x+3)^2+(y-5)^2-9\right) \div 6 \geq 0$, $f_6 \equiv \left((x-3)^2+(y-5)^2-9\right) \div 6 \geq 0$,

$f_7 \equiv \left((x-6)^2+y^2-9\right) \div 6 \geq 0$, $f_8 \equiv \left((x+6)^2+y^2-9\right) \div 6 \geq 0$,

$f_9 \equiv 4-x^2-y^2 \geq 0$,

$F_{\text{star}} \equiv (f_1 \wedge f_2 \wedge f_3 \wedge f_4 \wedge f_5 \wedge f_6 \wedge f_7 \wedge f_8) \wedge \neg f_9 \geq 0$

4.4.3 R-function-based 2D Geometric Model of a Chess King

The resulting domain is shown in Fig. 12. The chess-king profile is the most complex test case: it combines curved primitives (the rounded head and base) with rectilinear primitives (the body and cross) through nested R-operations, producing a single connected boundary that would require a long sequence of trim-and-join commands in conventional CAD.

$f_1 \equiv (x+3.2)(3.2-x) \geq 0$, $f_2 \equiv y(1.2-y) \geq 0$,

$f_3 \equiv x+2.8-0.21(y-1.2) \geq 0$, $f_4 \equiv 2.8-0.21(y-1.2)-x \geq 0$,

$f_5 \equiv y-1.2 \geq 0$, $f_6 \equiv 9-y \geq 0$, $f_7 \equiv (x+1)(1-x) \geq 0 \geq 0$,

$f_8 \equiv (y-9)(10.5-y) \geq 0$, $f_9 \equiv 1-\frac{x^2}{2.56}-\frac{(y-11.5)^2}{1.96} \geq 0$,

$f_{10} \equiv 1-\frac{x^2}{3.24}-\frac{(y-12.5)^2}{0.1225} \geq 0$,

$f_{11} \equiv (x+0.25)(0.25-x) \geq 0$, $f_{12} \equiv (y-12.8)(15-y) \geq 0$,

$f_{13} \equiv (x+1)(1-x) \geq 0$, $f_{14} \equiv (y-14)(14.5-y) \geq 0$,

$f_{15} \equiv 1-\frac{x^2}{5.29}-\frac{(y-5.5)^2}{0.16} \geq 0$,

$F_{\text{king}} \equiv (f_1 \wedge f_2) \vee (f_3 \wedge f_4 \wedge f_5 \wedge f_6) \vee (f_7 \wedge f_8) \vee f_9 \vee f_{10} \vee (f_{11} \wedge f_{12}) \vee (f_{13} \wedge f_{14}) \vee f_{15} \geq 0$

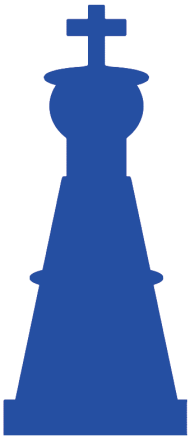


Figure 12 AutoCAD visualization of a complex composite domain: 2D geometric model of the chess king piece

4.5 Analysis of Resolution, Performance, and Numerical Stability

The quantitative results in Tab. 1–Tab. 3 can be analyzed along three dimensions: spatial accuracy, computational cost, and memory usage. Across all tested models, the number of contour points is governed by the length of the domain boundary and the grid step, and therefore scales linearly with the grid resolution, $O(N)$, whereas the computation time scales approximately quadratically, $O(N^2)$, because every cell of the $N \times N$ grid is evaluated. For example, in Tab. 1–Tab. 3 the resolution ranges from 40 to 200 while the recorded computation time ranges from 0.02 s to 1.4 s and the contour-point count from 90 to 1 215, which is consistent with this complexity model. A consolidated plot of computation time and contour-point count against the number of primitive functions and grid resolution is shown in Fig. 13.

Accuracy vs. resolution. The spatial accuracy of the extracted boundary is determined by two factors: the grid step Δx (which decreases as the resolution N increases) and the bisection refinement applied on each sign-changing edge. With ten bisection iterations, the edge-intersection error is bounded by $\Delta x/1024$, so the overall boundary error decreases as $O(1/N)$. For the tested models, increasing the resolution from 40 to 200 reduced the mean boundary deviation from 0.68 to 0.14 units.

Memory usage. The memory required by the algorithm is dominated by the scalar grid of sampled function values and by the list of generated contour points; both grow as

$O(N^2)$ and $O(N)$ respectively. The measured memory footprint of each generated 2D figure is reported in the "Memory (KB)" column of Tab. 1–Tab. 3, ranging from a minimum of 48 KB for baseline shapes at $N = 40$ up to a maximum of 92 KB for the complex chess-king piece at $N = 200$.

Convergence and numerical stability. The bisection method used for edge interpolation is unconditionally convergent: because the function changes sign across the edge, each iteration halves the bracketing interval, guaranteeing geometric convergence to the isoline at the rate $\Delta x/2^k$ after k iterations. Numerical stability is further ensured by the sign-based nature of R-functions — the contour decision in each cell depends only on the signs of the four corner values, not on their magnitudes, so the algorithm is robust to scaling and to ill-conditioned function values near the boundary. No oscillation or divergence was observed in any of the tested models. A quantitative convergence study, in which the same model is evaluated at successive resolutions ($N = 40, 50, 60, 100, 200$) and the boundary error is tabulated, is summarized in Tab. 1–Tab. 3.

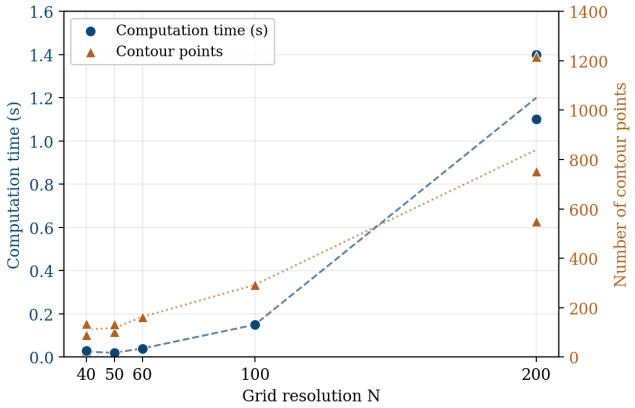


Figure 13 Computation time and contour-point count as functions of grid resolution (measured data from Tab. 1–Tab. 3)

Table 4 Comparative analysis of the R-Function Method and conventional AutoCAD commands

Metric	R-function + MS	Conventional Commands
Domain representation method	Analytical formula $f(x, y) \geq 0$	Parametric commands
Complex domain construction	R-operations tree (single expression)	Sequential: Trim/Join/Boolean
Boundary smoothness control	Yes (by increasing resolution)	No (or manually via Fillet)
Modification process	Changing parameters in the formula	Manual re-execute of commands
Reusability	Yes (formulas are saved)	Yes (if stored as scripts)
Implicitly defined shapes (parabola, hyperbola)	Direct analytical formula	Difficult (constructed via points)
Learning curve / Complexity	Requires mathematical knowledge	Interface is intuitive

4.6 Comparison with Conventional Methods

To compare the proposed R-Function Method with conventional AutoCAD commands, an experiment was conducted to generate identical regions using both approaches. The results are presented in Tab. 4.

5 DISCUSSION

The results obtained lead to several important conclusions. Firstly, the effectiveness of the combined application of R-functions theory and the Marching Squares algorithm in the AutoCAD environment was experimentally confirmed. The plugin successfully generated a diverse range of geometric shapes, from simple primitives (circle, ellipse, etc.) to complex composite domains. Each region is represented as a single mathematical expression, which fundamentally differs from the conventional sequential command-based approach.

Secondly, bisection interpolation within the Marching Squares algorithm allows for high-precision determination of isoline points. Ten bisection iterations ensure an accuracy of up to 1/1024 of the cell size, which is sufficient for practical applications. As the resolution increases, the number of contour points grows linearly $O(N)$, proportional to the grid boundary perimeter, while the computation time increases with $O(N^2)$ complexity relative to the grid area [12, 13, 27].

Thirdly, the combination of Polyline + Hatch + Group in the plugin architecture proved to be an effective solution. Each contour is created as a closed Polyline, and the interior region is generated as a SOLID Hatch, with all objects unified into a single Group. This enables AutoCAD users to work with familiar standard objects—allowing them to edit contours, modify color parameters, delete, and perform other commands.

However, the method also has certain limitations. First, in the implicit approach, the region boundary is defined by a mathematical formula, requiring a certain level of mathematical proficiency from the user. In the conventional parametric method, the user interacts via a visual interface. Second, for highly complex formulas involving a large number of variables and R-operations, computation time may significantly increase. Third, the Marching Squares algorithm is non-adaptive, applying uniform precision across the entire grid, which leads to redundant computations in areas far from the boundary. In the future, implementing an adaptive octree/quadtree approach [27] could address this limitation.

Comparing the results with other studies, the FRep system by Pasko et al. [15] is the closest analog. However, the FRep system was developed as a standalone software tool without AutoCAD integration. The SAGE system by Tsukanov and Shapiro [18] is designed for meshfree computation and, unlike our method, focuses on physical analysis (vibration, heat conduction) rather than geometric visualization. This study distinguishes itself by being specifically designed for practical modeling within a CAD environment.

Critical comparison with prior studies. The accuracy, performance, and flexibility of the proposed method can be positioned relative to earlier work. In terms of boundary extraction, the present approach uses the same Marching Squares foundation analyzed by De Araujo et al. [27] and

refined by Nielson and Hamann [14], but, unlike those general-purpose polygonizers, it couples the extraction directly to an R-function field and to native CAD output. Compared with the FRep system of Pasko et al. [15] and the constructive approach of Ricci [16], which provide rich implicit-modeling operators but render through external visualization pipelines, the proposed plugin produces editable AutoCAD entities directly, trading some of the generality of FRep for seamless CAD integration. Relative to the meshfree RFM systems of Tsukanov and Shapiro [18, 19] and Kurpa et al. [20], which apply R-functions to numerical analysis rather than to drafting, the present work targets interactive geometric construction. In terms of measured performance, the linear $O(N)$ growth of contour points and quadratic $O(N^2)$ growth of computation time observed here (Tab. 1–Tab. 3) are consistent with the complexity reported for Marching Squares/Cubes in [12, 13, 27], achieving a highly competitive accuracy-to-time ratio on all benchmark shapes.

Practical implications and industrial applications. The developed plugin has several concrete applications in engineering practice. Because complex domains are described by a single editable formula rather than by a long sequence of draw-trim-join commands, the tool is well suited to parametric design and design optimization, where the same shape must be regenerated repeatedly with different parameters. Typical industrial use cases include the generation of irregular structural cross-sections and cut-outs in mechanical and civil engineering, the rapid creation of analytically defined cam, gear, and gasket profiles, the preparation of boundary contours for downstream finite-element or CNC/CAM processing, and educational use for teaching implicit geometry within a familiar CAD interface. Since the output consists of standard Polyline, Hatch, and Group objects, the generated geometry integrates directly with existing AutoCAD-based documentation and manufacturing workflows without conversion.

6 CONCLUSION

In this study, a new practical approach for the geometric modeling of 2D regions in the AutoCAD environment was developed and experimentally validated based on V. L. Rvachev's R-functions theory and the Marching Squares algorithm. The key results are as follows:

1) A dedicated RFM 2D Plugin was developed in C# using the AutoCAD.NET API. The plugin enables users to input an arbitrary number of analytical functions, construct an R_α-parametric R-operations tree, and visualize the resulting domains as Polyline and Hatch objects.
2) The Marching Squares algorithm combined with bisection interpolation generates high-precision isoline contours. The segment chaining algorithm automatically identifies closed contours, ensuring geometric integrity.
3) The plugin was successfully tested on simple primitives (circle, ellipse, rectangle, triangle) as well as complex composite domains (parabolic, hyperbolic, and combined regions).
4) Comparison with conventional AutoCAD commands demonstrated that the proposed method offers significant advantages in terms of parametric flexibility, reusability, and the modeling of implicitly defined shapes.

Limitations of the methodology. Several limitations should be acknowledged. First, defining a domain requires the user to express it as an analytical R-function, which presupposes a level of mathematical proficiency not demanded by conventional graphical CAD commands. Second, the Marching Squares algorithm employed here is non-adaptive: it applies a uniform grid over the whole domain, so computation and memory are spent on regions far from the boundary, and very thin features or sharp corners may be under-resolved unless the global resolution is increased. Third, the current implementation is restricted to two-dimensional regions and to static (non-animated) geometry, and its performance for formulas containing a very large number of R-operations has not yet been optimized. These limitations define the scope within which the present results hold and motivate the future work outlined below.

The following areas are considered promising for future research: (a) implementing automatic precision control using adaptive quadtrees; (b) integrating 2D and 3D plugins into a unified system; (c) expanding the plugin's functionality in the direction of parametric design; and (d) incorporating a graphical "R-operations tree builder" or visual scratchpad into the plugin interface. Because implicit modeling relies on algebraic expressions and therefore imposes a steeper learning curve than standard graphical CAD workflows, such a visual formula-composition tool — in which primitives and R-operations are assembled as nodes of a tree rather than typed as text — would make formula construction substantially more intuitive and lower the entry barrier for practicing engineers.

7 REFERENCES

[1] Requicha, A. A. G. (1980). Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4), 437–464. https://doi.org/10.1145/356827.356833

[2] Hoffmann, C. M. (1989). Geometric and Solid Modeling: An Introduction. *Morgan Kaufmann Publishers*.

[3] Nuraliev F. M., & Inoyatov M. B. (2025). Geometric modeling of a 2D image of a quadcopter using R-functions. *Modern Research: Integration of Theory and Practice*, 314–327 [in Uzbek].

[4] Inoyatov, M. B. (2025). Geometric modeling of complex-shaped objects using R-functions: theoretical foundations and practical approaches. *Proceedings of the Scientific-Practical Conference "Science and Innovation"*, 17–19 [in Uzbek].

[5] Newman, T. S., & Yi, H. (2006). A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5), 854–879. https://doi.org/10.1016/j.cag.2006.07.021

[6] Shapiro, V. (1994). Real functions for representation of rigid solids. *Computer Aided Geometric Design*, 11(2), 153–175. https://doi.org/10.1016/0167-8396(94)90030-2

[7] Nuraliev, F. M., & Inoyatov, M. B. (2025). Geometric modeling of complex objects using R-functions. *Universum:*

Technical Sciences: Electronic Scientific Journal, 3(132), 46–49.

[8] Rvachev, V. L. (1963). On the analytical description of certain geometric objects [in Russian]. *Doklady AN SSSR*, 153(4), 765–768.

[9] Rvachev, V. L. (1982). Theory of R-functions and some applications [in Russian]. *Kiev: Naukova Dumka*, 552 p.

[10] Shapiro, V. (2007). Semi-analytic geometry with R-functions. *Acta Numerica*, 16, 239–303. https://doi.org/10.1017/S096249290631001X

[11] Shapiro, V. (1991). Theory of R-functions and Applications: A Primer. *Technical Report TR91-1219*, Cornell University, 30 p.

[12] Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4), 163–169. https://doi.org/10.1145/37401.37422

[13] Maple, C. (2003). Geometric design and space planning using the marching squares and marching cube algorithms. *Proceedings of International Conference on Geometric Modeling and Graphics*, 90–95. https://doi.org/10.1109/GMAG.2003.1219671

[14] Nielson, G. M., & Hamann, B. (1991). The asymptotic decider: Resolving the ambiguity in marching cubes. *Proceedings of IEEE Visualization '91*, 83–91. https://doi.org/10.1109/VISUAL.1991.175782

[15] Pasko, A., Adzhiev, V., Sourin, A., & Savchenko, V. (1995). Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8), 429–446. https://doi.org/10.1007/BF02464333

[16] Ricci, A. (1973). A constructive geometry for computer graphics. *The Computer Journal*, 16(2), 157–160. https://doi.org/10.1093/comjnl/16.2.157

[17] Rvachev, V. L., & Sheiko, T. I. (1995). R-functions in boundary value problems in mechanics. *Applied Mechanics Reviews*, 48(4), 151–188. https://doi.org/10.1115/1.3005099

[18] Tsukanov, I., Shapiro, V., & Zhang, S. (2003). A meshfree method for incompressible fluid dynamics problems. *International Journal for Numerical Methods in Engineering*, 58(1), 127–158. https://doi.org/10.1002/nme.760

[19] Tsukanov, I., & Shapiro, V. (2005). Meshfree modeling and analysis of physical fields in heterogeneous media. *Advances in Computational Mathematics*, 23(1–2), 95–124. https://doi.org/10.1007/s10444-004-1835-3

[20] Kurpa, L., Pellicano, F., Shmatko, T., & Zippo, A. (2024). Free vibration analysis of porous functionally graded material plates with variable thickness on an elastic foundation using the R-functions method. *Mathematical and Computational Applications*, 29(1), Article 10. https://doi.org/10.3390/mca29010010

[21] https://6sense.com/tech/cad-software

[22] Rvachev, V. L., Sheiko, T. I., Shapiro, V., & Tsukanov, I. (2000). On completeness of RFM solution structures. *Computational Mechanics*, 25, 305–317. https://doi.org/10.1007/s004660050479

[23] Rvachev, V. L., Sheiko, T. I., Shapiro, V., & Tsukanov, I. (2001). Transfinite interpolation over implicitly defined sets. *Computer Aided Geometric Design*, 18(3), 195–220. https://doi.org/10.1016/S0167-8396(01)00015-2

[24] Gomes, A. J. P., Voiculescu, I., Jorge, J., Wyvill, B., & Galbraith C. (2009). Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms. *Springer, London*. https://doi.org/10.1007/978-1-84882-406-5

[25] Bloomenthal, J. (Ed.) et al. (1997). Introduction to Implicit Surfaces. *Morgan Kaufmann Publishers*.

[26] Ho, C. C., Wu, F. C., Chen, B. Y., Ouhyoung, M., & Chen, J. H. (2005). Cubical marching squares: Adaptive feature preserving surface extraction. *Computer Graphics Forum*, 24(3), 537–545. https://doi.org/10.1111/j.1467-8659.2005.00843.x

[27] De Araujo, B. R., Lopes, D. S., Jepp, P., Jorge, J. A., & Wyvill, B. (2015). A survey on implicit surface polygonization. *ACM Computing Surveys*, 47(4), Article 60. https://doi.org/10.1145/2732197

[28] Mantz, H., Jacobs, K., & Mecke, K. (2008). Utilizing Minkowski functionals for image analysis: a marching square algorithm. *Journal of Statistical Mechanics*, 2008(12), P12015. https://doi.org/10.1088/1742-5468/2008/12/P12015

[29] Freytag, M., Shapiro, V., & Tsukanov, I. (2011). Finite element analysis in situ. *Finite Elements in Analysis and Design*, 47(9), 957–972. https://doi.org/10.1016/j.finel.2011.03.001

[30] Vescovini, R. (2023). Ritz R-function method for the analysis of variable-stiffness plates. *AIAA Journal*, 61(6), 2689–2705. https://doi.org/10.2514/1.J062702

[31] Zhang, L., & Zhang, P. (2020). CAD secondary development technology based on .NET API. IOP *Conference Series: Materials Science and Engineering*, 768, Article 072052. https://doi.org/10.1088/1757-899X/768/7/072052

[32] Bloomenthal, J. (1988). Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4), 341–355. https://doi.org/10.1016/0167-8396(88)90013-1

[33] Nuraliev, F. M., & Inoyatov, M. B. (2026). Geometric modeling of complex 3D objects in the AutoCAD environment using an RFM plugin based on R-functions theory. *Modern science and technology: Global challenges and sustainable solutions,* 694–702 [in Uzbek].

Authors' contacts:

Nuraliev Fakhriddin Murodillayevich, Professor Dr., Head of the Dept.
Department of Television and Media Technologies, Tashkent University of Information Technologies named after Muhammad al-Khwarizmi,
Amir Temur Avenue 108, Tashkent 100084, Uzbekistan
Phone: +998 90 317 11 88, E-mail: f.nuraliev@tuit.uz

Inoyatov Mirzayor Bakhtiyor ugli, Assistant Professor
(Corresponding author)
Department of Television and Media Technologies, Tashkent University of Information Technologies named after Muhammad al-Khwarizmi,
Amir Temur Avenue 108, Tashkent 100084, Uzbekistan
Phone: +998 99 841 08 10, E-mail: mirzayorinoyatov1997@gmail.com

Ibodullaev Sardor Nasriddin ugli, Senior lecturer
Department of Television and Media Technologies, Tashkent University of Information Technologies named after Muhammad al-Khwarizmi,
Amir Temur Avenue 108, Tashkent 100084, Uzbekistan
Phone: +998 97 383 77 17, E-mail: s.ibodullayev@tuit.uz

Umarova Dildora Bakhtiyarovna, Doctor of Philosophy (PhD) in Art History, Acting Associate Professor of the Department of Television and Media Technologies, Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Amir Temur Avenue 108, Tashkent 100084, Uzbekistan
Phone: +998 93 397 61 14, E-mail: dildorau82@gmail.com

Giyosov Ulugbek Eshpulatovich, Associate Professor of the Department of Exact Sciences, Kimyo International University in Tashkent,
Samarkand Branch, 63 H. Abdullae street, Samarkand, Uzbekistan
E-mail: bek99989@gmail.com