Gulf Journal of Mathematics
Vol 21, Issue 1 (2025) 439-456

ISSN: 2309-4966
https://doi.org/10.56947/gjom.v21i1.3506

# ON THE APPROXIMATION OF SOLUTION REGIONS FOR NONLINEAR INEQUALITIES

NOUR EL ISLAM HIBER[1*], DJAOUIDA GUETTAL[2] and MOHAMED RAHAL[3]

Abstract. In this paper, we present an algorithm for solving nonlinear inequalities involving a Lipschitz function $f$. The algorithm builds upon the covering principles of global optimization, drawing from existing works, which are based on adaptively constructing lower and upper bounds for $f$ over a given interval. Focusing on these bounds, the interval is classified: discarded if $f$ is strictly positive, accepted as part of the solution region if $f$ is non-positive, or subdivided if the sign of $f$ is indeterminate. For the third case, we suggest a partition of the interval into three equal sub-intervals. Numerical experiments demonstrate promising results compared with existing works employing different techniques.

*Keywords.* Lipschitz functions, global optimization, covering method, nonlinear inequalities, lower and upper bounds.
*2020 Mathematics Subject Classification.* Primary 65K10, 90C30; Secondary 90C26, 90C47

## 1. Introduction

Inequalities play a fundamental role in various real-world problems, particularly in the modeling of constraints for optimization tasks. Solving nonlinear inequalities is of considerable importance today due to its wide range of applications [6, 7, 11], especially in fields such as artificial intelligence and robotics [13].

Previous works, such as those by Lera et al. [13] and Evtushenko et al. [2, 3, 4](e.g., the sequential covering algorithm, see [1, 14]), have primarily focused on systems involving multivariate functions. In this work, however, we restrict our attention to the univariate case and propose several methodological improvements inspired by their foundational approaches.

We consider the problem of finding the solution set of inequalities given by:

$$f(x) \leq 0, \quad x \in [a, b] \tag{1.1}$$

*Date*: Received: Sep 6, 2025; Accepted: Oct 4, 2025.
* Corresponding author

where $f$ is a real-valued function defined on the interval $[a, b]$ and satisfies the Lipschitz condition:

$$\forall (x, y) \in [a, b]^2, \quad |f(x) - f(y)| \leq L|x - y|, \tag{1.2}$$

with $0 < L < +\infty$ being the Lipschitz constant. Importantly, $f$ is not assumed to be convex or differentiable. Our objective is to find an approximate set of solutions for problem (1.1).

To achieve this, we utilize a covering approach where the initial interval $D = [a, b]$ is recursively partitioned into smaller sub-intervals $D_i = [x_{i-1}, x_i]$. For each sub-interval $D_i$, our algorithm evaluates the potential sign of $f(x)$ and classifies it based on its relationship to the solution set. We define the following classifications for any given sub-interval $D_i$:

$$\begin{aligned} S_{\text{sol}} &= \{x \in D_i : f(x) < 0\}, \\ S_{\text{non-sol}} &= \{x \in D_i : f(x) > 0\}, \\ S_{\text{zeros}} &= \{x \in D_i : f(x) = 0\}. \end{aligned}$$

Our method is uniquely characterized by its adaptive multi-part subdivision. Specifically, in cases where the behavior of $f(x)$ within $D_i$ remains uncertain (i.e., it cannot be definitively classified into $S_{\text{sol}}$ or $S_{\text{non-sol}}$), we divide $D_i$ into three equal sub-intervals. This three-way splitting strategy distinguishes our approach from common binary partitioning methods.

Furthermore, we emphasize that our algorithm leverages both approximate upper and lower Lipschitz bounds to effectively characterize the function $f$ within each sub-interval.

This work addresses several limitations present in existing methods:

- Many algorithms for nonlinear inequalities rely on assumptions of differentiability or convexity, which are often unrealistic in practical applications. Our method avoids these restrictive assumptions.
- Most covering techniques commonly employ binary partitioning (e.g., as in [1, 2, 3, 4]). Adaptive strategies using multi-part subdivisions, such as our proposed three-way splitting, remain underexplored despite their potential for improved efficiency.
- Few recursive algorithms have been developed that simultaneously utilize both upper and lower Lipschitz bounds to rigorously classify domain intervals while ensuring guaranteed convergence for general non-convex, non-differentiable functions [4, 11, 12, 13].

To address these challenges, we propose a Lipschitz-based interval covering algorithm [1, 8, 9, 14]. This algorithm uses affine underestimators and overestimators of the function $f$ within each sub-interval $D_i$. It then classifies these sub-intervals based on their relation to the set of zeros and recursively refines uncertain regions until a specified stopping criterion is satisfied.

The paper is organized as follows: Section 2 details the construction of the affine underestimators and overestimators for the univariate function $f$, derived directly from the Lipschitz condition (1.2). Based on these bounds, the initial

domain of problem (1.1) is divided into sub-domains according to specific classification conditions. Section 3 presents the developed algorithm along with a detailed analysis of its convergence behavior. Section 4 is dedicated to numerical experiments, focusing on visualizing the solution sets of various inequalities and providing a comparative performance analysis of our algorithm against benchmark methods using graphical representations. The paper concludes with Section 5, which summarizes the findings and provides concluding remarks.

## 2. Support functions that underestimate or overestimate the objective function

In this section, our primary objective is to construct affine support functions that provide lower (underestimator) and upper (overestimator) bounds for the Lipschitz continuous function $f$. These bounds are crucial for classifying sub-intervals of the domain according to criteria essential for our algorithm [3, 4, 12, 13] .

**Definition 2.1** (Underestimator)**.** A function $\ell$ is defined as an *underestimator* of a function $f$ on a set $X$ if

$$\ell(x) \leq f(x), \quad \forall x \in X.$$

**Definition 2.2** (Overestimator)**.** A function $u$ is defined as an *overestimator* of a function $f$ on a set $X$ if

$$u(x) \geq f(x), \quad \forall x \in X.$$

2.1. **Construction of underestimators and overestimators of $f$.** To approximate the feasible domain of the original problem (1.1), we adopt a covering strategy inspired by global optimization techniques, specifically those developed by Lera et al. [13], and by Evtushenko et al. [4] (e.g., the Sequential Covering Algorithm, see [14]). This approach partitions the initial interval $[a, b]$ into sub-intervals based on the local behavior of $f$, which is assumed to satisfy the Lipschitz continuity condition as given in (1.2).

For each sub-interval $D_i = [x_{i-1}, x_i]$, we construct two affine approximations for $f(x)$:

- A lower envelope (an underestimator), which lies entirely below $f(x)$.
- An upper envelope (an overestimator), which lies entirely above $f(x)$.

Based on these constructed estimators, each sub-interval $D_i$ is classified as follows:

(1) If the upper bound of the overestimator on $D_i$ is strictly less than zero, then $f(x) < 0$ for all $x \in D_i$, and the sub-interval is definitively part of the solution set.
(2) If the lower bound of the underestimator on $D_i$ is strictly greater than zero, then $f(x) > 0$ for all $x \in D_i$, and the sub-interval is definitively excluded from the solution set.
(3) Otherwise, the sign of $f(x)$ is uncertain on this sub-interval. In this case, the sub-interval is subdivided further into three equal parts, and the

algorithm is recursively applied to each new sub-interval until a stopping criterion (e.g., interval length $\leq \delta$) is met.

This classification process allows the approximation of the solution set to be refined iteratively using only the Lipschitz property, without requiring differentiability or convexity of $f$.

Adhering to the Lipschitz condition defined in equation (1.2), for any arbitrary pair of points $x, y$ within the interval $[a, b]$ the following relationship holds:

$$f(y) - L|x - y| \leq f(x) \leq f(y) + L|x - y|.$$

This fundamental property allows us to construct piecewise affine envelopes. Now, considering a specific sub-interval $D_i = [x_{i-1}, x_i]$, we construct the affine underestimators and overestimators. For each sub-interval $D_i$, we have values $f(x_{i-1})$ and $f(x_i)$ at its endpoints.

The two affine functions forming the lower envelope (underestimator) are based on the Lipschitz constant $L$ and the function values at the endpoints:

$$\ell_i^{\text{left}}(x) = -Lx + f(x_{i-1}) + Lx_{i-1} \quad \text{for } x \in D_i$$
$$\ell_i^{\text{right}}(x) = Lx + f(x_i) - Lx_i \quad \text{for } x \in D_i$$

Consequently, the global underestimator $\ell_i(x)$ for $f$ within $[x_{i-1}, x_i]$ is established as the maximum of these two functions:

$$\ell_i(x) = \max\{\ell_i^{\text{left}}(x), \ell_i^{\text{right}}(x)\} \quad \text{for } x \in D_i.$$

This function $\ell_i(x)$ furnishes a precise, piecewise-linear lower approximation for $f(x)$ across the interval $D_i$.

Similarly, the two affine functions forming the upper envelope (overestimator) are:

$$u_i^{\text{left}}(x) = Lx + f(x_{i-1}) - Lx_{i-1} \quad \text{for } x \in D_i$$
$$u_i^{\text{right}}(x) = -Lx + f(x_i) + Lx_i \quad \text{for } x \in D_i$$

The global overestimator $u_i(x)$ for $f$ on $D_i$ is then defined as the minimum of these two functions:

$$u_i(x) = \min\{u_i^{\text{left}}(x), u_i^{\text{right}}(x)\} \quad \text{for } x \in D_i.$$

This function $u_i(x)$ provides a tight, piecewise-linear upper bound for $f(x)$ over the interval $D_i$.

However, while $\ell_i(x)$ and $u_i(x)$ are explicitly defined, their exact evaluation across the entire sub-interval to find their global minimum/maximum values (which are typically used for the classification criteria 1 and 2) can be computationally intensive as it involves identifying the maximum of $u_i(x)$ and minimum of $\ell_i(x)$ over the interval. Therefore, our algorithm focuses on an efficient approximate calculation method for these crucial bounds for classification.

2.2. **Determining classification bounds.** As discussed in the previous section, the exact underestimator $\ell_i(x)$ is defined as the maximum of two affine functions, and the exact overestimator $u_i(x)$ as the minimum of two affine functions over $D_i$. For the purpose of classifying these sub-intervals, we need to efficiently

determine the maximum value of the overestimator and the minimum value of the underestimator on $D_i$. These extreme values serve as the crucial bounds for our classification criteria.

The underestimator $\ell_i(x)$ is defined by $\ell_i(x) = \max\{\ell_i^{\text{left}}(x), \ell_i^{\text{right}}(x)\}$. This function $\ell_i(x)$ is piecewise linear and convex. Its minimum value over $D_i$ occurs at the intersection point of $\ell_i^{\text{left}}(x)$ and $\ell_i^{\text{right}}(x)$. Let $y_1^i$ be this intersection point. We find $y_1^i$ by solving $\ell_i^{\text{left}}(x) = \ell_i^{\text{right}}(x)$:

$$-Lx + f(x_{i-1}) + Lx_{i-1} = Lx + f(x_i) - Lx_i$$

$$f(x_{i-1}) - f(x_i) + Lx_{i-1} + Lx_i = 2Lx$$

Thus, the intersection point is:

$$y_1^i = \frac{1}{2}(x_i + x_{i-1}) - \frac{f(x_i) - f(x_{i-1})}{2L}. \tag{2.1}$$

The minimum value of the underestimator $\ell_i(x)$ over $D_i$, which we denote as $\alpha_i$, is then given by evaluating $\ell_i(x)$ at this intersection point $y_1^i$:

$$\alpha_i = \ell_i(y_1^i) = -Ly_1^i + f(x_{i-1}) + Lx_{i-1} = Ly_1^i + f(x_i) - Lx_i. \tag{2.2}$$

This value $\alpha_i$ represents the greatest lower bound provided by our underestimator on $D_i$.

Similarly, the overestimator $u_i(x)$ is defined by $u_i(x) = \min\{u_i^{\text{left}}(x), u_i^{\text{right}}(x)\}$. This function $u_i(x)$ is piecewise linear and concave. Its minimum value over $D_i$ occurs at the intersection point of $u_i^{\text{left}}(x)$ and $u_i^{\text{right}}(x)$. Let $y_2^i$ be this intersection point. We find $y_2^i$ by solving $u_i^{\text{left}}(x) = u_i^{\text{right}}(x)$:

$$Lx + f(x_{i-1}) - Lx_{i-1} = -Lx + f(x_i) + Lx_i$$

$$2Lx = f(x_i) - f(x_{i-1}) + Lx_i + Lx_{i-1}$$

Thus, the intersection point is:

$$y_2^i = \frac{1}{2}(x_i + x_{i-1}) + \frac{f(x_i) - f(x_{i-1})}{2L}. \tag{2.3}$$

The maximum value of the overestimator $u_i(x)$ over $D_i$, which we denote as $\beta_i$, is then given by evaluating $u_i(x)$ at this intersection point $y_2^i$:

$$\beta_i = u_i(y_2^i) = Ly_2^i + f(x_{i-1}) - Lx_{i-1} = -Ly_2^i + f(x_i) + Lx_i. \tag{2.4}$$

This value $\beta_i$ represents the smallest upper bound provided by our overestimator on $D_i$.

For all $x \in D_i$, these bounds satisfy:

$$\alpha_i \leq f(x) \leq \beta_i. \tag{2.5}$$

These values $\alpha_i$ and $\beta_i$ are the quantities used to classify the sub-intervals $D_i$ according to the rules outlined in Section 2.1.

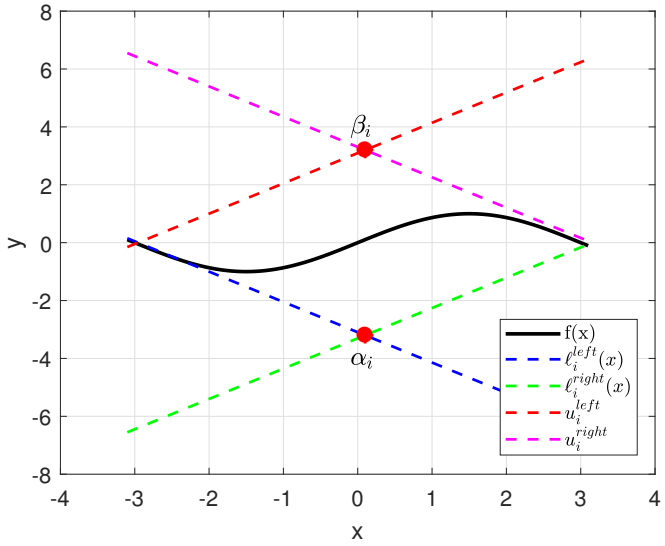



FIGURE 1. Illustration of the construction of Lipschitz bounds $\ell_i(x)$ and $u_i(x)$ over a sub-interval $D_i$. The values $\alpha_i$ and $\beta_i$ correspond to the minimum of the underestimator and maximum of the overestimator, respectively.

2.3. **Interval classification based on Lipschitz bounds.** Let the initial domain $[a, b]$ be adaptively partitioned into a collection of sub-intervals $\{D_i\}$. For each sub-interval $D_i = [x_{i-1}, x_i]$, we utilize the minimum value of the underestimator, $\alpha_i$ (as defined in (2.2)), and the maximum value of the overestimator, $\beta_i$ (as defined in (2.4)). Based on these bounds, we classify the sub-intervals into three distinct sets:

(1) Solution set candidates ($Q_{\text{Sol}}$): A sub-interval $D_i$ belongs to this set if its overestimator's maximum value is strictly negative, which implies $f(x) < 0$ for all $x \in D_i$.

$$Q_{\text{Sol}} = \{D_i \mid \beta_i < 0\}. \tag{2.6}$$

(2) Non-solution set ($Q_{\text{non-sol}}$): A sub-interval $D_i$ belongs to this set if its underestimator's minimum value is strictly positive, which implies $f(x) > 0$ for all $x \in D_i$.

$$Q_{\text{non-sol}} = \{D_i \mid \alpha_i > 0\}. \tag{2.7}$$

(3) Uncertain set ($Q_{\text{uncertain}}$): A sub-interval $D_i$ belongs to this set if it does not satisfy the conditions for either $Q_{\text{Sol}}$ or $Q_{\text{non-sol}}$. This indicates that the sign of $f(x)$ is not definitively determined over $D_i$, and the interval may contain parts of the solution set, non-solution set, or zeros.

$$Q_{\text{uncertain}} = \{D_i \mid D_i \notin Q_{\text{Sol}} \text{ and } D_i \notin Q_{\text{non-sol}}\}. \tag{2.8}$$

This classification forms the basis of our recursive algorithm for approximating the solution set. The procedure for processing each sub-interval $D_i$ is as follows:

- If $D_i \in Q_{\text{Sol}}$, it is definitively part of the solution set and is added to the final approximation of the feasible region. Further subdivision of this interval is not required.
- If $D_i \in Q_{\text{non-sol}}$, it is definitively excluded from the solution set. This interval is discarded.
- If $D_i \in Q_{\text{uncertain}}$, its behavior is ambiguous. To refine the approximation, this sub-interval is subdivided into three equal smaller sub-intervals. The algorithm then recursively applies the classification and subdivision process to each of these new sub-intervals. This recursive refinement continues until a predefined stopping criterion is met for all sub-intervals, such as their length being less than or equal to a tolerance $\delta$.

This adaptive partitioning and classification strategy allows for efficient identification of the solution set of nonlinear inequalities without requiring strong assumptions on $f$, while ensuring convergence to an accurate approximation.

## 3. The modified covering algorithm based on Lipschitz envelopes

In this section, we present a modified covering algorithm designed to efficiently approximate the solution set of nonlinear inequalities using the Lipschitz underestimators and overestimators constructed in Section 2. This algorithm adaptively refines the domain by employing a three-way splitting strategy for uncertain intervals, diverging from some traditional binary subdivision methods. This multi-part subdivision is chosen to potentially accelerate the convergence rate and to more finely localize the solution boundaries, particularly for functions with complex local behavior.

### 3.1. **Algorithm description.**

The proposed modification is based on dividing the interval $D_i$ into a three-way split(trisection). This approach can enable faster isolation of the solution boundaries compared to a bisection method. The algorithm iteratively processes a queue of sub-intervals, classifying each based on the Lipschitz bounds derived in Section 2.2. We recall that $\alpha_i$ (defined in (2.2)) represents the minimum value of the underestimator $\ell_i(x)$ over $D_i$, and $\beta_i$ (defined in (2.4)) represents the maximum value of the overestimator $u_i(x)$ over $D_i$. These are the critical values used for the classification criteria presented in Section 2.3.

If an interval $D_i$ falls into the $Q_{\text{uncertain}}$ category (i.e., it does not satisfy the conditions for $Q_{\text{Sol}}$ or $Q_{\text{non-sol}}$ from (2.6)-(2.7)), it is divided into three equal sub-intervals. This three-way division method, while differing from some prior works (e.g., Lera et al. [13]), is intended to provide a more granular refinement step, potentially reducing the total number of iterations required to achieve a desired tolerance.

To formally describe the algorithm, we introduce the following sets:

- $\mathcal{L}_{\text{active}}$: A list (or queue) of sub-intervals that are currently being processed and require further examination. Initially, this contains the entire domain $[a, b]$.
- $\mathcal{S}_{\text{sol}}$: The collection of sub-intervals definitively identified as part of the solution set, i.e., where $f(x) < 0$.

- $\mathcal{S}_{\text{zero}}$: The collection of sub-intervals where the sign of $f(x)$ is uncertain but the interval length has reached the tolerance $\delta$. These intervals are considered to contain potential zeros or boundary regions of the solution set.

Intervals definitively identified as non-solutions (where $f(x) > 0$) are discarded and not stored.

**Algorithm 1** Lipschitz-based interval covering algorithm

**Input:** Function $f(x)$, interval $[a, b]$, Lipschitz constant $L > 0$, tolerance $\delta > 0$ (minimum interval length), maximum iterations $T_{\max}$ (optional, for robustness against infinite loops)
**Output:** $\mathcal{S}_{\text{sol}}$: Set of sub-intervals where $f(x) < 0$; $\mathcal{S}_{\text{zero}}$: Set of sub-intervals where $f(x) \approx 0$ (boundary regions)
**Initialize:**
$\mathcal{L}_{\text{active}} \leftarrow \{[a, b]\}$ ▷ Queue of intervals to process
$\mathcal{S}_{\text{sol}} \leftarrow \emptyset$
$\mathcal{S}_{\text{zero}} \leftarrow \emptyset$
Set iteration counter $T \leftarrow 0$
**while** $\mathcal{L}_{\text{active}} \neq \emptyset$ **and** $T < T_{\max}$ **do**
    $D_i = [a_i, b_i] \leftarrow$ Pop an interval from $\mathcal{L}_{\text{active}}$
    Compute Lipschitz Bounds for $D_i$:
    $y_1^i = \frac{1}{2}(b_i + a_i) - \frac{f(b_i) - f(a_i)}{2L}$ ▷ Intersection for underestimator, see (2.1)
    $\alpha_i = -Ly_1^i + f(a_i) + La_i$ ▷ Minimum of underestimator, see (2.2)
    $y_2^i = \frac{1}{2}(b_i + a_i) + \frac{f(b_i) - f(a_i)}{2L}$ ▷ Intersection for overestimator, see (2.3)
    $\beta_i = Ly_2^i + f(a_i) - La_i$ ▷ Maximum of overestimator, see (2.4)
    Classify and Process Interval:
    **if** $\beta_i < 0$ **then** ▷ Case 1: $D_i \in Q_{\text{Sol}}$, definitely part of the solution set
        $\mathcal{S}_{\text{sol}} \leftarrow \mathcal{S}_{\text{sol}} \cup \{[a_i, b_i]\}$
    **else if** $\alpha_i > 0$ **then** ▷ Case 2: $D_i \in Q_{\text{non-sol}}$, definitely not part of the solution set
        **Discard** $[a_i, b_i]$ ▷ This interval is not part of the solution
    **else if** $(b_i - a_i) \leq \delta$ **then** ▷ Case 3: $D_i \in Q_{\text{uncertain}}$, but reached minimum length
        $\mathcal{S}_{\text{zero}} \leftarrow \mathcal{S}_{\text{zero}} \cup \{[a_i, b_i]\}$
    **else** ▷ Case 4: $D_i \in Q_{\text{uncertain}}$, and needs further subdivision
        $h \leftarrow (b_i - a_i)/3$
        $m_1 \leftarrow a_i + h$
        $m_2 \leftarrow a_i + 2h$
        Add $[a_i, m_1]$, $[m_1, m_2]$, and $[m_2, b_i]$ to $\mathcal{L}_{\text{active}}$
    **end if**
    $T \leftarrow T + 1$
**end while**
**return** $(\mathcal{S}_{\text{sol}}, \mathcal{S}_{\text{zero}})$

3.2. **Convergence analysis.** This section establishes a key convergence property of Algorithm 1. We prove that for any interval terminated due to the tolerance threshold $\delta$, the values of the function $f(x)$ within that interval are bounded. This result guarantees that as the tolerance $\delta$ approaches zero, the algorithm correctly identifies regions where the function's value is also close to zero.

**Theorem 3.1.** *Let $\mathcal{S}_{zero}$ be the set of sub-intervals collected by Algorithm 1, where each interval $D_i = [a_i, b_i]$ has a length satisfying $(b_i - a_i) \leq \delta$. For any interval $D_i \in \mathcal{S}_{zero}$, the function values $f(x)$ for all $x \in D_i$ are bounded as follows:*

$$|f(x)| \leq L\delta \quad \forall x \in D_i,$$

*where $L$ is the Lipschitz constant of $f$ on $[a, b]$.*

*Proof.* Consider an interval $D_i = [a_i, b_i]$ from the set $\mathcal{S}_{\text{zero}}$. By the definition of the algorithm, this interval must be uncertain, which implies its bounds satisfy:

$$\alpha_i = \min_{x \in D_i} \ell_i(x) \leq 0$$
$$\beta_i = \max_{x \in D_i} u_i(x) \geq 0.$$

We also know that for any $x \in D_i$, the function is bounded by its estimators:

$$\alpha \leq f(x) \leq \beta_i.$$

Our goal is to bound the width of the interval $[\alpha_i, \beta_i]$.

From (2.1), (2.2), (2.3) and (2.4), we have the explicit formulas for the bounds:

$$\alpha_i = \frac{f(a_i) + f(b_i)}{2} - \frac{L(b_i - a_i)}{2},$$
$$\beta_i = \frac{f(a_i) + f(b_i)}{2} + \frac{L(b_i - a_i)}{2}.$$

The difference between the maximum of the overestimator and the minimum of the underestimator is therefore:

$$\beta_i - \alpha_i = 2\frac{L(b_i - a_i)}{2} = Lh_i,$$

where $h_i = b_i - a_i$.

Since the interval $D_i$ is uncertain, we know that 0 is contained within the range $[\alpha_i, \beta_i]$. For any $x \in D_i$, we also have $f(x) \in [\alpha_i, \beta_i]$ . The maximum possible absolute value for any number in this range (including $f(x)$ and 0) is bounded by the length of the range itself. Therefore, for any $x \in D_i$:

$$|f(x)| \leq \beta_i - \alpha_i = Lh_i.$$

As the interval $D_i$ is in $\mathcal{S}_{\text{zero}}$, its length satisfies $h_i \leq \delta$. This gives us the final result:

$$|f(x)| \leq Lh_i \leq L\delta.$$

□

## 4. Numerical experiments

This section evaluates the performance of the proposed algorithm on a series of benchmark problems. The goal is to assess its effectiveness in accurately approximating solution sets for nonlinear inequalities.

All experiments were implemented in MATLAB R2017a and run on a system equipped with an Intel Core i5-10G processor (2.2GHz) and 16 GB RAM. Our analysis is presented in three parts:

(1) Graphical Visualization: We illustrate how the algorithm partitions the domain and converges to the final solution set.
(2) Performance Metrics: We report key metrics, including the number of iterations, and total execution time.
(3) Comparative Analysis: We compare our algorithm's performance against established methods to highlight its competitive advantages.

4.1. **Test functions and Lipschitz constants.** The algorithm was tested on the set of Lipschitz continuous functions listed in Table 1. These functions were chosen to represent a variety of challenges, including different numbers of roots and varying degrees of nonlinearity.

Table 1. Test functions that satisfy the Lipschitz condition.

| No. | Objective function | Region | Lipschitz constant $L$ | Ref. |
|---|---|---|---|---|
| 1 | $\sin(x)$ | $[-\pi, \pi]$ | 1 | |
| 2 | $-2 + x^2$ | $[-3, 3]$ | 6 | |
| 3 | $\sin(\frac{10}{3}x) + \sin(x)$ | $[2.70, 7.50]$ | 4.29 | [5] |
| 4 | $(-5 + 24x - 16x^2)e^{(-x)}$ | $[1.90, 3.90]$ | 3 | [5] |
| 5 | $(-1.4 + 3x)\sin(18x)$ | $[0, 1.20]$ | 36 | [5] |
| 6 | $-(x + \sin x)e^{(-x^2)}$ | $[-10, 10]$ | 2.5 | [5] |
| 7 | $3 - 0.84x + \ln(x)$ | $[2.70, 7.50]$ | 6 | [9] |
| 8 | $\sin(2x/3) + \sin(x)$ | $[3.10, 20.40]$ | 1.7 | [9] |
| 9 | $(-x)\sin(x)$ | $[0, 10]$ | 11 | [9] |
| 10 | $\cos(2x) + 2\cos(x)$ | $[-1.57, 6.28]$ | 3 | [9] |
| 11 | $\cos^3(x) + \sin^3(x)$ | $[0, 6.28]$ | 2.2 | [9] |
| 12 | $-x^{2/3} + (x^2 - 1)^{1/3}$ | $[0.001, 0.99]$ | 8.5 | [9] |
| 13 | $\left(e^{(-x)}\right)\sin(2\pi x)$ | $[0, 4]$ | 6.5 | [9] |
| 14 | $-(1 + x) + \sin(3x)$ | $[0, 6.5]$ | 4 | [9] |
| 15 | $\frac{1}{6}x^6 - \frac{52}{25}x^5 + \frac{39}{80}x^4 + \frac{71}{10}x^3 - \frac{79}{20}x^2 - x + \frac{1}{10}$ | $[-1.5, 1.1]$ | 13870 | [9] |
| 16 | $\sum_{k=1}^{5} \sin(x + kx + k)$ | $[-10, 10]$ | 67 | [10] |
| 17 | $\sum_{k=1}^{5} k \cdot \cos((x + kx + k)$ | $[-10, 10]$ | 67 | [10] |
| 18 | $\left(x\sqrt{2} - 3\sqrt{2}\right)^2 + e^{(x^2/2)}$ | $[-3, 3]$ | 85 | [10] |
| 19 | $-(1 + \cos(x)) + \sin(5x)$ | $[0, 7]$ | 5.951 | [15] |
| 20 | $\sum_{k=1}^{5} \cos(x + kx + k)$ | $[-10, 10]$ | 18.119 | [15] |

4.2. **Numerical results.** In all experiments, the algorithm's tolerance was set to $\delta = 10^{-4}$. The algorithm's performance and resulting solution sets for all test functions are summarized here, with full details provided in Appendix 5, specifically Tables 3 and 4.

The results in Tables 3 and 4, demonstrate the algorithm's effectiveness across a diverse set of problems. For simpler functions (e.g., Test 1-3), the algorithm rapidly converges in a few iterations. For more complex, oscillatory functions (e.g., Test 16 and 20), the algorithm successfully navigates intricate solution landscapes, producing a union of small intervals that accurately capture the solution set.

Notably, for functions where no solution exists (Test 8 and 18), the algorithm correctly terminates with an empty solution set ($\mathcal{S}_{\text{sol}}$=emptyset), demonstrating its robustness. The execution time is generally low, often under a tenth of a second, with the exception of highly complex cases like Test 15 and 16, which require more extensive domain exploration.

4.3. **Comparative performance analysis.** To benchmark our algorithm (Alg3), we conducted a comparative analysis against two methods from the literature: an algorithm by Evtushenko et al. [3] (Alg1) and a method by Lera et al. [13] (Alg2). The performance of all three was evaluated based on the number of iterations and the total CPU time required to solve 20 test problems.

The complete numerical results are presented in Table 2, and a visual summary is provided in Figure 2.

TABLE 2. Comparative numerical results.

| Problem No. | Alg1 | | Alg2 | | Alg3 | |
|---|---|---|---|---|---|---|
| | Iter. Num. | Time (s) | Iter. Num. | Time (s) | Iter. Num. | Time (s) |
| 1 | 14 | 0.1957 | 37 | 0.002453 | 19 | 0.009410 |
| 2 | 14 | 0.52157 | 103 | 0.03482 | 20 | 0.01141 |
| 3 | 14 | 0.569136 | 470 | 0.03990 | 12 | 0.009537 |
| 4 | 1 | 0.05390 | 5 | 0.005739 | 2 | 0.004940 |
| 5 | 12 | 0.69 | 1135 | 0.03168 | 18 | 0.021705 |
| 6 | 19 | 0.22243 | 1208 | 0.03168 | 20 | 0.975 |
| 7 | 20 | 0.8799 | 380 | 0.03037 | 13 | 0.02442 |
| 8 | 5 | 0.19003 | 3 | 0.005019 | 3 | 0.006844 |
| 9 | 21 | 0.23208 | 1227 | 0.03974 | 114 | 0.6234 |
| 10 | 21 | 0.795749 | 1002 | 0.03227 | 18 | 0.03763 |
| 11 | 21 | 0.571184 | 43 | 0.001439 | 6 | 0.02634 |
| 12 | 18 | 0.4344 | 284 | 0.03696 | 18 | 0.04318 |
| 13 | 13 | 0.2323137 | 1058 | 0.03321 | 20 | 0.08437 |
| 14 | 6 | 0.701372 | 14 | 0.006352 | 4 | 0.01198 |
| 15 | 38 | 0.4567 | 1012 | 0.001773 | 36 | 0.9642 |
| 16 | 27 | 0.804192 | 1372 | 0.2864 | 24 | 0.2175 |
| 17 | 22 | 0.2126830 | 1019 | 0.1865 | 21 | 0.7153 |
| 18 | 7 | 0.0841027 | 33 | 0.018763 | 5 | 0.007411 |
| 19 | 29 | 0.1364226 | 441 | 0.018971 | 14 | 0.027355 |
| 20 | 31 | 0.7179 | 1012 | 0.118006 | 21 | 0.037 |

The comparative analysis, presented in Table 2, details the number of iterations and time consumed for the three algorithms across various problems. Our proposed algorithm, Alg3, consistently demonstrates a significantly lower number of iterations compared to Alg2 [13] for most problems. When compared to Alg1 [3], Alg3 often achieves a comparable or slightly higher number of iterations but generally exhibits competitive performance in terms of execution time across a range of problems.

The performance of the three algorithms is visually summarized in the Figure 2. These plots compare the total number of iterations and the execution time required to solve each test problem.

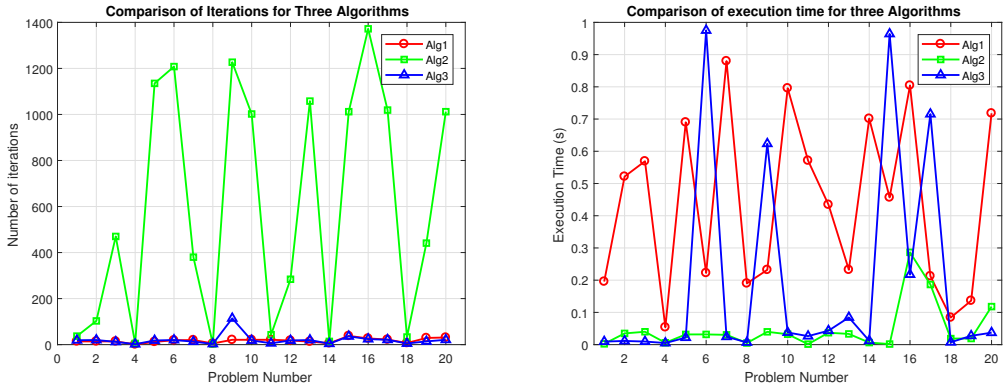



FIGURE 2. Performance comparison of Alg1 (red), Alg2 (green), and Alg3 (blue) across all test problems. Left: Total number of iterations. Right: Total execution time in seconds. Lower bars indicate better performance.

The plots clearly show that our proposed algorithm, Alg3 (blue), is unequivocally the most efficient in terms of iteration count. It consistently converges in the fewest steps for every problem, while Alg2 (green) requires a vastly higher number of iterations to find the solution but is often fast, implying a very low computational cost per iteration.

The right panel, which illustrates execution time, reveals a more nuanced trade-off. While Alg1 (red) is generally the slowest method, Alg2 often has the lowest runtime, despite its high iteration count. Our algorithm, Alg3, remains highly competitive, delivering solutions significantly faster than Alg1 and often rivaling the raw speed of Alg2.

Taken together, these results demonstrate that Alg3 offers a superior balance of efficiency and speed. By achieving convergence in a minimal number of steps while maintaining a low computational cost, it proves to be a robust and powerful solution for this class of problems.

## 5. Conclusion

In this research, we introduced a technique for solving nonlinear inequalities that satisfy the Lipschitz condition. Our method is fundamentally based on a

domain-covering strategy, wherein the search space is systematically partitioned. Each resulting subdomain is then rigorously analyzed using an adapted algorithm. This algorithm constructs upper and lower linear estimators, whose intersection points define crucial lower and upper boundary values for the function within the subdomain. By evaluating the sign of these estimators, we efficiently classify subdomains into three distinct categories: those entirely contained within the solution set, those entirely outside it, and those that intersect the boundary and thus require further subdivision. Subdomains belonging to this latter category are subsequently refined through uniform partitioning.

The practical value of this algorithm was confirmed through extensive numerical experiments. The comparative analysis demonstrated that our method achieves a superior balance of performance, significantly reducing the number of iterations compared to established algorithms while remaining highly competitive in computational time.

Future work could focus on extending this technique to solve systems of multivariate inequalities and exploring other adaptive multi-part subdivision strategies.

**Acknowledgement.** The authors would like to express their sincere thanks to the editor and the anonymous reviewers for their helpful comments and suggestions.

## Appendix: Detailed numerical results

This appendix provides the full numerical results for the proposed algorithm (Alg3) on the set of test problems, complementing the summarized findings in Section 4. The data includes CPU time, number of iterations, and the computed solution sets. Graphical representations of selected test functions are also presented.

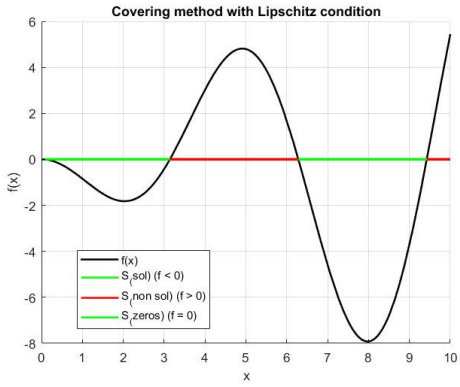
TABLE 3. Numerical results for the test examples

| No. | CPU Time (s) | Iter. Num. | Solution set of Intervals $S_{sol}$ |
|---|---|---|---|
| 1 | 0.009410 | 19 | $[-\pi, 0]$ |
| 2 | 0.011412 | 20 | $[-1.4142, 1.4142]$ |
| 3 | 0.009537 | 12 | $[-3.1416, 0.0000]$ |
| 4 | 0.004940 | 2 | $[1.9, 2.5667] \cup [2.5667, 3.2333] \cup [3.2333, 3.9]$ |
| 5 | 0.031858 | 12 | $[0, 0.1745] \cup [0.3491, 0.4666] \cup [0.5236, 0.6981] \cup [0.8727, 1.0472] \cup [0.9778, 1.0222]$ |
| 6 | 0.975000 | 20 | $[0.5320, 0.7168] \cup [1.2488, 1.2512] \cup [2.5, 3.75] \cup [5, 6.875] \cup [8.125, 9.375]$ |
| 7 | 0.024428 | 13 | $[5.6287, 7.5]$ |
| 8 | 0.006844 | 3 | $\emptyset$ |
| 9 ($Tmax = 1000$) | 0.099448 | 14 | $[0.0059, 3.1416] \cup [6.2832, 9.4248]$ |
| 10 ($Tmax = 100$) | 0.066888 | 16 | $[0, 0.1745] \cup [0.3491, 0.4667] \cup [0.5236, 0.6981] \cup [0.8727, 1.0472]$ |
| 10 ($Tmax = 1000$) | 0.037632 | 18 | $[0, 0.1745] \cup [0.3491, 0.4667] \cup [0.5236, 0.6981] \cup [0.8727, 1.0472]$ |
| 11 | 0.026344 | 6 | $[2.3562, 5.4978]$ |
| 12 ($Tmax = 1000$) | 0.043182 | 18 | $[0.3333, 0.9900]$ |
| 12 ($Tmax = 100$) | 0.026041 | 18 | $[0.3333, 0.9900]$ |
| 13 ($Tmax = 1000$) | 0.109065 | 20 | $[0.5, 1.0] \cup [1.5, 2.0] \cup [2.5, 3.0] \cup [3.5, 4.0]$ |
| 13 ($Tmax = 100$) | 0.084374 | 20 | $[0.5, 1.0] \cup [1.5, 2.0] \cup [2.5, 3.0] \cup [3.5, 4.0]$ |
| 14 ($Tmax = 100$) | 0.011987 | 4 | $[0, 6.5]$ |
| 15 | 30.964242 | 36 | $[-1.1000, 0.8089]$ |
| 16 | 0.217500 | 24 | $[-9.7011, -9.6959] \cup [-9.6229, -9.6210] \cup [-9.5481, -9.5447] \cup [-9.4735, -9.4682] \cup [-9.3969, -9.3931] \cup [-8.5822, -8.5816] \cup [-7.7954, -7.0730] \cup [-6.2477, -6.2477] \cup [-6.2473, -6.2472] \cup [-6.2439, -6.2435] \cup [-6.2406, -6.2398] \cup [-6.2368, -6.2365] \cup [-6.2335, -6.2320] \cup [-6.2290, -6.2287] \cup [-6.2257, -6.2251] \cup [-6.2221, -6.2188] \cup [-6.1467, -6.1463] \cup [-6.0742, -6.0704] \cup [-5.3870, -5.3781] \cup [-5.3141, -5.2959] \cup [-5.2244, -5.2218] \cup [-5.1503, -5.1484] \cup [-5.0778, -5.0734] \cup [-5.0028, -5.0010] \cup [-4.9302, -4.9237] \cup [-4.8487, -4.8484] \cup [-4.5467, -4.5353] \cup [-4.4653, -4.4586] \cup [-4.3886, -4.3756] \cup [-4.3028, -4.3002] \cup [-4.2274, -4.2257] \cup [-4.1537, -4.1494] \cup [-4.0775, -4.0614] \cup [-3.9941, -3.9863] \cup [-3.9127, -3.9120] \cup [-3.4404, -3.4345] \cup [-3.3600, -3.3572] \cup [-3.2827, -3.2809] \cup [-3.2073, -3.2028] \cup [-3.1291, -3.1249] \cup [-1.4632, -0.7748] \cup [0.1538, 0.1540] \cup [0.9206, 0.9426] \cup [1.0132, 1.0180] \cup [1.0885, 1.0907] \cup [1.1623, 1.1650] \cup [1.2366, 1.2375] \cup [1.3076, 1.3141] \cup [1.3842, 1.3949] \cup [1.6948, 1.6997] \cup [1.7701, 1.7767]$ |

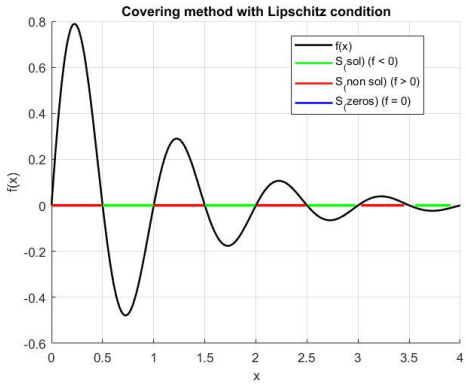
TABLE 4. Numerical results for the test examples (continued)

| No. | CPU Time (s) | Iter. Num. | Solution set of Intervals $S_{sol}$ |
|---|---|---|---|
| 17 | 0.733534 | 21 | $[-9.5548, -9.0342] \cup [-8.5761, -8.0549] \cup [-7.4054, -6.7396] \cup [-6.1589, -5.7098] \cup [-5.1967, -4.7169] \cup [-4.2365, -3.7313] \cup [-3.2716, -2.7510] \cup [-2.2929, -1.7717] \cup [-1.1222, -0.4565] \cup [0.1243, 0.5733] \cup [1.0865, 1.5663] \cup [2.0467, 2.5519] \cup [3.0116, 3.5322] \cup [3.9902, 4.5115] \cup [5.1609, 5.8267] \cup [6.4075, 6.8565] \cup [7.3697, 7.8494] \cup [8.3299, 8.8351] \cup [9.2948, 9.8154]$ |
| 18 | 0.007823 | 5 | $\emptyset$ |
| 19 | 0.025527 | 20 | $[0, 1.5708] \cup [1.6518, 2.5477] \cup [3.1416, 3.8138] \cup [4.2753, 7.0000]$ |
| 20 | 0.345500 | 32 | $[-8.9611, -8.6400] \cup [-8.9611, -8.5646] \cup [-8.1627, -7.5205] \cup [-6.5690, -6.0748] \cup [-5.7504, -5.7938] \cup [-5.7504, -4.5831] \cup [-4.5192, -4.5178] \cup [-4.4532, -4.4437] \cup [-4.3791, -4.3602] \cup [-2.6837, -2.3907] \cup [-2.3435, -2.2947] \cup [-1.8971, -1.2190] \cup [-0.2738, 0.1908] \cup [0.4480, 0.4546] \cup [1.1400, 1.1617] \cup [1.2010, 1.2053] \cup [1.2084, 1.2102] \cup [1.5699, 1.6012] \cup [1.6604, 1.6737] \cup [1.7329, 1.7414] \cup [1.8047, 1.8097] \cup [1.8730, 1.8985] \cup [3.6393, 3.9119] \cup [3.9538, 3.9828] \cup [4.3730, 5.1369] \cup [5.9850, 6.4687] \cup [6.7910, 7.4342] \cup [7.9231, 7.9382] \cup [7.9971, 8.0099] \cup [8.0749, 8.0779] \cup [8.1429, 8.1704] \cup [8.2150, 8.2335] \cup [9.8844, 9.9057] \cup [9.9443, 10.0000]$ |

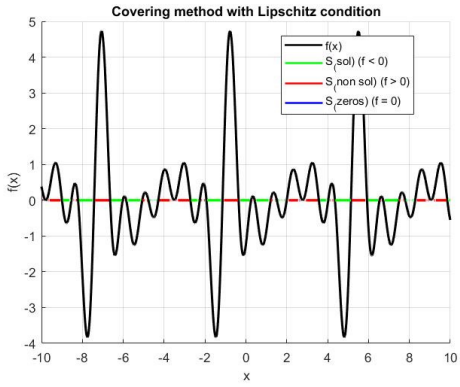
Figure 3 provides graphical illustrations of the solution sets for five selected test functions, complementing the numerical results presented in Tables 3 and 4. As depicted, the solution set for Function 8 is empty (shown in red), while for Functions 9, 13, 14, and 20, the solution sets are unions of intervals, highlighted in green. The function 14 shows an example where the solution set covers the entire domain.



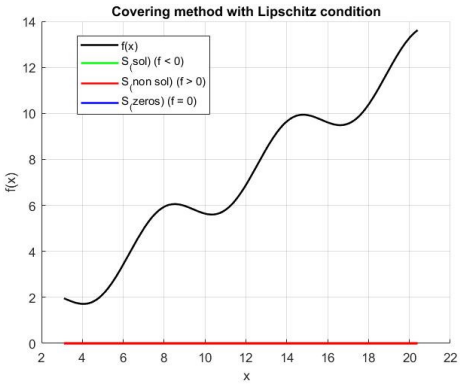

(a) Function 9: $f(x) = (-x)\sin(x)$



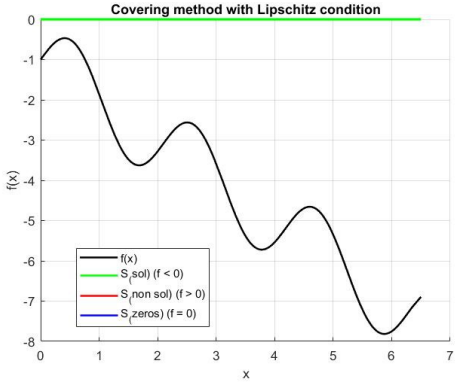

(b) Function 13: $f(x) = e^{(-x)}\sin(2\pi x)$




(c) Function 20: $f(x) = \sum_{k=1}^{5}\cos(x+kx+k)$




(d) Function 8: $f(x) = \sin(2x/3) + \sin(x)$




(e) Function 14: $f(x) = -(1+x) + \sin(3x)$

FIGURE 3. Graphical illustration of the solution regions. Green segments represent the union of solution intervals, while red indicates an empty solution region.

## References


1. Evtushenko, Y. G. (1971). Numerical methods for finding global extrema (case of a non-uniform mesh). USSR Computational Mathematics and Mathematical Physics, 11(6), 38–54. https://doi.org/10.1016/0041-5553(71)90065-6
2. Evtushenko, Y. G., Posypkin, M. A., Rybak, L. A., and Turkin, A. V. (2016). Numerical method for approximating the solution set of a system of non-linear inequalities. International Journal of Open Information Technologies, 4(12), 1–6.
3. Evtushenko, Y. G., Posypkin, M. A. E., Rybak, L. A., and Turkin, A. V. (2017). Finding sets of solutions to systems of nonlinear inequalities. Computational Mathematics and Mathematical Physics, 57(8), 1241–1247. https://doi.org/10.1134/S0965542517080073
4. Evtushenko, Y., Posypkin, M., Rybak, L., and Turkin, A. (2018). Approximating a solution set of nonlinear inequalities. Journal of Global Optimization, 71(1), 129–145. https://doi.org/10.1007/s10898-017-0576-z
5. Gourdin, E., Jaumard, B., and Ellaia, R. (1996). Global optimization of Hölder functions. Journal of Global Optimization, 8(4), 323–348. https://doi.org/10.1007/BF02403997
6. Gu, C., and Zhu, D. (2012). A filter algorithm for nonlinear systems of equalities and inequalities. Applied Mathematics and Computation, 218(20), 10289–10298. https://doi.org/10.1016/j.amc.2012.04.007
7. Guettal, D., and Rahal, M. (2023). Global optimisation procedures for solving systems of Hölder equations-inequations. International Journal of Computing Science and Mathematics, 17(3), 266–275. https://doi.org/10.1504/IJCSM.2023.131457
8. Hansen, P., Jaumard, B., and Lu, S. H. (1992). Global optimization of univariate Lipschitz functions: I. Survey and properties. Mathematical programming, 55(1), 251-272. https://doi.org/10.1007/BF01581202
9. Hansen, P., Jaumard, B., and Lu, S. H. (1992). Global optimization of univariate Lipschitz functions: II. New algorithms and computational comparison. Mathematical programming, 55(1), 273–292. https://doi.org/10.1007/BF01581203
10. Lera, D., and Sergeyev, Y. D. (2002). Global minimization algorithms for Holder functions. BIT Numerical mathematics, 42(1), 119–133. https://doi.org/10.1023/A:1021926320198
11. Lera, D., Posypkin, M. A., and Sergeyev, Y. D. (2019). Approximating the solution set of nonlinear inequalities by using Peano space-filling curves. Numerical Computations: Theory and Algorithms NUMTA 2019, 204.
12. Lera, D., Posypkin, M., and Sergeyev, Y. D. (2021). Space-filling curves for numerical approximation and visualization of solutions to systems of nonlinear inequalities with applications in robotics. Applied Mathematics and Computation, 390, 125660. https://doi.org/10.1016/j.amc.2020.125660
13. Lera, D., Nasso, M. C., Posypkin, M., and Sergeyev, Y. D. (2024). Determining solution set of nonlinear inequalities using space-filling curves for finding working spaces of planar robots. Journal of Global Optimization, 89(2), 415–434. https://doi.org/10.1007/s10898-023-01352-2
14. Rahal, M., and Guettal, D. (2014). Modified sequential covering algorithm for finding a global minimizer of nondifferentiable functions and applications. Gen, 22(1), 100–115.
15. Sergeyev, Y. D., Candelieri, A., Kvasov, D. E., and Perego, R. (2020). Safe global optimization of expensive noisy black-box functions in the $\delta$-Lipschitz framework. Soft Computing, 24(23), 17715–17735. https://doi.org/10.1007/s00500-020-05030-3


[1] Laboratory of Fundamental And Numerical Mathematics,, Department of mathematics, Setif 1 University Ferhat Abbas. Setif, Algeria

*Email address*: nourelislem.hiber@univ-setif.dz

[2] Laboratory of Fundamental And Numerical Mathematics,, Department of Basic Education in Technology, Setif 1 University Ferhat Abbas. Setif, Algeria
*Email address*: djaouida.guettal@univ-setif.dz

[3] Laboratory of Fundamental And Numerical Mathematics,, Department of mathematics, Setif 1 University Ferhat Abbas. Setif, Algeria
*Email address*: mrahal@univ-setif.dz