BIT Numer Math (2017) 57:901–926
DOI 10.1007/s10543-017-0657-x

Using interval unions to solve linear systems of equations with uncertainties

Tiago Montanher[1] · **Ferenc Domes**[1] · **Hermann Schichl**[1] · **Arnold Neumaier**[1]

Received: 8 September 2016 / Accepted: 7 April 2017 / Published online: 22 April 2017

Abstract An interval union is a finite set of closed and disjoint intervals. In this paper we introduce the interval union Gauss–Seidel procedure to rigorously enclose the solution set of linear systems with uncertainties given by intervals or interval unions. We also present the interval union midpoint and Gauss–Jordan preconditioners. The Gauss–Jordan preconditioner is used in a mixed strategy to improve the quality and efficiency of the algorithm. Numerical experiments on interval linear systems generated at random show the capabilities of our approach.

Keywords Interval union arithmetic · Interval union linear systems · Interval union Gauss–Seidel · Rigorous numerical linear algebra

Mathematics Subject Classification 65F10 · 65G20 · 65G30 · 65G40

Communicated by Lars Eldén.

This research was partially supported through the research Grants P25648-N25 of the Austrian Science Fund (FWF), 853930 of the Austrian Research Promotion Agency (FFG) and CNPQ-205557/2014-7 of the Brazilian council of research (CNPQ).

✉ Tiago Montanher
tiago.de.morais.montanher@univie.ac.at

Ferenc Domes
ferenc.domes@univie.ac.at

Hermann Schichl
hermann.schichl@univie.ac.at

Arnold Neumaier
arnold.neumaier@univie.ac.at

[1] Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

1 Introduction

In traditional interval arithmetic, division by an interval containing zero overestimates the range when the latter is disconnected. Treating this using complements of intervals (see e.g, [23]) only postpones the problem a little, while interval union arithmetic, introduced by [24] as arithmetic on finite ordered sets of disjoint closed, possibly unbounded intervals, allow a mathematically and computationally natural approach to this problem. Indeed, the collection of interval unions (treated as closed sets in the obvious way) is closed under set-theoretic addition, subtraction, multiplication, division (after adding end points in case of an unbounded divisor), and all continuous elementary operations.

Many theoretical results from interval analysis remain valid for interval unions. For example, elementary operations and standard functions are inclusion isotone and the fundamental theorem of interval analysis also generalizes to interval unions. On the other hand, properties based on convexity (like the interval mean value theorem) do not apply to interval unions.

In this paper we study the rigorous solution of interval union linear systems of equations (IULS). We denote interval unions and vectors of interval unions by bold calligraphic letters (such as $\boldsymbol{a}$, $\boldsymbol{x}$), while matrices of interval unions are denoted by capital bold calligraphic letters (e.g., $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$). Let $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{b}$ be a matrix and a vector with interval union entries respectively. If $\boldsymbol{x}$ is a given initial interval union vector, we are interested in finding an enclosure of the solution set for the family of equations

$$Ax = b, \quad (A \in \boldsymbol{\mathcal{A}}, b \in \boldsymbol{b}, x \in \boldsymbol{x}). \tag{1}$$

This problem has several applications in rigorous numerical analysis. Since interval linear systems are embedded into the interval union framework, any algorithm that relies on the rigorous solution of interval linear systems can benefit from the methods discussed in this paper. For example, constraint propagation methods [5] and the interval Newton operator [20,21] can be significantly improved with the use of interval union techniques. Moreover, interval union linear systems of equations can be used to define an interval union branch and bound framework for rigorous global optimization. This application will be detailed in a future work.

Related work: A closely related concept is that of multi-intervals, introduced independently by Yakovlev [28] and Telerman (see Telerman et al. [26]). According to [27], they are defined as a union of closed intervals that are not necessarily disjoint, making them slightly more general from the interval unions of the present paper.

Multi-interval arithmetic is (a not separately accessible) part of the publicly available software *Unicalc* [1,22] for solving constraint satisfaction problems and nonlinear systems of equations. Another implementation of multi-intervals is described in [25]. Parallel algorithms for interval and multi-interval arithmetic are the subject of [17]. Kreinovich et al. [18] use multi-intervals to study the existence of algorithms to solve algebraic systems. No systematic performance evaluation seems to be known. Multi-intervals were also applied to the analysis of analog circuits [7], to the modeling of financial models under partial uncertainty [19], and to bit-width optimization [2].

Another variant of interval unions are the discontinuous intervals by Hyvönen [11], applied in [12,13] to simple constraint satisfaction problems and spreadsheet computations. They are disjoint unions of closed, half-open, or open intervals. In our opinion, the extra bookkeeping effort to distinguished between closed and open endpoints is not warranted in most applications.

Content: We organized this paper as follows: Sect. 2 summarizes the fundamentals of the interval union arithmetic. In Sect. 3, we define interval union matrices, vectors and linear systems of equations.

In Sect. 4, we introduce two forms of the interval union Gauss–Seidel procedure to solve (1): the partial form and the complete form. In the partial form, we update only the variable corresponding to the main diagonal entry of A at each iteration. In the complete form, we update all variables in each row.

Preconditioner heuristics are the subject of Sect. 5. Interval algorithms usually precondition the initial interval linear system to improve the quality of the solution. We extend the idea of preconditioning to interval unions and study two different preconditioning heuristics. The first one is the midpoint method: it takes the inverse of the midpoint of the hull matrix of the system $\mathcal{A}$ as the preconditioner. The second one is the Gauss–Jordan preconditioner which is based on the Gauss–Jordan elimination as discussed in [6].

Since solving large systems—due to the cost of the matrix multiplication required in the preconditioning heuristics—becomes intractable, we propose a mixed strategy that combines the original system with its preconditioned form.

Section 6 presents results of our numerical experiments. We consider randomly generated interval linear systems in order to compare traditional interval methods with the our new approach. We take linear systems with $n \in \{2, 3, 5, 10, 15, 20, 30, 50\}$ where entries of $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{x}$ have radius $r \in \{0.1, 0.2, \ldots, 2.9, 3.0\}$.

The experiment shows that interval union methods produce better enclosures than their interval counterparts. The interval union Gauss–Seidel procedure with and without preconditioners produce enclosures up to 25% sharper than those obtained by interval methods. Moreover, there are no significant differences between the execution time of intervals and interval union methods.

Notation: We denote the vector space of all $m \times n$ matrices A with real entries A_{ik} $(i = 1, \ldots, m,\ k = 1, \ldots, n)$ by $\mathbb{R}^{m\times n}$. The vector space of all column vectors v of length n and entries v_i is denoted by $\mathbb{R}^n = \mathbb{R}^{n\times 1}$.

The n-dimensional identity matrix is given by I. We denote the set of induces $1, \ldots, N$ by $1 : N$ and write $A_{i:}$ and $A_{:j}$ to denote the i-th row and j-th column of the matrix A respectively.

We assume that the reader is familiar with basic interval arithmetic. A comprehensive approach to this subject is given by [21]. For the interval arithmetic notation, we mostly follow [16]. Let $\underline{a}, \overline{a} \in \mathbb{R}$ with $\underline{a} \le \overline{a}$ then $\mathbf{a} = [\underline{a}, \overline{a}]$ denotes an interval with $\inf(\mathbf{a}) := \min(\mathbf{a}) := \underline{a}$ and $\sup(\mathbf{a}) := \max(\mathbf{a}) := \overline{a}$. The set of nonempty compact real intervals is given by

$$\mathbb{IR} := \{[\underline{a}, \overline{a}] \mid \underline{a} \le \overline{a},\ \underline{a}, \overline{a} \in \mathbb{R}\}.$$

We will allow the extremes of the intervals to assume the ideal points $-\infty$ and ∞, and define $\overline{\mathbb{IR}}$ as the set of closed real intervals and write

$$\overline{\mathbb{IR}} := \left\{ [\underline{a}, \overline{a}] \cap \mathbb{R} \mid \underline{a} \leq \overline{a},\ \underline{a}, \overline{a} \in \mathbb{R} \cup \{-\infty, \infty\} \right\},$$

The width of the interval $\mathbf{a} \in \overline{\mathbb{IR}}$ is given by $\text{wid}(\mathbf{a}) := \overline{a} - \underline{a}$, its magnitude by $|\mathbf{a}| := \max(|\underline{a}|, |\overline{a}|)$ and its mignitude by

$$\langle \mathbf{a} \rangle := \begin{cases} \min(|\underline{a}|, |\overline{a}|) & \text{if } 0 \notin [\underline{a}, \overline{a}], \\ 0 & \text{otherwise.} \end{cases}$$

The midpoint of $\mathbf{a} \in \mathbb{IR}$ is $\check{\mathbf{a}} := \text{mid}(\mathbf{a}) := (\underline{a} + \overline{a})/2$ and the radius of $\mathbf{a} \in \overline{\mathbb{IR}}$ is $\hat{\mathbf{a}} := \text{rad}(\mathbf{a}) := (\underline{a} - \overline{a})/2$. An interval is called degenerate if $\text{wid}(\mathbf{a}) = 0$.

For any set $S \subseteq \mathbb{R}$, the smallest interval containing S is called the interval hull of S and denoted by $\square S$. The notions of elementary operations between intervals and inclusion properties are the same as presented in [21]. If $\mathbf{a}, \mathbf{b} \in \mathbb{IR}$ then the extended division is defined as follows (see e.g, [23])

$$\mathbf{a}/\mathbf{b} := \begin{cases} \mathbf{a} * [1/\overline{b}, 1/\underline{b}] & \text{if } 0 \notin \mathbf{b}, \\ (-\infty, +\infty) & \text{if } 0 \in \mathbf{a} \wedge 0 \in \mathbf{b}, \\ [\overline{a}/\underline{b}, +\infty) & \text{if } \overline{a} < 0 \wedge \underline{b} < \overline{b} = 0, \\ (-\infty, \overline{a}/\overline{b}] \cup [\overline{a}/\underline{b}, +\infty) & \text{if } \overline{a} < 0 \wedge \underline{b} < 0 < \overline{b}, \\ (-\infty, \overline{a}/\overline{b}] & \text{if } \overline{a} < 0 \wedge 0 = \underline{b} < \overline{b}, \\ (-\infty, \underline{a}/\underline{b}] & \text{if } 0 < \underline{a} \wedge \underline{b} < \overline{b} = 0, \\ (-\infty, \underline{a}/\underline{b}] \cup [\underline{a}/\overline{b}, +\infty) & \text{if } 0 < \underline{a} \wedge \underline{b} < 0 < \overline{b}, \\ [\underline{a}/\overline{b}, +\infty) & \text{if } 0 < \underline{a} \wedge 0 = \underline{b} < \overline{b}, \\ \emptyset & \text{if } 0 \notin \mathbf{a} \wedge \underline{b} = \overline{b} = 0. \end{cases} \tag{2}$$

An interval vector $\mathbf{x} = [\underline{x}, \overline{x}]$ is the Cartesian product of the closed real intervals $\mathbf{x}_i := [\underline{x}_i, \overline{x}_i] \in \overline{\mathbb{IR}}$. We denote the set of all interval vectors of dimension n by $\overline{\mathbb{IR}}^n$. We denote interval matrices by capital bold letters ($\mathbf{A}$, $\mathbf{B}$, ...) and the set of all $m \times n$ interval matrices is given by $\overline{\mathbb{IR}}^{m \times n}$.

For some applications, the interval subtraction may over-estimate the range of the real computation. For example, since $-\mathbf{a} := 0 - \mathbf{a} = [-\sup(\mathbf{a}), -\inf(\mathbf{a})]$ then

$$\mathbf{b} := \mathbf{a} + (-\mathbf{a}) = [\inf(\mathbf{a}) - \sup(\mathbf{a}), \sup(\mathbf{a}) - \inf(\mathbf{a})]$$

and $\mathbf{b} = [0, 0]$ only if $\inf(\mathbf{a}) = \sup(\mathbf{a})$. In order to cope with this situation we also define inner subtraction for intervals. If $\mathbf{a}, \mathbf{b} \in \overline{\mathbb{IR}}$ then

$$\mathbf{a} \ominus \mathbf{b} := \begin{cases} [\inf(\mathbf{a}) - \inf(\mathbf{b}), \sup(\mathbf{a}) - \sup(\mathbf{b})] & \text{if } \text{wid}(\mathbf{a}) \geq \text{wid}(\mathbf{b}) \\ [\sup(\mathbf{a}) - \sup(\mathbf{b}), \inf(\mathbf{a}) - \inf(\mathbf{b})] & \text{otherwise.} \end{cases} \tag{3}$$

For a comprehensive review of inner operations, see [3].

2 Interval unions

This section introduces the basics of interval unions. For more details on the topics covered in this section see [24].

Definition 1 An interval union $\boldsymbol{u}$ of length $l(\boldsymbol{u}) := k$ is a finite set of k intervals of form

$$\boldsymbol{u} := (\mathbf{u}_1, \ldots, \mathbf{u}_k) \text{ with } \begin{array}{ll} \mathbf{u}_i \in \overline{\mathbb{IR}} & \forall\, i = 1, \ldots, k, \\ \overline{\mathbf{u}}_i < \underline{\mathbf{u}}_{i+1} & \forall\, i = 1, \ldots, k-1. \end{array}$$

We denote by $\mathcal{U}_k$ the set of all interval unions of length $\leq k$. The set of all interval unions is then $\mathcal{U} := \bigcup_{k \geq 0} \mathcal{U}_k$ where we define $\mathcal{U}_0 := \emptyset$.

If $\boldsymbol{u} \in \mathcal{U}$ is an interval union with $l(\boldsymbol{u}) = k$ then for any $x \in \mathbb{R}$ we say

$$x \in \boldsymbol{u} \quad \Leftrightarrow \quad \text{there exists a } 1 \leq i \leq k \text{ such that } x \in \mathbf{u}_i.$$

The relation above extends naturally for intervals and another interval unions, so that if $\boldsymbol{v}$ is an interval union then

$$\boldsymbol{v} \subseteq \boldsymbol{u} \quad \Leftrightarrow \quad \text{for all } \boldsymbol{v} \in \boldsymbol{v} \text{ there exists a } 1 \leq i \leq k \text{ such that } \boldsymbol{v} \subseteq \mathbf{u}_i.$$

Let S be a set of k intervals with $k < \infty$. The smallest interval union with respect to inclusion that satisfies $\mathbf{a} \subseteq \boldsymbol{u}$ for all $\mathbf{a} \in S$ is called the union creator $\mathcal{U}(S)$ of S. Formally we have

$$\mathcal{U}(S) := \{u \in \overline{\mathbb{R}} \mid u \in \cup_{i=1}^{k} S_i\}. \tag{4}$$

Clearly, $\mathcal{U}(S) \in \mathcal{U}_k$, $\mathcal{U}(\mathcal{U}(S)) = \mathcal{U}(S)$ and $S_1 \subseteq S_2$ implies $\mathcal{U}(S_1) \subseteq \mathcal{U}(S_2)$. The interval hull of a union $\boldsymbol{u} \in \mathcal{U}$ is denoted by $\square\boldsymbol{u} := [\underline{\mathbf{u}}_1,\ \overline{\mathbf{u}}_{l(\boldsymbol{u})}]$.

Let $\boldsymbol{u} \in \mathcal{U}_k \setminus \{\emptyset\}$. The magnitude and mignitude of $\boldsymbol{u}$ are given by

$$|\boldsymbol{u}| := \max(|\mathbf{u}_1|, \ldots, |\mathbf{u}_k|) = \max(|\underline{\mathbf{u}}_1|, |\overline{\mathbf{u}}_k|)$$

and

$$\langle \boldsymbol{u} \rangle := \min(\langle \mathbf{u}_1 \rangle, \ldots, \langle \mathbf{u}_k \rangle).$$

The maximum, minimum and maximum width of the non-empty interval union $\boldsymbol{u}$ are defined by

$$\max(\boldsymbol{u}) := \overline{\mathbf{u}}_k, \quad \min(\boldsymbol{u}) := \underline{\mathbf{u}}_1$$

and

$$\max \operatorname{wid}(\boldsymbol{u}) := \max(\operatorname{wid}(\mathbf{u}_1), \ldots, \operatorname{wid}(\mathbf{u}_k)).$$

The projection of the point $x \in \mathbb{R}$ into the interval union $\boldsymbol{u} \in \mathcal{U}_k$ is given by

$$\text{proj}(x, \boldsymbol{u}) := \begin{cases} x & \text{if } x \in \boldsymbol{u} \\ \overline{\mathbf{u}}_i & \text{if } x \in]\overline{\mathbf{u}}_i, \underline{\mathbf{u}}_{i+1}[\text{ and } x - \overline{\mathbf{u}}_i < \underline{\mathbf{u}}_{i+1} - x, \\ \underline{\mathbf{u}}_{i+1} & \text{if } x \in]\overline{\mathbf{u}}_i, \underline{\mathbf{u}}_{i+1}[\text{ and } x - \overline{\mathbf{u}}_i \geq \underline{\mathbf{u}}_{i+1} - x, \\ \overline{\mathbf{u}}_k & \text{if } x > \overline{\mathbf{u}}_k, \\ \underline{\mathbf{u}}_1 & \text{if } x < \underline{\mathbf{u}}_1. \end{cases}$$

Definition 2 Let $\mathbf{x} \in \mathbb{IR}$ be an interval, $\boldsymbol{u} := (\mathbf{u}_1, \ldots, \mathbf{u}_k)$ and $\boldsymbol{\vartheta} := (\mathbf{s}_1, \ldots \mathbf{s}_t)$ interval unions and let $\circ_\bullet \in \{+, -, /, *, \ominus\}$ be an elementary interval operation with the division operator given by (2) and the inner subtraction by (3).

(i) The elementary interval union operation $\circ_\star : \mathcal{U} \times \overline{\mathbb{IR}} \to \mathcal{U}$ is given by

$$\boldsymbol{u} \circ_\star \mathbf{x} := \mathcal{U}(\{\mathbf{u}_1 \circ_\bullet \mathbf{x}, \ldots, \mathbf{u}_k \circ_\bullet \mathbf{x}\}).$$

(ii) The elementary interval union operation $\circ_\star : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$ is given by

$$\boldsymbol{u} \circ \boldsymbol{\vartheta} := \mathcal{U}(\{\boldsymbol{u} \circ_\star \mathbf{s}_1, \ldots, \boldsymbol{u} \circ_\star \mathbf{s}_t\}).$$

The following result gives basic properties of interval union arithmetic, see [24].

Proposition 1 *Let $\boldsymbol{u}, \boldsymbol{v}$ and $\boldsymbol{\vartheta}$ be interval unions. Then for $\circ \in \{+, -, /, *\}$,*

$$\boldsymbol{u} \subseteq \boldsymbol{u}', \ \boldsymbol{\vartheta} \subseteq \boldsymbol{\vartheta}' \Longrightarrow \boldsymbol{u} \circ \boldsymbol{\vartheta} \subseteq \boldsymbol{u}' \circ \boldsymbol{\vartheta}' \tag{5}$$

$$\boldsymbol{u}(\boldsymbol{v} \pm \boldsymbol{\vartheta}) \subseteq \boldsymbol{u}\boldsymbol{v} \pm \boldsymbol{u}\boldsymbol{\vartheta}. \tag{6}$$

$$a(\boldsymbol{u} + \boldsymbol{v}) = a\boldsymbol{u} + a\boldsymbol{v} \quad \textit{for } a \in \mathbb{R}, \tag{7}$$

3 Interval union vectors, matrices and linear systems

Definition 3 An $m \times n$ interval union matrix is a rectangular array of interval unions with m rows and n columns. We denote interval union matrices by capital bold calligraphic letters ($\mathcal{A}$, $\mathcal{B}$, ...) and the (i, j)—element of the interval union matrix $\mathcal{A}$ is given by $\mathcal{A}_{ij}$. The set of $m \times n$ interval union matrices is given by $\mathcal{U}^{m \times n}$. In a similar way, $n \times 1$ interval union matrices are called interval union vectors. We denote interval union vectors by bold calligraphic letters ($\boldsymbol{u}$, $\boldsymbol{x}$, ...) and the set of all n-dimensional interval union vectors is given by $\mathcal{U}^n$. We denote the set of n-dimensional vectors $\boldsymbol{u}$ satisfying $l(\boldsymbol{u}_i) = k_i$ by $\mathcal{U}^n_{k_1,\ldots,k_n}$.

Given a set of interval vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p\}$, the union creator vector is denoted by $\boldsymbol{v} := \mathcal{U}(\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p\})$ where the union creator $\mathcal{U}$ defined in (4) is applied componentwise. Let $\boldsymbol{u}$ be an n-dimensional interval union vector satisfying $l(\boldsymbol{u}_i) = k_i$ and $p = \prod_{i=1}^n k_i$. If we denote the Cartesian product between two interval unions by $\times$ then the mapping $\mathcal{S} : \mathcal{U}^n_{k_1,\ldots,k_n} \to (\mathbb{IR}^n)^p$ given by

$$\mathcal{S}(\boldsymbol{v}) := \boldsymbol{v}_1 \times \boldsymbol{v}_2 \times \cdots \times \boldsymbol{v}_n$$

splits the interval union $\boldsymbol{u}$ into a set of p disjoint interval vectors. Notice that interval union vectors can be used to represent p disjoint interval vectors storing only $\sum_{i=1}^{n} k_i$ elements. This is a clear advantage over traditional interval arithmetic, especially when n is large. The mapping $\mathcal{S}$ and the definition of union creator can be naturally extended to matrices.

Interval union matrices and vectors follow the usual definition of arithmetic operations. Formally, if $\mathcal{A}, \mathcal{B} \in \mathcal{U}^{m\times n}$ and $\mathcal{C} \in \mathcal{U}^{n\times p}$ then

$$(\mathcal{A} \pm \mathcal{B})_{ij} := \mathcal{A}_{ij} \pm \mathcal{B}_{ij} \tag{8}$$

and

$$(\mathcal{A}\mathcal{C})_{ij} := \sum_{k=1}^{n} \mathcal{A}_{ik}\mathcal{C}_{kj}. \tag{9}$$

Proposition 2 *Let $\mathcal{A}, \mathcal{A}', \mathcal{B}, \mathcal{B}' \in \mathcal{U}^{m\times n}$ and $\mathcal{C}, \mathcal{C}' \in \mathcal{U}^{n\times p}$. Then*

$$\begin{aligned}
&\mathcal{A}' \subseteq \mathcal{A},\ \mathcal{B}' \subseteq \mathcal{B} \quad \Rightarrow \quad \mathcal{A}' \pm \mathcal{B}' \subseteq \mathcal{A} \pm \mathcal{B},\\
&\mathcal{A}' \subseteq \mathcal{A},\ \mathcal{C}' \subseteq \mathcal{C} \quad \Rightarrow \quad \mathcal{A}'\mathcal{C}' \subseteq \mathcal{A}\mathcal{C},\\
&\mathcal{A}(\mathcal{C} + \mathcal{C}') \subseteq \mathcal{A}\mathcal{C} + \mathcal{A}\mathcal{C}',\\
&A(\mathcal{C} + \mathcal{C}') = A\mathcal{C} + A\mathcal{C}' \quad \textit{for } A \in \mathbb{R}^{m\times n},\\
&(\mathcal{A} + \mathcal{A}')C = \mathcal{A}C + \mathcal{A}'C \quad \textit{for } C \in \mathbb{R}^{n\times p}.
\end{aligned}$$

Proof Follows from Relations (5)–(7) applied to Definitions (8) and (9). □

An interval union linear system of equations (ILLS) with coefficients $\mathcal{A} \in \mathcal{U}^{n\times n}$ and $\mathcal{b} \in \mathcal{U}^{n}$ is the family of linear equations

$$Ax = b \quad (A \in \mathcal{A}, b \in \mathcal{b}). \tag{10}$$

This paper deals only with square systems though the generalization to systems of form $m \times n$ is straightforward. The solution set of (10) is defined by

$$\Sigma(\mathcal{A}, \mathcal{b}) := \{x \in \mathbb{R}^n \mid Ax = b \text{ for some } A \in \mathcal{A}, b \in \mathcal{b}\}. \tag{11}$$

As in the interval case, (11) can be a non-convex or disconnected set. Let $\boldsymbol{x}_0 \in \mathcal{U}^n$ be an interval union vector. The truncated solution set of (10) is

$$\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x}_0 := \{x \in \boldsymbol{x}_0 \mid Ax = b \text{ for some } A \in \mathcal{A}, b \in \mathcal{b}\}. \tag{12}$$

The following proposition states that (11) is identical to the union of solution sets from the interval components of $\mathcal{A}$ and $\mathcal{b}$.

Proposition 3 *Let $\mathcal{A} \in \mathcal{U}^{n\times n}$ and $\mathcal{b} \in \mathcal{U}^{n}$. Then*

$$\bigcup_{\substack{\mathbf{A}_i \in \mathcal{S}(\mathcal{A})\\ \mathbf{b}_j \in \mathcal{S}(\mathcal{b})}} \Sigma(\mathbf{A}_i, \mathbf{b}_j) \equiv \Sigma(\mathcal{A}, \mathcal{b}).$$

Proof Let $x \in \bigcup_{\substack{\mathbf{A}_i \in \mathcal{S}(\mathcal{A}) \\ \mathbf{b}_j \in \mathcal{S}(\mathcal{b})}} \Sigma(\mathbf{A}_i, \mathbf{b}_j)$. Then for some i and j there exist $A \in \mathbf{A}_i$ and $b \in \mathbf{b}_j$ such that $Ax = b$. Since $\mathbf{A}_i \in \mathcal{A}$ and $\mathbf{b}_j \in \mathcal{b}_j$ follows that $x \in \Sigma(\mathcal{A}, \mathcal{b})$. Conversely, if $x \in \Sigma(\mathcal{A}, \mathcal{b})$ then $Ax = b$ for some $A \in \mathcal{A}$ and $b \in \mathcal{b}$. The result follows from the definition of $\mathcal{S}(\mathcal{A})$ and $\mathcal{S}(\mathcal{b})$. □

Let $\mathbf{A}$ and $\mathbf{b}$ be an interval matrix and vector respectively. The problem of finding $\square\Sigma(\mathbf{A}, \mathbf{b})$ and $\square\Sigma(\mathbf{A}, \mathbf{b}) \cap \mathbf{x}_0$ is known to be NP–hard (see, e.g., [8,18]). Therefore, Proposition 3 implies that finding $\mathcal{U}(\Sigma(\mathcal{A}, \mathcal{b}))$ and $\mathcal{U}(\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x}_0)$ are also NP–hard problems. This paper focuses on algorithms to enclose $\mathcal{U}(\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x}_0)$. Formally, we are interested in finding nontrivial vectors $\boldsymbol{y}$ (i.e $\boldsymbol{y} \neq \boldsymbol{x}_0$) satisfying

$$\mathcal{U}(\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x}_0) \subseteq \boldsymbol{x}_0 \subseteq \boldsymbol{y}.$$

Proposition 3 gives a natural approach to this problem. It consists in the application of the interval Gauss–Seidel procedure described in [9,14,21] to each system obtained by splitting $\mathcal{A}$ and $\mathcal{b}$.

Let $p = \prod_{\substack{i=1:N, \\ j=1:N}} l(\mathcal{A}_{ij})$, $q = \prod_{i=1:N} l(\mathcal{b}_i)$ and $r = \prod_{i=1:N} l(\boldsymbol{x}_i)$. The method proposed above requires the solution of $p \cdot q \cdot r$ interval linear systems of equations and does not take the structure of the interval union matrix and vector into account. The next section presents extensions of the Gauss–Seidel procedure to interval unions. We show that even in problems where $\mathcal{A} \in \mathcal{U}_1^{n \times n}$ and $\mathcal{b} \in \mathcal{U}_1^n$, interval union algorithms give better results than their interval counterparts.

The interval union matrix $\mathcal{A} \in \mathcal{U}^{n \times n}$ is said to be regular if every real matrix $A \in \mathcal{A}$ is nonsingular. The interval union inverse of a regular matrix $\mathcal{A}$ is given by

$$\mathcal{A}^{-1} := \mathcal{U}\left(\left\{A^{-1} \mid A \in \mathcal{A}\right\}\right).$$

Proposition 4 *Let $\mathcal{A} \in \mathcal{U}^{n \times n}$ be a regular matrix and $\mathcal{b} \in \mathcal{U}^{n \times 1}$. Then*

$$\Sigma(\mathcal{A}, \mathcal{b}) \subseteq \mathcal{A}^{-1}\mathcal{b} := \mathcal{U}(\{x \in \mathbb{R}^n \mid x = A^{-1}b \text{ for some } A \in \mathcal{A}, b \in \mathcal{b}\}).$$

Proof Let $x \in \Sigma(\mathcal{A}, \mathcal{b})$. Then there are $A \in \mathcal{A}$ and $b \in \mathcal{b}$ such that $Ax = b$. Since $\mathcal{A}$ is regular, A^{-1} is well defined and therefore $x \in \mathcal{A}^{-1}\mathcal{b}$. □

4 The interval union Gauss–Seidel method

Let $\mathcal{A} \in \mathcal{U}^{n \times n}$, $\mathcal{b} \in \mathcal{U}^n$ and $\boldsymbol{x}_0 \in \mathcal{U}^n$. In this section we introduce the interval union Gauss–Seidel procedure to rigorously enclose the solution set of

$$Ax = b \quad (A \in \mathcal{A}, b \in \mathcal{b}, x \in \boldsymbol{x}_0).$$

We first discuss the univariate interval union Gauss–Seidel operator and show its properties using the definitions and results from [21].

For higher dimensions, we present two versions of the Gauss–Seidel procedure. In the first version, called the partial form, we update only the variable corresponding to

$\mathcal{A}_{ii}$ in the ith row. In the second, named complete, we consider all variables at each iteration.

4.1 Interval union Gauss–Seidel operator

Let $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x} \in \mathcal{U}$. The interval union linear system in this case reduces to

$$ax = b \quad (a \in \boldsymbol{a}, b \in \boldsymbol{b}, x \in \boldsymbol{x}).$$

As in the Definition (12), the truncated solution set is given by

$$\Sigma(\boldsymbol{a}, \boldsymbol{b}) \cap \boldsymbol{x} := \{x \in \boldsymbol{x} \mid ax = b \text{ for some } a \in \boldsymbol{a}, b \in \boldsymbol{b}\}. \tag{13}$$

The univariate interval union Gauss–Seidel operator is defined by

$$\Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) := \mathcal{U}(\{x \in \boldsymbol{x} \mid ax = b \text{ for some } a \in \boldsymbol{a}, b \in \boldsymbol{b}\}). \tag{14}$$

Proposition 5 *Let* $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x} \in \mathcal{U}$ *then*

$$\Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) = \frac{\boldsymbol{b}}{\boldsymbol{a}} \cap \boldsymbol{x}, \tag{15}$$

$$\Sigma(\boldsymbol{a}, \boldsymbol{b}) \cap \boldsymbol{x} \equiv \Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) \subseteq \boldsymbol{x}, \tag{16}$$

$$\Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) \equiv \varnothing \quad \Rightarrow \quad \Sigma(\boldsymbol{a}, \boldsymbol{b}) \cap \boldsymbol{x} \equiv \varnothing, \tag{17}$$

$$0 \notin \boldsymbol{b} - \boldsymbol{a}\boldsymbol{x} \quad \Rightarrow \quad \Sigma(\boldsymbol{a}, \boldsymbol{b}) \cap \boldsymbol{x} \equiv \varnothing, \tag{18}$$

$$0 \in \boldsymbol{a},\ 0 \in \boldsymbol{b} \quad \Rightarrow \quad \Sigma(\boldsymbol{a}, \boldsymbol{b}) \cap \boldsymbol{x} \equiv \boldsymbol{x}, \tag{19}$$

$$\boldsymbol{a}' \subseteq \boldsymbol{a},\ \boldsymbol{b}' \subseteq \boldsymbol{b},\ \boldsymbol{x}' \subseteq \boldsymbol{x} \quad \Rightarrow \quad \Gamma(\boldsymbol{a}', \boldsymbol{b}', \boldsymbol{x}') \subseteq \Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}). \tag{20}$$

Proof From Definition 2, we have

$$\frac{\boldsymbol{b}}{\boldsymbol{a}} = \mathcal{U}(\{x \in \mathbb{R} \mid \tilde{a}x = \tilde{b} \text{ for some } \tilde{a} \in \boldsymbol{a}, \tilde{b} \in \boldsymbol{b}\})$$

and (15) follows from taking the intersection with $\boldsymbol{x}$. To prove (16) note that Definitions (13) and (14) imply that $\Sigma(\boldsymbol{a}, \boldsymbol{b}) \cap \boldsymbol{x} \subseteq \Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x})$. Conversely, if $x \in \Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x})$ then Definition (4) implies that x is contained in some component of $\{x \in \boldsymbol{x} \mid ax = b \text{ for some } a \in \boldsymbol{a}, b \in \boldsymbol{b}\}$ and therefore $\Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) \subseteq \Sigma(\boldsymbol{a}, \boldsymbol{b}) \cap \boldsymbol{x}$. Relation (17) follows immediately from (16). If $0 \notin \boldsymbol{b} - \boldsymbol{a}\boldsymbol{x}$ then there is no $a \in \boldsymbol{a}, b \in \boldsymbol{b}$ and $x \in \boldsymbol{x}$ such that $ax = b$. Therefore $\{x \in \boldsymbol{x} \mid ax = b \text{ for some } a \in \boldsymbol{a}, b \in \boldsymbol{b}\}$ is empty and Relation (18) holds. Relations (19) and (20) follow immediately from the extended division in Definition (2) and the inclusion property respectively. □

Let $\mathcal{A} \in \mathcal{U}^{n \times n}$, $\boldsymbol{b} \in \mathcal{U}^n$, $A \in \mathcal{A}$ and $b \in \boldsymbol{b}$. If $A_{ii} \neq 0$ and $\tilde{x} \in \boldsymbol{x}$ is an approximation of the solution of $Ax = b$ then the Gauss–Seidel iteration is given by

$$\tilde{x}_i' := \frac{b_i - \sum_{j=1}^{i-1} A_{ij}\tilde{x}_j' - \sum_{j=i+1}^{n} A_{ij}\tilde{x}_j}{A_{ii}}.$$

Since all elementary operations are inclusion isotone we have

$$\tilde{x}_i' \in \frac{\mathcal{b}_i - \sum_{j=1}^{i-1} \mathcal{A}_{ij}\boldsymbol{x}_j' - \sum_{j=i+1}^{n} \mathcal{A}_{ij}\boldsymbol{x}_j}{\mathcal{A}_{ii}}. \tag{21}$$

Note that the right side of (21) truncated to $\boldsymbol{x}$ can be written in form of the Gauss–Seidel operator Γ. Denote by $\boldsymbol{y}_i$ the improved interval union enclosure obtained from $\boldsymbol{x}_i$ and let

$$\boldsymbol{y}_i := \Gamma\left(\mathcal{A}_{ii}, \mathcal{b}_i - \sum_{j=1}^{i-1} \mathcal{A}_{ij}\boldsymbol{y}_j - \sum_{j=i+1}^{n} \mathcal{A}_{ij}\boldsymbol{x}_j, \boldsymbol{x}_i.\right) \tag{22}$$

Finally, we denote by $\Gamma(\mathcal{A}, \mathcal{b}, \boldsymbol{x})$ the Cartesian product of variables $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ and we have the following result

Proposition 6 *Let $\mathcal{A} \in \mathcal{U}^{n\times n}$, $\mathcal{b} \in \mathcal{U}^n$ and $\boldsymbol{x} \in \mathcal{U}^n$. Then*

$$\mathcal{A}' \subseteq \mathcal{A},\ \mathcal{b}' \subseteq \mathcal{b},\ \boldsymbol{x}' \subseteq \boldsymbol{x} \quad \Rightarrow \quad \Gamma(\mathcal{A}', \mathcal{b}', \boldsymbol{x}') \subseteq \Gamma(\mathcal{A}, \mathcal{b}, \boldsymbol{x}). \tag{23}$$

$$\tilde{x} \in \Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x} \quad \Rightarrow \quad \tilde{x} \in \Gamma(\mathcal{A}, \mathcal{b}, \boldsymbol{x}), \tag{24}$$

Proof Relation (23) follows from the component-wise application of (20). Since $\tilde{x} \in \Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x}$, there are $A \in \mathcal{A}$ and $b \in \mathcal{b}$ such that $A\tilde{x} = b$. Relation (24) follows from (21) and Definition (22). □

4.2 Partial form

We implement the partial Gauss–Seidel procedure that is based on the Gauss–Seidel operator (15) in Algorithm 1. We incorporate Relations (18) and (19) to the algorithm in order to avoid unnecessary divisions. We stop the algorithm when the following criteria are reached for $\epsilon_{Abs} > 0$ and $\epsilon_{Rel} > 0$

$$\max \mathrm{wid}(\boldsymbol{x}) - \max \mathrm{wid}(\boldsymbol{y}) < \epsilon_{Abs} \quad \text{and} \quad 1 - \frac{\max \mathrm{wid}(\boldsymbol{y})}{\max \mathrm{wid}(\boldsymbol{x})} < \epsilon_{Rel}. \tag{25}$$

Example 1 Let $\mathcal{A}$, $\mathcal{b}$ and $\boldsymbol{x}$ be given by

$$\mathcal{A} = \begin{pmatrix} \{[-2.0, 2.0]\} & \{[0.5, 1.0]\} \\ \{[0.5, 1.0]\} & \{[-3.0, 3.0]\} \end{pmatrix}, \quad \mathcal{b} = \begin{pmatrix} \{[8.0, 8.0]\} \\ \{[12.0, 12.0]\} \end{pmatrix}$$

and $\boldsymbol{x} = (\{[-3, 2]\}, \{[-5, 6]\})^T$. The solution set $\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x}$ for this problem as well as the enclosures obtained by interval and interval union algorithms are shown in Fig. 1. Since $\mathcal{A}_{ij}, \mathcal{b}_i, \boldsymbol{x}_i \in \mathcal{U}_1$ for every i and j, we can compare the performance of Algorithm 1 with the traditional interval Gauss–Seidel procedure.

Algorithm 1 Interval union Gauss–Seidel—Partial update

Input: The interval union matrix $\mathcal{A}$ and interval union vectors $\mathcal{b}$ and $\boldsymbol{x}$. The tolerances $\epsilon_{Abs} > 0$ and $\epsilon_{Rel} > 0$. The maximum number of iterations K.
Output: The interval union vector $\boldsymbol{y}$ such that $\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x} \subseteq \boldsymbol{y} \subseteq \boldsymbol{x}$.

for $k = 1, \ldots K$ **do**
  for $i = 1, \ldots, n$ **do**
    $\mathcal{s} \leftarrow \mathcal{b}_i - \sum_{j=1}^{i-1} \mathcal{A}_{ij}\boldsymbol{y}_j - \sum_{j=i+1}^{n} \mathcal{A}_{ij}\boldsymbol{x}_j$;
    if $0 \notin \mathcal{s} - \mathcal{A}_{ii}\boldsymbol{x}_i$ **then**
      $\boldsymbol{y} \leftarrow \varnothing$;
      return $\boldsymbol{y}$;
    end if
    if $0 \in \mathcal{s}, 0 \in \mathcal{A}_{ii}$ **then**
      $\boldsymbol{y}_i \leftarrow \boldsymbol{x}_i$;
      continue;
    end if
    $\boldsymbol{y}_i \leftarrow \Gamma(\mathcal{A}_{ii}, \mathcal{s}, \boldsymbol{x}_i)$;
    if $\boldsymbol{y}_i == \varnothing$ **then**
      $\boldsymbol{y} \leftarrow \varnothing$;
      return $\boldsymbol{y}$;
    end if
  end for
  if $\max \mathrm{wid}(\boldsymbol{x}) - \max \mathrm{wid}(\boldsymbol{y}) < \epsilon_{Abs}$ and $1 - \frac{\max \mathrm{wid}(\boldsymbol{y})}{\max \mathrm{wid}(\boldsymbol{x})} < \epsilon_{Rel}$ **then**
    return $\boldsymbol{y}$;
  end if
  $\boldsymbol{x} \leftarrow \boldsymbol{y}$;
end for
return $\boldsymbol{y}$;

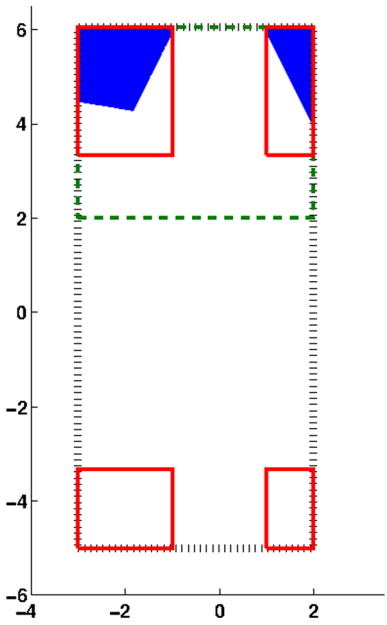

Fig. 1 Solution set of Example 1 and enclosures obtained with the partial form of the Gauss–Seidel procedure. The initial box is given in the *outer dotted line*. The enclosure obtained with one iteration of the interval Gauss–Seidel is given by the *dashed box* (an improvement of 63% in volume and 54% in the maximum width w.r.t the the initial box). The enclosures obtained by the interval union Gauss–Seidel procedure with $K = 1$ are given in *solid lines* (an improvement of 76% in volume and 60% in the maximum width w.r.t the initial box)

The interval Gauss–Seidel procedure applied to the permuted matrix gives

$$\mathbf{x}_I = ([-3, 2], [2, 6])^T .$$

This is an improvement of 63% in volume and 54% in the maximum width compared to the initial box. We describe now the application of Algorithm 1 to the problem. In this case, the interval union Gauss–Seidel procedure solves the problem directly, without any permutation.

In the first iteration ($i = 1$) we have

$$\mathcal{s} = \{[8.0, 8.0]\} - \{[0.5, 1.0]\}\{[-5, 6]\} = \{[2.0, 13.0]\} .$$

Since $\mathcal{A}_{11}\boldsymbol{x}_1 = \{[-6.0, 6.0]\}$ follows that $0 \in \mathcal{s} - \mathcal{A}_{11}\boldsymbol{x}_1$ and $0 \notin \mathcal{s}$. Applying the Gauss–Seidel operator we obtain

$$\boldsymbol{y}_1 = \{[-3, -1], [1, 2]\}$$

and conclude the first iteration. The second iteration ($i = 2$) starts with

$$\mathcal{s} = \{[10, 11.5], [12.5, 15]\} .$$

In this case, $\mathcal{A}_2\boldsymbol{x}_2 = \{[-18, 18]\}$ and applying the Gauss–Seidel operator we have

$$\boldsymbol{y}_2 = \{[-5, -3.3333], [3.3333, 6]\}$$

and we finish the internal loop. The interval union Gauss–Seidel procedure produces 4 disjoint boxes representing an improvement of 76% in volume and 60% in maximum width compared to the initial box. There is no improvement in $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ if we set $K = 2$ in Algorithm 1.

4.3 Complete form

Algorithm 1 is said to be partial since it considers only the variable corresponding to the diagonal entry at each iteration. In the following, we present the complete Gauss–Seidel procedure. It applies the Gauss–Seidel operator to all variables at each iteration.

The solution set obtained by the complete Gauss–Seidel procedure is at least as good as those given by the partial version. On the other hand, the complete procedure requires more calculations and may be prohibitive in higher dimensions.

In order to improve the efficiency of the complete Gauss–Seidel, we apply inner subtraction to each row. Note that the Gauss–Seidel operator applied to the variable $\boldsymbol{x}_j$ in the ith row is given by

$$\Gamma\left(\mathcal{A}_{ij}, \mathcal{b}_i - \sum_{\substack{k=1 \\ k\neq j}}^{n} \mathcal{A}_{ik}\boldsymbol{x}_k, \boldsymbol{x}_j\right).$$

Considering the auxiliary variable $\mathcal{s} := \mathcal{b}_i - \sum_{k=1}^{n} \mathcal{A}_{ik}\boldsymbol{x}_k$, the Gauss–Seidel operation becomes

$$\Gamma\left(\mathcal{A}_{ij}, \mathcal{s} \ominus \mathbf{A}_{ij}\boldsymbol{x}_j, \boldsymbol{x}_j\right)$$

where $\ominus$ is interval union generalization of the inner subtraction defined by Equation (3). Algorithm 2 gives the complete form of the interval union Gauss–Seidel procedure. It also implements Relations (18) and (19) to avoid unnecessary divisions. The stopping criteria adopted to this algorithm are the same as in the Algorithm 1.

Algorithm 2 Interval union Gauss–Seidel—Complete update

Input: The interval union matrix $\mathcal{A}$ and interval union vectors $\mathcal{b}$ and $\boldsymbol{x}$. The tolerances $\epsilon_{Abs} > 0$ and $\epsilon_{Rel} > 0$. The maximum number of iterations K.
Output: The interval union vector $\boldsymbol{y}$ such that $\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x} \subseteq \boldsymbol{y} \subseteq \boldsymbol{x}$.

for $k = 1, \ldots, K$ **do**
  for $i = 1, \ldots, n$ **do**
    $\mathcal{s} \leftarrow \mathcal{b}_i - \sum_{k=1}^{n} \mathcal{A}_{ik}\boldsymbol{x}_k$;
    for $j = 1, \ldots, n$ **do**
      $\mathcal{s}_j \leftarrow \mathcal{s} \ominus \mathcal{A}_{ij}\boldsymbol{x}_j$;
      if $0 \notin \mathcal{s}_j - \mathcal{A}_{ii}\boldsymbol{x}_i$ **then**
        $\boldsymbol{y} \leftarrow \varnothing$;
        return $\boldsymbol{y}$;
      end if
      if $0 \in \mathcal{s}_j, 0 \in \mathcal{A}_{ij}$ **then**
        $\boldsymbol{y}_j \leftarrow \boldsymbol{x}_j$;
        continue;
      end if
      $\boldsymbol{y}_j \leftarrow \Gamma\left(\mathcal{A}_{ij}, \mathcal{s}_j, \boldsymbol{x}_j\right)$;
      if $\boldsymbol{y}_j == \varnothing$ **then**
        $\boldsymbol{y} \leftarrow \varnothing$;
        return $\boldsymbol{y}$;
      end if
    end for
  end for
  if $\max \mathrm{wid}(\boldsymbol{x}) - \max \mathrm{wid}(\boldsymbol{y}) < \epsilon_{Abs}$ and $1 - \frac{\max \mathrm{wid}(\boldsymbol{y})}{\max \mathrm{wid}(\boldsymbol{x})} < \epsilon_{Rel}$ **then**
    return $\boldsymbol{y}$;
  end if
  $\boldsymbol{x} \leftarrow \boldsymbol{y}$;
end for
return $\boldsymbol{y}$;

Example 2 (Example 1 revisited) Let $\mathcal{A}$, $\mathcal{b}$ and $\boldsymbol{x}$ be given as in Example 1. The solution sets obtained by the application of the complete form of the interval and interval union Gauss–Seidel procedures are given in Fig. 2.

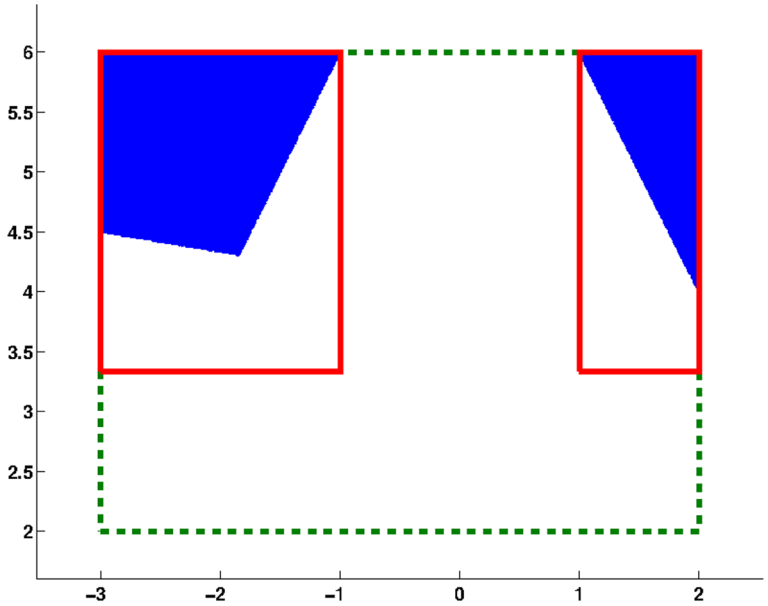

Fig. 2 Solution set of Example 1 with complete interval union Gauss–Seidel. The solution obtained with one iteration of the interval Gauss–Seidel is given by the *dashed box*. The solution obtained by the interval union Gauss–Seidel with $K = 1$ is given in *solid lines*

The complete interval Gauss–Seidel procedure produces the same result as the partial form and therefore $\mathbf{x}_I = ([-3, 2], [2, 6])^T$.

Applying the complete form with interval unions we obtain

$$y_1 = \{[-3, -1], [3.3333, 6]\} \text{ and } y_2 = \{[1, 2], [3.3333, 6]\}$$

representing an improvement of 85% in volume and 72% in the maximum width compared to the initial box. Note that the complete form removes two interval boxes that do not contain any solution and that could not be deleted with the partial form (see Figs. 1 and 2). Again, there is no improvement in y_1 and y_2 if we set $K = 2$ in Algorithm 2.

4.4 Gap filling

The number of boxes produced by Algorithms 1 and 2 may increase exponentially by the number of divisions with intervals containing zero. A similar phenomenon was already observed by Hyvönen [11] for the propagation of discontinuous intervals; however, the remedy proposed there—simply to take the interval hull—unnecessarily discards useful information. As a more flexible remedy, [24] introduced the notion of gap filling. In this section we describe a gap filling strategy that (among several strategies tried) proved useful for the interval union Gauss–Seidel procedure.

A gap filling is a mapping $g : \mathcal{U}_k \to \mathcal{U}_k$ satisfying $\boldsymbol{x} \subseteq g(\boldsymbol{x})$ and $\square\boldsymbol{x} \equiv \square g(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathcal{U}_k$. Two possible, trivial gap filling would be $g(\boldsymbol{x}) = \boldsymbol{x}$ and $g(\boldsymbol{x}) = \square\boldsymbol{x}$. The gap filling $g(\boldsymbol{x}) = \boldsymbol{x}$ however does not avoid the exponential increase on the number of boxes produced by Algorithms 1 and 2. In contrary, the gap filling $g(\boldsymbol{x}) = \square\boldsymbol{x}$ do not lead an increased number of boxes, but also loses valuable gap information. Therefore in Algorithm 3 we propose a gap filling that controls the maximum number of gaps produced.

Algorithm 3 Gap filling: maximum number of gaps

Input: The interval union $\boldsymbol{x}$ and the maximum number of gaps n.
Output: the interval union $\boldsymbol{y}$ such that $\boldsymbol{x} \subseteq \boldsymbol{y}$ and $\square\boldsymbol{x} \equiv \square\boldsymbol{y}$.
1: **if** $l(\boldsymbol{x}) \leq n$ **then**
2: return $\boldsymbol{x}$;
3: **end if**
4: **while** $l(\boldsymbol{x}) > n$ **do**
5: Find the gap $\boldsymbol{g}$ of $\boldsymbol{x}$ with smallest width;
6: $\boldsymbol{x} \leftarrow \boldsymbol{x} \cup \boldsymbol{g}$;
7: **end while**

Algorithm 3 can be modified to also handle interval union vectors and matrices. In this case we look for the gap with the smallest width in the whole vector or matrix and fill it in the while loop (Lines 4–7) of the algorithm. Note that using a multi-map data structure in the implementation of the gap filling for vectors and matrices allows faster access to the smallest gaps, improving the overall speed of the algorithm.

5 Preconditioners

In this section we present the midpoint and Gauss–Jordan preconditioners for interval union linear systems. It is usually necessary to precondition interval union linear systems of equations to obtain meaningful bounds on the solution set. A preconditioner is any real non-singular matrix C.

Given $\mathcal{A} \in \mathcal{U}^{n\times n}$, $\boldsymbol{b} \in \mathcal{U}^n$ and $\boldsymbol{x}_0 \in \mathcal{U}^n$, we are interested in preconditioners satisfying

$$\Sigma(\mathcal{A}, \boldsymbol{b}) \cap \boldsymbol{x}_0 \subseteq \Gamma(C\mathcal{A}, C\boldsymbol{b}, \boldsymbol{x}_0) \subseteq \Gamma(\mathcal{A}, \boldsymbol{b}, \boldsymbol{x}_0).$$

Since any non-singular matrix can be chosen as preconditioner, there are several heuristics to determine C according to the application. In the interval case, the midpoint preconditioner is the common choice in a number of problems. Optimal linear programming preconditioners are designed by [14] in the context of the interval Newton operator and the Gauss–Jordan preconditioner is proposed by [6]. See also [10] and [15] for recent methods on optimal preconditioning.

The midpoint preconditioner in the interval union framework takes the form

$$C = \text{proj}(\text{mid}(\square\mathcal{A}), \mathcal{A})^{-1}$$

where the midpoint and proj operators are applied component-wise.

The Gauss–Jordan preconditioner is based on the real Gauss–Jordan elimination algorithm with pivot search. Given a square matrix $A \in \mathbb{R}^{n\times n}$, the algorithm computes C and a permutation matrix $P \in \mathbb{R}^{n\times n}$ such that

$$CAP = I.$$

In this paper we take $A = \text{proj}(\text{mid}(\square\mathcal{A}), \mathcal{A})$. It is worth to note that due to the permutation matrix we apply the Gauss–Seidel procedure to the modified problem

$$My = r \quad (M \in C\mathcal{A}P, r \in C\mathcal{b}, y \in \boldsymbol{x}_0 P). \tag{26}$$

Example 3 Let $\mathcal{A}$, $\mathcal{b}$ and $\boldsymbol{x}$ be given by

$$\mathcal{A} = \begin{pmatrix} \{[0.00, 0.14]\} & \{[0.54, 1.23]\} \\ \{[-0.06, 1.67]\} & \{[0.31, 1.02]\} \end{pmatrix}, \quad \mathcal{b} = \begin{pmatrix} \{[1.73, 1.73]\} \\ \{[6.76, 6.76]\} \end{pmatrix}$$

and $\boldsymbol{x} = (\{[2.5, 3.5]\}, \{[3.0, 4.0]\})^T$. Applying the partial form of the Gauss–Seidel operator to each variable without preconditioner gives

$$\boldsymbol{y}_1 = \{[2.5, 3.5]\} \cap \mathcal{U}\left(\frac{[-3.19, 0.11]}{[0, 0.14]}\right) = \{[2.5, 3.5]\}$$

and

$$\boldsymbol{y}_2 = \{[3.0, 4.0]\} \cap \mathcal{U}\left(\frac{[0.9150, 6.9701]}{[0.31, 1.02]}\right) = \{[3.0, 4.0]\}$$

On the other hand, the Gauss–Jordan preconditioner presented in this section gives

$$C = \begin{pmatrix} 1.20894 & -0.10512 \\ -0.99869 & 1.32908 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The permuted system is given by

$$\mathcal{m} = \begin{pmatrix} \{[0.545, 1.454]\} & \{[-0.175, 0.175]\} \\ \{[-0.816, 0.816]\} & \{[-0.219, 2.219]\} \end{pmatrix}, \quad \mathcal{r} = \begin{pmatrix} \{[1.380, 1.380]\} \\ \{[7.256, 7.256]\} \end{pmatrix}$$

and $\boldsymbol{x}' = (\{[3.0, 4.0]\}, \{[2.5, 3.5]\})^T$. We obtain the following bounds with the Gauss–Seidel procedure applied to the permuted system

$$\boldsymbol{y}'_1 = \{[3.0, 4.0]\} \cap \mathcal{U}\left(\frac{[0.7663, 1.9953]}{[0.5455, 1.4545]}\right) = \{[3.0, 3.65]\}$$

and

$$y_2' = \{[2.5, 3.5]\} \cap \mathcal{U}\left(\frac{[4.2712, 10.2425]}{[-0.2196, 2.2196]}\right) = \{[2.5, 3.5]\}.$$

The new enclosure represents an improvement of 34% in volume compared to the initial box. Note that we must apply the inverse permute to y_1' and y_2' in order to obtain the correct enclosure. In this example, the same result would be obtained by applying the midpoint preconditioner.

The matrix C is dense in general. Therefore, preconditioner strategies may be prohibitive in large linear systems of equations. Moreover, systems of form (26) may overestimate the solution set in some problems. For example, let $\mathcal{A}$, $\mathcal{b}$ and $\boldsymbol{x}$ be given by

$$\mathcal{A} = \begin{pmatrix} \{[-2.0, 2.0]\} & \{[0.5, 1.0]\} \\ \{[0.5, 1.0]\} & \{[2.0, 3.0]\} \end{pmatrix}, \quad \mathcal{b} = \begin{pmatrix} \{[6.0, 6.0]\} \\ \{[6.0, 6.0]\} \end{pmatrix}$$

and $\boldsymbol{x} = (\{[-3, 2]\}, \{[-6, 6]\})^T$. If we apply the Algorithm 1 with $\epsilon_{Abs} = \epsilon_{Rel} = 10^{-4}$ and $K = 1$ to the original system, we obtain $y_{\text{UNP}} = (\{[-3, 2]\}, \{[1.333, 4.5]\})^T$. The resulting interval union vector represents an improvement of 73% in volume when compared with the initial box. On the other hand, applying the Algorithm 1 with the same parameters to the corresponding system of form (26), obtained with the Gauss–Jordan preconditioning gives $y_{\text{GJ}} = (\{[-3, -0.7826], [0.9729, 2]\}, \{[0, 6]\})^T$. The solution vector y_{GJ} is an improvement of 67% in volume when compared with the initial box.

We introduce a mixed strategy that combines the original linear system with its preconditioned form. Given $\mathcal{A} \in \mathcal{U}^{n\times n}$, $\mathcal{b} \in \mathcal{U}^n$ and $\boldsymbol{x} \in \mathcal{U}^n$ we alternate between the solution of the original system and the preconditioned form (26) until one of the following: (1) we prove that there is no solution in $\boldsymbol{x}$, (2) the maximum number of iterations is reached, or (3) we have not enough gain in the last solution of both the original and preconditioned systems.

Algorithm 4 implements the mixed strategy using the partial or complete forms of the interval union Gauss–Seidel procedure. The boolean variables gainUnprec and gainGS control the next iteration of the algorithm. If both are false then neither the Gauss–Seidel procedure without preconditioning nor the same procedure with preconditioning gave a substantial improvement on the current box and the mixed algorithms stops. Algorithm 4 can be modified to apply the midpoint preconditioner instead of the Gauss–Jordan method.

6 Numerical experiments

In this section we perform numerical experiments to compare the interval union Gauss–Seidel procedure with its interval counterpart. We consider the partial and complete forms of the Gauss–Seidel procedure as well as the midpoint and the Gauss–Jordan

Algorithm 4 Mixed preconditioner strategy

Input: The interval union matrix $\mathcal{A}$ and interval union vectors $\mathcal{b}$ and $\boldsymbol{x}$. The tolerances $\epsilon_{Abs} > 0$ and $\epsilon_{Rel} > 0$. The maximum number of iterations K.
Output: The interval union vector $\boldsymbol{y}$ such that $\Sigma(\mathcal{A}, \mathcal{b}) \cap \boldsymbol{x} \subseteq \boldsymbol{y} \subseteq \boldsymbol{x}$.

$\boldsymbol{y} \leftarrow \text{GS}(\mathcal{A}, \mathcal{b}, \boldsymbol{x}, \epsilon_{Abs}, \epsilon_{Rel}, 1)$;
if $\boldsymbol{y} == \varnothing$ **then**
  return $\boldsymbol{y}$;
end if
gainUnprec $\leftarrow$ true;
if Relations (25) are satisfied **then**
  gainUnprec $\leftarrow$ false;
end if
$(C, P) \leftarrow$ Gauss-Jordan(A);
$\mathcal{M} \leftarrow C\mathcal{A}P$; $\mathcal{r} \leftarrow C\mathcal{b}$;
gainGJ $\leftarrow$ true; iterateGJ $\leftarrow$ true;
for $i = 2, \ldots, K$ **do**
  if iterateGJ == true **then**
    $\boldsymbol{y} \leftarrow \text{GS}(\mathcal{M}, \mathcal{r}, \boldsymbol{y}P, \epsilon_{Abs}, \epsilon_{Rel}, 1)$;
    $\boldsymbol{y} \leftarrow \boldsymbol{y}P^{-1}$;
    iterateGJ $\leftarrow$ false;
    if Relations (25) are satisfied **then**
      gainGJ $\leftarrow$ false;
    else
      gainGJ $\leftarrow$ true; gainUnprec $\leftarrow$ true;
    end if
  else
    $\boldsymbol{y} \leftarrow \text{GS}(\mathcal{A}, \mathcal{b}, \boldsymbol{y}, \epsilon_{Abs}, \epsilon_{Rel}, 1)$;
    iterateGJ $\leftarrow$ true;
    if Relations (25) are satisfied **then**
      gainUnprec $\leftarrow$ false;
    else
      gainGJ $\leftarrow$ true; gain $\leftarrow$ true;
    end if
  end if
  if $\boldsymbol{y} == \varnothing$ or (gainGJ == false and gainUnprec == false) **then**
    return $\boldsymbol{y}$;
  end if
end for
return $\boldsymbol{y}$;

preconditioners. In this test, we take only interval linear systems of equations into account. The experiment is described in Algorithm 5.

In this section, we set the parameters of the Algorithms 1 and 2 as $\epsilon_{Abs} = \epsilon_{Rel} = 10^{-4}$ and $K = 2$ for the partial form and $K = 1$ for the complete form. In the gap filling Algorithm 3, we set the maximum number of gaps in an interval union as $g = 2$ and the maximum number of boxes for interval union vectors to 64.

In Algorithm 5, we set $\mathcal{R} := \{0.1, 0.2, \ldots, 2.9, 3.0\}$, $\mathcal{N} := \{2, 3, 5, 10, 15, 20, 30, 50\}$ and $T = 100$. The entries of $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{x}$ have radius given by $r \in \mathcal{R}$ and satisfy the rules described in Table 1.

Figures 3, 4 and 5 summarize the results of the experiment. For each point in these graphs we have the average of the maximum width gained with the methods in a set of 4000 problems taken at random (100 for each $n \in \mathcal{N}$ and for each one of the 5 cases

Algorithm 5 Performance analysis

Input: The set of radii $\mathcal{R}$, the set of sizes $\mathcal{N}$ and the number of trials T.
Output: The average maximum width gained and elapsed time for each combination of Gauss-Seidel procedure form and preconditioner.

```
for r ∈ R do
  for n ∈ N do
    for i = 1 : T do
      Generate the random matrix A of size n and such that rad(A) = r;
      Generate vectors b and x of size n and such that rad(b) = rad(x) = r;
      Run the instance with all variants of the Gauss-Seidel procedure;
      Save the data;
    end for
  end for
end for
```

Table 1 Description of the processes that generate matrices and vectors **A**, **b** and **x**

Case	**A**	**b**	**x**
1	$\check{\mathbf{A}}_{ii} \in [-1, 1], \check{\mathbf{A}}_{ij} \in [-5, 5]$	$\check{\mathbf{b}} \in [-1, 1]$	$\check{\mathbf{A}}^{-1}\check{\mathbf{b}} \in \mathbf{x}$
2	$\check{\mathbf{A}}_{ii} \in [-5, 5], \check{\mathbf{A}}_{ij} \in [-1, 1]$	$\check{\mathbf{b}} \in [-1, 1]$	$\check{\mathbf{A}}^{-1}\check{\mathbf{b}} \in \mathbf{x}$
3	$\check{\mathbf{A}}_{ij} \in [-1, 1]$	$\check{\mathbf{b}} \in [-1, 1]$	$\check{\mathbf{x}} \in [-1, 1]$
4	$\check{\mathbf{A}}_{ij} \in [-1, 1]$	$\check{\mathbf{b}} \in [n, 10n]$	$\check{\mathbf{x}} \in [-1, 1]$
5	$\check{\mathbf{A}}_{ii} \in [-1, 1], \check{\mathbf{A}}_{ij} \in [-5, 5]$	$\check{\mathbf{b}} \in [n, 10n]$	$\check{\mathbf{x}} \in [-1, 1]$

The number n stands for the dimension of the linear system

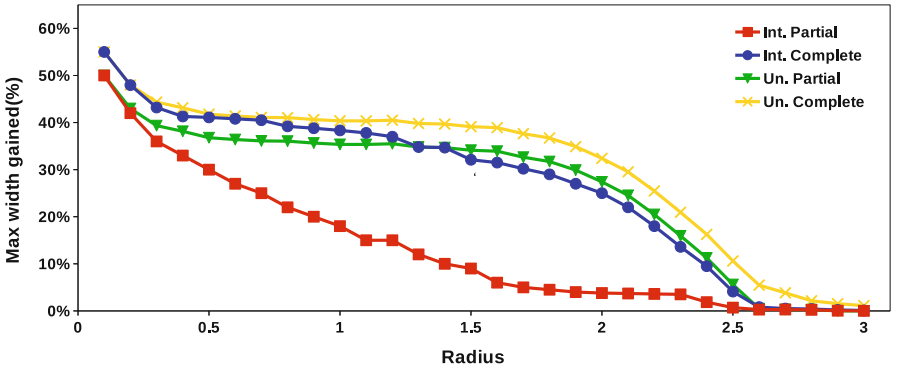

Fig. 3 Average maximum width gained with each method in problems of size $n \in \{2, 5, 10, 15, 20, 30, 50\}$. All possible forms of the Gauss–Seidel procedure without preconditioning

displayed in Table 1). Tables 2 and 3 show the average elapsed time for each method. All the algorithms were implemented in *JGloptlab* [4], a Java implementation of the state of the art global optimization algorithms. We run the experiment in a *corei7* processor with 6 Gb of RAM memory.

It is clear that the interval union Gauss–Seidel procedure produces better enclosures than the interval method. Tables 2 and 3 show that there are no significant differences

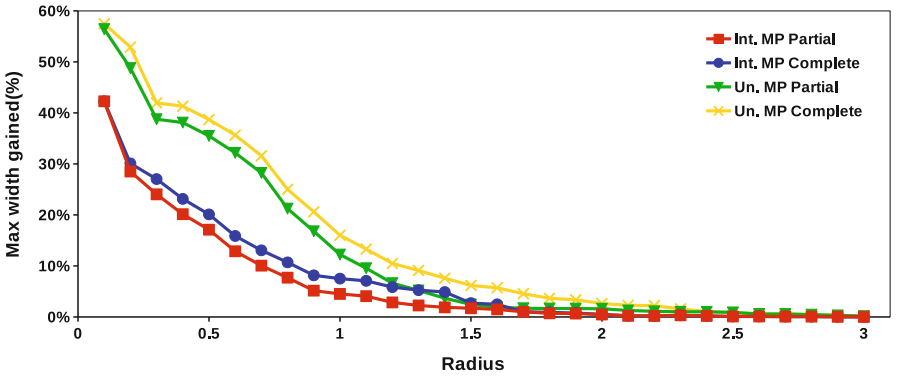

Fig. 4 Average maximum width gained with each method in problems of size $n \in \{2, 5, 10, 15, 20, 30, 50\}$. All possible forms of the Gauss–Seidel procedure with the midpoint preconditioner

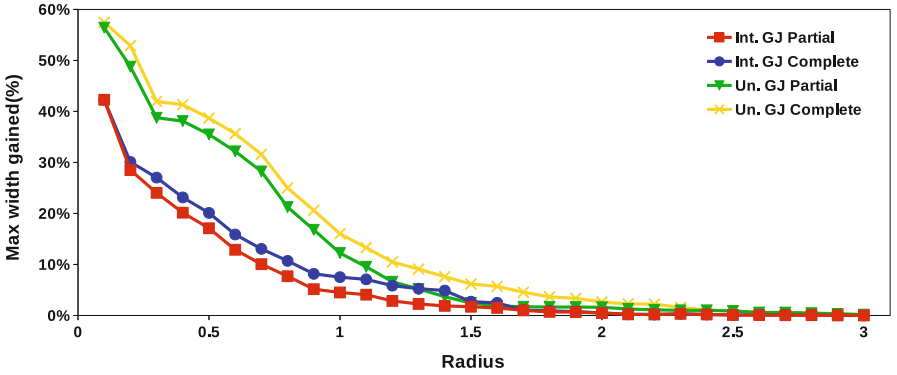

Fig. 5 Average maximum width gained with each method in problems of size $n \in \{2, 5, 10, 15, 20, 30, 50\}$. All possible forms of the Gauss–Seidel procedure with the Gauss–Jordan preconditioner

Table 2 Average elapsed time (in seconds) for the partial form

n	Interval			Union		
	Unprec.	Midpoint	Gauss–Jordan	Unprec.	Midpoint	Gauss–Jordan
15	0.001	0.001	0.001	0.001	0.03	0.012
20	0.001	0.39	0.404	0.001	0.538	0.515
30	0.001	3.135	3.23	0.001	3.621	3.726
50	0.001	15.806	16.752	0.001	16.72	17.772

Unprec. stands for algorithms without preconditioning

between the execution time of the Gauss–Seidel procedure with intervals and interval unions.

Figure 6 show the effect of the dimension on the quality of the computed enclosures considering the Gauss–Jordan preconditioner.

Table 3 Average elapsed time (in seconds) for the complete form

n	Interval			Union		
	Unprec.	Midpoint	Gauss–Jordan	Unprec.	Midpoint	Gauss–Jordan
15	0.001	0.003	0.001	0.001	0.041	0.043
20	0.001	0.396	0.408	0.001	0.696	0.655
30	0.009	3.163	3.266	0.001	3.812	3.898
50	0.004	15.986	16.89	0.001	17.689	18.644

Unprec. stands for algorithms without preconditioning

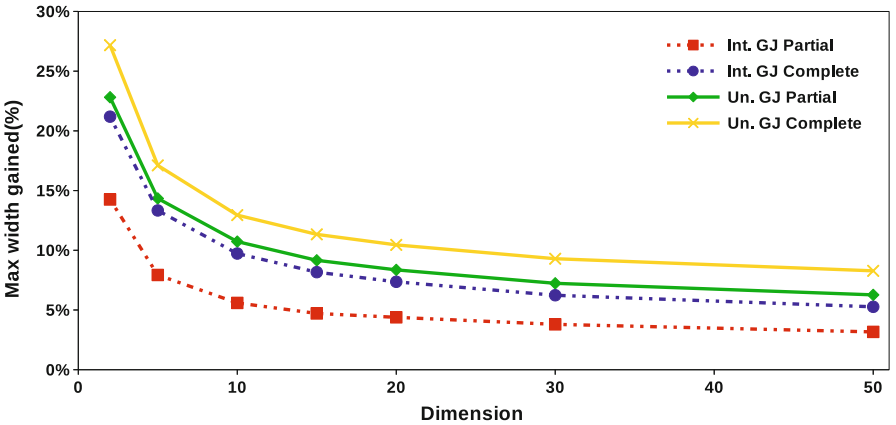

Fig. 6 Average maximum width gained with each method in the same problems used on Fig. 5 as function of the dimension

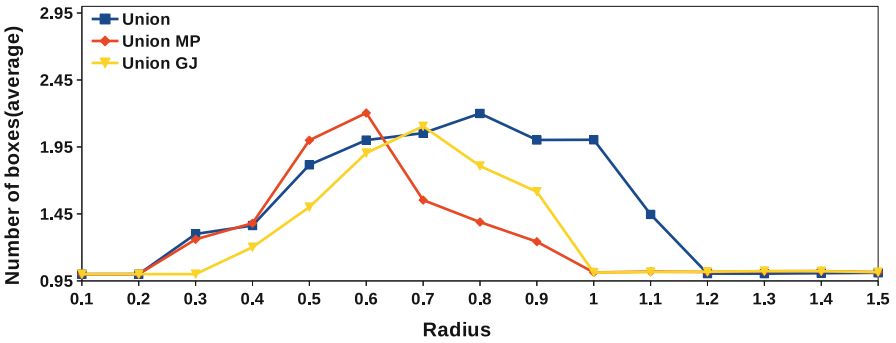

Fig. 7 Maximum number of boxes generated during the execution of the partial form of the interval union Gauss–Seidel procedure in average. *MP* stands for the midpoint precoditioner and *GJ* denotes the method with the Gauss–Jordan preconditioner

The exponential increase in the number of boxes produced by Algorithms 1 and 2 is one of the main concerns regarding the use of the interval union arithmetic. We note that the maximum number of boxes produced in during the interval union Gauss–Seidel procedure is, in average, never greater than 3 as showed by Figs. 7 and 8.

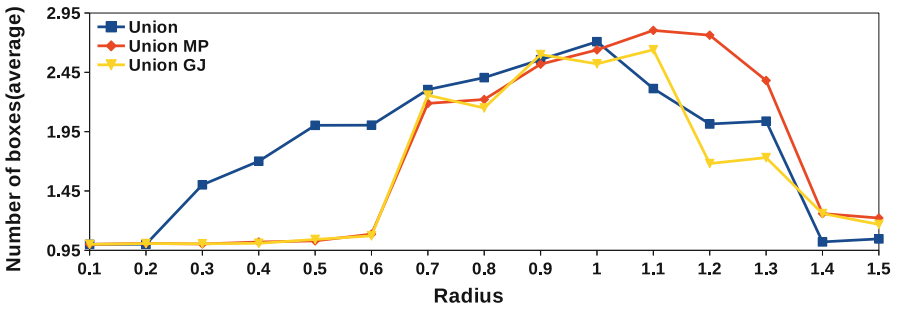

Fig. 8 Maximum number of boxes generated during the execution of the complete form of the interval union Gauss–Seidel procedure in average. *MP* stands for the midpoint preconditioner and *GJ* denotes the method with the Gauss–Jordan preconditioner

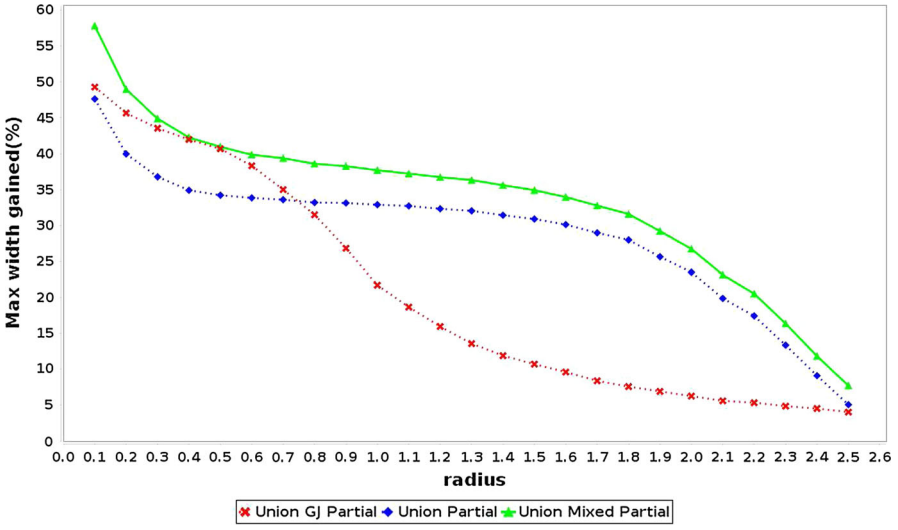

Fig. 9 Average maximum width gained with each method in problems of size $n \in \{2, 5, 10, 15, 20, 30, 50\}$. The figure displays the partial form of the interval union Gauss–Seidel without preconditioner, with the Gauss–Jordan preconditioner and the mixed strategy

Moreover, we reach the maximum number of boxes prescribed in Algorithm 3 during the execution of the procedure only in 10% of the 120,000 instances with the complete form. We never reach the maximum number of boxes with the partial form.

6.1 Mixed preconditioner strategy

It is clear from Tables 2 and 3 that the interval union Gauss–Seidel procedure without preconditioner is several times faster than the same method with preconditioners. Moreover, there are problems where the preconditioner leads to poorer bounds than the solution of the original system.

We finish this section comparing Algorithms 1 and 2 with the mixed strategy proposed in Algorithm 4. In this experiment we set the parameters of all algorithms as

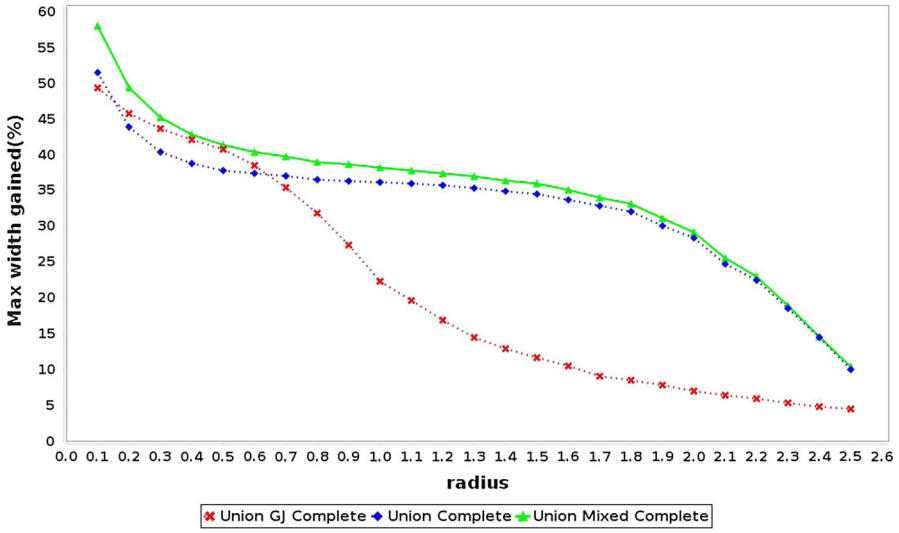

Fig. 10 Average maximum width gained with each method in problems of size $n \in \{2, 5, 10, 15, 20, 30, 50\}$. The figure displays the complete form of the interval union Gauss–Seidel without preconditioner, with the Gauss–Jordan preconditioner and the mixed strategy

Table 4 Average elapsed time (in seconds) for the partial and complete forms

n	Partial			Complete		
	Unprec.	Gauss–Jordan	Mixed	Unprec.	Gauss–Jordan	Mixed
15	0.015	2.583	0.919	0.121	6.479	2.835
20	0.026	16.582	12.765	1.125	21.56	16.471
30	0.106	59.697	50.741	12.361	73.247	66.726
50	2.9	285.19	252.845	54.105	325.764	314.148

Unprec. is the interval union Gauss–Seidel without preconditioner and Mixed is the strategy described in Algorithm 4

$\epsilon_{Abs} = \epsilon_{Rel} = 10^{-4}$ and $K = 2$. We perform the experiment in the same test set described previously.

Figures 9 and 10 show the results of the experiment. Table 4 compares the average elapsed time for each method.

The figures show that the mixed strategy produces bounds that are, in average, sharper than those obtained with simple methods. It can be explained by the observation that there is no dominant preconditioner strategy. The Gauss–Jordan preconditioner is better suited to cope with some problems (for example, ill conditioned problems) while the original system provides better solutions in other classes of interval linear systems (for example, diagonally dominant). On the other hand, Table 4 shows that the mixed strategy is not faster than the Gauss–Jordan preconditioner. It is due to the fact that in many problems the second iteration of the Algorithm 4 is needed.

7 Concluding remarks

In this paper, we introduce the interval union Gauss–Seidel procedure to rigorously enclose the solution set of

$$Ax = b \quad (A \in \mathcal{A}, b \in \mathcal{b}, x \in \boldsymbol{x}_0).$$

The Gauss–Seidel procedure is presented in two forms; the partial one (Algorithm 1) and the complete one (Algorithm 2). At each iteration, in the former we update only the variable corresponding to the main diagonal of the matrix $\mathcal{A}$, whereas in the latter every variable is updated.

We also studied two preconditioner heuristics for the interval union Gauss–Seidel procedure. The midpoint preconditioner takes the inverse of the midpoint of the interval hull of $\mathcal{A}$ and the Gauss–Jordan preconditioner that is based on the interval version of this method discussed by [6]. We also propose a mixed strategy that combines the original system and the Gauss–Jordan preconditioner to improve the efficiency and the quality of solutions, see the Algorithm 4.

Numerical experiments show that the interval union Gauss–Seidel procedure produces better enclosures than its interval counterparts. We performed tests on 120,000 problems generated at random as described by Table 1. Figures 3, 4 and 5 demonstrate that interval union procedures produce bounds that are up to 25% sharper than those obtained by the interval implementation of the method. Tables 2 and 3 show that there is no disadvantage in computation time when using interval union methods as compared to interval ones.

The potential increase in the number of boxes produced by Algorithms 1 and 2 is one of the main concerns in the use of interval union methods. We propose a gap filling strategy based on the ideas described by [24]. The resulting method is given by 3. We show that the maximum number of boxes produced by the complete form of the Gauss–Seidel procedure is reached only in 10% of instances. We never reach the maximum number of boxes with the partial form. The average number of boxes generated in this experiment is given by Figs. 7 and 8.

We note that the mixed strategy described in Algorithm 4 is faster and more accurate than the interval union Gauss–Seidel procedure with Gauss–Jordan preconditioner. It also produces better enclosures than those obtained with the method without preconditioner. On the other hand, if the maximum radius of $\mathcal{A}$, $\mathcal{b}$ and $\boldsymbol{x}$ are small enough then it is more efficient to turn off the preconditioning as suggested by Figs. 9 and 10.

Acknowledgements Open access funding provided by Austrian Science Fund (FWF).

References

1. Babichev, A., Kadyrova, O., Kashevarova, T., Leshchenko, A., Semenov, A.: Unicalc, a novel approach to solving systems of algebraic equations. Interval Comput. **2**, 29–47 (1993)
2. Carreras, C., López, J.A., Nieto-Taladriz, O.: Bit-width selection for data-path implementations. In: Proceedings of the 12th International Symposium on System Synthesis, 1999, pp. 114–119. IEEE (1999)
3. Dimitrova, N., Hayes, N., Markov, S.: Motion 12: Inner Addition/Subtraction Over Intervals (2014). http://grouper.ieee.org/groups/1788/email/pdfa3iJjAu21f.pdf
4. Domes, F.: JGloptLab—A Rigorous Global Optimization Software (2016). http://www.mat.univie.ac.at/~dferi/publications.html (in preparation)
5. Domes, F., Neumaier, A.: Constraint propagation on quadratic constraints. Constraints **15**, 404–429 (2010). http://www.mat.univie.ac.at/~dferi/research/Propag.pdf
6. Domes, F., Neumaier, A.: Rigorous filtering using linear relaxations. J. Glob. Optim. **53**, 441–473 (2012). http://www.mat.univie.ac.at/~dferi/research/Linear.pdf
7. Dreyer, A.: Interval Analysis of Analog Circuits with Component Tolerances. Ph.D. Thesis, Technische Universitat Kaiserslautern, Kaiserslautern, Germany (2005)
8. Fiedler, M., Nedoma, J., Ramik, J., Rohn, J., Zimmermann, K.: Linear Optimization Problems with Inexact Data. Springer, Berlin (2006)
9. Hansen, E.R.: Global Optimization Using Interval Analysis. Marcel Dekker Inc., New York (1992)
10. Hladík, M.: Optimal Preconditioning for the Interval Parametric Gauss–Seidel Method, pp. 116–125. Springer, Berlin (2016)
11. Hyvönen, E.: Constraint reasoning based on interval arithmetic: the tolerance propagation approach. Artif. Intell. **58**(1–3), 71–112 (1992)
12. Hyvönen, E., De Pascale, S.: Interval computations on the spreadsheet. In: Kearfott, R.B., Kreinovich, V. (eds.) Applications of Interval Computations, pp. 169–209. Springer, Boston, MA (1996)
13. Hyvönen, E., De Pascale, S.: InC++ library family for interval computations. In: International journal of reliable computing. Supplement to the international workshop on applications of interval computations, pp. 85–90. El Paso, Texas (1995)
14. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer Academic Publishers, Dordrecht (1996)
15. Kearfott, R.B.: A comparison of some methods for bounding connected and disconnected solution sets of interval linear systems. Computing **82**(1), 77–102 (2008)
16. Kearfott, R.B., Nakao, M.T., Neumaier, A., Rump, S.M., Shary, S.P., van Hentenryck, P.: Standardized notation in interval analysis. In: Proceedings of the XIII Baikal International School-seminar "Optimization Methods and Their Applications", vol. 4, pp. 106–113. Institute of Energy Systems, Baikal, Irkutsk (2005)
17. Kreinovich, V., Bernat, A.: Parallel algorithms for interval computations: an introduction. Interval Comput. **3**, 3–6 (1994)
18. Kreinovich, V., Lakeyev, A.V., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations, vol. 10. Springer, Berlin (1998)
19. Kuznetsov, N.: Development of financial models under partial uncertainty. Econ. Ann XXI **9–10**(2), 49–52 (2014)
20. Moore, R.E.: Interval Analysis. Prentice-Hall, Englewood Cliffs (1966)
21. Neumaier, A.: Interval Methods for Systems of Equations, Encyclopedia of Mathematics and its Applications, vol. 37. Cambridge University Press, Cambridge (1990)
22. Petunin, D., Semenov, A.: The use of multi-intervals in the unicalc solver. In: Scientific Computing and Validated Numerics. Proceedings of the International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics SCAN-95, Wuppertal, Germany, September 26–29, 1995, pp. 91–97. Akademie Verlag, Berlin (1996)
23. Ratz, D.: Inclusion isotone extended interval arithmetic. Technical Report, Institut für Angewandte Mathematik, Karlsruhe (1996). http://digbib.ubka.uni-karlsruhe.de/volltexte/67997
24. Schichl, H., Domes, F., Montanher, T., Kofler, K.: Interval Unions (2015). http://www.mat.univie.ac.at/~dferi/publications.html (in preparation)
25. Shvetsov, I., Telerman, V., Ushakov, D.: Nemo+: object-oriented constraint programming environment based on subdefinite models. In: International Conference on Principles and Practice of Constraint Programming, pp. 534–548. Springer (1997)

26. Telerman, V., Ushakov, D.: Data types in subdefinite models. In: International Conference on Artificial Intelligence and Symbolic Mathematical Computing, pp. 305–319. Springer (1996)
27. Walker, I.D., Carreras, C., McDonnell, R., Grimes, G.: Extension versus bending for continuum robots. Int. J. Adv. Robot. Syst. **3**(2), 171–178 (2006)
28. Yakovlev, A.G.: Computer arithmetics of multiintervals. Problems of cybernetics. Problem-oriented computer systems, pp. 66–81 (1987) (**In Russian**)