ELSEVIER Linear Algebra and its Applications 281 (1998) 293–309

LINEAR ALGEBRA AND ITS APPLICATIONS

Solving interval linear systems with linear programming techniques

O. Beaumont [1]

Campus Beaulieu, Institut de Recherche en Informatique et Systemes Aleatoires, 35042 Rennes, France

Received 13 June 1997; accepted 16 January 1998

Submitted by H. Schneider

Abstract

In this paper, we show how it is possible to use convex polyhedra for solving linear interval systems without using preconditioning. We first show how to derive, from an enclosure of $\square\Sigma([A],[b])$, a polyhedron which contains the convex hull of the solution set. Then, a simplex-like method enables us to find a new outer inclusion. Moreover, the constraints obtained may be used to compute an inner inclusion of $\square\Sigma([A],[b])$.

Keywords: Interval computations; Linear systems; Linear programming

1. Introduction

A lot of work [5] has been done in order to solve interval linear systems $[A]x = [b]$. Rohn and Kreinovich [9,3] have proved that the calculation of $\square\Sigma([A],[b])$, the smallest box that contains all the solutions of $[A]x = [b]$, is an NP-hard problem. On the other hand, several algorithms obtain good results, especially when the diameter of $[A]$ is small [7,5]. In this paper, we propose a new algorithm which is based on linear programming. We suppose that we know an enclosure of $\square\Sigma([A],[b])$, which implies that $[A]$ is regular. It consists of an iterative scheme which considers as input an enclosure of the solution of $\square\Sigma([A],[b])$ and returns an (usually better) enclosure.

[1] E-mail: obeaumon@irisa.fr.

0024-3795/98/$19.00
PII: S0024-3795(98)10044-7

This algorithm converges toward a superset of the convex hull of the united solution set $\Sigma([A],[b])$. In this paper, we use the following notations:

$$\Sigma([A],[b]) = \{x, \exists A \in [A],\ \exists b \in [b],\ Ax = b\},$$

$\Gamma([A],[b])$ denotes the convex hull of $\Sigma([A],[b])$,

$\square\Sigma([A],[b])$ denotes the interval hull of $\Sigma([A],[b])$.

The interval linear systems we consider in this paper are defined by the following notations:

$$[A] = [A_c - \Delta A, A_c + \Delta A], \qquad [b] = [\underline{b}, \bar{b}] = [b_c - \Delta b, b_c + \Delta b],$$

where ΔA and Δb are respectively nonnegative matrix and vector.

2. How to find a polyhedron that contains the convex hull of the united solution set

It is known [5] that the solution of the problem $[A]x = [b]$ is in general not convex. As far as we are only interested in $\square\Sigma([A],[b])$, we may use $\Gamma([A],[b])$ as intermediate set. Oettli and Prager [6] proved the following theorem.

Theorem 1.

$$(\exists A \in [A],\ \exists b \in [b],\ Ax = b) \iff |A_c x - b| \leqslant \Delta A|x| + \Delta b.$$

This expression does not lead directly to a polyhedron, because of the absolute value, which underlines the fact that the solution set is usually not a convex set.

We give a first result which provides a way to get rid of the absolute values.

Lemma 1. *If* $[\underline{x}, \bar{x}] \subset \mathbb{R}^n$, $\underline{x} < \bar{x}$ *and, if*

$$\alpha_j = \frac{|\bar{x}_j| - |\underline{x}_j|}{\bar{x}_j - \underline{x}_j} \quad \text{and} \quad \beta_j = \frac{\bar{x}_j|\underline{x}_j| - \underline{x}_j|\bar{x}_j|}{\bar{x}_j - \underline{x}_j},$$

where x_j *denotes the jth component of* x, *we have:*

$$\forall x \in [\underline{x}, \bar{x}],\ \forall j,\ 1 \leqslant j \leqslant n, \quad |x_j| \leqslant \alpha_j x_j + \beta_j.$$

Proof. In fact, several situations may occur (see Figs. 1–3):

- *First case:* $\bar{x}_j \leqslant 0$, then $\alpha_j = -1, \beta_j = 0$, and $\alpha_j x + \beta_j = -x = |x|$.
- *Second case:* $\underline{x}_j \geqslant 0$, then $\alpha_j = 1, \beta_j = 0$, and $\alpha_j x + \beta_j = x = |x|$.
- *Third case:* $\bar{x}_j \underline{x}_j \leqslant 0$, then $\alpha_j = (\bar{x}_j + \underline{x}_j)/(\bar{x}_j - \underline{x}_j), \beta_j = (-2\bar{x}_j \underline{x}_j)/(\bar{x}_j - \underline{x}_j)$.

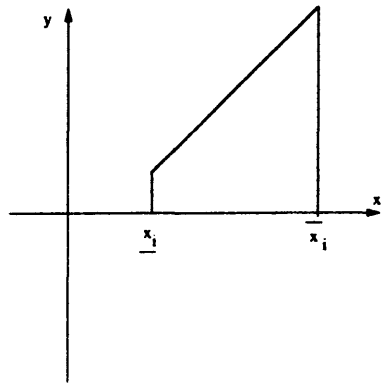


Fig. 1. Situation when both $\underline{x}_i$ and $\bar{x}_i$ are positive.

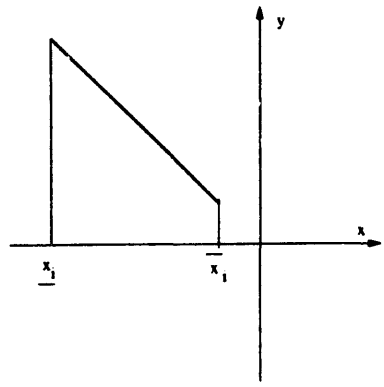


Fig. 2. Situation when both $\underline{x}_i$ and $\bar{x}_i$ are negative.

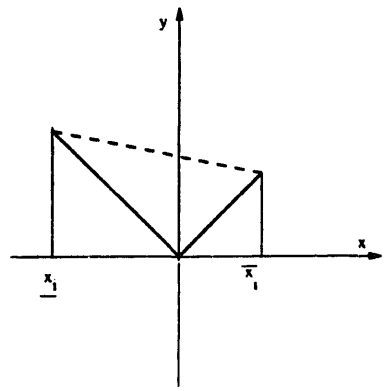


Fig. 3. Situation when $\underline{x}_i \leqslant 0$ and $\bar{x}_i \geqslant 0$.

- If $0 \leqslant z \leqslant \bar{x}_j$, then $\alpha_j z + \beta_j - |z| = -2\underline{x}_j/(\bar{x}_j - \underline{x}_j) * (\bar{x}_j - z) \geqslant 0$. Thus, $|z| \leqslant \alpha_j z + \beta_j$.
- If $\underline{x}_j \leqslant z \leqslant 0$, then $\alpha_j z + \beta_j - |z| = 2\bar{x}_j/(\bar{x}_j - \underline{x}_j) * (z - \underline{x}_j) \geqslant 0$. Thus, $|z| \leqslant \alpha_j z + \beta_j$. □

Lemma 2. *The convex hull of the set*

$$\mathscr{S} = \{(x,y) \in \mathbb{R}^2, \quad x \in [\underline{x}_j, \bar{x}_j], \ y \leqslant |x|\}$$

is the polyhedron defined by

$$\mathscr{P} = \{(x,y), x \in [\underline{x}_j, \bar{x}_j], 0 \leqslant y \leqslant \alpha_j x + \beta_j\}.$$

Proof. If $\underline{x}_j \geqslant 0$ or $\bar{x}_j \leqslant 0$, then $\mathscr{S} = \mathscr{P}$ and the property is trivial. We assume now that $\underline{x}_j < 0 < \bar{x}_j$. The set $\mathscr{P}$, which is defined by linear inequalities, is convex, and Lemma 2 implies $\mathscr{S} \subset \mathscr{P}$. Therefore, $\mathrm{Co}(\mathscr{S}) \subset \mathscr{P}$ (where $\mathrm{Co}(\mathscr{S})$ denotes the convex hull of $\mathscr{S}$).

Reciprocally, let us consider

$$(a,b) \in \{(x,y), x \in [\underline{x}_j, \bar{x}_j], y \leqslant \alpha_j x + \beta_j\} \setminus \mathscr{S}.$$

Let us consider now the line $y = \alpha_j x + (b - \alpha_j a)$, which intersects $y = x$ and $y = -x$ at the points $(a_1, b_1) = (b - \alpha_j a)/(1 - \alpha_j)(1, 1)$ and $(a_2, b_2) = (b - \alpha_j a)/(1 + \alpha_j)(-1, 1)$ respectively. Since $b - \alpha_j a \leqslant \beta$, we obtain $\underline{x}_j \leqslant a_1 \leqslant a_2 \leqslant \bar{x}_j$ and, therefore, $(a_1, b_1) \in \mathscr{S}$, $(a_2, b_2) \in \mathscr{S}$ and $(a,b) \in \mathrm{Co}(\mathscr{S})$. We have proved that $\mathrm{Co}(\mathscr{S}) = \mathscr{P}$. □

Let us assume that we know an enclosure of $\square\Sigma([A],[b])$, which may be obtained, for instance, by the algorithm proposed by Rump [7], $\square\Sigma([A],[b]) \subset [\underline{x}, \bar{x}]$. Therefore, we can expect that the following simplex $\Omega([A],[b],[\underline{x},\bar{x}])$ represents a good approximation of $\Gamma([A],[b])$. If we denote by D_x the diagonal matrix whose diagonal entries are the α_i's and β the vector of the β_i's obtained from $[\underline{x},\bar{x}]$, then we define $\Omega([A],[b],[\underline{x},\bar{x}])$ as follows:

$$\Omega([A],[b],[\underline{x},\bar{x}]) \begin{cases} A_c x - \Delta A \, D_x x \leqslant \bar{b} + \Delta A \beta, \\ A_c x + \Delta A \, D_x x \geqslant \underline{b} - \Delta A \beta. \end{cases}$$

$\Omega([A],[b],[\underline{x},\bar{x}])$ depends on the initial enclosure $[\underline{x},\bar{x}]$ because both D_x and β are defined under the condition $\square\Sigma([A],[b]) \subset [\underline{x},\bar{x}]$. We can now enunciate the main result of this paper.

Theorem 2. *Let* $\square\Sigma([A],[b]) \subset [\underline{x},\bar{x}]$. *Then, we have*

$$\Sigma([A],[b]) \subset \left\{ x ; \begin{pmatrix} A_c - \Delta A \, D_x \\ -A_c - \Delta A \, D_x \end{pmatrix} x \leqslant \begin{pmatrix} \bar{b} + \Delta A \beta \\ -\underline{b} + \Delta A \beta \end{pmatrix} \right\},$$

where

$$\alpha_j = \frac{|\bar{x}_j| - |\underline{x}_j|}{\bar{x}_j - \underline{x}_j} \quad \textit{and} \quad \beta_j = \frac{\bar{x}_j|\underline{x}_j| - \underline{x}_j|\bar{x}_j|}{\bar{x}_j - \underline{x}_j}$$

Proof. This theorem is a direct consequence of Theorem 1 and Lemma 1. □

We have seen above how an enclosure of $\square\Sigma([A],[b])$ leads to a simplex that describes a superset of $\Gamma([A],[b])$. The set of problems $\max_{x\in\Omega([A],[b],[\underline{x},\bar{x}])} x_i$ and $\min_{x\in\Omega([A],[b],[\underline{x},\bar{x}])} x_i$ can be solved, for instance, by applying $2n$ times the Simplex method [1]. Therefore, a new enclosure of $\square\Sigma([A],[b])$ is obtained, and an iterative scheme can be developed.

The limit of this iterative algorithm is usually a good enclosure of $\square\Sigma([A],[b])$ (not too large). Unfortunately, it is difficult to find a characterization of this limit in order to study its accuracy. The limit is usually not equal to $\square\Sigma([A],[b])$. We will display the results concerning the accuracy of the enclosure obtained in Section 4.

We may nevertheless conclude in several situations. For instance, when we know an enclosure of $\square\Sigma([A],[b])$ in which each x_i keeps a constant sign, it is easy to prove that the algorithm described above provides the exact solution $\square\Sigma([A],[b])$ after only one step. Indeed, in this case, the sets $\{(x,y),\ x\in[\underline{x}_j,\bar{x}_j],\ y\leqslant\alpha_j x+\beta_j\}$ and $\{(x,y),\ x\in[\underline{x}_j,\bar{x}_j],\ y\leqslant|x|\}$ are equal, and $\Omega([A],[b],[\underline{x},\bar{x}])$ is an exact representation of the solution set of $[A]x=[b]$, which is in this case convex.

We show on small examples given in [5] (matrices of size 2) the different situations that may occur:

If

$$[A] = \begin{pmatrix} [2,4] & [-1,1] \\ [-1,1] & [2,4] \end{pmatrix}, \qquad [b] = \begin{pmatrix} [-3,3] \\ 0 \end{pmatrix},$$

then

$$\square\Sigma([A],[b]) = \begin{pmatrix} [-2,2] \\ [-1,1] \end{pmatrix},$$

the enclosure obtained by the algorithm proposed by Rump [7] is

$$\square\Sigma([A],[b]) \subset \begin{pmatrix} [-2.11,2.11] \\ [-1.11,1.11] \end{pmatrix},$$

and, when starting our iterative algorithm from the enclosure obtained by the algorithm proposed by Rump, the limit we obtain is

$$\square\Sigma([A],[b]) \subset \begin{pmatrix} [-2,2] \\ [-1,1] \end{pmatrix}.$$

In this case, the algorithm we propose converges to $\square\Sigma([A],[b])$.

If

$$[A] = \begin{pmatrix} [2,4] & [-1,1] \\ [-1,1] & [2,4] \end{pmatrix}, \qquad [b] = \begin{pmatrix} [-0.5,6] \\ [1,1.5] \end{pmatrix},$$

then

$$\square\Sigma([A],[b]) = \begin{pmatrix} [-0.83,4.16] \\ [-1.16,2.83] \end{pmatrix},$$

the enclosure obtained by the algorithm proposed by Rump [7] is

$$\square\Sigma([A],[b]) \subset \begin{pmatrix} [-2.56,4.23] \\ [-2.06,2.89] \end{pmatrix},$$

and, when starting our iterative algorithm from the enclosure obtained by the algorithm proposed by Rump, the limit we obtain is

$$\square\Sigma([A],[b]) \subset \begin{pmatrix} [-1.50,4.16] \\ [-1.45,2.83] \end{pmatrix}.$$

In this case, the right bounds of the solution set we obtain are exact, but not the left ones. The algorithm converges to a strict superset of $\square\Sigma([A],[b])$.

If

$$[A] = \begin{pmatrix} 2 & [-1,0] \\ [-1,0] & 2 \end{pmatrix}, \qquad [b] = \begin{pmatrix} 1.2 \\ -1.2 \end{pmatrix},$$

then

$$\square\Sigma([A],[b]) = \begin{pmatrix} [0.3,0.6] \\ [-0.6,0.3] \end{pmatrix},$$

the enclosure obtained by the algorithm proposed by Rump [7] is

$$\square\Sigma([A],[b]) \subset \begin{pmatrix} [0.23,0.72] \\ [-0.72,-0.24] \end{pmatrix},$$

and, when starting our iterative algorithm from the enclosure obtained by the algorithm proposed by Rump, the result we obtain after one step is

$$\square\Sigma([A],[b]) \subset \begin{pmatrix} [0.3,0.6] \\ [-0.6,0.3] \end{pmatrix}.$$

In this case, the solution set we obtain is equal to $\square\Sigma([A],[b])$ after only one step of the algorithm.

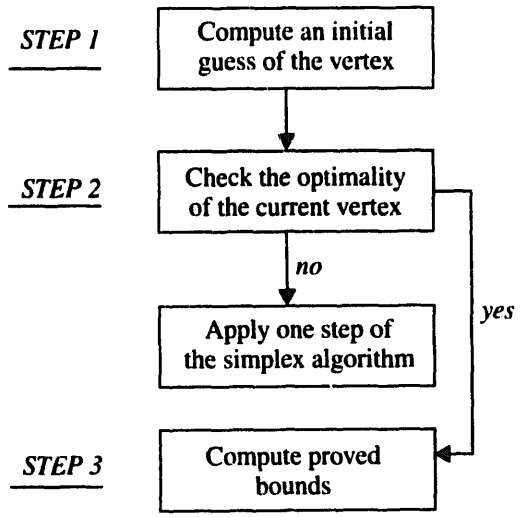
In Section 3, we present a modification of the algorithm presented above, in order to perform it in a reasonable amount of time.

3. Algorithm for the outer inclusion

In Section 2, we have presented an iterative method which solves interval linear systems. This method requires the execution of $2n$ Simplex algorithms during each step of the algorithm. Since an execution of the Simplex algorithm requires roughly $O(n^3)$ flops [1,11], the total amount of work per iteration is therefore of order $O(n^4)$. The algorithms proposed by Rump [7] and Neumaier [5] require an amount of work of order $5n^3$. We show in this section how to perform a step of the algorithm in time $O(n^3)$.

We propose a three-step algorithm in order to obtain an outer inclusion. The first step consists in using perturbation theory in order to find good starting points for the Simplex algorithm. The second step consists in checking the optimality of the starting points defined during step 1, and, if necessary in starting the Simplex algorithm. Indeed, as we will show in Section 4, the starting points obtained during step 1 are in many cases optimal especially when the perturbation matrix ΔA is small. Therefore, in many cases, the use of the Simplex algorithm is not necessary and the computation of the solution can be obtained in time of order $O(n^3)$. The last step is a correction step, which is necessary to obtain a proved outer inclusion, since all previous computations are not exact.

3.1. General sketch of the algorithm



3.2. First step

In order to obtain a cheap resolution of the problem, it is of importance to know good starting points for the Simplex algorithm. The first step of the algorithm consists therefore in using perturbation theory in order to find these

starting points. Let us suppose that x_0 is the solution of the system $A_c x_0 = b_c$, then $(x_0 + \delta x) \in \Sigma([A], [b])$ if and only if

$$\exists\, \delta A \in [-\Delta A, \Delta A],\ \exists\, \delta b \in [-\Delta b, \Delta b], \quad (A_c + \delta A)(x_0 + \delta x) = b_c + \delta b.$$

Therefore, at first order,

$$\delta x = A_c^{-1}(-\delta A x_0 + \delta b),$$

$$\forall i, 1 \leqslant i \leqslant n, \quad e_i^t\, \delta x = -e_i^t\, A_c^{-1}(\delta A x_0 + \delta b).$$

At this stage, we need to use the following basic lemma of interval arithmetic.

Lemma 3. *Let* $(x, y) \in \mathbb{R}^{n^2}$, $\Delta A \in M_n(\mathbb{R})$, $\Delta A \geqslant 0$, *then*

$$\{x^t \delta A y, \quad \delta A \in [-\Delta A, \Delta A]\} = [-|x^t|\Delta A|y|,\ |x^t|\Delta A|y|],$$

$$x^t \delta A y = -|x^t|\Delta A|y| \quad \text{if } \delta A = -D_1 \Delta A D_2,$$
$$x^t \delta A y = |x^t|\Delta A|y| \quad \text{if } \delta A = D_1 \Delta A D_2,$$

where D_1 *and* D_2 *denote the diagonal matrices whose entries are the sign vectors of* x *and* y *respectively.*

Therefore, when δA and δb describe respectively $[-\Delta A, \Delta A]$ and $[-\Delta b, \Delta b]$, $e_i^t \delta x$ describes, at first order,

$$[-|e_i^t A_c^{-1}|(\Delta A|x_0| + \Delta b),\ |e_i^t A_c^{-1}|(\Delta A|x_0| + \Delta b)].$$

Therefore, in order to approximately minimize $e_i^t \delta x$, a good choice consists in considering the system $A_c x - b_c = D_1(\Delta b + \Delta A D_2 x)$, where D_1 and D_2 are defined above.

Moreover, since $\Delta A D_2 x$ is supposed to be close to $\Delta A D_x x$, we can associate the solution of the previous equation with the vertex of $\Omega([A], [b], [\underline{x}, \bar{x}])$ defined by

$$A_c x - b_c = D_1(\Delta b + \Delta A D_x x).$$

Therefore, we can start the Simplex algorithm for minimizing $e_i^t \delta x$ from the vertex defined above. We will analyze the quality of this starting point in Section 4.

The total cost of this step consists in the inversion of A, i.e. n^3 flops when using an LU factorization [2].

3.3. Second step

This step consists in checking the optimality of the constraints set corresponding to a given point. In order to apply general theorems of linear pro-

gramming, we perform the following change of variables which induce use of positive variables. We set $y = x - \underline{x}$:

$$\Omega([A],[b],[\underline{x},\bar{x}]) = \left\{ y, \begin{pmatrix} A_c - \Delta A D_\alpha \\ -A_c - \Delta A D_\alpha \\ -Id_n \end{pmatrix} \right.$$
$$\left. y \leqslant \begin{pmatrix} \bar{b} + \Delta A\beta - (A_c - \Delta A D_\alpha)\underline{x} \\ -\underline{b} + \Delta A\beta + (A_c + \Delta A D_\alpha)\underline{x} \\ 0 \end{pmatrix} \right\}.$$

Therefore, a vertex y of $\Omega([A],[b],[\underline{x},\bar{x}])$ satisfies an equation of the following form

$$(DA_c - \Delta A D_\alpha)y = Db' + \Delta b', \tag{1}$$

where D is a diagonal matrix satisfying $|D| = Id_n$, $b'_c = b_c - A_c\underline{x}$ and $\Delta b' = \Delta b + \Delta A(D_\alpha \underline{x} + \beta)$.

3.3.1. Checking the optimality of the constraint set

Our aim is to check the optimality of the set of the constraints defined by Eq. (1) with respect to the minimization of y_1 over Ω. We use the following fundamental theorem of linear programming [1]:

Theorem 3. *Let u^t be the solution of*

$$u^t(DA_c - \Delta A D_\alpha) = e_1^t, \tag{2}$$

where e_1^t is the first canonical basis vector. If $u \geqslant 0$, then:

- *The set of constraints defined above is optimal for the problem* $\min_{y\in\Omega} y_1$.
- *The vertex y corresponding to the minimization satisfies $(DA_c - \Delta A D_\alpha)y = Db' + \Delta b'$.*
- $\min_{y\in\Omega} y_1 = e_1^t y = u^t(DA_c - \Delta A D_\alpha)y = u^t(Db' + \Delta A b')$.

3.3.2. Algorithm

We now present an algorithm to solve approximately Eq. (2). Let M be defined as $M = \Delta A D_\alpha B$, where B is the computed inverse of A_c. Note that M does not depend on the extremal point problem we consider. We use the following iterative scheme:

$$\delta_1^t = e_1^t B, \qquad \delta_{i+1}^t = (\delta_i^t D)M.$$

We stop the iteration scheme when $\delta_k \leqslant \epsilon_1$, a small threshold, and we set $\tilde{u}^t = (\sum_1^k \delta_i^t)D$.

We consider that the set of constraints is optimal if $\tilde{u} \leqslant \epsilon_2$, where ϵ_2 is a small positive vector. Therefore, if $\tilde{u} \leqslant \epsilon_2$, we go to the third step, otherwise, we perform a step of the Simplex algorithm and then we restart the second step. Note that if $\tilde{u}^t$

is the vector associated with the maximization of y_1, then $\tilde{u}'' = (\sum_1^k (-1)^{i+1}\delta_i^t)D$. Therefore, the computation of $\tilde{u}'$ only involves k additions of vectors.

3.4. Third step

3.4.1. Computing sure bounds

We are looking for a proved outer inclusion of the interval hull of the solution set. We therefore need to perform a correction step, since all previous computations were not sure. At this stage, we know that $\tilde{u} \leqslant \epsilon_2$.

Let us set

$$\hat{u} = \min(\tilde{u}, 0), \quad S = \begin{pmatrix} A_c - \Delta A D_\alpha \\ -A_c - \Delta A D_\alpha \end{pmatrix} \quad \text{and} \quad s = \begin{pmatrix} b_c' + \Delta b' \\ -b_c' + \Delta b' \end{pmatrix}.$$

If $\epsilon^t = \hat{u}^t S - e_1^t$, then we have

$$\begin{aligned} \hat{u}^t s &\leqslant \max\{z^t s,\ z \leqslant 0,\ z^t S \leqslant e_1^t + \epsilon^t\} \\ &\leqslant \min\{(e_1 + \epsilon)^t y, y \geqslant 0, Sy \leqslant s\} \text{ duality theorem of L.P.} \\ &\leqslant \min\{e_1^t y, y \geqslant 0, Sy \leqslant s\} + \max\{\epsilon^t y, y \geqslant 0, Sy \leqslant s\} \end{aligned}$$

and, therefore

$$\min_{y \in \Omega} y_1 \geqslant \hat{u}^t s - \max\{\epsilon^t y,\ 0 \leqslant y \leqslant \bar{x} - \underline{x}\} \geqslant \hat{u}^t s - ||\epsilon||\ ||\bar{x} - \underline{x}||.$$

The expression above gives a proved lower bound for $\min_{y \in \Omega} y_1$ and therefore for $\min_{y \in \Sigma} y_1$.

3.4.2. Practical implementation

In the sequel, we show how to bypass the computation of ϵ, by obtaining an upper bound of $||\epsilon||$. Let A' denote the exact inverse of the computed inverse B of A (whenever A' does not exist, we cannot apply what follows). Let us set

$$\Delta = A' - A_c, \qquad C = DA_c - \Delta A D_\alpha,$$

$$\mu = (\tilde{u} - \hat{u})^t \qquad \tilde{u}^t D = e_1^t B \sum_0^{k-1} (DM)^i,$$

$$u^t D = e_1^t B \sum_0^{\infty} (DM)^i \iff u^t (C - D\Delta) = e_1^t,$$

$$v^t D = e_1^t A_c \sum_0^{\infty} (DM)^i \iff u^t C = e_1^t.$$

Theorem 4. *If* $||M|| \leqslant \frac{1}{2}$, *and* $\epsilon' ||A_c||\ ||B|| \leqslant \frac{1}{2}$, *where* ϵ', *defined below, only depends on the machine precision, then*

$$||\epsilon|| \leqslant (||A_c|| + ||\Delta A||)(||\epsilon_1|| + ||\mu||) + 4\epsilon'||A_c||^2||B||(||\tilde{u}|| + ||\epsilon_1||).$$

Proof. Evaluation of $||(\tilde{u}-u)^tC||$: $(\tilde{u}-u)^tD = e_1^tB\sum_k^\infty (DM)^i = \delta_k DM\sum_0^\infty (DM)^i$. Since $||M|| \leqslant \frac{1}{2}$,

$$||(\tilde{u}-u)^tC|| \leqslant ||\epsilon_1||(||A_c|| + ||\Delta A||) \quad \text{and} \quad ||(\tilde{u}-u)^t|| \leqslant ||\epsilon_1||.$$

Evaluation of $||(v-u)^tC||$: $(v-u)^tC = u^tD\Delta$ and therefore $||(v-u)^tC|| \leqslant ||\Delta||(||\tilde{u}|| + ||\epsilon_1||)$.

Evaluation of $||(v-\tilde{u})^tC||$: $||(v-\tilde{u})^tC|| \leqslant ||(v-u)^tC|| + ||(\tilde{u}-u)^tC||$ and therefore

$$||(v-\tilde{u})^tC|| \leqslant ||\epsilon_1||(||A_c|| + ||\Delta A||) + ||\Delta||(||\tilde{u}|| + ||\epsilon_1||).$$

Evaluation of $||\Delta||$: If the computed inverse B of A_c is obtained with LU factorization, we know [4] that $A_cB = I + E$ where $||E|| \leqslant \epsilon'||A_c||||B||$, where ϵ' depends on machine accuracy. Thus, $A' = (I+E)^{-1}A_c$ and, since $||E|| \leqslant \epsilon'||A_c||||B|| \leqslant \frac{1}{2}$, $A' - A_c = A_cE\sum_0^\infty (-1)^{k+1}E^k$ and finally

$$||\Delta|| \leqslant 2\epsilon'||A_c||^2\,||B||.$$

Since $\epsilon = \tilde{u}^tC - e_1^t = (\tilde{u}-v)^tC + (v^tC - e_1^t) - \mu C$, we obtain the proof of the theorem. □

The crucial point is that the majoration of $||\epsilon||$ does not require additional computations.

4. Numerical results

In this section, we describe numerical results obtained with random matrices A_c and ΔA and for random vectors b_c and Δb. We compare the results of the proposed algorithm after one iteration with the results of the algorithm proposed by Rump [7]. The initial enclosure we consider is the result of Rump's algorithm. It is known [8] that, for Rump's algorithm, the ratio perimeter of Rump's outer inclusion/perimeter of the interval hull depends on $\text{norm}(A)/\text{norm}(\Delta A)\,\text{cond}(A)$.

We therefore display results according to $\text{norm}(A)/\text{norm}(\Delta A)\,\text{cond}(A)$. As the quality of the enclosure of Σ in Ω depends on the position of the solution set with respect to 0, we consider different situations for Σ (that is to say containing 0 or not intersecting any axis). The x-axis represents the size of all the test matrices (Fig. 4).

In Figs. 5–7 we display τ_1, the average number of the steps of the Simplex algorithm necessary for the computation of an extremal point from the initial guess (step 2) and

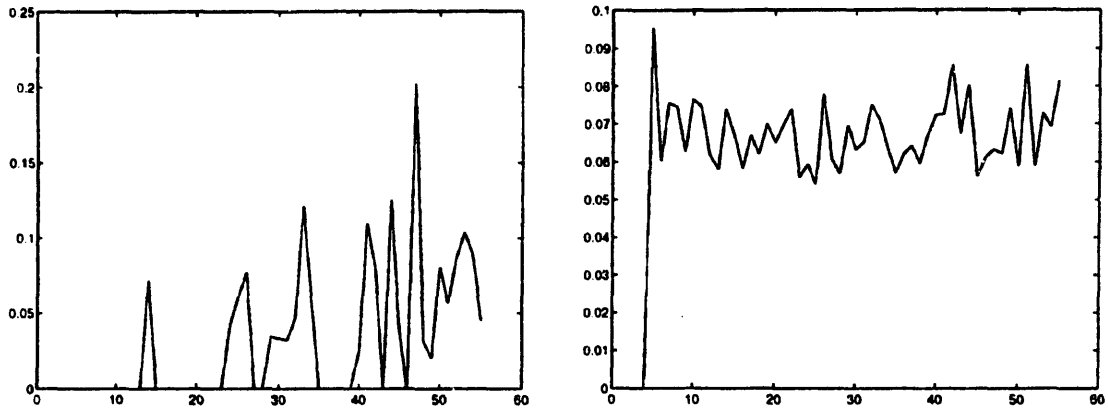


Fig. 4. τ_1 and τ_2 Σ not centered – norm(ΔA) = (norm$(A_c) * 0.1$)/cond(A_c).

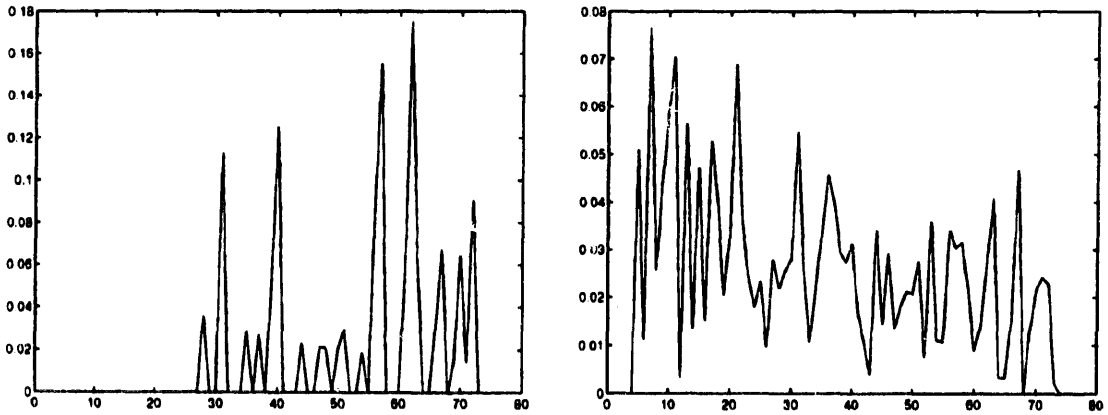


Fig. 5. τ_1 and τ_2, Σ centered – norm(ΔA) = (norm$(A_c) * 0.1$)/cond(A_c).

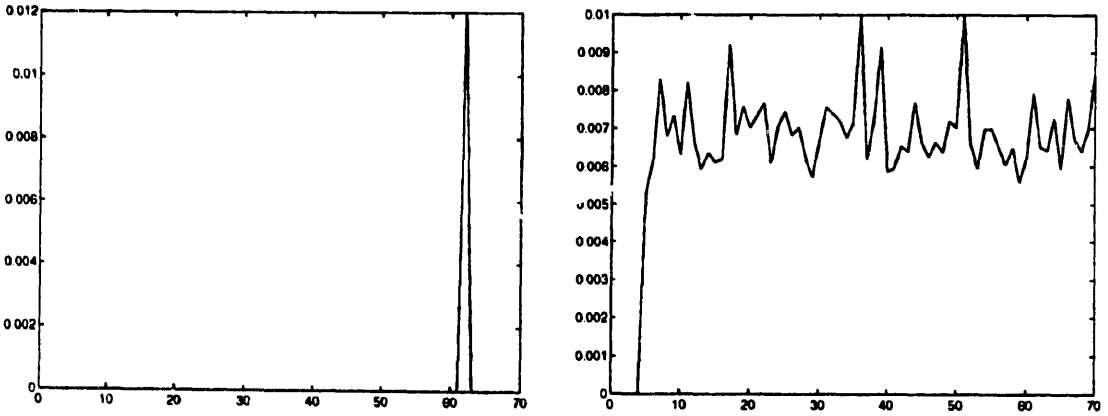


Fig. 6. τ_1 and τ_2, Σ not centered – norm(ΔA) = (norm$(A_c) * 0.1$)/cond(A_c).

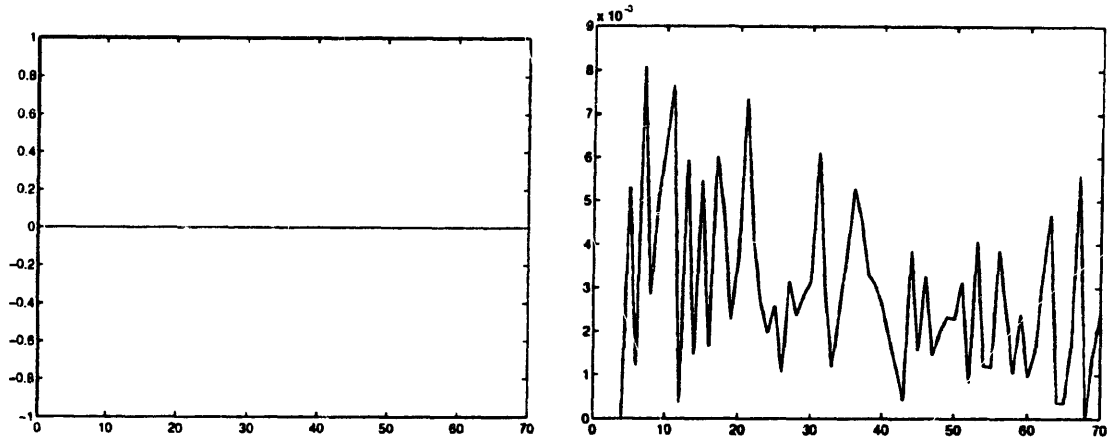


Fig. 7. τ_1 and τ_2, Σ centered – norm(ΔA) = (norm(A_c) * 0.01)/cond(A_c).

$$\tau_2 = 1 - \frac{\text{perimeter(Simplex method)}}{\text{perimeter(Rump's algorithm)}}.$$

5. Algorithm for the inner inclusion

We now present an algorithm which computes an inner inclusion of the solution set. By inner inclusion, we mean an interval vector $[\underline{y}, \bar{y}]$ such that

$$[\underline{y}, \bar{y}] \subset \square\Sigma([A], [b]) \subset \Omega([A], [b], [\underline{x}, \bar{x}]).$$

Note that it is a completely different problem to solve $[\underline{y}, \bar{y}] \subset \Sigma([A], [b])$. The main interest of the inner inclusion we compute is to estimate the accuracy of the outer inclusion and to enclose the interval hull of the solution set between two interval vectors. Moreover, numerical results indicate that the inner inclusion is very close to the interval hull (in fact, it is very often the exact interval hull).

The algorithm is based on the results for the outer inclusion. It computes $2n$ points of the solution set which are supposed to be close to extremal points.

5.1. How to find inner "extremal" points?

Let us consider again the problem of the minimization of y_1. We know that the extremal point of Ω which realizes this minimization is defined by

$$D(A_c x - b_c) = \Delta A(D_x x + \beta) + \Delta b. \tag{3}$$

Let us now consider a point on the frontier of Σ.

Lemma 4 (Rohn). *x belongs to the frontier of Σ $\iff$ $|A_c x - b_c| = \Delta A|x| + \Delta b$ and*

$$|A_c x - b_c| = \Delta A|x| + \Delta b \iff$$

$$\exists D' \text{ diagonal}, \ |D'| = Id(n), \ D'(A_c x - b_c) = \Delta A|x| + \Delta b. \tag{4}$$

This lemma is a direct application of the Oettli–Prager [8] theorem. Since $D_\alpha x + \beta$ represents an approximation of $|x|$, we can notice an analogy between the definition of x in expressions (3) and (4).

We therefore consider as extremal point associated to the minimization of x_1 for the inner inclusion the point x defined by

$$D(A_c x - b_c) = \Delta A|x| + \Delta b,$$

where D is the matrix associated to the minimization of y_1 over Ω.

5.2. Algorithm

The algorithm consists in solving the equations

$$D(A_c x - b_c) = \Delta A|x| + \Delta b \iff x = M|x| + a,$$

where $M = A_c^{-1} D \Delta A$ and $a = A_c^{-1}(D\Delta b + b)$.

Since such a point belongs to the frontier of Σ, the algorithm leads to an inner inclusion of Σ. If we suppose that $[A]$ is strongly regular, that is to say $\rho(|A_c^{-1}|\Delta A) < 1$, then $\rho(M) < 1$ and we can use the algorithm proposed by Rohn.

Theorem 5. *If $\rho(M) < 1$, then, for every a, the equation $x = M|x| + a$ has a unique solution, and the iteration*

$$x^{l+1} := M|x^l| + a (l = 0, 1, 2, \ldots)$$

converges to the solution for every choice of the starting vector x^0.

In order to obtain a good starting vector, we can solve the equation

$$D(A_c x - b_c) = \Delta A(D_\alpha x + \beta) + \Delta b.$$

We therefore start from the vector which corresponds to the minimization of x_1 over Ω. When $\rho(M)$ is not small enough, it is known that the convergence may be slow. Rohn therefore proposed the sign accord algorithm, which does not require strong regularity of $[A]$.

It consists in solving $x = MD'x + a$, for different matrices D'.

Sign accord algorithm (Rohn)

Step 1: Select D' with $|D'| = Id(n)$

Step 2: For $s = 1, \ldots, 2^n$ do
solve $x = MD'x + a$;
if $D'x \geqslant 0$ terminate (success); otherwise compute
$k := \min\{j = 1, \ldots, n; D'_{jj} x_j < 0\}$;
change the sign of D'_{kk};

Step 3: Terminate (failure).

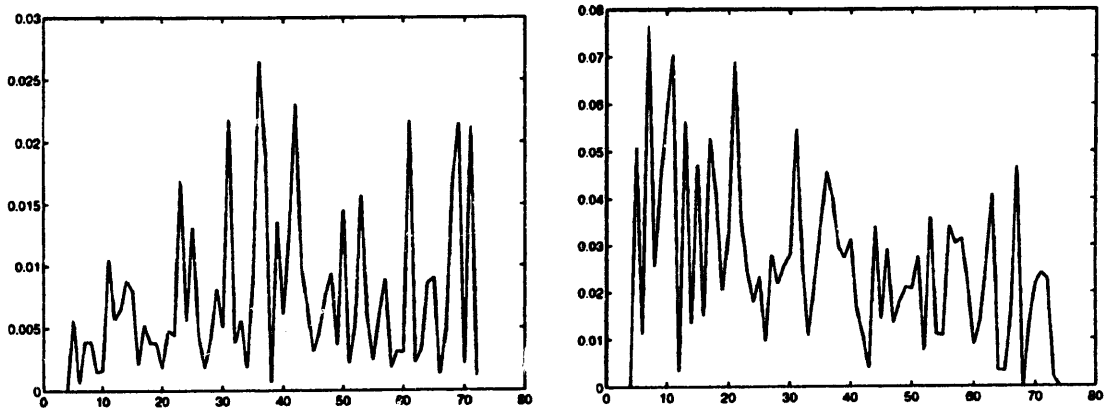


Fig. 8. τ_1 and τ_2, Σ centered – norm(ΔA) = (norm(A_c) * 0.1)/cond(A_c).

Rohn [10] has proved that the algorithm is finite when $[A]$ is regular. Although the number of steps may be exponential, it is generally reasonable. If we know the signs of the solution of $D(A_c x - b_c) = \Delta A(D_x x + \beta) + \Delta b$, then we can start with the matrix D' such that $D'x \geqslant 0$. In the cases such that the points that realize the minimization of x_1 over Σ and Ω have the same sign vector, we will not have to perform any change of sign. In fact, none of the cases we considered for the outer inclusion requires the execution of more than one step.

5.3. Numerical results

We display results for matrices of the same kind as those used for the outer inclusion. We only consider the case Σ centered since results for inner and outer inclusion are the same when Σ does not intersect any axis. Figs. 8 and 9 represent the evolution with n of both quantity τ_1 and τ_2, where

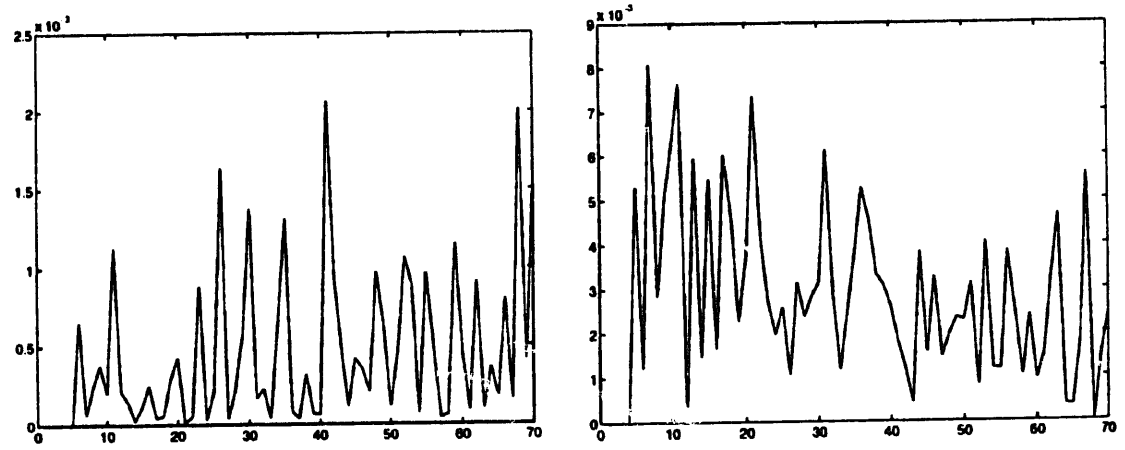


Fig. 9. τ_2 and τ_2, Σ centered – norm(ΔA) = (norm(A_c) * 0.01)/cond(A_c).

$$\tau_1 = 1 - \frac{\text{perimeter(inner inclusion)}}{\text{perimeter(outer inclusion)}}$$

$$\tau_2 = 1 - \frac{\text{perimeter(outer inclusion)}}{\text{perimeter(Rump's algorithm)}}.$$

We can see that inner and outer inclusions are usually very close. Even in the case where $\text{norm}(\Delta A) = (\text{norm}(A_c) * 0.1)/\text{cond}(A_c)$, we usually obtain

$$1 - \frac{\text{perimeter}(\square\Sigma)}{\text{perimeter(outer inclusion)}} \leqslant 1\%.$$

6. Conclusion

In this paper, we show how to derive from the Simplex algorithm an efficient method to compute inner and outer inclusion of the united solution set. The outer inclusion is obtained in $O(n^3)$ flops and does not require the preconditioning of the system. The inner inclusion coincides very often with the exact interval hull of the united solution set, but we have no way to prove this property. However, an estimation of the quality of the outer inclusion can be obtained, even without computing the inner inclusion.

Acknowledgements

The author wishes to thank Bernard Philippe and Jiri Rohn for careful reading and for for their proposals, which greatly improved the clarity of this paper.

References

[1] A. Chvatal, Linear Programming, Freeman, New York, 1983.
[2] P. Gill, W. Murray, M. Wright, Numerical Linear Algebra and Optimization, Addison-Wesley, Reading, MA, 1991.
[3] V. Kreinovich, J. Rohn, Computing exact componentwise bounds on solutions of linear systems with interval data is NP-Hard, SIAM J. Matrix Anal. Appl. 16 (2) (1995) 415–420.
[4] P. Lascaux, R Theodor, Analyse Numérique Matricielle Appliquée à l'Art de l'Ingénieur, Masson.
[5] A. Neumaier, Interval methods for systems of equations, Cambridge University Press, Cambridge, 1990.
[6] W. Oettli, W. Prager, Computability of approximate solution of linear equations with given error bounds for coefficients and right-hand sides, Numer. Math. 6 (1964) 405–409.
[7] S.M. Rump, On the solution of interval linear systems, Computing 47 (1992) 337–353.

[8] S.M. Rump, Verification methods for dense and sparse systems of equations, in: J. Herzberger (Ed.), Topics in Validated computations, Elsevier, Amsterdam, 1994.
[9] J. Rohn, NP-hardness results for linear algebraic problems with interval data, in: J. Herzberger (Ed.), Topics in Validated computations, Elsevier, Amsterdam, 1994.
[10] J. Rohn, Checking bounds on solutions of linear interval equation is NP-hard, Linear Algebra Appl. 223–224 (1995) 589–596.
[11] A. Schrijver, Theory of Linear and Integer Programming, Wiley, New York, 1986.