Table 5.4: Square interval linear systems – average computation times. The uniform radii was set to $r = 0.001$, the computation times are in seconds, n is the number of variables of a system.

n	ge+pre	jacobi+pre	hbr	krawczyk+pre
10	0.44	0.10	0.03	0.11
20	1.65	0.11	0.04	0.12
30	3.64	0.14	0.05	0.15
40	6.41	0.16	0.06	0.17
50	10.00	0.18	0.08	0.21
60	14.42	0.21	0.11	0.25
70	19.62	0.25	0.14	0.32
80	25.67	0.30	0.18	0.39
90	32.59	0.36	0.23	0.49
100	40.39	0.42	0.29	0.61

Table 5.5: Square interval linear systems – percentage of finite enclosures returned. The uniform radii was set to $r = 0.001$, n is the number of variables of a system.

n	ge+pre	jacobi+pre	hbr	krawczyk+qpre
10	100	98	100	98
20	100	93	100	93
30	97	90	97	91
40	96	91	96	91
50	95	84	95	85
60	97	88	97	90
70	94	80	94	85
80	94	71	94	78
90	89	68	89	75
100	90	55	90	65

Table 5.6: Square interval liner systems – enclosures comparison. The reference method is `hbr`, the uniform radii was set to $r = 0.01$, the symbol `'-'` means that a method returned no finite enclosure in all 100 test cases, n is the number of variables of a system.

n	ge+pre	jacobi+pre	krawczyk+pre
10	1.00430	1.00178	1.01213
20	1.00303	1.00247	1.01208
30	1.00444	1.00226	1.01004
40	1.00648	1.00251	1.01007
50	1.00678	1.00244	1.00911
60	1.00812	-	1.00939
70	1.00772	-	-
80	1.00842	-	-
90	1.00877	-	-
100	1.00749	-	-

Table 5.7: Square interval liner systems – average computation times. The uniform radii was set to $r = 0.01$, the computation times are in seconds, the symbol `'-'` means that a method returned no finite enclosure in all 100 test cases, n is the number of variables of a system.

n	ge+pre	jacobi+pre	hbr	krawczyk+pre
10	0.44	0.13	0.03	0.14
20	1.64	0.18	0.04	0.18
30	3.64	0.20	0.05	0.21
40	6.41	0.24	0.06	0.25
50	10.04	0.27	0.08	0.30
60	14.43	-	0.11	0.37
70	19.73	-	0.14	-
80	25.78	-	0.18	-
90	32.88	-	0.23	-
100	40.69	-	0.29	-

Table 5.8: Square interval liner systems – percentage of finite enclosures returned. The uniform radii is set to $r = 0.01$, n is the number of variables of a system.

n	`ge+pre`	`jacobi+pre`	`hbr`	`krawczyk+qpre`
10	98	90	98	90
20	92	80	92	82
30	83	56	83	60
40	70	19	70	28
50	59	6	59	11
60	43	0	43	1
70	31	0	31	0
80	10	0	10	0
90	5	0	5	0
100	3	0	3	0

interval hull of the preconditioned system, we still may get large overestimation, see Example 5.3.

The resulting enclosure can be further tightened/shaved. In this section we explain our method that provides such a shaving. The term "shaving" is borrowed from the area of solving constraint satisfaction problems [54, 214]. This section is an adapted version of our paper [81].

Let $\boldsymbol{x} \in \mathbb{IR}^n$ be an initial enclosure of the solution set Σ. The main idea behind shaving methods is to examine a slice of $\boldsymbol{x}$. An upper α-slice $\boldsymbol{x}(\uparrow, i, \alpha)$ is defined for ith variable and a nonnegative width α as

$$\boldsymbol{x}(\uparrow, i, \alpha)_j = \begin{cases} \boldsymbol{x}_j & \text{if } j \neq i, \\ [\overline{x}_j - \alpha, \overline{x}_j] & \text{if } j = i. \end{cases} \tag{5.8}$$

If we find that $\boldsymbol{x}(\uparrow, i, \alpha)$ contains no solution, then we cut off the slice and the tighter enclosure $\boldsymbol{x}'$ reads

$$\boldsymbol{x}'_j := \begin{cases} \boldsymbol{x}_j & \text{if } j \neq i, \\ [\underline{x}_j, \overline{x}_j - \alpha] & \text{if } j = i. \end{cases}$$

The situation is similar for the lower α-slice $\boldsymbol{x}(\downarrow, i, \alpha)$. The enclosure can be repeatedly shaved by choosing various variables for $i \in \{1, \ldots, n\}$ and their lower and upper slices. Naturally, the larger the width of a slice is the more efficient is the shaving. To develop an efficient shaving method, we need a shaving condition that decides whether there is no solution contained in a given box $\boldsymbol{x}$. Let us start with a real system first.

Lemma 5.21 (Hladík, Horáček [81])**.** *Let $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $\boldsymbol{x} \in \mathbb{IR}^n$. Then the linear system*

$$Ax = b, \quad x \in \boldsymbol{x}$$

has no solution if and only if the linear system

$$A^T w + y - z = 0, \quad b^T w + \overline{x}^T y - \underline{x}^T z = -1, \quad y, z \geq 0 \tag{5.9}$$

is solvable.

Proof. The system can be rewritten as a system of inequalities

$$\begin{aligned} Ax &\leq b, \\ -Ax &\leq -b, \\ Ix &\leq \overline{x}, \\ -Ix &\leq -\underline{x}. \end{aligned}$$

By the well-known Farkas lemma (cf. [40]), the system $Ax = b$, $\underline{x} \leq x \leq \overline{x}$ has no solution if and only if the linear system

$$A^T w_1 - A^T w_2 + y - z = 0, \quad b^T w_1 - b^T w_2 + \overline{x}^T y - \underline{x}^T z < 0, \quad w_1, w_2, y, z \geq 0$$

is solvable. After substituting $w = w_1 - w_2$ we obtain

$$A^T w + y - z = 0, \quad b^T w + \overline{x}^T y - \underline{x}^T z < 0, \quad y, z \geq 0 \tag{5.10}$$

Since every positive multiple of $(w, y, z)^T$ also solves the system (5.10), the system (5.9) can be obtained after normalization. □

Now, we see that

$$Ax = b, \quad A \in \boldsymbol{A}, \ b \in \boldsymbol{b}, \ x \in \boldsymbol{x} \tag{5.11}$$

has no solution if and only if (5.9) is solvable for each $A \in \boldsymbol{A}$ and $b \in \boldsymbol{b}$. Checking such a type of solvability (so called *strong solvability*) is known to be computationally difficult (more precisely, coNP-complete); see Chapter 11. Below, we present an adaptation of the sufficient condition developed in [70].

5.9.1 A sufficient condition for strong solvability

In this section we show a heuristic way to show that (5.9) is strongly solvable. First we try to guess a vector (w, y, z) that satisfies a special one particular instance of (5.9), where $A \in \boldsymbol{A}, b \in \boldsymbol{b}, x \in \boldsymbol{x}$, i.e., for the midpoint system $A_c x = b_c, x \in \boldsymbol{x}$. Such a vector will give us a hint how to transform the system (5.9) into a square interval system that can be solved using the before mentioned means. Enclosure of the solution set in a certain shape can prove strong solvability of (5.9).

First, we solve the linear programming problem

$$\min b_c^T w + \overline{x}^T y - \underline{x}^T z$$

subject to

$$A_c^T w + y - z = 0, \ -e \leq w \leq e, \ y, z \geq 0,$$

and denote an optimal solution by w^*, y^*, z^*. The w is additionally bounded to prevent infinite optimal value. Notice that the solution needs not to be verified as it plays a role of a heuristic only. Suppose that the optimal value is negative. If it is not the

case, then (5.9) is not solvable for $A = A_c$, $b = b_c$, and hence $\boldsymbol{x}$ contains a solution. If $y_i^* = 0$ for some i, then we fix the variable $y_i = 0$, and similarly for the entries of z^*. From now on, y, z denotes variables with the fixed values. This way we get rid of some columns (and also variables) of the system. After the fixation we obtain the potentially smaller system

$$A^T w + y - z = 0, \quad b^T w + \overline{x}^T y - \underline{x}^T z = -1, \tag{5.12}$$

where $A \in \boldsymbol{A}$ and $b \in \boldsymbol{b}$.

If it is an overdetermined system, then it has the form of $A^T w = 0$, $b^T w = -1$ (A is a square matrix, hence the only possibility to obtain an overdetermined system is $y = z = 0$). As a positive multiple of w^* solves $A_c^T w = 0$, we have that A_c is singular, which contradicts the assumption that Σ is bounded by $\boldsymbol{x}$. If (5.12) is underdetermined, then we add equations to the system to make it square. The left-hand side of the additional equations will be formed by an orthogonal basis of the null space of (5.12), and the right-hand side is calculated such that w^*, y^*, z^* solves the equations. Now we are sure that we have a square system. We denote it by

$$Cv = d, \quad C \in \boldsymbol{C}. \tag{5.13}$$

Let $v = (w, u)$ be the solution of the system, where u consists of the variables originating from (y, z). In a similar manner, let $\boldsymbol{v} = (\boldsymbol{w}, \boldsymbol{u})$ be an enclosure of the solution set of (5.13). If $\boldsymbol{u} \geq 0$, then (5.9) is solvable for each interval instance, which implies that (5.11) is not strongly solvable.

5.9.2 Computing the width of a slice

Now, we employ the above ideas to handle the problem of determining as large as possible slice of $\boldsymbol{x}$ containing no solution. Since the slice $\boldsymbol{x}$ has the form of (5.8), which depends on the parameter $\alpha \geq 0$, the interval system (5.12) and also (5.13) depend on α, too. Thus, we have to determine the largest value of α such that an enclosure to (5.13) still satisfies the nonnegativity condition $\underline{u} \geq 0$.

One possibility is to use a binary search for the optimal α. However, this would require solving plenty of interval linear systems. In the following, we rather describe a simple method for calculating a feasible, not necessary optimal, value of α.

Due to the way the lower and upper α-slice (5.8) are defined, when we take an α-slice of $\boldsymbol{x}$, such an α occurs only once in the system (5.13) (exactly in one coefficient of modified $\underline{x}$ or $\overline{x}$ after fixation). Moreover, the new system is

$$(C + \alpha E_{ij})v = d, \quad C \in \boldsymbol{C}, \tag{5.14}$$

where $E_{ij} = e_i e_j^T$ is the matrix with 1 at a certain position (i, j), and zeros elsewhere. The solution of the new system can be easily expressed.

Lemma 5.22 (Hladík, Horáček [81])**.** *Let $\widetilde{v}$ be a solution to $Cv = d$. Then the solution of $(C + \alpha E_{ij})v = d$ is*

$$\widetilde{v} - \frac{\alpha \widetilde{v}_j}{1 + \alpha C_{ji}^{-1}} C_{*i}^{-1}.$$

Proof. By the Sherman–Morrison formula for the inverse we get

$$\begin{aligned}(C + \alpha E_{ij})^{-1} &= (C + (\alpha e_i)e_j^T)^{-1} \\ &= C^{-1} - \frac{C^{-1}\left(\alpha e_i e_j^T\right)C^{-1}}{1 + e_j^T C^{-1}\alpha e_i} \\ &= C^{-1} - \frac{\alpha}{1 + \alpha C^{-1}_{ij}}\left(C^{-1}e_i\right)\left(e_j^T C^{-1}\right) \\ &= C^{-1} - \frac{\alpha}{1 + \alpha C^{-1}_{ji}} C^{-1}_{*i} C^{-1}_{j*}.\end{aligned}$$

Multiplying by d we get

$$(C + \alpha E_{ij})^{-1} d = C^{-1}d - \frac{\alpha}{1 + \alpha C^{-1}_{ji}} C^{-1}_{*i} C^{-1}_{j*} d = \widetilde{v} - \frac{\alpha \widetilde{v}_j}{1 + \alpha C^{-1}_{ji}} C^{-1}_{*i}.$$

□

Suppose we already have an enclosure $\boldsymbol{v}$ of the solution set of (5.13) for a 0-slice. By the above lemma, an enclosure of the solution set of (5.14) is

$$\boldsymbol{v} - \frac{\alpha \boldsymbol{v}_j}{1 + \alpha \boldsymbol{C}^{-1}_{ji}} \boldsymbol{C}^{-1}_{*i}.$$

Now we try to increase $\alpha > 0$ until it still satisfies several conditions of this formula. When $\alpha = 0$, then the denominator is 1. After inflating α the denominator should stay positive, otherwise it contains zero. If $\underline{\boldsymbol{C}^{-1}_{ji}} \geq 0$ then no restriction is forced on α. If $\underline{\boldsymbol{C}^{-1}_{ji}} < 0$ then it must hold that

$$-1 < \alpha \cdot \left[\underline{\boldsymbol{C}^{-1}_{ji}}, \overline{\boldsymbol{C}^{-1}_{ji}}\right],$$

which gives the first restriction on α

$$\alpha < -\frac{1}{\underline{\boldsymbol{C}^{-1}_{ji}}}. \tag{5.15}$$

In order to keep (5.11) unsolvable $\boldsymbol{u}$ must remain nonnegative after inflating the slice by α. For each variable corresponding to $\boldsymbol{u}$ it is necessary that

$$\boldsymbol{u}_k - \frac{\alpha \boldsymbol{u}_j}{1 + \alpha \boldsymbol{C}^{-1}_{ji}} \boldsymbol{C}^{-1}_{ki} \geq 0.$$

Since the left-hand side is an interval, its lower bound is required to be nonnegative, i.e.

$$\underline{u_k} - \frac{\alpha}{1 + \alpha \underline{\boldsymbol{C}^{-1}_{ji}}} \overline{\boldsymbol{u}_j \boldsymbol{C}^{-1}_{ki}} \geq 0.$$

By expressing α, we obtain

$$\alpha \leq \frac{\underline{u_k}}{\overline{\boldsymbol{u}_j \boldsymbol{C}^{-1}_{ki}} - \underline{u_k}\,\underline{\boldsymbol{C}^{-1}_{ji}}}. \tag{5.16}$$

for each k such that $\overline{\boldsymbol{u}_j \boldsymbol{C}_{ki}^{-1}} > \underline{u_k} \underline{\boldsymbol{C}_{ji}^{-1}}$.

Finally, from the formulas (5.15) and (5.16) we determine the maximal feasible α^* for inflating the α-slice. In order that the result is reliable, the formulas should be evaluated by interval arithmetic (even though they contain real variables only).

The computational cost of this method for computing α^* is low. We have to calculate $\boldsymbol{v}$, an enclosure to (5.13), and $\boldsymbol{C}_{*i}^{-1}$, which is an enclosure to the solutions set of the interval system

$$Cu = e_i, \quad C \in \boldsymbol{C}.$$

In total, we need to solve only two interval linear systems of equations. On the other hand, the computed α^* may not be the largest possible width of the slice.

5.9.3 Iterative improvement

Since α^* need not be optimal, we can think of improving it by repeating the whole process. We put $\alpha := \alpha^*$, and $\boldsymbol{v}$ will be an enclosure to (5.14). Similarly, $\boldsymbol{C}_{*i}^{-1}$ will be an enclosure to the solutions set of the interval system

$$(C + \alpha E_{ij})u = e_i, \quad C \in \boldsymbol{C}. \tag{5.17}$$

We determine the corresponding slice width α°, update $\alpha^* := \alpha^* + \alpha^\circ$ and repeat the process while improvement is significant (i.e., α° is large enough). Each iteration requires solving two interval systems, however, since the systems differ in one coefficient only, the new enclosures can be computed more effectively.

First, if we used the preconditioning by the (approximate) midpoint inverse, we can reuse the preconditioner from the previous iteration as the midpoint of (5.14) differs at the entry (i, j) only, its inverse is easily updated by using the Sherman–Morrison formula.

Updating the enclosure to (5.17) can be done even more efficiently. For a given $C \in \boldsymbol{C}$, we have by the Sherman–Morrison formula

$$(C + \alpha E_{ij})^{-1} = C^{-1} - \frac{\alpha}{1 + \alpha C_{ji}^{-1}} C_{*i}^{-1} C_{j*}^{-1}.$$

Its ith column draws

$$(C + \alpha E_{ij})_{*i}^{-1} = C_{*i}^{-1} - \frac{\alpha}{1 + \alpha C_{ji}^{-1}} C_{*i}^{-1} C_{ji}^{-1} = \frac{1}{1 + \alpha C_{ji}^{-1}} C_{*i}^{-1}.$$

Thus, $\boldsymbol{C}_{*i}^{-1}$ is updated as

$$\frac{1}{1 + \alpha \boldsymbol{C}_{ji}^{-1}} \boldsymbol{C}_{*i}^{-1}$$

without solving any system. Since the jth updated element $\boldsymbol{C}_{ji}^{-1}$ may be overestimated, we rather compute it by

$$\frac{1}{\alpha + 1/\boldsymbol{C}_{ji}^{-1}} \quad \text{instead of} \quad \frac{1}{1 + \alpha \boldsymbol{C}_{ji}^{-1}} \boldsymbol{C}_{ji}^{-1}.$$

In summary, while the first iteration needs to solve two interval systems, the others need to solve only one.

Table 5.9: Testing the shaving method without iterative improvement. The fixed radius is denoted by r, the computation time is in seconds, n is the number of variables of a system, the ratio column shows the improvement to `verifylss`, the last column shows the average number of shaved off slices.

n	r	time	ratio	# shavings
5	0.5	0.2568	0.7137	13.02
10	0.25	0.6375	0.7522	30.94
20	0.05	1.879	0.7848	61.09
50	0.025	14.58	0.8569	187.2
100	0.01	78.78	0.9049	373.8

5.9.4 Testing the shaving method

To give a hint about the cases for which the shaving method can help, let us present two tables from our previous work [81]. The method was tested on square interval systems with various fixed radii. Random square systems were generated and tested in the same way as in Example 5.16. The computations were carried out in Matlab 7.11.0.584 (R2010b) on a six-processor machine AMD Phenom(tm) II X6 1090T Processor, CPU 800 MHz, with 15579 MB RAM. Interval arithmetics and some basic interval functions were provided by the interval toolbox Intlab v6 [188]. The shaving method was run on an enclosure returned by Intlab method `verifylss`, which is a combination of a modified Krawczyk's method and the HBR method [61]. The quality of enclosures was compared using the formula (3.10). In Table 5.9 we see the results for shaving without iterative improvement of shaved slices widths (each variable is shaved only once from above and from below). In Table 5.10 the shaving method is tested on the same data but with added iterative improvement. For small interval radii the before mentioned methods return tight enough results and use of the shaving method is superfluous. However, for relatively large interval radii (such that the interval matrix is "nearly" singular) the shaving method pays off.

5.10 Some other references

There are other methods for solving interval linear systems, e.g., [14, 71]. Some methods can deal with matrices of a certain class [4, 93]. The results for systems with Toeplitz matrices are in [47]. To learn more about other concepts of solvability see, e.g., [178, 202]. More on block systems can be found in [48]. For verified solution of large systems see [192], for sparse systems see [193]. Methods for solving square interval linear systems were also compared in, e.g., [61, 143].

Table 5.10: Testing the shaving method with iterative improvement. The fixed radius is denoted by r, the computation time is in seconds, n is the number of variables of a system, the ratio column shows the improvement to `verifylss`, the last column shows the average number of shaved off slices.

n	r	time	ratio	# shavings
5	0.5	0.4977	0.6465	18.06
10	0.25	0.9941	0.6814	45.06
20	0.05	3.136	0.7161	87.77
50	0.025	26.65	0.8071	281.9
100	0.01	228.5	0.8693	946.3

6 Overdetermined interval linear systems

- The least squares solution
- Preconditioning of an overdetermined system
- Various methods for enclosing a solution set
- Subsquares method and its variations
- Comparison of methods

When a system has more equations than variables, we call it *overdetermined*. The previously described methods for solving square systems usually cannot be applied directly to them. In this chapter we first introduce the traditional approach to solving such systems via the least squares. We then discuss preconditioning for the overdetermined case. Then modification of some earlier known methods — Jacobi, Gaussian elimination — is shown. Rohn's method is introduced. All the methods are compared. The chapter is loosely based on our paper [83]. Similarly to introducing the shaving method in the previous chapter, here we introduce our subsquares method [84] that can further improve the obtained enclosure or can be used separately. Several variants of this method are developed. Its favorable properties are discussed. We end the chapter with more references to another methods for solving overdetermined and underdetermined systems.

6.1 Definition

Let us start with the formal definition.

Definition 6.1. (Overdetermined interval linear system) Let us have an interval matrix $\boldsymbol{A} \in \mathbb{IR}^{m\times n}$, where $m > n$ and an interval vector $\boldsymbol{b} \in \mathbb{IR}^m$. We call

$$\boldsymbol{A}x = \boldsymbol{b}$$

an *overdetermined* interval linear system.

To motivate the use of overdetermined interval systems see the following example.

Example 6.2. This example is borrowed from [165]. Let us have an $n \times n$ matrix A for which we want to compute an eigenvector corresponding to a known eigenvalue λ.

It is known that it can be computed as a solution of the following system

$$(A - \lambda I)v = 0.$$

However, the matrix on the left side is singular. To overcome this, let us "normalize" v. Either the first coefficient of v is 0 or v can be multiplied by a suitable scalar to make the first coefficient equal to 1. After setting $C = (A - \lambda I)$ for both cases the above system can be rewritten as the two following overdetermined systems:

$$\begin{pmatrix} c_{1,2} & \dots & c_{2,n} \\ \vdots & & \vdots \\ c_{n,2} & \dots & c_{n,n} \end{pmatrix} \cdot \begin{pmatrix} v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} -c_{1,1} \\ \vdots \\ -c_{n,1} \end{pmatrix}.$$

If the second system is solvable and x' is the solution of the system, then $(1, x')^T$ is the desired eigenvector. If the second system is not solvable, we can recursively repeat the procedure for the first system. The algorithm can be accordingly applied to an interval matrix $\boldsymbol{A}$.

6.2 The least squares approach

When most people work with an overdetermined system they understand its solution in the least squares perspective.

Definition 6.3 (Interval least squares)**.** For an overdetermined interval linear system $\boldsymbol{A}x = \boldsymbol{b}$ the least squares solution is defined as

$$\Sigma_{lsq} = \{\, x \mid A^T Ax = A^T b \text{ for some } A \in \boldsymbol{A}, b \in \boldsymbol{b} \,\}.$$

Such an approach can be found in [138] or [191] . It is easily seen that

$$\square(\Sigma) \;\subseteq\; \square(\Sigma_{lsq}).$$

Hence an enclosure of the set Σ_{lsq} is also an enclosure of the set Σ. For more information about this approach and relationship between Σ and Σ_{lsq} see [138]. The question is how to enclose Σ_{lsq}. The first idea is to solve the *interval normal equation*

$$\boldsymbol{A}^T \boldsymbol{A}x = \boldsymbol{A}^T \boldsymbol{b}.$$

This approach might not work because interval matrix multiplication can cause a huge overestimation (see Example 3.9; however later in Chapter 9 we will see that this approach can be used in some special cases). Even a use some preconditioner C

$$(C\boldsymbol{A})^T (C\boldsymbol{A})x = (C\boldsymbol{A})^T \boldsymbol{b},$$

does often not work either. Anyway, we can use an equivalent expression for the least squares formula (again see [138, 191])

$$\begin{pmatrix} I & \boldsymbol{A} \\ \boldsymbol{A}^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ 0 \end{pmatrix}.$$

Such a system is a square system and hence the methods from the previous chapter can be applied. After computing an enclosure $(\boldsymbol{y}, \boldsymbol{x})^T \in \mathbb{IR}^{m+n}$ of its solution set, the second part $\boldsymbol{x}$ is an enclosure of Σ_{lsq}.

Since the returned interval vector contains the solution of the interval least squares, this method returns a nonempty enclosure even if the original system is unsolvable. Another drawback is that if the original system is of size $m \times n$ we have to solve a new one of size $(m+n) \times (m+n)$. That is why we often refer to this method as *supersquare* approach.

Example 6.4. For the overdetermined system $\boldsymbol{A}x = \boldsymbol{b}$ with

$$\boldsymbol{A} = \begin{pmatrix} [-0.8, 0.2] & [-20.1, -19.5] \\ [-15.6, -15.2] & [14.8, 16.7] \\ [18.8, 20.1] & [8.1, 9.5] \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} [292.1, 292.7] \\ [-361.9, -361.1] \\ [28.4, 30.3] \end{pmatrix},$$

the solution set, the hull of the original system, the hull of the supersquare system and the hull of the interval normal equation are displayed in Figure 6.1.

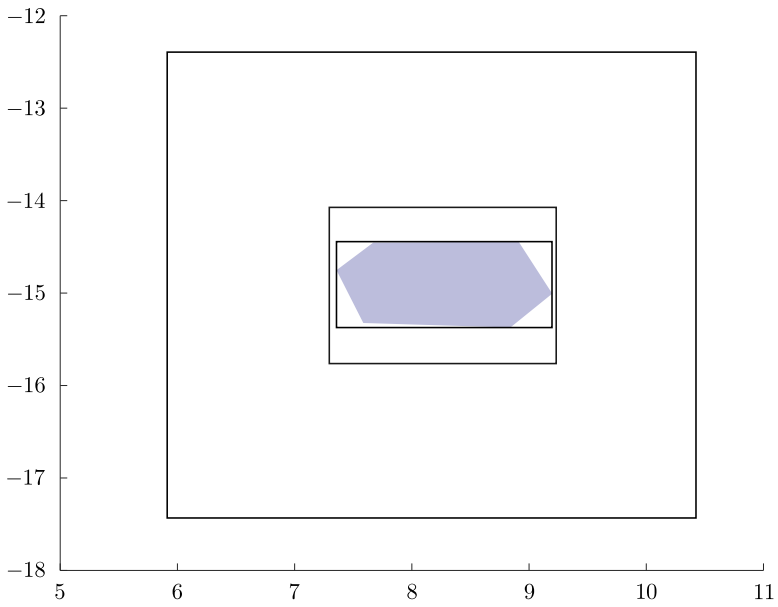

Figure 6.1: The interval least squares. Dark area is the solution set of the original system, the smallest rectangle is the hull of the original system, the intermediate rectangle is the hull of the supersquare system and the largest rectangle is the hull of the interval normal equation.

When solving a system by the supersquare approach, the new matrix is symmetric, hence dependencies occur in the new system (each interval coefficient from the original system is used twice in the new system, that is why when we choose one number from the first interval we should choose the same value in the second one to avoid overestimation). Here, methods dealing with dependencies between coefficients in interval linear systems could be used (e.g., [67, 152, 196]).

6.3 Preconditioning of an overdetermined system

Also an overdetermined system needs often to be transformed to a form that avoids expansion of intervals. It is achieved by multiplication with a preconditioner C of a corresponding size.

$$\boldsymbol{A}x = \boldsymbol{b} \quad \mapsto \quad C\boldsymbol{A}x = C\boldsymbol{b}.$$

A choice of C is proposed in [60]. Let $\boldsymbol{A}$ be of size $m \times n$. Transform its midpoint A^c into an upper trapezoidal form, Gaussian elimination with rounded arithmetics can be applied since we do not need exact result. The same elimination operations are simultaneously performed on an identity matrix of order m. Such a matrix is then taken as a preconditioner.

In [218] there is yet another slightly different possibility for preconditioning an overdetermined system by

$$C \approx \begin{pmatrix} A_1^c & 0 \\ A_2^c & I \end{pmatrix}^{-1}, \tag{6.1}$$

where A_1^c consists of the first n rows of A^c and A_2^c consists of the remaining $m - n$ rows of A^c, 0 is the $(m-n) \times n$ matrix of all zeros and I is the identity matrix of size $(m-n) \times (m-n)$. These preconditioners were designed for use with interval Gaussian elimination, that is why they might not be suitable for all methods. Later we will see that some methods use their own preconditioners (e.g., Rohn's method in Section 6.6). If not stated otherwise, when using a preconditioner for an overdetermined system, we prefer the second choice, since it is a generalization of the midpoint inverse preconditioning for square systems.

After an overdetermined system is preconditioned with C, the center of the resulting matrix is approximately of the shape

$$\begin{pmatrix} I \\ 0 \end{pmatrix},$$

where I is the $n \times n$ identity matrix and 0 is the $(m-n) \times n$ matrix of all zeros. The reasoning is illustrated by the Figure 6.2.

6.4 Gaussian elimination

Interval Gaussian elimination for overdetermined systems was proposed by Hansen in [60]. The idea is pretty the same as for square interval systems: rows are eliminated in the same way as explained in Section 5.6.1. The only difference is that the matrix $(\boldsymbol{A} \mid \boldsymbol{b})$ of size $m \times (n+1)$ corresponding to $\boldsymbol{A}x = \boldsymbol{b}$ is eliminated into the following shape:

$$(\boldsymbol{A} \mid \boldsymbol{b}) \quad \mapsto \quad \left(\begin{array}{cc|c} \boldsymbol{C} & \boldsymbol{d} & \boldsymbol{e} \\ 0 & \boldsymbol{u} & \boldsymbol{v} \end{array}\right),$$

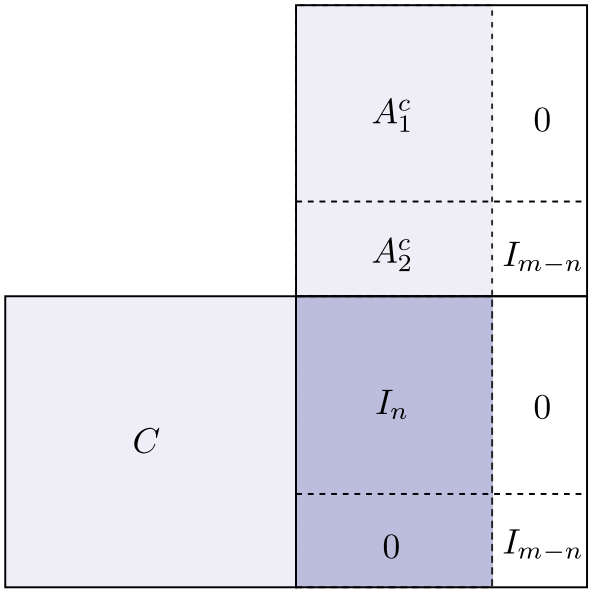

Figure 6.2: Illustration of the preconditioning by C computed by (6.1). The darkest area corresponds to the midpoint matrix of the preconditioned system.

where $\boldsymbol{C}$ is an $(n-1)\times(n-1)$ interval matrix in row echelon form, $\boldsymbol{d}, \boldsymbol{e}$ are $(n-1)\times 1$ interval vectors, 0 is an $(m-n+1)\times(n-1)$ matrix of all zeros and $\boldsymbol{u}, \boldsymbol{v}$ are $(m-n+1)\times 1$ interval vectors.

The vectors $\boldsymbol{u}, \boldsymbol{v}$ form $m-n+1$ interval equations in the shape

$$\boldsymbol{u}_i x_n = \boldsymbol{v}_i \quad \text{for } i = 1, \ldots, (m-n+1).$$

The solution of these equations gives the following enclosure for the variable x_n

$$\boldsymbol{x}_n = \bigcap_{i:\, 0 \notin \boldsymbol{u}_i} (\boldsymbol{v}_i / \boldsymbol{u}_i)\,.$$

If the intersection is empty, then the system has no solution. Nonetheless, if the intersection is unbounded, it can either mean that the solution set of the system is unbounded or huge overestimation due to large number of interval operations occurred. The enclosures for the other variables can be obtained using the backward substitution as described in Section 5.6.1.

6.5 Iterative methods

In the previous chapter we introduced three iterative methods for solving square interval linear systems – the Jacobi, the Gauss–Seidel and Krawczyk's method. After we apply preconditioning from Section 6.3 only the first n rows possibly do not contain zeros on the diagonal. That is why we can apply an iterative method to the square

subsystem consisting of the first n equations of the preconditioned system. The solution set of the original overdetermined interval system must lie inside the solution set of a square subsystem (*subsquare*).

Proposition 6.5. *Let Σ be the solution set of $\boldsymbol{A}x = \boldsymbol{b}$ and let Σ_{subs} be the solution set of a square subsystem of $\boldsymbol{A}x = \boldsymbol{b}$. Then*

$$\Sigma \subseteq \Sigma_{subs}.$$

Proof. The original system $\boldsymbol{A}x = \boldsymbol{b}$ has more equations that can put no or more restriction on the solution set of the square subsystem. □

6.6 Rohn's method

We would like to mention the method introduced by Rohn [174]. In his paper more information and theoretical insight can be found. The following theorem is the basis of the method.

Theorem 6.6. *Let $\boldsymbol{A}x = \boldsymbol{b}$ be an overdetermined interval linear system with $\boldsymbol{A}$ being an $m \times n$ interval matrix and Σ being its solution set. Let R be an arbitrary real $n \times m$ matrix, let x_0 and $d > 0$ be arbitrary n-dimensional real vectors such that*

$$Gd + g < d, \tag{6.2}$$

where

$$G = |I - RA_c| + |R|A_\Delta,$$

and

$$g = |R(A_c x_0 - b_c)| + |R|(A_\Delta |x_0| + b_\Delta).$$

Then

$$\Sigma \subseteq [x_0 - d, x_0 + d].$$

The question is how to find the vector d, the matrix R and the vector x_o. To compute d, we can, for example, rewrite the inequality (6.2) as

$$d = Gd + g + \varepsilon, \tag{6.3}$$

for some small vector $\varepsilon > 0$. Then, start with $d = 0$ and iteratively refine d. This algorithm will stop after a finite number of steps if $\varrho(G) < 1$ holds.

In [82] we proposed another option for finding d. One can rewrite the equality 6.3) as

$$(I - G)d = g + \varepsilon,$$

Table 6.1: Rohn's method – testing of the iterative and direct approach for finding d from Example 6.7. The second column displays the average ratio of the vectors d returned by the two methods computed by the formula (6.4), the last two columns display the average computation times, $m \times n$ is the size of a system matrix.

$m \times n$	rat	t iterative	t direct
5×3	1.0000	0.0047	0.0012
15×10	1.0000	0.0067	0.0024
25×21	1.0000	0.0068	0.0023
35×23	1.0000	0.0066	0.0024
50×35	1.0000	0.0067	0.0024
73×55	1.0000	0.0072	0.0024
100×87	1.0000	0.0077	0.0028
200×170	1.0000	0.0066	0.0024

and solve the real system directly. After finding a solution, the vector d is tested for positivity. In the two following examples we test the two methods on random overdetermined systems.

Example 6.7. A solvable random overdetermined system is generated in the following way. First, a midpoint matrix A_c is generated by uniformly randomly and independently choosing its coefficients from interval $[-10, 10]$. Second, a random solution vector x is generated also with coefficients from interval $[-10, 10]$. The right-hand side b_c is then computed as $b = A_c x$. An interval system is obtained by wrapping the A_c, b_c with intervals having a fixed radius r. Here, we test the systems for $r = 10^{-3}$. The small positive vector ε has its coefficients 10^{-6}. The iteration limit is 50. The results of the comparison are shown in Table 6.1. The second column shows the average ratios of d^{it}, d^{dir} returned by the iterative and direct method respectively. The ratio is computed as average ratio of the coefficients of the two n-dimensional vectors

$$\text{rat} = \left(\sum_{i=1}^{n} \frac{d_i^{it}}{d_i^{dir}} \right) / n. \tag{6.4}$$

For each size we test on 100 random systems. The ratios in Table 6.1 show that both methods return basically identical results. The next two columns show average computation times (using the LAPTOP setting). Even thought, the measured times are very small, the results are in favor of the direct method. The results, however, depend on the method used for solving real linear systems (here we used Octave's `linsolve`). In conclusion, we rather prefer the direct method for computation of d because this way we do not have to care about properly setting ε.

We still have to determine x_0 and R. Rohn recommends to take

$$x_0 \approx R b_c, \quad R \approx (A_c^T A_c)^{-1} A_c^T,$$

but not necessarily. Rohn suggests that Theorem 6.6 provides an instrument for iterative improvement of enclosure. We do not have to use only A_c to compute R, we can take any (e.g., random) matrix $A \in \boldsymbol{A}$, compute an enclosure and then intersect it with the old one. We can repeat this process as many times we want and provide an iterative improvement of the enclosure.

Example 6.8. For the overdetermined system from Example 6.4 we selected 20 random $A \in \boldsymbol{A}$ to compute $R \approx (A^T A)^{-1} A^T$. We also included $A = A_c$. For this system $A = A_c$ plays a prominent role, since for no other A was the enclosure better. The resulting boxes are displayed in Figure 6.3.

When the random systems were generated as in Example 6.7 for the same sizes, and the first enclosure was computed using $R \approx (A_c^T A_c)^{-1} A_c^T$, then no improvement of the enclosure was detected after 50 such iterations. Hence, it seems that testing other choices of R does not pay off in this case.

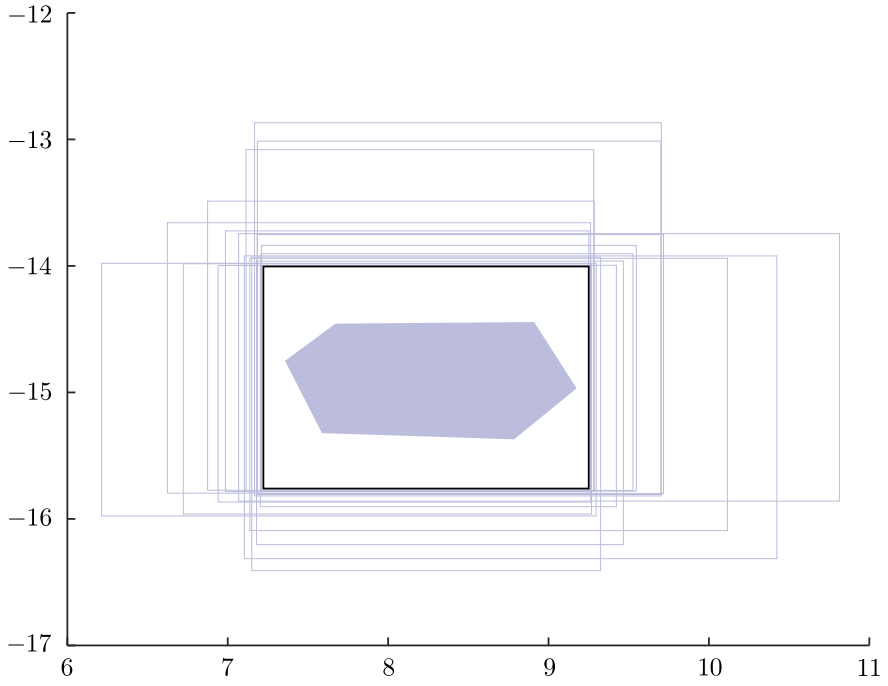

Figure 6.3: Result of Rohn's algorithm from Example 6.8 for various selections of $R \approx (A^T A)^{-1} A^T$ for $A \in \boldsymbol{A}$. The dark area is the solution set of the system. The darkest rectangle is the enclosure for $R \approx (A_c^T A_c)^{-1} A_c^T$. Other 20 enclosures (boxes) correspond to 20 random choices of $A \in \boldsymbol{A}$.

6.7 Comparison of methods

In this section the previously described methods for solving overdetermined systems are compared. First, we start with a simple examples revealing that the methods do not return empty enclosure for an unsolvable system. Next we compare the methods

on random overdetermined systems with various fixed radii of interval coefficients. We are aware that it is not completely fair to compare direct and iterative methods together. However, a comparison will give us at least a hint of the properties of such methods. The tested methods are:

- `rohn` – Rohn's method for overdetermined systems with direct computation of d,
- `jacobi` – preconditioned system with the Jacobi iterative method implemented in matrix multiplication form applied to the first n equations, with maximum number of iterations set to 20, and $\varepsilon = 10^{-5}$,
- `lsq` – enclosing the least squares solution, by transforming a system into a supersquare and then solving with the HBR mehod,
- `ge` – Gaussian elimination for overdetermined systems with preconditioning and mignitude pivoting.

When a method name has the suffix `-pre`, it means that it is applied without preconditioning, the suffix `+pre` means the preconditioning with midpoint inverse was used. First let us test the methods for an unsolvable system.

Example 6.9. Let us have the unsolvable system with the following A_c, b_c and fixed radii of interval coefficients set to $r = 0.1$.

$$A_c = \begin{pmatrix} -6 & 2 & -9 \\ 0 & 8 & 6 \\ 7 & -9 & -5 \\ 4 & -5 & -8 \\ -5 & -7 & 6 \end{pmatrix}, \quad b_c = \begin{pmatrix} 9 \\ 54 \\ -120 \\ -95 \\ 57 \end{pmatrix}.$$

The returned enclosures are in Table 6.2. We can see that no method can detect unsolvability of the system.

Table 6.2: Comparison of enclosures returned by various methods applied on the unsolvable system from Example 6.9.

$\boldsymbol{x}$	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$
`rohn`	$[-9.4682, -8.6938]$	$[2.6762, 3.2171]$	$[5.2755, 5.7940]$
`jacobi+pre`	$[-10.804, -9.2545]$	$[1.4635, 2.8699]$	$[5.5646, 6.7552]$
`lsq`	$[-9.4951, -8.6841]$	$[2.6655, 3.2364]$	$[5.2681, 5.8091]$
`ge+pre`	$[-10.404, -7.9421]$	$[2.6365, 3.6731]$	$[5.0720, 5.8993]$
`ge-pre`	$[-10.804, -9.2545]$	$[1.4653, 2.8699]$	$[5.5671, 6.7552]$
`hull`	$\emptyset$	$\emptyset$	$\emptyset$

Table 6.3: Overdetermined interval linear systems – average ratios of enclosures returned by various methods compared to the hull, $m \times n$ is the size of a system matrix.

$m \times n$	`rohn`	`jacobi`	`lsq`	`ge`
5×3	1.114	7.961	1.114	7.961
15×13	1.038	4.538	1.039	4.538
35×23	1.116	14.963	1.116	14.962
50×35	1.101	11.946	1.101	11.945
100×87	1.043	11.043	1.047	17.562

Next, all the methods are compared on 100 random solvable systems for each size. Such systems were generated by using the same procedure as described in Example 6.7. The radii of interval coefficients were fixed to $r = 10^{-4}$. The average ratios of enclosures are compared using the formula (3.9). They are displayed in Table 6.3 and the average computation times are in Table 6.4. For such a "small" radii, the returned enclosures lie in one orthant, that is why we compared the quality of enclosures to the hull. The hull was computed as in Section 5.2, however, because of computation time reasons, using only nonverified linear programming. The results must be hence taken with caution, nevertheless, empirically for such systems a nonverified hull is nearly identical to the verified hull.

The `ge` and `jacobi` return comparable enclosures. So do `rohn` and `lsq`. We believe it is no coincidence. The way `jacobi` and `ge` are defined for overdetermined systems suggest that only the first n rows are basically used for computing an enclosure. We explain the similarity of `rohn` and `jacobi` by the use of the matrix R in `rohn`. Since it is basically the Moore-Penrose pseudoinverse of A_c the solution set Σ of the preconditioned system and Σ_{lsq} tend to coincide for small radii. In Table 6.5 we show that this is not the case for larger radii.

The `rohn` is the winner from computation time perspective, since the disadvantage of need to solve much larger supersquare system in `lsq` manifests itself. The `ge` shows excessive demands regarding computational time. In all cases methods returned finite enclosures, except for the size 100×87 `jacobi` returned infinite enclosures in 5 cases, `ge` in 2 cases.

6.8 Subsquares approach

In this section we present a scheme for solving overdetermined systems, which we developed in [84]. This method uses algorithms described in Chapter 5 and applies them on selected square subsystems of the original system. Although, we mentioned the term "subsquare" earlier in Section 6.5, we rather define it more formally here.

Definition 6.10 (Subsquare)**.** By a *square subsystem* or *subsquare* of an overdetermined system $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is of size $m \times n$, we mean any choice of n equations