and $u = (1,1,1)^T > 0$ it holds that $\langle \boldsymbol{A} \rangle u > 0$. Finally, it is inverse nonnegative since

$$\underline{A}^{-1} = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 3 \end{pmatrix} \geq 0.$$

and, according to the Sherman–Morrison formula,

$$\overline{A}^{-1} = \begin{pmatrix} \frac{3}{1+3\varepsilon} & \frac{1}{1+3\varepsilon} & 0 \\ \frac{1}{1+3\varepsilon} & \frac{4+11\varepsilon}{1+3\varepsilon} & 1 \\ 0 & 1 & 3 \end{pmatrix} \geq 0.$$

4.6 Strongly regular matrices

Usually, before computing with an interval matrix some kind of preconditioning is applied. It means a matrix $\boldsymbol{A}$ is multiplied with a real regular matrix C

$$\boldsymbol{A} \mapsto C\boldsymbol{A}.$$

Such a resulting matrix might possess properties more suitable for further processing (for example it will prevent growth of intervals' widths). A usual preconditioner is $C = A_c^{-1}$. Of course, in finite arithmetic we can not often get precise midpoint inverse. Nevertheless, we can use $C \approx A_c^{-1}$. If we precondition with the midpoint inverse, first thing we want is $A_c^{-1}\boldsymbol{A}$ to be regular.

Definition 4.32 (Strongly regular matrix)**.** Let $\boldsymbol{A}$ be a square interval matrix. Let A_c be regular. If $A_c^{-1}\boldsymbol{A}$ is regular, then $\boldsymbol{A}$ is called *strongly regular.*

In [139] there are many useful conditions for deciding strong regularity.

Theorem 4.33. *Let $\boldsymbol{A}$ be a square interval matrix and A_c be regular, then the following statements are equivalent:*

1. *$\boldsymbol{A}$ is strongly regular,*
2. *$\boldsymbol{A}^T$ is strongly regular,*
3. *$\varrho(|A_c^{-1}|A_\Delta) < 1$,*
4. *$\|I - A_c^{-1}\boldsymbol{A}\| < 1$ for some consistent matrix norm,*
5. *$A_c^{-1}\boldsymbol{A}$ is an H-matrix.*

Note that the statement *3.* is a sufficient condition for regularity, hence every strongly regular matrix is regular. Let us comment on the statement *4.* When, assuming the exact arithmetics, $\boldsymbol{A}$ is preconditioned by A_c^{-1}, the resulting matrix has an identity matrix I as its midpoint. Hence we can view the resulting interval matrix as wrapping around I. To maintain its regularity, such a matrix cannot reach too far from I, which we can formulate as $\|I - A_c^{-1}\boldsymbol{A}\| < 1$.

We can also apply preconditioning from both sides

$$\boldsymbol{A} \mapsto C_1 \boldsymbol{A} C_2.$$

However, that does not extend the class of strongly regular matrices [139].

Theorem 4.34. *Let $\boldsymbol{A}$ be a square interval matrix and C_1, C_2 be square real matrices. Suppose that $C_1 \boldsymbol{A} C_2$ is an H-matrix, then $\boldsymbol{A}$ is strongly regular.*

Example 4.35. If we set $C_1 = C_2 = I$ in Theorem 4.34, then it implies that every H-matrix is strongly regular.

Here we have an example from [139] that not every regular matrix is strongly regular.

Example 4.36. The matrix

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

is regular (e.g., since $\det(\boldsymbol{A}) = [1,5] > 0$). We have

$$A_c^{-1} = \begin{pmatrix} 0.5 & -0.5 \\ 0.5 & 0.5 \end{pmatrix} \quad \text{and} \quad A_\Delta = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

hence,

$$\varrho(|A_c^{-1}|A_\Delta) = \varrho \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = 1,$$

which, according to Theorem 4.33, means $\boldsymbol{A}$ is not strongly regular.

Example 4.37. The matrix from Example 4.15 is strongly regular.

Example 4.38. Regular matrices with inverse nonnegative midpoint are strongly regular [139]. Hence inverse nonnegative matrices are strongly regular too.

However, not every strongly regular matrix is inverse nonnegative.

Example 4.39. The matrix

$$\boldsymbol{A} = \begin{pmatrix} [0,1] & 1 \\ -1 & [0,1] \end{pmatrix},$$

is regular (because $\det(\boldsymbol{A}) = [1,2]$), and it is strongly regular because $\varrho(|A_c^{-1}|A_\Delta) = 0.6 < 1$. Nevertheless, its inverse is

$$\boldsymbol{A}^{-1} = \begin{pmatrix} [0.5,1] & [-1,-0.5] \\ [0.5,1] & [0,1] \end{pmatrix},$$

which means $\boldsymbol{A}$ is not inverse nonnegative.

4.7 Mutual relations

For the sake of clarity, the relations between the mentioned classes of interval matrices are captured in Figure 4.1.

4.8 More on interval matrices

There is a survey on properties of matrices that are computable in polynomial time is [75]. Discussion about the relationship between regularity and singularity can be found in [186]. We saw that when upper and lower bound matrix is an M-matrix then the whole interval matrix is an M-matrix. When checking a certain property for certain boundary matrices implies the property for all matrices included in the interval matrix, it is called *interval property*. There is a survey paper on such matrices [50]. A survey devoted to checking various matrix properties is [173]. Results regarding positive definiteness, stability and P-matrices can be found there. More on interval P-matrices can be found, e.g., in [20, 73, 185]. Other results regarding stability are [31, 63, 199]. For more on totally nonnegative interval matrices see [1, 44]. To know more about sign regular matrices see [2, 45]. For information about matrices with parametric dependencies see [76, 153]. Complexity issues related to interval matrices can be found in [112, 85]. More about inverse interval matrix can be found in [169, 183].

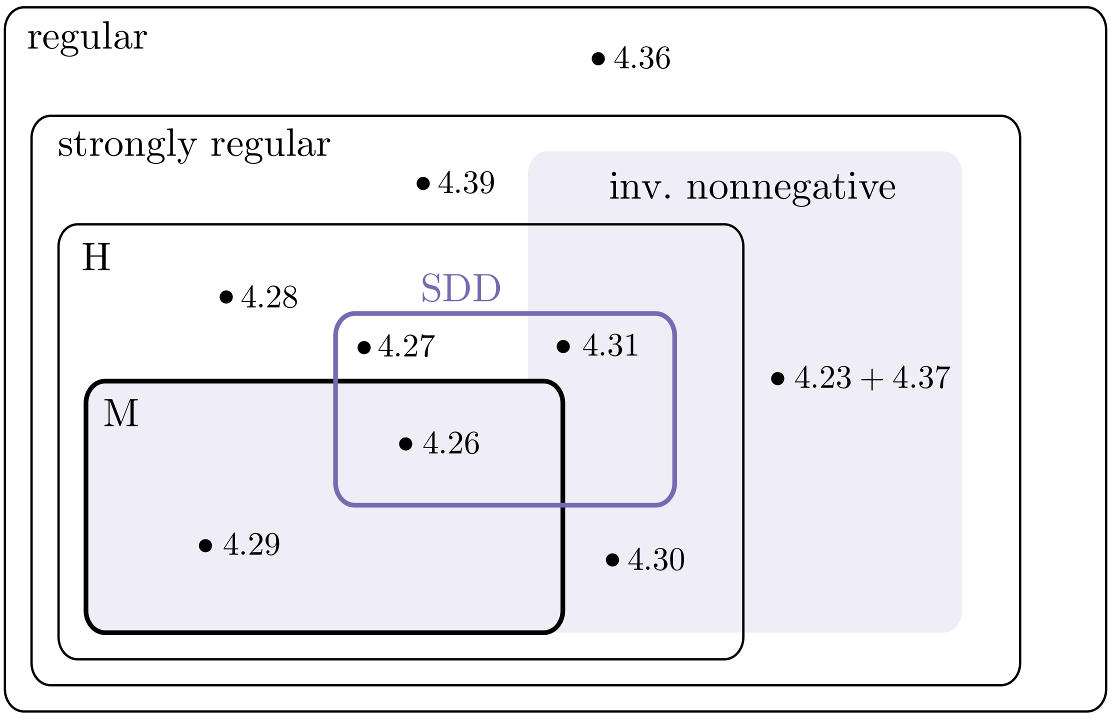

Figure 4.1: Inclusion relations between the mentioned classes of interval matrices; **SDD** (strictly diagonally dominant matrices), **M** (M-matrices), **H** (H-matrices). The numbers refer to examples that show existence of an interval matrix lying in the intersection of two particular classes. The darker area corresponds to the set of inverse nonnegative matrices.

5 Square interval linear systems

- Solution set of an linear system
- Interval hull and enclosure
- Verified solution of a real system
- Preconditioning
- Direct and iterative methods for enclosures
- Comparison of methods
- Shaving method

Interval linear systems form a crucial part of interval linear algebra. Moreover, many linear algebraic problems can be transformed to solving a square interval system. That is why in this chapter we deal with solving square interval systems first. We define what do we mean by the solution set of an interval linear system. We discuss a characterization of a solution set. As this set might be of a complex shape it is usually enclosed with an n-dimensional box for further processing. We address computation of the tightest n-dimensional box enclosing this set (the hull). However, computing the hull is an NP-hard problem, that is why we sometimes need to be satisfied with a larger box (an enclosure). Of course, the tighter the enclosure is the better. There are various methods for computing enclosures of the solution set. We divide them into two groups – the iterative methods and direct methods. We introduce some representatives for each group – Krawczyk's method, the Jacobi and Gauss–Seidel method as iterative methods and the Hansen–Bliek–Rohn–Ning–Kearfott–Neumaier method and Gaussian elimination as direct methods. Sometimes a verified solution of a real system is needed, hence we describe Rump's ε-inflation method, which can be also used as an enclosure method. The related topics such as preconditioning, finding an initial enclosure and stopping criteria are discussed as well. At the end we compare the mentioned methods, since we need to know which method to use when solving of square interval systems is later needed as a subtask of a problem. We also introduce and demonstrate our shaving method introduced in [81] that is able to further improve obtained enclosures. In this chapter we deal only with square interval systems, overdetermined systems are described in the next chapter. We conclude the chapter with a list of further references.

5.1 Solution set and its characterization

For the sake of clarity let us first define a system of interval linear equations or, as we abbreviate it, an *interval linear system*. It can be defined as a set of all real systems that are contained within bounds given by an interval matrix and an interval vector.

Definition 5.1 (Interval linear system)**.** For an $m \times n$ interval matrix $\boldsymbol{A}$ and an m-dimensional interval vector $\boldsymbol{b}$ we call the following structure an *interval linear system*

$$\{Ax = b \mid A \in \boldsymbol{A}, b \in \boldsymbol{b}\}.$$

Note that when a matrix (vector) is selected from an interval matrix (vector) each coefficient is selected independently. For the sake of simplicity it will be denoted by

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

When $m = n$ holds, in another words a system has the same number of variables and equations, we call it a *square system*. In practical applications, descriptions of problems often lead to square systems. If $m > n$ then a system is called *overdetermined* and if $m < n$, then a system is called *underdetermined*. Moreover, solving of square systems will be in later chapters useful for dealing with other problems, e.g., solving overdetermined systems, computing determinants, constructing the least squares regression, etc.

First, it is necessary to define what is meant by the solution set of an interval linear system.

Definition 5.2 (Solution set)**.** The solution set Σ of an interval linear system $\boldsymbol{A}x = \boldsymbol{b}$ is the defined as follows

$$\Sigma = \{\, x \mid Ax = b \text{ for some } A \in \boldsymbol{A}, b \in \boldsymbol{b}\,\}.$$

Example 5.3. The following examples are inspired by [116]. The solution set of the system

$$\begin{pmatrix} [2,4] & [-2,1] \\ [-1,2] & [2,4] \end{pmatrix} x = \begin{pmatrix} [-2,2] \\ [-2,2] \end{pmatrix},$$

forms four spikes. To obtain its top left spike we take its subsystem

$$\begin{pmatrix} [2,4] & [0,1] \\ [0,2] & [2,4] \end{pmatrix} x = \begin{pmatrix} [-2,0] \\ [0,2] \end{pmatrix}.$$

Both solution sets are depicted in Figure 5.1.

If the solution set Σ corresponding to $\boldsymbol{A}x = \boldsymbol{b}$ is nonempty, we call the system *(weakly) solvable*. If it is empty, we call the system *unsolvable*. If every real system $(Ax = b) \in (\boldsymbol{A}x = \boldsymbol{b})$ is solvable, we call the interval system *strongly solvable*. We deal with unsolvability and solvability more in Chapter 7.

It can be seen that a solution set may be of a complicated shape – it is generally nonconvex, however, it is convex in each orthant. The shape of the solution set is described by the following theorem stated in [144].

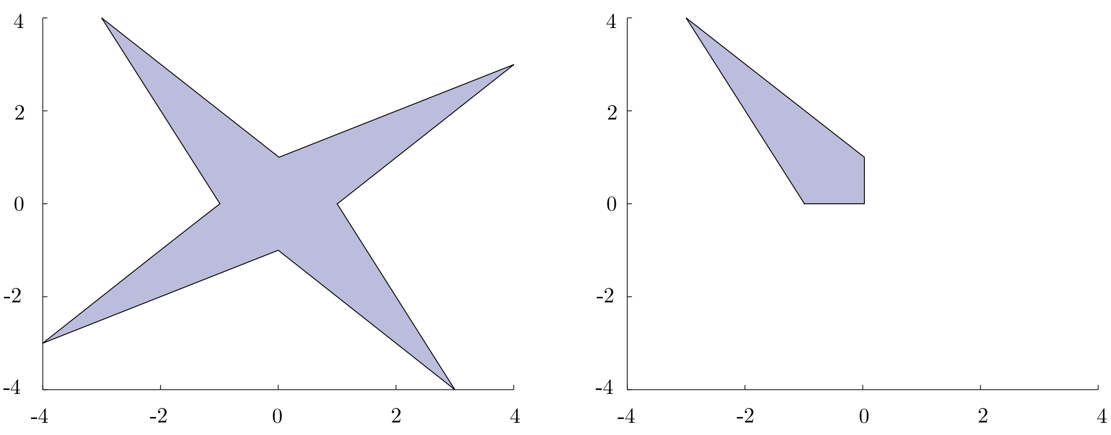

Figure 5.1: The solution sets of the interval linear systems from Example 5.3.

Theorem 5.4 (Oettli–Prager)**.** *Let us have an interval linear system $\boldsymbol{A}x = \boldsymbol{b}$. Vector $x \in \mathbb{R}^n$ is a solution of this system ($x \in \Sigma$) if and only if*

$$|A_c x - b_c| \leq A_\Delta |x| + b_\Delta.$$

Proof. There are various proofs of this theorem. The first proof was given in [144], a constructive proof is given in [178] and a simple proof can be found in [139]. □

Such nonlinear inequalities can be for each orthant transformed into a set of linear inequalities. That explains the convexity of the solution set in each orthant. We will show the transformation in the next section.

5.2 Interval hull

Because a solution set might be of a complicated shape, for practical use its simplified representation is more suitable. The simplest idea is to enclose it by an n-dimensional box aligned with axes. If such a box is the tightest possible we call it the interval hull. Let us define it more formally.

Definition 5.5 (Interval hull)**.** When $\boldsymbol{A}$ is regular and Σ is the solution set of $\boldsymbol{A}x = \boldsymbol{b}$, the interval vector $\boldsymbol{h} = [\underline{h}, \overline{h}]$ given by

$$\begin{aligned}\underline{h}_i &= \min_{x\in\Sigma} x_i,\\ \overline{h}_i &= \max_{x\in\Sigma} x_i \quad i = (1, \ldots, n),\end{aligned}$$

is called the *interval hull.*

The formula in the Oettli–Prager theorem can be rewritten using linear inequalities only. The absolute values can be rewritten in the following way. We can get rid of the first one by breaking it down into the two cases

$$\begin{aligned} A_c x - b_c &\leq A_\Delta |x| + b_\Delta, && (5.1)\\ -(A_c x - b_c) &\leq A_\Delta |x| + b_\Delta. && (5.2)\end{aligned}$$

The second absolute value can be rewritten with the use of knowledge of the orthant we currently work with. The following holds

$$|x| = D_z x, \quad \text{where } z = \operatorname{sign} x.$$

That gives rise to the condition

$$0 \leq D_z x. \tag{5.3}$$

For every orthant the conditions (5.1), (5.2) and (5.3) form a system of linear inequalities. Therefore we can use linear programming. Generally, we have to solve $(2^n \times 2n)$ linear programming problems (for each orthant in each coordinate compute the upper and lower bound). That is obviously too much computing. However, if we know some enclosure of the solution set (known in advance or computed with some of the later mentioned methods), then we can apply linear programming to only those orthants across which the enclosure stretches.

Of course, in both cases the linear programming needs to be verified. For information about its verification see, e.g., [16, 106].

If we are unlucky, we must test all the exponentially many orthants. It is no surprise since computing the exact hull is an NP-hard problem [184]. However, as we will further see, for certain classes of matrices (or systems), the orthant search and even linear programming could be avoided and there is much more convenient way to compute the interval hull.

There are other methods for computing the hull which are also possibly of exponential nature [3, 59, 94, 139, 176].

5.3 Enclosure of Σ

As computing the hull is generally a computationally difficult task, we must sometimes lower our demands and compute only an interval box containing the solution set Σ. Of course, the tighter is the box the better.

Definition 5.6 (Enclosure)**.** For an interval system $\boldsymbol{A}x = \boldsymbol{b}$ any $\boldsymbol{x} \in \mathbb{IR}^n$ such that

$$\Sigma \subseteq \boldsymbol{x}$$

is called an *enclosure*.

As enclosing the solution set of an interval linear system is a crucial task applicable to many other problems, in the next two chapters the main goal will be the following:

Problem: Compute a tight enclosure of the solution set of $\boldsymbol{A}x = \boldsymbol{b}$.

In another words our goal is always to compute the tightest enclosure in a reasonable amount of time. Note that in this work we address both the computation of hull and the computation of enclosure as *solving*. Sometimes, we refer to an enclosure of the solution set of an interval linear system just as *enclosure of the interval linear system*.

In order to obtain a finite enclosure we need the solution set to be bounded. Rohn proved in [171] that the solution set is bounded if and only if $\boldsymbol{A}$ is regular. Regularity can be checked by means discussed in Chapter 4.

In the rest of the chapter we are going to introduce various approaches to this problem. Before that, we briefly explain the concept of preconditioning borrowed from numerical mathematics.

5.4 Preconditioning of a square system

In the case of interval systems preconditioning means transforming the system into a more feasible form for further processing. Mostly, to overcome uncontrollable growth of widths of intervals during computation. Here we assume that $\boldsymbol{A}$ is a square matrix. The general transformation is

$$\boldsymbol{A}x = \boldsymbol{b} \quad \mapsto \quad (C\boldsymbol{A})x = (C\boldsymbol{b}),$$

where C is a real square matrix of a corresponding size.

The most promising choice is usually $C = A_c^{-1}$. It is also optimal from a certain viewpoint [139]. Such a preconditioning leads to a new system where the matrix has I as a midpoint (if we assume exact arithmetics). Such matrices are theoretically nice. As we saw in the previous chapter our goal is to transform $\boldsymbol{A}$ into an H-matrix. That is why strongly regular matrices play a prominent role in our problems, specially in solving of interval linear systems.

Because we work only with finite arithmetics a typical choice is preconditioning with approximate midpoint inverse, i.e., $C \approx A_c^{-1}$.

Unfortunately, beside the positive effect of preconditioning, it will often enlarge the solution set of the new system. This is the cost we need to pay.

Example 5.7. Let us take the first system from Example 5.3 and use two preconditioning matrices

$$C_1 \approx A_c^{-1} = \begin{pmatrix} 3 & -0.5 \\ 0.5 & 3 \end{pmatrix}^{-1}, \quad C_2 \approx \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}^{-1}.$$

The solution sets of the two new resulting systems are depicted in Figure 5.2. In the first case the hull of the new system is

$$\boldsymbol{h}_1 = \begin{pmatrix} [-14, 14] \\ [-14, 14] \end{pmatrix}.$$

However, in the second case the hull remains the same.

$$\boldsymbol{h}_2 = \begin{pmatrix} [-4, 4] \\ [-4, 4] \end{pmatrix}.$$

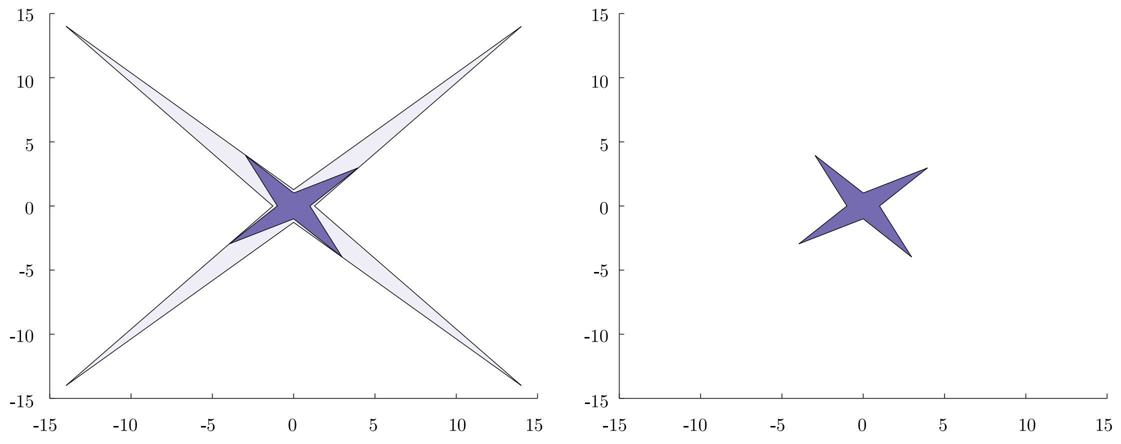

Figure 5.2: The two preconditionings from Example 5.7 – C_1(left) and C_2 (right). The darker area is the original solution set, the lighter area is the solution set of the preconditioned system.

This is no coincidence since preconditioning with a diagonal matrix D where $D_{ii} \neq 0$ preserves the original solution set [143]. This can be used for example when $\boldsymbol{A}$ is strictly diagonally dominant [189].

The preconditioning by A_c^{-1} is not always the optimal choice [74, 103]. There are other possibilities [59, 105] of preconditioning. In some cases the preconditioning is not favorable, e.g., when applying Gaussian elimination on a system where the matrix is an M-matrix [139]. In some cases preconditioning can even be avoided [201].

5.5 ε-inflation method

In further text we are also going to need a verified enclosure of a solution of a real square linear system. That is why we start with this topic first. It seems to be the same problem as solving an interval system because we need to enclose the coefficients of the real system with intervals to prevent rounding errors anyway. This is basically true. However, the radii of intervals are extremely small and hence specific methods can be used that return tight enclosures and are fast. We chose to present an efficient method introduced by Rump in his dissertation thesis [190]. In English it is described, e.g., in [191, 196]. Here we use the version described in [196].

Let us have a square real system $Ax = b$ with a nonsingular A. Our goal is to compute a tight verified enclosure of $x = A^{-1}b$. Many methods introduced later start with some initial enclosure of the solution and try to contract it or shave it. This method follows a rather opposite approach. It starts with some approximation of the solution. Let us say

$$\widetilde{x} = Cb,$$

where $C \approx A^{-1}$. The initial degenerated enclosure $\boldsymbol{x}^{(0)} = [\widetilde{x}, \widetilde{x}]$ is then being inflated until a certain condition is met

$$\begin{aligned} \boldsymbol{y} &:= \boldsymbol{x}^{(k)} \cdot [0.9, 1.1] + [-\varepsilon, \varepsilon], \\ \boldsymbol{x}^{(k+1)} &:= Cb + (I - CA)\boldsymbol{y}. \end{aligned}$$

Relative inflation by the interval $[0.9, 1.1]$ and absolute inflation by the interval $[-\varepsilon, \varepsilon]$ for some small ε (Rump chooses $\varepsilon = 10^{-20}$) are used. Both of the intervals are set empirically. If at some iteration point

$$\boldsymbol{x}^{(k+1)} \subseteq \text{interior}(\boldsymbol{y})$$

holds, then according to the fix point theorem [189] we know that A and C are regular and

$$A^{-1}b \in \boldsymbol{x}^{(k+1)}.$$

If $\varrho(I - CA) < 1$, then the algorithm converges [196]. The algorithm works also for interval matrices (one can just replace A, b with $\boldsymbol{A}, \boldsymbol{b}$ respectively). The algorithm works also for multiple right-hand sides at once, hence it can be used to compute a verified inverse of a real matrix.

5.6 Direct computation

As we implied in the introduction, in this work we distinguish between methods with direct computation and methods with iterative computation. We start with direct methods first. The number of steps that a direct method executes can be counted in advance and for every size of input they basically provide the same number of steps.

5.6.1 Gaussian elimination

Interval version of Gaussian elimination has already been described many times, see [3, 58, 139]. It works similarly to real Gaussian elimination. Only some minor changes are needed. First, for elimination into row echelon form we use interval arithmetics instead of the real one. Second, if we view the elimination process as simultaneous elimination on all real systems contained in an interval system, then we can put an interval $[0, 0]$ instead of each eliminated element under a pivot.

Example 5.8. Notice the elimination of the element under the pivot by subtraction of the first row from the second one.

$$\begin{pmatrix} [1,2] & [1,2] \\ [1,2] & [3,3] \end{pmatrix} \sim \begin{pmatrix} [1,2] & [1,2] \\ [0,0] & [1,2] \end{pmatrix}.$$

Solving of an interval system consists of two phases – elimination and backward substitution (described as Algorithm 5.9 and 5.10 respectively).

The elimination assumes that pivot intervals do not contain zero. Sometimes a matrix can be rearranged in such a form. However, as shown in [3], such a rearrangement might not exist even if $\boldsymbol{A}$ is regular. Nevertheless, the elimination can be carried out without row interchange for H-matrices and tridiagonal matrices [3].

Algorithm 5.9 (Elimination phase)**.** The algorithm takes an interval matrix $\boldsymbol{A}$ and an interval vector $\boldsymbol{b}$ of the corresponding size and eliminates the matrix $(\boldsymbol{A} \mid \boldsymbol{b})$ into row echelon form.

1. For rows $i = 1, \ldots, (n-1)$ do the following steps.
2. For $j \in \{i, \ldots, n\}$ a find row with $0 \notin \boldsymbol{a}_{ji}$.
3. If such row cannot be found, notify that $\boldsymbol{A}$ is possibly singular.
4. For every row $j > i$ set

$$\begin{aligned} \boldsymbol{a}_{ji} &:= [0, 0], \\ \boldsymbol{a}_{(j,i+1:n)} &:= \boldsymbol{a}_{(j,i+1:n)} - \frac{\boldsymbol{a}_{ji}}{\boldsymbol{a}_{ii}} \cdot \boldsymbol{a}_{(i,i+1:n)}, \\ \boldsymbol{b}_j &:= \boldsymbol{b}_j - \frac{\boldsymbol{a}_{ji}}{\boldsymbol{a}_{ii}} \cdot \boldsymbol{b}_i. \end{aligned}$$

The step 2. can be combined with some kind of pivoting. As in [61] we select a pivot with the smallest mignitude (*mignitude pivoting*).

Algorithm 5.10 (Backward substitution)**.** The algorithm takes $(\boldsymbol{A} \mid \boldsymbol{b})$ in the row echelon form and computes enclosures of all variables by systematic substitution from below.

1. For each row $i = n, \ldots, 1$ do the following steps.
2. Compute enclosure of $\boldsymbol{x}_i$ as

$$\boldsymbol{x}_i = \frac{1}{\boldsymbol{a}_{ii}} \left(\boldsymbol{b}_i - \sum_{j=i+1}^{n} \boldsymbol{a}_{ij} \boldsymbol{x}_i \right).$$

Gaussian elimination without preconditioning may work when intervals are small. However Gaussian elimination generally suffers from multiple use of interval coefficients during elimination. The widths of resulting interval enclosures tend to grow exponentially. For more information on such a phenomenon see, e.g., [133, 196].

Example 5.11. Let A_c be the 10×10 Toeplitz matrix with the first row equal to $(1, 2, 3, 4, \ldots, 9, 10)^T$. Let b_c be $(1, 1, \ldots, 1)^T$. The radii of all intervals are set to 10^{-6}. Widths of variable enclosures returned by Gaussian elimination without preconditioning are shown in Table 5.1. In each next coefficient the width of enclosure widens "roughly" by 3.

Fortunately, for special classes of matrices this algorithm works (at least theoretically) well. It can be performed without preconditioning on H-matrices with certain overestimation and it returns the hull for M-matrices and $\boldsymbol{b} > 0$ or $0 \in \boldsymbol{b}$ or $\boldsymbol{b} > 0$ [139]. For other classes of matrices use of preconditioning might be needed. Gaussian elimination with preconditioning can be proved to work better than the later introduced Jacobi and Gauss–Seidel method [139]. Gaussian elimination was also a subject to various improvements, e.g., [42, 49].

Table 5.1: Overestimation of variable enclosures by Gaussian elimination and backward substitution without preconditioning from Example 5.11. The first column indicates a variable; the width of each variable enclosure is $(\alpha \cdot 10^e)$.

variable	α	10^e
x_{10}	1.08	10^{-2}
x_9	2.62	10^{-2}
x_8	8.69	10^{-2}
x_7	2.97	10^{-1}
x_6	1.01	10^0
x_5	3.46	10^0
x_4	1.18	10^1
x_3	4.03	10^1
x_2	1.38	10^2
x_1	1.67	10^2

5.6.2 The Hansen–Bliek–Rohn–Ning–Kearfott–Neumaier method

This method was first developed by Hansen in [57] and also independently by Bliek in his dissertation thesis. The stronger results were reformulated by Rohn in [167] using only one matrix inverse. In [143], Ning and Kearfott generalized the method for H-matrices. A simpler proof was given by Neumaier in [140]. The following version of the theorem is from [143]. For simplicity we refer to this method as the HBR method.

Theorem 5.12 (HBR). *Let $\boldsymbol{A}x = \boldsymbol{b}$ a square interval system, with $\boldsymbol{A}$ being an H-matrix of order n,*

$$u = \langle \boldsymbol{A} \rangle^{-1} \operatorname{mag}(\boldsymbol{b}), \quad d_i = (\langle \boldsymbol{A} \rangle^{-1})_{ii},$$

and

$$\alpha_i = \langle \boldsymbol{A} \rangle_{ii} - 1/d_i, \quad \beta_i = u_i/d_i - \operatorname{mag}(\boldsymbol{b}_i),$$

for $i = 1, \ldots, n$. Then Σ is contained in $\boldsymbol{x}$ with components

$$\boldsymbol{x}_i = \frac{\boldsymbol{b}_i + [-\beta_i, \beta_i]}{\boldsymbol{A}_{ii} + [-\alpha_i, \alpha_i]},$$

for $i = 1, \ldots, n$.

This method has a nice feature; when A_c is a diagonal matrix, then the returned $\boldsymbol{x}$ is the hull, the proof can be found found in [140, 143]. In this theorem only one computation of a verified matrix inverse is needed. The verified bounds on $\langle \boldsymbol{A} \rangle^{-1}$ can

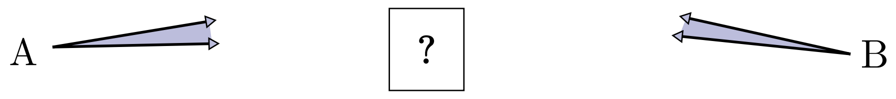

Figure 5.3: Two colliding projectiles and the area of interest.

be computed using the ε-inflation method. Another approach for finding the upper bound on $\langle \boldsymbol{A} \rangle^{-1}$ can be found in [140].

5.7 Iterative computation

In iterative computation we usually start with an initial enclosure $\boldsymbol{x}^{(0)}$ containing the solution set. Nevertheless, some methods do not require that. If they are given a box containing only a part of the solution, they compute an enclosure of this part, if they are given a box with no solution, they can usually recognize that. Such methods generate a sequence of enclosures

$$\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}, \boldsymbol{x}^{(k+1)}.$$

Such a sequence is often nested

$$\boldsymbol{x}^{(0)} \supseteq \boldsymbol{x}^{(1)} \supseteq \cdots \supseteq \boldsymbol{x}^{(k)} \supseteq \boldsymbol{x}^{(k+1)}.$$

Regarding the sequence, three issues need to be addressed — how to determine $\boldsymbol{x}^{(0)}$, how to derive $\boldsymbol{x}^{(k+1)}$ from $\boldsymbol{x}^{(k)}$ and how to stop the iteration. We start with the first and the third issue. The second one depends on a particular method – we later introduce approaches by Krawczyk's method, the Jacobi method and the Gauss–Seidel method.

5.7.1 Initial enclosure

All the next methods rely on some existing enclosure $\boldsymbol{x}^{(0)}$ of the solution set. There are some ways how to determine an initial enclosure. First, we might guess it from the nature of a problem to be solved.

Example 5.13. Let us have two projectiles A, B as depicted in Figure 5.3. Both projectiles move with a known constant velocity in directions having some added uncertainty. We might be interested in whether it is possible that the two projectiles collide in the marked area. For example, when we know that there is a city in this area, we are extremely interested in solutions only in this area.

Second, we can compute an initial enclosure using some direct method and try improve it iteratively. This approach will be later used for overdetermined interval linear systems.

And third, we can use an explicit formula giving us an initial enclosure. For example the next proposition comes from [139].

Proposition 5.14. *Let $\boldsymbol{A}x = \boldsymbol{b}$ be a square interval system and C be a square real matrix of the corresponding size. If it holds that*

$$\langle C\boldsymbol{A}\rangle u \geq v > 0, \quad \textit{for some } u \geq 0,$$

then

$$\Sigma \subseteq \|C\boldsymbol{b}\|_v[-u, u].$$

A good candidate for such a vector u may be an approximate solution of the system $\langle C\boldsymbol{A}\rangle u = e$ and v can be set to $v := \langle C\boldsymbol{A}\rangle u$. The argumentation behind this can be also found in [139]. As usual, we use $C \approx A_c^{-1}$.

Sometimes much computationally easier initial enclosure can be obtained by using just maximum norm [133], when we set $\boldsymbol{A}' = I - C\boldsymbol{A}$ and $\|\boldsymbol{A}'\| < 1$ ($C\boldsymbol{A}$ is an H-matrix), then

$$\Sigma \subseteq \frac{\|C\boldsymbol{b}\|}{1 - \|\boldsymbol{A}'\|}[-e, e]. \tag{5.4}$$

5.7.2 Stopping criteria

The stopping criterion reflects similarity of two consequent enclosures in a nested sequence. Unless stated otherwise stopping criterion will be a combination of maximum number of steps (usually 20) and the following condition which takes into account the difference of lower and upper bounds separately. For two subsequent enclosures $\boldsymbol{x}^{(k)}, \boldsymbol{x}^{(k+1)}$ we stop when

$$|\underline{x}^{(k)} - \underline{x}^{(k+1)}| < \varepsilon \text{ and } |\overline{x}^{(k)} - \overline{x}^{(k+1)}| < \varepsilon,$$

where ε is a vector with all coefficients equal to some small positive number. It can be heuristically preset with respect to widths of intervals,

$$\varepsilon \approx \min_{ij}(\text{wid}(\boldsymbol{A}_{ij})) \times 10^{-5}. \tag{5.5}$$

5.7.3 Krawczyk's method

The method is described in e.g., [133, 139]. For a given interval linear system $\boldsymbol{A}x = \boldsymbol{b}$ let us suppose there is an initial enclosure $\boldsymbol{x}^{(0)} \supseteq \Sigma$. For every $x = A^{-1}b$, where $A \in \boldsymbol{A}, b \in \boldsymbol{b}$ and C being a suitable real matrix it holds that

$$x = A^{-1}b = Cb - (CA - I)A^{-1}b \in C\boldsymbol{b} - (C\boldsymbol{A} - I)\boldsymbol{x}^{(0)}.$$

Hence, the iteration is

$$\begin{aligned}
\boldsymbol{y}^{(i+1)} &:= C\boldsymbol{b} - (C\boldsymbol{A} - I)\boldsymbol{x}^{(i)}, \\
\boldsymbol{x}^{(i+1)} &:= \boldsymbol{y}^{(i+1)} \cap \boldsymbol{x}^{(i)}.
\end{aligned}$$

Due to the intersection the algorithm creates a sequence of nested interval vectors. Another point of view on Krawczyk's method is that it is a restriction of a more general method for nonlinear systems to linear systems only [110]. This method is very simple and better enclosures can be obtained by other methods. An advantage is that when a preconditioner C is available there are no divisions in the algorithm (unlike in the further introduced Jacobi and Gauss–Seidel method).

5.7.4 The Jacobi and Gauss–Seidel method

In this subsection we discuss the well-known iterative algorithms – the Jacobi method and the Gauss–Seidel method. Both methods need some initial enclosure $\boldsymbol{x}^{(0)}$. First, let us start with the Jacobi method. The ith equation of a real system $Ax = b$ is the following

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ii}x_i + \cdots + a_{in}x_n = b_i.$$

The variable x_i can be expressed as

$$x_i = \frac{1}{a_{ii}}\Big[b_i - (a_{i1}x_1 + \cdots + a_{i(i-1)}x_{i-1} + a_{i(i+1)}x_{i+1} + \cdots + a_{in}x_n)\Big].$$

When we have an interval system $\boldsymbol{A}x = \boldsymbol{b}$ and a known enclosure $\boldsymbol{x}$ we get

$$\boldsymbol{x}_i \subseteq \frac{1}{\boldsymbol{a}_{ii}}\Big[\boldsymbol{b}_i - (\boldsymbol{a}_{i1}\boldsymbol{x}_1 + \cdots + \boldsymbol{a}_{i(i-1)}\boldsymbol{x}_{i-1} + \boldsymbol{a}_{i(i+1)}\boldsymbol{x}_{i+1} + \cdots + \boldsymbol{a}_{in}\boldsymbol{x}_n)\Big].$$

This formula gives rise to iterative Algorithm 5.15.

Algorithm 5.15 (Jacobi method)**.** Input is a square system $\boldsymbol{A}x = \boldsymbol{b}$ and some initial enclosure of the solution set $\boldsymbol{x}^{(0)}$. It returns an enclosure $\boldsymbol{x}$ of a solution set.

1. For each variable $\boldsymbol{x}_i$ compute a new enclosure as

$$\boldsymbol{y}_i^{(k+1)} = \frac{1}{\boldsymbol{a}_{ii}}\left(\boldsymbol{b}_i - \sum_{j\neq i}\boldsymbol{a}_{ij}\boldsymbol{x}_j^{(k)}\right) \quad \text{for } i = (1,\ldots,n). \tag{5.6}$$

2. Intersect with the old enclosure

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k \cap \boldsymbol{y}^{k+1}.$$

3. Repeat steps 1. to 2. until stopping criteria are not met.

From the prescription of iterative improvement it is clear that it can be computed in parallel for each variable. It is possible to rewrite (5.6) in a form that helps mathematical software with optimized matrix multiplication:

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{D}^{-1}(\boldsymbol{b} - \boldsymbol{J}\boldsymbol{x}^k), \tag{5.7}$$

where $\boldsymbol{D}$ is the main diagonal of $\boldsymbol{A}$ and $\boldsymbol{J}$ is $\boldsymbol{A}$ with the intervals on the main diagonal set to $[0,0]$.

Example 5.16. In this example we show the differences between the two versions of the Jacobi iteration (5.6) and (5.7). They were compared on random interval linear systems. The midpoint coefficients of a system were taken independently uniformly from interval $[-10, 10]$ and then wrapped with intervals with radii 10^{-3}. The simple initial enclosure (5.4) was used and we set $\epsilon = 10^{-6}$. We tested using the DESKTOP setting. Both methods returned identical enclosures and did identical number of iterations, only the computation times differed. For each size we took the average computation time on 100 systems. The average computation times in seconds are displayed in Table 5.2. The average times include preconditioning. The matrix version of the Jacobi method is clearly faster.

Table 5.2: The two implementations of the Jacobi method (5.6) and (5.7) – average computation times (in seconds), n is the number of variables of a system.

n	Jacobi (5.6)	matrix Jacobi (5.7)
10	0.26	0.07
20	0.52	0.07
30	0.92	0.08
40	1.31	0.10
50	1.72	0.11
60	2.25	0.14
70	2.67	0.17
80	3.21	0.20
90	3.76	0.25
100	4.37	0.30

The previous results need to be taken with caution, because they may be partly system/software dependent. However, when optimized matrix multiplication is accessible, we recommend to use the (5.7) version of the Jacobi algorithm.

The Gauss–Seidel method is an improvement of the Jacobi method. The only difference is that it immediately uses the newly computed enclosures of variables.

Algorithm 5.17 (Gauss–Seidel method)**.** Substitute the formula (5.6) in step 2. of the Jacobi method by

$$\boldsymbol{y}_i^{(k+1)} = \frac{1}{\boldsymbol{a}_{ii}}\left(\boldsymbol{b}_i - \sum_{j<i} \boldsymbol{a}_{ij}\boldsymbol{x}_j^{(k+1)} - \sum_{j>i} \boldsymbol{a}_{ij}\boldsymbol{x}_j^{(k)}\right) \quad \text{for } i = (1, \ldots, n).$$

The advantage of the Gauss–Seidel method is its faster convergence, the drawback is that it cannot be parallelized. Since the the number of operations per one iteration is the same as for the Jacobi method (5.6), when comparing the Gauss–Seidel and the parallel Jacobi we could expect the similar results regarding computation time as in Example 5.16. During experiments testing the fewer iterations did compensate for much larger time needed per one iteration.

Both methods assume there are no intervals containing 0 on the main diagonal of $\boldsymbol{A}$. If this is not the case, then extended interval arithmetic can be used [133]. This does not happen for Krawczyk's method.

It is proved that when used with a preconditioner C, the Gauss–Seidel method never yields worse bounds than any method based on matrix splitting of $C\boldsymbol{A}$ (i.e., the Jacobi, Krawczyk's, etc.) [139]. However, the Jacobi and the Gauss–Seidel practically converge to the same enclosures.

When applied to a system with an M-matrix, both methods can be used without preconditioning and return the hull. Similarly to Krawczyk's method, for both methods, if some $\emptyset \neq \boldsymbol{y}^{(k+1)} \subseteq$ interior $\boldsymbol{x}^{(0)}$ (the initial enclosure is strictly improved during iterations) then it proves that $\boldsymbol{A}$ is an H-matrix [139].

5.8 Small comparison of methods

In upcoming problems we are going to exploit existence of methods for solving square interval linear systems. That is why at the end of this chapter we provide a small comparison of the mentioned methods. We are going to compare:

- `ge` – Gaussian elimination with mignitude pivoting,
- `jacobi` – matrix version of the Jacobi method (5.7) with maximum number of iterations set to 20 and ε chosen according to (5.5); for initial enclosure we use the formula (5.4),
- `krawczyk` – Krawczyk's method with the same setting as `Jacobi`,
- `hbr` – the Hansen–Bliek–Rohn method; the enclosure on $\langle A\rangle^{-1}$ is computed using the inflation method.

The suffix `+pre` means that the method is used with preconditioning by midpoint inverse.

In [143] they compare several methods for computing enclosures of interval linear systems; mostly variants of the HBR-method and Gaussian elimination. We borrow their three examples and demonstrate also properties of other methods.

Example 5.18. Let us have the system $\boldsymbol{A}x = \boldsymbol{b}$, where

$$\boldsymbol{A} = \begin{pmatrix} [4,6] & [-1,1] & [-1,1] & [-1,1] \\ [-1,1] & [-6,-4] & [-1,1] & [-1,1] \\ [-1,1] & [-1,1] & [9,11] & [-1,1] \\ [-1,1] & [-1,1] & [-1,1] & [-11,-9] \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} [-2,4] \\ [1,8] \\ [-4,10] \\ [2,12] \end{pmatrix}.$$

Note that $\boldsymbol{A}$ is a strictly diagonally dominant, hence Gaussian elimination can be used without preconditioning [139]. The resulting enclosure is

$$\boldsymbol{x}^{\texttt{ge}} = \begin{pmatrix} [-2.60, 3.10] \\ [-3.90, 1.50] \\ [-1.43, 2.15] \\ [-2.35, 0.60] \end{pmatrix}.$$

When using the Jacobi method we obtain

$$\boldsymbol{x}^{\texttt{jacobi}} = \begin{pmatrix} [-2.60, 3.10] \\ [-3.90, 1.65] \\ [-1.48, 2.15] \\ [-2.35, 0.79] \end{pmatrix}.$$

Notice that in each interval coefficient at least one bound is exactly the same as in $\boldsymbol{x}^{\mathtt{ge}}$. The HBR method returns the narrowest enclosure

$$\boldsymbol{x}^{\mathtt{hbr}} = \begin{pmatrix} [-2.50, 3.10] \\ [-3.90, 1.20] \\ [-1.40, 2.15] \\ [-2.35, 0.60] \end{pmatrix},$$

and since the midpoint matrix A_c of $\boldsymbol{A}$ is diagonal, it is the interval hull.

When $\boldsymbol{A}$ is an H-matrix and A_c is diagonal then it can be proved that $\boldsymbol{x}^{\mathtt{ge}} \subseteq \boldsymbol{x}^{\mathtt{jac}}$ and in each interval coefficient has at least one bound the same [139]. Let us use another example from [143], in which $\boldsymbol{A}$ is an M-matrix.

Example 5.19. Let us have a system $\boldsymbol{A}x = \boldsymbol{b}$, where

$$\boldsymbol{A} = \begin{pmatrix} [3.7, 4.3] & [-1.5, -0.5] & [0, 0] \\ [-1.5, -0.5] & [3.7, 4.3] & [-1.5, 0.5] \\ [0, 0] & [-1.5, -0.5] & [3.7, 4.3] \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} [-14, 14] \\ [-9, 9] \\ [-3, 3] \end{pmatrix}.$$

Using the previously tested four methods we get

$$\boldsymbol{x}^{\mathtt{ge}} = \boldsymbol{x}^{\mathtt{jacobi}} = \boldsymbol{x}^{\mathtt{hbr}} = \begin{pmatrix} [-6.38, 6.38] \\ [-6.40, 6.40] \\ [-3.40, 3.40] \end{pmatrix},$$

which is the interval hull.

In the next example Gaussian elimination gives better bounds.

Example 5.20. Using the previous example with a different right-hand side

$$\boldsymbol{b} = \begin{pmatrix} [-14, 0] \\ [-9, 0] \\ [-3, 0] \end{pmatrix}.$$

The tightest enclosure is returned by Gaussian elimination and the Jacobi method without preconditioning

$$\boldsymbol{x}^{\mathtt{ge}} = \boldsymbol{x}^{\mathtt{jacobi}} \begin{pmatrix} [-6.38, 0] \\ [-6.40, 0] \\ [-3.40, 0] \end{pmatrix}.$$

The enclosure returned by Gaussian elimination with preconditioning gives

$$\boldsymbol{x}^{\mathtt{ge+pre}} = \begin{pmatrix} [-6.38, 1.35] \\ [-6.40, 1.74] \\ [-3.40, 1.40] \end{pmatrix}.$$

Table 5.3: Square interval linear systems – comparison of enclosures. The reference method is `hbr`, the uniform radii was set to $r = 0.001$, n is the number of variables of a system.

n	`ge+pre`	`jacobi+pre`	`krawczyk+pre`
10	1.00083	1.00012	1.00187
20	1.00091	1.00005	1.00139
30	1.00122	1.00021	1.00222
40	1.00081	1.00025	1.00207
50	1.00091	1.00024	1.00200
60	1.00065	1.00021	1.00192
70	1.00058	1.00031	1.00232
80	1.00085	1.00032	1.00231
90	1.00086	1.00039	1.00238
100	1.00119	1.00038	1.00240

The HBR method gives wider bounds

$$\boldsymbol{x}^{\texttt{hbr}} = \begin{pmatrix} [-6.38, 1.12] \\ [-6.40, 1.54] \\ [-3.40, 1.40] \end{pmatrix}.$$

Sometimes the HBR method gives better bounds, sometimes Gaussian elimination does. In some cases both methods return the same bounds and in some cases the intersection of their results gives even shaper bounds [143].

Now let us test more thoroughly for larger random systems. The systems are generated as in Example 5.16. We test on 100 systems for each size (number of variables n). The reference method is `hbr`; other methods are compared to it using 3.9. The ratios of enclosures for radii $r = 0.001$ are in Table 5.3, computation times in Table 5.4 and number of finite enclosures returned in Table 5.5. Clearly `hbr` is the winner from both computational time and enclosure tightness perspective. The similar results for radii $r = 0.01$ are displayed in Tables 5.6, 5.7 and 5.8.

It can be seen, that the methods return similar results. It happens because all methods actually use preconditioning (explicitly or implicitly) after which the resulting matrix in system is an H-matrix. For slightly larger radii the ratios are still similar, however, some methods fail to produce finite enclosures for larger systems. This happens mostly because of failure of the initial enclosure (5.4).

5.9 Shaving method

Most of the previously mentioned methods use preconditioning. We saw that preconditioning can inflate the original solution set and even though we can get close to the