or limits of our software tools. Here we present two more sufficient conditions for checking regularity and four sufficient conditions for checking singularity.

Theorem 11.12 (Sufficient conditions for regularity)**.** *An interval matrix* $\boldsymbol{A}$ *is regular if at least one of the following conditions holds:*

1. $\lambda_{\max}(A_\Delta^T A_\Delta) < \lambda_{\min}(A_c^T A_c)$ [193] ,
2. $A_c^T A_c - \|A_\Delta^T A_\Delta\| I$ *is positive definite for some consistent matrix norm* $\|\cdot\|$ [164].

Theorem 11.13 (Sufficient conditions for singularity)**.** *An interval matrix* $\boldsymbol{A}$ *is singular if at least one of the following conditions holds:*

1. $\lambda_{\max}(A_c^T A_c) \leq \lambda_{\min}(A_\Delta^T A_\Delta)$ [164],
2. $\max_j(|A_c^{-1}|A_\Delta)_{jj} \geq 1$ [166],
3. $(A_\Delta - |A_c|)^{-1} \geq 0$ [173],
4. $A_\Delta^T A_\Delta - A_c^T A_c$ *is positive semidefinite* [164].

In Section 4.1 we have already met some classes of interval matrices that are regular (strictly diagonally dominant matrices, M-matrices and H-matrices). Checking that a matrix belongs to these classes can be done in strongly polynomial time.

11.3.1 Summary

Problem	**Complexity**
Is $\boldsymbol{A}$ regular?	coNP-complete
Is $\boldsymbol{A}$ singular?	NP-complete

11.4 Full column rank

Checking full column rank was addressed in Section 7.3. Deciding whether an interval matrix has full column rank is connected to checking regularity. If an interval matrix $\boldsymbol{A}$ of size $m \times n$, $m \geq n$, contains a regular interval sub-matrix of size n, then obviously $\boldsymbol{A}$ has full column rank. What is surprising is that the implication does not

hold conversely (in contrast to real matrices). The following interval matrix by Irene Sharaya (see [204]) serves as a counterexample.

Example 11.14. The matrix

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

has full column rank, but contains no regular submatrix of size 2.

For square matrices, checking regularity can can be polynomially reduced to checking full column rank (we just check whether $\boldsymbol{A}$ has full column rank). Therefore, checking full column rank is coNP-hard. Finding a polynomial certificate for an interval matrix not having full column rank can be done by orthant decomposition similarly as in the case of singularity. That is why, checking full column rank is coNP-complete.

Again, fortunately, we have some sufficient conditions that are computable in polynomial time. In Section 7.3 we mentioned several polynomially checkable conditions for an interval matrix having full column rank.

11.4.1 Summary

Problem	Complexity
Does $\boldsymbol{A}$ have full column rank?	coNP-complete

11.5 Solving a system of linear equations

Solving of interval linear systems was the main topic of Chapter 5 and 6. We have the following theorem by Rohn [171].

Theorem 11.15. *Computing an enclosure of the solution set of $\boldsymbol{A}x = \boldsymbol{b}$ when it is bounded, and otherwise returning an error message, is* NP*-hard.*

If such an algorithm existed, it could be used to decide regularity of an interval matrix, since regularity of $\boldsymbol{A}$ implies bounded solution set of $\boldsymbol{A}x = \boldsymbol{b}$ for arbitrary $\boldsymbol{b}$ [171].

Computing the optimal bounds (the hull) on the solution set is also NP-hard [184]. The problem stays NP-hard even if we limit widths of intervals of the system matrix with some $\delta > 0$, or allow the bounds to consist of 0 or 1 only [112]. Unfortunately, even computing various ε-approximations of the hull components is an NP-hard problem [112].

Theorem 11.16. *Let $\varepsilon > 0$, then computing the relative and absolute ε-approximation of the hull (its components) of $\boldsymbol{A}x = \boldsymbol{b}$ are both* NP*-hard problems.*

Fortunately, in Chapter 5 we saw various methods and special conditions on $\boldsymbol{A}, \boldsymbol{b}$ under which the hull can be computed in polynomial time.

11.5.1 Overdetermined systems

In Chapter 6 we defined overdetermined systems. The problem of computing the interval hull of Σ^{lsq} is NP-hard, since when $\boldsymbol{A}$ is square and regular, then $\Sigma^{lsq} = \Sigma$.

11.5.2 Restricted interval coefficients

We can try to identify some classes of systems with exact hull computation algorithms that run in polynomial time. If we restrict the right hand side $\boldsymbol{b}$ to contain only degenerate intervals, we have $\boldsymbol{A}x = b$. Such a problem is still NP-hard [112]. If we, however, restrict the matrix to be consisting only of degenerate intervals A and we have a system $Ax = \boldsymbol{b}$, then, computing the exact bounds of the solution set is polynomial, since it can be rewritten as a pair of linear programs

$$\max(\min)\ e_i^T x, \quad Ax \geq \underline{b}, \quad Ax \leq \overline{b},$$

for each variable x_i.

11.5.3 Structured systems

We can also explore band and sparse matrices.

Definition 11.17. A matrix $\boldsymbol{A}$ is a *w-band* matrix if $\boldsymbol{a}_{ij} = 0$ for $|i - j| \geq w$.

Band matrices with $d = 1$ are diagonal and computing the hull is clearly strongly polynomial. For $d = 2$ (tridiagonal matrix) it is an open problem. And for $d \geq 3$ it is already NP-hard [111]. We can also get strong polynomial time in case of bidiagonal systems.

Proposition 11.18 (Horáček et al., [85])**.** *For a bidiagonal matrix (the matrix with only the main diagonal and an arbitrary neighboring diagonal) computing the exact hull of $\boldsymbol{A}x = \boldsymbol{b}$ is strongly polynomial.*

Proof. Without the loss of generality let us suppose that a matrix $\boldsymbol{A}$ consists of the main diagonal and the one below it. By the forward substitution, we have $\boldsymbol{x}_1 = \frac{\boldsymbol{b}_1}{\boldsymbol{a}_{11}}$ and

$$\boldsymbol{x}_i = \frac{\boldsymbol{b}_i - \boldsymbol{a}_{i,i-1}\boldsymbol{x}_{i-1}}{\boldsymbol{a}_{ii}}, \quad i = 2, \ldots, n.$$

By induction, $\boldsymbol{x}_{i-1}$ is optimally computed with no use of interval coefficients of the ith equation. Since an evaluation in interval arithmetic is optimal when there are no multiple occurrences of variables (Theorem 3.13), $\boldsymbol{x}_i$ is optimal as well. □

Definition 11.19. A matrix $\boldsymbol{A}$ is *d-sparse* if in each row i at most d elements $\boldsymbol{a}_{ij} \neq 0$.

For sparse matrices with $d = 1$ computing the hull is clearly strongly polynomial. For $d \geq 2$ it is again NP-hard [112]. Nevertheless, if we combine w-band matrix with system coefficient bounds coming from a given finite set of rational numbers, then we have a polynomial algorithm for computing the hull [112].

11.5.4 Parametric systems

A natural generalization of an interval linear system is by incorporating linear dependencies of coefficients. That is, we have a family of linear systems

$$A(p)x = b(p), \quad p \in \boldsymbol{p}, \tag{11.5}$$

where $A(p) = \sum_{k=1}^{K} A^k p_k$, $b(p) = \sum_{k=1}^{K} b^k p_k$ and K is number of parameters. Here, p is a vector of parameters varying in $\boldsymbol{p}$. Since this concept generalizes the standard interval systems, many related problems are intractable [206]. The reason is that an interval system $\boldsymbol{A}x = \boldsymbol{b}$ can be considered as a parametric system $A(\boldsymbol{p})x = b(\boldsymbol{p})$ with $n^2 + n$ interval parameters.

Nevertheless, we point out one particular efficiently solvable problem. Given $x \in \mathbb{R}^n$, deciding whether it is a solution of a standard interval system $\boldsymbol{A}x = \boldsymbol{b}$ is strongly polynomial. For systems with linear dependencies, the problem still stays polynomial (just by checking if x satisfies the Oettli–Prager theorem), but we can show weak polynomiality only; this is achieved by rewriting (11.5) as a linear program. For more information on parametric systems see, e.g., [67, 151, 206].

11.5.5 Summary

Problem	Complexity
Is x a solution of $\boldsymbol{A}x = \boldsymbol{b}$?	strongly P
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$	NP-hard
Computing the hull of $\boldsymbol{A}x = b$	NP-hard
Computing the hull of $Ax = \boldsymbol{b}$	P
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is regular	NP-hard
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is an M-matrix	strongly P
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is diagonal	strongly P
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is bidiagonal	strongly P
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is tridiagonal	?
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is 3-band	NP-hard
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is 1-sparse	strongly P
Computing the hull of $\boldsymbol{A}x = \boldsymbol{b}$, where $\boldsymbol{A}$ is 2-sparse	NP-hard
Computing the exact least squares hull of $\boldsymbol{A}x = \boldsymbol{b}$	NP-hard
Is Σ bounded?	coNP-complete
Computing the hull of $A(\boldsymbol{p})x = b(\boldsymbol{p})$	NP-hard
Is x a solution of $A(\boldsymbol{p})x = b(\boldsymbol{p})$?	P

11.6 Matrix inverse

Interval inverse matrix was defined in Section 4.2. For a square interval matrix $\boldsymbol{A}$ it can be computed using knowledge of inverses of 2^{2n-1} matrices in the form

$$A_{yz} = A_c - D_y A_\Delta D_z,$$

where y, z are n-dimensional vectors from Y_n; [169].

Theorem 11.20. *Let $\boldsymbol{A}$ be regular. Then its inverse $\boldsymbol{A}^{-1} = [\underline{B}, \overline{B}]$ is described by*

$$\underline{B}_{ij} = \min_{y,\, z \in Y_n} \{(A_{yz}^{-1})_{ij}\},$$

$$\overline{B}_{ij} = \max_{y,\, z \in Y_n} \{(A_{yz}^{-1})_{ij}\},$$

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

Since ith column of the matrix inverse of $\boldsymbol{A}$ is equivalently computed as the hull of $\boldsymbol{A}x = e_i$, the problem is NP-hard (for another reasoning see [30]).

However, when $A_c = I$, we can compute the exact inverse in polynomial time according to the next theorem from [180].

Theorem 11.21. *Let $\boldsymbol{A}$ be a regular interval matrix with $A_c = I$. Let $M = (I - A_\Delta)^{-1}$. Then its inverse $\boldsymbol{A}^{-1} = [\underline{B}, \overline{B}]$ is described by*

$$\begin{aligned} \underline{B} &= -M + D_v, \\ \overline{B} &= M, \end{aligned}$$

where $v_j = \frac{2m_{jj}^2}{2m_{jj}-1}$ for $j = 1, \ldots, n$, with m_{jj} being the main diagonal elements of M.

When an interval matrix is of uniform width, i.e., $\boldsymbol{A} = [A_c - \alpha E, A_c + \alpha E]$, for a sufficiently small $\alpha > 0$ the inverse can be also expressed explicitly [183].

If we wish to only compute an enclosure $\boldsymbol{B}$ of the matrix inverse we can use any method for computing enclosures of interval linear systems. We get the ith column of $\boldsymbol{B}$ by solving the systems $\boldsymbol{A}x = e_i$.

Not all interval matrix classes imply intractability. In Section 4.2 we showed that checking inverse nonnegativity and also computing the exact interval inverse of an inverse nonnegative matrix are strongly polynomial tasks (see Theorem 4.13).

11.6.1 Summary

Problem	**Complexity**
Computing the exact inverse of $\boldsymbol{A}$	NP-hard
Computing the exact inverse of $\boldsymbol{A}$, $A_c = I$	strongly P
Computing the exact inverse of $\boldsymbol{A}$, $A_\Delta = \alpha E$, α suff. small	strongly P
Is $\boldsymbol{A}$ inverse nonnegative?	strongly P
Computing the exact inverse of inverse nonnegative $\boldsymbol{A}$	strongly P

11.7 Solvability of a linear system

In Chapter 7 we distinguished between weak and strong solvability. Checking whether an interval linear system is solvable is an NP-hard problem [112]. The sign coordinates of the orthant containing the solution can serve as a polynomial witness and existence of a solution can be verified by linear programming, hence this problem is NP-complete. Checking unsolvability as its complement is coNP-complete. The problem of deciding strong solvability is also coNP-complete. It can be reformulated as checking unsolvability of a certain linear system using the well known Farkas lemma, see [178].

Sometimes, we look only for a nonnegative solution (i.e., *nonnegative solvability*). Checking whether an interval linear system has a nonnegative solution is weakly polynomial. We know the orthant in which the solution should lie (the positive one). Therefore, we can get rid of the absolute values in the Oettli–Prager theorem and apply linear programming. However, checking whether a system is nonnengative strongly solvable is still coNP-complete [40]. We summarize the results in the following table.

Theorem 11.22. *Checking various types of solvability of $\boldsymbol{A}x = \boldsymbol{b}$ is of the following complexity:*

	weak	*strong*
solvability	NP-*complete*	coNP-*complete*
nonnegative solvability	P	coNP-*complete*

In Chapter 7 we introduced several methods for detecting solvability and unsolvability that work in polynomial time.

11.7.1 Linear inequalities

Just for comparison, considering systems of interval linear inequalities, the problems of checking various types of solvability become much easier. The results from [40] are summarized in the following table.

Theorem 11.23. *Checking various types of solvability of $\boldsymbol{A}x \leq \boldsymbol{b}$ is of the following complexity.*

	weak	*strong*
solvability	NP-*complete*	P
nonnegative solvability	P	P

We also would like to mention an interesting nontrivial property of strong solvability of systems of interval linear inequalities. When a system $\boldsymbol{A}x \leq \boldsymbol{b}$ is strongly solvable (i.e., every $Ax \leq b$ has a solution), then there exists a solution x satisfying $Ax \leq b$ for every $A \in \boldsymbol{A}$ and $b \in \boldsymbol{b}$ [40].

11.7.2 ∀∃-solutions.

Let us return to interval linear systems. The concept of a weak solution employs existential quantifiers: x is a solution if $\exists A \in \boldsymbol{A}$, $\exists b \in \boldsymbol{b} : Ax = b$. Nevertheless, in some applications, another quantification makes sense. In particular, $\forall\exists$ quantification was deeply studied [203]. For illustration of complexity of such solution, we will focus on two concepts of solutions – tolerance [40] and control solution [40, 200].

Definition 11.24. A vector x is a *tolerance* solution of $\boldsymbol{A}x = \boldsymbol{b}$ if

$$\forall A \in \boldsymbol{A}, \exists b \in \boldsymbol{b}, \quad Ax = b.$$

A vector x is a *control* solution of $\boldsymbol{A}x = \boldsymbol{b}$ if

$$\forall b \in \boldsymbol{b}, \exists A \in \boldsymbol{A}, \quad Ax = b.$$

Notice that a tolerance solution can equivalently be characterized as $\{Ax \mid A \in \boldsymbol{A}\} \subseteq \boldsymbol{b}$ and a control solution as $\boldsymbol{b} \subseteq \{Ax \mid A \in \boldsymbol{A}\}$.

Both solutions can be described by a slight modification of the Oettli–Prager theorem (one sign change in the Oettli–Prager formula) [40].

Theorem 11.25. *Let us have a system $\boldsymbol{A}x = \boldsymbol{b}$, then x is*

- *a tolerance solution if it satisfies $|A_c x - b_c| \leq -A_\Delta |x| + \delta$,*
- *a control solution if it satisfies $|A_c x - b_c| \leq A_\Delta |x| - \delta$.*

In case of tolerance solution this change makes checking whether a systems has this kind of solution decidable in weakly polynomial time. In the case of control solution the decision problem stays NP-complete [112].

11.7.3 Summary

Problem	Complexity
Is $\boldsymbol{A}x = \boldsymbol{b}$ solvable?	NP-complete
Is $\boldsymbol{A}x = \boldsymbol{b}$ strongly solvable?	coNP-complete
Is $\boldsymbol{A}x = \boldsymbol{b}$ nonnegative solvable?	P
Is $\boldsymbol{A}x = \boldsymbol{b}$ nonnegative strongly solvable?	coNP-complete
Is $\boldsymbol{A}x \leq \boldsymbol{b}$ solvable?	NP-complete
Is $\boldsymbol{A}x \leq \boldsymbol{b}$ strongly solvable?	P
Is $\boldsymbol{A}x \leq \boldsymbol{b}$ nonnegative solvable?	P
Is $\boldsymbol{A}x \leq \boldsymbol{b}$ nonnegative strongly solvable?	P
Is x a tolerance solution of $\boldsymbol{A}x = \boldsymbol{b}$?	strongly P
Is x a control solution of $\boldsymbol{A}x = \boldsymbol{b}$?	strongly P
Does $\boldsymbol{A}x = \boldsymbol{b}$ have a tolerance solution?	P
Does $\boldsymbol{A}x = \boldsymbol{b}$ have a control solution?	NP-complete

11.8 Determinant

Determinants of interval matrices were studied in Chapter 8. Several result about complexity of such a problem were stated there. Here we summarize the results in the following table.

11.8.1 Summary

Problem	Complexity
Computing $\underline{\det}(\boldsymbol{A})$	NP-hard
Computing $\overline{\det}(\boldsymbol{A})$	NP-hard
Computing relative ε-approximation of $\det(\boldsymbol{A})$ for $0 < \varepsilon < 1$	NP-hard
Computing absolute ε-approximation of $\det(\boldsymbol{A})$ for $0 < \varepsilon$	NP-hard
Computing $\overline{\det(\boldsymbol{A}^S)}$ for positive definite $\boldsymbol{A}^S$	P
Computing $\underline{\det}(\boldsymbol{A})$, where $\boldsymbol{A}$ is a tridiagonal H-matrix	strongly P
Computing $\overline{\det}(\boldsymbol{A})$, where $\boldsymbol{A}$ is a tridiagonal H-matrix	strongly P
Computing $\underline{\det}(\boldsymbol{A})$ for $\boldsymbol{A}$ regular with $A_c = I$	strongly P
Computing $\overline{\det}(\boldsymbol{A})$ for $\boldsymbol{A}$ regular with $A_c = I$	?

11.9 Eigenvalues

We briefly start with general matrices, then we continue with the symmetric case. Checking singularity of $\boldsymbol{A}$ can be polynomially reduced to checking whether 0 is an eigenvalue of some matrix $A \in \boldsymbol{A}$. Using the reasoning from Section 11.1.7, checking whether λ is an eigenvalue of some matrix $A \in \boldsymbol{A}$ is NP-hard. Surprisingly, checking an eigenvector is strongly polynomial [168].

How it is with the Perron theory? An interval matrix $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ is *nonnegative irreducible* if every $A \in \boldsymbol{A}$ is nonnegative irreducible (a definition can be found in [91]). For Perron vectors (positive vectors corresponding to the dominant eigenvalues), we have the following result [177].

Theorem 11.26. *Let $\boldsymbol{A}$ be nonnegative irreducible. Then the problem of deciding whether x is the Perron eigenvector of some matrix $A \in \boldsymbol{A}$ is strongly polynomial.*

For the sake of simplicity we mentioned only some results considering eigenvalues of a general matrix $\boldsymbol{A}$. We will go into more detail with symmetric matrices, which have real eigenvalues. In Chapter 8 we defined a symmetric interval matrix as a subset of all symmetric matrices in $\boldsymbol{A}$, that is,

$$\boldsymbol{A}^S := \{A \in \boldsymbol{A} \mid A = A^T\}.$$

For a symmetric $A \in \mathbb{R}^{n\times n}$, we denote its smallest and largest eigenvalue by $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ respectively. For a symmetric interval matrix $\boldsymbol{A}^S$, we define the smallest and largest eigenvalue as

$$\begin{aligned}\lambda_{\min}(\boldsymbol{A}^S) &:= \min\{\lambda_{\min}(A) \mid A \in \boldsymbol{A}^S\},\\ \lambda_{\max}(\boldsymbol{A}^S) &:= \max\{\lambda_{\max}(A) \mid A \in \boldsymbol{A}^S\}.\end{aligned}$$

Even if we consider the symmetric case, some problems remain NP-hard [112, 173].

Theorem 11.27. *Let $A_c \in \mathbb{Q}^{n\times n}$ be a symmetric positive definite and entry-wise nonnegative matrix, and $A_\Delta = E$. Then*

- *checking whether 0 is an eigenvalue of some matrix $A \in \boldsymbol{A}^S$ is NP-hard,*
- *checking $\lambda_{\max}(\boldsymbol{A}^S) \in (\underline{a}, \overline{a})$ for a given open interval $(\underline{a}, \overline{a})$ is coNP-hard.*

However, there are some known subclasses for which the eigenvalue range or at least one of the extremal eigenvalues can be determined efficiently [72]:

- If A_c is *essentially nonnegative*, i.e., $(A_c)_{ij} \geq 0\ \forall i \neq j$, then $\lambda_{\max}(\boldsymbol{A}^S) = \lambda_{\max}(\overline{A})$.

- If A_Δ is *diagonal*, then $\lambda_{\min}(\boldsymbol{A}^S) = \lambda_{\min}(\underline{A})$ and $\lambda_{\max}(\boldsymbol{A}^S) = \lambda_{\max}(\overline{A})$.

In contrast to the extremal eigenvalues $\lambda_{\min}(\boldsymbol{A}^S)$ and $\lambda_{\max}(\boldsymbol{A}^S)$, the largest of the minimal eigenvalues and the smallest of the largest eigenvalues,

$$\max\{\lambda_{\min}(A) \mid A \in \boldsymbol{A}^S\},$$
$$\min\{\lambda_{\max}(A) \mid A \in \boldsymbol{A}^S\},$$

can be computed with an arbitrary precision in polynomial time by semidefinite programming [98]. As in the general case, checking whether a given vector $0 \neq x \in \mathbb{R}^n$ is an eigenvector of some matrix in $\boldsymbol{A}^S$ is a polynomial time problem. Nevertheless, strong polynomiality has not been proved yet.

We already know that computing exact bounds on many problems with interval data is intractable. Since we can do no better, we can inspect the hardness of various approximations of their solutions. The terms absolute and relative approximation are meant in the same way as in Section 8.3. While doing this we use the following assumption: *Throughout this section, we consider a computational model, in which the exact eigenvalues of rational symmetric matrices are polynomially computable.* The table below from [72] summarizes the main results. We use the symbol ∞ in case there is no finite approximation factor with polynomial complexity.

Theorem 11.28. *Approximating the extremal eigenvalues of $\boldsymbol{A}^S$ is of the following complexity.*

	abs. error	*rel. error*
NP*-hard with error*	*any*	< 1
polynomial with error	∞	1

The table below, also from [72], gives analogous results for the specific case of approximating $\lambda_{\max}(\boldsymbol{A}^S)$ when A_c is positive semidefinite.

Theorem 11.29. *Approximating the extremal eigenvalues of $\boldsymbol{A}^S$ with A_c rational positive semidefinite is of the following complexity.*

	abs. error	*rel. error*
NP*-hard with error*	*any*	$1/(32n^4)$
polynomial with error	∞	$1/3$

The tables sums up the generalized idea behind several theorems on computing extremal eigenvalues. For more information and formal details see [72].

At the end of this section we mention spectral radius.

Definition 11.30. Let $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$, we define the range of *spectral radius* naturally as

$$\varrho(\boldsymbol{A}) = \{\varrho(A) : A \in \boldsymbol{A}\}.$$

Notice that $\varrho(\boldsymbol{A})$ is a compact real interval due to continuity of eigenvalues. We define spectral radius for $\boldsymbol{A}^S$ similarly.

Complexity of computing $\overline{\varrho(\boldsymbol{A})}$ is an open problem (as Schur stability is; see Section 11.11), and, to the best of our knowledge, complexity of computing $\underline{\varrho(\boldsymbol{A})}$ has not been investigated yet.

Clearly, we have the two following polynomially solvable subclasses:

- If $\underline{A} \geq 0$, then $\varrho(\boldsymbol{A}) = [\varrho(\underline{A}), \varrho(\overline{A})]$ (follows from the Perron–Frobenius theory).
- If $\boldsymbol{A}$ is diagonal, then $\varrho(\boldsymbol{A}) = [\max_i \operatorname{mig}(\boldsymbol{a}_{ii}), \max_i \operatorname{mag}(\boldsymbol{a}_{ii})]$.

11.9.1 Summary

Problem	**Complexity**
Is λ eigenvalue of some $A \in \boldsymbol{A}$?	NP-hard
Is x eigenvector of some $A \in \boldsymbol{A}$?	strongly P
Is x Perron vector of nonnegative irreducible $\boldsymbol{A}$?	strongly P
Is 0 eigenvalue of some $A \in \boldsymbol{A}^S$?	NP-hard
Is x eigenvector of some $A \in \boldsymbol{A}^S$?	P
Does $\lambda_{\max}(\boldsymbol{A}^S)$ belong to a given open interval?	coNP-hard
Computing $\overline{\varrho(\boldsymbol{A})}$	?
Computing $\underline{\varrho(\boldsymbol{A})}$	?
Computing exact bounds on $\varrho(\boldsymbol{A})$ with $\boldsymbol{A}$ nonnegative	strongly P
Computing exact bounds on $\varrho(\boldsymbol{A})$ with $\boldsymbol{A}$ diagonal	strongly P

11.10 Positive definitness and semidefiniteness

We should not leave out positive definiteness and semidefiniteness. Here without the loss of the generality symmetric matrices are of the only interest. We distinguish between weak and strong definiteness.

Definition 11.31. A symmetric interval matrix $\boldsymbol{A}^S$ is weakly positive (semi)definite if some $A \in \boldsymbol{A}^S$ is positive (semi)definite.

Definition 11.32. A symmetric interval matrix $\boldsymbol{A}^S$ is strongly positive (semi)definite if every $A \in \boldsymbol{A}^S$ is positive (semi)definite.

Checking strong positive definiteness [170] and strong positive semidefiniteness [136] are both coNP-hard. Considering positive definiteness, there are some sufficient conditions that can be checked polynomially [172].

Theorem 11.33. *An interval matrix $\boldsymbol{A}^S$ is strongly positive definite if at least one of the following condition holds:*

- $\lambda_{\min}(A_c) > \varrho(A_\Delta)$,
- A_c *is positive definite and* $\varrho(|A_c^{-1}|A_\Delta) < 1$.

The second condition can be reformulated as $\boldsymbol{A}^S$ being regular and A_c positive definite. If the first condition holds with $\geq$, then $\boldsymbol{A}^S$ is strongly positive semidefinite.

In contrast to checking strong positive definiteness, weak positive definiteness can be checked in polynomial time by semidefinite programming [98]; this polynomial result holds also for a more general class of symmetric interval matrices with linear dependencies [76]. For positive semidefiniteness it need not be the case since semidefinite programming methods work only with some given accuracy.

11.10.1 Summary

Problem	Complexity
Is $\boldsymbol{A}^S$ strongly positive definite?	coNP-hard
Is $\boldsymbol{A}^S$ strongly positive semidefinite?	coNP-hard
Is $\boldsymbol{A}^S$ weakly positive definite?	P
Is $\boldsymbol{A}^S$ weakly positive semidefinite?	?

11.11 Stability

The last section is dedicated to an important and more practical problem – deciding a stability of a matrix. There are many types of stability. For illustration, we chose two of them – Hurwitz and Schur.

Definition 11.34. An interval matrix $\boldsymbol{A}$ is *Hurwitz stable* if every $A \in \boldsymbol{A}$ is Hurwitz stable (i.e., all eigenvalues have negative real parts).

Similarly, we define Hurwitz stability for symmetric interval matrices. Due to their relation to positive definiteness ($\boldsymbol{A}^S$ is Hurwitz stable if $-\boldsymbol{A}^S$ is positive definite), we could presume that the problem is coNP-hard [170]. The problem remains coNP-hard even if we limit the number of interval coefficients in our matrix [136].

Theorem 11.35. *Checking Hurwitz stability of $\boldsymbol{A}$ is* coNP*-hard on a class of interval matrices with intervals in the last row and column only.*

Likewise, as for checking regularity, also checking Hurwitz stability of $\boldsymbol{A}$ cannot be done by checking stability of matrices of type A_{yz} (see, e.g., [102]). On the other hand, it can be checked in this way for $\boldsymbol{A}^S$. For more discussion and historical context see [112] or [176]. As sufficient conditions we can use conditions for positive definiteness applied to $-\boldsymbol{A}$. For more sufficient conditions see, e.g., [122].

Definition 11.36. An interval matrix $\boldsymbol{A}$ is *Schur stable* if every $A \in \boldsymbol{A}$ is Schur stable (i.e., $\varrho(A) < 1$).

In a similar way, we define Schur stability for symmetric interval matrices. For general interval matrices, complexity of checking Schur stability is an open problem, however, for the symmetric case the problem is coNP-hard [170].

11.11.1 Summary

Problem	Complexity
Is $\boldsymbol{A}$ Hurwitz stable?	coNP-hard
Is $\boldsymbol{A}^S$ Hurwitz stable?	coNP-hard
Is $\boldsymbol{A}$ Schur stable?	?
Is $\boldsymbol{A}^S$ Schur stable?	coNP-hard

11.12 Further topics

We conclude the section about complexity with three particular problems:

- *Matrix power.* For an interval matrix $\boldsymbol{A}$ computing the exact bounds on $\boldsymbol{A}^2$ is strongly polynomial (just by evaluating by interval arithmetic), but computing the cube $\boldsymbol{A}^3$ turns out to be NP-hard [109].
- *Matrix norm.* Computing the range of $\|A\|$ when $A \in \boldsymbol{A}$ is a trivial task for vector ℓ_p-norms applied on matrices (including Frobenius norm or maximum norm) or for induced 1- and ∞-norms. On the other hand, determining the largest value of the spectral norm $\|A\|_2$ (the largest singular value) subject to $A \in \boldsymbol{A}$ is NP-hard [136].
- *Membership in matrix classes.* Based on Chapter 4 we can state the following results. Checking whether a matrix is nonnegative invertible, strictly diagonally dominant, Z-matrix, M-matrix or H-matrix can be done in strongly polynomial

time. Checking whether a matrix $\boldsymbol{A}$ is a P-matrix is coNP-complete [29]. Strong regularity is according to Theorem 4.33 equivalent to checking whether $A_c^{-1}\boldsymbol{A}$ is an H-matrix. Therefore, it is strongly polynomial.

11.12.1 Summary

Problem	**Complexity**
Compute $\boldsymbol{A}^2$	strongly P
Compute $\boldsymbol{A}^3$	NP-hard
Compute $\|\boldsymbol{A}\|_1$	strongly P
Compute $\|\boldsymbol{A}\|_\infty$	strongly P
Compute $\|\boldsymbol{A}\|_F$	strongly P
Compute $\|\boldsymbol{A}\|_2$	NP-hard
Is $\boldsymbol{A}$ a Z-matrix?	strongly P
Is $\boldsymbol{A}$ an M-matrix?	strongly P
Is $\boldsymbol{A}$ an H-matrix?	strongly P
Is $\boldsymbol{A}$ an strictly diagonally dominant?	strongly P
Is $\boldsymbol{A}$ a P-matrix?	coNP-complete
Is $\boldsymbol{A}$ strongly regular?	strongly P

12 LIME^2: interval toolbox

- History of LIME
- Features and goals
- Structure and packages
- Installation and use

In the last chapter we introduce our interval toolbox LIME (Library of Interval MEthods). It is not a direct part of this work, however it is strongly connected to it. Most of the methods mentioned in the previous chapters are implemented in LIME and the implementations are used for comparison of the methods. In this brief chapter we describe its history, background, goals and purpose. The overall structure and selected methods are discussed. At the end we mention installation, use and extension of LIME.

12.1 History

During our research there was a need to compare various algorithms solving a given task (e.g., computing determinants of interval matrices, computing enclosures of interval linear systems, etc.). LIME occurred as a by-product to keep all the implemented methods at one place.

LIME (it could be also called LIME^1) was originally a part of the master's thesis [82]. It was implemented for Matlab and Intlab [188]. It mostly contained methods related to solving square and overdetermined interval linear systems.

After Intlab became commercial, LIME was moved under Octave and its Interval package by Oliver Heimlich [62]. Since then new packages of LIME have appeared and methods have been rewritten many times with effort to make the code more efficient and clear. We sometimes refer to it as LIME^2. The last LIME logo is in Figure 12.1.

12.2 Features and goals

Most of the methods tested in this work are implemented in LIME. Many additional useful methods are also implemented. LIME has also instruments to produce graphical outputs. All graphical outputs are produced by LIME or Octave (however, most of

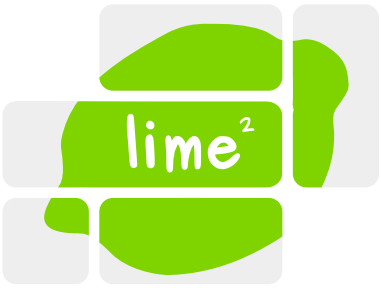

Figure 12.1: LIME^2 logo.

them were stored in an `.svg` file and further enhanced in Inkscape).

There are several goals of LIME:

- It is free for noncommercial use.
- It contains various methods solving one problem.
- The methods should be easy to use.
- It should be easy to develop and add new parts.
- It should be easily usable for interval educational purposes.
- The code should be clear and extensively documented (input and output parameters are described, implementation details are described, theorems on which the algorithms are based are cited, history of changes and known errors and future to do's are listed).
- It does not compete with existing interval toolboxes since their purpose is different.
- Packages are accompanied with examples or at least they are prepared for easy adding of new ones.

Most of the code is written solely by the author of this work. Some functions were implemented by other people (the author of the source code is referenced in each `.m` file). Unfortunately, there is still a lot of work to do (testing all the possible input cases for methods, correct handling of flags, adding more verification to some methods, etc.). Nevertheless the current state of LIME should allow other users to orient themselves quickly an to easily extend existing methods and functionality. The main goal of LIME is rather to share the code and be prepared for possible extension by others.

12.2.1 Verification and errors

Since LIME is a work of a few people it still might contain errors, even though we tried hard to catch most of the flaws. Some known errors are pointed out in `.Todo.` section

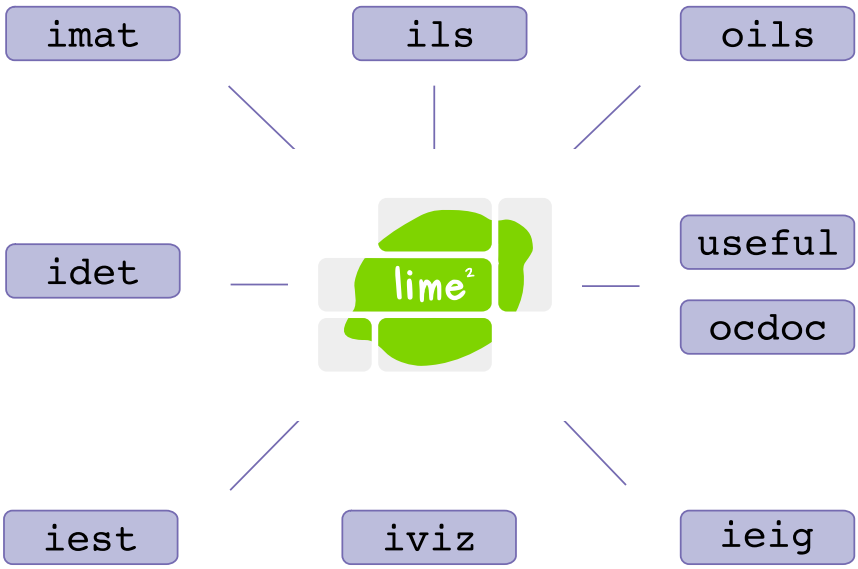

Figure 12.2: LIME structure.

of each `.m` file. There is an effort to make all the methods return verified results. In some cases it is not possible. In some cases verification is omitted to spare computational time. Such situations are documented in `.m` files if necessary. Most of the current code is implemented by the author of this work. Some functions were originally implemented by students supervised by David Hartman, Milan Hladík and Jaroslav Horáček (eigenvalues, matrix powers, interval estimations, interval determinants, etc.). Many more methods were written for Matlab and are waiting to be transfered to LIME (parametric interval systems, evaluation of polynomials, nonlinear solver, etc.)

12.3 Structure

For a better logical structure LIME is divided into several parts, we call them *packages*. They are depicted in Figure 12.2.

Here we list the packages with a brief description:

- `imat` – functions related to interval matrices,
- `ils` – methods connected to (square) interval linear systems,
- `oils` – methods connected to overdetermined interval linear systems,
- `idet` – methods for computing determinants of interval matrices,
- `iest` – interval data estimations and regressions,
- `ieig` – eigenvalues of (symmetric) interval matrices,
- `iviz` – methods for visualizations of intervals,

- `useful` – various methods that can be helpful,
- `ocdoc` – our minimalistic HTML documentation system.

Further we describe each package in bigger detail.

12.4 Packages

Each package is contained in a unique folder. It further contains three subfolders. The first is `doc` which contains the `.html` documentation of the package (every package has standalone documentation). The second folder is `test` that can contain examples corresponding to the package. For example in package `ils`, the folder `test` contains a function returning the example of interval linear system according to a given keyword. The origin of examples is referenced. The third is `develop` which contains functions under development.

12.4.1 `imat`

This package contains various methods working on interval matrices. Moreover, it contains methods for generating random matrices.

Function	Description
`ifcr`	full column rank test
`isregular`	regularity test
`issingular`	singularity test
`ismmatrix`	M-matrix test
`ishmatrix`	H-matrix test
`imatnorm`	various matrix norms
`imatinv`	inverse interval matrix
`vinv`	verified inverse of a real matrix
`imatpow`	power of an interval matrix

12.4.2 `ils`

Various methods connected to square interval linear systems are implemented here. Some methods work also for overdetermined interval systems (e.g., solvability and unsolvability testing, hull computation etc.).