Universität Ulm
Abteilung Theoretische Informatik
Leiter: Prof. Dr. Uwe Schöning

ON THE COMPLEXITY OF SOME PROBLEMS IN LINEAR ALGEBRA

DISSERTATION

zur Erlangung des Doktorgrades Dr. rer. nat.
der Fakultät für Informatik der Universität Ulm

vorgelegt von
THANH MINH HOANG
Ulm, November 2003

Amtierender Dekan:	Prof. Dr. F. W. von Henke
Gutachter:	Prof. Dr. Uwe Schöning
	Prof. Dr. Thomas Thierauf
	Prof. Dr. Jacobo Torán
Tag der Promotion:	19. Januar 2004

Acknowledgments

Foremost, I would like to thank my supervisor, Thomas Thierauf. His continuous support and guidance have been greatly influenced my research and this work. I am grateful to Uwe Schöning and Jacobo Torán for teaching me in theoretical computer science and for refereeing my thesis. I wish to thank Eric Allender and Meena Mahajan for many helpful comments and discussions on the topic of my thesis. Finally, I thank my parents, my family, Hang, Viet, and Duc for love and patience.

Contents

Chapter 1

Introduction

One of the main subjects of theoretical computer science is complexity theory which is more or less concerned with determining the intrinsic complexity of computational problems. A very important aim of complexity theory is to classify computational problems into different complexity classes defined by various bounds and types of resources (usually time and space). For such classification, the difficulty of some given computational problems within a complexity class can be meaningfully compared by using the fundamental concept of reductions. Thereby, in many instances the hardest of all computational problems in a complexity class can be identified, they are called the *complete problems* for the considered complexity class. Another obvious aim of complexity theory is to search for the interrelationship between various complexity classes.

Linear algebra is one of the most known mathematical disciplines because of its rich theoretical foundations and its many useful applications to science and engineering. Solving systems of linear equations and computing determinants are two examples of fundamental problems in linear algebra that have been studied for a long time ago. Leibnitz found the formula for determinants in 1693, and in 1750 Cramer presented a method for solving systems of linear equations, which is today known as *Cramer's Rule* (see [AM87]). This is the first foundation stone on the development of linear algebra and matrix theory. At the beginning of the evolution of digital computers, the matrix calculus has received very much attention. John von Neumann and Alan Turing were the world-famous pioneers of computer science. They introduced significant contributions to the development of computer linear algebra. In 1947, von Neumann and Goldstine [vNG47] investigated the effect of rounding errors on the solution of linear equations. One year later, Turing [Tur48] initiated a method for factoring a matrix to a product of a lower triangular matrix with an echelon matrix (the factorization is known

as *LU decomposition*). At present, computer linear algebra is broadly of interest. This is due to the fact that the field is now recognized as an absolutely essential tool in many branches of computer applications that require computations which are lengthy and difficult to get right when done by hand, for example: in computer graphics, in geometric modeling, in robotics, etc.

In the complexity-theoretic viewpoint, in particular with respect to parallel computations, computational problems in linear algebra provide a sure enough interesting topic. The present thesis focuses on this topic. The main goal of this thesis is to determine precisely the complexity of some fundamental problems in linear algebra. On the other hand, the interrelationship between complexity classes in which the considered problems are located will be clarified.

The motivation for this thesis comes mainly from the purpose to understand the complexity of computational problems in linear algebra. Many tasks of linear algebra are recognized usually as elementary problems, but the precise complexity of them was not known for a long time ago. Computing the determinant is an example. There are polynomial-time algorithms for the determinant (see [vzGG99], Part I, Chapter 1, Section 5.5., Algorithm 5.10). With respect to parallel computations, the problem of computing determinants attracted a great attention. It was shown in [Ber84, BvzGH82, Chi85, Csa76] that the determinant is computable simultaneously in polylog-time by using a polynomial number of processors. In particular, the result of Berkowitz [Ber84] showed that the problem of computing the determinant is solvable by uniform Boolean circuits of $O(log^2 n)$-depth and polynomial-size, i.e. the determinant is in the class $\mathbf{NC}^2$ (see Chapter 2 below for more detail on $\mathbf{NC}$ and its subclasses). Many computational problems in linear algebra are reducible in a natural way to computing determinants, and hence they are known to be in $\mathbf{NC}^2$. However, $\mathbf{NC}^2$ does not capture the exact complexity of linear-algebraic problems. Taking in consideration that the determinant is not known to be $\mathbf{NC}^2$-complete, it is natural to ask for which complexity class this problem is complete, and whether the complexity of other fundamental problems in linear algebra can be found.

Counting problems and counting classes

Counting problem is a type of computational problems, which is more difficult than *decision problem* and *search problem*. The major difference between these three types of computational problems can be explained as follows: a decision problem asks whether a solution exists, a search problem demands to compute

a solution, but a counting problem counts the number of all solutions. The perfect matching problem for graphs seems to be a good example: for a given graph G, the decision problem asks whether there is a perfect matching in G, the search problem demands to construct one of the perfect matchings in G (if one exists), and the counting version requires to compute the number of all perfect matchings in G. Although the perfect matching decision problem can be solved in deterministic polynomial-time [Edm65], counting the number of all perfect matchings in a graph is a very difficult problem for which maybe no polynomial-time algorithm can be developed. Just now is the question: *How difficult the problem of counting the number of all perfect matchings in a graph will be stepped?*.

The two most known time-complexity classes are **P** and **NP**. **P** is the class of all *decision problems* solvable in deterministic polynomial-time, and **NP** is the class of all decision problems solvable in nondeterministic polynomial-time. A deterministic polynomial-time algorithm is usually called *efficient*. Therefore, one can intuitively say that **P** contains only efficient computational problems. In contrast to **P**, **NP**-complete problems are called *intractable*, since no polynomial-time algorithm for any of these problems is known today. A large number of computational problems were shown by Cook and Karp [Coo71, Kar72] to be **NP**-complete. Whether there is a polynomial-time algorithm for any **NP**-complete problem is a formulation (in the algorithmic viewpoint) for the number-one open question $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ in theoretical computer science.

Other complexity classes beyond **NP** were (and are) widely of interest. One of them is the counting class #**P** which is extended from **NP** among the most natural way. In 1979, Valiant [Val79b, Val79c] initiated the study of the computational complexity of counting problems. He introduced the counting class #**P** that counts the number of solutions of **NP**-problems, or equivalently, the number of all accepting computation paths of a nondeterministic Turing machine on an input.

Computing the number of all truth assignments satisfying a given Boolean expression is the counting version corresponding to the very popular **NP**-complete decision problem SAT. This counting problem is known to be complete for #**P** (the proof can be found in [Pap94], Chapter 18, page 442). More interestingly, there are #**P**-complete counting problems derived from problems solvable in polynomial-time. For example: one can decide in polynomial-time whether a bipartite graph has a perfect matching [Edm65], but the number of all perfect matchings in a bipartite graph is a #**P**-complete function [Val79b]. Moreover, since the number of all perfect matchings in a bipartite graph G is equal to the

permanent of the adjacency matrix of G, the problem of computing the permanent of 0-1 matrices is also #**P**-complete.

In the logspace setting, some interesting classes related to the number of accepting computation paths of a nondeterministic logspace Turing machine (**NL** machine) are defined. Further on, we mention briefly some of these classes that are characterized by linear-algebraic problems.

The starting point in this direction is the class #**L**. In analogy to #**P**, a function of #**L** counts intuitively the number of all accepting computation paths of an **NL** machine on an input. Counting the number of all paths from node s to node t in an acyclic directed graph is a natural example of #**L**-complete functions. Furthermore, there are also #**L**-complete functions that are descended from linear algebra. An example of them is the problem of computing an element of the power matrix A^m, for a given 0-1 matrix A and an exponent $m \geq 1$.

Due to the fact that functions in #**P** are restricted to nonnegative integers, Fenner, Fortnow, and Kurtz [FFK94] extended #**P** to the class **GapP** which is the closure of #**P** under subtraction, i.e. any difference of two #**P** functions is a **GapP** function. The permanent of an integer matrix is an example of functions in **GapP**. Corresponding to **GapP** in the logspace setting is the class **GapL** defined as the class of differences of #**L**-functions. Since the determinant of an integer matrix is a **GapL**-complete function [Dam91, Tod91, Vin91, Val92], **GapL** seems to be one of the most suitable classes for elementary linear-algebraic problems. The huge difference in the complexity of the permanent and the determinant is somewhat surprising because of the fact that these functions have almost the same cofactor expansion[1], the only difference comes from the sign. This difference is more obvious by the following formulas

$$\det(A) = \sum_{\delta \in S_n} \operatorname{sign}(\delta) \prod_{i=1}^{n} a_{i,\delta(i)} \ , \quad \operatorname{perm}(A) = \sum_{\delta \in S_n} \prod_{i=1}^{n} a_{i,\delta(i)},$$

where $A = [a_{i,j}]$ is a matrix of order n, S_n is the symmetric permutation group of $\{1, 2, \ldots, n\}$, and $\operatorname{sign}(\delta)$ is the sign of the permutation δ.

GapL turns out to capture the complexity of many other natural problems in linear algebra. For example, computing an element in integer matrix powering, or iterated integer matrix multiplication, is complete for **GapL**. There are also graph-theoretic problems which are complete for **GapL** [Tod91] (see also [ST98]). An example is the problem of counting the number of all shortest s-t-paths in a given directed graph G, for given two distinguished nodes s and t.

[1]Note that **GapL** $\subseteq$ **GapP**. However, there is no proof yet that **GapL** $\neq$ **GapP**.

Important and fundamental decision problems can be derived from **GapL** problems. For example: testing singularity of matrices, i.e. testing if the determinant of a given matrix is zero. Probably, the singularity problem is relevant for computing the inverse of a matrix because of the fact that, often in practice, it is necessary to decide whether there exists the inverse before computing it. The importance of decision problems derived from **GapL** problems motivates onward the complexity class $\mathbf{C_{=}L}$ (*Exact Counting in Logspace*). Allender and Ogihara [AO96] introduced $\mathbf{C_{=}L}$ as the class of decision problems that *verify* **GapL** functions. Obviously, the class $\mathbf{C_{=}L}$ captures the complexity of problems defined in a most natural way because **GapL**-complete functions yield decision problems that are complete for $\mathbf{C_{=}L}$. For example, verifying the determinant is a $\mathbf{C_{=}L}$-complete problem [AO96, ST98].

Apart from the importance that $\mathbf{C_{=}L}$ characterizes many fundamental problems in linear algebra, $\mathbf{C_{=}L}$ is widely of interest because it is still open whether this class is closed under complement [ABO99]. As usual in complexity theory, it is plausible to expect that there is a positive answer to this open question because the following classes related to $\mathbf{C_{=}L}$ are closed under complement.

- **NL** (*Nondeterministic Logspace*) [Imm88, Sze88].
- **SL** (*Symmetric Logspace*) [NTS95].
- **PL** (*Probabilistic Logspace*) [Ogi98].
- **UL**/*poly* (*nonuniform Unambiguous Logspace*) [RA00], this result gives rise to conjecture that (uniform) **UL** is closed under complement as well.

Actually, together with the wish to understand the complexity of fundamental problems in linear algebra, the open question $\mathbf{C_{=}L} \stackrel{?}{=} \mathbf{coC_{=}L}$ and other unsolved questions about classes related to counting the number of accepting computation paths of an **NL** machine motivate the present thesis.

This thesis

The main contribution of this thesis is in determining the complexity of some problems in linear algebra.

The problems considered in this thesis can be divided into two categories. The first category consists of problems related to the structure of a matrix: about the characteristic polynomial, the minimal polynomial, and the invariant factor system. Testing similarity and testing diagonalizability of matrices, two classical

problems, belong to the first category. Problems in the second category are related to counting the eigenvalues: the rank, the inertia, and stability of matrices.

The main results presented in Chapter 3 and 4 of this thesis have been published in refereed form in the papers [HT00, HT01, HT03a, HT02b, HT02a, HT03b].

The contribution of Chapter 5 hasn't been published yet. Chapter 5 deals with the complexity of the unique perfect matching problem, and with some necessary and sufficient conditions for collapsing logspace counting classes.

In the remaining text of this chapter we explain in detail the backgrounds and the results of this thesis.

The characteristic polynomial

Suppose the output of a computational problem is a matrix or a tuple of numbers. There is a subtlety one has to be careful about: for instance, by saying that integer matrix powering is in **GapL** we mean that each element of the resulting power matrix can be computed within **GapL**, i.e., for an $n \times n$ matrix A, this yields n^2 **GapL**-functions according to n^2 elements of the power matrix A^m, and each of them is a **GapL**-complete computation. Now there are two variants of the verification version derived from matrix powering:

1. one has to verify *one* element of A^m, i.e. verifying $(A^m)_{i,j} = a$, for given integer matrix A, integers a, numbers m, i, and j,

2. the other has to verify *all* elements of A^m, i.e., verifying $A^m = B$, for given integer matrices A and B, and a number m.

These both decision problems are complete for $\mathbf{C_=L}$. But the situation can be different: an example taken from [ST98] is provided by the inverse of a matrix (assume that the inverse exists). There are again two following variants of the verification version.

- v-InverseElement (verifying *one element* of the inverse)
  Input : A regular $n \times n$ integer matrix A, integers i, j, a, and $b \neq 0$.
  Question : $(A^{-1})_{i,j} = \frac{a}{b}$?

- v-Inverse (verifying the *inverse*)
  Input: Two regular matrices A and B.
  Question: $A^{-1} = B$?

The first problem, v-InverseElement, is known to be complete for $\mathbf{C}_{=}\mathbf{L}$. The second problem can be solved by computing the product AB and comparing it with the identity matrix I. Testing whether $AB = I$ can be done in $\mathbf{NC}^1$, a subclass of $\mathbf{C}_{=}\mathbf{L}$. Thus v-Inverse is in $\mathbf{NC}^1$. Now, under the assumption that there is a logspace reduction from v-InverseElement to v-Inverse, there exists always a positive answer to the most prominent open question $\mathbf{NL} \stackrel{?}{=} \mathbf{L}$ ($\mathbf{L}$ is the deterministic logspace)! Obviously, in the second decision problem above, namely v-Inverse, we put too much information into the input. This is the reason why the problem of verifying one element is harder than the problem of verifying all elements of the inverse[2].

The observation about two different variants of the inverse decision problem inspires the problem of verifying the characteristic polynomial by Santha and Tan [ST98]: given a matrix A and the coefficients of a polynomial p, one has to verify whether p is the characteristic polynomial of A. It was shown by Berkowitz [Ber84] that the coefficients of the characteristic polynomial of a matrix are reducible to the elements of an iterated matrix multiplication. Therefore, verifying one or all these coefficients can be done in $\mathbf{C}_{=}\mathbf{L}$ (see Chapter 3, Section 3.1.1 for more detail). Since the determinant is the constant term (apart from the sign) of the characteristic polynomial, it is obvious to see that verifying one coefficient of the characteristic polynomial is a $\mathbf{C}_{=}\mathbf{L}$-complete problem. Now, with the different complexities of two in above mentioned inverse problems in mind, v-InverseElement and v-Inverse, the question is: is it easier to verify all the coefficients of the characteristic polynomial than to verify just one of them? We show in Chapter 3, Section 3.1.2 that this is *not* true, and in particular, verifying the characteristic polynomial is still $\mathbf{C}_{=}\mathbf{L}$-complete. This result is the positive answer to an open problem by Santha and Tan [ST98]: whether verifying the characteristic polynomial of a matrix is complete for the class $m - V - DET$ (note that the latter class and $\mathbf{C}_{=}\mathbf{L}$ are the same).

The minimal polynomial, similarity, and diagonalizability

Let A be a square matrix. Obviously, the invariant factors of A are significant because these factors determine completely the structure of A. For a matrix, the invariant factors are usually given by the rational canonical form of the matrix (the word *rational* corresponds to the fact that the rational canonical form can be computed by using only rational operations on the elements of the matrix). The minimal polynomial of A is known to be the invariant factor with the highest

[2]Note however that it is still open whether $\mathbf{NC}^1 \neq \mathbf{C}_{=}\mathbf{L}$.

degree under all invariant factors of A.

Due to the importance of the invariant factors, and in particular, of the minimal polynomial, algorithms for the rational canonical form have been intensively studied before. Both problems, computing the rational canonical form as well as computing the minimal polynomial of a matrix, can be done in randomized $\mathbf{NC}^2$ [KS87]. For integer matrices, there are even $\mathbf{NC}^2$-algorithms [Vil97]. The best known deterministic algorithm for the minimal polynomial of an $n \times n$ integer matrix makes $O(n^3)$ field operations [Sto98]. Moreover, some procedures to compute the rational canonical form can be found in the standard textbooks of linear algebra (e.g. [Gan77a, Gan77b, HJ85, HJ91]).

The invariant factors and the minimal polynomial of an integer matrix belong to the topic of this thesis. By taking a different approach to compute the minimal polynomial of an integer matrix, we show in Chapter 3 that the problem of computing the minimal polynomial can be reduced to matrix powering and solving systems of linear equations. Therefore, the minimal polynomial can be computed within the $\mathbf{TC}^0$-closure of $\mathbf{GapL}$. Note that this closure, denoted by $\mathbf{TC}^0(\mathbf{GapL})$, is a subclass of $\mathbf{TC}^1$ which is contained in $\mathbf{NC}^2$. Furthermore, we show that the problem of computing the invariant factors is hard for $\mathbf{GapL}$.

With respect to the verification of the minimal polynomial of a matrix, for a square matrix A, observe that it is the same situation as for the characteristic polynomial: deciding whether the constant term d_0 of the minimal polynomial of A is identically equal to zero is a $\mathbf{C}_=\mathbf{L}$-complete problem because of the fact: $d_0 = 0$ if and only if A is singular. By comparing these two polynomials, there is a question: Is verifying the minimal polynomial complete for $\mathbf{C}_=\mathbf{L}$ as the problem of verifying the characteristic polynomial? We show that verifying the minimal polynomial can be done in $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$, the class of sets that can be written as the conjunction of sets in $\mathbf{C}_=\mathbf{L}$ and in $\mathbf{coC}_=\mathbf{L}$, and it is hard for $\mathbf{C}_=\mathbf{L}$.

A difference between the characteristic polynomial and the minimal polynomial of a matrix is composed of their degrees. For a matrix of order n, although the degree of its characteristic polynomial is exactly n, the degree of its minimal polynomial is at most n. Actually, determining the degree of the minimal polynomial is a fundamental and important task in linear algebra. The complexity of the problem of computing the degree of the minimal polynomial is investigated in Chapter 3, Section 3.2.3 of this thesis where it is shown that this problem is computationally equivalent to the problem of computing matrix rank. Note that the rank of matrix A, denoted by $\mathrm{rank}(A)$, is the number of all linearly independent rows of A. The complexity of matrix rank has been studied systematically by Allender, Beals, and Ogihara [ABO99]. They showed that, given a matrix A

and a number r, the problem of

- deciding whether $\text{rank}(A) \leq r$ is $\mathbf{C_=L}$-complete,
- deciding whether $\text{rank}(A) = r$ is complete for $\mathbf{C_=L} \wedge \mathbf{coC_=L}$, and
- computing a bit of $\text{rank}(A)$ is complete for $\mathbf{AC^0(C_=L)}$.

Similarly to these results we show that the problem of

- deciding whether the degree of the minimal polynomial is less than some given m is $\mathbf{C_=L}$-complete,
- deciding whether the degree of the minimal polynomial is equal to some given m is complete for $\mathbf{C_=L} \wedge \mathbf{coC_=L}$, and
- computing a bit of the degrees of the minimal polynomial is complete for $\mathbf{AC^0(C_=L)}$.

As mentioned before, the problem of deciding whether the constant term d_0 of the minimal polynomial is equal to zero is still $\mathbf{C_=L}$-complete. Let's consider the constant term of the characteristic polynomial: this is a **GapL**-complete function and the corresponding verification problem is complete for $\mathbf{C_=L}$. By the minimal polynomial, the situation is inverted: from the $\mathbf{C_=L}$ completeness result of the decision problem whether the constant term d_0 is identically equal to zero we cannot definitely say that the computation of d_0 is **GapL**-complete. It is natural to ask whether the constant term of the minimal polynomial can be computed in **GapL**, too. We show that this question is strongly connected to another open problem about $\mathbf{C_=L}$, namely *if the constant term of the minimal polynomial can be computed in* **GapL***, then* $\mathbf{C_=L}$ *is closed under complement*. This is an immediate consequence of a hardness result: the problem of deciding whether two matrices have the same constant term of the minimal polynomials is complete for $\mathbf{AC^0(C_=L)}$. Our results on the constant term of the minimal polynomial offer a new point of attack to the open question of whether $\mathbf{C_=L}$ is closed under complement.

A fundamental topic in linear algebra is the study of equivalence relations on matrices that naturally arise in theory and in applications. Similarity of matrices is one of such equivalence relations: two square matrices A and B are similar if there exists a nonsingular transformation matrix P such that $A = P^{-1}BP$. A fact in linear algebra states that A and B are similar if and only if they have the same rational canonical form. Santha and Tan [ST98] observed that testing

similarity of two matrices can be done in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. The question whether the problem of testing similarity of matrices is $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$-complete was still open in [ST98]. In Chapter 3, Section 3.3.1 we give a positive answer to this question: the similarity problem is complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Related to similarity is the problem of deciding whether a matrix is diagonalizable. A matrix is called *diagonalizable* if it is similar to a diagonal matrix. We show in Chapter 3, Section 3.3.2 that testing diagonalizability of matrices is $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$-complete as well as testing similarity. We extend the result to *simultaneous diagonalizability* which is the problem of deciding whether *all* k given matrices are diagonalizable by the same diagonalizing matrix.

Matrix inertia for testing stability and congruence

Besides the similarity relation on matrices, there are still two other relations, namely *equivalence* and *congruence* of matrices. Matrices A and B are equivalent if there are nonsingular matrices P and Q such that $A = PBQ$. Actually, testing equivalence of matrices is equivalent to computing the rank of a matrix, i.e. it is complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Note that congruence of matrices is defined only for symmetric matrices: two symmetric matrices A and B are *congruent* if there exists a nonsingular matrix P such that $A = PBP^T$. Sylvester's Law of Inertia states that matrices A and B are congruent if and only if they have the same *inertia*. Hence, an approach towards the inertia of a symmetric matrix is useful for testing matrix congruence. In this direction, we give an approach towards the inertia of an arbitrary square matrix. In general, the inertia of a square matrix A is defined to be the triple of the number of eigenvalues of A, counting multiplicities, with positive, negative, and zero real part, respectively. Matrix inertia belongs to a difficult topic in linear algebra. In the linear-algebraic context, the inertia problem is well known under alias *the problem of Routh-Hurwitz* (see e.g. [Gan77b], Volume II, Chapter XV). Chapter 4 of this thesis studies the complexity of matrix inertia and its related problems.

A simple idea to compute the inertia could be to determine all the roots of the characteristic polynomial of the given matrix. With the $\mathbf{NC}^2$-algorithm provided by Neff and Reif [Nef94, NR96] these roots can be approximated to some precision [ABO]. However, it is not clear to what precision we have to approximate a root in order to tell it apart from zero. The result by Neff and Reif is different from our approach because our aim is to compute precisely the inertia. Using *Routh-Hurwitz's Theorem*, we show that the inertia of a matrix

(under the restrictions of the Routh-Hurwitz Theorem) can be computed in the probabilistic logspace **PL**, and furthermore, the inertia is hard for **PL**.

Consider the classical verification of the inertia: for a given square matrix A and integers $\mathfrak{p}$, $\mathfrak{n}$, and $\mathfrak{z}$, one has to decide whether $(\mathfrak{p},\mathfrak{n},\mathfrak{z})$ is the inertia of A. We show in Section 4.1 that for certain matrices the verification of the inertia is complete for **PL**. Verifying the inertia is a general version for the decision problem whether a system of differential equations is stable, a important task in engineering science. Note that a system of differential equations is stable if and only if its coefficient matrix is stable, i.e. all eigenvalues of its coefficient matrix have negative real parts. We show in Section 4.2 that testing stability of matrices is complete for **PL**.

By modifying the standard Routh-Hurwitz method, we show in Section 4.1 that the inertia of a symmetric integer matrix can be computed within **PL**. It follows that testing congruence of matrices can be done in **PL**. In addition, we show that $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ is a lower bound for the latter problem. Note that there are deterministic sequential algorithms to compute the inertia of a symmetric integer matrix, an example of them can be found in [For00].

Unique perfect matching

The problem of deciding whether a graph has a perfect matching is a classical example for which there are polynomial-time algorithms [Edm65] and randomized **NC** algorithms [KUW86, MVV87], but at present no fast parallel algorithm in deterministic **NC** has been found. Whether there is a deterministic **NC** algorithm for this decision problem is an outstanding open question in the area of parallel computation. Many researchers believe that the answer to this question lies in the affirmative.

The perfect matching problem can be equivalently represented as two linear-algebraic problems: given graph G, one asks whether the symbolic Tutte matrix corresponding to G is nonsingular [Tut47], the other asks whether the permanent of the 0-1 adjacency matrix of G is positive. Note that the problem of computing the symbolic determinant in variables $x_1, x_2, \ldots, x_m$ is very hard: even the problem of deciding whether a symbolic determinant contains a nonzero multiple of the term $x_1 x_2 \cdots x_k$ is **NP**-complete (see e.g. [Pap94], Problem 11.5.4). Therefore, one can believe that an algorithm for solving (in deterministic **NC**) the perfect matching problem would require some advances and tools of (linear) algebra.

Although it is still open whether the perfect matching problem belongs to

NC, there are deterministic **NC** algorithms for some of its special cases. An example of them is for the *unique perfect matching problem* where one has to decide whether a given graph has only *one* perfect matching. The unique perfect matching problem for bipartite graphs has been studied first of all by Rabin and Vazirani [RV89], and later by Kozen et al. [KVV85] (see also [Vaz93]). In Chapter 5 we redemonstrate the problem as a new example of graph theoretic problems that can be solved in $\mathbf{C_{=}L}$. Another motivation for our interest in the unique perfect matching problem comes from a totally unrelated subject, i.e. from the question: which class is characterized by this problem. Furthermore, we show that the problem is hard for **NL** and that the unique perfect matching in a given graph can be constructed in **GapL**.

Conditions for collapsing logspace counting hierarchies

Allender, Ogihara, and Beals [ABO99] noted that whether $\mathbf{C_{=}L}$ is closed under complement is more or less (with respect to the used notions of reducibility) equivalent to the open question by Von zur Gathen [vzG93]: Is there a reduction from the problems of deciding whether given rational vectors are linearly independent (INDEPENDENCE) to the problem of deciding whether a rational matrix is singular (SINGULAR)? $\mathbf{C_{=}L} \stackrel{?}{=} \mathbf{coC_{=}L}$ is an interesting question because a positive answer to it would close some gaps between lower and upper bounds on the complexity of some fundamental problems in linear algebra, and on the other hand, many still unknown relations between small complexity classes in the logspace setting would be clarified. The question seems to be simple, although at present no affirmative answer is known.

A part of Chapter 5 discusses the mentioned open question by showing some necessary and sufficient conditions for the collapse of the $\mathbf{C_{=}L}$ hierarchy. More precisely, we show that $\mathbf{C_{=}L} = \mathbf{coC_{=}L}$ if and only if for given matrix A one can compute in **GapL** two numbers r and s such that the rank of A is equal to the fraction $\frac{r}{s}$. A condition for $\mathbf{C_{=}L} = \mathbf{coC_{=}L}$ is similarly established over the degree of the minimal polynomial.

Obviously, $\mathbf{C_{=}L} = \mathbf{coC_{=}L}$ if the matrix rank can be computed in **GapL**. Is the rank of a matrix computable in **GapL**? In Chapter 5, we show that the latter happens if and only if $\mathbf{C_{=}L} = \mathbf{SPL}$. In analogy to the class **SPP** [FFK94], **SPL** [ARZ99] is the class of all languages having characteristic functions in **GapL**. Allender and Ogihara [AO96] noted that there is no reason to believe that **NL** is a subset of (uniform) **SPL**. It follows that there is no reason to expect that the rank of a matrix is computable in **GapL**.

Similarly to the conditions concerning the rank, for further collapse of **PL** we determine some conditions concerning matrix rank. In particular, we show that $\mathbf{PL} = \mathbf{SPL}$ if and only if the signature, or the number of all positive eigenvalues of a symmetric matrix can be computed in **GapL**.

Organization of this thesis

The present thesis consists of five chapters. The remainder of this thesis is organized as follows.

In preliminary Chapter 2, we describe briefly some basic notions, definitions, and concepts of complexity theory and linear algebra that are used throughout this thesis.

In Chapter 3, we study mainly the rational canonical form of matrices; the results on the characteristic polynomial, the invariant factor system, and the minimal polynomial are presented. Furthermore, the chapter treats the problem of testing similarity and diagonalizability of matrices.

In Chapter 4, we investigate the complexity of problems concerning inertia, stability, and congruence of matrices.

In Chapter 5, after the unique perfect matching problem we show some necessary and sufficient conditions for collapsing logspace counting hierarchies.

This thesis ends with a summary of results and with some open problems.

Chapter 2

Preliminaries

This chapter describes some basic materials, standard definitions and notations that are used though the thesis. The specified preliminaries will be given at the corresponding places of their use. We assume familiarity with fundamental notions and concepts of complexity theory as well as with basic linear algebra that can be found in standard textbooks in the area of complexity theory (e.g. [Pap94, BDG88, BDG91, HO02]), and of linear algebra (e.g. [Gan77a, Gan77b, HJ85, HJ91]), respectively. This chapter consists of two sections concerning linear algebra and complexity theory, respectively.

2.1 Linear algebra

2.1.1 Basic notations

We gather some standard notations from linear algebra. Let's denote by

- $\mathbb{F}$ an arbitrary field,
- $\mathbb{N}$ the set of natural numbers,
- $\mathbb{Z}$ the ring of integers, and
- $\mathbb{Q}$ the set of rational numbers.

An $m \times n$ matrix over the field $\mathbb{F}$ is denoted by $A = [a_{i,j}] \in \mathbb{F}^{m \times n}$, i.e.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}.$$

The element at position (i, j) in the matrix A sometimes is denoted by $A_{i,j}$. Partitioning a matrix into blocks we get a so-called *block* matrix whose elements are again matrices. Matrix A is called real, rational, integer, or 0-1 matrix if all of its elements are real numbers, rational numbers, integers, or from $\{0, 1\}$, respectively. A matrix whose elements are purely zero is called zero-matrix which is denoted by $\mathbf{0}$. In the case when $m = n$, A is a square matrix of order n, and we write shortly $A \in \mathbb{F}_n$. A square matrix A is called *diagonal* if $a_{i,j} = 0$, for all $i \neq j$, and we denote it by $A = \text{diag}[a_{1,1}, a_{2,2}, \ldots, a_{n,n}]$. A diagonal matrix whose diagonal elements are purely 1 is called *identity matrix* which is denoted by I. For clarity, sometimes we denote the identity matrix of order n by I_n. The *transpose* of matrix A is denoted by A^T, i.e., $A^T = [a_{j,i}] \in \mathbb{F}^{n\times m}$, for $A = [a_{i,j}] \in \mathbb{F}^{m\times n}$. A square matrix that fulfills $A = A^T$ is called *symmetric*.

By a *column vector*, or shortly a *vector*, of length m we mean an $m \times 1$ matrix. We denote vectors by bold letters. A *row vector*, or shortly a *line*, is the transpose of a vector. For $A = [a_{i,j}] \in \mathbb{F}^{m\times n}$, there is another representation of A, we write

$$A = [\boldsymbol{a}_1\ \boldsymbol{a}_2\ \cdots\ \boldsymbol{a}_n],$$

where $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n$ are the columns of A. We denote the vector of length nm that concatenates all the n columns of A by $vec(A)$, i.e.

$$vec(A) = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_n \end{bmatrix}.$$

For $A = [a_{i,j}] \in \mathbb{F}^{m\times n}$, $B = [b_{i,j}] \in \mathbb{F}^{k\times l}$, and $s \in \mathbb{F}$, one can define

- the *addition* of two matrices by $A + B = [a_{i,j} + b_{i,j}]$, for $m = k$ and $n = l$,
- the *product* of a scale with a matrix by $sA = [sa_{i,j}]$,
- the *product* of two matrices by $AB = [\sum_{l=1}^{n} a_{i,l}b_{l,j}] \in \mathbb{F}^{m\times l}$, for $n = k$
- the *tensor product* of two matrices by $A \otimes B = [a_{i,j}B] \in \mathbb{F}^{mk\times nl}$, and
- the *Kronecker sum* of two square matrices by $A \oplus B = I_k \otimes A + B \otimes I_m$, for $m = n$ and $k = n$.

These basic operations are associative, and except for the matrix multiplication and the tensor product they are commutative.

2.1.2 The rank, the determinant, and the characteristic polynomial

The vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ are called *linearly dependent* if there exist non-trivial $s_1, s_2, \ldots, s_n$ from $\mathbb{F}$ such that $s_1\boldsymbol{a}_1 + s_2\boldsymbol{a}_2 + \cdots + s_n\boldsymbol{a}_n = \boldsymbol{0}$, otherwise these vectors are called *linearly independent*.

Basically, it is known that the number of all linearly independent columns of matrix A is equal to the number of all the linearly independent rows of A. This number is called the *rank* of A and denoted by $\mathrm{rank}(A)$.

One of the most important functions for square matrices is the determinant. Recall that the determinant and the permanent of matrix $A \in \mathbb{F}_n$ is defined by

$$\det(A) = \sum_{\delta \in \mathcal{S}_n} \mathrm{sign}(\delta) \prod_{i=1}^{n} a_{i,\delta(i)}, \tag{2.1}$$

$$\mathrm{perm}(A) = \sum_{\delta \in \mathcal{S}_n} \prod_{i=1}^{n} a_{i,\delta(i)}, \tag{2.2}$$

where $\mathcal{S}_n$ is the symmetric permutation group of $\{1, 2, \ldots, n\}$, and $\mathrm{sign}(\delta)$ is the sign of the permutation δ ($\mathrm{sign}(\delta)$ is $+1$ or -1, according to whether the permutation is even or odd, respectively). There is a 1-1 relation mapping each permutation δ from $\mathcal{S}_n$ to a *permutation matrix* P_δ which is obtained by permuting the rows of I in conformity with δ. In particular, $\det(P_\delta) = \mathrm{sign}(\delta)$.

Let A be a matrix of order n. If $\det(A) = 0$, then A is called *singular*, otherwise A is called *nonsingular* (or *regular*, or *invertible*). A very fundamental connection between the rank and the determinant is stated by

$$\mathrm{rank}(A) = n \iff \det(A) \neq 0.$$

If A is nonsingular, then there always exists the inverse of A. The inverse is denoted by A^{-1} that fulfills $AA^{-1} = I_n$. Matrix A^{-1} can be computed by Cramer's rule, i.e. by the following formula

$$A^{-1} = \frac{1}{\det(A)} \mathrm{adj}(A) \tag{2.3}$$

where $\mathrm{adj}(A)$ is the *(classical) adjoint* of A. The matrix $\mathrm{adj}(A)$ is defined as the transpose of the matrix $B = [b_{i,j}]$ with the elements $b_{i,j} = (-1)^{i+j} \det(A_{j|i})$, for all i, j, where $A_{j|i}$ is the matrix obtained by deleting the j-th row and the i-th column of A.

The determinant of a square sub-matrix of a matrix A is called *minor*. For $A \in \mathbb{F}_n$ and $\alpha, \beta \subseteq \{1, 2, \ldots, n\}$, we denote the sub-matrix of A by $A_{\alpha|\beta}$ obtained

by deleting all rows and columns with indexes from α and β, respectively. For $\alpha = \beta$, the minor $\det(A_{\alpha|\beta})$ is called *principal*.

With respect to minors, the determinant of a matrix can be inductively computed by the Laplace expansion along a row or a column, i.e.,

$$\det(A) = \sum_{i=1}^{n} (-1)^{i+k} a_{i,k} \det(A_{i|k}) \tag{2.4}$$

$$= \sum_{j=1}^{n} (-1)^{j+l} a_{l,j} \det(A_{l|j}), \tag{2.5}$$

for each $1 \leq k, l \leq n$. Based on the Laplace expansion one can show that the determinant of a triangular matrix is equal to the product of all diagonal elements in the matrix.

There are other ways to compute the determinant of a matrix. For example, using elementary operations on rows and columns one can transform a matrix to a diagonal matrix and the determinant of the resulting matrix is the product of all diagonal elements of the resulting matrix. This method is usually called *Gaussian elimination* provided by the following observations. For $A \in \mathbb{F}_n$ and $c \in \mathbb{F}, c \neq 0$,

1. the sign of $\det(A)$ changes by changing two rows,
2. $\det(A)$ changes to $c \det(A)$ by adding i-th row multiplied by c to j-th row,
3. the last two statements hold by substituting the word "column" into "row".

Note that the rank of an arbitrary matrix can be determined by using similar operations on its rows and columns.

The characteristic polynomial of a matrix $A \in \mathbb{F}_n$ is defined to be the determinant of the polynomial matrix $xI - A$, where x is an indeterminate. We denote this polynomial by $\chi_A(x)$, i.e.

$$\chi_A(x) = \det(xI - A).$$

By using inductively the Laplace expansion for computing $\chi_A(x)$, one can show that the degree of $\chi_A(x)$ is precisely n and the coefficient corresponding to the term with highest degree, i.e. to the term x^n in $\chi_A(x)$, is equal to 1. Note that a polynomial whose highest coefficient is equal to 1 is called *monic*. Let's c_i be the coefficients of $\chi_A(x)$, i.e.

$$\chi_A(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0.$$

The constant term c_0 is important as well as the determinant because

$$c_0 = \chi_A(0) = \det(-A) = (-1)^n \det(A).$$

The roots $\lambda_1, \lambda_2, \ldots, \lambda_n$ of $\chi_A(x)$ over the set of complex numbers are called the eigenvalues of the matrix A, i.e. one can write

$$\begin{aligned} \chi_A(x) &= \prod_{i=1}^{n}(x - \lambda_i) \ , \text{ and} \\ \det(A) &= \prod_{i=1}^{n} \lambda_i \ . \end{aligned}$$

The latter equation implies the fact: A is singular if and only if one of its eigenvalues is zero. Let trace(A) be the sum of all diagonal elements of A. A theorem in linear algebra states that

$$\text{trace}(A^i) = \sum_{j=1}^{n} \lambda_j^i, \text{ for all } i. \tag{2.6}$$

A non-zero polynomial $p(x)$ over $\mathbb{F}$ is called an *annihilating polynomial* for A if $p(A) = \mathbf{0}$. Cayley-Hamilton's Theorem states that $\chi_A(x)$ is an annihilating polynomial for A, i.e.

$$\chi_A(A) = A^n + c_{n-1}A^{n-1} + \cdots + c_1 A + c_0 I = \mathbf{0}. \tag{2.7}$$

2.1.3 The invariant factors and canonical forms

For $A \in \mathbb{F}_n$, the *minimal polynomial* (of A) is defined to be a monic polynomial with minimal degree that annihilates the matrix A. We denote the minimal polynomial of A by $\mu_A(x)$. Let m and $1, d_{m-1}, \ldots, d_0$ be the degree and the arranged coefficients of $\mu_A(x)$, respectively. Then $\mu_A(x)$ is presented by

$$\mu_A(x) = x^m + d_{m-1}x^{m-1} + \cdots + d_1 x + d_0.$$

In analogy to (2.7) we have

$$\mu_A(A) = A^m + d_{m-1}A^{m-1} + \cdots + d_1 A + d_0 I = \mathbf{0}. \tag{2.8}$$

Let's denote the degree and the constant term of a polynomial p by $\deg(p)$ and $\text{ct}(p)$, respectively. We have $\deg(\mu_A(x)) = m$ and $\text{ct}(\mu_A(x)) = d_0$.

Let's define

$$i_{n-j+1}(x) = \frac{D_j(x)}{D_{j-1}(x)}, \quad D_0(x) \equiv 1, \quad \text{for } j = 1, 2, \ldots, n, \tag{2.9}$$

where $D_j(x)$ is the greatest common divisor (for short: gcd) of all minors of order j of the characteristic matrix $xI - A$. Then the monic polynomials $i_1, i_2, \ldots, i_n$ are called the *invariant factors* of the matrix A. They remain unchanged by every similarity transformation. The collection of these factors is called the *invariant factor system* of A. An observation about this system states that

$$\begin{array}{l} i_j(x) \text{ divides } i_{j-1}(x), \text{ for } j = 2, 3, \ldots, n+1, \\ \chi_A(x) \;=\; D_n(x) \;=\; \prod_{j=1}^{n} i_j(x), \text{ and} \\ \mu_A(x) \;=\; i_1(x). \end{array} \tag{2.10}$$

Obviously, if A is an integer matrix, then all coefficients of $\chi_A(x)$, $\mu_A(x)$, and invariant factors are also integers. Moreover, the set of all distinct roots of $\chi_A(x)$ is equal to the set of all distinct roots of $\mu_A(x)$. It is obvious to see from (2.10) that the minimal polynomial is a factor of the characteristic polynomial, and $1 \leq \deg(\mu_A(x)) \leq n = \deg(\chi_A(x))$. Thus, if the eigenvalues of matrix A are pairwise different, then $\mu_A(x) = \chi_A(x)$. There are other matrices fulfilling the latter property. Let's consider the following matrix P, constructed from the polynomial $p(x) = x^n + p_{n-1}x^{n-1} + \cdots + p_1 x + p_0$,

$$P = \begin{bmatrix} 0 & 0 & \cdots & 0 & -p_0 \\ 1 & 0 & \cdots & 0 & -p_1 \\ 0 & 1 & \cdots & 0 & -p_2 \\ & & \cdots\cdots\cdots & & \\ 0 & 0 & \cdots & 1 & -p_{n-1} \end{bmatrix}. \tag{2.11}$$

By using the Laplace expansion (2.4) along the last column of $xI - P$ we can compute $\chi_P(x)$. Observe that $\det((xI - P)_{1|n}) = (-1)^{n-1}$, thus the gcd of all minors of order $n-1$ of $xI - P$ is exactly 1. Therefore, due to (2.9) and (2.10) we get

$$\chi_P(x) = \mu_P(x) = p(x). \tag{2.12}$$

The matrix P such as in (2.11) is usually called the *companion matrix* of the polynomial $p(x)$.

Some canonical forms are defined for square matrices. As follows we describe three important kinds of them.

The $n \times n$ diagonal polynomial matrix

$$\mathcal{S}_A(x) = \operatorname{diag}[i_n(x), i_{n-1}(x), \ldots, i_1(x)] \tag{2.13}$$

is called the *Smith canonical form* of $xI - A$ (or shortly of A).

The *rational canonical form* of a square matrix A is given by

$$\mathcal{R}_A(x) = \text{diag}[C_1, C_2, \ldots, C_n], \tag{2.14}$$

where C_j is the companion matrix of the invariant factor $i_j(x)$ of A, for $j = 1, 2, \ldots, n$, respectively.

The invariant factors can be decomposed into irreducible divisors over the given field $\mathbb{F}$ approximately as follows

$$\begin{aligned} i_1(x) &= (e_1(x))^{j_{1,1}}(e_2(x))^{j_{1,2}} \cdots (e_s(x))^{j_{1,s}}, \\ i_2(x) &= (e_1(x))^{j_{2,1}}(e_2(x))^{j_{2,2}} \cdots (e_s(x))^{j_{2,s}}, \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ i_n(x) &= (e_1(x))^{j_{n,1}}(e_2(x))^{j_{n,2}} \cdots (e_s(x))^{j_{n,s}}, \end{aligned}$$

where $j_{1,k} \geq j_{2,k} \geq \cdots \geq j_{n,k} \geq 0$, for $k = 1, \ldots, s$. The monic irreducible divisors $e_1(x), e_2(x), \ldots, e_s(x)$ are distinct and they occur in $i_1(x), i_2(x), \ldots, i_n(x)$. The powers $(e_1(x))^{j_{1,1}}, \ldots, (e_s(x))^{j_{n,s}}$ that are different from 1 are called *elementary divisors* of A over $\mathbb{F}$. For simplicity, we denote the elementary divisors of A by $\varepsilon_1(x), \varepsilon_2(x), \ldots, \varepsilon_k(x)$.

The $n_l \times n_l$ upper triangular matrix of the form

$$J_l = \begin{bmatrix} \lambda & 1 & & & \mathbf{0} \\ & \lambda & 1 & & \\ & & \ddots & \ddots & \\ & & & \lambda & 1 \\ \mathbf{0} & & & & \lambda \end{bmatrix}$$

is called the *Jordan block* corresponding to the invariant divisor of the form $\varepsilon_l(x) = (x - \lambda)^{n_l}$ (where λ is in $\mathbb{F}$). The *Jordan canonical form* of the matrix A is defined to be the matrix

$$\mathcal{J}_A = \text{diag}[J_1, J_2, \ldots, J_k].$$

Thereby, the diagonal elements of $\mathcal{J}_A$ are the eigenvalues of A.

2.1.4 Equivalence relations on matrices

We are interested in three types of equivalence relations on matrices. Arbitrary matrices A and B are said to be *equivalent* if there are nonsingular matrices P and Q such that $A = PBQ$. Matrices $A, B \in \mathbb{F}_n$ are called *similar* if there is a nonsingular matrix $P \in \mathbb{F}_n$ such that $A = PBP^{-1}$. Symmetric matrices

$A, B \in \mathbb{F}_n$ are called *congruent* if there is a nonsingular matrix $P \in \mathbb{F}_n$ such that $A = PBP^T$.

The first type of these relations, namely matrix equivalence, is simply related to the rank: A and B are equivalent if and only if $\text{rank}(A) = \text{rank}(B)$.

For the second relation, a fact in linear algebra states that $xI - A$ is similar to $\mathcal{S}_A(x)$, and A is similar to $\mathcal{R}_A$ and $\mathcal{J}_A$, for every square matrix A. Therefore, there is a condition for similarity: A and B are similar if and only if $\mathcal{S}_A(x) = \mathcal{S}_B(x)$, or equivalently $\mathcal{R}_A = \mathcal{R}_B$, or equivalently $\mathcal{J}_A = \mathcal{J}_B$. Diagonalizability of matrices is immediately related to similarity of matrices. A matrix A is called *diagonalizable* if A is similar to a diagonal matrix. For example, symmetric matrices are diagonalizable because the Jordan canonical form of a symmetric matrix is a diagonal matrix. Matrices $A_1, \ldots, A_k$ are called *simultaneously diagonalizable* if there is a nonsingular matrix P such that $PA_1P^{-1}, \ldots, PA_kP^{-1}$ are diagonal matrices.

Congruence of symmetric matrices is conditional on the *inertia*. We explain the term "the inertia" in more detail.

Recall that the eigenvalues of a square matrix A are the roots of the characteristic polynomial $\chi_A(x)$ over the set of complex numbers. For a given matrix, counting the eigenvalues satisfying a particular property is meaningful and important to tell more about the matrix. The *inertia* of an $n \times n$ matrix A is defined to be the triple $(i_+(A), i_-(A), i_0(A))$, where $i_+(A)$, $i_-(A)$, and $i_0(A)$ are the number of eigenvalues of A, counting multiplicities, with positive, negative, and zero real part, respectively. Note that the inertia of a matrix consists of nonnegative integers and the sum of these is exactly n. Furthermore, square matrix A is called *positive stable* if $i(A) = (n, 0, 0)$ *negative stable* if $i(A) = (0, n, 0)$, and *positive semi-stable* if $i_-(A) = 0$. In the case when A is a *Hermitian* matrix, i.e. A and the conjugation of its transpose are equal: $A = A^{T*}$, all eigenvalues of A are real and the word *stable* will be replaced by *definite*. Note that in case of integer matrices we use the word "symmetric" instead of "Hermitian".

A theorem in linear algebra states that symmetric matrices A and B are congruent if and only if $i(A) = i(B)$.

2.2 Complexity theory

Let Σ be a finite alphabet. A *string* is an element of Σ^*, and a subset of Σ^* is called a *language* or shortly a *set*. The complement of a set L is denoted by $\overline{L}$. The complement of a class $\mathcal{C}$ of sets is defined by $\mathbf{co}\mathcal{C} = \{\overline{L} \mid L \in \mathcal{C}\}$. A computational *integer problem* is defined to be a subset of $\{0,1\}^* \times \{0,1\}^*$.

In this thesis, we consider only integer problems, hence we fix our alphabet to $\Sigma = \{0, 1\}$.

For $x \in \Sigma^*$, by $|x|$ we denote the length of x. By $\mathrm{abs}(a)$ we denote the absolute value of a real number a. For a set S, its *characteristic function* is defined by $c_S : \Sigma^* \to \{0, 1\}$ such that for all $x \in \Sigma^*$:

$$c_S(x) = \begin{cases} 1, & \text{if } x \in S \\ 0, & \text{if } x \notin S. \end{cases} \tag{2.15}$$

We denote the base 2 logarithm function by log. The keyword *logspace* is abbreviated for *logarithmic space bounded*.

We assume familiarity with the basic computational models such as standard Turing machine, either deterministic or nondeterministic, or Boolean circuits. In particular, we refer the reader to the papers [AO96, ABO99] for more detail on the logspace counting classes considered in this thesis.

2.2.1 Logspace counting classes

First of all, we assume familiarity with the following classes

- **NP**, the class of sets accepted by nondeterministic polynomial-time Turing machines,
- **P**, the class of sets accepted by deterministic polynomial-time Turing machines,
- **NL**, the class of sets accepted by nondeterministic logspace Turing machines,
- **L**, the class of sets accepted by deterministic logspace Turing machines,
- **FL**, the class of functions computed by deterministic logspace Turing machines, and
- **SL**, the class of sets accepted by deterministic symmetric logspace Turing machines.

Furthermore, $\mathbf{AC}^0$, **TC**-, and **NC**-classes, which are for integer problems, attend to our interest.

- $\mathbf{AC}^0$ is the class of problems solvable by a uniform family of Boolean circuits of polynomial-size and constant-depth with unbounded fan-in AND- and OR-gates, and NOT-gates.

- $\mathbf{TC}^i$, for each $i \geq 0$, is the class of problems solvable by a uniform family of Boolean circuits of polynomial-size and depth $O(\log^i n)$ with unbounded fan-in AND- and OR-gates, NOT-gates, and unbounded fan-in MAJ-gates (majority gates). Thereby, a majority gate outputs 1 if at least a half of its inputs is purely 1. $\mathbf{TC} = \cup_{i \geq 0} \mathbf{TC}^i$.

- $\mathbf{NC}^i$, for each $i \geq 0$, is the class of problems solvable by a uniform family of Boolean circuits of polynomial-size and depth $O(\log^i n)$ with bounded fan-in AND- and OR-gates, NOT-gates. $\mathbf{NC} = \cup_{i \geq 0} \mathbf{NC}^i$.

These classes and their properties can be found in [Pap94] (among other standard textbooks in the area of complexity theory). A simple relationship between them is well known as follows:

$$\mathbf{AC}^0 \subset \mathbf{TC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{SL} \subseteq \mathbf{NL} \subseteq \mathbf{TC}^1 \subseteq \mathbf{NC}^2 \subseteq \mathbf{NC} \subseteq \mathbf{P} \subseteq \mathbf{NP}.$$

As stated below, we describe the logspace counting classes.

For a nondeterministic Turing machine M, we denote the number of accepting and rejecting computation paths on input x by $\mathrm{acc}_M(x)$ and $\mathrm{rej}_M(x)$, respectively. The difference of these two quantities is denoted by $\mathrm{gap}_M(x)$, i.e., for all x,

$$\mathrm{gap}_M(x) = \mathrm{acc}_M(x) - \mathrm{rej}_M(x).$$

For counting problems, the class $\#\mathbf{P}$ is defined by Valiant [Val79b] to be the class of functions of the form $\mathrm{acc}_M(x)$ where M is an **NP** machine. In the logspace setting, $\#\mathbf{L}$ is defined by Álvarez and Jenner [ÀJ93], in analogy to $\#\mathbf{P}$, to be the class of all functions acc_M where M is an **NL** machine.

Definition 2.2.1 ([ÀJ93])
$\#\mathbf{L} = \{\mathrm{acc}_M \mid M$ *is a nondeterministic logspace Turing machine*$\}$.

It was noted in [ÀJ93] that **FL** is contained in $\#\mathbf{L}$.

The class **GapL** is defined by Allender and Ogihara [AO96] to be the closure of $\#\mathbf{L}$ under subtraction. Actually, **GapL** is the set of all functions gap_M such that M is a nondeterministic logspace Turing machine.

Definition 2.2.2 ([AO96])
$\mathbf{GapL} = \{\mathrm{gap}_M \mid M$ *is a nondeterministic logspace Turing machine* $\}$.

As we shall see, in the logspace setting, **GapL** is defined analogously to class **GapP** in [FFK94]. Note that, in all the above mentioned definitions of $\#\mathbf{L}$ and **GapL**, M is restricted (as in [ÀJ93, AO96]) to such a nondeterministic logspace

machine that halts on all computation paths on all inputs. For function classes $\mathcal{C}_1$ and $\mathcal{C}_2$, define

$$\mathcal{C}_1 - \mathcal{C}_2 = \{f - g \mid f \in \mathcal{C}_1,\ g \in \mathcal{C}_2\}.$$

Then **GapL** can be defined ([AO96], Proposition 2) by

$$\mathbf{GapL} = \#\mathbf{L} - \#\mathbf{L} = \#\mathbf{L} - \mathbf{FL} = \mathbf{FL} - \#\mathbf{L}.$$

Based on **GapL**, the classes $\mathbf{C_=L}$ and **PL** are defined.

Definition 2.2.3 ([AO96])

(i) $\mathbf{C_=L} = \{\ S \mid \exists f \in \mathbf{GapL},\ \forall x :\ x \in S \iff f(x) = 0\ \}$.

(ii) $\mathbf{PL} = \{\ S \mid \exists f \in \mathbf{GapL},\ \forall x :\ x \in S \iff f(x) \geq 0\ \}$.

Since it is open whether $\mathbf{C_=L}$ is closed under complement, it makes sense to consider the *Boolean closure of* $\mathbf{C_=L}$, i.e., the class of sets that can be expressed as a Boolean combination of sets in $\mathbf{C_=L}$. The class $\mathbf{C_=L} \wedge \mathbf{coC_=L}$ is defined in [ABO99] to be the class of intersections of sets in $\mathbf{C_=L}$ with sets in $\mathbf{coC_=L}$.

Definition 2.2.4 ([ABO99])
$L \in \mathbf{C_=L} \wedge \mathbf{coC_=L} \iff \exists L_1 \in \mathbf{C_=L},\ L_2 \in \mathbf{coC_=L} :\ L = L_1 \cap L_2$.

Note that $\mathbf{C_=L} \subseteq \mathbf{C_=L} \wedge \mathbf{coC_=L} \subseteq \mathbf{PL}$.

2.2.2 Reducibility and logspace counting hierarchies

In complexity theory, reducibility is a useful and central concept for comparing the difficulty of computational problems within a complexity class. The complexity classes described in the preceding section have been defined in a very formal way. There is another way to define these classes by using reduction concepts. For example, we can define **NL** to be the class of computational problems which are *logspace many-one* reducible to the *s-t* connectivity problem. (The latter is the problem of deciding whether there is a path in G from s to t, for a given acyclic directed graph G and two distinguished vertices s and t). In general, most of the complexity classes can be naturally defined to be the class of things reducible to some important problems.

Roughly, we say that problem P_1 reduces to problem P_2 if there is a function $\mathcal{R}$ to transform every input x of P_1 to an equivalent input $\mathcal{R}(x)$ of P_2 such that for solving P_1 on input x we have equivalently to solve P_2 on input $\mathcal{R}(x)$. The function $\mathcal{R}$ is called a *reduction* from P_1 to P_2.

Depending on the upper bound on the complexity of the used transformation function there are different kinds of reducibility that have been studied in the literature of complexity theory. For example, there are *Turing* reduction, *logspace many-one* reduction, *quantifier-free projection*, $\mathbf{AC}^0$ reduction, $\mathbf{NC}^1$ reduction, etc.. Allender and Ogihara [AO96] observed that most of the important complexity classes remain unchanged by defining as the class of all things reducible to a complete problem regardless of the notion of used reducibility. **NL** is an example of them. But the situation seems to be not the same as for classes characterized by linear-algebraic problems, i.e. they don't have the same considered property as of **NL**. It is necessary to describe in detail some reductions which are used in forthcoming.

For sets S_1 and S_2, we say that S_1 is logspace many-one reducible to S_2, denoted by $S_1 \leq_m^{\mathbf{L}} S_2$, if there is a function $\mathcal{R}$ computable in deterministic logspace such that for all inputs x, $x \in S_1 \iff \mathcal{R}(x) \in S_2$.

A *logspace disjunctive truth table* reduction from set S_1 to set S_2, denoted by $S_1 \leq_{dtt}^{\mathbf{L}} S_2$, is defined in [ABO99] to be a function f, computable in logspace, such that for all x, $f(x)$ produces a list of strings $(y_1, y_2, \ldots, y_r)$, with the property that $x \in S_1$ if and only if at least one of the y_i is in S_2. By substituting "all" into "at least one" in the definition of logspace dtt reduction we get the definition of *logspace conjunctive truth table* reduction from S_1 to S_2 (notation: $S_1 \leq_{ctt}^{\mathbf{L}} S_2$).

For sets S_1 and S_2, we say that S_1 is $\mathbf{AC}^0$*-reducible to* S_2, denoted by $S_1 \leq^{\mathbf{AC}^0} S_2$, if there is a family of logspace uniform circuits over unbounded fan-in AND- and OR-gates, NOT-gates, and (unbounded fan-in) oracle gates for S_2, with polynomial-size and constant-depth (for short: a uniform $\mathbf{AC}^0$ family of circuits) that computes S_1. Note that this kind of reducibility is not the same as $\leq_{\mathbb{F}}^{\mathbf{AC}^0}$ (see e.g. [ST98] for $\mathbf{AC}^0$-reductions over some field $\mathbb{F}$). A family of logspace uniform circuits means in the sense of [Ruz81, Gre93] that there is a deterministic logspace Turing machine to compute on input 0^n the description of the n-th circuit in the family.

An $\mathbf{NC}^1$-reduction, denoted by $\leq^{\mathbf{NC}^1}$, is a family of logspace uniform circuits over fan-in two AND- and OR-gates, NOT-gates, and oracle gates, with polynomial-size and logarithmic depth ([Bal91], see [ABO99]).

For functions f and g, we say that f is logspace many-one reducible to g, denoted by $f \leq_m^{\mathbf{L}} g$, if there is a function $\mathcal{R}$ computable in deterministic logspace such that for all inputs x, $f(x) = g(\mathcal{R}(x))$. In an analogous way one can define $\mathbf{AC}^0$-, $\mathbf{TC}^0$-, and $\mathbf{NC}^1$-many-one reductions from function f to function g: if there is an $\mathbf{AC}^0$, $\mathbf{TC}^0$, and $\mathbf{NC}^1$ family of logspace uniform circuits α such that $f(x) = g(\alpha(x))$ for every input x, respectively.

Under a particular notion of reducibility, we say that computational problem P is hard for class $\mathcal{C}$ if all problems in $\mathcal{C}$ can be reduced to P. Additionally, if the hard problem P is itself in $\mathcal{C}$, then P is called a complete problem for $\mathcal{C}$. We say that problem P_1 is equivalent to problem P_2 under a reduction if P_1 is reducible to P_2 and P_2 is reducible to P_1 (under the particular notion of reducibility).

In this thesis, we will show some hardness and completeness results. Unless otherwise stated, the used reductions are logspace many-one reductions.

We continue to gather some complexity classes for linear-algebraic problems.

Let $\mathcal{C}$ be a complexity class. Based on the $\mathbf{AC}^0$-reduction one can define $\mathbf{AC}^0(\mathcal{C})$, the so-called $\mathbf{AC}^0$-*closure* of $\mathcal{C}$, to be the class of all sets $\mathbf{AC}^0$-reducible to a set in $\mathcal{C}$. For our purpose, we consider the classes $\mathbf{AC}^0(\mathbf{GapL})$, $\mathbf{AC}^0(\mathbf{C_=L})$ and $\mathbf{AC}^0(\mathbf{PL})$, which are the classes of problems $\mathbf{AC}^0$-reducible to a $\mathbf{GapL}$-complete function, a $\mathbf{C_=L}$- and a $\mathbf{PL}$-complete set, respectively.

Firstly, Cook [Coo85] defined and studied the class of problems $\mathbf{NC}^1$-reducible to the determinant function. He denoted this class by $\mathbf{DET}$. As mentioned before, some important complexity classes are unchanged by using different notations of reducibility. However, it isn't known whether $\mathbf{DET}$ fulfills this property. The question whether $\mathbf{DET}$ is the same as the class of problems $\mathbf{AC}^0$-reducible to the determinant was first posed by Allender and Ogihara [AO96]. They have defined the logspace versions of the counting hierarchy by using *Ruzzo-Simon-Tompa reducibility* [RST84]. It was shown in [AO96] (see also [ABO99]) that the $\mathbf{C_=L}$-, the $\mathbf{PL}$-, and the $\#\mathbf{L}$-hierarchy correspond to $\mathbf{AC}^0$-closures, respectively, in the following sense.

- The $\mathbf{C_=L}$-hierarchy is defined to be $\mathbf{C_=L}^{\mathbf{C_=L}^{\cdot^{\cdot^{\cdot^{\mathbf{C_=L}}}}}}$.
  It was shown in [ABO99] that this hierarchy collapses to $\mathbf{AC}^0(\mathbf{C_=L}) = \mathbf{L}^{\mathbf{C_=L}} = \mathbf{NC}^1(\mathbf{C_=L})$.

- The $\mathbf{PL}$-hierarchy is defined to be $\mathbf{PL}^{\mathbf{PL}^{\cdot^{\cdot^{\cdot^{\mathbf{PL}}}}}}$.
  It was shown in [Ogi98, BF97] that $\mathbf{PL}$-hierarchy coincides with $\mathbf{AC}^0(\mathbf{PL}) = \mathbf{PL} = \mathbf{NC}^1(\mathbf{PL})$.

- The $\#\mathbf{L}$-hierarchy is defined to be $\#\mathbf{L}^{\#\mathbf{L}^{\cdot^{\cdot^{\cdot^{\#\mathbf{L}}}}}}$ which coincides with $\mathbf{AC}^0(\mathbf{GapL})$. Any sort of collapse of this hierarchy isn't known today.

Moreover, it was noted in [ABO99] that all these hierarchies are contained in $\mathbf{TC}^1$, a subclass of $\mathbf{NC}^2$, and that the above mentioned hierarchies are contained in the class $\mathbf{DET}$. Furthermore, Allender [All97] proved that if $\mathbf{DET} = \mathbf{AC}^0(\mathbf{GapL})$, then the $\#\mathbf{L}$-hierarchy collapses to some of its levels.

In analogy to the closure $\mathbf{AC}^0(\mathbf{GapL})$, we can introduce $\mathbf{TC}^0(\mathbf{GapL})$, i.e. the $\mathbf{TC}^0$-closure of $\mathbf{GapL}$, as the class of problems $\mathbf{TC}^0$-reducible to the determinant. Since $\mathbf{AC}^0 \subset \mathbf{TC}^0 \subseteq \mathbf{NC}^1$, it is obvious to see the containments $\mathbf{AC}^0(\mathbf{GapL}) \subseteq \mathbf{TC}^0(\mathbf{GapL}) \subseteq \mathbf{DET} \subseteq \mathbf{TC}^1$. But we don't know whether these classes are the same.

The following relation between the classes are known

$$\mathbf{C_=L} \subseteq \mathbf{C_=L} \wedge \mathbf{coC_=L} \subseteq \mathbf{AC}^0(\mathbf{C_=L}) \subseteq \mathbf{PL} \subseteq \mathbf{AC}^0(\mathbf{GapL}).$$

Some other complexity classes are defined in terms of $\mathbf{GapL}$. For example, $\mathbf{Mod}_m\mathbf{L}$ is the class of $\mathbf{GapL}$ functions f such that $f(x) \neq 0 \pmod m$, for every natural number m, respectively. The most important class of them is $\mathbf{Mod}_2\mathbf{L}$, i.e. $\oplus\mathbf{L}$. Over $\mathbb{Z}_2$, computing the determinant of a 0-1 matrix as well as computing an element in a power of a 0-1 matrix are complete for $\mathbf{Mod}_2\mathbf{L}$ [Dam90]. At present, no relationship between $\mathbf{Mod}_m\mathbf{L}$ and $\mathbf{C_=L}$ is known. A further important open problem attracting a great attention is whether $\mathbf{NL} \subseteq \oplus\mathbf{L}$.

$\mathbf{SPL}$ is another small logspace counting class which is defined in [ARZ99] to be the class of all languages having characteristic function in $\mathbf{GapL}$, i.e. $\mathbf{SPL} = \{L \mid c_L \in \mathbf{GapL}\}$. It is known that $\mathbf{SPL}$ is contained in $\mathbf{C_=L}$, $\mathbf{coC_=L}$, and $\mathbf{Mod}_m\mathbf{L}$ for each $m \geq 2$. However, we don't know any complete problem for $\mathbf{SPL}$. It was noted in [AO96] that there is no reason to conjecture that $\mathbf{NL}$ is contained in $\mathbf{GapL}$, or $\mathbf{SPL}$. The following relations are known.

$$\begin{array}{lcl} \mathbf{NC}^1 & \subseteq & \mathbf{GapL} \subseteq \mathbf{AC}^0(\mathbf{GapL}) \subseteq \mathbf{TC}^0(\mathbf{GapL}) \subseteq \mathbf{DET} \subseteq \mathbf{TC}^1 \\ \mathbf{NL} & \subseteq & \mathbf{C_=L},\ \mathbf{coC_=L},\ \mathbf{C_=L} \wedge \mathbf{coC_=L},\ \mathbf{AC}^0(\mathbf{C_=L}),\ \mathbf{PL} \\ \mathbf{SPL} & \subseteq & \mathbf{C_=L},\ \mathbf{coC_=L},\ \mathbf{Mod}_m\mathbf{L} \text{ for each } m \geq 2. \end{array}$$

2.2.3 Characterizations of logspace counting classes

Since each $\mathbf{NL}$ computation can be interpreted by an acyclic directed graph, the following problem

- PATH
  Input: An acyclic directed graph G, and two nodes s and t
  Output: $\mathrm{path}(G, s, t)$ (the number of paths $s \rightsquigarrow t$ in G)

is known to be complete for $\#\mathbf{L}$. Moreover, due to $\mathbf{NL}$ computations note that w.l.o.g. we can assume that the maximal out-degree of the input graph G is equal to 2.

We are interested in the logspace counting classes **GapL**, $\mathbf{C_=L}$, $\mathbf{C_=L} \wedge \mathbf{coC_=L}$, $\mathbf{AC^0(C_=L)}$, and **PL**. These classes are characterized by linear-algebraic problems. It is necessary to describe in detail some basic properties, including complete problems, of these classes.

W.l.o.g. we can restrict all matrix problems in this thesis to problems for integer matrices, i.e., unless otherwise stated, all input matrices are restricted to integer matrices. The reason for this restriction was already mentioned in [ABO99] that integer matrix problems are equivalent to rational matrix problems (under logspace reductions) based on the following observation: for given rational matrix A (each element of A is viewed as a division of two given integers), and for integer a and b, in order to verify $\det(A) = \frac{a}{b}$ one can decide whether $b\det(cA) - a\det(cI) = 0$ where integer c is the product of all the denominators appearing in elements of A.

The class GapL

Let's define the problem of computing the determinant of a matrix by

- DETERMINANT
  Input: An $n \times n$ integer matrix A.
  Output: $\det(A)$.

The class **GapL** can be defined in terms of the function DETERMINANT in the sense of the following theorem.

Theorem 2.2.5 ([Tod91, Dam91, Vin91, Val92], see also [AO96, MV97]) DETERMINANT *is complete for* **GapL** *under logspace many-one reductions.*

As in [AO96] we note that a function f is logspace many-one reducible to DETERMINANT if there is a function g computable in logspace such that, for all inputs x, $f(x)$ (which is viewed as a number written in binary) is equal to $\det(g(x))$.

GapL possesses some standard closure properties by the following theorem.

Theorem 2.2.6 ([AO96], Theorem 9) *Let f be any function in* **GapL**. *The following functions are in* **GapL**

(1) $f(g(\cdot))$, *for any function g in* **FL**,

(2) $\sum_{i=0}^{|x|^c} f(x,i)$,

(3) $\prod_{i=0}^{|x|^c} f(x,i)$, *and*

(4) $\binom{f(x)}{g(x)}$, *for any function* g *in* **FL** *such that* $g(x) = O(1)$.

Closure properties according to (2) and (3) of Theorem 2.2.6 state that **GapL** is closed under addition and multiplication. An improvement of the closure property according to (1) is given by Corollary 3.3 of [AAM03] that **GapL** is closed under composition.

Corollary 2.2.7 ([AAM03], Corollary 3.3) *The determinant of a matrix having* **GapL***-computable elements can be computed in* **GapL**.

The problem of computing an element of a power matrix is defined by

- PowerElement
  Input: An $n \times n$ matrix A, and natural numbers $1 \leq m, i, j \leq n$.
  Output: $(A^m)_{i,j}$.

PowerElement is complete for **GapL**. This result was shown by several people, including Berkowitz [Ber84], and Toda [Tod91].

Since a graph can be represented by its adjacency matrix, there are some problems about graphs that are complete for **GapL** [Val79a, Tod91, Dam91, ST98]. For example:

- WPath
  Input: An acyclic directed graph G with edges weighted by integers, and two distinguished nodes s and t.
  Output: The sum of weights of all paths from s to t in G.
  (Note that the weight of a path is defined to be the product of weights of all edges belonging to the path.)

- PathDifference
  Input: An acyclic directed graph G, and distinguished nodes s, t_1, t_2.
  Output: $\mathrm{path}(G, s, t_1) - \mathrm{path}(G, s, t_2)$.
  (Recall that $\mathrm{path}(G, s, t)$ is the number of paths from s to t in G. Similarly to Path, we can w.l.o.g. assume that the maximal out-degree of G is exactly 2.)

The class $\mathbf{C_=L}$

As an immediate consequence of Theorem 2.2.5, the problem of deciding whether an integer matrix is singular, i.e. whether the determinant of given matrix is zero, is $\mathbf{C_=L}$-complete. We denote this decision problem by Singularity.

Theorem 2.2.8 ([AO96, ST98]) SINGULARITY *is complete for* $\mathbf{C_=L}$ *under logspace many-one reductions.*

For a fixed function f, we define V-f to be the set of all pairs (x, y) such that $y = f(x)$. The set V-f is called the verification of the function f. For example, verifying the determinant is defined by

$$\text{V-DETERMINANT} = \{\ (A, a) \mid \det(A) = a\ \}.$$

The fact that V-DETERMINANT and V-f, for any **GapL**-complete function f, are $\mathbf{C_=L}$-complete follows directly from part (2) of Theorem 2.2.6. Hence, V-POWERELEMENT and V-WPATH are known to be complete for $\mathbf{C_=L}$.

Some closure properties of $\mathbf{C_=L}$ can be immediately transformed from **GapL** closure properties appearing in part (1), (2), and (3) of Theorem 2.2.6.

Proposition 2.2.9 ([AO96]) $\mathbf{C_=L}$ *is closed under logspace many-one, logspace conjunctive truth table, and logspace disjunctive truth table reductions.*

Let **FNL** be the class of all functions computable in nondeterministic logspace. In analogy to $\leq^{\mathbf{L}}_m$, $\leq^{\mathbf{L}}_{ctt}$, and $\leq^{\mathbf{L}}_{dtt}$, the nondeterministic reductions $\leq^{\mathbf{FNL}}_m$, $\leq^{\mathbf{FNL}}_{ctt}$, and $\leq^{\mathbf{FNL}}_{dtt}$ were defined respectively in [AO96]. It was shown by Allender and Ogihara [AO96] (Theorem 16, Proposition 17) that Proposition 2.2.9 can be strengthened by adding the word "nondeterministic" to each place before "logspace". Proposition 2.2.9 states the fact that $\mathbf{C_=L}$ is closed under intersection and union.

Although many closure properties of $\mathbf{C_=L}$ are known, it is still unknown whether $\mathbf{C_=L}$ is closed under complement. Note that there is a positive answer to the latter question if and only if there exists a logspace many-one reduction from SINGULARITY to $\overline{\text{SINGULARITY}}$.

The class PL

The properties of **PL** have been completely summarized in [AO96].

Let's denote by POSDETERMINANT the set of all square integer matrices with positive determinants. Then POSDETERMINANT is a complete set for **PL**. More general, the problem of deciding whether $f(x) \geq a$, for given a and **GapL**-function f, is complete for **PL**.

PL is known to be closed under complement [Ogi98]. Furthermore, this class is closed under (deterministic and nondeterministic) logspace many-one, ctt, and dtt reductions.

The classes $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ and $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$

It is known that the classes $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ and $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$ are characterized by problems concerning matrix rank.

Recall that the rank of a matrix A is the number of all linearly independent rows of A. The complexity of matrix rank has been studied by Allender, Beals, and Ogihara [ABO99]. They showed that the problem of computing any bit of the rank is complete for the $\mathbf{C}_=\mathbf{L}$ hierarchy. The problem of computing matrix rank can be interpreted by the following decision problem

$$\textsc{Rank} = \{(A, k, b) \mid \text{the } k\text{-th bit of } \mathrm{rank}(A) \text{ is } b\}.$$

Since the $\mathbf{C}_=\mathbf{L}$-hierarchy collapses to $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$, Rank is a complete problem for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. Furthermore, it was shown in [ABO99] that the set of all matrices having odd ranks, denoted by OddRank, the set of all matrices having even ranks, denoted by EvenRank, and the decision problem whether a system of linear equations is feasible, i.e. the set

$$\mathrm{FSLE} = \{(A, b) \mid A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^{m\times 1}, \exists x \in \mathbb{Q}^{n\times 1} : Ax = b\},$$

are also complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ under logspace many-one reductions. For simplicity, one can say that matrix rank characterizes $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Theorem 2.2.10 ([ABO99]) Rank, OddRank, EvenRank, *and* FSLE *are complete for* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

We denote the problem of deciding whether the rank of a matrix is equal to some number k by v-Rank, i.e.

$$\text{v-}\textsc{Rank} = \{\ (A, k) \mid \mathrm{rank}(A) = k\ \}.$$

The latter set is known to be complete for the second level of the Boolean hierarchy over $\mathbf{C}_=\mathbf{L}$.

Theorem 2.2.11 ([ABO99]) v-Rank *is complete for* $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$.

Obviously, the rank of an $m \times n$ matrix A can be determined by finding a number $i : 0 \leq i \leq \min\{m, n\}$ such that $\mathrm{rank}(A) = i$. Therefore, sets from $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ are logspace disjunctive truth table reducible to $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$. Conversely, each set which is logspace disjunctive truth table reduced to $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$ is logspace many-one reducible to FSLE, a complete set for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ [ABO99] (Lemma 2.11). However, note that it is still unknown whether $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$ is closed under logspace disjunctive truth table reductions.

Furthermore, the problem of deciding whether the rank of a given matrix is smaller than some given number k, denoted by $\textsc{Rank}_{\leq}$, is complete for $\mathbf{C}_=\mathbf{L}$.

Chapter 3

On the Matrix Structure

This chapter deals with the complexity of linear-algebraic problems concerning the structure of matrices. Results presented in this chapter are mainly summarized from the results presented in the papers [HT00, HT01, HT03a, HT02b, HT03b]. The text of this chapter is divided into three sections. So, in turn, we study the problem of verifying the characteristic polynomial in Section 3.1, some problems concerning the minimal polynomial and the invariant factor system in Section 3.2, and testing similarity and diagonalizability of matrices in Section 3.3.

3.1 The characteristic polynomial of a matrix

In this section we investigate the complexity of computing and verifying the characteristic polynomial of a matrix. Some basic facts about computing the characteristic polynomial are presented in 3.1.1. The main result that verifying the characteristic polynomial is complete for $\mathbf{C}_{=}\mathbf{L}$ is shown in 3.1.2.

3.1.1 Computing the characteristic polynomial

Recall that the characteristic polynomial of an $n \times n$ matrix A, denoted by $\chi_A(x)$, is defined to be $\det(xI - A)$ where x is a formal variable. Inductively, by using the Laplace expansion for computing $\det(xI - A)$ (see equations (2.4) and (2.5)), we see the fact that the degree of $\chi_A(x)$ is equal to n. Let $1, c_{n-1}, \ldots, c_1, c_0$ be the arranged coefficients of $\chi_A(x)$, i.e.

$$\begin{aligned} \chi_A(x) &= \det(xI - A) \\ &= x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0. \end{aligned}$$

The problem of computing the coefficients of the characteristic polynomial is a fundamental task in linear algebra. These coefficients can be derived to get some precise information about the mathematical object represented by the matrix. In the complexity-theoretic viewpoint, and especially with respect to parallel computations, this problem is also fundamental and interesting.

We define the problem of computing the characteristic polynomial by

- CharPolynomial
  Input: An $n \times n$ matrix A, and a natural number $i \leq n$.
  Output: The ith coefficient of the characteristic polynomial $\chi_A(x)$.

Our starting point is the following relation between the last coefficient c_0 of $\chi_A(x)$ and the determinant:

$$c_0 = (-1)^n \det(A). \tag{3.1}$$

Therefore, $\det(A)$ can be read off from c_0 and vice versa. Actually, this is the first idea to design a parallel $\mathbf{NC}^2$ algorithm for the determinant in the context of polynomials.

Furthermore, we concentrate on the question: "how to compute the other coefficients of $\chi_A(x)$?". Using a fact in linear algebra that (apart from the sign) c_i is equal to the sum of all the principal minors of order $n-i$ in A, for $i = 1, 2, \ldots, n-1$, one can not answer the question in the affirmative because there are $\binom{n}{i}$ minors of order $n-i$ in A.

A well known method for computing the coefficients of $\chi_A(x)$ is given by Leverrier (see [Gan77a], page 87):

1. Compute the traces $s_i = \mathrm{trace}(A^i)$, for $i = 0, 1, \ldots, n$.

2. Compute the coefficients c_j successively by Newton's formula

$$c_{n-j} = -\frac{1}{j}(s_j + c_{n-1}s_{k-1} + \cdots + c_{n-k-1}s_1), \text{ for } j = 1, 2, \ldots, n. \tag{3.2}$$

Using Newton's formula (3.2) and a parallel algorithm for solving systems of linear recurrences, Csanky [Csa76] presented the first $\mathbf{NC}^2$ parallel algorithm for the coefficients of the characteristic polynomial, and consequently for the determinant of a matrix. Unfortunately, Csanky's algorithm can not be used over finite fields, and obviously, it isn't division-free.

Berkowitz [Ber84] found the first $\mathbf{NC}^2$ algorithm for the characteristic polynomial and for the determinant. He showed that, for a given matrix A, a sequence

of matrices can be constructed in logspace such that all the coefficients of $\chi_A(x)$ correspond to the elements in the iterated product of the constructed matrices. Berkowitz's Theorem is given by [ABO99] without a proof as follows.

Theorem 3.1.1 [Ber84] *Given an $n \times n$ matrix A, one can construct in logspace a sequence of $p \times p$ matrices B_i such that the coefficients $c_{n-1}, \ldots, c_1, c_0$ of $\chi_A(x)$ appear respectively at positions $(1,n), \ldots, (1,2), (1,1)$ in the matrix $\prod_i B_i$.*

Since iterated matrix multiplication is equivalent to matrix powering [vzG93], the coefficients of the characteristic polynomial of a matrix are computable in **GapL**.

Proposition 3.1.2 *The coefficients of the characteristic polynomial of a matrix are computable in* **GapL**. CharPolynomial *is complete for* **GapL**.

3.1.2 Verifying the characteristic polynomial

As mentioned in Chapter 1 (see page 6), there are two versions of verifying the characteristic polynomial: one has to verify only one coefficient of this polynomial, the other has to verify all coefficients of this polynomial. By (3.1), apart from the sign $\det(A)$ is the constant term of the characteristic polynomial $\chi_A(x)$. Hence, the first verification version, i.e. the verification of one coefficient of the characteristic polynomial, is $\mathbf{C_=L}$-complete. The second verification version of the characteristic polynomial can be formally defined by

- v-CharPolynomial =
  $\{ (A, c_0, c_1, \ldots, c_{n-1}) \mid \chi_A(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0 \}$.

By Proposition 3.1.2, each coefficient of $\chi_A(x)$ is computable in **GapL**. So, verifying each coefficient is a $\mathbf{C_=L}$ predicate. Since $\mathbf{C_=L}$ is closed under logspace ctt reductions, v-CharPolynomial is in $\mathbf{C_=L}$. Santha and Tan [ST98] asked whether v-CharPolynomial is complete for $\mathbf{C_=L}$. For a positive answer to this question we have to show that v-CharPolynomial is hard for $\mathbf{C_=L}$.

Recall that a **GapL**-complete function is PowerElement. The function PowerElement computes the element $(A^m)_{i,j}$, for an $n \times n$ matrix A and for natural numbers m, i, j; note that w.l.o.g. we can restrict PowerElement to the problem of computing $(A^m)_{1,n}$. Thus the problem v-PowerElement is complete for $\mathbf{C_=L}$. We take v-PowerElement as the reference problem to show that v-CharPolynomial is hard for $\mathbf{C_=L}$. The reduction from v-PowerElement to v-CharPolynomial is based on techniques of

Toda [Tod91] and Valiant [Val79a], which show that iterated matrix multiplication is reducible to the determinant. In parts of our presentation we follow the reduction presented in Proposition 2.2 of [ABO99].

Theorem 3.1.3 v-PowerElement $\leq_m^{AC^0}$ v-CharPolynomial.

Proof. Let A be an $n \times n$ matrix and $1 \leq m \leq n$. We will construct a matrix B such that the value $(A^m)_{1,n}$ occurs as one of the coefficients of $\chi_B(x)$.

Interpret A as the representation of a directed bipartite graph G_0 on $2n$ nodes and e edges. That is, the nodes of G_0 are arranged in two columns of n nodes each. In both columns, nodes are numbered from 1 to n. If element $a_{k,l}$ of A is not zero, then there is an edge labeled $a_{k,l}$ from node k in the first column to node l in the second column. The number of non-zero elements in A is exactly e.

Now, take m copies of graph G_0, put them in a sequence and identify each second column of nodes with the first column of the next graph in the sequence. Call the resulting graph G'. Graph G' has $m+1$ columns of nodes, and each column has exactly n nodes. Recall that the weight of a path p in a graph is the product of all labels on the edges belonging to the path p. The crucial observation now is that the element at position $(1, n)$ in A^m is the sum of the weights of all paths in G' from node 1 in the first column to node n in the last column. Call these two nodes s and t, respectively. Add an edge labeled 1 from t to s, and call the resulting graph G. An example for the above construction of G for $A = \left[\begin{smallmatrix} 2 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 3 & 0 \end{smallmatrix}\right]$ is shown in figure 3.1.

Let B be the adjacency matrix of G. So B is an $N \times N$ matrix, where $N = (m+1)n$ is the number of nodes of G. Let the characteristic polynomial of B have the form

$$\chi_B(x) = \det(xI_N - B) = x^N + \sum_{i=0}^{N-1} c_i x^i,$$

where I_N is the $N \times N$ identity matrix. We give two ways how to compute the coefficients c_i in $\chi_B(x)$

1. one way is to use elementary linear transformations and bring the polynomial matrix $xI_N - B$ into triangular block form. Then the characteristic polynomial of B can be computed from the resulting polynomial matrix.

2. a very elegant proof is provided by combinatorial matrix theory from which it is known that the coefficients of the characteristic polynomial can be expressed as cycle covers in the graph G (see e.g. [BR91, CDS80, Zei85, MV97, MV99]).

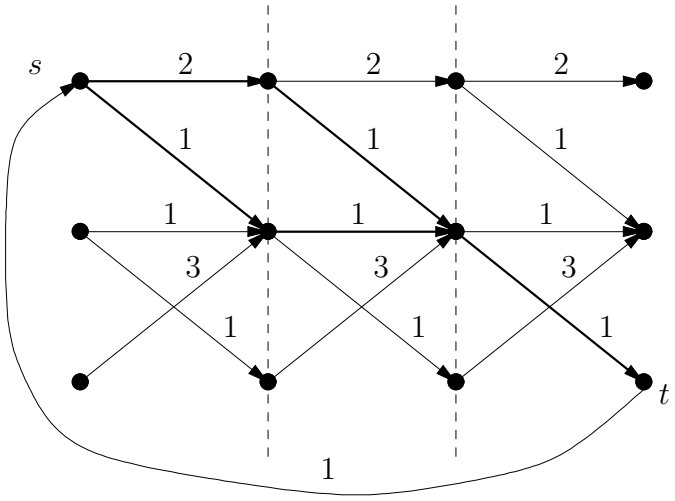

Figure 3.1: *The graph G constructed from matrix A for $m = 3$. The three copies of G_0 are indicated by the dashed lines. The edge labels are the corresponding elements of A. The thicker edges indicate the two paths from s to t. The weights of these two paths sum up to 3, which is the value of $(A^3)_{1,3}$. For the characteristic polynomial of the adjacency matrix B we get $\chi_B(x) = x^{12} - 3x^8$.*

We start by giving the combinatorial argument which is much shorter than the linear-algebraic argument.

The combinatorial way

It is known that, for each i, the coefficient c_i in $\chi_B(x)$ is equal to the sum of the disjoint weighted cycles that cover $N - i$ nodes in G, with appropriate sign (see [BR91] or [CDS80] for more detail). In the graph G, all edges go from a layer to the next layer. The only exception is the edge (t, s). So any cycle in G must use precisely this edge (t, s), and then trace out a path from s to t. Therefore, each cycle in G has exactly the length $m + 1$, and the weighted sum of all these cycles is precisely $(-1)^{m+1}(A^m)_{1,n}$ (for the sign, recall that we consider $xI_N - B$). The sign of the cycle (as a permutation) is $(-1)^m$. Hence,

$$\begin{aligned} c_{N-(m+1)} &= (-1)^{m+1}(-1)^m(A^m)_{1,n} \\ &= -(A^m)_{1,n}, \end{aligned}$$

and all other coefficients must be zero, i.e

$$\chi_B(x) = x^N - ax^{N-(m+1)}, \text{ for } a = (A^m)_{1,n}.$$

The linear-algebraic way

Consider the adjacency matrix B of the graph G. Except for the edge (t, s), graph G is acyclic. Thus we can put the nodes of G in such an order that adjacency matrix B is upper triangular for the first $N-1$ rows with zeros along the main diagonal. The last row of B has 1 in the first position (representing edge (t, s)), and all the other elements are zero.

Now we can write B as a $(m+1) \times (m+1)$ block matrix by

$$B = \left[\begin{array}{c|ccc} & A & & \\ & & \ddots & \\ & & & A \\ \hline L & & & \end{array}\right].$$

Matrix A occurs m-times on the upper sub-diagonal of B. L is the $n \times n$ matrix having 1 at position $(n, 1)$ and 0 elsewhere. All the empty places in B are filled with zero (matrices).

Therefore, $xI_N - B$ has the form

$$xI_N - B = \begin{bmatrix} xI_n & -A & & \\ & \ddots & \ddots & \\ & & xI_n & -A \\ -L & & & xI_n \end{bmatrix}.$$

To compute $\chi_B(x)$ we transform $xI_N - B$ into an upper triangular block matrix. Note that it is already upper triangular except for matrix L in the lower left corner. We want to eliminate this block.

The first step is to multiply the last block row by xI_n, and add to it the first block row multiplied by L (from right). This transforms the last block row into

$$\mathbf{0},\ -AL,\ \mathbf{0},\ \ldots,\ \mathbf{0},\ x^2 I_n.$$

In the second step, we multiply the last block row again by xI_n, and add to it the second block row multiplied by AL (from right). This transforms the last block row into

$$\mathbf{0},\ \mathbf{0},\ -A^2 L,\ \mathbf{0},\ \ldots,\ \mathbf{0},\ x^3 I_n.$$

Continuing that way for m iterations, we bring the last block row into

$$\mathbf{0},\ \ldots,\ \mathbf{0},\ x^{m+1} I_n - A^m L.$$

Let $D(x)$ be the resulting upper triangular matrix, i.e.,

$$D(x) = \mathrm{diag}[xI_n,\ \ldots,\ xI_n,\ x^{m+1} I_n - A^m L].$$

The determinant of $D(x)$ is the product of the determinants of diagonal blocks, i.e.

$$\det(D(x)) = x^{N-n} \det(x^{m+1} I_n - A^m L). \tag{3.3}$$

We compute the determinant of $x^{m+1} I_n - A^m L$. Recall the form of matrix L: the only non-zero element is a 1 in the lower left corner. Therefore, $A^m L$ has the last column of A^m as its first column and 0 elsewhere. Hence, $x^{m+1} I_n - A^m L$ is an $n \times n$ lower triangular matrix with the diagonal

$$x^{m+1} - (A^m)_{1,n},\ x^{m+1},\ \ldots,\ x^{m+1},$$

that has the determinant

$$\det(x^{m+1} I_n - A^m L) = x^{(n-1)(m+1)} (x^{m+1} - a), \tag{3.4}$$

where $a = (A^m)_{1,n}$.

Based on (3.3) and (3.4) we get

$$\det(D(x)) = x^{N-n}\, x^{(n-1)(m+1)} (x^{m+1} - a). \tag{3.5}$$

Note, however, that this is not the same as $\chi_B(x)$ because we changed $\chi_B(x)$ with each multiplication of the last block row by xI_n, and because we did this m times. Therefore,

$$\begin{aligned} \chi_B(x) &= \det(D(x)) / \det(x^m I_n) \\ &= x^{N-n}\, x^{(n-1)(m+1)} (x^{m+1} - a)\, x^{-mn} \\ &= x^N - a x^{N-(m+1)}. \end{aligned}$$

In summary, both methods explicitly yield the coefficients of $\chi_B(x)$ such that

$$(A^m)_{1,n} = a \iff \chi_B(x) = x^N - a x^{N-(m+1)}. \tag{3.6}$$

Since the graph G has been constructed in $\mathbf{AC}^0$, the used reduction is $\mathbf{AC}^0$ many-one. □

By Proposition 3.1.2 and Theorem 3.1.3 we obtain the following corollary.

Corollary 3.1.4
v-CharPolynomial *is complete for* $\mathbf{C_=L}$.

3.2 The invariant factors of a matrix

In this section we investigate the complexity of some problems concerning the invariant factors of a matrix. Recall that the invariant factors $i_1(x), i_2(x), \ldots, i_n(x)$, of an $n \times n$ matrix A are defined by formula 2.9 (see page 19) as follows

$$i_{n-j+1}(x) = \frac{D_j(x)}{D_{j-1}(x)},\ D_0(x) \equiv 1, \quad \text{for } j = 1, 2, \ldots, n,$$

where $D_j(x)$ be the greatest common divisor of all minors of order j of the characteristic matrix $xI - A$. Note that the polynomials $i_1(x), i_2(x), \ldots, i_n(x)$ are invariant under every similarity transformation, i.e. two square matrices of order n are similar if and only if they have the same invariant factor system, or equivalently, if and only if they have the same Smith normal form. Recall that the diagonal polynomial matrix $\mathcal{S}_A(x) = \text{diag}[i_n(x), \cdots, i_2(x), i_1(x)]$ is called the Smith normal form of A.

In computer algebra, the problem of computing the Smith canonical form of a polynomial matrix is widely of interest. Note that computing the invariant factors of a matrix A and computing the Smith normal form of the polynomial matrix $xI - A$ are in fact the same. Polynomial-time algorithms to compute the Smith normal form of an integer matrix have been developed in [Fru77, KB79]. An improvement of these algorithms can be found in [Sto96]. Kaltofen et al. [KS87, KKS90] presented the first $\mathbf{RNC}^2$-algorithm for the Smith normal form of a rational polynomial matrix. An $\mathbf{NC}^2$ algorithm for computing the Smith normal form of a polynomial matrix is given by Villard [Vil97]. Therefore, each invariant factor of an integer matrix can be computed also in $\mathbf{NC}^2$. Since the minimal polynomial of matrix A is the first invariant factor, i.e. $\mu_A(x) = i_1(x)$, the minimal polynomial can be computed in $\mathbf{NC}^2$ as well.

Our approach towards the invariant factors concentrates on the minimal polynomial. We show in this section some new bounds on the complexity of the minimal polynomial and the invariant factor system. Furthermore, we show that the considered logspace counting classes can be characterized by some interesting problems concerning the degree and the constant term of the minimal polynomial.

This section is organized as follows: the minimal polynomial and the invariant factor system are studied in 3.2.1 and 3.2.2, respectively; problems concerning the degree and the constant term of the minimal polynomial are presented in 3.2.3.

3.2.1 Computing and verifying the minimal polynomial

For an $n \times n$ matrix A, let

$$\mu_A(x) = x^m + d_{m-1}x^{m-1} + \cdots + d_1 x + d_0.$$

We define respectively two problems of computing and verifying the minimal polynomial of an integer matrix as follows:

- MinPolynomial
  Input: An $n \times n$ matrix A, and a natural number $1 \leq i \leq n$.
  Output: The i-th coefficient of $\mu_A(x)$.
- v-MinPolynomial $= \{ (A, p(x)) \mid \mu_A(x) = p(x) \}$.

These problems are known to be in $\mathbf{NC}^2$ [Vil97]. In this section, we show that MinPolynomial is in $\mathbf{TC}^0(\mathbf{GapL})$ and hard for $\mathbf{GapL}$, and v-MinPolynomial is in $\mathbf{C_=L} \wedge \mathbf{coC_=L}$ and hard for $\mathbf{C_=L}$.

An algorithm for computing the minimal polynomial

The algorithm for the minimal polynomial below is based on [HJ85], Section 3.3, Problem 5.

Let A be an $n \times n$ integer matrix. Let $p(x) = x^k + p_{k-1}x^{k-1} + \cdots + p_0$ where all p_i are integers. Observe that $p(x)$ is the minimal polynomial of A if and only if

(*i*) $p(A) = A^k + p_{k-1}A^{k-1} + \cdots + p_0 I = \mathbf{0}$, and

(*ii*) $q(A) \neq \mathbf{0}$, for every monic polynomial $q(x)$ of degree $\deg(q(x)) < k$

(see the definition of the minimal polynomial on page 19).

Define vectors $\boldsymbol{a}_i = vec(A^i)$ for $i = 0, 1, 2, \ldots, n$. Recall that $vec(A^i)$ is the n^2-dimensional vector obtained by putting the columns of A^i below each other (see page 16). The equation $p(A) = \mathbf{0}$ above can be equivalently rewritten as

$$\boldsymbol{a}_k + p_{k-1}\boldsymbol{a}_{k-1} + \cdots + p_0\boldsymbol{a}_0 = \mathbf{0}. \tag{3.7}$$

By (3.7), the vectors $\boldsymbol{a}_k, \ldots, \boldsymbol{a}_0$ are linearly dependent. Furthermore, for monic polynomial q of degree l less than k, it follows from the inequation $q(A) \neq \mathbf{0}$ that the vectors $\boldsymbol{a}_l, \ldots, \boldsymbol{a}_0$ are linearly independent.

For each $i = 1, 2, \ldots, n$, we define C_i to be the following $n^2 \times i$ matrix

$$C_i = [\boldsymbol{a}_{i-1} \ \cdots \ \boldsymbol{a}_0].$$

Let $\boldsymbol{x}^{(i)}$ be a variable vector of length i over rational numbers. Consider the following systems of linear equations

$$C_i \boldsymbol{x}^{(i)} = -\boldsymbol{a}_i, \text{ for } i = 1, 2, \ldots, n. \tag{3.8}$$

Obviously, there exist always indices $1 \leq i \leq n$ such that the corresponding systems (3.8) are feasible, and the minimum of them is exactly the degree of the minimal polynomial of A. Let m be the degree of $\mu_A(x)$. Then all the columns of C_m are linearly independent, i.e. $\text{rank}(C_m) = m$, and the columns of C_{m+1} are linearly dependent, i.e. $\text{rank}(C_{m+1}) = m$. Thus, the coefficient vector $\boldsymbol{d} = [d_{m-1}, \ldots, d_0]^T$ of $\mu_A(x)$ is the unique solution of the system of linear equations

$$C_m \boldsymbol{x}^{(m)} = -\boldsymbol{a}_m.$$

Based on the above observations, one could obtain the following algorithm for the minimal polynomial.

MINPOL(A)
1 $\boldsymbol{a}_i \leftarrow vec(A^i)$, for $i = 0, \ldots, n$
  $C_i \leftarrow [\boldsymbol{a}_{i-1} \cdots \boldsymbol{a}_0]$, for $i = 1, \ldots, n$
2 determine m such that $\text{rank}(C_m) = \text{rank}(C_{m+1}) = m$
3 solve the system $C_m \boldsymbol{x}^{(m)} = -\boldsymbol{a}_m$
4 **return** the solution as the coefficients $d_{m-1}, \ldots, d_0$ of $\mu_A(x)$.

Upper bounds

We prove the following theorem.

Theorem 3.2.1 MinPolynomial *is in* $\mathbf{TC}^0(\mathbf{GapL})$.

Proof. Let's analyse Algorithm MINPOL(A).

In step 1, each element of $\boldsymbol{a}_i$, or equivalently, each element of C_i is an element of a power matrix, thus it is computable in **GapL**.

Step 2 can be done in $\mathbf{C_=L} \wedge \mathbf{coC_=L}$ because $\text{rank}(C_m) = m$ and $\text{rank}(C_{m+1}) = m$ are $\mathbf{coC_=L}$- and $\mathbf{C_=L}$-predicate, respectively.

In step 3, note that we have to solve a uniquely feasible system of linear equations with the solution $\boldsymbol{d} = [d_{m-1}, \ldots, d_0]^T$. Define the $m \times m$ matrix B_m and vector $\boldsymbol{b}_m$ of length m by

$$B_m = C_m^T C_m \quad \text{and} \quad \boldsymbol{b}_m = -C_m^T \boldsymbol{a}_m. \tag{3.9}$$

Since C_m has the full column rank, we get $\text{rank}(B_m) = m$, i.e. the matrix B_m is nonsingular. We show that $\boldsymbol{d} = B_m^{-1}\boldsymbol{b}_m$ is the solution of the system of the linear equations in step 3.

Obviously, $C_m^T C_m \boldsymbol{d} = -C_m^T \boldsymbol{a}_m$, i.e. $C_m^T(C_m\boldsymbol{d} + \boldsymbol{a}_m) = \boldsymbol{0}$. Assume for a moment that $C_m\boldsymbol{d} + \boldsymbol{a}_m \neq \boldsymbol{0}$. Then the equality $C_m^T(C_m\boldsymbol{d} + \boldsymbol{a}_m) = \boldsymbol{0}$ can not be true because $\text{rank}(C_m^T) = m$. Therefore, $\boldsymbol{d} = B_m^{-1}\boldsymbol{b_m}$ is the unique solution of linear equation system in step 3.

Expressing

$$B_m^{-1} = \frac{\text{adj}(B_m)}{\det(B_m)}$$

where $\text{adj}(B_m)$ is the adjoint matrix of B_m (see formula (2.3) on page 17), we get

$$\boldsymbol{d} = \frac{\text{adj}(B_m)\ \boldsymbol{b}_m}{\det(B_m)}. \tag{3.10}$$

For any i, all elements of B_i, $\boldsymbol{b}_i$, and $\text{adj}(B_i)$ are computable in **GapL**. Thus $\det(B_i)$ and all elements of $\text{adj}(B_i)\boldsymbol{b}_i$ are computable in **GapL** because **GapL** is closed under composition. Since m is determined in $\mathbf{C_=L} \wedge \mathbf{coC_=L}$, $\det(B_m)$ and all elements of $\text{adj}(B_m)\boldsymbol{b}_m$ are computable in $\mathbf{AC^0(GapL)}$. According to formula (3.10), each element of $\boldsymbol{d}$ is expressed as a division of two integers which are computable in $\mathbf{AC^0(GapL)}$. A breakthrough by Hesse [Hes01] shows that integer division with remainder can be done in $\mathbf{TC^0}$, therefore all elements of $\boldsymbol{d}$ are computable in $\mathbf{TC^0(GapL)}$. Furthermore, note that $\boldsymbol{d}$ is anyway an integer vector.

In summary, MinPolynomial is in $\mathbf{TC^0(GapL)}$. □

Remark 3.2.2 There is another method to compute the minimal polynomial. Let's sketch briefly its idea. The minimal polynomial will be computed by the following formula

$$\mu_A(x) = \frac{\chi_A(x)}{D_{n-1}(x)}, \tag{3.11}$$

where $D_{n-1}(x)$ is the greatest common divisor of all n^2 minors of order $n-1$ in the polynomial matrix $xI - A$. To do so, we use Algorithm 15.4 and 15.1 in [IK93] for computing $D_{n-1}(x)$, and thereafter, Algorithm 15.2 in [IK93] for computing the polynomial division presented in (3.11). By this means, the idea is very simple, but it is more tricky to show that MinPolynomial is in $\mathbf{TC^0(GapL)}$ than in the proof of Theorem 3.2.1.

Remark 3.2.3 It is important to make a remark about matrix inversion. For an invertible integer matrix A, since the inverse A^{-1} is uniquely equal to

$\mathrm{adj}(A)/\det(A)$, the elements of A^{-1} seem not to be division-free. Even in the case when A^{-1} is an integer matrix, it is not clear how to develop a division-free algorithm for the inverse A^{-1}. Therefore, it seems to be incorrect in saying that matrix inversion is in **GapL**. With respect to MinPolynomial, the situation is the same. As we have seen, although the coefficients of $\mu_A(x)$ are integers (A is an integer matrix), we don't know how to get a division-free algorithm for computing the minimal polynomial. Therefore, the statement in [HT01, HT03a] that $\mathbf{AC}^0(\mathbf{GapL})$ is an upper bound for MinPolynomial is only flawless if there is a division-free procedure for solving the system of linear equations in step 3 of Algorithm MinPol. Let's hope it goes well!

We consider the *verification version* of the minimal polynomial. Recall that v-MinPolynomial is the problem of deciding whether a given monic integer polynomial is the minimal polynomial of a given integer matrix A. We show the following corollary.

Corollary 3.2.4 v-MinPolynomial *is in* $\mathbf{C_=L} \wedge \mathbf{coC_=L}$.

Proof. To verify the minimal polynomial we can simplify Algorithm MinPol presented on page 42 as follows:

v-MINPOL$(A, d_{m-1}, \ldots, d_0)$
1 $\boldsymbol{a}_i \leftarrow vec(A^i)$, for $i = 0, \ldots, m$
2 **if** $\boldsymbol{a}_{m-1}, \ldots, \boldsymbol{a}_1, \boldsymbol{a}_0$ are linearly independent, **and**
$\boldsymbol{a}_m + d_{m-1}\boldsymbol{a}_{m-1} + \cdots + d_0\boldsymbol{a}_0 = \mathbf{0}$
3 **then** accept **else** reject.

Since in step 1 all the elements of vectors $\boldsymbol{a}_i$ are computable in **GapL**, the equality in step 2 can be verified in $\mathbf{C_=L}$. For checking linear independence in step 2, let B_m be the symmetric $m \times m$ matrix defined by (3.9), i.e.,

$$B_m = [\boldsymbol{a}_{m-1} \ \cdots \ \boldsymbol{a}_1 \ \boldsymbol{a}_0]^T [\boldsymbol{a}_{m-1} \ \cdots \ \boldsymbol{a}_1 \ \boldsymbol{a}_0].$$

Then the vectors $\boldsymbol{a}_{m-1}, \ldots, \boldsymbol{a}_1, \boldsymbol{a}_0$ are linearly independent if B_m is nonsingular. Since each element of B_m is computable in **GapL**, so is the determinant of B_m (see Corollary 2.2.7). Therefore, $\det(B_m) \neq 0$ is a $\mathbf{coC_=L}$-predicate.

In summary, v-MinPolynomial can be presented as a $\mathbf{C_=L}$-predicate in conjunction with a $\mathbf{coC_=L}$-predicate, i.e. v-MinPolynomial is in $\mathbf{C_=L} \wedge \mathbf{coC_=L}$. □

Lower bounds

Lower bounds on the complexity of MinPolynomial and v-MinPolynomial are given by showing the following theorem.

Theorem 3.2.5 PowerElement $\leq_m^{AC^0}$ MinPolynomial.

Proof. We reconsider the $N \times N$ matrix B in the proof of Theorem 3.1.3 (see page 36). Recall that the characteristic polynomial of B is as follows:

$$\chi_B(x) = x^N - ax^{N-(m+1)}, \text{ for } a = (A^m)_{1,n}.$$

For the reduction PowerElement $\leq_m^{AC^0}$ MinPolynomial, we show that the value $(A^m)_{1,n}$ is one of the coefficients of the minimal polynomial of B. In particular, we show $\mu_B(x) = x^{2m+2} - ax^{m+1}$.

Let $D_{N-1}(x)$ be the greatest common divisor of all minors of order $N-1$ in $(xI_N - B)$. Observe that the sub-matrix obtained by deleting the first row and the first column of $(xI_N - B)$ is a triangular matrix having purely x on its diagonal. Therefore, its determinant is equal to x^{N-1}. It follows that $D_{N-1}(x) = x^l$ for some $l \geq 0$. Putting this in (3.11) we get

$$\mu_B(x) = \frac{\chi_B(x)}{D_{N-1}(x)} = x^{N-l} - ax^{N-(m+1)-l}, \text{ for some } l \geq 0.$$

By defining polynomials $p_k(x) = x^{(m+1)+k} - ax^k$, for $0 \leq k \leq N-(m+1)$, we shall show $\mu_B(x) = p_{m+1}(x)$. In order to prove this, we show that the following two statements hold:

(i) $p_{m+1}(B) = \mathbf{0}$, and

(ii) $p_k(B) \neq \mathbf{0}$ for all $k < m+1$.

Our first step is to compute explicitly the powers B^i, for $i = 2, \ldots, m+1$. Elementary calculations yield

$$B^2 = \left[\begin{array}{cc|c} & & A^2 \\ & & \ddots \\ & & A^2 \\ \hline AL & & \\ & LA & \end{array}\right], \quad B^3 = \left[\begin{array}{cc|c} & & A^3 \\ & & \ddots \\ & & A^3 \\ \hline A^2L & & \\ & ALA & \\ & & LA^2 \end{array}\right], \quad \cdots.$$

For each $i \leq m$, B^i can be generalized by

$$
\begin{array}{rcl}
 & & \begin{array}{cc} i & i+1 \\ \downarrow & \downarrow \end{array} \\
B^i & = & \left[\begin{array}{c|c}
 & A^i \\
 & \ddots \\
 & A^i \\
\hline
A^{i-1}L & \\
A^{i-2}LA & \\
\ddots & \\
LA^{i-1} &
\end{array}\right]
\begin{array}{l}
\leftarrow 1 \\
 \\
\leftarrow m+1-i \\
\leftarrow m+2-i \\
 \\
 \\
\leftarrow m+1
\end{array}
\end{array}
$$

Furthermore, the matrix B^{m+1} is a diagonal block matrix having $A^{m+1-i}LA^{i-1}$ as its i-th diagonal block, for each $1 \leq i \leq m+1$. Hence, matrix B^{2m+2} $(= (B^{m+1})^2)$ is also a diagonal block matrix where its i-th diagonal block is equal to the square of the i-th diagonal block of B^{m+1}:

$$(A^{m+1-i}LA^{i-1})^2 = A^{m+1-i}LA^mLA^{i-1}.$$

Let's observe that the factor LA^mL occurring in each diagonal block of B^{2m+2} is of an easy form, in particular, $LA^mL = aL$. It follows that on the matrix B^{2m+2} we can pull the factor a in front of the matrix and what remains is again B^{m+1}, i.e., $B^{2m+2} = aB^{m+1}$. Therefore,

$$p_{m+1}(B) = B^{2m+2} - aB^{m+1} = \mathbf{0}.$$

It remains to prove (ii) (see page 45): $p_k(B) = B^{m+1+k} - aB^k \neq \mathbf{0}$, for all $k \leq m$. Note that it is sufficient to prove this for $k = m$, because, for some k, $p_k(B) = \mathbf{0}$ implies $p_{k+1}(B) = \mathbf{0}$.

Assume for a moment that $p_m(B) = B^{2m+1} - aB^m = \mathbf{0}$. Observe that the blocks at position $(1, m+1)$ in B^{2m+1} and B^m are A^mLA^m and A^m, respectively. Hence, $A^mLA^m = aA^m$. The latter equality implies $\operatorname{rank}(A^mLA^m) = \operatorname{rank}(aA^m)$. Due to Lemma 3.2.7 below we can assume that A is nonsingular. Therefore,

$$
\begin{array}{rcl}
\operatorname{rank}(A^mLA^m) & = & \operatorname{rank}(L) = 1, \\
\operatorname{rank}(aA^m) & = & \begin{cases} n, \text{ for } a \neq 0 \\ 0, \text{ otherwise.} \end{cases}
\end{array}
$$

Obviously, since $\operatorname{rank}(A^mLA^m) \neq \operatorname{rank}(aA^m)$, $p_m(B) \neq \mathbf{0}$.

In summary, we get $\mu_B(x) = x^{2m+2} - ax^{m+1}$ where $a = (A^m)_{1,n}$. As mentioned in the proof of Theorem 3.1.3, the used reduction is $\mathbf{AC}^0$ many-one. □

Corollary 3.2.6 MinPolynomial *is hard for* **GapL**.
v-MinPolynomial *is hard for* $\mathbf{C_=L}$.

To complete the proof of Theorem 3.2.5 we show the following lemma.

Lemma 3.2.7 *Given an $n \times n$ matrix A and $m \geq 1$, there is a nonsingular upper triangular $p \times p$ matrix C constructed from A and m such that $(C^m)_{1,p} = (A^m)_{1,n}$.*

Proof. Define C to be the $(m+1) \times (m+1)$ block matrix

$$C = \begin{bmatrix} I & A & & \\ & \ddots & \ddots & \\ & & I & A \\ & & & I \end{bmatrix},$$

where I is the $n \times n$ identity matrix. Then C is nonsingular and C^m has the following form

$$C^m = \begin{bmatrix} I & mA & mA^2 & \cdots & mA^{m-1} & A^m \\ & I & mA & \cdots & mA^{m-2} & mA^{m-1} \\ & & \ddots & \ddots & \vdots & \vdots \\ & & & \ddots & mA & mA^2 \\ & & & & I & mA \\ & & & & & I \end{bmatrix},$$

and, for $p = (m+1)n$, we have $(C^m)_{1,p} = (A^m)_{1,n}$. □

3.2.2 The invariant factor system of a matrix

Let's define the problem of computing the invariant factor system of an integer matrix as follows.

- InvSystem
  Input: An $n \times n$ matrix A, and two natural numbers $1 \leq k, j \leq n$.
  Output: The k-th coefficient of the j-th invariant factor $i_j(x)$ of A.

Note that InvSystem is in $\mathbf{NC}^2$ [Vil97].

By Corollary 3.2.6, MinPolynomial is hard for **GapL**, therefore InvSystem is hard for **GapL** as well.

Corollary 3.2.8 InvSystem *is hard for* **GapL**.

Furthermore, we define the problem of verifying the invariant factor system of an integer matrix by

- v-InvSystem =
  $\{ (A, i_1(x), \ldots, i_n(x)) \mid \mathcal{S}_A(x) = \mathrm{diag}[i_n(x), i_{n-1}, \ldots, i_1(x)] \}$.

Theorem 3.2.9 v-InvSystem *is in* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Proof. Let $i_1(x), \ldots, i_n(x)$ be n given monic polynomials, and let A be an $n \times n$ matrix. For each non-constant polynomial of the given polynomials, we construct a companion matrix corresponding to it. Let's denote the diagonal block matrix of all constructed companion matrices by D. Then the polynomials $i_1(x), \ldots, i_n(x)$ are the invariant factors of A if and only if A is similar to D. Since testing similarity can be done in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ [ST98], v-InvSystem is in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ as well. □

By Corollary 3.2.8, InvSystem is hard for **GapL**. Unfortunately, from this result one can not directly justify that $\mathbf{C}_=\mathbf{L}$ is a lower bound for v-InvSystem because the latter problem requires to verify *all* invariant factors of a given matrix. However, we show the following theorem.

Theorem 3.2.10 v-PowerElement $\leq_m^{\mathbf{AC}^0}$ v-InvSystem.

Proof. We continue with the setting in the proof of Theorem 3.2.5 (see page 45), in particular, with the matrix B of order N. Our goal is to determine explicitly the invariant factor system of B. From the proof of Theorem 3.2.5, we have already received $i_1(x) = \mu_B(x) = x^{2m+2} - ax^{m+1}$, where $a = (A^m)_{1,n}$. It remains to compute the invariant factors $i_2(x), \ \ldots, \ i_N(x)$ of B.

Recall from the proof of Theorem 3.2.5 that $D_{N-1}(x) = x^{N-(2m+2)}$. According to the expression $D_{N-1}(x) = i_2(x) \cdots i_N(x)$, each of the invariant factors $i_2(x), \ldots, i_N(x)$ is of the form x^j, for some $j \geq 0$.

Since the non-constant invariant factors of the form x^j are already elementary divisors (see page 21), it is sufficient to determine all elementary divisors of B.

Define g_j to be the *number of occurrences of the elementary divisor* x^j, and let r_j denote the rank of B^j. The numbers g_j can be determined from the rank values r_j by the following formula (see [Gan77a], Chapter VI, page 155).

$$g_j = r_{j-1} + r_{j+1} - 2r_j, \text{ for } j = 1, \ldots, t, \tag{3.12}$$

where $r_0 = N$ and t is the smallest index satisfying the condition

$$r_{t-1} > r_t = r_{t+1}.$$

Fortunately, we can compute all the values r_j as follows.

By Lemma 3.2.7, we may assume that the input matrix A to v-PowerElement is nonsingular, i.e. $\text{rank}(A) = n$. Therefore, we have $\text{rank}(A^j) = n$, for every $j \geq 1$.

Consider the general form of B^j (see page 46), for $1 \leq j \leq m$. The rank of B^j is equal to the sum of all ranks of the matrices on the lower and upper sub-diagonals. The following two observations are useful for computing $\text{rank}(B^j)$.

(i) Each of the $m + 1 - j$ blocks on the upper sub-diagonal of B^j has the form A^j.

(ii) Each of the j blocks on the lower sub-diagonal of B^j has the form $A^{j-k}LA^{k-1}$, for $1 \leq k \leq j$, where $\text{rank}(A^{j-k}LA^{k-1}) = \text{rank}(L) = 1$.

Therefore, we get

$$\text{rank}(B^j) = (m + 1 - j)n + j, \text{ for } 1 \leq j \leq m.$$

Similarly, we compute $\text{rank}(B^{m+1})$ and $\text{rank}(B^{m+2})$, and we get

$$\text{rank}(B^{m+1}) = \text{rank}(B^{m+2}) = m + 1.$$

According to formula (3.12), it is obvious to see that $t = m + 1$ because $r_m > r_{m+1} = r_{m+2}$. Therefore,

$$r_j = \begin{cases} (m + 1 - j)n + j, & \text{for } j = 1, \ldots, m, \\ m + 1, & \text{for } j = m + 1, m + 2. \end{cases}$$

Plugging the values r_j into formula (3.12), we obtain

$$g_i = \begin{cases} N - n(m + 1), & \text{for } i = 1, \\ 0, & \text{for } i = 2, \ldots, m, \\ n - 1, & \text{for } i = m + 1. \end{cases} \tag{3.13}$$

From (3.13) we can deduce the invariant factors, there are

(a) $n - 2$ factors x^{m+1} (note that one of the $n - 1$ elementary divisors x^{m+1} occurs in $i_1(x)$),

(b) $N - n(m + 1)$ factors x, and

(c) $nm + 1$ factors 1.

In summary, $(A^m)_{1,n} = a$ if and only if the invariant factors of A are as follows:

$$i_k(x) = \begin{cases} x^{2m+2} - ax^{m+1}, & \text{for } k = 1 \\ x^{m+1}, & \text{for } k = 2, \ldots, n-1, \\ x, & \text{for } k = n, \ldots, N - nm - 1, \\ 1, & \text{for } k = N - nm, \ldots, N. \end{cases}$$

□

3.2.3 More about the minimal polynomial

The main contribution of this section is in obtaining some new characterizations for the logspace counting classes $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$, $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$, and $\mathbf{C}_=\mathbf{L}$, by investigating the complexity of problems concerning the degree and the constant term of the minimal polynomial of a matrix.

For square matrices A and B, and for a natural number m, we define the problem of

- computing the k-th bit of $\deg(\mu_A(x))$ by
  DEGMINPOL $= \{ (A, k, b) \mid$ the k-th bit of $\deg(\mu_A(x))$ is $b \}$,
- verifying the value $\deg(\mu_A(x))$ by
  V-DEGMINPOL $= \{ (A, m) \mid \deg(\mu_A(x)) = m \}$,
- deciding whether the value $\deg(\mu_A(x))$ is at most m by
  DEGMINPOL$_\leq$ $= \{ (A, m) \mid \deg(\mu_A(x)) \leq m \}$,
- deciding whether two minimal polynomials have the same degree by
  EQDEGMINPOL $= \{ (A, B) \mid \deg(\mu_A(x)) = \deg(\mu_B(x)) \}$,
- deciding whether two minimal polynomials are equal by
  EQMINPOLYNOMIAL $= \{ (A, B) \mid \mu_A(x) = \mu_B(x) \}$,
- computing the value $\mathrm{ct}(\mu_A(x))$ to be the function CTMINPOL(A), and
- deciding whether two minimal polynomials have the same constant term by
  EQCTMINPOL $= \{ (A, B) \mid \mathrm{ct}(\mu_A(x)) = \mathrm{ct}(\mu_B(x)) \}$.

An aim of this section is to show that the degree of the minimal polynomial is computationally equivalent to the rank of a matrix. Recall that the complexity of matrix rank has been studied in [ABO99], where it was shown that

- RANK is complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$,
- v-RANK is complete for $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$, and
- $\text{RANK}_\leq = \{ (A, r) \mid \text{rank}(A) \leq r \}$ is complete for $\mathbf{C}_=\mathbf{L}$.

We will show that these results can be extended respectively for the sets DEGMINPOL, v-DEGMINPOL, and $\text{DEGMINPOL}_\leq$.

Furthermore, we show that EQMINPOLYNOMIAL, EQDEGMINPOL, and EQCTMINPOL are complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. It follows from the latter result that CTMINPOL can not be computable in **GapL** unless the $\mathbf{C}_=\mathbf{L}$-hierarchy collapses to $\mathbf{C}_=\mathbf{L}$.

Upper bounds

For $\text{DEGMINPOL}_\leq$ and v-DEGMINPOL, we show the following proposition.

Proposition 3.2.11
(1) $\text{DEGMINPOL}_\leq$ *is in* $\mathbf{C}_=\mathbf{L}$.
(2) v-DEGMINPOL *is in* $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$.

Proof. Let A be given $n \times n$ matrix, and let m be a given natural number. As stated in Section 3.2.1 (see page 42), we define

$$\begin{aligned} C_j &= [\boldsymbol{a}_0\ \boldsymbol{a}_1\ \cdots\ \boldsymbol{a}_{j-1}], \text{ and} \\ B_j &= C_j^T\ C_j, \text{ for } j = 1, \ldots, n. \end{aligned}$$

Let $\deg(\mu_A(x)) = k \leq n$. Observe that the matrices $C_k, \ldots, C_n$ and $B_k, \ldots, B_n$ have the same rank which is equal to k, i.e.

$$\text{rank}(B_n) = \deg(\mu_A(x)) = k.$$

Let

$$\chi_{B_n}(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0.$$

Since B_n is a symmetric matrix, we get $\text{rank}(B_n) = n - l$, for the smallest index l so that $c_l \neq 0$. It follows that

$$\deg(\mu_A(x)) = n - l.$$

Therefore, we get the following equivalences

$$\begin{aligned} \deg(\mu_A(x)) \leq m &\iff c_0 = c_1 = \cdots = c_{n-m} = 0, \\ \deg(\mu_A(x)) = m &\iff c_0 = c_1 = \cdots = c_{n-m} = 0 \textbf{ and } c_{n-m+1} \neq 0. \end{aligned}$$

Since all elements of B_n are computable in **GapL**, the coefficients c_i of $\chi_{B_n}(x)$ are also computable in **GapL** because **GapL** is closed under composition. Moreover, testing whether $c_i = 0$ simultaneously for multiple values of i can be done in $\mathbf{C_=L}$ because $\mathbf{C_=L}$ is closed under conjunction (see Proposition 2.2.9). It follows that the sets DegMinPol$_{\leq}$ and v-DegMinPol are in $\mathbf{C_=L}$ and $\mathbf{C_=L} \wedge \mathbf{coC_=L}$, respectively. □

For EqDegMinPol, DegMinPol, EqMinPolynomial, and EqCTMinPol we show the following proposition.

Proposition 3.2.12 EqDegMinPol, DegMinPol, EqMinPolynomial, *and* EqCTMinPol *are in* $\mathbf{AC^0(C_=L)}$.

Proof. Let A and B be matrices of order n and p, respectively.

The polynomials $\mu_A(x)$ and $\mu_B(x)$ have the same degree if and only if there is a number $m \in \{1, \ldots, \min\{n, p\}\}$ such that $\deg(\mu_A(x)) = m$ and $\deg(\mu_B(x)) = m$. By Proposition 3.2.11, verifying the degree of the minimal polynomial can be done in $\mathbf{C_=L} \wedge \mathbf{coC_=L}$, therefore EqDegMinPol is in $\mathbf{AC^0(C_=L)}$.

Let (A, k, b) be an input to DegMinPol and let n be the order of A. A straightforward approach to obtain an upper bound for DegMinPol might be to use the fact that

$$(A, k, b) \in \text{DegMinPol} \iff (B_n, k, b) \in \text{Rank}, \tag{3.14}$$

where B_n is defined as in the proof of Proposition 3.2.11 (see page 51). However, every element of B_n seems to require a **GapL**-computation because $B_n = C_n^T C_n$ and the elements of C_n are computable in **GapL**. Therefore the right-hand side of (3.14) verifies the k-th bit of the rank of matrix B_n computable in **GapL**. Using the fact that Rank is complete for $\mathbf{AC^0(C_=L)}$, we can argue that (3.14) can be done in $\mathbf{AC^0(C_=L)}$. For simplicity, we explain another way as follows.

We construct an $\mathbf{AC^0}$-circuit with oracle gates from $\mathbf{C_=L}$ for DegMinPol: for each number $m \in \{1, \ldots, n\}$ whose k-th bit is b we construct an $\mathbf{AC^0(C_=L)}$ circuit to decide whether $\deg \mu_A(x) = m$. The final output is the disjunction of these circuits. It follows that DegMinPol is in $\mathbf{AC^0(C_=L)}$.

Due to formula (3.10), the coefficient vector $\boldsymbol{d}_A$ of $\mu_A(x)$ can be expressed as follows.

$$\boldsymbol{d}_A = \frac{\text{adj}(B_m)\ \boldsymbol{b}_m}{\det(B_m)},$$

where $m = \deg(\mu_A(x))$, and B_m and $\boldsymbol{b}_m$ are defined in terms of powers of A (see page 43). Recall from the proof of Theorem 3.2.1 that, for each i, $\det(B_i)$ and all elements of $\text{adj}(B_i)\boldsymbol{b}_i$ are **GapL**-computable.

Similarly, we can express the coefficient vector $\boldsymbol{d}_B$ as in (3.15). It follows that in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ we can compare $\boldsymbol{d}_A$ with $\boldsymbol{d}_B$. Therefore, EqMinPolynomial and EqCTMinPol are in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. □

Lower bounds

It was shown in [ABO99] that $\text{Rank}_{\leq}$ is hard for $\mathbf{C}_=\mathbf{L}$ and that v-Rank is hard for $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$. We show the following theorem.

Theorem 3.2.13
(1) $\text{DegMinPol}_{\leq}$ *is hard for* $\mathbf{C}_=\mathbf{L}$.
(2) v-DegMinPol *is hard for* $\mathbf{C}_=\mathbf{L} \wedge \mathbf{coC}_=\mathbf{L}$.

Proof. (1) For the first part of the theorem we show that v-PowerElement is reducible to $\text{DegMinPol}_{\leq}$.

Let an $n \times n$ matrix A and an integer $m \geq 1$ be given as input to v-PowerElement. Recall that, for v-PowerElement, one has to decide whether $(A^m)_{1,n} = 0$.

As already seen in the proof of Theorem 3.2.5, one can construct matrix B such that

$$\mu_B(x) \;=\; x^{2m+2} - ax^{m+1}, \text{ where } a = (A^m)_{1,n}.$$

Define C to be the companion matrix of the polynomial x^{2m+2}, i.e. C is the following $(2m+2) \times (2m+2)$ matrix

$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ & & \cdots\cdots\cdots & & \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

It is known that $\chi_C(x) = \mu_C(x) = x^{2m+2}$ (see (2.12) on page 20).

Define the diagonal *block matrix*

$$D = \begin{bmatrix} B & \mathbf{0} \\ \mathbf{0} & C \end{bmatrix}.$$

It is known that the minimal polynomial of D is the *least common multiple* (for short: lcm) of $\mu_B(x)$ and $\mu_C(x)$ (see [HJ85], Section 3.3, exercise 8). Using this

fact we get

$$\begin{aligned}\mu_D(x) &= \mathrm{lcm}\{x^{m+1}(x^{m+1}-a), x^{2m+2}\} \\ &= \begin{cases} x^{2m+2}, & \text{for } a=0, \\ x^{2m+2}(x^{m+1}-a), & \text{for } a\neq 0. \end{cases}\end{aligned}$$

Therefore,

$$a = (A^m)_{1,n} = 0 \iff \deg(\mu_D(x)) = 2m+2.$$

Note that the used reduction is $\mathbf{AC}^0$ many-one.

(2) To show the second part of the theorem, we construct a reduction from an arbitrary language L in $\mathbf{C}_{=}\mathbf{L} \wedge \mathbf{coC}_{=}\mathbf{L}$ to the set v-DegMinPol.

Since v-PowerElement is complete for $\mathbf{C}_{=}\mathbf{L}$, in logspace we can compute matrices A_1 and A_2 of order n_1 and n_2, respectively, and integers $m, l \geq 1$, such that for every w:

$$w \in L \iff (A_1^m)_{1,n_1} = 0 \text{ and } (A_2^l)_{1,n_2} \neq 0. \tag{3.15}$$

Due to Lemma 3.2.15 below we may assume w.l.o.g. that $m > l$.

Let $a_1 = (A_1^m)_{1,n_1}$ and $a_2 = (A_2^l)_{1,n_2}$. As explained in the first part of the proof, in logspace we can compute matrices B_1 and B_2 such that

$$\begin{aligned}\mu_{B_1}(x) &= x^{2m+2} - a_1 x^{m+1}, \\ \mu_{B_2}(x) &= x^{2l+2} - a_2 x^{l+1}.\end{aligned}$$

Define the matrix

$$D = \begin{bmatrix} B_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & C \end{bmatrix},$$

where C is the companion matrix of x^{2m+2}. Then the minimal polynomial of D can be computed as follows

$$\begin{aligned}\mu_D(x) &= \mathrm{lcm}\{\mu_{B_1}(x),\ \mu_{B_2}(x),\ \mu_C(x)\} \\ &= \mathrm{lcm}\{x^{m+1}(x^{m+1}-a_1),\ x^{l+1}(x^{l+1}-a_2),\ x^{2m+2}\} \\ &= x^{2m+2}\mathrm{lcm}\{x^{m+1}-a_1,\ x^{l+1}-a_2\}.\end{aligned}$$

For $m > l$, we get

$$\deg(\mu_D(x)) = \begin{cases} 2m+l+3, & \text{for } a_1=0,\ a_2\neq 0, \\ 3m+3, & \text{for } a_1\neq 0,\ a_2=0, \\ 2m+2, & \text{for } a_1=0,\ a_2=0, \\ 3m+3+r, & \text{for } a_1\neq 0,\ a_2\neq 0, \text{ where } r>0. \end{cases} \tag{3.16}$$

Based on (3.15) and (3.16), we have for every w:

$$\begin{aligned} w \in L &\iff a_1 = 0 \text{ and } a_2 \neq 0 \\ &\iff \deg(\mu_D(x)) = 2m + l + 3. \end{aligned}$$

Therefore, v-DegMinPol is hard for $\mathbf{C}_{=}\mathbf{L} \wedge \mathbf{coC}_{=}\mathbf{L}$. □

By Proposition 3.2.11 and Theorem 3.2.13 we obtain the following corollary.

Corollary 3.2.14
(1) DegMinPol$_{\leq}$ *is complete for* $\mathbf{C}_{=}\mathbf{L}$.
(2) DegMinPol$_{=}$ *is complete for* $\mathbf{C}_{=}\mathbf{L} \wedge \mathbf{coC}_{=}\mathbf{L}$.

The following lemma completes the proof of Theorem 3.2.13.

Lemma 3.2.15 *Given an $n \times n$ matrix A and $m \geq 1$, there is a matrix $\widetilde{A}$ of order $p = n(mk+1)$ such that $(A^m)_{1,n} = (\widetilde{A}^{km})_{1,p}$, for any $k \geq 1$.*

Proof. For a number k, define the following $(mk+1) \times (mk+1)$ block matrix

$$\widetilde{A} = \begin{bmatrix} \mathbf{0} & A & & & & & & & \\ & \mathbf{0} & I & & & & & & \\ & & \ddots & \ddots & & & & & \\ & & & \mathbf{0} & I & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & \ddots & \ddots & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & & \mathbf{0} & A & & \\ & & & & & & \mathbf{0} & I & \\ & & & & & & & \ddots & \ddots \\ & & & & & & & & \mathbf{0} & I \\ & & & & & & & & & \mathbf{0} \end{bmatrix}.$$

Each block of $\widetilde{A}$ is a matrix of order n. In the first block super-diagonal of $\widetilde{A}$ the pattern of an A followed by $(k-1)$ times I is repeated m times. All the other blocks are $\mathbf{0}$.

An elementary calculation shows that $\widetilde{A}^{mk}$ has A^m as its upper right block at position $(1, mk+1)$, and all other blocks are $\mathbf{0}$, i.e.

$$\widetilde{A}^{mk} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & A^m \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Therefore, we have $(A^m)_{1,n} = (\widetilde{A}^{km})_{1,p}$. □

Continuously, we prove the following theorem.

Theorem 3.2.16 EqMinPolynomial, EqDegMinPol, DegMinPol, *and* EqCTMinPol *are hard for* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Proof. FSLE (see page 32) was shown in [ABO99] to be complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. We show that FSLE is reducible to the considered sets.

Let $(A, \boldsymbol{b})$ be an input to FSLE. Define the symmetric $(m+n) \times (m+n)$ matrix

$$B = \begin{bmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{bmatrix}$$

and the vector $\boldsymbol{c} = (\boldsymbol{b}^T, \mathbf{0})^T$ of length $m+n$. We define further the following two $(m+n+1) \times (m+n+1)$ matrices

$$C = \begin{bmatrix} B & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \text{ and } D = \begin{bmatrix} B & \boldsymbol{c} \\ \mathbf{0} & 0 \end{bmatrix}.$$

Let $\lambda_1, \ldots, \lambda_k$ be distinct eigenvalues of C. It will be useful later on to observe that

(I) C is a symmetric matrix. Therefore, C is diagonalizable, its elementary divisors have the form $(x-\lambda_i)$, and $\mu_C(x) = (x-\lambda_1)\cdots(x-\lambda_k)$ (see [HJ85], Section 3.3, Theorem 3.3.6 and Corollary 3.3.8).

(II) C and D are singular matrices. They have the same characteristic polynomial: $\chi_C(x) = \chi_D(x) = x\ \chi_B(x)$, and consequently they have the same eigenvalues. It follows that $\deg(\mu_C(x)) \leq \deg(\mu_D(x))$, and the elementary divisors of D have the form $(x-\lambda_i)^{t_i}$, for some $t_i \geq 1$.

In order to show that FSLE is reducible to EqMinPolynomial, EqDegMinPol, DegMinPol, and EqCTMinPol, we prove the following equivalences

$$\begin{aligned}
(A, \boldsymbol{b}) \in \text{FSLE} &\iff (B, \boldsymbol{c}) \in \text{FSLE} && (3.17)\\
&\iff C \text{ is similar to } D && (3.18)\\
&\iff D \text{ is diagonalizable} && (3.19)\\
&\iff \mu_C(x) = \mu_D(x) && (3.20)\\
&\iff \deg(\mu_C(x)) = \deg(\mu_D(x)) && (3.21)\\
&\iff \deg(\mu_D(x)) \text{ is odd} && (3.22)\\
&\iff \mathrm{ct}(\mu_{C_\alpha}(x)) = \mathrm{ct}(\mu_{D_\alpha}(x)), && (3.23)
\end{aligned}$$

where $C_\alpha = C + \alpha I$ and $D_\alpha = D + \alpha I$ for an appropriate positive integer α to be chosen later.

Note that equivalences (3.18) and (3.19) will be used later for showing Corollary 3.3.3 and Theorem 3.3.4 in Section 3.3.

Equivalence (3.17). The equivalence holds because the system $A^T\boldsymbol{x} = \mathbf{0}$ is always feasible.

Equivalence (3.18). Consider the case where the system $B\boldsymbol{x} = \boldsymbol{c}$ is feasible. Let $\boldsymbol{x}_0$ be a solution of the system. Define the $(m+n+1) \times (m+n+1)$ matrix T by

$$T = \begin{bmatrix} I & \boldsymbol{x}_0 \\ \mathbf{0} & -1 \end{bmatrix}.$$

Obviously, T is nonsingular and the equalities $CT = TD = D$ hold. Thus, C is similar to D.

Conversely, if the above system is not feasible, then C and D have different ranks. Hence they can not be similar.

Equivalence (3.19). Based on observation (I) (see page 56), matrix C is similar to a diagonal matrix C'. If C is similar to D, then D is similar to C' because the similarity relation is transitive. Hence D is diagonalizable.

Conversely, if D is diagonalizable, then all elementary divisors of D are linear. Based on observation (II) (see page 56), C and D have the same eigenvalues. It follows that C and D must have the same system of elementary divisors. Thus they are similar.

Equivalence (3.20). If C is similar to D, then clearly $\mu_C(x) = \mu_D(x)$.

Conversely, if $\mu_C(x) = \mu_D(x)$, then $\mu_D(x)$ contains only linear irreducible factors, because $\mu_C(x)$ has this property due to observation (I). Therefore, D is diagonalizable (see [HJ85], Section 3.3, Corollary 3.3.10).

Equivalence (3.21). Based on observation (II) (see page 56), we have $\deg(\mu_C(x)) \leq \deg(\mu_D(x))$. These degrees are *equal* if and only if every root of $\mu_D(x)$ has multiplicity 1. The latter holds if and only if D is diagonalizable.

Equivalence (3.22). Let the distinct non-zero eigenvalues of the matrix A^TA be $\delta_1, \delta_2, \ldots, \delta_l$ (note that these numbers are positive). Then the distinct eigenvalues of C are

$$-\sqrt{\delta_l}\,, -\sqrt{\delta_{l-1}}\,, \ldots, -\sqrt{\delta_1}\,, 0\,, \sqrt{\delta_1}\,, \ldots, \sqrt{\delta_{l-1}}\,, \sqrt{\delta_l} \tag{3.24}$$

(see [HJ91], Chapter 3). The matrix C is singular because its last row is $\mathbf{0}$. Therefore, the number of all distinct eigenvalues of C is exactly $2l+1$. Suppose $\deg(\mu_C(x)) = k$. Then $k = 2l+1$, k is always odd.

To prove the claim, we show that $\deg(\mu_D(x)) \in \{k,\ k+1\}$. By observation (II), $\deg(\mu_D(x)) \geq k$. Hence, it is sufficient to show $\deg(\mu_D(x)) \leq k+1$.

For each $i \geq 0$, we have

$$C^i = \begin{bmatrix} B^i & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}, \tag{3.25}$$

$$D^i = \begin{bmatrix} B^i & B^{i-1}\boldsymbol{c} \\ \mathbf{0} & 0 \end{bmatrix}. \tag{3.26}$$

Let $\mu_C(x) = x^k + d_{k-1}x^{k-1} + \cdots + d_1x + d_0$. Using the fact $\mu_C(C) = \mathbf{0}$ we have

$$C^k = -(d_{k-1}C^{k-1} + \cdots + d_1C + d_0I) = -\sum_{i=0}^{k-1} d_iC^i. \tag{3.27}$$

From (3.27) and (3.25) we get

$$B^k = -\sum_{i=0}^{k-1} d_iB^i. \tag{3.28}$$

Putting (3.28) into (3.26) we get

$$\begin{aligned} D^{k+1} &= \begin{bmatrix} B^{k+1} & B^k\boldsymbol{c} \\ \mathbf{0} & 0 \end{bmatrix} \\ &= \begin{bmatrix} -\sum_{i=0}^{k-1} d_iB^{i+1} & -\sum_{i=0}^{k-1} d_iB^i\boldsymbol{c} \\ \mathbf{0} & 0 \end{bmatrix} \\ &= -\sum_{i=0}^{k-1} d_iD^{i+1}. \end{aligned} \tag{3.29}$$

Let's consider the following polynomial

$$p(x) = x\mu_C(x) = x^{k+1} + d_{k-1}x^k + \cdots + d_1x^2 + d_0x.$$

By (3.29) we have $p(D) = \mathbf{0}$. Therefore,

$$\deg(\mu_D(x)) \leq \deg(p) = k+1.$$

Equivalence (3.23). Observe that, for any α, equivalences (3.17) to (3.22) still hold by substituting C_α and D_α into C and D, respectively. Therefore, the following implication is true

$$\mu_{C_\alpha}(x) = \mu_{D_\alpha}(x) \Longrightarrow \mathrm{ct}(\mu_{C_\alpha}(x)) = \mathrm{ct}(\mu_{D_\alpha}(x)).$$

We have still to select an appropriate value for α such that

$$\mathrm{ct}(\mu_{C_\alpha}(x)) = \mathrm{ct}(\mu_{D_\alpha}(x)) \Longrightarrow \mu_{C\alpha}(x) = \mu_{D_\alpha}(x). \tag{3.30}$$

Fix any α. Let's denote the distinct eigenvalues of C by $\lambda_1, \ \ldots, \ \lambda_k$. Then the distinct eigenvalues of $C_\alpha = C + \alpha I$ are $\lambda_1 + \alpha, \ \ldots, \ \lambda_k + \alpha$. Observe that C_α is symmetric and $\chi_{C_\alpha}(x) = \chi_{D_\alpha}(x)$. So, we can write

$$\begin{aligned} \mu_{C_\alpha}(x) &= \prod_{i=1}^{k}(x - (\lambda_i + \alpha)), \text{ and} \\ \mu_{D_\alpha}(x) &= \prod_{i=1}^{k}(x - (\lambda_i + \alpha))^{t_i}, \text{ where } t_i \geq 1. \end{aligned}$$

Suppose $\lambda_i + \alpha > 1$ for all i. Then the implication in (3.30) holds. Therefore, it is sufficient to choose such an α that $\lambda_i + \alpha > 1$ for all i.

Let $\|C\|$ be the *maximum column sum matrix norm* of the $(m+n+1) \times (m+n+1)$ matrix $C = [c_{i,j}]$, i.e.

$$\|C\| = \max_{1 \leq j \leq m+n+1} \sum_{i=1}^{m+n+1} |c_{i,j}|.$$

Let $\rho(C)$ be the *spectral radius* of C, i.e.

$$\rho(C) = \max_{1 \leq i \leq k} |\lambda_i|.$$

Then it is known that $\rho(C) \leq \|C\|$ (see [HJ85], Section 5.6). Define

$$\alpha = 2 + \sum_{i,j=1}^{m+n+1} c_{i,j}^2.$$

Obviously, $\rho(C) \leq \|C\| < \alpha$ and $\lambda_i + \alpha > 1$, for $i = 1, 2, \ldots, k$. Note that α can be computed in logspace. □

By Proposition 3.2.11 and 3.2.12, and by Theorem 3.2.16 we get the following corollary.

Corollary 3.2.17 EqMinPolynomial, EqDegMinPol, DegMinPol, *and* EqCTMinPol *are complete for* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

As we have seen in Section 3.2.3, for given matrix A, there is a matrix B_n with **GapL**-computable elements such that $\deg(\mu_A(x)) = \mathrm{rank}(B_n)$. On the other hand, we don't know whether there exists a converse reduction, i.e. given matrix A, compute matrix B such that $\mathrm{rank}(A) = \deg(\mu_B(x))$. Note that Corollary 3.2.17 provides such a reduction only for the bitwise versions of the corresponding functions, namely DegMinPol and Rank.

Recall that the constant term of the characteristic polynomial $\chi_A(x)$ is $(-1)^n \det(A)$. This term is computable in **GapL**. Now assume for a moment that the constant term of the minimal polynomial is in **GapL** as well. It follows that EqCTMinPol is in $\mathbf{C}_=\mathbf{L}$, because this is asking whether the difference of two constant terms (a **GapL**-function) is zero. By Theorem 3.2.16, EqCTMinPol is complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. Therefore, $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L}) = \mathbf{C}_=\mathbf{L}$.

Corollary 3.2.18 *If* CTMinPol *is computable in* **GapL**, *then* $\mathbf{C}_=\mathbf{L}$ *is closed under complement.*

We can considerably weaken the assumption in Corollary 3.2.18: it is sufficient to have a certain *addition property* of the constant term of the minimal polynomial. Namely, given matrices A and B, suppose there is a matrix C such that each element of C is computable in **GapL**, and

$$\mathrm{ct}(\mu_C(x)) = \mathrm{ct}(\mu_A(x)) - \mathrm{ct}(\mu_B(x)).$$

Then we have

$$(A, B) \in \text{EqCTMinPol} \iff \mathrm{ct}(\mu_C(x)) = 0 \iff \det(C) = 0.$$

Therefore, $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ would be collapsed to $\mathbf{C}_=\mathbf{L}$.

Corollary 3.2.19 *If the constant term of the minimal polynomial has the above addition property, then* $\mathbf{C}_=\mathbf{L}$ *is closed under complement.*

3.3 Similarity and diagonalizability of matrices

As a consequence of the results presented in the preceding sections, in this section we show that testing similarity and testing diagonalizability of matrices are complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

3.3.1 Testing similarity

Let A and B be matrices of order n. Recall that A is similar to B if and only if there exists a non-singular matrix P such that $A = PBP^{-1}$. We define the problem of testing similarity of two matrices by

$$\text{SIMILARITY} = \{\ (A, B) \mid A \text{ is similar to } B\ \}.$$

Testing similarity of matrices is known as a classical and fundamental task in linear algebra. There are some necessary and sufficient conditions for similarity of matrices. One of these criteria states that A and B are similar if and only if they have the same invariant factor system, i.e. if and only if $\mathcal{S}_A(x) = \mathcal{S}_B(x)$ (see Chapter 2, page 22). Thus, testing similarity of matrices A and B can be done by computing and comparing the Smith normal form $\mathcal{S}_A(x)$ and $\mathcal{S}_B(x)$. Since the invariant factors of matrices can be computed in $\mathbf{NC}^2$ [Vil97], testing similarity is also in $\mathbf{NC}^2$. But the $\mathbf{NC}^2$ upper bound on the complexity of SIMILARITY has been improved.

A very simple criterion for similarity of matrices was shown by Byrnes and Gauger [BG77].

Theorem 3.3.1 [BG77] *Matrices A and B of the same order are similar if and only if*

(i) $\chi_A(x) = \chi_B(x)$, *and*

(ii) $\operatorname{rank}(A \otimes I - I \otimes A) = \operatorname{rank}(A \otimes I - I \otimes B) = \operatorname{rank}(B \otimes I - I \otimes B)$.

By modifying Theorem 3.3.1 of Byrnes and Gauger, Dixon [Dix79] proved another similarity-criterion based on matrix rank.

Theorem 3.3.2 [Dix79] *Matrices A and B of the same order are similar if and only if* $\operatorname{rank}^2(A \otimes I - I \otimes B) = \operatorname{rank}(A \otimes I - I \otimes A)\ \operatorname{rank}(B \otimes I - I \otimes B)$.

Using Theorem 3.3.2 of Dixon, Garzon and Zalctein [GZ89] presented the first parallel algorithm for testing similarity of matrices. Since computing matrix rank is in $\mathbf{NC}^2$ [Mul87], SIMILARITY is in $\mathbf{NC}^2$ as well.

Santha and Tan [ST98] reconsidered similarity of matrices. They observed that SIMILARITY is dtt reducible to verifying matrix rank. Therefore, SIMILARITY is in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ ([ST98], Theorem 4.4). It was open in the work of Santha and Tan [ST98] whether the problem of testing similarity of matrices is hard for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. Indeed, equivalence (3.18) in the proof of Theorem 3.2.16 (see page 56) shows a many-one reduction from FSLE to SIMILARITY. We get the following corollary.

Corollary 3.3.3 SIMILARITY *is complete for* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Let's say a few words about *testing simultaneous similarity* of matrices. By SIMSIMILARITY we denote the problem of deciding whether there is a nonsingular matrix P such that $A_i = PB_iP^{-1}$, for all pairs of matrices (A_i, B_i), where $i = 1, 2, \ldots, k$ and $k \geq 2$. Although SIMILARITY is complete for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$, it is still open whether SIMSIMILARITY can be solved in $\mathbf{P}$. SIMSIMILARITY belongs to the so-called *"wild" matrix problems* in the work of Grigoriev [Gri83]. He showed that SIMSIMILARITY is in $\mathbf{NP}$ and asked whether SIMSIMILARITY is in $\mathbf{P}$. A recent result by Ogihara and Zalcstein [OZ02] shows that SIMSIMILARITY is solvable in nonuniform $\mathbf{TC}^1$ (i.e. in logspace uniform randomized $\mathbf{TC}^1$). This gives rise to conjecture that SIMSIMILARITY is efficiently solvable in $\mathbf{P}$. Moreover, note that SIMSIMILARITY is hard for $\mathbf{AC}^0$ because asking if A is similar to B is equivalent to asking if (A, I) is simultaneously similar to (B, I).

3.3.2 Testing diagonalizability

Recall that a square matrix A is called diagonalizable if it is similar to a diagonal matrix. Hence, diagonalizability is strongly related to similarity of matrices. Let's define DIAGONALIZABLE to be the set of all diagonalizable matrices. The following theorem shows that similarity and diagonalizability of matrices are in fact equivalent.

Theorem 3.3.4 DIAGONALIZABLE *is complete for* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Proof. Equivalence 3.19 in the proof of Theorem 3.2.16 shows that FSLE is reducible to DIAGONALIZABLE. Thus, DIAGONALIZABLE is hard for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. We have still to show that DIAGONALIZABLE is in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

It was shown in Section 3.2.3 how to construct a matrix B_n, for a given $n \times n$ matrix A, such that $\deg(\mu_A(x)) = \text{rank}(B_n)$. Moreover, it is known from linear algebra that A is diagonalizable if and only if $\mu_A(x)$ contains only linear irreducible factors. Therefore, A is diagonalizable if and only if $\deg(\mu_A(x))$ is equal to the number of distinct eigenvalues of A.

Let l be the number of all distinct eigenvalues of A. Let $H_A = [h_{i,j}]$ be the Hankel matrix associated with A. The elements of H_A are defined as follows

$$h_{i,j} = \text{trace}(A^{i+j-2}), \text{ for } i, j = 1, \ldots, n,$$

where $\text{trace}(X)$ is the sum of all elements on the diagonal of the matrix X. The rank of H_A is related to l with the fact $l = \text{rank}(H_A)$ (see e.g. [Gan77a], Chapter XV, Theorem 6).

In summary,

$$\begin{aligned} A \text{ is diagonalizable} &\iff \deg(\mu_A(x)) = l \\ &\iff \operatorname{rank}(B_n) = \operatorname{rank}(H_A). \end{aligned}$$

Since all elements of B_n and H_A are computable in **GapL**, testing if $\operatorname{rank}(B_n) = \operatorname{rank}(H_A)$ can be done in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. So, DIAGONALIZABLE is in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. □

Testing diagonalizability of a matrix can be extended to testing simultaneous diagonalizability of several matrices. Matrices $A_1, A_2, \ldots, A_k$ are called simultaneously diagonalizable if there exists a nonsingular matrix P such that all the matrices $PA_1P^{-1}, \ldots, PA_kP^{-1}$ are diagonal. Let's define

SIMDIAGONALIZABLE =
$\{ (A_1, A_2, \ldots, A_k) \mid \exists \text{ regular } P : PA_1P^{-1}, \ldots, PA_kP^{-1} \text{ are diagonal } \}$.

In sharp contrast to SIMSIMILARITY where we don't know whether SIMSIMILARITY is in **P**, SIMDIAGONALIZABLE is efficiently solvable in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ as follows.

Consider the case when all matrices A_i are diagonalizable: these matrices are simultaneously diagonalizable if and only if they are pairwise commutable, i.e., $A_iA_j = A_jA_i$ for all $1 \leq i, j \leq k$ (see [HJ85], Section 1.3). For every pair i, j, the equality $A_iA_j = A_jA_i$ can be verified in $\mathbf{NC}^1$. Therefore, the main part of SIMSIMILARITY is to decide whether $A_i \in$ DIAGONALIZABLE for all $1 \leq i \leq k$. This can be done in $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ by Theorem 3.3.4. We get the following corollary.

Corollary 3.3.5 SIMDIAGONALIZABLE *is complete for* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Chapter 4

The Inertia of a Matrix

In this chapter, we investigate the complexity of some problems concerning the inertia of a matrix. The main contribution of this chapter is in improving the results presented in [HT02a]. The text of this chapter is divided into two sections. In Section 4.1, we show how to compute and verify the inertia of a matrix. Section 4.2 deals with the complexity of testing stability of matrices.

4.1 Computing and verifying matrix inertia

Recall from Chapter 2, Section 2.1, that the inertia of an $n \times n$ matrix A, denoted by $i(A)$, is defined to be the triple $(i_+(A), i_-(A), i_0(A))$, where $i_+(A)$, $i_-(A)$, and $i_0(A)$ are the number of eigenvalues of A, counting multiplicities, with positive, negative, and zero real part, respectively.

Let A be a matrix of order n, and let k, $\mathfrak{p}$, $\mathfrak{n}$, and $\mathfrak{z}$ be nonnegative integers. We define the problem of

- computing one bit of the inertia (regarding to some fixed coding) by INERTIA $= \{ (A, k, b) \mid$ the k-th bit of $i(A)$ is $b \}$, and
- verifying the inertia by v-INERTIA $= \{ (A, p, n, z) \mid i(A) = (p, n, z) \}$.

Additionally, we define the following two sets

- POSPOWERELEMENT $= \{ (A, m) \mid (A^m)_{1,n} > 0 \}$, and
- NEGDETERMINANT $= \{ A \mid \det(A) < 0 \}$.

Since the functions POWERELEMENT and DETERMINANT are complete for **GapL**, POSPOWERELEMENT and NEGDETERMINANT are known to be complete for **PL**.

4.1.1 Routh-Hurwitz's Theorem

At the end of the 19th century, Routh and Hurwitz have found a necessary and sufficient condition under which all the roots of a given polynomial lie in the left half plane. Well, actually Routh-Hurwitz's Theorem provides a method for determining the number of roots in the right half-plane of a given real polynomial (see e.g. [Gan77b], Chapter XV). Since the roots of the characteristic polynomial $\chi_A(x)$ are the eigenvalues of matrix A, the inertia of A can be computed by applying the Routh-Hurwitz method to $\chi_A(x)$. We describe this in detail.

Let A be an $n \times n$ matrix. Let the characteristic polynomial of A be the polynomial

$$\chi_A(x) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0.$$

Define $c_n = 1$. The *Routh-Hurwitz matrix* $\Omega(A) = [\omega_{i,j}]$ is defined to be the following $n \times n$ matrix

$$\Omega(A) = \begin{bmatrix} c_{n-1} & c_{n-3} & c_{n-5} & c_{n-7} & \cdots & 0 \\ c_n & c_{n-2} & c_{n-4} & c_{n-6} & \cdots & 0 \\ 0 & c_{n-1} & c_{n-3} & c_{n-5} & \cdots & 0 \\ 0 & c_n & c_{n-2} & c_{n-4} & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & c_0 \end{bmatrix}.$$

That is, the diagonal elements of $\Omega(A)$ are $\omega_{i,i} = c_{n-i}$. In the i-th column, the elements above the diagonal are $\omega_{i-1,i} = c_{n-i-1}$, $\omega_{i-2,i} = c_{n-i-2}$, ... until we reach either the first row $\omega_{1,i}$ or c_0. In the latter case, the remaining entries are filled with zeros. The elements below $\omega_{i,i}$ are $\omega_{i+1,i} = c_{n-i+1}$, $\omega_{i+2,i} = c_{n-i+2}$, ..., c_{n-1}, c_n, 0, 0, ... down to the last row $\omega_{n,i}$.

The successively leading principal minors D_i of $\Omega(A)$ are called the *Routh-Hurwitz determinants*, they are

$$D_1 = c_{n-1}, \quad D_2 = \det(\begin{bmatrix} c_{n-1} & c_{n-3} \\ c_n & c_{n-2} \end{bmatrix}), \ \ldots, \quad D_n = \det(\Omega(A)).$$

For computing the inertia of a matrix, Routh-Hurwitz's Theorem can be formulated as follows.

Theorem 4.1.1 (Routh-Hurwitz) *If $D_n \neq 0$, then the number of roots of the polynomial $\chi_A(x)$ in the right half-plane is determined by the formula*

$$i_+(A) = V(1, D_1, \frac{D_2}{D_1}, \ldots, \frac{D_n}{D_{n-1}}),$$

where $V(x_1, x_2, \dots)$ computes the number of sign alternations in the sequence of numbers $x_1, x_2, \dots$. For the calculation of the values of V, for every group of p successive zero Routh-Hurwitz determinants (p is always odd!)

$$D_s \neq 0,\ D_{s+1} = \cdots = D_{s+p} = 0,\ D_{s+p+1} \neq 0$$

we have to set $V(\frac{D_s}{D_{s-1}}, \frac{D_{s+1}}{D_s}, \dots, \frac{D_{s+p+2}}{D_{s+p+1}}) = h + \frac{1-(-1)^h \varepsilon}{2}$ *, where* $p = 2h - 1$ *and* $\varepsilon = \text{sign}(\frac{D_s}{D_{s-1}} \frac{D_{s+p+2}}{D_{s+p+1}})$*. For* $s = 1$, $\frac{D_s}{D_{s-1}}$ *is to be replaced by* D_1*; and for* $s = 0$*, by* c_0*.*

A proof of this theorem can be found in [Gan77b], Chapter XV, Section 6.

Let's discuss the case when $D_n = 0$. It is known that $D_n = 0$ if and only if $\chi_A(x)$ has at least a pair of opposite roots x_0 and $-x_0$ (see [Gan77b], Chapter XV). Define

$$\begin{array}{rcl} p_1(x) & = & x^n + c_{n-2}x^{n-2} + c_{n-4}x^{n-4} + \cdots, \text{ and} \\ p_2(x) & = & c_{n-1}x^{n-1} + c_{n-3}x^{n-3} + \cdots. \end{array} \tag{4.1}$$

Obviously, we have $\chi_A(x) = p_1(x) + p_2(x)$ and $p_1(x_0) = p_2(x_0) = 0$. Therefore, x_0 is also a root of the polynomial $g(x) = \gcd(p_1(x), p_2(x))$. Observe that by decomposing

$$\chi_A(x) = g(x)\ \chi_A^*(x) \tag{4.2}$$

the polynomial $\chi_A^*(x)$ has no pair of opposite roots, i.e. the Routh-Hurwitz matrix of $\chi_A^*(x)$ is nonsingular. Thus we can use the Routh-Hurwitz theorem to compute the inertia of the companion matrix of $\chi_A^*(x)$. Let B and C be the companion matrix of $g(x)$ and $\chi_A^*(x)$, respectively. Then we get

$$i(A) = i(B) + i(C).$$

Since all non-zero eigenvalues of B are pairs of opposite values, the Routh-Hurwitz method can not be used for computing $i(B)$. No method is known to get the exact number of roots of an integer polynomial on an axis (to the best of our knowledge). However, the number of *distinct* roots of an integer polynomial $p(x)$ on an axis can be determined as follows. Let P be the companion matrix of a polynomial $p(x)$, and let $\deg(p(x)) = n$. Recall that the Hankel matrix $H = [h_{i,j}]$ associated with $p(x)$ is the $n \times n$ matrix with the elements $h_{i,j} = \text{trace}(P^{i+j-2})$, for $i, j = 1, \dots, n$, where $\text{trace}(P^{i+j-2})$ is the sum of all diagonal elements of P^{i+j-2}. So H is always symmetric. We denote the *signature* of H by $\text{signature}(H)$, i.e.

$$\text{signature}(H) = i_+(H) - i_-(H).$$

The following is a useful theorem relating the signature to the rank of the Hankel matrix H associated with the polynomial $p(x)$.

Theorem 4.1.2 ([Gan77b], Theorem 6, page 202)
(1) The number of all distinct real roots *of* $p(x)$ *is equal to* signature(H).
(2) The number of all distinct roots *of* $p(x)$ *is equal to* rank(H).

We will show below that using this theorem we can compute the inertia $i(A)$ in at least some cases where $D_n = 0$.

4.1.2 Upper bounds

We analyze the computation of $i(A)$ by using Theorem 4.1.1.

Using Routh-Hurwitz's method, we must compute all the coefficients c_i of $\chi_A(x)$, and then all Routh-Hurwitz determinants D_i, for $i = 1, \ldots, n$. Since the coefficients $c_1, \ldots, c_n$ are computable in **GapL**, each of the determinants $D_1, \ldots, D_n$ is computable in **GapL** as well (note that by Corollary 2.2.7 **GapL** is closed under composition). Therefore, in **PL** one can decide, for $i = 1, \ldots, n$, whether D_i is positive, negative, or zero.

If $D_n \neq 0$, i.e. if $\Omega(A)$ is nonsingular, then $i_+(A)$ can be determined by using Theorem 4.1.1. Since $i_-(A) = i_+(-A)$, we can apply the same method to compute $i_-(A)$, and then we get $i_0(A) = n - i_+(A) - i_-(A)$. Recall that INERTIA is the problem of computing one bit of $i(A)$ regarding to some fixed coding. Actually, each bit of $i(A)$ is computable in **PL** as follows. According to Theorem 4.1.1, we construct n $\mathbf{AC}^0$-circuits with oracle gates from **PL** so that $i_+(A)$ is equal to the number of 1 in the output-vector of these circuits. To verify the k-th bit of the number of 1 in a binary vector v, we construct an $\mathbf{AC}^0$-circuit with oracle gates from $\mathbf{C}_=\mathbf{L}$ for verifying the k-th bit of the rank of the diagonal matrix having diagonal v. In summary, INERTIA is solvable by an $\mathbf{AC}^0$-circuit with oracle gates from **PL**. Recall that $\mathbf{AC}^0(\mathbf{PL}) = \mathbf{PL}$.

Theorem 4.1.3 *Each bit of the inertia of a matrix* A *where* $\Omega(A)$ *is nonsingular can be computed in* **PL**.

Let's consider the case when $D_n = 0$, i.e. when $\Omega(A)$ is singular. We decompose $\chi_A(x) = g(x)\chi_A^*(x)$, as described in the previous section (see equation (4.2) on page 67). Recall that $g(x) = \gcd(p_1(x), p_2(x)) = \gcd(\chi_A(x), p_1(x))$ is the gcd of two monic polynomials consisting of **GapL**-computable coefficients. The problem of computing the gcd of two polynomials can be reduced to the problem of solving systems of linear equations (see [Koz91], Lecture 34). Based on this reduction, we can prove that the coefficients of $g(x)$ are computable in $\mathbf{TC}^0(\mathbf{GapL})$.

We describe another way to argue that $\mathbf{TC}^0(\mathbf{GapL})$ is an upper bound for computing the gcd of two monic polynomials $g_1(x)$ and $g_2(x)$. Let G_1 and G_2 be the companion matrices of $g_1(x)$ and $g_2(x)$, respectively. Let $G = \mathrm{diag}[G_1, G_2]$. Observe that

$$\begin{aligned}\gcd(g_1(x), g_2(x)) &= \frac{g_1(x)\ g_2(x)}{\mathrm{lcm}(g_1(x), g_2(x))} \\ &= \frac{\chi_G(x)}{\mathrm{lcm}(\mu_{G_1}(x), \mu_{G_2}(x))} \\ &= \frac{\chi_G(x)}{\mu_G(x)}.\end{aligned}$$

Obviously, the coefficients of $\chi_G(x)$ are computable in **GapL**. By Theorem 3.2.1, the minimal polynomial $\mu_G(x)$ is computable in $\mathbf{TC}^0(\mathbf{GapL})$. In order to compute $\gcd(g_1(x), g_2(x))$, we use Algorithm 15.2 in [IK93] for the polynomial division $\chi_G(x)/\mu_G(x)$. By Algorithm 15.2 in [IK93], the coefficients of the quotient q and the remainder r of the polynomial division f/h (where h is monic) are expressed as **GapL** computations of the coefficients of f and h. Thus, in turn, $\gcd(g_1(x), g_2(x))$ is computable in $\mathbf{TC}^0(\mathbf{GapL})$.

When $g(x)$ has been found, the polynomial $\chi_A^*(x) = \frac{\chi_A(x)}{g(x)}$ can be computed by using Algorithm 15.2 in [IK93]. This can be done also in $\mathbf{TC}^0(\mathbf{GapL})$. In summary, each element of B, the companion matrix of $g(x)$, and C, the companion matrix of $\chi_A^*(x)$), is computable in $\mathbf{TC}^0(\mathbf{GapL})$. Furthermore, it is important to note that these elements (of B and C) can be expressed as divisions (without remainders) of **GapL** values.

Since each element of C is expressed as a division (without remainder) of two **GapL** values, we can compute $i(C)$ by formula $i(C) = \mathrm{sign}(d)i(C')$, where d is the product of all denominators in C and $C' = dC$. Note that $i(C')$ can be computed by using Routh-Hurwitz's Theorem. Since each bit of $i(C')$ is computable in **PL**, so is each bit of $i(C)$.

It remains a major problem: *How to compute $i(B)$?*. There is no method for computing $i(B)$ in general. However, in some cases when $g(x)$ is simple, we can do so anyway.

Suppose for example that

$$g(x) = x^t, \text{ for some } t \geq 0,$$

equivalently, we can say that B (and hence A) has *no opposite non-zero eigenvalues*. Then we get $i(B) = (0, 0, t)$ and $i(A) = (0, 0, t) + i(C)$. Note that in $\mathbf{coC}_{=}\mathbf{L}$ we can decide whether A has no opposite non-zero eigenvalues as follows:

with the greatest t so that x^t is a divisor of $\chi_A(x)$ (it is possible that $t = 0$) we decide whether the Routh-Hurwitz matrix associated with the polynomial $\frac{\chi_A(x)}{x^t}$ is nonsingular.

Corollary 4.1.4 *Each bit of the inertia of a matrix having no opposite non-zero eigenvalues can be computed in* **PL**.

We can considerably extend Corollary 4.1.4 to the following theorem.

Theorem 4.1.5 *Each bit of the inertia of a matrix A satisfying one of the following properties:*

(1) A has all *opposite eigenvalues on the imaginary axis, and*

(2) A has no *opposite eigenvalues on the imaginary axis*

is computable in **PL**.

Proof. Assume that the condition on A is fulfilled. Let B and C be respectively the companion matrix of $g(x)$ and $\chi_A^*(x)$ by decomposing $\chi_A(x) = g(x)\chi_A^*(x)$, as mentioned before. We compute the inertia of A by $i(A) = i(B) + i(C)$.

The triple $i(B)$ can be easily expressed from $\deg(g(x))$ as follows:

- $i(B) = (0, 0, \deg(g(x))$ if all opposite eigenvalues of A lie on the imaginary axis, and
- $i(B) = (\frac{1}{2}\deg(g(x)), \frac{1}{2}\deg(g(x)), 0)$, if no opposite eigenvalues of A lie on the imaginary axis.

Moreover, note that $\deg(g(x))$ and $i(C)$ can be determined in **PL**. Therefore, each bit of $i(A)$ is computable in **PL**.

Theorem 4.1.2 can be used to decide whether A fulfills one of the considered properties. This can be done in **PL** as follows.

Since Theorem 4.1.2 deals with the real axis instead of the imaginary axis, we first turn $g(x)$ by 90°. It is well known from linear algebra that the eigenvalues of the matrix $P \otimes Q$, for matrices P and Q of order n and m, are $\lambda_j(P)\lambda_k(Q)$ where $\lambda_j(P)$ and $\lambda_k(Q)$ are the eigenvalues of P and Q, for all $1 \leq j \leq n$ and $1 \leq k \leq m$, respectively. For our purpose, observe that the eigenvalues of the matrix $E = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$ are $+i$ and $-i$. Define $D = A \otimes E$. Then the eigenvalues of D are $i\lambda_k(A)$ and $-i\lambda_k(A)$ where $\lambda_k(A)$ runs through all eigenvalues of A. W.l.o.g. we can assume that A is nonsingular. It follows that the number of distinct

imaginary eigenvalues of A is equal to the number of distinct real eigenvalues of D.

Let H be the Hankel matrix of $\chi_D(x)$. By Theorem 4.1.2 we get the following equivalences

$$A \text{ fulfills (1)} \iff \text{signature}(H) = \deg(g(x)), \tag{4.3}$$

$$A \text{ fulfills (2)} \iff \text{signature}(H) = 0. \tag{4.4}$$

Since H is symmetric, all eigenvalues of H are real.

Let's decompose $\chi_H(x) = h(x)\chi_H^*(x)$ as described in (4.2). Then signature(H) is equal to the difference $i_+(H^*) - i_-(H^*)$, where H^* is the companion matrix of $\chi_H^*(x)$, because all roots of $h(x)$ are in pairs of opposite values. By Corollary 4.1.4, $i(H^*)$ can be determined in **PL**. Therefore, the right-hand sides of (4.3) and (4.4) can be done in **PL** as well. □

Based on closure properties of **PL**, we get the same upper bounds for verifying as for computing the inertia.

Proposition 4.1.6 *Verifying the inertia of a matrix A having one of the following properties*

1. $\Omega(A)$ *is nonsingular,*
2. *A has no opposite non-zero eigenvalues,*
3. *A has all opposite eigenvalues on the imaginary axis,*
4. *A has no opposite eigenvalues on the imaginary axis*

can be done in **PL**.

4.1.3 Lower bounds

The following theorem shows a lower bound on the complexity of Inertia and v-Inertia.

Theorem 4.1.7 Inertia *and* v-Inertia *are hard for* **PL**.

Proof. We construct a many-one reduction from a **PL**-complete problem, in particular from PosPowerElement, to Inertia and v-Inertia. Let A be an $n \times n$ matrix, and let m be a positive integer. PosPowerElement is the problem of deciding whether $(A^m)_{1,n} > 0$.

Recall the reduction from matrix powering to computing the characteristic polynomial of a matrix (Theorem 3.1.3): a matrix B has been constructed such that

$$(A^m)_{1,n} = a \iff \chi_B(x) = x^{N-2m-1}\left(x^{2m+1} - a\right),$$

where $N = m(n+d)+n$, and d is the number of non-zero elements in A. We show that $i(B)$ or one of its bits can be used to tell the sign of $(A^m)_{1,n}$.

Let's consider the eigenvalues of B, i.e. the roots of $\chi_B(x)$. Observe that in the case when $a = (A^m)_{1,n} \neq 0$, then the roots of $x^{2m+1} - a$ are the corners of a regular $(2m+1)$-gon inscribed in a circle of radius $a^{\frac{1}{2m+1}}$ with its center at the origin. Since $2m+1$ is odd, none of these roots lies on the imaginary axis. This implies that $i_0(B) = N - (2m+1)$, and one of $i_+(B)$ and $i_-(B)$ is m and the other is $m+1$. Moreover, these values depend only on the sign of a. Namely, if $a > 0$, then

$$i_+(B) = \begin{cases} m+1, & \text{if } 2m+1 \equiv 1 \pmod 4, \\ m, & \text{if } 2m+1 \equiv 3 \pmod 4. \end{cases} \tag{4.5}$$

This implies that $i_+(B)$ in (4.5) is always odd. Analogously, $i_+(B)$ is even if $a < 0$. Note that in the case where $(A^m)_{1,n} = 0$, we have $i(B) = (0, 0, N)$.

In summary, in logspace we can compute $\mathfrak{p}$, $\mathfrak{n}$, $\mathfrak{z}$ such that

$$\begin{aligned} (A^m)_{1,n} > 0 &\iff i(B) = (\mathfrak{p},\mathfrak{n},\mathfrak{z}) \\ &\iff i_+(B) = \text{odd}. \end{aligned}$$

□

Note also that the matrix B in the proof of Theorem 4.1.7 has no pair of opposite non-zero eigenvalues. Therefore, B fulfills the condition of Corollary 4.1.4. We get the following corollary.

Corollary 4.1.8 *The computation and the verification of the inertia of a matrix with no opposite non-zero eigenvalues are complete for* **PL**.

Congruence of symmetric matrices

Recall that two symmetric matrices A and B are called congruent if there is an invertible matrix P such that $A = PBP^T$. Sylvester's Law of Inertia states the following relation between matrix congruence and matrix inertia: A and B are congruent if and only if they have the same inertia. Let's consider

- Congruence = $\{\ (A, B) \mid A, B :$ symmetric, $i(A) = i(B)\ \}$.

We show the following proposition.

Proposition 4.1.9 CONGRUENCE *is in* **PL**, *and is hard for* $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$.

Proof. Since all eigenvalues of a symmetric matrix are real, by Theorem 4.1.5 the inertia of a symmetric matrix can be computed and verified in **PL**. Thus CONGRUENCE $\in$ **PL**.

CONGRUENCE is hard for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$ by showing a reduction from FSLE to CONGRUENCE. Let $(A, \boldsymbol{b})$ be an input to FSLE. The reduction is presented as follows.

$$
\begin{aligned}
(A,\boldsymbol{b}) \in \mathrm{FSLE} &\iff \mathrm{rank}(A) = \mathrm{rank}([A|\boldsymbol{b}]) \\
&\iff \mathrm{rank}(A^TA) = \mathrm{rank}([A|\boldsymbol{b}]^T[A|\boldsymbol{b}]) \\
&\iff \mathrm{rank}(\begin{bmatrix} A^TA & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}) = \mathrm{rank}([A|\boldsymbol{b}]^T[A|\boldsymbol{b}] \\
&\iff \begin{bmatrix} A^TA & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \text{ is congruent to } [A|\boldsymbol{b}]^T[A|\boldsymbol{b}] \\
&\iff (\begin{bmatrix} A^TA & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}, [A|\boldsymbol{b}]^T[A|\boldsymbol{b}]) \in \textsc{Congruence}.
\end{aligned}
$$

□

4.2 Stability of matrices

By POSSTABLE and POSSEMISTABLE we denote the sets of all positive stable and positive semi-stable matrices, respectively, i.e.

- POSSTABLE $= \{\ A \mid i(A) = (n, 0, 0)\ \}$, and
- POSSEMISTABLE $= \{\ A \mid i_-(A) = 0\ \}$,

where A is a matrix of order n. (In case of Hermitian matrices, we call these sets POSDEFINITE and POSSEMIDEFINITE, respectively.)

We show the following theorem.

Theorem 4.2.1 POSSTABLE *and* POSSEMISTABLE *are in* **PL**.

Proof. Let A be a matrix of order n. It is known from linear algebra that A is a positive stable matrix if and only if all the Routh-Hurwitz determinants of the

matrix $\Omega(A)$ are positive. Hence, positive stability of matrices can be decided in **PL**, i.e. PosStable $\in$ **PL**.

If $\Omega(A)$ is nonsingular, then PosSemistable $\in$ **PL** due to Theorem 4.1.3.

So assume that $\Omega(A)$ is singular. As mentioned in the preceding section (see page 67), decomposing $\chi_A(x) = g(x)\chi_A^*(x)$ can be done in $\mathbf{TC}^0(\mathbf{GapL})$. Let B and C be the companion matrices of $g(x)$ and $\chi_A^*(x)$, respectively. Then matrix A is positive semi-stable if and only if matrix B is positive semi-stable and matrix C is positive stable. Furthermore, since all non-zero eigenvalues of matrix B are pairs of opposite values, B is positive semi-stable if and only if all eigenvalues of B are on the imaginary axis. It follows from Theorem 4.1.5 that the inertia of B can be computed within **PL**.

In summary, PosSemistable is in **PL**. □

A simple observation leads to lower bounds for PosStable and PosSemistable is as follows: a matrix A is nonsingular if and only if the symmetric matrix AA^T is positive definite. Since the product AA^T can be computed in $\mathbf{NC}^1$, PosDefinite is hard for $\mathbf{coC}_=\mathbf{L}$.

Corollary 4.2.2 PosDefinite *is hard for* $\mathbf{coC}_=\mathbf{L}$.

As an immediate consequence of Corollary 4.2.2, $\mathbf{coC}_=\mathbf{L}$ is a lower bound for PosStable and PosSemistable (note that PosDefinite is a subset of PosStable). Actually, this bound can be improved by the following theorem.

Theorem 4.2.3 PosStable *and* PosSemistable *are hard for* **PL**.

Proof. Recall that the set of all matrices with negative determinant, denoted by NegDeterminant, is complete for **PL**. To show that PosStable is hard for **PL** we construct a reduction from NegDeterminant to PosStable.

Let A be an $n \times n$ integer matrix. Let $d_{1,1}, \ldots, d_{n,n}$ be the diagonal elements of A^TA. The *Hadamard Inequality* (see [Gan77a], Chapter IX) states that

$$|\det(A)| \leq (d_{1,1} \cdots d_{n,n})^{1/2}. \tag{4.6}$$

Allender [All02] pointed out that w.l.o.g. one can assume that the input matrix A is over $\{0, 1\}$, in which no row of A consists of more than two one's. For clarity, this restriction will be shown by Lemma 4.2.4 below. As a consequence, putting $d_{i,i} \leq 2$ in Hadamard's Inequality (4.6) we get the following bound for $\det(A)$

$$-2^n < \det(A) < 2^n.$$

Define

$$t = \lceil \frac{n}{2m+1} \rceil (2m+1), \text{ for an integer } m \geq 1.$$

Since $n \leq t$, we have

$$\begin{array}{l} \det(A) + 2^t \geq \det(A) + 2^n > 0, \text{ and} \\ \det(A) < 0 \iff \det(A) + 2^t < 2^t. \end{array} \tag{4.7}$$

Lemma 4.2.5 below states that in logspace a matrix B of order k and a positive integer m can be computed such that

$$(B^m)_{1,k} = \det(A) + 2^t, \text{ for } t = \lceil \frac{n}{2m+1} \rceil (2m+1). \tag{4.8}$$

Note that m depends on t, and we defined t in terms of m. This makes the construction a bit tricky.

We further reduce B to a matrix C such that

$$\chi_C(x) = x^{N-2m-1}(x^{2m+1} - b),$$

where $N = m(k+d) + k$, d is the number of elements different from zero of B, and $b = (B^m)_{1,k}$. Recall that this is the $\mathbf{AC}^0$-reduction from PowerElement to CharPolynomial in the proof of Theorem 3.1.3.

As explained in the proof of Theorem 4.1.7, matrix C has $N-2m-1$ eigenvalues, which are equal to zero, and $2m+1$ eigenvalues as the roots of the polynomial $x^{2m+1} - b$. The latter $2m+1$ eigenvalues of C are complex and lie on the circle of radius $r = b^{\frac{1}{2m+1}}$ (with the origin as center). Since $b > 0$, the eigenvalue with the largest real part is $\lambda_{\max}(C) = r$.

Finally, we define

$$D = -C + sI, \text{ for } s = 2^{\frac{t}{2m+1}} = 2^{\lceil \frac{n}{2m+1} \rceil}.$$

Thereby, to obtain the eigenvalues of D we add s to the eigenvalues of $-C$, i.e. the eigenvalues of D can be computed by the following formula

$$\lambda_i(D) = s - \lambda_i(C), \text{ for all } 1 \leq i \leq n.$$

Hence, the eigenvalue of C with the largest real part (which is equal to r) becomes the eigenvalue of D with the smallest real part (which is denoted by $\lambda_1(D)$), i.e.

$$\lambda_1(D) = s - r.$$

Therefore,

$$b < 2^t \iff r < s \iff \lambda_1(D) > 0. \tag{4.9}$$

From (4.7), (4.8), and (4.9) we get

$$A \in \text{NegDeterminant} \iff D \in \text{PosStable}.$$

An analogous argument reduces the set of matrices having non-positive determinants, $\overline{\text{PosDeterminant}}$, to PosSemistable. □

We complete the proof of Theorem 4.2.3 by showing the following two lemmas.

Lemma 4.2.4 *The problem of computing the determinant of a* 0-1 *matrix, in which no row consists of more than two 1's, is complete for* **GapL**

Proof. It is sufficient to show that the considered problem is **GapL**-hard.

Recall that PathDifference is the problem of computing $\text{path}(G, s, t_1) - \text{path}(G, s, t_2)$, given an acyclic directed graph G, and nodes s, t_1, and t_s. PathDifference is complete for **GapL** (see page 30). We show a reduction from PathDifference to the determinant of a 0-1 matrix.

By using Valiant's technique [Val79a] (see also the proof of Theorem 3.2 in [ST98]), we modify the directed graph G as follows.

(1) Add the edges (t_1, t) and (t_2, t) where t is a new node.

(2) Replace every edge $e = (u, v)$, except (t_2, t) by two new edges (u, u_e) and (u_e, v) where u_e is a new node corresponding to edge e.

(3) Add a self-loop on each node, except s and t.

We denote the resulting graph by H, and we show that $\det(B) = \text{path}(G, s, t_1) - \text{path}(G, s, t_2)$, where B is the adjacency matrix of H.

A combinatorial method for computing the determinant (see e.g. [MV97, BR91, CDS80]) states that $\det(B)$ is the sum of all *cycles covering* H, with appropriate sign. Consider the graph H. Any cycle covering H consists of a cycle consisting of a path $s \rightsquigarrow t$ and the edge (t, s), and self-loops on the remaining nodes. Furthermore, observe that

- there is a 1-1 relation mapping paths $s \rightsquigarrow t_1$ and $s \rightsquigarrow t_2$ in G to cycles consisting of a path $s \rightsquigarrow t$ passing the node t_1 and t_2, respectively,

- since a self-loop and every cycle consisting of a path $s \rightsquigarrow t$ passing the node t_1 are of odd length, they have the sign 1, and

- since every cycle consisting of a path $s \rightsquigarrow t$ passing the node t_2 is of even length, its sign is -1.

Therefore, we get $\det(B) = \mathrm{path}(G, s, t_1) - \mathrm{path}(G, s, t_2)$.

As mentioned on the problem PATH (see page 28), w.l.o.g. we can assume that the maximal out-degree of G equals 2. Hence, no row of the 0-1 matrix B consists of more than two one's. □

Lemma 4.2.5 *Given an $n \times n$ matrix A, one can construct in logspace a matrix B of order k and a positive integer m such that*

$$(B^m)_{1,k} = \det(A) + 2^t, \text{ for } t = \lceil \frac{n}{2m+1} \rceil (2m+1).$$

Proof. The construction of the matrix B can be described as follows.

1. Since the function POWERELEMENT is complete for **GapL**, there is a logspace many-one reduction from DETERMINANT to POWERELEMENT, i.e. in logspace one can compute a matrix B_0 of order l and an exponent $m > 0$ such that $\det(A) = (B_0^m)_{1,l}$, for given matrix A of order n.

2. Define an $(m+1) \times (m+1)$ block matrix F whose all m blocks on the first upper sub-diagonal are purely B_0, and whose all other blocks are zero, i.e.

$$F = \begin{bmatrix} & B_0 & & & \mathbf{0} \\ & & B_0 & & \\ & & & \ddots & \\ & & & & B_0 \\ \mathbf{0} & & & & \end{bmatrix}.$$

3. Define

$$S = \begin{bmatrix} s^2 & s^3 \\ 0 & 0 \end{bmatrix}, \text{ for } s = 2^{\lceil \frac{n}{2m+1} \rceil}.$$

   An elementary calculation yields

$$S^m = \begin{bmatrix} s^{2m} & s^{2m+1} \\ 0 & 0 \end{bmatrix} \text{ and } s^{2m+1} = 2^t.$$

4. Define

$$T = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \text{ of the dimension } l(m+1) \times 2.$$

5. Define

$$B = \begin{bmatrix} F & FT+TS \\ \mathbf{0} & S \end{bmatrix} \text{ of order } k = l(m+1)+2.$$

Now we show $(B^m)_{1,k} = \det(A) + 2^t$.
An elementary calculation yields

$$B^m = \begin{bmatrix} F^m & F^mT + 2F^{m-1}TS + \cdots + 2FTS^{m-1} + TS^m \\ \mathbf{0} & S^m \end{bmatrix}. \tag{4.10}$$

Observe that, for each $1 \leq i \leq m$, the power matrix F^i is of a very simple form: on its i-th upper sub-diagonal are purely B^i and all the other blocks are zero-matrix. Furthermore, we have $F^{m-i}TS^i = \mathbf{0}$, for all $i < m$. Therefore, we can deduce the form of B^m in (4.10) into

$$B^m = \begin{bmatrix} F^m & F^mT + TS^m \\ \mathbf{0} & S^m \end{bmatrix}.$$

It follows that

$$\begin{aligned} (B^m)_{1,k} &= (F^mT + TS^m)_{1,2} \\ &= (F^mT)_{1,2} + (TS^m)_{1,2} \\ &= (B_0^m)_{1,l} + s^{2m+1} \\ &= \det(A) + 2^t, \text{ for } t = \lceil \frac{n}{2m+1} \rceil (2m+1). \end{aligned}$$

□

Chapter 5

Further Results

This chapter is organized as follows: in Section 5.1, we give an approach towards the complexity of the unique perfect matching problem for bipartite graphs; in Section 5.2 and 5.3, we show some necessary and sufficient conditions for collapsing counting logspace hierarchies.

5.1 Unique perfect matching

For a graph $G = (V, E)$ with n nodes and m edges, a subset of edges, $M \subseteq E$, with the property that no two edges of M are adjacent by a common node, is called a *matching* of G. A matching which is incident to all nodes in V is called *perfect*. We denote the number of all perfect matchings of G by $\#\mathrm{pm}(G)$.

One of the most prominent open questions in complexity theory regarding parallel computations asks whether there is an **NC** algorithm to decide whether a given graph contains a perfect matching. The perfect matching problem is known to be in **P** [Edm65], and in randomized $\mathbf{NC}^2$ by Mulmuley, Vazirani, and Vazirani [MVV87]. The randomized $\mathbf{NC}^2$ algorithm presented in [MVV87] has been used in [ABO99] for showing that the perfect matching problem is in $\mathbf{coC_{=}L}/poly$. The latter result has been improved by Allender, Reinhardt, and Zhou [ARZ99] to nonuniform **SPL**.

Since no **NC** algorithm for detecting perfect matchings in a graph is known, some special cases of perfect matching problem attract a great attention. For example, an **NC** upper bound has been found even for regular bipartite graphs [LPV81], dense graphs [DHK93], and strongly chordal graphs [DK86]. Although the number of all perfect matchings in a graph is hard for $\#\mathbf{P}$ [Val79b], the perfect matching problem for graphs having polynomially bounded number of perfect matchings is in **NC** by Grigoriev and Karpinski [GK87].

The unique perfect matching problem is given by Lovász. He asked whether there is an **NC** algorithm for testing if a given graph has a unique perfect matching. Kozen, Vazirani, and Vazirani showed in [KVV85] that the unique perfect matching problem for bipartite graphs as well as for general graphs is in **NC**. This result seems to be corrected only for bipartite graphs [KVV86], and the unique perfect matching problem for general graphs is still open.

We reconsider the unique perfect matching problem. Taking in consideration that a completeness result on the complexity of the considered problem isn't known, an approach towards the complexity of the problem is motivated always. We show that the unique perfect matching problem for bipartite graphs is sandwiched between $\mathbf{C_=L}$ and **NL**.

Formally, we denote the problem of deciding whether a graph has exactly one perfect matching by the set UPM $= \{\ G \mid \#\mathrm{pm}(G) = 1\ \}$. In this thesis we consider the case when the given graph is bipartite, and then we give a method for constructing the unique perfect matching in a general graph.

Let $G = (U, V; E)$ be a bipartite graph with $2n$ nodes from $U = \{1, 2, \ldots, n\}$ and $V = \{1, 2, \ldots, n\}$, and m edges from $E \subseteq U \times V$. Let's denote by A the 0-1 bipartite adjacency matrix corresponding to G, i.e. for all $1 \leq i, j \leq n$, we have

$$a_{i,j} \;=\; \begin{cases} 1\ , & \text{if } (i,j) \in E, \\ 0\ , & \text{otherwise.} \end{cases}$$

Define matrices $B = [b_{i,j}]$ by $b_{i,j} = a_{i,j}\ \det^2(A_{i|j})$, for all $1 \leq i, j \leq n$, and $C = AB^T - I$. Let M be a subset of E. To obtain a lower bound on the complexity of UPM for bipartite graphs we prove the following lemma.

Lemma 5.1.1 *The bipartite graph G has a unique perfect matching if and only if*

(1) B is a permutation matrix, and

(2) $\chi_C(x) = x^n$.

Proof. Consider the case when G has exactly one perfect matching M.

- Observe that the determinant of the adjacency matrix corresponding to a bipartite graph having a unique perfect matching equals either +1 or -1. For each edge (i, j) from the unique perfect matching M, the bipartite graph obtained by deleting nodes $i \in U$ and $j \in V$ from G belongs to UPM. Therefore, the corresponding adjacency matrix $A_{i|j}$ satisfies $\det^2(A_{i|j}) = 1$.

Based on the fact that the perfect matching M is unique in G, each row as well as each column of the 0-1 matrix B contains only one 1, i.e. B is a permutation matrix.

- The directed graph corresponding to $C = AB^T - I$ doesn't have any directed cycles, therefore $\chi_C(x) = x^n$ (see Theorem 3.1.3, page 37).

Conversely, if (1) and (2) are satisfied, then we show that G is in UPM by using the following two observations.

- The permutation matrix B from (1) corresponds to a perfect matching M in G, and the diagonal elements of AB^T are 1 corresponding to the edges from M.
- Since x^n is the characteristic polynomial of the 0-1 matrix C, the directed graph interpreted by C contains no cycle, i.e. M cannot be alternated. The perfect matching M is unique.

□

Let G be an arbitrary graph with $2n$ nodes from $U = \{1, 2, \ldots, 2n\}$ and m edges from $E \subseteq U \times U$. Let $A = [a_{i,j}]$ be the adjacency matrix of G, i.e. for all $1 \leq i, j \leq 2n$,

$$a_{i,j} \;=\; a_{j,i} \;=\; \begin{cases} 1\ , & \text{if } (i,j) \in E \\ 0\ , & \text{otherwise.} \end{cases}$$

Let $A' = [a'_{i,j}]$ be the skew-symmetric matrix obtained by negating elements below the diagonal of A, i.e. for all $1 \leq i, j \leq n$,

$$a'_{i,j} \;=\; \begin{cases} a_{i,j}\ , & \text{if } i \leq j \\ -a_{i,j}\ , & \text{otherwise.} \end{cases}$$

The pfaffian of A', denoted by $\mathrm{pf}(A')$, is defined by $\det(A') = \mathrm{pf}^2(A')$ (see e.g. [MSV99] for more detail). In particular, $\mathrm{pf}(A')$ is the signed sum of all perfect matchings in G. For example, if G has no perfect matching, then $\mathrm{pf}(A') = 0$, and if G has a unique perfect matching, then $\mathrm{pf}(A') = +1$ or -1 depending on the sign of the unique perfect matching.

Furthermore, we define $B = [b_{i,j}]$ where

$$b_{i,j} = a_{i,j}\ \det(A'_{i,j|i,j}),$$

B' to be the matrix obtained by negating elements below the diagonal of B, and $C = A'B'^T - I$.

We show the following lemma for arbitrary graphs.

Lemma 5.1.2 *If the graph G has a unique perfect matching, then*

(1) B is a symmetric permutation matrix, and

(2) $\chi_C(x) = x^{2n}$.

Proof. Suppose G has the unique perfect matching M. Observe that each graph obtained from G by deleting two adjacent nodes i and j has either only one perfect matching, if $(i,j) \in M$, or no perfect matching, if $(i,j) \notin M$. Therefore, we have $\mathrm{pf}(A'_{i,j|i,j}) \in \{+1,-1,0\}$, i.e. $\det(A'_{i,j|i,j}) \in \{0,1\}$. Since M is unique in G, a node i of G can be only matched by an edge $(i,j) \in M$. Therefore, each row of B contains only one 1. Consequently, B is a symmetric permutation matrix.

Observe that $B'^T B' = I$ and $\det(B'^T) = 1$. Thus we get $\chi_C(x) = \det(xB' - A' + B')$. Since the unique perfect matching M is interpreted by B', the latter determinant is equal to x^{2n}. □

Note that a perfect matching M corresponds to the symmetric permutation matrix B: $b_{i,j} = 1$ indicates that $(i,j) \in M$. Hence, the unique perfect matching in an arbitrary graph $G \in$ UPM can be determined by computing the matrix B.

Using Lemma 5.1.1 we show that $\mathbf{C_=L}$ is an upper bound for bipartite UPM. Note thereby that all elements of the matrices mentioned in Lemma 5.1.1 and 5.1.2 are computable in **GapL**, and that a matrix $B = [b_{i,j}]$ is a permutation matrix if and only if B is a 0-1 matrix such that $\sum_{i=1}^{n} b_{i,j} = \sum_{i=1}^{n} b_{k,i} = 1$, for each $1 \leq j,k \leq 2n$. We conclude:

Theorem 5.1.3 UPM *for bipartite graphs is in* $\mathbf{C_=L}$.

Now we show that UPM is hard for **NL** (note that **NL** = **coNL** [Imm88, Sze88]). Let G be a directed acyclic graph, and let s and t be two designed nodes. For simplicity, by $\#\mathrm{path}(G,s,t)$ we denote the number of paths in a graph G going from s to t. The connectivity problem which is hard for **coNL** asks whether $\#\mathrm{path}(G,s,t) = 0$. The construction of a new bipartite graph H is followed by Chandra, Stockmeyer, and Vishkin [CSV84]:

1. Retain nodes s and t in H.
2. Add to H new nodes u_{in} and u_{out} and new edges (u_{in}, u_{out}), for each node u of G except for s and t.
3. Add to H edges (s, u_{in}), for each node u of G that u is adjacent to s.
4. Add to H edges (u_{out}, t), for each node u of G that u is adjacent to t.

5. Add the edge (s,t) to H.

Let's denote by H' the graph obtained from H by deleting the edge (s,t). Then we get

$$\begin{aligned} \#\text{path}(G,s,t) &= \#\text{pm}(H') \text{ (see [MSV99])}, \text{ and} \\ \#\text{pm}(H) &= \#\text{pm}(H') + 1. \end{aligned}$$

Note that there is a perfect matching M consisting of the edge (s,t) and all the edges (u_{in}, u_{out}). Therefore,

$$\#\text{path}(G,s,t) = 0 \iff \#\text{pm}(H) = 1.$$

Mahajan [Mah03] pointed out that the used reduction is based on the reduction from the directed connectivity problem to the perfect matching problem by Chandra, Stockmeyer, and Vishkin [CSV84]. Furthermore, there is another way by Mahajan [Mah03] to argue a proof. We describe it as follows.

(i) Since $\mathbf{NL} = \mathbf{coNL}$, there are functions $f, g \in \#\mathbf{L}$ such that

$$\begin{aligned} f(x) > 0 &\Longrightarrow g(x) = 0, \quad \text{and} \\ f(x) = 0 &\Longrightarrow g(x) > 0. \end{aligned}$$

(ii) For a new function $h = g + 1$ we get the following implications

$$\begin{aligned} f(x) > 0 &\Longrightarrow h(x) = 1, \quad \text{and} \\ f(x) = 0 &\Longrightarrow h(x) > 1. \end{aligned}$$

(iii) We shall construct, as described above, the graph H' from G corresponding to the **NL** machine computing h. It follows that

$$f(x) > 0 \iff \#\text{pm}(H') = 1.$$

We conclude:

Theorem 5.1.4 UPM *for bipartite graphs is hard for* **NL**.

Let's say some words about the complexity of UPM for bipartite graphs. Obviously, it will be very interesting and surprising, if the unique perfect matching problem for bipartite graphs is in (uniform) **SPL**. Assume for a moment, if the considered problem is sandwiched between **NL** and **SPL** (in the uniform setting),

then this would imply that $\mathbf{NL} \subseteq \mathbf{SPL}$. As already noted in [AO96], there is no reason in believing that the latter conclusion is true. With this in mind one can say that there is no reason to expect that the unique perfect matching problem is in **SPL**. So as to get the exact complexity of UPM, one can think over whether UPM is complete for either $\mathbf{C_=L}$ or **NL**. Unfortunately, all these two variants seem to be very difficult!

5.2 Conditions for collapsing the $\mathbf{C_=L}$-hierarchy

The complexity of matrix rank has been studied in [ABO99]. The problem of verifying the rank of a matrix, v-Rank, is known to be complete for $\mathbf{C_=L} \wedge \mathbf{coC_=L}$. It is obvious to see that $\mathbf{C_=L} = \mathbf{coC_=L}$ if the rank of a matrix is computable in **GapL**. It is natural to ask whether the rank can be computed in **GapL**. We show the following theorem.

Theorem 5.2.1 $\text{rank} \in \mathbf{GapL} \iff \mathbf{C_=L} = \mathbf{SPL}$.

Proof. Suppose $\text{rank} \in \mathbf{GapL}$. Then $\mathbf{C_=L} = \mathbf{coC_=L}$ because v-Rank $\in \mathbf{C_=L}$. Furthermore, recall that there is a reduction [ABO99] (Corollary 2.3, see also Chapter 3) from v-PowerElement to v-Rank in the following sense

$$\begin{aligned} (A^m)_{1,n} = 0 &\iff \text{rank}(B) = N - 1, \text{ and} \\ (A^m)_{1,n} \neq 0 &\iff \text{rank}(B) = N, \end{aligned}$$

where matrix B of order N can be computed from the input to v-PowerElement. Define a **GapL**-function g to be the difference between the order and the rank of a matrix, i.e. $g(B) = N - \text{rank}(B)$. One can use this function as the characteristic function for deciding whether $(A, m, 1, n, 0) \in$ v-PowerElement

$$g(B) = \begin{cases} 1, & \text{if } (A^m)_{1,n} = 0 \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, $\mathbf{AC^0(C_=L)} = \mathbf{C_=L} = \mathbf{SPL}$. We explain briefly another method for showing the implication: Taking $h = 2 - \text{rank}(A')$ as the seeking characteristic function, where A' is the 2×2 diagonal matrix having $(A^m)_{1,n}$ and 1 on its diagonal, we get

$$h = \begin{cases} 1, & \text{if } (A^m)_{1,n} = 0, \\ 0, & \text{otherwise.} \end{cases}$$

Suppose $\mathbf{C_=L} = \mathbf{SPL}$. Then $\mathbf{AC^0(C_=L)} = \mathbf{C_=L} \wedge \mathbf{coC_=L} = \mathbf{C_=L} = \mathbf{SPL}$, because **SPL** is closed under complement. As a consequence, the rank of a matrix can be verified in **SPL**. Hence, there is a function $g \in \mathbf{GapL}$ such that for a given matrix A of order n and for all $1 \leq i \leq n$ we have the following implications

$$\begin{aligned} \mathrm{rank}(A) = i &\implies g(A,i) = 1, \\ \mathrm{rank}(A) \neq i &\implies g(A,i) = 0. \end{aligned}$$

It follows that

$$\mathrm{rank}(A) = \sum_{i=1}^{n} i\ g(A,i).$$

Since g is computable in **GapL**, so is the rank. □

As mentioned in [AO96], **NL** seems unlikely to be a sub-class of **GapL** or **SPL**. Since **NL** is contained in $\mathbf{C_=L}$ and **PL**, the assumption that $\mathbf{C_=L}$, or **PL**, collapses to **SPL** doesn't have much intuitive clout. Thus, Theorem 5.2.1 states another fact that there is no reason to believe that the rank function is computable in **GapL**.

The following theorem shows a necessary and sufficient condition for the collapse $\mathbf{C_=L} = \mathbf{coC_=L}$.

Theorem 5.2.2 *Let g and h be integer functions such that* $\mathrm{rank}(A) = \frac{g(A)}{h(A)}$*, for any integer matrix A. Then*

$$g, h \in \mathbf{GapL} \iff \mathbf{C_=L} = \mathbf{coC_=L}.$$

Proof. Suppose the functions g and h are computable in **GapL**. Define $f(A,k) = g(A) - k\ h(A)$ for any integer matrix A and for any integer $k \geq 0$. Then $f \in \mathbf{GapL}$ and we have the following equivalence

$$\mathrm{rank}(A) = k \iff f(A,k) = 0.$$

It follows that the rank of a matrix can be verified in $\mathbf{C_=L}$. Therefore, $\mathbf{C_=L} = \mathbf{coC_=L}$.

Conversely, if $\mathbf{C_=L} = \mathbf{coC_=L}$, then verifying the rank of a matrix is complete for $\mathbf{coC_=L}$. It follows that there are matrices B_i computable in logspace such that for any matrix A of order n and for $i = 1, 2, \ldots, n$,

$$\mathrm{rank}(A) = i \iff \det(B_i) \neq 0.$$

From these equivalences we get

$$\mathrm{rank}(A) = \frac{\sum_{i=1}^{n} i\ \det(B_i)}{\sum_{i=1}^{n} \det(B_i)}. \tag{5.1}$$

Therefore the following functions are in **GapL**.

$$g(A) = \sum_{i=1}^{n} i \ \det(B_i), \qquad h(A) = \sum_{i=1}^{n} \det(B_i).$$

□

Let's say more about the formula (5.1). If $\mathbf{C_=L} = \mathbf{coC_=L}$, then the rank can be expressed as a division (without remainder) of two **GapL** values. Unfortunately, we don't know how to obtain a division-free and **GapL**-computable formula for the rank. Under assumption that such a formula exists, we get the following implication: if $\mathbf{C_=L} = \mathbf{coC_=L}$, then $\mathbf{C_=L} = \mathbf{SPL}$. Since the latter seems to be impossible, there is no reason to expect that $\mathbf{C_=L}$ is closed under complement.

We generalize the condition for the collapse $\mathbf{C_=L} = \mathbf{coC_=L}$. Recall that the rank of an $n \times n$ symmetric matrix A can be determined from the coefficients of $\chi_A(x)$ as follows. Let

$$\chi_A(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0.$$

For $0 \leq k \leq n$, $\text{rank}(A) = k$ if and only if $c_{n-k} \neq 0$ and $c_{n-k-1} = c_{n-k-2} = \cdots = c_0 = 0$. By defining

$$\boldsymbol{w} = [w_n, w_{n-1}, \cdots, w_1, w_0]^T \text{ where } w_{n-j} = \sum_{i=0}^{n-j} c_i^2, \text{ for } j = 0, 1, \ldots, n,$$

every element of $\boldsymbol{w}$ is computable in **GapL**. Let's consider the following property of $\boldsymbol{w}$: there is a number k such that $w_n, w_{n-1}, \ldots, w_{n-k+1}$ are positive and $w_{n-k} = w_{n-k-1} = \cdots = w_0 = 0$. Whereby k is exactly the rank of A. Therefore, the function rank computes the number of all positive elements appeared in $\boldsymbol{w}$. Conversely, for a given nonnegative **GapL** vector, the number of its positive elements is exactly the rank of the diagonal matrix whose diagonal is the given vector. We conclude another simple question which is equivalent to the open question $\mathbf{C_=L} \stackrel{?}{=} \mathbf{coC_=L}$: *Is it possible to compute in* **GapL** *two integers a and b such that $\frac{a}{b}$ is the number of all positive elements in a given nonnegative* **GapL***-vector?*

As stated in Section 3.2.3, the degree of the minimal polynomial is computationally equivalent to matrix rank. We can extend the mentioned conditions concerning the rank to ones regarding the degree of the minimal polynomial. For example, the degree of the minimal polynomial is **GapL** computable if and only if rank $\in$ **GapL**.

5.3 Conditions for further collapse of PL

Recall that the signature of a symmetric integer matrix A is defined by signature$(A) = i_+(A) - i_-(A)$, where $i_+(A)$ and $i_-(A)$ are the number of positive and negative eigenvalues of A, respectively. The function signature is very useful for testing the congruence of two symmetric matrices A and B, i.e. whether $i(A) = i(B)$. The set of congruent matrices is denoted by CONGRUENCE. By Corollary 4.1.9 (see page 73), CONGRUENCE is in **PL** and it is hard for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. The signature and the number i_+ of a symmetric integer matrix are related to the collapse of **PL** by the following theorem.

Theorem 5.3.1 $\mathbf{PL} = \mathbf{SPL} \iff$ signature $\in \mathbf{GapL} \iff i_+ \in \mathbf{GapL}$.

Proof. Suppose $\mathbf{PL} = \mathbf{SPL}$. Let A be a symmetric matrix of order n. For $j = 1, 2, \ldots, n$, we define matrices B_j, C_j, D_j, E_j of order $n + j$ as follows

$$B_j = \begin{bmatrix} A & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}, \quad D_j = \begin{bmatrix} A & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix},$$
$$C_j = -B_j, \quad E_j = -D_j.$$

Then we have

$$\begin{aligned} \text{signature}(A) = j &\iff (B_j, C_j) \in \text{CONGRUENCE}, \\ \text{signature}(A) = -j &\iff (D_j, E_j) \in \text{CONGRUENCE}. \end{aligned}$$

By Corollary 4.1.9, CONGRUENCE is in **PL** and it is hard for $\mathbf{AC}^0(\mathbf{C}_=\mathbf{L})$. Since $\mathbf{PL} = \mathbf{SPL}$, CONGRUENCE is complete for **SPL**. Hence, there are functions $f, g \in \mathbf{GapL}$ such that

$$\begin{aligned} (B_j, C_j) \in \text{CONGRUENCE} &\implies f(B_j, C_j) = 1, \\ (B_j, C_j) \notin \text{CONGRUENCE} &\implies f(B_j, C_j) = 0, \end{aligned}$$

and analogously for (D_j, E_j) and the function g.

Therefore, we can write

$$\text{signature}(A) = \sum_{j=1}^{n} j f(B_j, C_j) - \sum_{j=1}^{n} j g(D_j, E_j).$$

This shows that signature$(A) \in \mathbf{GapL}$.

From $\text{rank}(A) = i_+(A) + i_-(A)$ and $\text{signature}(A) = i_+(A) - i_-(A)$ we get

$$i_+(A) \;=\; \frac{1}{2}(\text{rank}(A) + \text{signature}(A)).$$

By Theorem 5.2.1 rank $\in$ **GapL** because $\mathbf{C_=L} = \mathbf{SPL}$. Therefore, i_+ is computable in **GapL**.

Conversely, if i_+ is in **GapL**, then the function signature is anyway in **GapL** because $\text{signature}(A) = i_+(A) - i_+(-A)$. (Note that the functions i_+ and i_- are equivalent because $i_+(A) = i_-(-A)$.) Therefore, it remains to show that if signature $\in$ **GapL** then $\mathbf{PL} = \mathbf{C_=L} = \mathbf{SPL}$.

Observe that the function rank is in **GapL** because $\text{rank}(A) = \text{rank}(A^TA) = \text{signature}(A^TA)$. It follows that $\mathbf{AC^0(C_=L)} = \mathbf{SPL}$. Now the function $2i_+(A)$ is in **GapL** because $2i_+(A) = \text{rank}(A) + \text{signature}(A)$. We further observe that $\det(A) > 0$ if and only if $i_+(B) = 1$, where $B = [\det(A)]$. Since **GapL** is closed under composition, testing $i_+(B) = 1$ can be done in $\mathbf{C_=L}$. Hence, $\mathbf{PL} = \mathbf{C_=L} = \mathbf{SPL}$. □

We show another condition for the collapse **PL** = **SPL**. In the following theorem, by $\text{abs}(f)$ we denote the absolute value of f.

Theorem 5.3.2 $\mathbf{PL} = \mathbf{SPL} \iff \text{abs}(f) \in \mathbf{GapL}$, *for every* **GapL***-function* f.

Proof. Suppose $\mathbf{PL} = \mathbf{SPL}$. Define $\text{abs}(f) = (2g - 1)f$ where f is any **GapL** function and g is the characteristic function for deciding whether f is nonnegative, i.e.

$$g = \begin{cases} 0\ , & \text{if } f < 0, \\ 1\ , & \text{if } f \geq 0. \end{cases}$$

Since g is in **GapL**, so is $\text{abs}(f)$.

Conversely, suppose $\text{abs}(f) \in \mathbf{GapL}$ for every $f \in \mathbf{GapL}$. Then the following functions g and h are in **GapL**:

$$\begin{aligned} g &= \text{abs}(f+1) - \text{abs}(f), \text{ and} \\ h &= \binom{g+1}{2}. \end{aligned}$$

Therefore, we get

$$h = \begin{cases} 0\ , & \text{if } f < 0, \\ 1\ , & \text{if } f \geq 0 \end{cases}$$

as the characteristic function for testing if $f \geq 0$. It follows that $\mathbf{PL} = \mathbf{SPL}$. □

At the end of this chapter we show the following theorem for collapsing $\mathbf{PL} = \mathbf{C_=L}$.

Theorem 5.3.3 *Suppose A is a square integer matrix having no opposite non-zero eigenvalues.* $\mathbf{PL} = \mathbf{C_=L}$ *if and only if $i_+(A)$ can be expressed by r/s where r and s are computable in* **GapL**.

Proof. Recall from Corollary 4.1.8 that verifying the inertia of a matrix with no opposite non-zero eigenvalues it is complete for **PL**.

If $\mathbf{PL} = \mathbf{C_=L}$, then verifying $i_+(A)$ can be done in $\mathbf{coC_=L}$, for every $n \times n$ matrix A with no opposite non-zero eigenvalues. It follows that in logspace we can compute B_k such that

$$i_+(A) = k \iff \det(B_k) \neq 0, \text{ for } k = 0, 1, \ldots, n.$$

Thus we get a division of two **GapL** values for $i_+(A)$ as follows

$$i_+(A) = \frac{\sum_{i=0}^{n} i \ \det(B_i)}{\sum_{i=0}^{n} \det(B_i)}.$$

Conversely, if $i_+(A)$ can be expressed by r/s where r and s are computable in **GapL**, then it is clearly that verifying the inertia of A can be done in $\mathbf{C_=L}$. Therefore, $\mathbf{PL} = \mathbf{C_=L}$. □

Conclusions and Open Problems

The following table summarizes some bounds on the complexity of problems in linear algebra that have been considered in this thesis.

Problem	hard for	contained in	see
CharPolynomial	**GapL**	**GapL**	Section 3.1.1
v-CharPolynomial	$\mathbf{C_=L}$	$\mathbf{C_=L}$	Section 3.1.2
MinPolynomial	**GapL**	$\mathbf{TC^0(GapL)}$	
v-MinPolynomial	$\mathbf{C_=L}$	$\mathbf{C_=L \wedge coC_=L}$	Section 3.2.1
InvSystem	**GapL**	$\mathbf{NC^2}$	and 3.2.2
v-InvSystem	$\mathbf{C_=L}$	$\mathbf{AC^0(C_=L)}$	
DegMinPol$_\leq$	$\mathbf{C_=L}$	$\mathbf{C_=L}$	
DegMinPol$_=$	$\mathbf{C_=L \wedge coC_=L}$	$\mathbf{C_=L \wedge coC_=L}$	
DegMinPol	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	
v-DegMinPol	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	Section 3.2.3
EqDegMinPol	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	
EqMinPolynomial	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	
EqCTMinPol	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	
Similarity	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	
Diagonalizable	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	Section 3.3
SimDiagonalizable	$\mathbf{AC^0(C_=L)}$	$\mathbf{AC^0(C_=L)}$	
Inertia	**PL**	**PL**, or $\mathbf{NC^2}$	Section 4.1
v-Inertia	**PL**	**PL**, or $\mathbf{NC^2}$	
PosStable	**PL**	**PL**	
PosSemistable	**PL**	**PL**	Section 4.2
Congruence	$\mathbf{AC^0(C_=L)}$	**PL**	
UPM	**NL**	$\mathbf{C_=L}$	Section 5.1

Some necessary and sufficient conditions for collapsing the $\mathbf{C_=L}$-Hierarchy and the class $\mathbf{PL}$ have been presented in Section 5.2 and 5.3.

(1) $\mathbf{C_=L} = \mathbf{SPL} \iff \text{rank} \in \mathbf{GapL}$.

(2) $\mathbf{C_=L} = \mathbf{coC_=L} \iff g, h \in \mathbf{GapL}$, where g and h are integer functions such that $\text{rank}(A) = g(A)/h(A)$.

(3) $\mathbf{PL} = \mathbf{SPL} \iff \text{signature} \in \mathbf{GapL} \iff i_+ \in \mathbf{GapL}$.
(signature and i_+ are functions for symmetric integer matrices).

(4) $\mathbf{PL} = \mathbf{C_=L} \iff g, h \in \mathbf{GapL}$, where g and h are integer functions such that $i_+(A) = g(A)/h(A)$, for every matrix A having no opposite non-zero eigenvalues.

The conditions in (1) and (2) can be extended to the corresponding conditions concerning the degree of the minimal polynomial of a matrix, simply by substituting the function $\deg(\mu)$ into the function rank. Two other weaker conditions for the collapse $\mathbf{C_=L} = \mathbf{coC_=L}$ are given by Corollary 3.2.18, and 3.2.19. Obviously, these conditions offer a new insight into the open question $\mathbf{C_=L} \stackrel{?}{=} \mathbf{coC_=L}$. However, we don't know how to prove or disprove one of the following conjectures:

(a) $\mathbf{C_=L} = \mathbf{coC_=L} \iff \text{rank} \in \mathbf{GapL}$. (In particular, if it is true, then there are some interesting consequences. For instance, if $\mathbf{C_=L} = \mathbf{coC_=L}$ then all classes between $\mathbf{C_=L}$ and $\mathbf{PL}$ are equal to $\mathbf{SPL}$.)

(b) If $\mathbf{C_=L} = \mathbf{coC_=L}$ then $\mathbf{C_=L} \subseteq \oplus\mathbf{L}$.

(c) $\mathbf{C_=L} = \mathbf{coC_=L} \iff \mathbf{PL} = \mathbf{C_=L}$.

(d) If $\mathbf{PL} = \mathbf{C_=L}$ then $\mathbf{PL} \subseteq \oplus\mathbf{L}$.

(e) $\mathbf{PL} = \mathbf{C_=L} \iff \mathbf{PL} = \mathbf{SPL}$.

An important task for further research is to close the gap between the lower and the upper bound where it doesn't match in the above table.

Bibliography

[AAM03] E. Allender, V Arvind, and M. Mahajan. Arithmetic complexity, Kleene closure, and formal power series. *Theory of Computing Systems*, 36(4):303–328, 2003.

[ABO] E. Allender and M. Ben-Or. Electronic communication, 2001.

[ABO99] E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8:99–126, 1999.

[ÀJ93] C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107:3–30, 1993.

[All97] E. Allender. A clarification concerning the #**L** hierarchy. Available at ftp://ftp.cs.rutgers.edu/pub/allender/lh.pdf, 1997.

[All02] E. Allender. Personal communication, 2002.

[AM87] S. Athloen and R. McLaughlin. Gauss-jordan reduction: A brief history. *American Mathematical Monthly*, 94:130–142, 1987.

[AO96] E. Allender and M. Ogihara. Relationship among **PL**, #**L**, and the determinant. *RAIRO-Theoretical Informatics and Applications*, 30:1–21, 1996.

[ARZ99] E. Allender, K. Reinhardt, and S. Zhou. Isolating, matching, and counting: uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[Bal91] J. L. Balcázar. Adaptive logspace and depth-bounded reducibilities. In *Structure in Complexity Theory Conference (CoCo)*, pages 240–254. IEEE Computer Society Press, 1991.

[BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity I.* Springer-Verlag, 1988.

[BDG91] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity II.* Springer-Verlag, 1991.

[Ber84] S. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150, 1984.

[BF97] R. Beigel and B. Fu. Circuits over **PP** and **PL**. In *12th IEEE Conference on Computational Complexity (CCC)*, pages 24–35. IEEE Computer Society Press, 1997.

[BG77] C. Byrnes and M. Gauger. Characteristic free, improved decidability criteria for the similarity problem. *Linear and Multilinear Algebra*, 5:153–158, 1977.

[BR91] R. Brualdi and H. Ryser. *Combinatorial matrix theory.* Cambridge University Press, 1991.

[BvzGH82] A. Borodin, J. von zur Gathen, and J. E. Hopcroft. Fast parallel matrix and gcd computations. *Information and Control*, 52:241–256, 1982.

[CDS80] D. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs.* Academic Press, 1980.

[Chi85] A. L. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *International Conference on Foundations of Computation Theory (FCT)*, Lecture Notes in Computer Science 199, pages 63–69. Springer-Verlag, 1985.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *3rd Symposium on Theory of Computing (STOC)*, pages 151–158. ACM Press, 1971.

[Coo85] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[Csa76] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976.

[CSV84] A. K. Chandra, L. Stockmeyer, and U. Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, 1984.

[Dam90] C. Damm. Problems complete for Parity-**L**. *Information Processing Letters*, 36:247–250, 1990.

[Dam91] C. Damm. DET = $L^{(\#L)}$. Technical Report Informatik-Preprint 8, Fachbereich Informatik der Humboldt Universitaet zu Berlin, 1991.

[DHK93] E. Dahlhaus, P. Hajnal, and M. Karpinski. On the parallel complexity of hamiltonian cycles and matching problem in dense graphs. *Journal of Algorithms*, 15:367–384, 1993.

[Dix79] J. Dixon. An isomorphism criterion for modules over a principal ideal domain. *Linear and Multilinear Algebra*, 8:69–72, 1979.

[DK86] E. Dahlhaus and M. Karpinski. The matching problem for strongly chordal graphs is in **NC**. Technical Report 855-CS, University of Bonn, 1986.

[Edm65] J. Edmonds. Maximum matching and a polyhedron with 0-1 vertices. *Journal of Research National Bureau of Standards*, 69:125–130, 1965.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48:116–148, 1994.

[For00] S. Fortune. Exact computation of the inertia of symmetric integer matrices. In *32nd Symposium on Theory of Computing (STOC)*, pages 556–564. ACM Press, 2000.

[Fru77] M. A. Frumkin. Polynomial time algorithms in the theory of linear diophantine equations. In *International Conference on Foundations of Computation Theory (FCT)*, Lecture Notes in Computer Science 56, pages 386–392. Springer-Verlag, 1977.

[Gan77a] F. Gantmacher. *The theory of matrices*, volume 1. AMS Chelsea Publishing, 1977.

[Gan77b] F. Gantmacher. *The theory of matrices*, volume 2. AMS Chelsea Publishing, 1977.

[GK87] D. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanent is in **NC**. In *28th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 166–172. IEEE Computer Society Press, 1987.

[Gre93] R. Greenlaw. Polynomial completeness and parallel computation. In J. H. Reif, editor, *Synthesis of parallel algorithms*, pages 901–953. Morgan Kaufmann, 1993.

[Gri83] D. Grigoriev. On the complexity of the "wild" matrix problems, of the isomorphism of algebras and graphs. *Journal of Soviet Mathematics*, 22:1285–1289, 1983.

[GZ89] M. Garzon and Y. Zalcstein. An $\mathbf{NC}^2$ algorithm for testing similarity of matrices. *Information Processing Letters*, 30:253–254, 1989.

[Hes01] W. Hesse. Division is in uniform $\mathbf{TC}^0$. In *Twenty-Eighth International Colloquium on Automata, Languages and Programming (ICALP)*, pages 104–114. Springer-Verlag, 2001.

[HJ85] R. Horn and C. Johnson. *Matrix analysis*. Cambridge University Press, 1985.

[HJ91] R. Horn and C. Johnson. *Topics in matrix analysis*. Cambridge University Press, 1991.

[HO02] L. A. Hemaspaandra and M. Ogihara. *The complexity theory companion*. Springer-Verlag, 2002.

[HT00] T. M. Hoang and T. Thierauf. The complexity of verifying the characteristic polynomial and testing similarity. In *15th IEEE Conference on Computational Complexity (CCC)*, pages 87–95. IEEE Computer Society Press, 2000.

[HT01] T. M. Hoang and T. Thierauf. The complexity of the minimal polynomial. In *26th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science 2136, pages 408–420. Springer-Verlag, 2001.

[HT02a] T. M. Hoang and T. Thierauf. The complexity of the inertia. In *22nd Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, Lecture Notes in Computer Science 2556, pages 206–217. Springer-Verlag, 2002.

[HT02b] T. M. Hoang and T. Thierauf. On the minimal polynomial of a matrix. In *8th Annual International Conference on Computing and Combinatorics (COCOON)*, Lecture Notes in Computer Science 2387, pages 37–46. Springer-Verlag, 2002.

[HT03a] T. M. Hoang and T. Thierauf. The complexity of the characteristic and the minimal polynomial. *Theoretical Computer Science*, 295:205–222, 2003.

[HT03b] T. M. Hoang and T. Thierauf. On the minimal polynomial of a matrix. Invited paper to the special issue in *International Journal of Foundations of Computer Science* of the 8th COCOON conference 2002, to appear 2003.

[IK93] D. Ierardi and D. C. Kozen. Parallel resultant computation. In J. H. Reif, editor, *Synthesis of parallel algorithms*, pages 679–720. Morgan Kaufmann, 1993.

[Imm88] N. Immerman. Nondeterministic space is closed under complement. *SIAM Journal on Computing*, 17:935–938, 1988.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KB79] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8:499–507, 1979.

[KKS90] E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra and its Applications*, 136:189–208, 1990.

[Koz91] D. C. Kozen. *The design and analysis of algorithms*. Springer-Verlag, 1991.

[KS87] E. Kaltofen and B. D. Saunders. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM Algebraic and Discrete Methods*, 8:683–690, 1987.

[KUW86] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is random **NC**. *Combinatorica*, 6(1):35–48, 1986.

[KVV85] D. C. Kozen, U. V. Vazirani, and V. V. Vazirani. **NC** algorithms for comparability graphs, interval graphs, and testing for unique perfect matchings. In *5th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 496–503. Springer-Verlag, 1985.

[KVV86] D. C. Kozen, U. V. Vazirani, and V. V. Vazirani. **NC** algorithms for comparability graphs, interval graphs, and testing for unique perfect matchings. Technical Report TR86-799, Cornell University, 1986.

[LPV81] G. Lev, M. Pippenger, and L. Valiant. A fast parallel algorithm for routing in permutation networks. *IEEE Transactions on Computers*, C-30:93–100, 1981.

[Mah03] M. Mahajan. Electronic communication, 2003.

[MSV99] M. Mahajan, P. Subramanya, and V Vinay. A combinatorial algorithm for pfaffians. In *5th Annual International Conference on Computing and Combinatorics (COCOON)*, Lecture Notes in Computer Science 1627, pages 134–143. Springer-Verlag, 1999.

[Mul87] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[MV97] M. Mahajan and V Vinay. Determinant: combinatorics, algorithms, and complexity. *Chicago Journal of Theoretical Computer Science*, 5, 1997.

[MV99] M. Mahajan and V Vinay. Determinant: old algorithms, new insights. *SIAM Journal on Discrete Mathematics*, 12(4):474–490, 1999.

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[Nef94] C. A. Neff. Specified precision root isolation is in **NC**. *Journal of Computer and System Science*, 48:429–463, 1994.

[NR96] C. A. Neff and J. H. Reif. An efficient algorithm for the complex roots problem. *Journal of Complexity*, 12:81–115, 1996.

[NTS95] N. Nisan and A. Ta-Shma. Symmetric logspace is closed under complement. *Chicago Journal of Theoretical Computer Science*, 1995.

[Ogi98] M. Ogihara. The **PL** hierarchy collapses. *SIAM Journal on Computing*, 27:1430–1437, 1998.

[OZ02] M. Ogihara and Y. Zalcstein. Testing simultaneous similarity of matrices and related problems for matrix semigroups. Technical Report GIT-CC-02-01, Georgia Institute of Technology, College of Computing, 2002.

[Pap94] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[RA00] K. Reinhardt and E. Allender. Making nondeterminism unambiguous. *SIAM Journal on Computing*, 29:1118–1131, 2000.

[RST84] W. Ruzzo, J. Simon, and M. Tompa. Space-bounded hierarchies and probabilistic computation. *Journal of Computer and System Sciences*, 28:216–230, 1984.

[Ruz81] W. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.

[RV89] M. O. Rabin and V. V. Vazirani. Maximum matching in general graphs through randomization. *Journal Algorithms*, 10:557–567, 1989.

[ST98] M. Santha and S. Tan. Verifying the determinant in parallel. *Computational Complexity*, 7:128–151, 1998.

[Sto96] A. Storjohann. Near optimal algorithms for computing Smith normal form of integer matrices. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 267–274. ACM press, 1996.

[Sto98] A. Storjohann. An $O(n^3)$ algorithm for Frobenius normal form. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 101–105. ACM press, 1998.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.

[Tod91] S. Toda. Counting problems computationally equivalent to the determinant. Technical Report CSIM 91-07, Dept. of Computer Science and Information Mathematics, University of Electro-Communications, Chofu-shi, Tokyo 182, Japan, 1991.

[Tur48] A. M. Turing. Rounding-off errors in matrix processes. *Quarterly Journal of Mechanics and Applied Mathematics*, 1:287–308, 1948.

[Tut47] W. T. Tutte. The factorization of linear graphs. *London Math Society*, 22:107–111, 1947.

[Val79a] L. Valiant. Completeness classes in algebra. In *11th Symposium on Theory of Computing (STOC)*, pages 249–261. ACM Press, 1979.

[Val79b] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Val79c] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.

[Val92] L. Valiant. Why is Boolean complexity theory difficult. In M. S. Paterson, editor, *Boolean Function Complexity*, London Mathematical Society Lecture Notes Series 169. Cambridge University Press, 1992.

[Vaz93] V. V. Vazirani. Parallel graph matching. In J. H. Reif, editor, *Synthesis of parallel algorithms*, pages 783–811. Morgan Kaufmann, 1993.

[Vil97] G. Villard. Fast parallel algorithms for matrix reduction to normal forms. *Applicable Algebra in Engineering Communication and Computing (AAECC)*, 8:511–537, 1997.

[Vin91] V Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *6th IEEE Conference on Structure in Complexity Theory (CCC)*, pages 270–284. IEEE Computer Society Press, 1991.

[vNG47] J. von Neumann and H. H. Goldstine. Numerical inverting of matrices of high order. *Bulletin of American Mathematical Society*, 53:1021–1099, 1947.

[vzG93] J. von zur Gathen. Parallel linear algebra. In J. H. Reif, editor, *Synthesis of parallel algorithms*, pages 573–617. Morgan Kaufmann, 1993.

[vzGG99] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999.

[Zei85] D. Zeilberger. A combinatorial approach to matrix algebra. *Discrete Mathematics*, 56:61–72, 1985.

Deutsche Zusammenfassung

Diese Dissertation befasst sich mit der Komplexität von einigen fundamentalen Problemen aus der linearen Algebra. Die untersuchten Probleme, die das charakteristische Polynom, das Minimalpolynom, das System der Invariantenteiler und die Inertia einer Matrix betreffen, werden nach entsprechenden Komplexitätsklassen geordnet.

Bevor wir einen Überblick über Resultate der vorliegenden Arbeit geben, erwähnen wir kurz die Namen der Klassen, die durch linear algebraische Probleme charakterisiert sind. Álvarez and Jenner [ÀJ93] definierten #**L** als die Klasse von allen Funktionen, die die Anzahl der akzeptierenden Berechnungen einer nichtdeterministischen logarithmisch platzbeschränkten Turingmaschine (kurz: **NL** Turingmaschine) berechnen. Dann definiert man **GapL** als die Klasse von allen Funktionen, die als Differenzen von #**L**-Funktionen darstellbar sind. Anhand von **GapL** kann man weitere Klassen definieren. $\mathbf{C_=L}$ (*Exact Counting in Logspace*) ist die Klasse, in der alle Verifikationen von **GapL**-Funktionen sind, und in **PL** (*Probabilistic Logspace*) sind alle Vorzeichenberechnungen von **GapL**-Funktionen. Die Berechnung der Determinante ist zum Beispiel vollständig für **GapL** [Tod91, Dam91, Vin91, Val92], folglich ist die Verifikation der Determinante vollständig für $\mathbf{C_=L}$, und das Entscheidungsproblem, ob die Determinante einer Matrix positiv ist, ist vollständig für **PL**. Ausserdem interessieren wir uns für den $\mathbf{AC}^0$-Abschluss von $\mathbf{C_=L}$, $\mathbf{AC}^0(\mathbf{C_=L})$, welcher durch die bitweise Rangberechnung charakterisiert ist, und den $\mathbf{TC}^0$-Abschluss von **GapL**, $\mathbf{TC}^0(\mathbf{GapL})$, der die Division von **GapL**-Werten enthält.

Für das charakteristische Polynom beweisen wir, dass die Verifikation aller ihrer Koeffizienten vollständig für $\mathbf{C_=L}$ ist.

Die Berechnung sowie die Verifikation des Minimalpolynoms werden in dieser Arbeit studiert. Wir zeigen, dass sich das Minimalpolynom in $\mathbf{TC}^0(\mathbf{GapL})$ berechnen lässt und es hart für **GapL** ist, und dass die Verifikation des Minimalpolynoms in der zweiten Stufe der Booleschen Hierarchie über $\mathbf{C_=L}$ liegt und sie hart für $\mathbf{C_=L}$ ist. Unsere Untersuchung wird auf die Invariantenteiler

erweitert. Wir zeigen, dass die Verifikation der Invariantenteiler in $\mathbf{AC}^0(\mathbf{C_=L})$ liegt und sie hart für $\mathbf{C_=L}$ ist. Einige interessante Probleme hinsichtlich des Grads sowie der Konstante des Minimalpolynoms werden definiert und genau analysiert. Dazu zeigen wir, dass die betrachteten Klassen durch diese Probleme charakterisiert werden können. Insbesondere wird bewiesen, dass die bitweise Berechnung des Grads des Minimalpolynoms vollständig für $\mathbf{AC}^0(\mathbf{C_=L})$ ist, d.h. dass die Berechnung des Rangs und die Berechnung des Grads des Minimalpolynoms äquivalent sind.

Als eine Konsequenz der Resultate über das Minimalpolynom können wir die Komplexität zweier klassischer Probleme bestimmen. Nämlich sind das Entscheidungsproblem, ob zwei gegebene Matrizen ähnlich sind, und das Entscheidungsproblem, ob eine gegebene Matrix diagonalisierbar ist, vollständig für $\mathbf{AC}^0(\mathbf{C_=L})$.

Die Inertia einer $n \times n$ Matrix A ist definiert als das Tripel $i(A) = (i_+(A), i_-(A), i_0(A))$, wobei $i_+(A)$, $i_-(A)$, und $i_0(A)$ die jeweilige Anzahl der Eigenwerte von A mit positivem, negativem, und 0-Realteil sind. Für bestimmte Matrizen zeigen wir, dass die Berechnung sowie die Verifikation der Inertia vollständig für $\mathbf{PL}$ sind. Wir zeigen weiter, dass das Entscheidungsproblem, ob eine gegebene Matrix positiv stabil ist, auch vollständig für $\mathbf{PL}$ ist, und dass das Entscheidungsproblem, ob symmetrische Matrizen kongruent sind, in $\mathbf{PL}$ liegt und es hart für $\mathbf{AC}^0(\mathbf{C_=L})$ ist.

Ein sehr wichtiges Ziel dieser Dissertation ist die Untersuchung der Beziehungen zwischen Komplexitätsklassen, in denen Probleme aus der linearen Algebra liegen. Wir versuchen einerseits die betrachteten Komplexitätsklassen sowie ihre Eigenschaften gründlich zu erläutern, andererseits beweisen wir einige notwendige und hinreichende Bedingungen für die Beziehungen wie $\mathbf{C_=L} = \mathbf{coC_=L}$, $\mathbf{C_=L} = \mathbf{SPL}$, $\mathbf{PL} = \mathbf{C_=L}$ oder $\mathbf{PL} = \mathbf{SPL}$, wobei $\mathbf{coC_=L}$ das Komplement von $\mathbf{C_=L}$ ist und $\mathbf{SPL}$ (*Stoic Probabilistic Logspace*) die Klasse bestehend aus Sprachen mit charakteristischen Funktionen in $\mathbf{GapL}$ ist. Unter dem komplexitätstheoretischen Aspekt sind diese Resultate möglicherweise interessant für eine potenzielle Problemlösung der offenenen Frage, ob $\mathbf{C_=L}$ unter Komplement abgeschlossen ist.