

The Complexity of the Inertia and some Closure Properties of GapL *

Thanh Minh Hoang
Abt. Theoretische Informatik
Universität Ulm
89069 Ulm, Germany

Thomas Thierauf
FB Elektronik und Informatik
FH Aalen
73430 Aalen, Germany

{hoang,thierauf}@informatik.uni-ulm.de

Abstract

The inertia of an $n \times n$ matrix A is defined as 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. It is known that the inertia of a large class of matrices can be determined in PL (probabilistic logspace). However, the general problem, whether the inertia of an arbitrary integer matrix is computable in PL , was an open question. In this paper we give a positive answer to this question and show that the problem is complete for PL .

As consequences of this result we show necessary and sufficient conditions that certain algebraic functions like the rank or the inertia of an integer matrix can be computed in GapL .

1 Introduction

The problem of computing the inertia of a matrix plays an important role in a number of applied fields, in particular in control theory and in robotics. Matrix inertia, a fundamental topic in linear algebra, has been studied by many researchers (for more detail, see e.g. [Gan77], Chapter XV) a long time ago.

We are interested in the *computational complexity* of the inertia of a matrix, which is the main topic of this paper. It is well known that many problems from linear algebra can be solved within certain *logspace counting classes*, all of which are contained in the parallel complexity class (uniform) NC^2 .

Maybe the most important class is GapL (studied by [AO96]) that seems to capture the complexity of a lot of problems from linear algebra quite naturally. GapL is the extension of $\#\text{L}$ (first studied by [AJ93]) in the same way as $\#\text{P}$ [Val79] can be extended to GapP [FFK94] in the polynomial time setting. GapL is characterized precisely by the determinant of an integer matrix [Ber84, Dam91, Tod91, Vin91, Val92, MV97].

There are some interesting classes of decision problems based on GapL .

- Verifications of GapL -functions define the class C=L (*exact counting in logspace*). The singularity problem, i.e. the problem of deciding whether the determinant of a matrix is zero, is complete for C=L .
- Inequalities on GapL -functions define the complexity class PL (*probabilistic logspace*), in which one has to decide whether the value of a GapL function on an input is positive. For example, the problem of testing if the determinant of a matrix is positive, is complete for PL .

In a preceding paper [HT02a], we have studied the complexity of the inertia. Using Routh-Hurwitz Theorem for computing the inertia, we have proved that the inertia can be verified in PL for many cases, in particular for symmetric matrices and for matrices with no opposite nonzero eigenvalues. However, it remained open in [HT02a] whether the inertia of an *arbitrary* integer matrix is in PL . In Section 3 we give a positive answer to this question. The inertia has been shown to be hard for PL under logspace many-one reductions [HT02a]. It follows that the inertia is complete for PL .

We want to mention that Neff and Reif [Nef94, NR96] have developed a method to approximate the roots of a poly-

*Supported by DFG grants Th 472/3-2 and Scho 302/7-1.

nomial. However, it is not clear how to compute the inertia of a matrix by approximating the roots of its characteristic polynomial. The problem is that one must be able to decide whether a root lies on an axis or not.

The second part of the paper is motivated by the question whether one can improve the upper bounds on some functions like the rank or the inertia to **GapL**. Our main results are that these questions are equivalent to the collapse of certain complexity classes.

We show in Section 4.1 that the rank of a matrix can be computed in **GapL** if and only if $\mathbf{C=L} = \mathbf{SPL}$, where **SPL** is the class of all sets with the characteristic function in **GapL**. As a consequence of the results about the inertia, we show in Section 4.2 that the inertia of a matrix can be computed in **GapL** if and only if $\mathbf{PL} = \mathbf{SPL}$. Note that $\mathbf{NL} \subseteq \mathbf{C=L}$ and $\mathbf{SPL} \subseteq \oplus\mathbf{L}$. Hence, as a corollary of our results we get: if the rank or the inertia of a matrix can be computed in **GapL** then $\mathbf{NL} \subseteq \oplus\mathbf{L}$ and $\mathbf{C=L}$ is closed under complement. Both consequences are open problems right now.

We also consider a relaxed version of the above question. **GapL** is not known to be closed under division. Hence it is natural to ask whether we can write the rank or the inertia of a matrix as a quotient of two **GapL**-functions. We show in Section 4.1 that this is true for the rank of a matrix if and only if $\mathbf{C=L} = \mathbf{coC=L}$. In Section 4.2 we show that this is true for the inertia of a matrix if and only if $\mathbf{PL} = \mathbf{C=L}$.

Finally, in Section 4.3, we characterize the case that the absolute value of any **GapL**-function can be computed in **GapL** too.

2 Preliminaries

Complexity Classes. For a nondeterministic Turing machine M on input x , we denote the number of accepting and rejecting computation paths by $acc_M(x)$ and $rej_M(x)$, respectively. The difference of these two quantities is denoted by $gap_M(x)$. That is, $gap_M(x) = acc_M(x) - rej_M(x)$. The complexity class $\#\mathbf{L}$ consists of all functions f such that $f = acc_M$, for some nondeterministic logspace Turing machine. Similarly, The class **GapL** consists of all functions f such that $f = gap_M$, for some nondeterministic logspace Turing machine. Based on these function classes, we define the following counting complexity classes [AO96, ARZ99].

$$\begin{aligned} \mathbf{C=L} &= \{ S \mid \exists f \in \mathbf{GapL}, \forall x : x \in S \iff f(x) = 0 \}, \\ \mathbf{PL} &= \{ S \mid \exists f \in \mathbf{GapL}, \forall x : x \in S \iff f(x) > 0 \}, \\ \mathbf{SPL} &= \{ S \mid \chi_S \in \mathbf{GapL} \}, \end{aligned}$$

where χ_S is the characteristic function of set S . It is known that

$$\mathbf{SPL} \subseteq \mathbf{C=L} \subseteq \mathbf{PL} \subseteq \mathbf{NC}^2.$$

Also we have $\mathbf{NL} \subseteq \mathbf{C=L}$.

Counting logspace hierarchies over these classes were defined in [AO96]. We list some properties of these classes.

- It has been shown in [ABO99] that the *Exact Counting Logspace Hierarchy* (over $\mathbf{C=L}$) collapses to $\mathbf{L}^{\mathbf{C=L}} = \mathbf{AC}^0(\mathbf{C=L})$, and that one of the complete problems for this hierarchy is the problem of computing one bit of the rank of a matrix. Note that $\mathbf{AC}^0(\mathbf{C=L})$ is called the \mathbf{AC}^0 -closure of $\mathbf{C=L}$, it is the class of problems \mathbf{AC}^0 -reducible to the sets of $\mathbf{C=L}$.
- The *Probabilistic Logspace Hierarchy* (over \mathbf{PL}) collapses to \mathbf{PL} by the fact $\mathbf{AC}^0(\mathbf{PL}) = \mathbf{NC}^1(\mathbf{PL}) = \mathbf{PL}$ [Ogi98, BF00]. That is, \mathbf{PL} is closed under \mathbf{AC}^0 - and \mathbf{NC}^1 -reductions. In particular, \mathbf{PL} is closed under union, intersection and complement.
- $\mathbf{C=L}$ is closed under union and intersection. Whether $\mathbf{C=L}$ is closed under complement is an open problem.
- $\mathbf{SPL}^{\mathbf{SPL}} = \mathbf{SPL}$. In particular, \mathbf{SPL} is closed under union, intersection and complement.

Complete Problems. Logspace counting classes are interesting because of the complete problems therein. We give some examples from linear algebra for these classes. When nothing else is said, by matrices we mean square integer matrices of order n .

Problems complete for **GapL** are to compute one element of the m -th power of a matrix and the determinant [Tod91, Dam91, Vin91, Val92].

The singularity problem, i.e. the set

$$\mathbf{SINGULARITY} = \{ A \mid \det(A) = 0 \},$$

is complete for $\mathbf{C=L}$. More general, the sets

$$\begin{aligned} \mathbf{V-POWELEM} &= \{ (A, a, m) \mid (A^m)_{1,n} = a \}, \\ \mathbf{V-DET} &= \{ (A, a) \mid \det(A) = a \}, \text{ and} \\ \mathbf{RANK}_{<} &= \{ (A, r) \mid \text{rank}(A) < r \} \end{aligned}$$

are also complete for $\mathbf{C=L}$. Consequently

$$\mathbf{RANK}_{\geq} = \{ (A, r) \mid \text{rank}(A) \geq r \}$$

is complete for $\mathbf{coC=L}$. The verification of the rank can be written as the intersection of a set in $\mathbf{C=L}$ and in $\mathbf{coC=L}$:

$$\begin{aligned} \mathbf{V-RANK} &= \{ (A, r) \mid \text{rank}(A) = r \} \\ &= \mathbf{RANK}_{<} \cap \mathbf{RANK}_{\geq}. \end{aligned}$$

This means that $\mathbf{V-RANK} \in \mathbf{C=L} \wedge \mathbf{coC=L}$. Moreover, it is complete for this class. The problem of computing (one bit of) the rank, i.e. the set

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

is a complete problem for $\mathbf{AC}^0(\mathbf{C=L})$ [ABO99].

Since \mathbf{PL} is the class of \mathbf{AC}^0 -reducible to the problem of computing the high-order bit of the determinant, the set

$$\text{POSDET} = \{ A \mid \det(A) > 0 \}$$

is complete for \mathbf{PL} . Furthermore, \mathbf{PL} is characterized by some problems related to matrix inertia [HT02a].

Recall that the *inertia* of an $n \times n$ matrix A is defined as the triple $i(A) = (i_+(A), i_-(A), i_0(A))$, 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. We can define (with respect to some fixed coding) the problem of computing the k -th bit of the inertia as follows:

$$\text{INERTIA} = \{ (A, k, b) \mid \text{the } k\text{-th bit of } i(A) \text{ is } b \}.$$

The verification of the inertia is the set

$$\text{V-INERTIA} = \{ (A, p, n, z) \mid i(A) = (p, n, z) \}.$$

It has been shown by [HT02a] that INERTIA and V-INERTIA are hard for \mathbf{PL} under logspace many-one reductions, and these problems are located in \mathbf{PL} when A is symmetric or A has no opposite nonzero-eigenvalues.

Note: INERTIA as defined above is a decision problem. When we say that *the inertia is in \mathbf{PL}* , we actually refer to the decision problem INERTIA (and not to a function computing the inertia).

Closure Properties of \mathbf{GapL} . To analyze the complexity of the inertia later on, we sum up some closure properties of the considered classes.

Theorem 2.1 [AO96] *Let $f \in \mathbf{GapL}$. The following functions are in \mathbf{GapL} as well:*

1. $f(g(\cdot))$, for any $g \in \mathbf{FL}$,
2. $\sum_{i=0}^{2^{|x|^c}} f(x, i)$, for any constant c ,
3. $\prod_{i=0}^{|x|^c} f(x, i)$, for any constant c ,
4. $\left(\frac{f(x)}{g(x)}\right)$, for any $g \in \mathbf{FL}$ such that $g(x) = O(1)$.

The first property has been improved considerably. Essentially, \mathbf{GapL} is closed under a restrictive kind of composition as follows.

Theorem 2.2 [AAM03] *The determinant of a matrix having \mathbf{GapL} -computable elements can be computed in \mathbf{GapL} .*

For a given integer matrix A and an positive integer m , each element of the power-matrix A^m is known to be equal to a sum of weighted (s, t) -paths in a directed graph H constructed easily from A and m . Therefore, one can show the fact that each element of A^m is computable in \mathbf{GapL} when the elements of A are computable in \mathbf{GapL} .

Parallel Polynomial GCD Computation. In the computation of the inertia that we present in Section 3, we need to compute the gcd and the division of some univariate polynomials. Parallel algorithms for polynomial gcd and polynomial division are known [BvzGH82]. There are excellent textbooks (see for example: Kozen [Koz91], or Ieradi and Kozen [IK93]) where these algorithms are explained in detail.

We consider the polynomial gcd computation in parallel. For univariate polynomials with leading coefficients different from zero:

$$\begin{aligned} p(x) &= a_m x^m + \dots + a_0, \text{ and} \\ q(x) &= b_n x^n + \dots + b_0, \quad n \leq m, \end{aligned}$$

the *Sylvester matrix* is defined as a matrix of order $n + m$, where n and m columns are taken from the coefficients of p and q , respectively. The following Sylvester matrix is an example for $m = 4$ and $n = 2$:

$$S = \begin{pmatrix} a_4 & 0 & b_2 & 0 & 0 & 0 \\ a_3 & a_4 & b_1 & b_2 & 0 & 0 \\ a_2 & a_3 & b_0 & b_1 & b_2 & 0 \\ a_1 & a_2 & 0 & b_0 & b_1 & b_2 \\ a_0 & a_1 & 0 & 0 & b_0 & b_1 \\ 0 & a_0 & 0 & 0 & 0 & b_0 \end{pmatrix}$$

Let

$$g = \text{gcd}(p, q) = x^d + c_{d-1}x^{d-1} + \dots + c_0.$$

(W.l.o.g. we can choose g to be monic). For $1 \leq e \leq n$, let $S^{(e)}$ be the matrix obtained from S by deleting the last $2e$ rows, the last e columns of coefficients of p , and the last e columns of coefficients of q , and let $\mathbf{c}^{(e)}$ be the vector $(0, 0, \dots, 0, 1)^T$ of length $(m + n - 2e)$. Then the parallel computation of g can be done by the following two steps:

1. determine the degree of g : this is the value d such that $\det(S^{(d)}) \neq 0$ and $\det(S^{(e)}) = 0$ for all $e < d$;
2. compute the vector of the coefficients of g by the product $S_d \mathbf{x}_0$, where S_d is the matrix obtained from S by deleting the last d columns of coefficients of p , and the last d columns of coefficients of q , and \mathbf{x}_0 is a solution of $S^{(d)} \mathbf{x} = \mathbf{c}^{(d)}$.

For the complexity of the above computation of $\text{gcd}(p, q)$ we observe that:

- Step 1 can be done in $\mathbf{C}=\mathbf{L} \wedge \mathbf{coC}=\mathbf{L}$.
- In step 2, observe that $S^{(d)}\mathbf{x} = \mathbf{c}^{(d)}$ has the unique solution $\mathbf{x}_0 = (S^{(d)})^{-1} \mathbf{c}^{(d)}$ which is the last column of the inverse $(S^{(d)})^{-1}$. Therefore, the coefficients c_{d-1}, \dots, c_0 of g can be expressed by rational form a/b where $b = \det(S^{(d)})$ and all a 's are certain determinants.

As a consequence of the parallel polynomial gcd computation, there are parallel algorithms for the polynomial division with remainder, which are based on computing determinants (for more detail: see [IK93], Algorithm 15.2). It is not hard to argue that by writing the coefficients of p/q in the rational form a/b where we get the numbers a and b by some determinant computations.

3 Computing the Inertia in PL

We start by describing the Routh-Hurwitz method in Section 3.1, In Section 3.2 we show how it can be used to compute the inertia of matrices, where the associated Routh-Hurwitz matrix is regular. The main part is Section 3.3 where we show how to compute the inertia in the singular case in PL.

3.1 The Routh-Hurwitz Theorem

Let A be an $n \times n$ matrix. The eigenvalues of A are the roots of the characteristic polynomial $\chi_A(x) = \det(xI - A)$. For the computation of $i(A)$, the inertia of A , it suffices to compute $i_+(A)$ because we have $i_-(A) = i_+(-A)$ and $i_0(A) = n - i_+(A) - i_+(-A)$.

In order to compute $i_+(A)$, we show how to determine $i_+(p)$, the number of roots with positive real part of an integer polynomial $p(x) = x^n + c_1x^{n-1} + c_2x^{n-2} + \dots + c_n$, counting multiplicities. A known method to determine the number of roots in the right half-plane of a given real polynomial $p(x)$ is provided by Routh and Hurwitz (see e.g. [Gan77], Volume 2, Chapter XV).

Define $c_0 = 1$. The *Routh-Hurwitz matrix* of p is defined by the $n \times n$ matrix $\Omega(p)$,

$$\Omega(p) = \begin{pmatrix} c_1 & c_3 & c_5 & c_7 & \dots & 0 \\ c_0 & c_2 & c_4 & c_6 & \dots & 0 \\ 0 & c_1 & c_3 & c_5 & \dots & 0 \\ 0 & c_0 & c_2 & c_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & c_n \end{pmatrix}.$$

That is, the diagonal elements of $\Omega(p)$ are $\omega_{i,i} = c_i$. In the i -th column, the elements above the diagonal are $\omega_{i-1,i} = c_{i+1}$, $\omega_{i-2,i} = c_{i+2}$, \dots until we reach either the first row

$\omega_{1,i}$ or c_n . In the latter case, the remaining entries are filled with zeros. The elements below $\omega_{i,i}$ are $\omega_{i+1,i} = c_{i-1}$, $\omega_{i+2,i} = c_{i-2}$, \dots , c_1 , c_0 , 0 , 0 , \dots down to the last row $\omega_{n,i}$.

The successive leading principal minors D_i of $\Omega(p)$ are called the *Routh-Hurwitz determinants*. They are $D_1 = \det(c_1)$, $D_2 = \det\begin{pmatrix} c_1 & c_3 \\ c_0 & c_2 \end{pmatrix}$, \dots , $D_n = \det(\Omega(p))$.

Theorem 3.1 (Routh-Hurwitz) *If $D_n \neq 0$, then the number of roots of the polynomial $p(x)$ in the right half-plane is determined by the formula*

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

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 l successive zero Routh-Hurwitz determinants (l is always odd!)

$$D_s \neq 0, D_{s+1} = \dots = D_{s+l} = 0, D_{s+l+1} \neq 0$$

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

A proof of this theorem can be found in [Gan77], Volume 2, Chapter XV, Section 6.

Note the assumption $D_n \neq 0$ in the theorem. That is, we can apply the theorem directly only in the case that the Routh-Hurwitz matrix $\Omega(p)$ is regular. We analyze this *regular case* in Section 3.2 and then turn to the *singular case* in Section 3.3.

3.2 The Regular Case

Assume that $D_n \neq 0$. We apply the Routh-Hurwitz Theorem 3.1 to determine $i_+(A)$. It is clear that all elements of the Routh-Hurwitz matrix are computable in **GapL**. Therefore, by Theorem 2.2, all the Routh-Hurwitz determinants D_i are computable in **GapL**, too. It follows that one can decide in **PL** whether D_i is positive, negative, or zero. The k -th bit of i_+ can be therefore computed by a family of \mathbf{AC}^0 -circuits with **PL** oracles. Since $\mathbf{AC}^0(\mathbf{PL}) = \mathbf{PL}$, the sets **INERTIA** and **V-INERTIA** are in **PL**.

Theorem 3.2 [HT02a] *For matrices that have a regular Routh-Hurwitz matrix, **INERTIA** and **V-INERTIA** are in **PL**.*

3.3 The Singular Case

It is known from linear algebra that $D_n = 0$ if and only if $p(x)$ has a pair of opposite roots. Let us split $p(x)$ into even and odd terms: $p(x) = p_1(x) + p_2(x)$ where

$$\begin{aligned} p_1(x) &= x^n + c_{n-2}x^{n-2} + c_{n-4}x^{n-4} + \dots, \\ p_2(x) &= c_{n-1}x^{n-1} + c_{n-3}x^{n-3} + \dots. \end{aligned}$$

Define $g(x) = \gcd(p_1(x), p_2(x))$ and consider the decomposition

$$p(x) = g(x)p_0(x). \quad (1)$$

It follows that the opposite roots of $p(x)$ are precisely the roots of $g(x)$, and furthermore, $p_0(x)$ has no pair of opposite roots. Therefore we can determine $i_+(p_0)$ by Theorem 3.2.

Since $i_+(p) = i_+(g) + i_+(p_0)$, it now suffices to compute $i_+(g)$. However, the Routh-Hurwitz method doesn't apply to $g(x)$ because $g(x)$ has purely pairs of opposite roots (and $g(x)$ is therefore of even degree). Nevertheless we have

$$i_+(g) = \frac{1}{2}(\deg(g) - i_0(g)). \quad (2)$$

Consider Equation (2). The degree $\deg(g)$ can be determined by polynomial gcd computation (see page 3). Therefore, it remains to compute $i_0(g)$.

Observe that $i_0(g)$ can be easily determined when all the roots of g are real because in this case $i_0(g)$ is exactly the multiplicity of $x = 0$ as a root of $g(x)$. Recall that if $p(x)$ is the characteristic polynomial of a symmetric matrix, then all the roots of $p(x)$ are real. According to this observation we have the following theorem which is useful later on.

Theorem 3.3 [HT02a] *For symmetric matrices INERTIA and V-INERTIA are in PL.*

Coming back to the general case, by decomposition (1) we have

$$i_0(g) = i_0(p) - i_0(p_0).$$

Note that we can easily determine $i_0(p_0)$:

$$i_0(p_0) = \begin{cases} 1 & \text{if } p_0(0) = 0, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Therefore it suffices to compute $i_0(p)$: this gives us $i_0(g)$, from which we get $i_+(g)$ by Equation (2).

Let us summarize: in order to compute $i_+(p)$ it suffices to compute $i_0(p)$, the number of purely imaginary roots of $p(x)$.

We will explain below a theorem from linear algebra that shows how to determine the number of *distinct real roots* of a polynomial $q(x)$. In order to apply this theorem to

determine $i_0(p)$, we first have to turn $p(x)$ by 90° . This is done as follows.

It is known from linear algebra that for matrices A_1 and A_2 of order n and m , respectively, the eigenvalues of the *Kronecker product* $A_1 \otimes A_2$ are $\lambda_j(A_1)\lambda_k(A_2)$, for all j, k , where $\lambda_j(A_1)$ and $\lambda_k(A_2)$ are the eigenvalues of A_1 and A_2 . For our purpose, observe that the eigenvalues of the skew-symmetric matrix $E = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ are the imaginary

numbers $+i$ and $-i$. Define $B = E \otimes A = \begin{pmatrix} 0 & -A \\ A & 0 \end{pmatrix}$, where A is the given matrix having the characteristic polynomial $p(x)$. Then the eigenvalues of B are $i\lambda_k(A)$ and $-i\lambda_k(A)$ where $\lambda_k(A)$ runs through all eigenvalues of A . It follows that the number of real eigenvalues of B is exactly equal to $2i_0(p)$. Let $q(x) = \chi_B(x)$, the characteristic polynomial of B . Then we have

$$i_0(p) = \frac{1}{2} \text{ the number of real roots of } q$$

We conclude that in order to compute $i_0(p)$, it suffices to compute the number of real roots of $q(x)$.

The companion matrix of polynomial $q(x) = x^n + c_1x^{n-1} + c_2x^{n-2} + \dots + c_n$ is defined by

$$Q = \begin{pmatrix} 0 & 0 & \dots & 0 & -c_n \\ 1 & 0 & \dots & 0 & -c_{n-1} \\ 0 & 1 & \dots & 0 & -c_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -c_1 \end{pmatrix}.$$

The *Hankel matrix* $H = (h_{i,j})$ associated to $q(x)$ is defined by

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

where $\text{trace}(Q^{i+j-2})$ is the sum of all diagonal elements of Q^{i+j-2} . Note that H is symmetric. By $\text{sig}(H)$ we denote the *signature* of H , i.e. $\text{sig}(H) = i_+(H) - i_-(H)$. The following theorem can be found in [Gan77], Volume 2, Chapter XV:

Theorem 3.4 *Let H be the Hankel matrix associated with polynomial $q(x)$. Then*

1. $\text{sig}(H)$ is the number of distinct real roots of $q(x)$,
2. $\text{rank}(H)$ is the number of distinct roots of $q(x)$.

Obviously, for the given polynomial q , the elements of its Hankel matrix H are computable in **GapL**. By the fact after Theorem 2.2 this remains true when the coefficients of $q(x)$ are itself computable in **GapL**. As a consequence of Theorem 3.3 and 3.4 we have the following corollary:

Corollary 3.5 *The number of distinct real roots of a polynomial with coefficients computable in **GapL** can be determined in **PL**.*

According to Theorem 3.4 and Corollary 3.5 the idea for computing the number of real roots of $q(x)$ is as follows. $q(x)$ will be decomposed into factors by

$$q(x) = q_1(x)q_2(x) \cdots q_t(x)$$

such that each polynomial q_j has only roots of multiplicity 1. Polynomials with this property are called *square free*. That is, for each of these polynomials the number of its real roots is the same as the number of *distinct* real roots. Hence these numbers can be determined by Theorem 3.4. Finally, the sum of these numbers yields the number of real roots of $q(x)$.

It remains to find a suitable decomposition for $q(x)$. Let α be a root of $q(x)$, with multiplicity m , i.e.

$$q(x) = (x - \alpha)^m h(x) \text{ and } h(\alpha) \neq 0.$$

and $h(\alpha) \neq 0$. Consider the 1-st derivative $q^{(1)}$ of $q(x)$:

$$q^{(1)}(x) = m(x - \alpha)^{m-1} h(x) + (x - \alpha)^m h^{(1)}(x).$$

Since α is a root of $q^{(1)}(x)$ with multiplicity $m - 1$, it is also a root of $\gcd(q(x), q^{(1)}(x))$ with multiplicity $m - 1$. It follows that the polynomial

$$q_1(x) = \frac{q(x)}{\gcd(q(x), q^{(1)}(x))}$$

is square free. Similarly, for the i -th derivative $q^{(i)}(x)$, we can show that all the polynomials

$$q_i(x) = \frac{\gcd(q(x), q^{(i-1)}(x))}{\gcd(q(x), q^{(i)}(x))}$$

are square free and they yield the desired decomposition

$$q(x) = q_1(x)q_2(x) \cdots q_n(x).$$

We summarize the algorithm for computing $i_+(A)$ on input A :

1. Compute the polynomials

$$\begin{aligned} p(x) &= \chi_A(x), \\ p_0(x) &= \frac{p(x)}{\gcd(p_1(x), p_2(x))}, \\ &\quad \text{according to decomposition (1),} \\ q(x) &= \chi_{E \otimes A}(x), \\ q^{(i)}(x) &= \text{the } i\text{-th derivative of } q(x), \\ &\quad \text{for } i = 1, \dots, n, \\ q_i(x) &= \frac{\gcd(q(x), q^{(i-1)}(x))}{\gcd(q(x), q^{(i)}(x))}, \\ &\quad \text{for } i = 1, \dots, n. \end{aligned}$$

2. Compute the Hankel matrix H_i of $q_i(x)$, for $i = 1, \dots, n$.

3. Compute the values

$$i_+(H_i) \text{ and } i_-(H_i), \text{ for } i = 1, \dots, n, \\ \text{by Theorem 3.3,}$$

$$i_0(A) = \frac{1}{2} \sum_{i=1}^n \text{sig}(H_i),$$

$$i_+(p_0), \text{ by Theorem 3.2,}$$

$$i_0(g), \text{ by Equation (3),}$$

$$i_0(g) = i_0(A) - i_0(p_0),$$

$$i_+(g) = \frac{1}{2}(\deg(g) - i_0(g)).$$

4. Output $i_+(A) = i_+(p_0) + i_+(g)$.

We show the main theorem.

Theorem 3.6 INERTIA and v-INERTIA are in **PL**.

Proof. It is sufficient to show that $i_+(A)$ can be verified in **PL**.

By considering the computation of $i_+(A)$, described above, we observe that the polynomials p , q and $q^{(i)}$, for $i = 1, \dots, n$ are integer polynomials with coefficients computable in **GapL**.

As explained in the preliminary section, the degree of a polynomial gcd can be verified in **PL**. Therefore, the verifications of $\deg(p_0)$ and $\deg(q_i)$ are in **PL**. Furthermore, the coefficients of polynomials $q_i(x)$, and hence the elements of matrices H_i , have a rational form a/b where a and b are computable in **GapL**. Based on these observations, by Theorem 3.5, and by Corollary 3.5 we can show that the set $\{(A, i, k, l, s) \mid k = \deg(\gcd(q, q^{(i)})), l = \deg(\gcd(q, q^{(i-1)})), s = \text{sig}(H_i)\}$ is in **PL**. That means that the signature of each H_i can be verified in **PL**. The remaining values in Step 3 can be verified in **PL** as well. Therefore the algorithm to verify $i_+(A)$ can be implemented by a family of **AC**⁰-circuits with **PL**-oracles. Recall that **PL** is closed under **AC**⁰-reductions. \square

It has been shown in [HT02a] that INERTIA and v-INERTIA are hard for **PL** under logspace many-one reductions. Therefore we obtain the following corollary.

Corollary 3.7

INERTIA and v-INERTIA are complete for **PL**.

4 Closure Properties of GapL

In this section we show necessary and sufficient conditions that certain algebraic functions like the rank or the inertia of an integer matrix can be computed in **GapL**.

4.1 Matrix Rank

Assume that the rank of a matrix could be computed in \mathbf{GapL} . Then the verification of the rank, $\mathbf{V-RANK}$, would be in $\mathbf{C=L}$. On the other hand $\mathbf{V-RANK}$ is complete for $\mathbf{C=L} \wedge \mathbf{coC=L}$. Hence this would imply $\mathbf{C=L} = \mathbf{coC=L}$. The following theorem strengthens this collapse considerably.

Theorem 4.1 $\mathbf{C=L} = \mathbf{SPL} \iff \mathbf{rank} \in \mathbf{GapL}$.

Proof. Assume that $\mathbf{C=L} = \mathbf{SPL}$. Then $\mathbf{V-RANK} \in \mathbf{SPL}$. Hence, there is a function $g \in \mathbf{GapL}$ such that for a given matrix A of order n and a number r we have

$$\begin{aligned} \mathbf{rank}(A) = r &\implies g(A, r) = 1, \\ \mathbf{rank}(A) \neq r &\implies g(A, r) = 0. \end{aligned}$$

It follows that

$$\mathbf{rank}(A) = \sum_{r=1}^n r g(A, r),$$

and therefore $\mathbf{rank} \in \mathbf{GapL}$.

Conversely, suppose $\mathbf{rank} \in \mathbf{GapL}$. Then $\mathbf{C=L} = \mathbf{coC=L}$ as explained above. To show that $\mathbf{C=L} = \mathbf{SPL}$, we show that $\mathbf{V-POWELEM}$, a complete problem for $\mathbf{C=L}$, is in \mathbf{SPL} . Recall that there is a reduction by [ABO99] from $\mathbf{V-POWELEM}$ to $\mathbf{V-RANK}$ in the following way

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

where matrix B of order N can be easily computed from A . Define a \mathbf{GapL} -function g as

$$g(B) = N - \mathbf{rank}(B).$$

Then we have

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

Hence g is the characteristic function for deciding whether $(A, m, 1, n, 0) \in \mathbf{V-POWELEM}$. This shows that $\mathbf{V-POWELEM} \in \mathbf{SPL}$. \square

Next we weaken the assumption for the rank-function: instead of one \mathbf{GapL} -function that computes the rank directly, suppose there are two \mathbf{GapL} -functions g and h such that the rank can be written as the quotient of g and h , i.e., $\mathbf{rank}(A) = g(A)/h(A)$. We show that this is a necessary and sufficient condition for $\mathbf{C=L}$ being closed under complement.

Theorem 4.2

$\mathbf{C=L} = \mathbf{coC=L} \iff \exists g, h \in \mathbf{GapL} \text{ rank} = g/h.$

Proof. Assume that $\mathbf{C=L} = \mathbf{coC=L}$. Then the problem of verifying the rank of a matrix, $\mathbf{V-RANK}$, is in $\mathbf{coC=L}$. That is, there is a function $f \in \mathbf{GapL}$ such that for any matrix A and any r ,

$$\mathbf{rank}(A) = r \iff f(A, r) \neq 0.$$

Define functions

$$\begin{aligned} g(A) &= \sum_{r=0}^n r f(A, r), \\ h(A) &= \sum_{r=0}^n f(A, r). \end{aligned}$$

Then we have $g, h \in \mathbf{GapL}$ and $\mathbf{rank} = g/h$ as claimed.

Conversely, let $g, h \in \mathbf{GapL}$ such that $\mathbf{rank} = g/h$. For a given matrix A and an integer $k \geq 0$, define

$$f(A, k) = g(A) - k h(A).$$

Then $f \in \mathbf{GapL}$ and we have

$$\mathbf{rank}(A) = r \iff f(A, r) = 0.$$

It follows that the rank of a matrix can be verified in $\mathbf{C=L}$. Hence $\mathbf{C=L} = \mathbf{coC=L}$. \square

The *minimal polynomial* of a matrix A is the smallest degree monic polynomial $\mu(x)$, that fulfills the characteristic equation of A , $\mu(A) = \mathbf{0}$. In [HT02b], it has been shown that the degree of the minimal polynomial is computationally equivalent to matrix rank. Therefore, we can formulate the above theorems also in terms of the degree of the minimal polynomial.

There is an interesting alternative way of representing the rank of a matrix. Consider an $n \times n$ symmetric matrix A with the characteristic polynomial

$$\chi_A(x) = x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0.$$

It is well known from linear algebra that

$$\mathbf{rank}(A) = k \iff c_{n-k} \neq 0 \text{ and}$$

$$c_{n-k-1} = c_{n-k-2} = \dots = c_0 = 0.$$

Furthermore, all coefficients c_i are computable in \mathbf{GapL} [Ber84].

Define a vector $\mathbf{w} = (w_n, w_{n-1}, \dots, w_1, w_0)^T$, where $w_j = \sum_{i=0}^j c_i^2$, for $j = 0, 1, \dots, n$. Hence every element of \mathbf{w} is computable in \mathbf{GapL} . Furthermore we have: $\mathbf{rank}(A) = k$ if and only if

- (i) \mathbf{w} has precisely $k + 1$ positive elements, $w_n, w_{n-1}, \dots, w_{n-k}$, and
- (ii) precisely $n - k$ zero elements, $w_{n-k-1} = w_{n-k-2} = \dots = w_0 = 0$.

Conversely, for a given nonnegative **GapL**-vector v , the number of its positive elements is exactly the rank of the diagonal matrix whose diagonal is v .

In summary, the problem of determining the rank of a matrix is (logspace) equivalent to the problem of determining the number of consecutive zeros at the right end in a **GapL**-vector.

4.2 Matrix Inertia

It is known that the rank can be reduced to the inertia by

$$\text{rank}(A) = \text{rank}(A^T A) = i_+(A^T A). \quad (4)$$

Recall that the functions i_+ , i_- , and i_0 of the inertia are computationally equivalent because $i_+(A) = i_-(-A)$ and $i_0(A) = n - i_+(A) - i_-(A)$. The following theorem characterizes the case that the upper bound for computing the inertia can be improved from **PL** to **GapL**.

Theorem 4.3 $\text{PL} = \text{SPL} \iff i_+ \in \text{GapL}$.

Proof. Assume that $\text{PL} = \text{SPL}$. By Theorem 3.6, the verification of i_+ is in **PL**, and hence in **SPL**. That is, there exists a function $g \in \text{GapL}$ such that

$$\begin{aligned} i_+(A) = j &\implies g(A, j) = 1, \\ i_+(A) \neq j &\implies g(A, j) = 0, \end{aligned}$$

for a matrix A and for all $0 \leq j \leq n$. It follows that

$$i_+(A) = \sum_{j=1}^n j g(A, j),$$

and therefore $i_+ \in \text{GapL}$.

Conversely, suppose $i_+ \in \text{GapL}$. Then the verification of i_+ is in **C=L**. Because i_+ is complete for **PL** by Corollary 3.7, we have $\text{PL} = \text{C=L}$. By Equation (4), rank is in **GapL** too. Thus $\text{C=L} = \text{SPL}$ by Theorem 4.1. Therefore $\text{PL} = \text{SPL}$. \square

Like for the rank in Section 4.1 we show the following theorem for a weaker condition that we can express i_+ as a quotient of two **GapL**-functions.

Theorem 4.4

$\text{PL} = \text{C=L} \iff \exists g, h \in \text{GapL} \ i_+ = g/h$.

Proof. Assume that $\text{PL} = \text{C=L}$. Then the problem of verifying i_+ is in **coC=L**. That is, there is a function $f \in \text{GapL}$ such that for any matrix A and any j , we have:

$$i_+(A) = j \iff f(A, j) \neq 0.$$

Define functions

$$\begin{aligned} g(A) &= \sum_{j=0}^n j f(A, j), \\ h(A) &= \sum_{j=0}^n f(A, j). \end{aligned}$$

Then we have $g, h \in \text{GapL}$ and $i_+(A) = g/h$ as claimed.

Conversely, let $g, h \in \text{GapL}$ such that $i_+ = g/h$. For a given matrix A and an integer $k \geq 0$, define

$$f(A, k) = g(A) - k h(A).$$

Then $f \in \text{GapL}$ and we have

$$i_+(A) = j \iff f(A, j) = 0.$$

That is, we can verify i_+ in **C=L**. Therefore, $\text{PL} = \text{C=L}$. \square

4.3 Absolute value

For any function f mapping to integers, by $\text{abs}(f)$ we denote the function of absolute values of f . That is

$$\text{abs}(f)(x) = \begin{cases} f(x) & \text{if } f(x) \geq 0, \\ -f(x) & \text{otherwise.} \end{cases}$$

Theorem 4.5

$\text{PL} = \text{SPL} \iff \forall f \in \text{GapL} \ \text{abs}(f) \in \text{GapL}$.

Proof. Suppose $\text{PL} = \text{SPL}$ and let $f \in \text{GapL}$. Define the set $S = \{x \mid f(x) > 0\}$. By definition $S \in \text{PL}$ and therefore $S \in \text{SPL}$, by assumption. That is, there is $g \in \text{GapL}$ such that for all x :

$$g(x) = \begin{cases} 1, & \text{if } x \in S, \\ 0, & \text{otherwise.} \end{cases}$$

Then we can write $\text{abs}(f) = (2g - 1)f$, and therefore $\text{abs}(f) \in \text{GapL}$.

Conversely, let $S \in \text{PL}$. That is, for some function $f \in \text{GapL}$, we can write $S = \{x \mid f(x) > 0\}$. We define the following functions

$$\begin{aligned} g &= \text{abs}(f) - \text{abs}(f - 1), \\ h &= \binom{g + 1}{2}. \end{aligned}$$

We have $g \in \text{GapL}$, by assumption. It follows that $h \in \text{GapL}$ by the closure properties of **GapL**. Now observe that

$$h(x) = \begin{cases} 1 & \text{if } f(x) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

This shows that $S \in \text{SPL}$, and therefore $\text{PL} = \text{SPL}$. \square

Open Problems

In the polynomial time setting it is known that $\mathbf{PP} \subseteq \mathbf{SPP}^{\mathbf{C=P}}$. The proof is quite easy:

Let $A = \{x \mid f(x) > 0\} \in \mathbf{PP}$, for some $f \in \mathbf{GapP}$. A nondeterministic machine M on input x guesses $k > 0$ and asks its $\mathbf{C=P}$ -oracle whether $f(x) = k$. If the answer is “yes”, then M accepts. If the answer is “no”, then M branches once and accepts on one branch and rejects on the other branch. This shows that $A \in \mathbf{SPP}^{\mathbf{C=P}}$.

Note that this proof doesn’t work in the logspace setting: in the Ruzzo-Simon-Tompa model of space-bounded oracle machines, the machine has to be deterministic while writing a query. Hence we ask

- Is $\mathbf{PL} \subseteq \mathbf{SPL}^{\mathbf{C=L}}$?

Because $\mathbf{SPP}^{\mathbf{SPP}} = \mathbf{SPP}$, the above inclusion implies that $\mathbf{C=P} = \mathbf{SPP} \implies \mathbf{PP} = \mathbf{SPP}$. In the logspace setting, we also have $\mathbf{SPL}^{\mathbf{SPL}} = \mathbf{SPL}$ [ARZ99], but the above conclusion is open.

- Does $\mathbf{C=L} = \mathbf{SPL} \implies \mathbf{PL} = \mathbf{SPL}$?

In particular, this question is equivalent to finding a reduction from the inertia to the rank of a matrix, and the latter functions don’t look very different (in complexity).

Acknowledgments

We wish to thank Eric Allender and the referees for very helpful comments on the paper.

References

- [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.
- [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 logspace counting class. *Theoretical Computer Science*, 107:3–30, 1993.
- [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.
- [Ber84] S. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150, 1984.
- [BF00] R. Beigel and B. Fu. Circuits over PP and PL. *Journal of Computer and System Sciences*, 60:422–441, 2000.
- [BvzGH82] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52:241–256, 1982.
- [Dam91] C. Damm. $\mathbf{DET} = \mathbf{L}^{\#\mathbf{L}}$. Technical Report Informatik-Preprint 8, Fachbereich Informatik der Humboldt-Universität zu Berlin, 1991.
- [FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48:116–148, 1994.
- [Gan77] F. Gantmacher. *The Theory of Matrices*, volume 1 and 2. AMS Chelsea Publishing, 1977.
- [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 Computing & Combinatorics Conference (COCOON)*, Lecture Notes in Computer Science 2387, pages 37–46. Springer-Verlag, 2002.
- [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.
- [Koz91] Dexter Kozen. *The Design and Analysis of Algorithms*. Springer-Verlag, 1991.
- [MV97] M. Mahajan and V Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago Journal of Theoretical Computer Science*, 1997(5), 1997.

- [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.
- [Ogi98] M. Ogihara. The PL hierarchy collapses. *SIAM Journal on Computing*, 27:1430–1437, 1998.
- [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.
- [Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 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.
- [Vin91] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *6th IEEE Conference on Structure in Complexity Theory*, pages 270–284, 1991.