Theory Comput Syst
DOI 10.1007/s00224-008-9136-8

On the Complexity of Matrix Rank and Rigidity

Meena Mahajan · Jayalal M.N. Sarma

Abstract We revisit a well studied linear algebraic problem, computing the rank and determinant of matrices, in order to obtain completeness results for small complexity classes. In particular, we prove that computing the rank of a class of diagonally dominant matrices is complete for L. We show that computing the permanent and determinant of tridiagonal matrices over $\mathbb{Z}$ is in GapNC^1 and is hard for NC^1. We also initiate the study of computing the rigidity of a matrix: the number of entries that needs to be changed in order to bring the rank of a matrix below a given value. We show that some restricted versions of the problem characterize small complexity classes. We also look at a variant of rigidity where there is a bound on the amount of change allowed. Using ideas from the linear interval equations literature, we show that this problem is NP-hard over $\mathbb{Q}$ and that a certain restricted version is NP-complete. Restricting the problem further, we obtain variations which can be computed in PL and are hard for $\mathsf{C}_=\mathsf{L}$.

Keywords Complexity classes · Matrix rank · Determinant · Matrix rigidity

1 Introduction

A series of seminal papers by a variety of people including Valiant, Mulmuley, Toda, Vinay, Grigoriev, Cook, and McKenzie, set the stage for studying the complexity of computing matrix properties (in particular, determinant and rank) in terms of logspace computation and poly-size polylog-depth circuits. This area has been active for many years, and an NC upper bound is known for many related problems in linear algebra;

M. Mahajan (✉) · J.M.N. Sarma
The Institute of Mathematical Sciences, 600 113 Chennai, India
e-mail: meena@imsc.res.in

J.M.N. Sarma
e-mail: jayalal@imsc.res.in

Table 1 RANK BOUND, SINGULAR, and DETERMINANT for special matrices

Matrix type (over $\mathbb{Q}$)	RANK BOUND	SINGULAR	DETERMINANT
general (even 0-1)	$\mathsf{C_{=}L}$-complete [3]	$\mathsf{C_{=}L}$-complete [3]	GapL-complete [13, 34] [31, 33]
symmetric non-neg.	$\mathsf{C_{=}L}$-complete [3]	$\mathsf{C_{=}L}$-complete [3]	GapL-hard under $\leq_T^{\log}$ reductions [18]
symmetric non-neg. diag. dominant (d.d.)	L-complete (Theorem 5)	L-complete (Theorem 5)	?
symmetric d.d.	L-hard even when det $\in \{0, 1\}$ (Theorem 10)		?
diagonal	TC^0-complete (Prop 1)	AC^0 (Prop 1)	TC^0-complete (Prop 1)
tridiagonal	?	$\mathsf{C_{=}NC}^1$ (Theorem 11)	GapNC^1 (Theorem 11)
tridiagonal non-neg.	non-negative perm equivalent to planar #BWBP (Theorem 11)		

see for instance [5]. Some of the major results in this area are that computing the determinant of integer matrices is GapL-complete and that testing singularity of integer matrices is $\mathsf{C_{=}L}$-complete. In particular, the complexity of computing the rank of a given matrix over $\mathbb{Q}$ has been well studied. For general matrices, checking if the rank is at most r is $\mathsf{C_{=}L}$-complete [3].

Complete problems for complexity classes are always promising, since they provide a set of possible techniques that are associated with the problem to attack various questions regarding the complexity class. Such results can be expected to flourish when the complete problem has well-developed tools associated with it. With this motivation, we look at special cases of the matrix rank problem and try to characterize small complexity classes. We consider restrictions which are combinations of non-negativity, 0-1 entries, symmetry, diagonal dominance, and tridiagonal support, and we consider the complexities of three problems: computing the rank, computing the determinant and testing singularity. These, though intimately related, can have differing complexities, as Table 1 shows.

However, the corresponding optimization search problems can be considerably harder. Consider the following existential search question: Given a matrix M over a field $\mathbb{K}$, a target rank r and a bound k, decide whether the rank of M can be brought down to below r by changing at most k entries of M. Intuitively, one would expect such a question to be in $\exists \cdot \mathsf{NC}$: guess k locations where M is to be changed, guess the new entries to be inserted there, and compute the rank in NC [22]. However, this intuition, while correct for finite fields (this case was recently shown to be NP-complete

[14]), does not directly translate to a proof for $\mathbb{Q}$ and $\mathbb{Z}$[1] since the required new entries may not have representations polynomially-bounded in the input size. Using results about matrix completion problems, we can obtain upper bounds of PSPACE over reals and complex numbers, and decidability over p-adic numbers, [8]. However, in the case of arbitrary infinite fields, the best upper bound we can see in the general case is recursive enumerability, and in particular, this is the situation over $\mathbb{Q}$. We also do not know any lower bounds for this question over $\mathbb{Q}$. In this paper, we explore the computational complexity of several variants of this problem.

The above question is a computational version of rigidity of a matrix, which is the smallest value of k for which the answer to the above question is yes. The notion of rigidity was introduced by Valiant [32] and independently proposed by Grigoriev [17]. The main motivation for studying rigidity is that good lower bounds on rigidity give important complexity-theoretic results in other computational models, like linear algebraic circuits and communication complexity. Though the question we address is in fact a computational version of rigidity, it has no direct implications for these lower bounds. However, it provides natural complete problems based on linear algebra for important complexity classes.

An important aspect of computing rigidity is its possible connection to the theory of natural proofs developed by Razborov and Rudich [30]. Valiant's reduction [32] identifies "high rigidity" as a combinatorial property of functions, based on which he proves linear-size lower bounds for log-depth circuits. However, the model of arithmetic circuits has not been studied in sufficient detail such that in the setting of natural proofs this can directly provide some evidence about the power of the proof technique. Nevertheless, this could be thought of as motivation for the computational question of rigidity.

Our question bears close resemblance to the body of problems considered under *matrix completion*, see for instance [8, 19]. Given a matrix with indeterminates in some locations, can we instantiate them in such a way that some desired property (e.g. non-singularity) is achieved? In Sect. 4, we discuss how results from matrix completion can yield upper bounds for our question.

In this paper, we restrict our attention to $\mathbb{Z}$ and $\mathbb{Q}$ (some extensions to finite fields are discussed at the end). Since even an upper bound of NP is not obvious, we restrict the choice available in changing matrix entries. We consider two variants: (1). In the input, a finite subset $S \subseteq \mathbb{K}$ is given. M has entries over S, and the changed entries must also be from S; rank computation continues to be over $\mathbb{K}$. (For instance, we may consider Boolean matrices, so $S = \{0, 1\}$, while rank computation is over $\mathbb{K}$.) It is easy to see that this variant is indeed in NP. (2). In the input, a bound θ is given. We require that the changes be bounded by θ; we may apply the bound to each change, or to the total change, or to the total change per row/column. (See for instance [20].) This version has close connections with another well-studied area called linear interval equations which arises naturally in the context of control systems theory (see [28]).

[1]Technically, rank over $\mathbb{Z}$ is not defined, since $\mathbb{Z}$ is not a field. In Sect. 2, we define a natural notion of rank over rings. Under this, since $\mathbb{Z}$ is an integral domain, the rank is the same as over the corresponding division ring $\mathbb{Q}$.

Table 2 Our bounds on RIGID when $k \in O(1)$ or $r = n$

$\mathbb{K}$, $S \subset \mathbb{K}$ (if $*$, then $S = \mathbb{K}$)	Restriction	Bound
$\mathbb{Z}$ or $\mathbb{Q}$, $\{0,1\}$		in NP
$\mathbb{Z}$ or $\mathbb{Q}$, $\{0,1\}$	$k \in O(1)$	$\mathsf{C_{=}L}$-complete (Theorem 13)
$\mathbb{Z}$ or $\mathbb{Q}$, $*$	$k \in O(1)$	$\mathsf{C_{=}L}$-hard [3]
$\mathbb{Q}$, $*$	$r = n$	$\mathsf{C_{=}L}$-complete [3]
		witness-search in $\mathsf{L}^{\mathsf{GapL}}$ (Theorem 15)
$\mathbb{Z}$, $*$	$r = n$ and $k = 1$	in $\mathsf{L}^{\mathsf{GapL}}$ (Theorem 16)
$\mathbb{Z}$ or $\mathbb{Q}$, $*$	bounded rigidity, $r = n$	NP-complete (Theorem 18)
$\mathbb{Z}$ or $\mathbb{Q}$, $*$	bounded rigidity, $r = n, k = 1$	In PL, and $\mathsf{C_{=}L}$-hard (Theorem 22)

We obtain tighter lower and upper bounds for some of these questions. We show completeness for $\mathsf{C_{=}L}$ when $k \in O(1)$ in the first variant, for NP when the target rank r equals n in the second variant, and for $\mathsf{C_{=}L}$ when $r = n$ in the general case. Table 2 summarizes the results.

2 Preliminaries

Over any field $\mathbb{F}$, the rank of a matrix $M \in \mathbb{F}^{n \times n}$ (we consider only square matrices in this paper) has the following equivalent definitions: (1) The maximum number of linearly independent rows or columns in M. (2) The maximum size of a non-singular square submatrix of M. (3) The minimum r such that $M = AB$ for some $A \in \mathbb{F}^{n \times r}$ and $B \in \mathbb{F}^{r \times n}$. (4) The minimum r such that M is the sum of r rank-1 matrices, where a rank-1 matrix is one for which there exists a vector v (not necessarily in the matrix) such that every row in the matrix can be expressed as a multiple of v. These definitions need not be equivalent when the underlying algebraic structure is not a field. Hence, the notion of rank is not well-defined over arbitrary rings. However, if the ring under consideration is an infinite integral domain (like $\mathbb{Z}$) (notice that a finite integral domain has to be a field), then the above definitions are indeed equivalent, and can be taken as a definition of rank. In fact, the rank in that case can be easily seen to be same as the rank over the corresponding quotient field; thus rank over $\mathbb{Z}$ as defined above is the same as rank over $\mathbb{Q}$.

Now we introduce the basic notions in complexity theory that we need. $\mathsf{L}$and NL denote languages accepted by deterministic and nondeterministic logspace classes respectively, and $\mathsf{FL}$is the class of logspace-computable functions. #L is the class of functions that count the number of accepting paths of an NL machine, and GapL is its closure under subtraction. Computing the determinant over $\mathbb{Z}$ is complete for GapL. In contrast, computing the permanent is complete for $\mathsf{\#P}$, the class of functions counting accepting paths of an NP machine. NC^1 is the class of languages with polynomial-size logarithmic-depth Boolean circuits. $\mathsf{\#NC}^1$ is the class of functions computed by arithmetic circuits (gates compute $+$ and $\times$) with the same size and depth bounds as NC^1, and GapNC^1 is its closure under subtraction. AC^0 (TC^0) is the class of languages with polynomial-size constant-depth unbounded fanning Boolean circuits, where gates compute AND, OR, NOT (and MAJORITY). For more details, see [35].

A language L is in the exact counting logspace class $\mathsf{C_{=}L}$ (or probabilistic logspace PL) if and only if it consists of exactly those strings where a certain GapL function is zero (positive, respectively). The languages

$$\text{SINGULAR}(\mathbb{K}) = \{M \mid \text{Over } \mathbb{K},\ M \text{ is not full rank}\},$$

$$\text{RANK BOUND}(\mathbb{K}) = \{(M, r) \mid \text{Over } \mathbb{K},\ \text{rank}(M) < r\}$$

for $\mathbb{K} = \mathbb{Z}$ or $\mathbb{Q}$ are complete for $\mathsf{C_{=}L}$ [3]. Note that for any type of matrices, and any complexity class $\mathcal{C}$, $\mathcal{C}$-hardness of SINGULAR implies $\mathcal{C}$-hardness of RANK BOUND. However the converse is not true:

Proposition 1 (folklore) *Restricted to diagonal matrices,* SINGULAR($\mathbb{Z}$) *is in* AC^0 *while* RANK BOUND($\mathbb{Z}$) *and* DETERMINANT *are* TC^0*-complete.*

The rigidity function, and its decision version, are as defined below.[2] (Here $\text{support}(N) = \#\{(i, j) \mid N(i, j) \neq 0\}$.)

$$R_M(r) \overset{def}{=} \min_N \{\text{support}(N) : \text{rank}(M + N) < r\},$$

$$\text{RIGID}_{\mathbb{K}} = \{(M, r, k) \mid R_M(r) \leq k\}.$$

Lemma 2 (Valiant, folklore) *Over any field* $\mathbb{F}$, $\text{rank}(M) - r \leq R_M(r + 1) \leq (n - r)^2$.

The inequality on the left also follows from the following lemma which we will use later:

Lemma 3 (folklore) *Over any field* $\mathbb{F}$, *for any two matrices* M *and* N *of the same order,*

$$\text{support}(M - N) = 1 \Longrightarrow |\text{rank}(M) - \text{rank}(N)| \leq 1.$$

To see this, use the facts that rank is sub-additive and that $\text{rank}(-A) = \text{rank}(A)$. Hence for any two matrices A and B, $\text{rank}(A) - \text{rank}(B) \leq \text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$. Further, $\text{rank}(A) \leq \text{support}(A)$, yielding the claim.

3 Computing the Rank for Special Matrices

Computation of rank is intimately related to computation of the determinant. Mulmuley [22] showed that over arbitrary fields, rank can be computed in NC (with the field

[2]In much of the rigidity literature, $\text{rank}(M + N) \leq r$ is required. We use strict inequality to be consistent with the definition of RANK BOUND from [3].

operations as primitives). Over $\mathbb{Z}$ and $\mathbb{Q}$, RANK BOUND is $\mathsf{C_{=}L}$-complete [3], and we wish to characterize its subclasses by restricting the types of matrices. A natural approach is to use characterizations of matrix rank in terms of associated combinatorial objects, like graphs. However, no known parameter of the graph of a matrix characterizes the matrix rank in general.

The following is easy to see:

Proposition 4 *The languages* RANK BOUND($\mathbb{Z}$) *and* SINGULAR($\mathbb{Z}$) *remain* $\mathsf{C_{=}L}$-*hard even if the instances are restricted to be symmetric* 0-1 *matrices.*

Proof Let A' be the symmetric matrix $\left[\begin{smallmatrix} 0 & A \\ A_T & 0 \end{smallmatrix}\right]$. Since $\mathrm{rank}(A') = 2(\mathrm{rank}(A))$, RANK BOUND($\mathbb{Z}$) remains $\mathsf{C_{=}L}$-hard when restricted to symmetric matrices. Further, DETERMINANT remains GapL-hard even when the matrices are restricted to be 0-1 (see for instance [31]). Thus SINGULAR remains $\mathsf{C_{=}L}$-hard even when restricted to 0-1 matrices. Since M is in SINGULAR if and only if (M, n) is in RANK BOUND if and only if $(M', 2n)$ is in RANK BOUND, it follows that RANK BOUND($\mathbb{Z}$) remains $\mathsf{C_{=}L}$-hard for symmetric 0-1 matrices as well. □

DETERMINANT remains GapL-hard for 0-1 matrices, but it is not clear that symmetric instances are GapL-hard under many-one reductions. The above trick does not work for computing determinants, because $\det(A')$ will equal $\pm\det(A)^2$ and GapL is not known to be closed under taking square-roots. We do not know (any other way of showing) many-one hardness for symmetric DETERMINANT. Recently, Kulkarni [18] has observed that symmetric instances are GapL-hard under Turing reductions. The idea is to first use Chinese Remaindering: any determinant can be computed in L if its residues modulo polynomially many primes are available. Small primes (logarithmically many bits) suffice and can be obtained explicitly. Now to find the determinant modulo a small prime p, range over all $a \in \{0, 1, \dots, p-1\}$ and test if it equals a modulo p. But this can be recast, using the GapL-completeness proofs of the determinant, as asking if a related determinant is 0 modulo p. Finally, using the idea in the proof of Proposition 4, we can ask the oracle for the determinant of a related symmetric matrix and test (in L) if it is 0 modulo p.

We now consider an additional restriction. A matrix M is said to be *diagonally dominant* if for every i, $|m_{i,i}| \geq \sum_{j\neq i} |m_{i,j}|$. (If all the inequalities are strict, then M is said to be *strictly diagonally dominant*.) We show:

Theorem 5 SINGULAR($\mathbb{Z}$) *restricted to non-negative diagonally dominant symmetric matrices is* L-*complete. The hardness is via uniform* AC^0 *many-one reductions.*

Proof This result exploits a very nice combinatorial connection between such matrices and graphs. For a non-negative symmetric diagonally-dominant matrix M, its *support graph* $G_M = (V, E_M)$ has $V = \{v_1, \dots, v_n\}$, and $E_M = \{(v_i, v_j) \mid i \neq j\ m_{i,j} > 0\} \cup \{(v_i, v_i) \mid m_{i,i} > \sum_{i\neq j} m_{i,j}\}$. The following is shown in [12] for $\mathbb{R}$, and it can be verified that the same holds over $\mathbb{Q}$.

Lemma 6 [12] *Let M be a non-negative symmetric diagonally dominant matrix of order n over $\mathbb{Q}$ or $\mathbb{R}$. Then $\mathrm{rank}(M) = n - c$, where c is the number of bipartite components in the support graph G_M.*

Hardness: The reduction is from undirected forest accessibility UFA, which is L-complete and remains L-hard even when the graph has exactly 2 components [10]. Without loss of generality, we can assume that the input instances have a nice form, as stated in the following lemma.

Lemma 7 ([10]) *Given an undirected forest G, of bounded degree with exactly two components, and three special vertices s, t and q, with the guarantee that t and q are in different components, deciding which component s belongs to is L-hard.*

Proof The reduction is from the machine model for L, and is essentially reproduced from [10]. We rephrase the proof here to highlight the fact that the normal form we need is indeed achievable.

To begin with, modify the machine description such that whenever the computation is on an infinite loop, the machine clears off the work-tape and goes to an error state e. Thus there are only two possible final states for the machine, one is the error configurations e, and the other is the accepting configuration t.

The set of configurations of a Turing machine with a fixed input w forms the vertices of such a graph G, and the (unique) accepting configuration is accessible from the initial configuration if and only if the Turing machine accepts the input w. G can be made acyclic by associating a time stamp with the configurations, and insisting that an edge always joins a configuration at time i to a configuration at time $i+1$. If $p(n)$ is an upper bound on the computation time of the Turing machine with input w, then we let the node t in the graph be the accepting configuration with time stamp $p(n)$, and s will be the initial configuration with time stamp 0.

By definition, the number of possible (in/out)-neighbors of any node is bounded by a constant. In addition there are exactly two nodes of outdegree 0, and they correspond to the configurations e and t.

Viewing each edge in the resulting digraph as undirected yields an undirected forest such that s and t belong to the same tree if and only if a directed path existed from s to t in the original digraph. Note that the resulting undirected forest has precisely two components, and the three vertices satisfy the required properties of the reduction. □

We now construct G' as follows: Make two copies G_1 and G_2 of G. Add a new vertex u. Add edges (s_1, s_2), (t_1, u), (t_2, u). Add self-loops at q_1 and q_2.

G' has at most three components (copies of the components containing t join up via u). The component(s) containing copies of q are necessarily non-bipartite.

If there is an $s \rightsquigarrow t$ path ρ in G, then in G' the two copies of the path, along with the edges (s_1, s_2), (t_1, u), (u, t_2) create an odd cycle, so the new joined up component is also not bipartite. Hence G' has no bipartite components.

If there is no $s \rightsquigarrow t$ path in G, the component containing t_1 and t_2 will remain bipartite. Thus there is exactly one bipartite component now.

To complete the proof, we need to produce a matrix M such that G' is its support graph. We construct M as follows:

$$\text{For each } i \neq j \quad m_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E', \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{For each } i \quad m_{i,i} = \begin{cases} 1 + \sum_{j \neq i} m_{i,j} & \text{if } (i,i) \in E', \\ \sum_{j \neq i} m_{i,j} & \text{otherwise.} \end{cases}$$

From Lemma 6, M is singular if and only if there is no $s \rightsquigarrow t$ path in G.

It is clear that M can be constructed from G', and hence from G, by a uniform TC^0 circuit. Now we show that in fact it can be constructed in AC^0. First, observe that the forest that we start with (as the L-hard instance) has bounded degree. So we would like to rewrite the summation $\sum_{j \neq i} m_{i,j}$ as $\sum_{j \neq i; m_{i,j} \neq 0} m_{i,j}$. But how do we know *a priori* which entries are non-zero? For a node i, define L_i to be the list of nodes for which $m_{i,j}$ can possibly be non-zero. Since the logspace Turing machine alters only a small part of the configuration in one step, this list is of bounded length, with the bound l depending only on the machine's description and not on the input length. Let $\text{list}(i,t)$ denote the t^{th} element in a lexicographical enumeration of L_i; on input i, t, $\text{list}(i,t)$ can be determined in AC^0. Now the required summation is exactly $\sum_{j \in L_i} m_{i,j} = \sum_{t=1}^{l} m_{i,\text{list}(i,t)}$, and thus it can be computed by an AC^0 circuit.

Membership in L: Given a matrix M satisfying the stated conditions, it is straightforward to construct the support graph G_M. By [4, 25, 27], checking whether two vertices belong to the same component in an undirected graph, counting the number of components, and checking bipartiteness of a named component are all in L. Hence, by Lemma 6, $\text{rank}(M)$ can be computed in L. □

Corollary 8 *The language* RANK BOUND($\mathbb{Z}$), *restricted to symmetric non-negative diagonally dominant instances, is* L*-complete.*

However, the hardness of RANK BOUND($\mathbb{Z}$) is not derived just from the hardness of SINGULAR. An obvious way to obtain hardness at other values of rank (rather than $r = n$ in the case of SINGULAR) is to pad out the matrix with zero rows and/or columns. We present here a slightly different proof of Theorem 5, establishing hardness of deciding whether the rank is $n-1$ or $n-2$.

Proof The reduction is again from undirected forest accessibility UFA, using nice instances as guaranteed by Lemma 7.

Let G, s, t be an instance of UFA, where G has two trees. We construct a new graph $G' = (V', E')$ as follows: take two disjoint copies of G. Add a new vertex u and connect it to both copies of t. Connect the two copies of s. Also, add self-loops at both copies of t.

If there is an $s \rightsquigarrow t$ path ρ in G, then G' has three components: the copies of the component containing s and t join up, while the copies of the other component remain disconnected (and hence bipartite). The two copies of the path, along with the edges (s_1, s_2), (t_1, u), (u, t_2) create an odd cycle, so the new joined up component is not bipartite. Hence G' has exactly two bipartite components.

If there is no $s \rightsquigarrow t$ path in G, the component containing s_1 and s_2 will remain bipartite. The other component is not bipartite due to the self loops at $t_, t_2$. Thus there is exactly one bipartite component now.

To complete the proof, we need to produce a matrix M such that G' is its support graph. This can be done in AC^0 as described in the previous proof. □

Note that though rank for these matrices can be computed in L, we do not know how to compute the exact value of the determinant itself. (Note that by Theorem 5, this is hard for FL.) In a brief digression, we note the following: if a matrix is to have no trivial (all-zero) rows, and yet be diagonally dominant, then it cannot have any zeroes on the diagonal. One may ask if it is easier to compute determinants when there can be no zeros on the main diagonal. We do not know if computing determinants of such matrices is complete under many-one reductions, although the following lemma shows that it is complete under a restrictive type of Turing reduction.

Lemma 9 *For every* GapL *function f and every input x, $f(x)$ can be expressed as* $\det(M) - 1$, *where M has no zeroes on the diagonal. M can be obtained from x via projections (each output bit is dependent on at most one bit of x).*

Proof Consider Toda's proof [31] for showing that DETERMINANT is GapL-hard (see also [3, 24]). Given any GapL function f and input x, it constructs a directed graph G with self-loops at every vertex except a special vertex s. G also has the property that every non-trivial cycle (not a self-loop) in G passes through s. If A is the adjacency matrix of G, then the construction satisfies $f(x) = \det(A)$. Now consider the matrix B obtained by adding a self-loop at s. What additional terms does $\det(B)$ have that were absent in $\det(A)$? Such terms must correspond to cycle covers using the self-loop at s; i.e. cycle covers in $G \setminus \{s\}$. But $G \setminus \{s\}$ has no non-trivial cycles, so the only additional cycle cover is all self-loops, contributing $a + 1$. Thus $\det(B) = 1 + \det(A)$, and B is the required matrix. □

We also show via a somewhat different reduction that the L-hardness of SINGULAR in Theorem 5 holds even if we allow negative values, but disallow matrices with determinant other than 0 or 1.

Theorem 10 SINGULAR($\mathbb{Z}$) *for symmetric diagonally dominant matrices is* L*-hard, even when restricted to instances with* 0*-or-*1 *determinant.*

Proof As in the proof of Theorem 5, we begin with an instance (G, s, t) of UFA where G has exactly two components. Add edge (s, t) to obtain graph H. By the matrix-tree theorem (see for example Theorem II-12 in [9]), if A is the Laplacian matrix of H (defined below), and B is obtained by deleting the topmost row and leftmost column of A, then $\det(B)$ equals the number of spanning trees of H.

The Laplacian matrix A is defined as follows:

$$\begin{aligned} a_{i,i} &= \text{the degree of vertex } i \text{ in } H, \\ a_{i,j} &= -1 \text{ if } i \neq j \text{ and } (i, j) \text{ is an edge in } H, \\ a_{i,j} &= \ \text{ if } i \neq j \text{ and } (i, j) \text{ is not an edge in } H. \end{aligned}$$

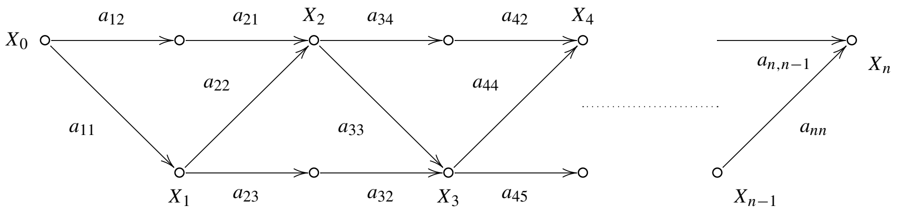

Fig. 1 Width-2 branching program for tridiagonal permanent

Clearly, A is diagonally dominant (in fact, for each i, the constraint is an equality); also, since H is an undirected graph, A is symmetric.

Now the number of spanning trees in H is 1 if $s \not\rightsquigarrow_G t$ (H itself is a tree) and is 0 if $s \rightsquigarrow_G t$ (H still has two components, since the edge in $H \setminus G$ joins vertices in the same component of G). □

The next restriction we consider is tridiagonal matrices: $m_{i,j} \neq 0 \Longrightarrow |i-j| \leq 1$. We show that DETERMINANT and PERMANENT are in GapNC^1, by using bounded-width branching programs BWBP. In the Boolean context, BWBP equals NC^1. However, in the arithmetic context, they are not that well understood. It is still open [5, 11] whether the containment $\#\mathsf{BWBP} \subseteq \#\mathsf{NC}^1$ is in fact an equality (though it is known that $\mathsf{GapBWBP} = \mathsf{GapNC}^1$). Layered planar BWBP are the G-graphs referred to in [1]. Counting paths in G-graphs may well be simpler than GapNC^1 due to planarity. However [1] (see also [5]) shows that even over width-2 G-graphs, it is hard for NC^1 (under AC^0 [5] reductions). We show that the permanent and determinant of tridiagonal matrices are essentially equivalent to counting in width-2 G-graphs. In what follows we have a weighted BWBP, where the weight of a path is the product of the weights of the edges on the path. The value of a weighted BWBP is the sum, over all s-t paths, of the weights of the paths.

Theorem 11 *Computing the permanent and determinant of a tridiagonal matrix over $\mathbb{Z}$ is equivalent to evaluating a layered planar weighted* BWBP *of width* 2.

Proof Given a tridiagonal matrix A, let A_i be the top-left submatrix of A of order i, and let X_n and Y_n denote its permanent and determinant respectively. We have the following recurrences:

$$X_0 = Y_0 = 1, \qquad X_1 = Y_1 = a_{1,1},$$
$$X_i = a_{i,i}X_{i-1} + a_{i-1,i}a_{i,i-1}X_{i-2}, \qquad Y_i = a_{i,i}Y_{i-1} - a_{i-1,i}a_{i,i-1}Y_{i-2}.$$

Fig. 1 shows a weighted branching program for X_n that has width 2 and can be drawn in a layered planar fashion. The construction for the determinant Y_n is similar, using some negative weights. This completes the proof of one direction.

We remark that in the construction for the permanent (X_n), when all entries are nonnegative, this problem reduces to counting paths in unweighted planar branching programs of width 5. To see this, replace each weighted edge in Fig. 1 with a width-three gadget having the appropriate number of paths in a standard way.

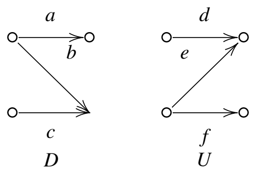

Fig. 2 Components of width-2 layered planar graphs

To see the other direction, notice that any layer of a planar width-2 BWBP should look like one of the following structures.

Any width-2 graph G corresponding to the BP can be encoded as a sequence of D and U components as indicated in Fig. 2. First consider the case where the sequence consists of alternating D and U; that is consider sequences in $(DU)^*$. Each such sequence looks exactly like the graph in Fig. 1. By just reading off the weights on the corresponding edges in the graph, we can produce two matrices M_1 and M_2 such that permanent of M_1 and the determinant of M_2 (by putting in appropriate negations) equal the value of the weighted BWBP.

Now it is sufficient to argue that the graph corresponding to any BWBP can be transformed to this form. If the string does not start with a D component, we will just put in a prefix D with $abc = 101$. Similarly, add a suffix U component with $def = 101$ if necessary. We need to handle the case when there are two consecutive components of the same type; UU or DD. Simply put in a D component with $abc = 101$ between two Us, and a U component with $def = 101$ between two Ds. Notice that the new width-2 graph when encoded will be an element of $(DU)^*$, and the weights of the paths are preserved in the transformation. The above reduction now gives the two matrices M_1 and M_2.

In addition, observe that if the BWBP is unweighted, then the matrix M_1 that we produce has only 0,1 entries, and M_2 will have entries from $\{-1, 0, 1\}$. □

From Theorem 11 and the discussion preceding it, we have the following corollary.

Corollary 12 *Computing the permanent and determinant of a tridiagonal matrix over $\mathbb{Z}$ is in* GapNC^1, *and is hard for* NC^1 *under* AC^0 [5] *reductions.*

4 Complexity Results on Rigidity

In this section we study the problem of computing matrix rigidity, $\textsc{rigid}_{\mathbb{K}}$, and also its restriction $\textsc{rigid}_{\mathbb{K},S}$ defined below, where the matrices can have entries only from $S \subseteq \mathbb{K}$.

$$\textsc{rigid}_{\mathbb{K},S} = \left\{ (M, r, k) \;\middle|\; \begin{array}{l} M \text{ over } S,\ \exists M' \text{ over S}: \\ \operatorname{rank}(M') < r \wedge \operatorname{support}(M - M') \leq k \end{array} \right\}.$$

We will mostly consider S to be either all of $\mathbb{K}$, or only $\mathbb{B} = \{0, 1\}$. We also consider the complexity of $\textsc{rigid}$ when k is fixed, via the following language:

$$\textsc{rigid}_{\mathbb{K},S}(k) = \{(M, r) \mid (M, r, k) \in \textsc{rigid}_{\mathbb{K},S}\}.$$

As mentioned in the introduction, matrix rigidity and matrix completion are related. The MinRank problem takes as input a matrix with variables, and asks for the minimum rank achievable under all instantiations of the variables in the underlying field, see for instance [8]. 1-MinRank is its restriction where every variable occurs at most once, and is also called *minimum rank completion*. MaxRank and 1-MaxRank are similarly defined. The naive algorithm for rigidity, mentioned in the introduction, easily translates to an upper bound of NP(1-MinRank). More precisely, RIGID is in $\exists \cdot$ 1-MinRank. While MinRank over $\mathbb{Z}$ is undecidable [8], this hardness proof does not carry over for 1-MinRank. Nonetheless, the best known upper bound for 1-MinRank is recursive enumerability. Thus the naive algorithm does not give any reasonable upper bound for RIGID.

Theorem 13 *For each fixed k, $\text{RIGID}_{\mathbb{Z},\mathbb{B}}(k)$ is complete for $\mathsf{C_{=}L}$.*

Proof Membership: We show that for each k, $\text{RIGID}_{\mathbb{Z},\mathbb{B}}(k)$ is in $\mathsf{C_{=}L}$. An instance (M, r) is in $\text{RIGID}_{\mathbb{Z},\mathbb{B}}(k)$ if there is a set of $0 \le s \le k$ entries of M, which, when flipped, yield a matrix of rank less than r. The number of such sets is bounded by $\Sigma_{s=0}^{k}\binom{n}{s} = t \in n^{O(1)}$. Let the corresponding matrices be denoted $M_1, M_2 \dots M_t$; these can be generated from M in logspace. Now $(M, r) \in \text{RIGID}_{\mathbb{Z},\mathbb{B}}(k) \iff \exists i : (M_i, r) \in \text{RANK BOUND}(\mathbb{Z})$. Hence $\text{RIGID}_{\mathbb{Z},\mathbb{B}}(k) \le_{dtt}^{\log} \text{RANK BOUND}(\mathbb{Z})$. Since $\text{RANK BOUND}(\mathbb{Z})$ is in $\mathsf{C_{=}L}$, and since $\mathsf{C_{=}L}$ is closed under logspace disjunctive truth-table reductions (see [6]), it follows that $\text{RIGID}_{\mathbb{Z},\mathbb{B}}(k)$ is in $\mathsf{C_{=}L}$.
Hardness: To show a corresponding hardness result, we use Lemma 3 below. The hardness for $\text{RIGID}_{\mathbb{Z},\mathbb{B}}(0)$ holds because SINGULAR remains $\mathsf{C_{=}L}$-hard even when restricted to 0-1 matrices (Proposition 4). Hardness for all the languages mentioned in the lemma also follows from this fact, and from the following claim:

(1) $M \in \text{SINGULAR}(\mathbb{Z}) \implies (M \otimes I_{k+1}, n(k+1)-k) \in \text{RIGID}_{\mathbb{Z},\mathbb{B}}(0) \subseteq \text{RIGID}_{\mathbb{Z},\mathbb{B}}(k)$.
(2) $M \notin \text{SINGULAR}(\mathbb{Z}) \implies (M \otimes I_{k+1}, n(k+1)-k) \notin \text{RIGID}_{\mathbb{Z}}(k)$.

Here $\otimes$ denotes tensor product and I_{k+1} denotes the $(k+1) \times (k+1)$ identity matrix. Note that $\text{rank}(M \otimes I_{k+1}) = (k+1)\text{rank}(M)$. To see the claim, observe that if $M \in \text{SINGULAR}(\mathbb{Z})$, then $\text{rank}(M) \le n-1$ and so $\text{rank}(M \otimes I_{k+1}) \le (k+1)(n-1) < n(k+1)-k$. If $M \notin \text{SINGULAR}(\mathbb{Z})$, then $\text{rank}(M \otimes I_{k+1}) = n(k+1)$. Thus we want to reduce the rank by at least $k+1$. By Lemma 3, we need to change at least $k+1$ entries. □

Remark 14 The membership bound of Theorem 13 crucially uses the fact that $\mathsf{C_{=}L}$ is closed under $\le_{dtt}^{\log}$ reductions. We also observe that this result holds for any finite S, even if S is not fixed *a priori* but supplied explicitly as part of the input.

The hardness of Theorem 13 essentially exploits the hardness of testing singularity. Therefore we now consider the complexity of RIGID at the singular-vs-non-singular threshold, i.e. when $r = n$. From Lemma 2 we know that over any field $\mathbb{F}$, (M, n, k) is in RIGID whenever $k \ge 1$. And $(M, n, 0)$ is in RIGID if and only if $M \in \text{SINGULAR}(\mathbb{F})$. So the complexity of deciding this predicate over $\mathbb{Q}$ is already well understood. We then address the question of how difficult it is to come up with a witnessing matrix.

Theorem 15 *Given a non-singular matrix M over $\mathbb{Q}$, a singular matrix N satisfying* $\text{support}(M-N)=1$ *can be constructed in* $\mathsf{L}^{\mathsf{GapL}}$.

Proof For each (i,j), let $M(i,j)$ be the matrix obtained from M by replacing $m_{i,j}$ with an indeterminate x. Then $\det(M(i,j))$ is of the form $ax+b$, and a and b can be determined in GapL (see for instance [2]). Since $R_M(n)=1$ (Lemma 2), there is at least one position (i,j) where the determinant is sensitive to the entry, and hence $a\neq 0$. Setting $m_{i,j}$ to be $-b/a$ gives the desired N. □

Another question that arises naturally is the complexity of RIGID at the singularity threshold over rings. Note that Lemma 2 does not necessarily hold for rings. For instance, changing one entry of a non-singular rational matrix M suffices to make it singular. But even if M is integral, the changed matrix may not be integral, and over $\mathbb{Z}$, $R_M(n)$ may well exceed 1. (It does, for the matrix $\left[\begin{smallmatrix}2&3\\5&7\end{smallmatrix}\right]$.) Thus, the question of deciding $R_M(n)$ over $\mathbb{Z}$ is non-trivial. We show:

Theorem 16 *Given $M\in\mathbb{Z}^{n\times n}$, deciding if (M,n,k) is in* RIGID($\mathbb{Z}$) *is* (1) *trivial for $k\geq n$,* (2) $\mathsf{C_{=}L}$ *complete for $k=0$, and* (3) *in* $\mathsf{L}^{\mathsf{GapL}}$ *for $k=1$.*

Proof (1) holds because zeroing out an entire row always gets singularity. (2) merely says that SINGULAR($\mathbb{Z}$) is $\mathsf{C_{=}L}$-complete. (3) follows from the proof of Theorem 15 and additionally checking the integrability of b/a. □

In particular, (3) above implies that if over $\mathbb{Z}$, $R_M(n)=1$, then the non-zero entry of a witnessing matrix is polynomially bounded in the size of M.

However, if $R_M(n)>1$ we do not know such a size bound. To demonstrate this difficulty, consider the case in which $k=2$. Following the general idea in Theorem 15, for each choice of two entries in the matrix, replace them by variables x and y. This defines a family of $\binom{n}{2}=O(n^2)$ matrices and a family $\mathcal{P}$ of bilinear bivariate polynomials representing the corresponding determinants. The coefficients of each $p\in\mathcal{P}$ can be computed in GapL. Now, to test if $R_M(n)\leq 2$, it suffices to check if at least one of the Diophantine equations defined by $p\in\mathcal{P}$ (or equivalently, the single multilinear Diophantine equation $q(x_1,x_2,\ldots,y_1,y_2\ldots)=\prod_{p\in\mathcal{P}}p(x_p,y_p)\ =0$) has an integral solution. However, we do not know how to do this.

5 Computing Bounded Rigidity

We now consider the bounded norm variant of rigidity described in Sect. 1: changed matrix entries can differ from the original entries by at most a pre-specified amount θ. Formally, the functions of interest are the *norm rigidity* $\Delta_M(r)$ and the *bounded rigidity* $R_M(r,\theta)$, as defined in [20], and their decision version, as given below.

$$\Delta_M(r)\overset{def}{=}\inf_N\left\{\sum_{i,j}|n_{i,j}|^2 : \text{rank}(M+N)<r\right\},$$

$$R_M(r,\theta) \overset{def}{=} \min_N \left\{\text{support}(N) : \text{rank}(M+N) < r, \forall i,j : |n_{i,j}| \le \theta\right\},$$

$$\text{B-RIGID}_{\mathbb{K}} = \{(M,r,k,\theta) \mid R_M(r,\theta) \le k\}.$$

Over $\mathbb{Z}$, the naive algorithm for B-RIGID$_{\mathbb{Z}}$ is now in NP. However over $\mathbb{Q}$, the bound θ still does not imply an *a priori* poly-size bound on the changed entries. Thus, unlike in Sect. 4, here computation over $\mathbb{Q}$ appears harder than over $\mathbb{Z}$.

The following lemma shows that the bounded rigidity functions can behave very differently from the standard rigidity function.

Lemma 17 *For any ϵ, and for any sufficiently large n such that $\frac{n}{\log n} > \epsilon + 1$, there is an $n \times n$ matrix M over $\mathbb{Q}$ such that $R_M(n) = 1$, $\Delta_M(n) = \Theta(4^n)$, and the bounded rigidity $R_M(n, n^\epsilon)$ is undefined.*

Proof Let M be an $n \times n$ diagonal matrix with $m_{i,i} = 2^n$ and $m_{i,j} = 0$ for $i \neq j$. Clearly, $R_M(n) = 1$; just zero out any diagonal entry. This involves a norm change of 4^n. Can M be made singular by a smaller norm-change, even allowing more entries to be changed? Recall the definition of strict diagonal dominance from Sect. 3. We invoke the Levy-Desplanques theorem (see for instance Theorem 2.1 in [21]) that says that the determinant of a strictly diagonally dominant matrix is non-zero. Now, a total norm-change less than 4^n will not destroy strict diagonally dominance, and the matrix will remain non-singular. Hence $\Delta_M(n) = 4^n$, and $R_M(n, n^\epsilon)$ is undefined. □

Since for a given matrix M, a rank r and a bound θ, $R_M(r,\theta)$ can be undefined, we examine how difficult is it to check this. We show the following:

Theorem 18

1. *Given a matrix $M \in \mathbb{Q}^{n\times n}$, and a rational number $\theta > 0$, testing if $R_M(n,\theta)$ is defined is* NP*-complete.*
2. *Given M and θ as above, and further given an integer k, testing if $R_M(n,\theta)$ is at most k is* NP*-complete.*

Proof To begin with, notice that $R_M(r,\theta)$ is defined if and only if $R_M(r,\theta) \le n^2$.

Membership: We first show the membership in NP for (2). Membership in (1) follows by using this with $k = n^2$. We use a result from the *linear interval equations* literature. For two matrices A and B, we say that $A \le B$ if for each i,j, $A_{ij} \le B_{ij}$. For $A \le B$, the interval of matrices $[A,B]$ is the set of all matrices C such that $A \le C \le B$. An interval is said to be singular if it contains at least one singular matrix; otherwise it is said to be regular. By Theorem 2.8 of [26] (or directly from Lemma 21), checking singularity of a given interval matrix is in NP.

Given M, θ and k, we want to test whether $R_M(n,\theta)$ is at most k. In NP, we guess k positions $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$ and construct the matrix $V_{i_m j_m} = \theta$ for all $1 \le m \le k$ and 0 elsewhere. Now let $\underline{A} = M - V$ and $\overline{A} = M + V$. Then $R_M(n,\theta) \le k$ if and only if for some such guessed V, the interval $[\underline{A}, \overline{A}]$ is singular, and this can be tested in NP.

Hardness: It suffices to prove hardness for (1), since hard instances of (1) along with $k = n^2$ gives hard instances of (2). We start with the *maximum bipartite subgraph problem*: Given an undirected graph $G = (V, E)$, with n vertices and m edges and a number k, check whether there is bipartite subgraph with at least k edges. This problem is known to be NP-complete (see [16]). In [26], there is a reduction from this problem to computing the *radius of non-singularity*, defined as follows: Given a matrix A, its radius of non-singularity $d(A)$ is the minimum $\epsilon > 0$ such that the interval $[A - \epsilon J, A + \epsilon J]$ is singular, where J is the all-1s matrix.

We sketch the reduction of [26] below and observe that it yields NP-hardness for our problem as well.

Given an instance G, k of the maximum bipartite subgraph problem, we define the matrix N as,

$$N_{ij} = \begin{cases} -1 & \text{if } i \neq j \text{ and } i \text{ and } j \text{ are adjacent in } G, \\ 2m+1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that since N is diagonally dominant, by Levy-Desplanques theorem (see for instance Theorem 2.1 in [21]), N is invertible. Let $M = N^{-1}$.

By Theorems 2.6 and 2.2 of [26],

$$\begin{aligned} (G, k) \text{ is a Yes instance} &\iff 1/d(M) \geq (2m+1)n + 4k - 2m \\ &\iff d(M) \leq \theta = \frac{1}{(2m+1)n + 4k - 2m} \\ &\iff \text{the interval } [M - \theta J, M + \theta J] \text{ is singular} \\ &\iff R_M(n, \theta) \text{ is defined.} \end{aligned}$$ □

Remark 19

1. It is easy to see that, by clearing denominators, we have hard instances where M, θ take integral values. Thus, the hardness result holds for $\mathbb{Z}$ as well.
2. The matrices that are produced in the above reduction are all symmetric as well. Rohn [29] considered the case when the interval of matrices under consideration is symmetric; that is both the boundary matrices are symmetric. Notice that the interval can still contain non-symmetric matrices. He proved that in such an interval, if there is a singular matrix, then there must be a symmetric singular matrix too.

Unraveling the NP algorithm described in the membership part of Theorem 18, and its proof of correctness, is illuminating. Essentially, what is established in [28] and used in [26] is the following:

Lemma 20 [28] *If an interval $[A, B]$ is singular, i.e. the determinant vanishes for some matrix C within the bounds $A \leq C \leq B$, then the determinant vanishes for a matrix $D \in [A, B]$ which, at all but at most one position, takes an extreme value (d_{ij} is either a_{ij} or b_{ij}).*

In particular, this implies that there is a matrix in the interval whose entries have representations polynomially long in that of A and B. To see this, let D be the matrix claimed to exist as above, and let k, l be the (only) position where $a_{kl} < d_{kl} < b_{kl}$. The other entries of D match those of A or B and hence are polynomially bounded anyway. Now replace d_{kl} by a variable x to get matrix D_x. Its determinant is a univariate linear polynomial $\alpha x + \beta$ which vanishes at $x = d_{kl}$. Now α and β can be computed from D_x in GapL, and hence are polynomially bounded. If $\alpha = 0$, then $\beta = 0$ and the polynomial is identically zero. Otherwise, the zero of the polynomial is $-\beta/\alpha$. Either way, there is a zero with a polynomially long representation.

In [28], the above lemma is established as part of a long chain of equivalences concerning determinant polynomials. However, it is in fact a general property of arbitrary multilinear polynomials, as we show below.

Lemma 21 (Zero-on-an-Edge Lemma) *Let $p(x_1, \ldots, x_t)$ be a multilinear polynomial over $\mathbb{Q}$. If it has a zero in the hypercube H defined by $[\ell_1, u_1], \ldots, [\ell_t, u_t]$, then it has a zero on an edge of H, i.e. a zero $(a_1, \ldots, a_t)$ such that for some k, $\forall(i \neq k)$, $a_i \in \{\ell_i, u_i\}$.*

Proof The proof is by induction on the dimension of the hypercube. The case when $t = 1$ is vacuously true, since H is itself an edge. Consider the case $t = 2$. Let $p(x_1, x_2)$ be the multilinear polynomial which has a zero (z_1, z_2) in the hypercube H; $\ell_i \leq z_i \leq u_i$ for $i = 1, 2$. Assume, to the contrary, that p has no zero on any edge of H. Define the univariate polynomial $q(x_1) = p(x_1, z_2)$. Since $q(x_1)$ is linear and vanishes at z_1, $p(\ell_1, z_2)$ and $p(u_1, z_2)$ must be of opposite sign. But the univariate linear polynomials $p(\ell_1, x_2)$ and $p(u_1, x_2)$ do not change signs on the edges either, and so $p(\ell_1, u_2)$ and $p(u_1, u_2)$ also have opposite sign. By linearity of $p(x_1, u_2)$, there must be a zero on the edge $x_2 = u_2$, contradicting our assumption.

Let us assume the statement for hypercubes of dimension less than t. Consider the hypercube of dimension t and the polynomial $p(x_1, \ldots, x_t)$. Let $(z_1, \ldots, z_t)$ be the zero inside the hypercube. The multilinear polynomial r corresponding to $p(x_1, \ldots, x_{n-1}, z_t)$ has a zero inside the $(t-1)$-dimensional hypercube H' defined by intervals $[\ell_1, u_1], \ldots, [\ell_{t-1}, u_{t-1}]$. By induction, r has a zero on an edge of H'. Without loss of generality, assume that this zero is $(z'_1, \alpha_2 \ldots \alpha_{t-1})$ where $\alpha_i \in \{\ell_i, u_i\}$. Thus the polynomial $q(x_1, x_t) = p(x_1, \alpha_2 \ldots \alpha_{t-1}, x_t)$ has a zero in the hypercube defined by intervals $[\ell_1, u_1], [\ell_t, u_t]$. Hence the base case applies again, completing the induction. □

Analogous to Theorems 13, 15 and 16, we consider B-RIGID$_{\mathbb{K}}$ when $k \in O(1)$.

Theorem 22 B-RIGID$_{\mathbb{Q}}$ *and* B-RIGID$_{\mathbb{Z}}$ *are* C$_=$L*-hard for each fixed choice of k, and remain hard when $r = n$. When $k = 1$ and $r = n$,* B-RIGID$_{\mathbb{Q}}$ *is in* PL, *while* B-RIGID$_{\mathbb{Z}}$ *is in* $\mathsf{L}^{\mathsf{GapL}}$.

Proof For any k, $(M, n, k, 0) \in$ B-RIGID$_{\mathbb{K}}$ $\Longleftrightarrow$ M is singular; hence C$_=$L-hardness.

To see the PLupper bound over $\mathbb{Q}$, let $\theta = \frac{p}{q}$. For each element (i, j), define the $(i, j)^{th}$ element as variable x and then write the determinant as $ax + b$. Thus, if $|x| =$

$|\frac{b}{a}| \leq \frac{p}{q}$ for at least one such (i, j) pair, we are done. This is equivalent to checking if $(bq)^2 \leq (ap)^2$. The values of a and b can be written as determinants, hence $(ap)^2$ and $(bq)^2$ are GapL functions, and comparison of two GapL functions can be done in PL. Since PL is closed under disjunction (see [6]), the entire computation can be done in PL. Over $\mathbb{Z}$, $q = 1$ and $\theta = p$, but we need an integral value for x as well. That is, we want an (i, j) pair where $|\frac{b}{a}| \leq \theta$ and a divides b. This can be checked in $\mathsf{L}^{\mathsf{GapL}}$. □

6 Discussion

While the matrix rigidity problem over finite fields is NP-complete ([14]), we can consider restricted versions there too. It is known [7] that SINGULAR($\mathbb{F}_p$) is complete for $\mathsf{Mod}_p\mathsf{L}$ (computing the exact value of the determinant over $\mathbb{F}_p$ is in $\mathsf{Mod}_p\mathsf{L}$), and that (see e.g. [5]), for any prime p, RANK BOUND($\mathbb{F}_p$) is in $\mathsf{Mod}_p\mathsf{L}$. Using this, and closure properties of $\mathsf{Mod}_p\mathsf{L}$, we can obtain analogues of Theorem 13 and 15 over $\mathbb{F}_p$: (1) for each k, and each prime p, $\text{RIGID}_{\mathbb{F}_p}(k)$ is complete for $\mathsf{Mod}_p\mathsf{L}$, and (2) given a non-singular matrix, a singular matrix can be obtained by changing just one entry, and the change can be computed in $\mathsf{Mod}_p\mathsf{L}$.

We can also consider the complementary question to matrix rigidity, namely, computing the number of entries that need to be changed to increase the rank above a given value. Using arguments similar to the case of decreasing rank, we can obtain similar complexity results in this case also. However, notably in this case, we not only have decidability, we also have an upper bound of NP. This follows from the framework of *maximum rank matrix completion*, which is known to be in P[15, 23].

For the most general question of testing rigidity over arbitrary infinite fields, as an optimization problem, a natural direction to explore is the existence of fixed parameter tractable algorithms. More specifically, given an $n \times n$ matrix, and rank r and a value k, is it possible to test $R_M(r) \leq k$ in time $n^c.f(k)$ for a constant c and an arbitrary function f. However, in this problem we do not see how such additional time can used.

Acknowledgements We thank V. Arvind, N.S. Narayanaswamy, R. Balasubramanian, Kapil Paranjape and Raghav Kulkarni for many insightful discussions. We thank the anonymous referees for useful comments which helped improve the readability of the paper. Further, one of the referees pointed out how to improve the bound on the reduction in Theorem 5 from TC^0 to AC^0.

References

1. Allender, E., Ambainis, A., Mix Barrington, D.A., Datta, S., LeThanh, H.: Bounded-depth arithmetic circuits: counting and closure. In: Proc. 26th ICALP. Lecture Notes in Computer Science, vol. 1644, pp. 149–158. Springer, Berlin (1999)
2. Allender, E., Arvind, V., Mahajan, M.: Arithmetic complexity, Kleene closure, and formal power series. Theory Comput. Syst. **36**(4), 303–328 (2003)
3. Allender, E., Beals, R., Ogihara, M.: The complexity of matrix rank and feasible systems of linear equations. Comput. Complex. **8**(2), 99–126 (1999)
4. Àlvarez, C., Greenlaw, R.: A compendium of problems complete for symmetric logarithmic space. Comput. Complex., **9**, 73–95 (2000)

5. Allender, E.: Arithmetic circuits and counting complexity classes. In: Krajicek, J. (ed.) Complexity of Computations and Proofs. Quaderni di Matematica, vol. 13, pp. 33–72. Seconda Universita di Napoli, Napoli (2004)
6. Allender, E., Ogihara, M.: Relationships among PL, #l, and the determinant. RAIRO Theor. Inform. Appl. **30**(1), 1–21 (1996)
7. Buntrock, G., Damm, C., Hertrampf, U., Meinel, C.: Structure and importance of logspace MOD-classes. Math. Syst. Theory **25**, 223–237 (1992)
8. Buss, J.F., Frandsen, G.S., Shallit, J.: The computational complexity of some problems of linear algebra. J. Comput. Syst. Sci. **58**, 572–596 (1999)
9. Bollobas, B.: Modern Graph Theory. GTM, vol. 184. Springer, Berlin (1984)
10. Cook, S.A., McKenzie, P.: Problems complete for L. J. Algorithms **8**, 385–394 (1987)
11. Caussinus, H., McKenzie, P., Thérien, D., Vollmer, H.: Nondeterministic NC^1 computation. J. Comput. Syst. Sci. **57**, 200–212 (1998)
12. Dahl, G.: A note on nonnegative diagonally dominant matrices. Linear Algebra Appl. **317**, 217–224 (1999)
13. Damm, C.: $\mathrm{DET} = \mathrm{L}^{(\#\mathrm{L})}$. Technical Report Informatik-Preprint 8, Fachbereich Informatik der Humboldt–Universität zu Berlin (1991)
14. Deshpande, A.: Sampling-based dimension reduction algorithms. PhD thesis, MIT, May 2007
15. Geelen, J.F.: Maximum rank matrix completion. Linear Algebra Appl. **288**(1–3), 211–217 (1999)
16. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
17. Grigoriev, D.Y.: Using the Notions of Separability and Independence for Proving the Lower Bounds on the Circuit Complexity. Notes of the Leningrad Branch of the Steklov Mathematical Institute. Nauka, Moscow (1976) (in Russian)
18. Kulkarni, R.: Personal communication, January 2007
19. Laurent, M.: Matrix completion problems. In: Floudas, C.A., Pardalos, P.M. (eds.) The Encyclopedia of Optimization, vol. 3, pp. 221–229. Kluwer, Dordrecht (2001)
20. Lokam, S.V.: Spectral methods for matrix rigidity with applications to size-depth tradeoffs and communication complexity. In: Proc. 36th FOCS, pp. 6–15, 1995; J. Comput. Syst. Sci. **63**(3):449–473 (2001)
21. Marcus, M., Minc, H.: A Survey of Matrix Theory and Matrix inequalities. The Prindle, Weber and Schmidt Complementary Series in Mathematics, vol. 14. Allyn and Bacon, Boston (1964)
22. Mulmuley, K.: A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. Combinatorica **7**, 101–104 (1987)
23. Murota, K.: Mixed matrices—irreducibility and decomposition. In: Brualdi, R.A., Friedland, S., Klee, V. (eds.) Combinatorial and Graph-Theoretic Problems in Linear Algebra. The IMA Volumes in Mathematics and Its Applications, vol. 50, pp. 39–71. Springer, Berlin (1993)
24. Mahajan, M., Vinay, V.: Determinant: combinatorics, algorithms, complexity. Chic. J. Theor. Comput. Sci. **5** (1997)
25. Nisan, N., Ta-Shma, A.: Symmetric logspace is closed under complement. Chic. J. Theor. Comput. Sci., (1995).
26. Poljak, S., Rohn, J.: Checking robust nonsingularity is NP-hard. Math. Control Signals Syst. **6**, 1–9 (1993)
27. Reingold, O.: Undirected st-conenctivity in logspace. In Proc. 37th STOC, pp. 376–385 (2005).
28. Rohn, J.: Systems of linear interval equations. Linear Algebra Appl. **126**, 39–78 (1989)
29. Rohn, J.: Checking positive definiteness or stability of symmetric interval matrices is NP-hard. Comment. Math. Univ. Carol. **35**, 795–797 (1994)
30. Razborov, A.A., Rudich, S.: Natural proofs. J. Comput. Syst. Sci. **55**(1), 24–35 (1997)
31. Toda, S.: Counting problems computationally equivalent to the determinant. Technical Report CSIM 91-07, Dept. of Comput. Sci. & Information Mathematics, Univ of Electro-Communications, Chofu-shi, Tokyo (1991)
32. Valiant, L.G.: Graph theoretic arguments in low-level complexity. In: Proc. 6th MFCS. Lecture Notes in Computer Science, vol. 53, pp. 162–176. Springer, Berlin (1977)
33. Valiant, L.G.: Why is Boolean complexity theory difficult? In: Proceedings of the London Mathematical Society symposium on Boolean function complexity, pp. 84–94. Cambridge University Press, New York (1992)
34. Vinay, V.: Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In: Proc. 6th Structure in Complexity Theory Conference. Lecture Notes in Computer Science, vol. 223, pp. 270–284. Springer, Berlin (1991)
35. Vollmer, H.: Introduction to Circuit Complexity: A Uniform Approach. Springer, Berlin (1999)