# ON THE ENUMERABILITY OF
# THE DETERMINANT AND THE RANK

Alina Beygelzimer, Mitsunori Ogihara*
*Department of Computer Science, University of Rochester, Rochester, NY 14627*
{beygel,ogihara}@cs.rochester.edu

**Abstract**    We investigate the complexity of enumerative approximation of two elementary problems in linear algebra, computing the rank and the determinant of a matrix. In particular, we show that if there exists an enumerator that, given a matrix, outputs a list of constantly many numbers, one of which is guaranteed to be the rank of the matrix, then it can be determined in $AC^0$ (with oracle access to the enumerator) which of these numbers is the rank. Thus, for example, if the enumerator is an FL function, then the problem of computing the rank is in FL. The result holds for matrices over any commutative ring whose size grows at most polynomially with the size of the matrix. The existence of such an enumerator also implies a slightly stronger collapse of the exact counting logspace hierarchy. For the determinant function we establish the following two results: (1) If the determinant is poly-enumerable in logspace, then it can be computed exactly in FL. (2) For any prime $p$, if computing the determinant modulo $p$ is $(p-1)$-enumerable in FL, then computing the determinant modulo $p$ can be done in FL. This gives a new perspective on the approximability of many elementary linear algebra problems equivalent to computing the rank or the determinant.

## Introduction

Valiant (Valiant, 1979) proved that the permanent of integer matrices characterizes #P, the class of functions definable as the number of accepting computations of a nondeterministic polynomial-time Turing machine. A similar connection has been shown between the complexity of computing the determinant and #L, the logspace analog of #P (Toda, 1991; Valiant, 1992; Vinay, 1991; Damm, 1991). (Since the determinant of integer matrices can take on negative values, the determinant is in fact complete for GapL, the class of functions that can be expressed as the difference of two #L functions.) Toda's

result (Toda, 1989) showing the surprising power of counting in the context of nondeterministic polynomial-time, namely that the polynomial hierarchy is contained in $P^{\#P}$, naturally raises the question of whether $\#P$ functions are at least easy to approximate. In the standard sense of coming close to the correct value (i.e., within a multiplicative factor), approximating $\#P$ functions is in $\Delta_3^p$. Moreover, any technique for showing that it has complexity lower than $\Delta_2^p$, would have to be non-relativizable. (See (Stockmeyer, 1985).) In search of a better answer, Cai and Hemachandra (Cai and Hemachandra, 1989) proposed an alternative notion of approximation, called enumerative counting. Instead of restricting the range of a function value to an interval, they consider enumerating a short list of (not necessarily consecutive) values, guaranteed to contain the correct one. Which of these approximation tasks is more natural depends on the function one is trying to approximate: enumerative counting is more suitable in cases when there is no natural total ordering on the range of the function; or when the range is either exponentially large (as in the case of the determinant), or some intervals of the range are substantially denser than others, and it is preferred to obtain a fixed number of candidates for every input of the same length, instead of dealing with an interval that, though bounded, may contain vastly different numbers of possible values for different inputs (depending *not* on the input itself, but on the value of the function on this input).

More formally, a function $f$ is said to be $g(n)$-*enumerable* if there exists a function that, on input $x$, outputs a list of at most $g(|x|)$ values that is guaranteed to contain $f(x)$. Cai and Hemachandra (Cai and Hemachandra, 1991), and also Amir, Beigel, and Gasarch (Amir et al., 1990), showed that if the permanent function is poly-enumerable in polynomial time, then $P = P^{\#P}$. (Their result is actually stated for $\#SAT$, another classical $\#P$-complete function.) Thus if we are given an $n \times n$ integer matrix $A$, and instead of $n!(2^n)^n$ values that perm($A$) can possibly take, we get a restricted list of polynomially-many values, guaranteed to contain perm($A$), then we can easily (in polynomial time) determine which one is the correct one. This is certainly a *non*-enumerability result, as it says that a very hard to compute function (hard for the whole polynomial hierarchy) can easily bootstrap its exact value if it is left with polynomially many candidates.

This paper investigates the enumerability of functions complete for logspace counting classes. In particular, it is interesting whether enumerability implies a similar decrease in the complexity of the determinant; and if so, to what extent. Since $\#L$ functions have significantly less computational power (they are contained in the $NC^2$, or even $TC^1$), the enumerability properties of logspace counting analogs can be very different from those of $\#P$-complete functions. This is even more interesting, because, as far as we know, there are no results on approximating the determinant (or the rank) in the standard sense. Another purpose of such an investigation is to get a better understanding of the relationships among the complexity classes sandwiched between NL and uniform $TC^1$.

We show that if there exists an enumerator that, given a matrix, outputs a list of constantly many numbers, one of which is guaranteed to be the rank of the matrix, then it can be determined in $AC^0$ (with oracle access to the enumerator) which of these numbers is the rank. Thus, for example, if the enumerator runs in logspace, then the problem of computing the rank is in logspace. The result holds for any commutative ring with identity whose size grows polynomially with the size of the matrix. The existence of such an enumerator implies a slightly stronger collapse of the exact counting logspace hierarchy; namely, it shows that the the hierarchy collapses to the closure of its base class $C_=L$ under $\leq^{AC^0}_{O(1)\text{-tt}}$ reductions. We also consider a related problem of computing the number of dependent vectors in a given set, and prove it to be $\leq^{AC^0}_{O(1)\text{-tt}}$-equivalent to computing the rank. For the determinant function we show that if the determinant is poly-enumerable in logspace, then it can be computed exactly in logspace. We also establish a similar result for computing the determinant modulo any prime.

## 1.    Preliminaries

We will be concerned with the complexity of computing the following functions (with input and output in $\{0,1\}^*$):

- DET$_F$: Given $A \in F^{n \times n}$, compute the determinant of $A$.
- RANK$_F$: Given $A \in F^{n \times n}$, compute the rank of $A$.
- SINGULARITY$_F$: Given $A \in F^{n \times n}$, determine whether $A$ is singular.
- INDEPENDENCE$_F$: Given a set of vectors in $F^n$, determine whether they are linearly independent.

Here $F$ is any commutative ring with unity. When $F = \mathbb{Z}$, the ring of integers, $F$ is dropped. With some abuse of notation, we will consider these functions as sets by associating the function $f : \{0,1\}^* \to \{0,1\}^*$ with the set $\{(x, i) \mid$ the $i$-th bit of $f(x)$ is $1\}$. Notice that a function is in $NC^k$ if and only if its associated set (its characteristic function) is in $NC^k$ and is polynomially bounded; hence there will be no confusion in viewing circuit classes as functional, sets being identified with their characteristic functions.

**Uniformity.**    In order to make circuit classes comparable to traditional classes defined by time and space, we need to place uniformity restrictions on circuit families. For our purposes it will be sufficient to use logspace uniformity, meaning that there exists a logspace machine that, on input $1^n$, generates a description of the circuit for $n$ inputs. For a detailed treatment of uniformity and a discussion of other uniformity conditions, see (Ruzzo, 1981; Barrington and Immerman, 1997).

**Reductions.**    We use Wilson's model (Wilson, 1985) of oracle circuits to define the reductions. A function $f$ is $AC^0$-*reducible* to function $g$, if there is a logspace uniform $AC^0$ family of circuits that computes $f$, where in addition

to the usual gates, oracle gates for $g$ are allowed. Similarly we define $\mathrm{NC}^1$-*reducibility* except that now the circuits have a bounded fan-in, and thus an oracle gate with fan-in $m$ has to count as depth $\log m$. For a circuit class $\mathcal{C}$, we write $\mathcal{C}(f)$ to denote the class of functions $\mathcal{C}$-reducible to $f$. For a function class $\mathcal{F}$, $\mathcal{C}(\mathcal{F})$ denotes the class of functions $\mathcal{C}$-reducible to some function in $\mathcal{F}$. A function $f$ is $\mathrm{AC}^0$ *many-one reducible* to a function $g$ (written as $f \leq_m^{\mathrm{AC}^0} g$) is there exists an $\mathrm{AC}^0$ family of circuits $\langle C_n \rangle$ such that for every $x$ of length $n$, we have $f(x) = g(C_n(x))$.

## Logspace counting classes and the complexity of problems in linear algebra

Many basic linear algebra problems are known to be in $\mathrm{NC}^2$. In order to classify and capture the exact complexity of these problems, Cook (Cook, 1985) defined the class of problems $\mathrm{NC}^1$-reducible to the determinant of integer matrices, and showed that most linear algebra problems with fast parallel algorithms are in this class. Many are in fact complete for this class; others were shown to be complete for the (potentially smaller) class of problems reducible to computing the rank (von zur Gathen, 1993; Santha and Tan, 1998). Santha and Tan (Santha and Tan, 1998) defined a more refined hierarchy of problems that reflects the computational difference between the functional and the verification versions of the problems under $\mathrm{AC}^0$-Turing and $\mathrm{AC}^0$-many-one reductions. Toda (Toda, 1991) gave many examples of graph-theoretic problems that are (under appropriate reductions) equivalent to computing the determinant. (Although some of these problems still have natural matrix interpretations when graphs are identified with their adjacency matrices.)

Allender and Ogihara (Allender and Ogihara, 1996) observed that, even though for most natural problems the closures under $\mathrm{AC}^0$ and $\mathrm{NC}^1$-reductions coincide, this does not seem to be apparent for the determinant. This motivated the definition of the following hierarchies (defined using the "Ruzzo-Simon-Tompa" oracle access model (Ruzzo et al., 1984), which is standard for defining Turing reductions for space-bounded nondeterministic machines, see (Allender and Ogihara, 1996)):

- The exact counting logspace hierarchy $\mathrm{C_{=}L} \cup \mathrm{C_{=}L}^{\mathrm{C_{=}L}} \cup \mathrm{C_{=}L}^{\mathrm{C_{=}L}^{\mathrm{C_{=}L}}} \cdots = \mathrm{AC}^0(\mathrm{C_{=}L})$

    The class $\mathrm{C_{=}L}$ is defined as the class of languages, for which there exists a GapL function $f$ such that for every $x$, $x$ is in the language if and only if $f(x) = 0$. If follows immediately that the set of singular matrices is complete for $\mathrm{C_{=}L}$. Allender, Beals, and Ogihara (Allender et al., 1999) showed that $\mathrm{AC}^0$ and $\mathrm{NC}^1$ reducibilities coincide on $\mathrm{C_{=}L}$; furthermore, the hierarchy collapses to $\mathrm{L}^{\mathrm{C_{=}L}}$. We show that, if RANK is $O(1)$-enumerable in logspace, then it collapses to the $\leq_{O(1)\text{-tt}}^{\mathrm{AC}^0}$-closure of $\mathrm{C_{=}L}$.

- The PL hierarchy $\mathrm{PL} \cup \mathrm{PL}^{\mathrm{PL}} \cup \mathrm{PL}^{\mathrm{PL}^{\mathrm{PL}}} \cup \cdots = \mathrm{AC}^0(\mathrm{PL})$

Ogihara (Ogihara, 1998) showed that the PL hierarchy collapses to PL under $AC^0$ (in fact $TC^0$) reductions, which was improved to $NC^1$ reducibility by Beigel and Fu (Beigel and Fu, 1997). A problem, easily seen to be complete for PL, is checking whether the determinant of integer matrices is positive.

■ The #L hierarchy $L \cup L^{\#L} \cup L^{\#L^{\#L}} \cdots = AC^0(\#L) = AC^0(DET)$

It is not known whether the #L hierarchy collapses, or whether $AC^0(\#L) = NC^1(\#L)$. The latter would imply the collapse (Allender, 1997).

Allender et al. (Allender et al., 1999) showed that the problems of computing the rank of integer matrices, determining whether the rank is odd, and determining the solvability of a system of linear equations, are all complete for $AC^0(C_=L)$. Clearly, the problems of computing and verifying the rank of a matrix are $AC^0$-equivalent (since there are just $n+1$ possibilities for the rank). However, Allender et al. classified the complexity of verifying the rank exactly, showing that it is complete for the second level of the Boolean Hierarchy above $C_=L$ (i.e. the class of sets expressible as an intersection of a $C_=L$ and a co-$C_=L$ set).

**Our results.** We show that if there exists a logspace computable $O(1)$-enumerator for RANK, then RANK is $\leq^{AC^0}_{O(1)\text{-tt}}$-reducible to INDEPENDENCE, and thus to SINGULARITY. (The reduction holds for arbitrary rings.) SINGULARITY is complete for $C_=L$, and thus the existence of the enumerator implies that $AC^0(C_=L)$ coincides with the closure of $C_=L$ under $O(1)$-tt-reductions, a slight improvement over $O(\text{poly}(n))$-tt that follows from (Allender et al., 1999) (unconditionally). We also show that if $RANK_F$ is $O(1)$-enumerable in logspace, then $RANK_F \in FL$, where $F$ is any commutative ring with identity whose size grows at most polynomially with the size of the input matrix. Finally, we consider a related problem of computing the number of dependent vectors in a given set (i.e., vectors involved in some non-trivial linear dependencies with other vectors in the set), and show it to be $\leq^{AC^0}_{O(1)\text{-tt}}$-equivalent to computing the rank. For the determinant function DET we establish the following two results: (1) If DET is poly-enumerable in logspace, then DET is in FL. (2) For any prime $p$, if the determinant-modulo-$p$ function is $(p-1)$-enumerable in FL, then it can be computed exactly in FL.

**Organization of the paper.** All the results pertaining to the rank and the determinant are collected in Sections 3 and 4, respectively. Section 5 concludes with a discussion.

## 2.    Enumerability of the Rank

Recall that a function $f$ is logspace $g(n)$-enumerable if there exists a logspace computable function that, on input $x$, outputs a list of at most $g(|x|)$ values, one

of which is $f(x)$. The following lemma shows how to combine several matrices into a single matrix such that the ranks of the original matrices can be read off the rank of the combined matrix.

**Lemma 2.1** [Block diagonal construction] There exists a logspace computable function $S$ that given an ordered list $Q = \langle A_1, \ldots, A_q \rangle$ of $n \times n$ matrices, outputs a single matrix $S(Q)$ of dimension $O(n^q)$ such that a logspace procedure can uniquely decode the sequence of ranks $\langle \text{rank}(A_1), \ldots, \text{rank}(A_q) \rangle$ from the value of $\text{rank}(S(Q))$. Moreover, both procedures can be implemented by uniform $AC^0$ circuit families.

**Proof:** Consider the following combining construction. On input $Q = \langle A_1, \ldots, A_q \rangle$, $S$ outputs a block diagonal matrix (i.e. a matrix of $n \times n$ blocks sitting on the main diagonal) with the following block structure. The first block of $S(Q)$ corresponds to $A_1$, the next $(n+1)$ blocks correspond to $A_2$, and so on, until we get to $n^q + n^{q-1} + \cdots + 1$ blocks of $A_q$. The multiplicity of $A_i$ as a block is $\sum_{j=1}^{i} n^{j-1}$, thereby the dimension of $S(Q)$ is $\sum_{i=1}^{q} n^{i+1}(q - i + 1) = O(n^q)$. The rank of $S(Q)$ is the sum of all block ranks, and since the rank of each block is at most $n$, the original sequence of ranks $\langle \text{rank}(A_1), \ldots, \text{rank}(A_q) \rangle$ can be read off from the value of $\text{rank}(S(Q))$. It is easy to see that both the construction and the decoding can be done in uniform $AC^0$. ∎

The combining construction above allows one to eliminate many candidate rank sequences. For example, if we were to feed each of $q$ matrices to an $r$-enumerator separately, we would get $r^q$ purported rank sequences, whereas combining the matrices into a single query reduces the number of candidates to $r$. In order for the dimension of $S(Q)$ to be polynomial in $n$, the number of matrices, $q$, has to be constant. A simple information-theoretic argument shows that this is the best possible. Indeed, the dimension of a matrix whose rank can encode $(n+1)^q$ possible rank sequences must be at least $(n+1)^q$. Notice that combining matrices into a single query to an $r$-enumerator allows one to link matrices in the following sense.

**Definition 1** Two $r$-element sequences $\{p_1, \ldots, p_r\}$ and $\{q_1, \ldots, q_r\}$ are said to be *linked* if $p_i = p_j$ if and only if $q_i = q_j$, for all $1 \le i < j \le r$.

In other words, two matrices are linked – relative to an enumerator – if there is a direct correspondence between the values on their claimed lists of ranks; hence knowing the rank of one immediately gives the rank of the other.

**Claim 2.1** There exists $r_0$ such that for any $r > r_0$, any set of $(\frac{r}{\ln r})^r$ $r$-element sequences, contains at least one linked pair.

**Proof:** The number of $r$-element sequences sufficient to guarantee the existence of a linked pair is precisely one more than the number of partitions of an $r$-element set into non-empty subsets. The latter is known as the $r$th Bell number, $B_r$. De Bruijn (de Bruijn, 1970) gave the asymptotic formula

$$\ln B_r r = \ln r - \ln\ln r - 1 + \ln\ln r \ln r + 1\ln r + 12\left(\ln\ln r \ln r\right)^2 + O\left(\ln\ln r(\ln r)^2\right),$$

immediately yielding the claim. Other (less explicit) asymptotic approximations for $B_r$ are known (see, for example, (Lovász, 1993)). ∎

Let $\kappa(r)$ be the minimum number of matrices that are guaranteed to contain a linked pair. By Claim 2.1, $\kappa \stackrel{\text{def}}{=} \kappa(r) \le (\frac{r}{\ln r})^r$. Given an $n \times n$ matrix $A$, let $A_i$ (for $1 \le i \le n$) denote the $n \times n$ matrix with the first $i$ rows of $A$ and 0s elsewhere, so that $A_n = A$. An $r$-enumerator for the rank function defines the *equivalence graph* of $A$, a labeled graph on $[n] \stackrel{\text{def}}{=} \{1, \dots, n\}$ with the set of nodes corresponding to $\mathcal{A} = \{A_1, \cdots, A_n\}$, and an edge between nodes $i$ and $j$ if and only if there is a set of $\kappa - 2$ matrices in $\mathcal{A} - \{A_i, A_j\}$ certifying the equivalence between $A_i$ and $A_j$ (i.e. witnessing that $A_i$ and $A_j$ are linked). (We can assume without loss of generality that the enumerator is deterministic and the combining encoding of queries in Lemma 2.1 is symmetric.) The label of an edge is defined by the equivalence (i.e. direct correspondence between the $r$ claimed values for rank($A_i$) and the $r$ claimed values for rank($A_j$)) given by the lexicographically smallest $\kappa$-tuple linking $A_i$ and $A_j$.

Notice that by definition every subset of $\kappa$ nodes in the equivalence graph induces at least one edge. Hence the number of connected components in the equivalence graph is at most $\kappa - 1$. The following proposition shows that in this case every pair of nodes is connected by a short path (where the length of a path is the number of edges it contains). The proof can be found in (Beygelzimer and Ogihara, 2002).

**Proposition 2.1** *Any pair of nodes in the equivalence graph is connected by a path of length at most $2\kappa - 3$.*

We will use equivalence graphs in the proof of the theorem below.

**Theorem 1** *If, for some integer $r$, there exists an $r$-enumerator for $\text{RANK}_F$, then $\text{RANK}_F$ is computable in $\text{AC}^0$ with oracle calls to the enumerator. Here $F$ is any commutative ring with identity whose size grows polynomially with the size of the input matrix.*

**Proof:** Given an enumerator and a set of matrices, the equivalence graph is uniquely defined. Recall that the number of equivalence classes (i.e. the number of connected components in the graph) is at most $\kappa - 1$. Consider guessing the number of equivalence classes, a representative matrix from each equivalence class, and, finally, the ranks of the representatives chosen. Since there are at most $(\kappa - 1)n()\kappa - 1r^{\kappa-1}$ possibilities total, which is polynomial in $n$ when $r$ is constant, we have no problem checking them all in parallel; thus we will concentrate on a single guess.

Once we have guessed the number of equivalence classes and their representatives, we can check whether every node is reachable from at least one

representative, and whether the representatives are not reachable from each other. Recall that we only need to check all paths of length at most $2\kappa - 3$ from every representative node. If at least one of these conditions is not satisfied, we reject; otherwise we proceed to checking the consistency of ranks, as we do next.

Let $R_1, \ldots, R_k \in \mathcal{A}$ be the representative matrices, and $\tau_1, \ldots, \tau_k$ be the corresponding ranks, where $1 \leq k < \kappa$ is the number of equivalence classes. Note that $\tau_1, \ldots, \tau_k$ uniquely define the ranks of all matrices in $\mathcal{A}$. Of course, we do not know $\tau_1, \ldots, \tau_k$. Instead, we will use the block diagonal construction in Lemma 2.1 to pack $R_1, \ldots, R_k$ into a single matrix, which we can then feed to the enumerator to get a list of $r$ sequences of ranks, one of which is $\langle \tau_1, \ldots, \tau_k \rangle$. Denote the sequences by $\langle \tau_1^1, \ldots, \tau_k^1 \rangle, \ldots, \langle \tau_1^r, \ldots, \tau_k^r \rangle$. Each $\langle \tau_1^i, \ldots, \tau_k^i \rangle$ uniquely defines the rank sequence $\langle v_1^i, \ldots, v_n^i \rangle$ claimed to be $\langle \mathrm{rank}(A_1), \ldots, \mathrm{rank}(A_n) \rangle$. Let $U_i = \{1 \leq j \leq n \mid v_j^i = v_{j-1}^i + 1\}$, where we define $v_0^i = 0$ for all $1 \leq i \leq r$; thus each of the $U_1, \cdots, U_r$ claims to be a maximal set of linearly independent rows of $A$. Our goal is to test whether each $U_i$ is indeed maximal, i.e. whether every remaining row of $A$ is a linear combination of the rows in $U_i$. As there are only constantly many $U_i$'s, testing them in parallel causes no problem. The rank of $A$ is given by the size of the smallest $U_i$ that passes the maximality test. (This number can be found as the corresponding $v_n^i$.)

**Remark 1** Alternatively, we could have obtained the (alleged) maximally independent sets of columns $V_1, \ldots, V_r$ (using the same procedure as for the rows). The square submatrices indexed by $U_1 \times V_1, \cdots, U_r \times V_r$ all claim to be non-singular. (We can obviously discard all candidate sequences with $|U_i| \neq |V_i|$.) Now instead of verifying the maximality claim, we can test, in parallel, which submatrices are indeed non-singular, and then take the maximum over all that pass the test.

Notice that the discussion above is valid for arbitrary matrices. Now we show how to test the maximality of $U_i$'s for matrices with entries from any commutative ring whose size does not grow more than polynomially with the size of the input matrix.

**Testing maximality:.**    Given row vectors $v_1, \ldots, v_q, w \in F^n$, verify that $w$ is in subspace generated by $v_1, \ldots, v_q$.

Let $F = \{a_1, \cdots, a_m\}$ be the ground field. If $v_1, \ldots, v_q$ are linearly independent, $w$ is dependent on $v_1, \ldots, v_q$, and $w \neq 0^n$ (where $0$ is the null element of $F$), then there must exist unique coefficients $c_1, \ldots, c_q \in F$ such that $c_1 v_1 + \cdots + c_q v_q + w = 0$. For each $i$ and $j$, $1 \leq i \leq m$, $1 \leq j \leq q$, define the

matrix

$$M_i^j = \begin{pmatrix} v_1 \\ \vdots \\ a_i v_j + w \\ \vdots \\ v_q \end{pmatrix}.$$

If the above conditions hold, then for each $j$, there is a unique $i$ such that $\mathrm{rank}(M_i^j) = q - 1$; namely, $\mathrm{rank}(M_i^j) = q - 1$ iff $a_i = c_j$; otherwise $\mathrm{rank}(M_i^j) = q$. We have $qm < nm$ matrices[1], which is polynomial in $n$, provided that $m$ grows at most polynomially in $n$. We want to reduce the number of possibilities for the ranks of these matrices to a constant, and we already know how to force the enumerator to do this for us: Recall that there is a constant $\kappa = \kappa(r)$ such that combining any $\kappa$ matrices into a single query witnesses at least one equivalence relation between a pair of claimed lists of ranks. Furthermore, there are at most $\kappa - 1$ equivalence classes, and thus only polynomially (in $nm$) many choices of how to partition $M_i^j$s into the equivalence classes. All choices can be verified in parallel, each one gives only a constant number of possible values for the ranks of $M_i^j$s. For each candidate sequence of ranks we collect the coefficients $c_i$s, assuming that this sequence is correct (i.e. for each $j$ there is a unique $i$ such that $\mathrm{rank}(M_i^j) = q - 1$, and $\mathrm{rank}(M_l^j) = q$ for all $l \neq i$); then we verify (column-wise in parallel) that the equality $c_1 v_1 + \cdots + c_q v_q + w = 0$ holds. Notice that the maximality test can be run in parallel not only for all of $U_1, \cdots, U_r$, but also for *all* rows $w$ claimed to be linear combinations of the rows in $U_h$ (for each $h$, $1 \leq h \leq r$). The number of matrices that we are dealing with for each $h$ is less than $n^2 m$, so if $m$ is polynomial in $n$, testing them in parallel causes no problem. It is easy to see that the entire computation can be done in $\mathrm{AC}^0$ with oracle gates for the enumerator, since we are dealing with fields of polynomial size (and thus elements of logarithmic length).

Now, to extend this method to commutative rings of polynomial size, we just have to test whether, for each $d \in F$, there exist coefficients $c_1, \ldots, c_q \in F$ satisfying $c_1 v_1 + \cdots + c_q v_q + dw = 0$, which can be done in parallel for each possible $d$. For some $i$ there can be more than one value of $j$ for which $\mathrm{rank}(M_i^j) = q - 1$, but since the underlying computation is $\mathrm{AC}^0$ we have only to select the smallest such $j$. ∎

**Corollary 2.1** Let $F$ be any commutative ring with identity whose size grows polynomially with the size of the input matrix. If there exists a $O(1)$-enumerable for $\mathrm{RANK}_F$ that runs in logspace, then $\mathrm{RANK}_F$ is computable in logspace.

---

[1] If $q = n$ (i.e. if the enumerator has claimed that $A$ has full rank), we can verify whether each row is a linear combination of the remaining rows. Since there are just $n$ rows, the verification can be done in parallel. The sequence passes the test if and only if it passes all of the $n$ tests. Thus we may assume that $q < n$.

# Computing the number of dependent rows

Consider the following related problem: Given a set of row vectors $v_1, \ldots, v_n \in F^n$, determine how many of them are *dependent*, i.e., are involved in some non-trivial linear dependency with other vectors in the set. Define the problem DEP$_F$: Given a matrix $A \in F^{n \times n}$, compute the number of dependent rows of $A$, written as dep$(A)$.

**Proposition 2.2** If, for some integer $r$, there exists an $r$-enumerator for DEP$_F$, then DEP$_F$ is computable in TC$^0$ with oracle calls to the enumerator. Here $F$ is any commutative ring of polynomial size.

**Proposition 2.3** For an arbitrary ring $F$, DEP$_F$ is $\leq_{O(n)\text{-tt}}^{\text{TC}^0}$-equivalent to RANK$_F$.

The proofs are not included due to space limitations; they can be found in (Beygelzimer and Ogihara, 2002).

## 3.    Enumerability of the Determinant

**Theorem 2** If, for some $k$, DET is $n^k$-enumerable in logspace, then DET $\in$ FL.

**Proof:**    Let $G$ be the configuration graph of a nondeterministic logspace machine on some input $x$. Let #path$_G(s, t)$ denote the number of directed paths from node $s$ to node $t$ in $G$. Define $f(G)$ as the function, whose value (written in binary) consists of a sequence of $n^2$ blocks of length $s = 2n\lceil \log n \rceil$, the $(n(i-1)+j)$th segment corresponding to #path$_G(i, j)$, where $1 \leq i, j \leq n$; thus

$$f(G) = \sum_{i,j=1}^{n} 2^{(n(i-1)+j)s} \#\text{path}_G(i, j).$$

It is easy to see that $f$ is a #L function; this can be done by exhibiting an NL machine $N$ whose number of accepting computation paths on $G$ is $f(G)$. The machine $N$, on input $G$, nondeterministically guesses a number $p = n(i-1)+j$, $1 \leq p \leq n^2$, after which it guesses $q$, $1 \leq q \leq 2^{ps}$, followed by a guess of path from node $i$ to node $j$ in $G$. If the guess is correct, $N$ accepts; otherwise, it rejects. It is easy to see that $N$ is a nondeterministic logspace machine that has the required number of accepting paths. Hence $f \in$ GapL, and a function is in GapL if and only if it is logspace many-one reducible to the determinant. Thus there must exist a logspace function $g$ such that for all $G$, $f(G) = \text{DET}(g(G))$. We shall use $g$ to transform $G$ into a matrix $M = g(G)$, and then run the $n^k$-enumerator on $M$ to obtain a list of $n^k$ values, one of which is the determinant of $M$. Using the equality in the above reduction, we convert this list to a list of $n^k$ candidates for $f(G)$ (each of which, if correct, certifies that the corresponding claimed value for det$(M)$ is correct). Since we have only $n^k$ candidates for $f(G)$, they can be checked in parallel. Given a purported $f(G)$, we can uniquely read off #path$_G(i, j)$ for each pair $(i, j)$, $1 \leq i, j \leq n$. These values can then be locally checked using the self-reducibility of #path$_G$.    ∎

A natural question is whether the same theorem holds for finite fields. We can only show a similar result for the determinant of integer matrices modulo some integer $p$. The problem of computing the determinant mod $p$ is $\leq_m^{\log}$-complete for $\text{Mod}_p\text{L}$, defined in (Buntrock et al., 1992). $\text{Mod}_p\text{L}$ is the class of sets $A$ for which there exists $f \in \#\text{L}$ such that for all $x$, $x \in A$ iff $f(x) \not\equiv 0 \bmod p$. We will also use the notion of membership comparability, due to Ogihara (Ogihara, 1995).

**Definition 2** A set $S$ is $g(n)$-*membership comparable*, written as $S \in \text{L-mc}(g(n))$, if there exists a function $f \in \text{FL}$ such that for any set of $g(n)$ inputs $x_1, \dots, x_{g(n)}$, each of length at most $n$, $f$ excludes one of $2^{g(n)}$ candidates for the characteristic sequence $\chi_S(x_1, \dots, x_{g(n)})$.

We will also use the predicate version of the problem, namely $\text{DET-mod-}p = \{(A, i) \mid \det(A) \equiv i \ (\text{modulo } p)\}$.

**Theorem 3** Let $p$ be any prime. If there exists a logspace computable $(p-1)$-enumerator for $\text{DET-mod-}p$, then $\text{DET-mod-}p \in \text{FL}$.

**Proof:** Omitted due to space limitations; see (Beygelzimer and Ogihara, 2002). ∎

## 4. Open questions

A natural question is whether the rank being, say, $O(\log n)$-enumerable implies that the rank is in logspace. As we mentioned, it does not seem possible to combine more than a constant number of queries into a single query to the enumerator. Another improvement would be to show that Proposition 2.2 holds for $\text{AC}^0$ in place of $\text{TC}^0$, the question being whether counting the number of dependent basis vectors can be avoided in this context.

## References

Allender, E. (1997). A clarification concerning the #L hierarchy. Available at http://www.cs.rutgers.edu/~allender/.

Allender, E., Beals, R., and Ogihara, M. (1999). The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8:99–126.

Allender, E. and Ogihara, M. (1996). Relationships among PL, #L, and the determinant. *Theoretical Informatics and Applications*, 30(1):1–21.

Amir, A., Beigel, R., and Gasarch, W. (1990). Some connections between bounded query classes and nonuniform complexity. In *5th Structure in Complexity Theory Conference*, pages 232–243.

Barrington, D. and Immerman, N. (1997). Time, hardware, and uniformity. In Hemaspaandra, L. and Selman, A., editors, *Complexity Theory Restrospective II*, pages 1–22. Springer-Verlag.

Beigel, R. and Fu, B. (1997). Circuits over PP and PL. In *12st IEEE Conference on Computational Complexity*, pages 24–35.

Beygelzimer, A. and Ogihara, M. (2002). On the enumerability of the determinant and the rank. Electronic Colloquium on Computational Complexity TR02-016.

Buntrock, G., Damm, C., Hertrampf, U., and Meinel, C. (1992). Structure and importance of Logspace-MOD class. *Mathematical Systems Theory*, 25(3):223–237.

Cai, J. and Hemachandra, L. (1989). Enumerative counting is hard. *Information and Computation*, 82(1):34–44.

Cai, J. and Hemachandra, L. (1991). A note on enumerative counting. *Information Processing Letters*, 38:215–219.

Cook, S. (1985). A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22.

Damm, C. (1991). DET = $L^{\#L}$? Informatik-Preprint 8, Fachbereich Informatik der Humboldt-Universität zu Berlin.

de Bruijn, N. G. (1970). *Asymptotic methods in analysis*. North-Holland, Amsterdam.

Lovász, L. (1993). *Combinatorial problems and exercises*. North-Holland, 2nd edition.

Ogihara, M. (1995). Polynomial-time membership comparable sets. *SIAM Journal on Computing*, 24(5):1168–1181.

Ogihara, M. (1998). The PL hierarchy collapses. *SIAM Journal on Computing*, 27:1430–1437.

Ogihara, M. and Tantau, T. (2001). On the reducibility of sets inside NP to sets with low information content. Preprint.

Ruzzo, W. (1981). On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383.

Ruzzo, W., Simon, J., and Tompa, M. (1984). Space-bounded hierarchies and probabilistic computations. *Journal of Computer and System Sciences*, 28:216–230.

Santha, M. and Tan, S. (1998). Verifying the determinant in parallel. *Computational Complexity*, 7:128–151.

Stockmeyer, L. (1985). On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861.

Toda, S. (1989). On the computational power of PP and ⊕P. In *30th IEEE Symposium on Foundations of Computer Science*, pages 514–519.

Toda, S. (1991). Counting problems computationally equivalent to computing the determinant. Technical Report CSIM 91-07, Department of Computer Science, University of Electro-Communications, Tokyo, Japan.

Valiant, L. (1979). The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201.

Valiant, L. (1992). Why is boolean complexity theory difficult. In Paterson, M., editor, *Boolean Function Complexity*, London Mathematical Society Lecture Notes Series 169, pages 84–94. Cambridge University Press.

Vinay, V. (1991). Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *6th IEEE Structure in Complexity Theory Conference*, pages 270–284.

von zur Gathen, J. (1993). Parallel linear algebra. In Reif, J., editor, *Synthesis of Parallel Algorithms*, pages 574–615. Morgan Kaufmann.

Wilson, C. (1985). Relatizived circuit complexity. *Journal of Computer and System Sciences*, 31:169–181.