A Deterministic PTAS for the Algebraic Rank of Bounded Degree Polynomials

Vishwas Bhargava[1], Markus Bläser[2], Gorav Jindal [*3], and Anurag Pandey[4]

[1]Rutgers University, vishwas1384@gmail.com
[2]Department of Computer Science, Saarland University, mblaeser@cs.uni-saarland.de
[3]Department of Computer Science, Aalto University, gorav.jindal@gmail.com
[4]Max-Planck-Institut für Informatik, apandey@mpi-inf.mpg.de

Abstract

We present a deterministic polynomial time approximation scheme (PTAS) for computing the algebraic rank of a set of bounded degree polynomials. The notion of algebraic rank naturally generalizes the notion of rank in linear algebra, i.e., instead of considering only the linear dependencies, we also consider higher degree *algebraic dependencies* among the input polynomials.

More specifically, we give an algorithm that takes as input a set $\mathbf{f} := \{f_1, \ldots, f_n\} \subset \mathbb{F}[x_1, \ldots, x_m]$ of polynomials with degrees bounded by d, and a rational number $\epsilon > 0$ and runs in time $O((\frac{nmd}{\epsilon})^{O(\frac{d^2}{\epsilon})} \cdot M(n))$, where $M(n)$ is the time required to compute the rank of an $n \times n$ matrix (with field entries), and finally outputs a number r, such that r is at least $(1 - \epsilon)$ times the algebraic rank of $\mathbf{f}$.

Our key contribution is a new technique which allows us to achieve the higher degree generalization of the results by Bläser, Jindal, Pandey (CCC'17) who gave a deterministic PTAS for computing the rank of a matrix with homogeneous linear entries. It is known that a deterministic algorithm for exactly computing the rank in the linear case is already equivalent to the celebrated Polynomial Identity Testing (PIT) problem which itself would imply circuit complexity lower bounds (Kabanets, Impagliazzo, STOC'03).

Such a higher degree generalization is already known to a much stronger extent in the non-commutative world, where the more general case in which the entries of the matrix are given by polysized formulas reduces to the case where the entries are given by linear polynomials using Higman's trick, and in the latter case, one can also compute the exact rank in polynomial time (Garg, Gurvits, Oliviera, Wigderson, FOCS'16, Ivanyos, Qiao, Subrahmanyam, ITCS'17). Higman's trick only preserves the co-rank, hence it cannot be used to reduce the problem of rank approximation to the case when the matrix entries are linear polynomials. Thus our work can also be seen as a step towards bridging the knowledge gap between the non-commutative world and the commutative world.

1 Introduction

This article is a result of an exploration of three related fundamental themes in algebra from a computational perspective - *polynomial identities*, *algebraic dependence of polynomials* and *rank of symbolic matrices*. These are already known to be crucial in several aspects of the theory of computation. In this introduction, we give a brief overview of these three themes, the corresponding naturally arising computational problems, the interrelations among them and their applications to theoretical computer science.

1.1 Polynomial identity testing. Polynomial identities are useful and ubiquitous in mathematics and computer science. Simple examples include the *two square identity*: $(a^2 + b^2)(x^2 + y^2) = (ax - by)^2 + (bx + ay)^2$, which can be used to show that the distance is invariant under a rotation of axes. Similarly the identity $\sum_{1 \le i < j \le 4} (x_i + x_j)^4 + (x_i - x_j)^2 = 6(x_1^2 + x_2^2 + x_3^2 + x_4^2)^2$ was used by Liouville to show that every positive integer is a sum of at most 53 fourth powers of integers. More recently, even an identity as simple as $2xy = (x + y)^2 - x^2 - y^2$ was crucial in demonstrating the power of an approximative model of computation in

[*]Supported by European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No 759557) and by Academy of Finland, under grant number 310415. This work was done while the author was a graduate student at Saarland University and the Max-Planck-Institut für Informatik.

algebraic complexity theory [BIZ18, Kum18]. Having an awareness of the pervasiveness of polynomial identities in mathematics, perhaps one would not be surprised to discover the extent of applications of the computational problem of testing if a given compact representation of a polynomial (arithmetic circuit) is indeed computing the identically zero polynomial. The celebrated *Polynomial Identity Testing* (PIT) question asks that given an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_m]$, test if f is the zero polynomial. The PIT problem captures several problems in algebra and combinatorics. For example, its special case captures the *perfect matching* problem via the *Tutte Matrix* [Tut47, Lov79]. The breakthrough *primality testing* algorithm by Agrawal, Kayal and Saxena [AKS04] also reduced the primality testing problem to a special case of the PIT problem. PIT was also crucial in Shamir's proof of $\mathsf{IP} = \mathsf{PSPACE}$ [Sha92]. Kabanets and Impagliazzo showed that a deterministic polynomial time algorithm would imply circuit complexity lower bounds, i.e., either $\mathsf{NEXP} \not\subset \mathsf{P/Poly}$ or the permanent does not have polynomial sized arithmetic circuits [KI04].

1.2 Algebraic dependence of polynomials. Algebraic dependence is a fundamental concept in algebra that captures polynomial relationships among polynomials. Polynomials $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ are called *algebraically dependent* over a field $\mathbb{F}$ if there exists a non-zero polynomial $A(y_1, \ldots, y_m) \in \mathbb{F}[y_1, \ldots, y_m]$ such that $A(f_1, \ldots, f_m) = 0$ and such an A is called an *annihilating polynomial* of $f_1, \ldots, f_m$. If no such nonzero polynomial A exists, then the given polynomials are called *algebraically independent* over $\mathbb{F}$.

For example, $f_1 = (x + y)^2$ and $f_2 = (x + y)^3$ are algebraically dependent over any field, as $y_1^3 - y_2^2$ is an annihilating polynomial. Polynomials $x + y$ and $x^p + y^p$ are dependent over $\mathbb{F}_p$, but independent over $\mathbb{Q}$. The monomials $x_1, x_2, \ldots, x_n$ are an example of algebraically independent polynomials over any field.

Algebraic dependence can be viewed as a generalization of linear dependence as the former captures algebraic relationships of any degree, whereas the latter captures linear relationships. Algebraic dependence shares a few combinatorial properties (known as matroid properties, see [Oxl06]) with linear dependence. For example, if a set of polynomials is algebraically independent then any subset of them is algebraically independent.

The *algebraic rank*, also known as *transcendence degree*, of a set of polynomials is defined as the maximal number of algebraically independent polynomials and it is well defined thanks to the matroid properties. The concepts of rank and basis in linear algebra have analogs here as algebraic rank and *transcendence basis*, respectively.

Algebraic rank is a generalization of several natural problems in algebra and combinatorics, for example, computing the size of the *maximum matching* in general graphs is also a special case of the problem. More generally, it is a generalization of the celebrated *Polynomial Identity Testing* (PIT) problem, which itself generalizes several problems including the *Primality Testing* problem. The notion of algebraic rank has been used to make progress on PIT by helping in the hitting set construction for $\overline{\mathsf{VP}}$ [GSS18] and by being crucial in solving several special cases of the PIT problem [ASSS12, KS16], and very recently has also been used in the famous *Integer Factorization* problem [ASS16]. It has also been used to construct explicit extractors, condensers and dispersers for polynomial sources [DGW09], crucial in the area of pseudorandom generators. It was also important in proving the best known general formula lower bounds for determinant [Kal85], and more recently for proving strong lower bounds for restricted class of arithmetic circuits [ASSS12, KS16, PSS18].

1.3 Rank of symbolic matrices. Symbolic matrices, i.e., matrices with polynomial entries are another ubiquitous objects in mathematics. Edmonds [Edm67] was the first one who explicitly stated the problem of computing rank of the symbolic matrix when the entries are linear forms. Some applications of symbolic rank computation in computer science include the algebraic algorithm for *maximum matching* problem for bipartite and general graphs [Lov79, MVV87, FGT16]. Even the *linear matroid parity problem* and the *linear matroid intersection problem* are special cases of the commutative rank problem of symbolic matrices with linear forms as entries (see [SV07, GS86, Orl08, Har09, GT17]). Owing to the works of Valiant and Mahajan-Vinay, we know that the rank computation of symbolic matrices (the decision version) with linear entries is equivalent to *Polynomial Identity Testing* (PIT) for Algebraic Branching Programs (ABP) [Val79, MV97] which became a central problem in complexity theory after the results of Kabanets and Impagliazzo showing that a deterministic algorithm for PIT would imply circuit complexity lower bounds. When the entries are allowed to be higher degree polynomials, too, the symbolic matrix rank computation also captures the computation of the rank of the Jacobian matrix, which in turn captures the algebraic rank problem over fields of zero characteristic via the classical Jacobian criterion. It also captures the rank of the Hessian matrix which like the Jacobian matrix is pervasive in mathematics and physics. For example, using the Katz's dimension formula, the rank of the Hessian matrix corresponds to the dimension of the dual va-

rieties of hypersurfaces, which is useful in studying the determinantal complexity in the Geometric Complexity Theory (see e.g. [HJ12, LMR10]).

1.4 A tale of three computational problems.

In this section, we give an account of the three main computational problems relevant to the paper, each inspired by one of themes discussed in the above subsections. In order to discuss them more precisely, we have to specify the representation of the input polynomials. An *arithmetic circuit* is a directed acyclic graph consisting of addition ($+$) and multiplication ($\times$) gates as nodes, takes variables $x_1, \ldots, x_n$ and field constants as input (leaves), and outputs a polynomial $f(x_1, \ldots, x_n)$. This is a succinct representation of multivariate polynomials, as polynomials of high degree (or having many monomials) can be represented by small circuits.

PROBLEM 1.1. (**PIT**) *Given an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_m]$, test if f is identically zero.*

PROBLEM 1.2. (**AlgRank**) *Given arithmetic circuits $C_1, \ldots, C_n$ computing polynomials $f_1, \ldots, f_n \in \mathbb{F}[x_1, \ldots, x_m]$, compute* $\text{algRank}(\{f_1, \ldots, f_n\})$.

PROBLEM 1.3. (**RANK**) *Given an $n \times n$ matrix $Q(x_1, x_2, \ldots, x_m) = (q_{ij})_{n \times n}$ whose entries are given by arithmetic circuits C_{ij} computing polynomials $q_{ij} \in \mathbb{F}[x_1, \ldots, x_m]$, compute the rank of Q over $\mathbb{F}(x_1, \ldots, x_m)$.*

1.4.1 Connections among the three problems.

It is clear that **PIT** reduces to the decision version of **RANK**. In fact, it is known that the case of **RANK**, where the entries are just *linear forms*, is strong enough to capture **PIT** for *algebraic branching programs* (ABP) [Val79, MV97]. It is also easy to see that **PIT** reduces to the decision version of **AlgRank**: we can just give our input instance to the **AlgRank** problem and ask whether the algebraic rank is 0 or 1. It might happen that the circuit was computing a non-zero constant polynomial, in this case the algebraic rank will not be able to certify the non-zeroness, because the algebraic rank is still 0 in this case. However, this is an easy case anyway, because we can test these cases beforehand simply by evaluating the circuit on the point $(0, \ldots, 0)$ and checking if the circuit evaluates to 0. It might be the case that the value at $(0, \ldots, 0)$ is too large to compute, since in the most general setting, the formal degree of the circuit can be exponential. In this case, we can alternatively check whether $x_1 \cdot f$ has algebraic rank 0 or 1, where f is the polynomial computed by the circuit.

Over fields of characteristic zero, the problem **AlgRank** reduces to the problem **RANK** via the classical Jacobian criterion:

DEFINITION 1.1. (JACOBIAN) *The* Jacobian *of polynomials $\mathbf{f} = \{f_1, \cdots, f_n\} \subset \mathbb{F}[x_1, \cdots, x_m]$ is the matrix $\mathcal{J}_{\mathbf{x}}(\mathbf{f}) = (\partial_{x_j} f_i)_{m \times n}$, where $\partial_{x_j} f_i := \partial f_i / \partial x_j$.*

We state the classical Jacobian criterion [Jac41, PSS18].

LEMMA 1.1. (JACOBIAN CRITERION) *Let $\mathbf{f} \subset \mathbb{F}[\mathbf{x}]$ be a finite set of polynomials of degree at most d, and $\text{algRank}_{\mathbb{F}} \mathbf{f} \leq r$. If $\text{char}(\mathbb{F}) = 0$, or $\text{char}(\mathbb{F}) > d^r$, then $\text{algRank}_{\mathbb{F}} \mathbf{f} = \text{rank}_{\mathbb{F}(\mathbf{x})} \mathcal{J}_{\mathbf{x}}(\mathbf{f})$.*

Thus, in order to solve **AlgRank** for a set of polynomials $\mathbf{f}$, it suffices to solve **RANK** for the matrix where the entries are the first order partial derivatives of the elements in $\mathbf{f}$. Now, we know that for a given arithmetic circuit C of size s computing a polynomial f, there exists an arithmetic circuit C' of size $O(s)$ computing all the first order partial derivatives of f ([BS83], see also [SY10, Section 2.3]). Thus having a deterministic poly-time algorithm for computing the rank of a matrix with entries given by arithmetic circuits, we can solve the **AlgRank** problem in deterministic polynomial time when the input is given as the arithmetic circuits of the set of polynomials whose algebraic rank we want to compute.

Similarly, if the input to our **AlgRank** problem is a set of polynomials with bounded degrees (say, with an upper bound of d), the Jacobian matrix will have entries which are polynomials with degrees bounded by $d-1$. So, in order to solve the bounded degree version of the **AlgRank** problem, it suffices to solve the bounded degree version of the **RANK** problem.

Thus, over fields of characteristic 0, it suffices to solve **RANK** in order to solve **PIT** and **AlgRank**. In order to give a PTAS for the **AlgRank** problem for bounded degree polynomials, we give a PTAS for the **RANK** problem where entries of the matrix are bounded degree polynomials. We remind the reader that the decision version of the **RANK** problem in the bounded degree case still gives an unbounded degree PIT instance (simply recall that it is already true when the entries were linear forms). In fact, for the **RANK** and the **AlgRank** problem, we need such restrictions, else we will have to solve the general PIT problem beforehand.

1.4.2 The current status of the three problems.

All of the three problems can be solved in randomized polynomial time thanks to the Schwarz-Zippel lemma [Sch80, Zip79, DL78]. For **RANK**, we just need

to evaluate our polynomials at a random point to obtain a matrix of field elements, and the lemma guarantees that with high probability the rank of the obtained matrix over the base field $\mathbb{F}$ would be the same as the rank of the original matrix (over the function field $\mathbb{F}(x_1, \ldots, x_m)$). And as we already pointed out in Section 1.4.1, **RANK** is the most general of the three problems, all the three problems can be solved in randomized polynomial time. However, a deterministic algorithm has remained elusive for all of the three problems. For the simplest problem among the three, i.e., the **PIT** problem, we know deterministic polynomial time algorithms only in special cases. For example, when the input is given in the sparse representation, a deterministic polynomial time algorithm is known [KS01]. Similarly, if the input is a diagonal depth 3 circuit, we know a deterministic polynomial time algorithm [Sax08]. There has been a plethora of works derandomizing special cases in polynomial or quasipolynomial time. We refer the reader to the excellent surveys on the problem by Saxena [Sax09, Sax14] and Shpilka and Yehudayoff [SY10] for a detailed account of the progress and techniques involved in derandomizing the PIT problem. Derandomizing the **RANK** problem in its simplest case, i.e., when the entries are just *linear forms* has already proven to be very challenging. It is equivalent to solving PIT problem for Algebraic Branching Programs (ABPs). Only for very restricted classes of ABPs (the so called ROABP model and its variants), we know how to derandomize PIT ([RS05a, FS13, FSS14, GKS16]). Recently, [BJP18] gave a derandomization for approximately computing the rank, i.e., they gave a deterministic PTAS for the **RANK** problem, when the entries are linear forms.

1.4.3 RANK in the non-commutative world.

We point out that in the non-commutative world, several computational problems are better understood as compared to the commutative world. For example, PIT for non-commutative formulas is known to be in polynomial time [RS05b]. Moreover, exponential lower bounds are known against non-commutative formulas and algebraic branching programs [Nis91]. The same is true for the **RANK** problem in the non-commutative world. Here, Garg, Gurvits, Oliviera, and Wigderson [GGOW16] and Ivanyos, Qiao, and Subrahmanyam [IQS17], gave a deterministic polynomial time algorithm for the **RANK** problem when the entries of the matrices are linear forms. In fact, they also solved the RANK problem when the entries are given by formulas, because in the non-commutative world, the case in which the entries are given by formulas reduces to the case in which the entries are given by linear forms using Higman's trick ([Hig40], see [GGOW15, Appendix A.1]). One would be tempted to use the same trick for the commutative rank and then use the deterministic PTAS for linear forms case given by [BJP18] to have a deterministic PTAS for the case in which the entries are given by formulas. Unfortunately, this trick does only preserves the co-rank. Hence, it is not useful for computing an approximation of the rank in the general **RANK** problem, since it enlarges the size of the matrix. Another interesting fact is that in the case when the entries are linear forms, we know that the non-commutative rank (see [GGOW15] for a definition) is at most twice the commutative rank [FR04]. Thus, an algorithm for the non-commutative rank gives a 1/2-approximation for the commutative rank when the entries are linear forms. Here, one would be tempted to claim that even when the entries are given by formulas, we get a 1/2-approximation for the commutative rank using the known exact algorithms for the non-commutative rank. This also does not work unfortunately. The following very simple example denies any such possibilities when entries compute higher degree polynomials.

Let $f = xy - yx$. Consider, the following 1×1 matrix Q,

$$Q = \begin{bmatrix} f \end{bmatrix}.$$

Notice that the non-commutative rank of Q is 1, but the commutative rank is 0. This gap can be made arbitrarily large by simply taking a diagonal matrix with all the diagonal entries being $xy - yx$. Thus, in general, we cannot approximate the commutative rank with non-commutative rank.

Thus, there is a huge knowledge gap that we are observing between the commutative world and the non-commutative world with respect to the **RANK** problem. On the one hand, we have polynomial time algorithms for exact rank computation in the non-commutative world even when the entries are given by formulas, whereas in the commutative case, all we have is a deterministic PTAS, that only works in the case when the entries of the matrix are *linear forms*. No deterministic PTAS was known even when the entries of the matrix are given by *quadratic forms*. In this work, we solve precisely a more general version of this, i.e., we give a deterministic PTAS in the case when the entries are given by polynomials whose degrees are bounded by an arbitrary constant, hence taking another step towards bridging this knowledge gap between the two worlds.

2 Our Results

In this paper, we give the first deterministic polynomial time approximation scheme (PTAS) for the **RANK** problem under the restriction that the entries of the matrix are bounded degree polynomials. We give a new technique which allows us to achieve generalizations to higher degrees of the results of [BJP18], who gave a PTAS for the **RANK** problem when the entries were linear forms.

We need to formalize the setup of the problem and fix some notations to formally state our main result.

Consider a matrix $Q(x_1,\ x_2,\ \ldots,\ x_m) = (q_{ij})_{n\times n}$ of size $n \times n$, the entries q_{ij} of which are polynomials of degrees bounded by some constant d in the variables $\mathbf{x} = (x_1,\ x_2,\ \ldots,\ x_m)$. We want to compute the rank of Q over the rational function field $\mathbb{F}(x_1, x_2, \ldots, x_m)$. In fact, it suffices to consider the case when the entries are homogeneous forms of degree d (see Appendix A).

To this end, we define the following problem.

Problem 2.1. *Given a matrix $Q(x_1,\ x_2,\ \ldots,\ x_m) = (q_{ij})_{n\times n}$ of size $n \times n$, the entries q_{ij} of which are homogeneous forms of constant degree d, compute the rank of Q over $\mathbb{F}(x_1, x_2, \ldots, x_m)$.*

Since the degrees of the polynomials in the entries are bounded by a constant d, we can assume that they are given explicitly as the list of coefficients.

As stated above in Section 1.4.2, this problem has a very simple randomized algorithm. But we want deterministic algorithms to compute the rank of Q. We know that finding deterministic algorithms for Problem 2.1 is hard. Thus in this paper, we consider whether one can approximate $\operatorname{rank}(Q)$ deterministically. Following is the main contribution of this paper.

Theorem 2.1. (PTAS for RANK) *Given Q as in Problem 2.1 over a field $\mathbb{F}$ with $|\mathbb{F}| > nd$ and a constant $0 < \epsilon < 1$, there exists a deterministic algorithm which computes an assignment $(\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_m) \in \mathbb{F}^m$ such that*

$$\operatorname{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m)) \geq (1-\epsilon)\operatorname{rank}(Q(x_1, x_2, \ldots, x_m)).$$

This algorithm runs in time $O\left((nmd)^{O\left(\frac{d^2}{\epsilon}\right)} \cdot M(n)\right)$, where $M(n)$ is the time required to compute the rank of an $n \times n$ matrix over $\mathbb{F}$.

Clearly, the above running time is polynomial when d is a constant.

Now we see that it suffices to solve Problem 2.1 for ϵ being a constant, as this implies the general case via tensoring. That is, for an $n\times n$ matrix Q of Problem 2.1, we can tensor Q with itself to get an $n^2\times n^2$ size matrix, where $\operatorname{rank}(Q)^2 = \operatorname{rank}(Q \otimes Q)$. Also, if Q has degree d entries, then $Q \otimes Q$ has degree $2d$ entries. Thus, if we tensor k times we get a matrix of size $n^k \times n^k$ with entries of degree dk and rank $(\operatorname{rank}(Q))^k$. If we get a $(1-\epsilon)$ approximation to this rank, then taking the k^{th} root of this value is a $(1-\epsilon)^{\frac{1}{k}}$-approximation to the original rank. As this is approximately $(1-\frac{\epsilon}{k})$, it follows that we can get a pretty good approximation this way. By this method, we get a trade-off between the degree and the approximation parameter. That is, if an algorithm finds a $1-\epsilon$ approximation of the rank when the entries are degree dk polynomials, then this algorithm can be used to find a $(1-\frac{\epsilon}{k})$ approximation of the rank when the entries are degree d polynomials. This reduction also shows that at least a linear dependence on d in the exponent is essentially required for this problem, even for $\epsilon = O(1)$, as otherwise via tensoring we can solve PIT non-trivially fast. We remark that our algorithm directly tackles the problem of $1-\epsilon$ rank approximation without using this tensoring idea.

Since we have already established in the previous section that **AlgRank** reduces to **RANK** using the Jacobian criterion, it is obvious that the deterministic PTAS for **AlgRank** under the restriction that the input polynomials are of bounded degree is an easy consequence of the above stated result.

Theorem 2.2. (PTAS for AlgRank) *Given a set $\boldsymbol{f} := \{f_1, \ldots, f_n\} \subset \mathbb{F}[x_1, \ldots, x_m]$ of polynomials of degrees bounded by a constant d with $\operatorname{char}(\mathbb{F}) = 0$, and a rational number $\epsilon > 0$, there is a deterministic algorithm that outputs a number r, such that $r \geq (1-\epsilon)\cdot \operatorname{algRank}(\boldsymbol{f})$. The algorithms runs in time $O\left((nmd)^{O\left(\frac{d^2}{\epsilon}\right)} \cdot M(n)\right)$, where $M(n)$ is the time required to compute the rank of an $n \times n$ matrix over $\mathbb{F}$.*

Again, the above algorithm is a polynomial time algorithm when d is a constant.

2.1 Comparison with the techniques of [BJP18].

The PTAS for the linear case in [BJP18] greedily increased the rank starting with the zero matrix, and the proof of correctness of the algorithm rested on the guarantee that when we are unable to increase the rank greedily, we are already done, i.e. the current matrix already has the desired approximation of the rank. The main component of the proof of this guarantee was a refined analysis of the so-called Wong sequence which are defined for matrix spaces and a matrix with entries as linear forms can be interpreted as a matrix space. [BJP18] introduced a novel notion of

Wong index. It was shown in [BJP18] that if the Wong index of a matrix is "high" then this matrix is a good approximation of the commutative rank. If the Wong index of a matrix is "low" then it was shown in [BJP18] that one can find a matrix of higher rank efficiently.

The limitation of the techniques in [BJP18] is that the Wong sequences are defined and studied only in the case of matrix spaces and a correspondence between the matrix spaces and matrix with higher degree (non-linear) polynomials does not exist. So, for the higher degree case, it is not clear how to define a notion of a Wong sequence and hence the techniques of [BJP18] do not generalize. Thus, one needs to find a new technique to deal with the higher degree case. This is precisely what we do in this paper. We find a new way to analyze the low degree components of the minors of the matrix obtained in the greedy step (see Section 4 for the main proof ideas), which allows us to use the same algorithm strategy as in [BJP18] for higher degree forms as well. It can be shown that the Wong index of [BJP18] corresponds to the degree of the least degree monomial of a suitable minor.

2.2 Organization of paper. In the next section, we define some notations and recall some linear-algebraic tools that will be useful for us. In Section 4, we discuss our main idea and give an overview of the proof strategy. Section 5 contains the technical details of the proof. We present our commutative rank algorithm in Section 6. We conclude with some discussion and open questions in Section 7. Appendix A contains the reduction of the arbitrary case to the homogeneous case.

3 Preliminaries

In the following, we present some of the definitions and tools which are used frequently in this paper. When we speak of a matrix polynomial, we mean a matrix with polynomials as entries.

1. For an $r \times r$ matrix $A \in \mathbb{F}^{r\times r}$, we use $A_{\widehat{ij}}$ to denote the sub-matrix of A obtained by removing the i^{th} column and the j^{th} row.

2. I_r is used to denote the $r \times r$ identity matrix.

3. For a polynomial f, $\hom_k(f)$ denotes the homogeneous degree k part of f.

4. We also use the same notation $\hom_k(M)$ to denote the homogeneous degree k part of a matrix polynomial M. Note that $\hom_k(M)$ is also a matrix polynomial.

5. For a polynomial f, $\text{ord}(f)$ is used to denote the degree of the least degree monomial in f. We use the same notation $\text{ord}(M)$ for matrix polynomials M also. Notice that $\text{ord}(f)$ and $\text{ord}(M)$ are just natural numbers.

Definition 3.1. (Characteristic Polynomial) *For an $r \times r$ matrix A, its* characteristic polynomial *$p_A(t)$ is defined as:*

$$p_A(t) \stackrel{def}{=\!=} \det(tI - A) = p_0 t^r + p_1 t^{r-1} + \cdots + p_r.$$

Note that in Definition 3.1, $p_0 = 1$ is always true.

Fact 3.1. *Over all fields, for any $r \times r$ matrix A, $\det(A) = (-1)^r p_A(0) = (-1)^r p_r$.*

Definition 3.2. (Adjoint) *For an $r\times r$ matrix A, the* adjoint $\text{adj}(A)$ *is also an $r\times r$ matrix whose $(i,j)^{th}$ entry is $(-1)^{i+j}\det(A_{\widehat{ij}})$.*

Theorem 3.1. *For a square $r \times r$ matrix L, define $q_L(t) \stackrel{def}{=\!=} \frac{p_L(t)-p_L(0)}{t}$. Over all fields, we have the following equality for* $\text{adj}(L)$*:*

$$\text{adj}(L) = (-1)^{r+1} q_L(L). \tag{3.1}$$

Proof. Here we only prove this claim for $\mathbb{F} = \mathbb{C}$ but it is true for all fields. We use the following facts:

1. If L is non-singular, then $\text{adj}(L) = \det(L)L^{-1}$.

2. The Cayley-Hamilton theorem, which states that for any L, $p_L(L) = 0$.

3. The set GL_r of non-singular matrices is dense (under the Euclidean topology) in the set $\mathbb{F}^{r\times r}$ of all the matrices.

We first prove the claim when L is non-singular. Let $p_L(t) = p_0 t^r + p_1 t^{r-1} + \cdots + p_r$. By using the Cayley-Hamilton theorem, we have the following equality:

$$p_0 L^r + p_1 L^{r-1} + \cdots + p_r = 0. \tag{3.2}$$

Since L is non-singular, we multiply by L^{-1} on both the sides of Equation (3.2). Thus

$$\begin{aligned} p_0 L^{r-1} + p_1 L^{r-2} + \ldots p_{r-1} &= -p_r L^{-1} \\ &= (-1)^{r+1}\det(L)L^{-1} \\ &= (-1)^{r+1}\,\text{adj}(L). \end{aligned} \tag{3.3}$$

Note that $q_L(L) = p_0 L^{r-1} + p_1 L^{r-2} + \cdots + p_{r-1}$. Therefore Equation (3.3) implies $\text{adj}(L) = (-1)^{r+1} q_L(L)$.

Now notice that Equation (3.1) is an equation where entries of the matrices on both sides are polynomials in the entries of L. Now the claim follows using the denseness of GL_r in $\mathbb{F}^{r\times r}$. □

THEOREM 3.2. *For a square $r \times r$ matrix L with $p_L(t) = p_0 t^r + p_1 t^{r-1} + \cdots + p_r$, the following equality holds over any field:*

$$\mathrm{adj}(I+L) = \sum_{i=0}^{r-1} (-1)^i p_i \cdot \left(\sum_{j=0}^{r-i-1} (-L)^j \right).$$

Proof. First we compute the characteristic polynomial p_{I+L} of $I+L$. We have:

$$\begin{aligned} p_{I+L}(t) &= \det(tI - (I+L)) \\ &= \det((t-1)I - L)) \\ &= p_L(t-1). \end{aligned}$$

Thus we have,

$$\begin{aligned} q_{I+L}(t) &\overset{def}{=\!=} \frac{p_{I+L}(t) - p_{I+L}(0)}{t} \\ &= \frac{p_L(t-1) - p_L(-1)}{t} \\ &= \sum_{i=0}^{r} \frac{p_{r-i} \cdot ((t-1)^i - (-1)^i)}{t} \\ &= \sum_{i=1}^{r} p_{r-i} \cdot \left(\sum_{j=0}^{i-1} (-1)^j (t-1)^{i-j-1} \right). \end{aligned}$$

Therefore,

$$\begin{aligned} \mathrm{adj}(I+L) &= (-1)^{r+1} q_{I+L}(I+L) \\ &= (-1)^{r+1} \sum_{i=1}^{r} p_{r-i} \cdot \left(\sum_{j=0}^{i-1} (-1)^j (L)^{i-j-1} \right) \\ &= p_0(I - L + \cdots + (-L)^{r-1}) \\ &\quad - p_1(I - L + \ldots (-L)^{r-2}) \\ &\quad + \cdots + (-1)^{r-1} p_{r-1}(I) \\ &= \sum_{i=0}^{r-1} (-1)^i p_i \cdot \left(\sum_{j=0}^{r-i-1} (-L)^j \right). \end{aligned}$$

□

Next come some easy facts from linear algebra.

FACT 3.2. *Let $\mathbb{F}$ be any field. If A is an $n \times n$ matrix of rank r over $\mathbb{F}$, then there exist two $n \times n$ non-singular matrices $P, R \in \mathbb{F}^{n \times n}$ such that:*

$$PAR = \begin{array}{c} \overset{r \text{ columns}}{\overbrace{}} \\ r \text{ rows} \left\{ \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix} \right. \\ \underbrace{}_{n-r \text{ columns}} \end{array} \left. \begin{array}{l} \\ \\ \} n-r \text{ rows} \end{array} \right. \tag{3.4}$$

FACT 3.3. *Let $\mathbb{F}$ be any field and let M be a matrix of the following form over $\mathbb{F}$:*

$$M = \begin{array}{c} \overset{r \text{ columns}}{\overbrace{}} \\ r \text{ rows} \left\{ \begin{pmatrix} L & B \\ A & C \end{pmatrix} \right. \\ \underbrace{}_{n-r \text{ columns}} \end{array} \left. \begin{array}{l} \\ \\ \} n-r \text{ rows} \end{array} \right. \tag{3.5}$$

Also, let $\mathrm{rank}\left(\begin{bmatrix} A & C \end{bmatrix}\right) = a$ *and* $\mathrm{rank}\left(\begin{bmatrix} B \\ C \end{bmatrix}\right) = b$. *Then* $\mathrm{rank}(M) \leq r + \min(a, b)$.

LEMMA 3.1. *If $|\mathbb{F}| > nd$, then we can construct a hitting set $H_{m,d,\ell}$ of size $O((m(d+1))^\ell)$ for the set $F_{m,d,\ell}$ of polynomials defined by:*

$$F_{m,d,\ell} \overset{def}{=\!=} \{f \in \mathbb{F}[x_1, \ldots, x_m] \mid \deg(f) \leq d, \mathrm{ord}(f) \leq \ell\}.$$

Proof. Let $f \in F_{m,d,\ell}$. Since $\mathrm{ord}(f) \leq \ell$, there exists a non-zero monomial $x_{i_1} \cdot x_{i_2} \cdot \ldots x_{i_\ell}$ of f. The variables x_{i_j} need not be distinct here. We first do a brute force search for these ℓ variables by making all the other $m-\ell$ variables zero. This can be done using $\binom{m}{\ell} = O(m^\ell)$ assignments. Now we are left with a polynomial f' of degree d in at most ℓ variables. By using Schwartz-Zippel lemma [Zip79, Sch80], we can find a non-zero assignment of f' using $(d+1)^\ell$ assignments. Thus there exists a hitting set of size $O(m^\ell \cdot (d+1)^\ell) = O((m(d+1))^\ell)$. □

4 Main proof ideas

Here we explain the main idea used in devising the desired algorithm claimed in Theorem 2.1. Since Theorem 2.1 is essentially a generalization of [BJP18], a direct approach seems to be converting a matrix of degree d forms to a matrix of linear forms (using the equivalence of ABPs and determinants, see [MV97]). However, such a direct approach (although it preserves non-zeroness) gives no information about the rank of the matrix.

So, instead of directly reducing an instance of Problem 2.1 to an instance of linear forms case as in [BJP18], we follow the high level approach of [BJP18] of greedily increasing the rank of Q starting with the zero matrix. Suppose we have found $\lambda_1, \lambda_2, \ldots, \lambda_m$ such that $\mathrm{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m)) = r$. We want to find an assignment of the form $(x_1 + \lambda_1, x_2 + \lambda_2, \ldots, x_m + \lambda_m)$ such that $\mathrm{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots, x_m + \lambda_m)) > r$. This step of finding a matrix of bigger rank is called a rank increasing step. Under this transformation ($x_i \to x_i + \lambda_i$), we have the following equality:

$$\begin{aligned} &Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots, x_m + \lambda_m) \\ &\quad = Q(\lambda_1, \lambda_2, \ldots, \lambda_m) + Q_d(x_1, x_2, \ldots, x_m). \end{aligned} \tag{4.6}$$

Here $Q_d(x_1, x_2, \ldots, x_m)$ is some matrix whose entries are polynomials of degree at most d. By using Fact 3.2, we know that there exists non-singular matrices $P, R \in \mathbb{F}^{n\times n}$ such that:

$$P \cdot Q(\lambda_1, \lambda_2, \ldots, \lambda_m) \cdot R = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}.$$

Now consider the following equation:

$$\begin{aligned} P \cdot Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m) \cdot R \\ = P \cdot Q(\lambda_1, \lambda_2, \ldots, \lambda_m) \cdot R \\ + P \cdot Q_d(x_1, x_2, \ldots, x_m) \cdot R. \end{aligned} \tag{4.7}$$

Since P, R are non-singular, we know that

$$\begin{aligned} &\text{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)) \\ &\quad = \text{rank}(P \cdot Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m) \cdot R). \end{aligned}$$

Thus it is enough to find an assignment to the variables $x_1, \ldots, x_m$ such that

$$\text{rank}(P \cdot Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m) \cdot R) > r.$$

The following Lemma 4.1 is easy to verify.

LEMMA 4.1. *For any* $(\lambda_1, \lambda_2, \ldots, \lambda_m) \in \mathbb{F}^m$,

$$\begin{aligned} &\text{rank}(Q(x_1, x_2, \ldots, x_m)) \\ &\quad = \text{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)) \\ &\quad = \text{rank}(P \cdot Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m) \cdot R). \end{aligned}$$

Proof. We assume that $|\mathbb{F}| > dn$. Now suppose that $s = \max\{\text{rank}(Q(\lambda_1, \ldots, \lambda_m)) \mid (\lambda_1, \ldots, \lambda_m) \in \mathbb{F}^m\}$ and $r = \text{rank}(Q(x_1, x_2, \ldots, x_m))$. We want to show that $s = r$. We know that there exists a non-zero $r \times r$ minor M_r of $Q(x_1, x_2, \ldots, x_m)$. Notice that M_r is a polynomial of degree at most $rd \leq nd$. Thus by the Schwartz-Zippel lemma [Zip79, Sch80], there exists $(\lambda_1, \lambda_2, \ldots, \lambda_m) \in \mathbb{F}^m$ such that $M_r(\lambda_1, \lambda_2, \ldots, \lambda_m) \neq 0$. Therefore $s \geq r$. The other direction is trivial.

This also implies that

$$\begin{aligned} &\text{rank}(Q(x_1, x_2, \ldots, x_m)) \\ &\quad = \text{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)) \\ &\quad = r. \end{aligned}$$

This is because there is a bijection from $\mathbb{F}^m$ to $\mathbb{F}^m$ given by $x_i \mapsto x_i + \lambda_i$. This last equality is trivial because we only multiply by non-singular matrices P and R. □

By using Lemma 4.1, we can omit P, R in the discussion of our rank increasing step. Thus we can assume that:

$$Q(\lambda_1, \lambda_2, \ldots, \lambda_m) = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}.$$

We want to ensure that at least one of the following two scenarios happens.

1. We can "easily" find an assignment of the form $(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)$ such that $\text{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)) > r$. This is our rank increasing step. "Easily" here means in time $O((nmd)^{O(\frac{d}{\epsilon})})$ deterministically.
2. $r \geq (1 - \epsilon) \cdot \text{rank}(Q(x_1, x_2, \ldots, x_m))$, i.e., we are already done.

We decompose $Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)$ as:

$$Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m) = \begin{bmatrix} I_r + L & B \\ A & C \end{bmatrix}. \tag{4.8}$$

We write $L = L_1 + \cdots + L_d$, where L_i is a matrix whose entries are homogeneous polynomials of degree i. Similarly we decompose A, B, C into A_i, B_i, C_i. In other words, we have $L_s = \text{hom}_s(L)$, $A_s = \text{hom}_s(A)$, $B_s = \text{hom}_s(B)$, and $C_s = \text{hom}_s(C)$.

We now describe when the first of the two scenarios described above happens. When is the condition "$\text{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots, x_m + \lambda_m)) > r$" true? It happens when there exists a non-zero $(r + 1) \times (r + 1)$ minor of $Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)$. Consider a sub-matrix $M_{k,\ell}$ of $Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)$ of size $(r + 1) \times (r + 1)$ obtained by taking $I_r + L$, the k^{th} row of A, the ℓ^{th} column of B, and also the $(k, \ell)^{\text{th}}$entry of C. Thus $M_{k,\ell}$ looks like below:

$$M_{k,\ell} = \begin{pmatrix} 1 + l_{11} & l_{12} & \ldots & l_{1r} & b_1 \\ l_{12} & l_{22} & \ldots & l_{2r} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ l_{r1} & l_{r2} & \ldots & 1 + l_{rr} & b_r \\ a_1 & a_2 & \ldots & a_r & c \end{pmatrix}. \tag{4.9}$$

Here l_{ij} is the $(i, j)^{\text{th}}$ entry of L. To ensure $\text{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots, x_m + \lambda_m)) > r$, we want to find an assignment to the x_i's such that $\exists k, \ell \in [n - r]$ satisfying $\det(M_{k,\ell}) \neq 0$.

How to find an assignment to the x_i's such that $\det(M_{k,\ell}) \neq 0$? Note that $\det(M_{k,\ell})$ is a polynomial of degree at most $(r + 1)d$ in the variables $\mathbf{x} = (x_1, x_2, \ldots, x_m)$. Suppose $\det(M_{k,\ell})$ has a non-zero monomial of some constant degree s then we can easily (see Lemma 3.1) find an assignment to the x_i's such that $\det(M_{k,\ell}) \neq 0$. To check if $\det(M_{k,\ell})$ has a non-zero monomial of degree s, we just need to analyze $\text{hom}_s(\det(M_{k,\ell}))$. This is our overall strategy. Therefore the scenarios described above can be reformulated as below.

1. For an appropriately chosen s (depending upon d and ϵ), $\exists k, \ell \in [n - r]$ such that $\det(M_{k,\ell})$ has a non-zero monomial of degree at most s. In this case, we can "easily" find an assignment to the

x_i's such that $\det(M_{k,\ell}) \neq 0$. This ensures that $Q(x_1+\lambda_1, x_2+\lambda_2, \ldots, x_m+\lambda_m) > r$. This is our rank increasing step.

2. $\forall k, \ell \in [n-r]$, $\det(M_{k,\ell})$ has no non-zero monomials of degree at most s. In this case, we show that $r \geq (1-\epsilon)\cdot \mathrm{rank}(Q(x_1, x_2, \ldots, x_m))$.

To analyze $\hom_s(\det(M_{k,\ell}))$, it is useful to find a compact expression for $\det(M_{k,\ell})$. We now give such a compact expression for $\det(M_{k,\ell})$. In whatever follows, we use the symbol $\mathbf{a}$ to denote the row vector $\begin{bmatrix} a_1 & a_2 & \ldots & a_r \end{bmatrix}$, symbol $\mathbf{b}$ to denote the column vector $\begin{bmatrix} b_1 & b_2 & \ldots & b_r \end{bmatrix}^t$ and

$$p_L(t) \stackrel{def}{=\!=} p_0 t^r + p_1 t^{r-1} + \cdots + p_r.$$

LEMMA 4.2. *Let $M_{k,\ell}$ be as in Equation (4.9). Then we have the following equality:*

$$\det(M_{k,\ell}) = -\mathbf{a}\cdot(\mathrm{adj}(I_r+L))\cdot\mathbf{b} + c\cdot(\det(I_r+L)).$$

Proof. By Laplace expansion, we know that the following equality holds for $\det(M_{k,\ell})$:

$$\begin{aligned}
&\sum_{1\leq i\leq r}(-1)^{i+r}a_i\cdot\left(\sum_{1\leq j\leq r}(-1)^{j+r-1}\cdot b_j\cdot\det((I_r+L)_{\widehat{ij}})\right)\\
&\quad + c\cdot(\det(I_r+L))\\
&= -\sum_{1\leq i,j\leq r} a_i b_j (-1)^{i+j}\det((I_r+L)_{\widehat{ij}})\\
&\quad + c\cdot(\det(I_r+L))\\
&= -\sum_{1\leq i,j\leq r} a_i b_j (\mathrm{adj}(I_r+L))_{ij} + c\cdot(\det(I_r+L))\\
&= -\mathbf{a}\cdot\mathrm{adj}(I_r+L)\cdot\mathbf{b} + c\cdot(\det(I_r+L))\\
&= \det(M_{k,\ell}). \qquad \square
\end{aligned}$$

LEMMA 4.3. *Let $M_{k,\ell}$ be as in Equation (4.9). Then the following equality holds for $\det(M_{k,\ell})$:*

(4.10)
$$\det(M_{k,\ell}) = -\mathbf{a}\cdot\left(\sum_{i=0}^{r-1}(-1)^i p_i\cdot\left(\sum_{j=0}^{r-i-1}(-L)^j\right)\right)\cdot\mathbf{b} + c\cdot(p_0 - p_1 + \cdots + (-1)^r p_r).$$

Proof. By using Fact 3.1 we know that for any matrix A of size $r\times r$, $\det(A) = (-1)^r p_A(0)$. Now observe that $p_{I_r+L}(t) = p_L(t-1)$. Thus $\det(I_r+L) = (-1)^r p_L(-1) = (p_0 - p_1 + \cdots + (-1)^r p_r)$. Now the claim follows by using Lemma 4.2 and Theorem 3.2. $\square$

COROLLARY 4.1. *If M is the $(n-r)\times(n-r)$ matrix polynomial having the polynomial $\det(M_{u,v})$ as its $(u,v)^{th}$-entry for all $1\leq u,v\leq n-r$, then the following equality holds for M:*

(4.11)
$$M = -A\cdot\left(\sum_{i=0}^{r-1}(-1)^i p_i\cdot\left(\sum_{j=0}^{r-i-1}(-L)^j\right)\right)\cdot B + (p_0 - p_1 + \cdots + (-1)^r p_r)\cdot C.$$

Proof. It immediately follows from Lemma 4.3. $\square$

By using Corollary 4.1, it is easy to observe the following Lemma 4.4.

LEMMA 4.4. *There are $k,\ell \in [n-r]$ such that $\hom_s(\det(M_{k,\ell})) \neq 0$ if and only if $\hom_s(M)\neq 0$.*

5 The proof: analyzing the degree

In this section, we formally describe the idea described sketched in Section 4. Here we want to analyze the homogeneous degree s component $\hom_s(M)$ of M in Corollary 4.1. Recall that $p_L(t) = p_0 t^r + p_1 t^{r-1} + \cdots + p_r$. In Corollary 4.1, the coefficient of p_i is the sum of powers of $(-L)$ up to $r-i-1$. Thus, if we only want to analyze the degree s component $\hom_s(M)$ of M for some $s < \frac{r}{2}$, then we only need to consider p_i and $(-L)^i$ for $i < \frac{r}{2}$. To this end, we use the following notations in this section:

$$T \stackrel{def}{=\!=} \sum_{j=0}^{\lceil \frac{r}{2}\rceil}(-L)^j,$$

$$f \stackrel{def}{=\!=} -p_1 + \cdots + (-1)^{\lceil\frac{r}{2}\rceil} p_{\lceil\frac{r}{2}\rceil}.$$

THEOREM 5.1. *Suppose $s\in\mathbb{N}$ is such that the condition $1\leq s\leq \lfloor\frac{r}{2}\rfloor - 1$ holds. Then we have that:*

$$\begin{aligned}
\hom_s(M) &= -\hom_s((ATB-C)\cdot(p_0+f))\\
&= -\hom_s((ATB-C)\cdot(1+f)).
\end{aligned}$$

Proof. We use the fact that for $0\leq k\leq r$, we have $\mathrm{ord}(p_k)\geq k$ and $\mathrm{ord}(L^k)\geq k$. Thus to obtain the homogeneous degree s part in Corollary 4.1, it is enough to consider the p_i and L^i with $i\leq s$. Using $1\leq s\leq\lfloor\frac{r}{2}\rfloor - 1$, we obtain that $r-1-s\geq r-\lfloor\frac{r}{2}\rfloor = \lceil\frac{r}{2}\rceil$. Therefore the claimed equality follows. $\square$

By using Theorem 5.1, we see that $\hom_1(M) = C_1$ and $\hom_2(M) = C_2 + C_1\hom_1(f) - A_1B_1$. Extending this argument, we observe the following equality.

(5.12)
$$-\hom_s(M) = \hom_s(ATB-C) + \sum_{i=1}^{s-1}\hom_i(f)\cdot\hom_{s-i}(ATB-C).$$

With the aid of Equation (5.12), it is easy to prove the following Theorem 5.2.

THEOREM 5.2. *Suppose $s \in \mathbb{N}$ is such that the condition $1 \leq s \leq \lfloor \frac{r}{2} \rfloor - 1$ holds. If $\hom_\ell(M) = 0$ for all $\ell \in [s]$, then $\hom_\ell(ATB - C) = 0$ for all $\ell \in [s]$.*

Proof. We prove it by induction on ℓ. For the base case $\ell = 1$, we have $\hom_1(M) = -\hom_1(ATB - C)$. For the induction step, consider for $\ell + 1 \leq s$. By using Equation (5.12), we have that

$$-\hom_{\ell+1}(M) = \hom_{\ell+1}(ATB - C) + \sum_{i=1}^{\ell} \hom_i(f) \cdot \hom_{\ell+1-i}(ATB - C). \tag{5.13}$$

By induction hypothesis, we have $\hom_k(ATB - C) = 0$ for all $k \in [\ell]$. Therefore we obtain that:

$$\sum_{i=1}^{\ell} \hom_i(f) \cdot \hom_{\ell+1-i}(ATB - C) = 0.$$

Thus $-\hom_{\ell+1}(M) = \hom_{\ell+1}(ATB - C) = 0$. □

Let us now further reformulate the two scenarios we described above.

1. If $\hom_\ell(M) \neq 0$ for some $\ell \in [s]$ then we can implement our rank increasing step due to Lemma 4.4.
2. If $\hom_\ell(M) = 0$ for all $\ell \in [s]$ then Theorem 5.2 gives us a set of conditions on matrices A, B, C, T. We will show that these conditions on A, B, C, T can be used to bound $\text{rank}(Q(x_1, x_2, \ldots, x_m))$.

The rest of this section analyzes the condition "$\forall \ell \in [s] : \hom_\ell(ATB - C) = 0$" and gives an upper bound on $\text{rank}(Q(x_1, x_2, \ldots, x_m))$ in terms of r. In whatever follows, we use $\mathcal{A}$ to denote the $(n - r) \times rd$ matrix $\begin{bmatrix} A_1 & A_2 & \ldots & A_d \end{bmatrix}$ and $\mathcal{B}$ to denote the $rd \times (n-r)$ matrix $\begin{bmatrix} B_1^t & B_2^t & \ldots & B_d^t \end{bmatrix}^t$. To simplify the notation, define R_s to be the $rd \times rd$ block matrix (composed of d^2 blocks of size $r \times r$) whose $(i,j)^{\text{th}}$ block is $\hom_{s-(i+j)}(T)$. (We here use the convention that $\hom_\ell(T) = 0$ if $\ell < 0$.)

5.1 Analyzing the degree $s \leq d$.

LEMMA 5.1. *For all $s \geq 1$, $\hom_s(ATB) = \mathcal{A} \cdot R_s \cdot \mathcal{B}$.*

Proof. We have

$$\hom_s(ATB) = \sum_{i=1}^{d} \sum_{j=1}^{d} A_i \hom_{s-(i+j)}(T) B_j = \mathcal{A} \cdot R_s \cdot \mathcal{B}. \qquad \square$$

COROLLARY 5.1. *Suppose $r \in \mathbb{N}$ is such that the condition $1 \leq d \leq \lfloor \frac{r}{2} \rfloor - 1$ holds. If $\hom_s(M) = 0$ for all $s \in [d]$ then $C_s = \mathcal{A} \cdot R_s \cdot \mathcal{B}$ for all $s \in [d]$.*

Proof. It immediately follows from Theorem 5.2 and Lemma 5.1. □

We use the notations

$$M_1 \stackrel{def}{=\!=} \begin{bmatrix} A & C \end{bmatrix}_{(n-r)\times n} \quad \text{and} \quad M_2 \stackrel{def}{=\!=} \begin{bmatrix} B \\ C \end{bmatrix}_{n\times(n-r)}$$

in whatever follows.

LEMMA 5.2. *Suppose $r \in \mathbb{N}$ is such that the condition $1 \leq d \leq \lfloor \frac{r}{2} \rfloor - 1$ holds. If $\hom_s(M) = 0$ for all $s \in [d]$ then the following inequalities are true:*

$$\text{rank}(M_1) \leq \text{rank}(\mathcal{A}), \qquad \text{rank}(M_2) \leq \text{rank}(\mathcal{B}).$$

Proof. By using Corollary 5.1, we have the following equality:

$$C = \sum_{i=1}^{d} C_i = \mathcal{A} \cdot (\sum_{i=1}^{d} R_i) \cdot \mathcal{B} \tag{5.14}$$

Let N_1 be the $rd \times n$ matrix whose first r columns form the matrix $(\begin{bmatrix} I_r & I_r & \ldots & I_r \end{bmatrix}^t)_{r \times rd}$ and whose last $n - r$ columns are the matrix $(\sum_{i=1}^{d} R_i) \cdot \mathcal{B}$. Now the following Equation (5.15) follows from Equation (5.14):

$$M_1 = \begin{bmatrix} A & C \end{bmatrix}_{(n-r)\times n} = \mathcal{A} \cdot N_1. \tag{5.15}$$

Thus $\text{rank}(M_1) \leq \text{rank}(\mathcal{A})$. Let N_2 be the $n \times rd$ matrix whose first r rows form the matrix $\begin{bmatrix} I_r & I_r & \ldots & I_r \end{bmatrix}_{r \times rd}$ and last $n - r$ rows are the matrix $\mathcal{A} \cdot (\sum_{i=1}^{d} R_i)$. The following equality Equation (5.16) follows from Equation (5.14):

$$M_2 = \begin{bmatrix} B \\ C \end{bmatrix}_{n\times(n-r)} = N_2 \cdot \mathcal{B}. \tag{5.16}$$

Thus $\text{rank}(M_2) \leq \text{rank}(\mathcal{B})$. □

5.2 Analyzing the higher degrees.

LEMMA 5.3. *If $s \in \mathbb{N}$ is such that the condition $1 \leq s \leq \lceil \frac{r}{2} \rceil$ is true, then we have $\hom_s(T) = -\sum_{i=1}^{d-1} L_i \hom_{s-i}(T)$.*

Proof. Since $1 \leq s$, we can safely ignore the term I in the summation in the definition of T, since it has degree 0. Since $s \leq \lceil \frac{r}{2} \rceil$, we can also add the term $(-L)^{\lceil \frac{r}{2} \rceil + 1}$, since it will not contribute to $\hom_s(T)$ either. Therefore, we have

$$\begin{aligned} \hom_s(T) &= \hom_s\big(-L(I - L + \ldots \\ &\qquad \cdots + (-L)^{\lceil \frac{r}{2} \rceil - 1} + (-L)^{\lceil \frac{r}{2} \rceil}) \\ &\qquad + (-L)^{\lceil \frac{r}{2} \rceil + 1}\big) \\ &= \hom_s(-LT). \end{aligned}$$

Now the claim follows. □

LEMMA 5.4. *If $s \in \mathbb{N}$ is such that the condition $d+2 \leq s \leq \lceil \frac{r}{2} \rceil + 2$ holds, then we have $R_s = ER_{s-1}$, where*

$$E \stackrel{def}{=\!=} \begin{bmatrix} -L_1 & -L_2 & \dots & -L_d \\ I_r & 0 & \dots & 0 \\ 0 & \dots & \ddots & \vdots \\ 0 & 0 & I_r & 0 \end{bmatrix}_{rd \times rd}$$

.

Proof. It immediately follows from Lemma 5.3. □

THEOREM 5.3. *Suppose $s \in \mathbb{N}$ is such that the condition $d+1 \leq s \leq \lfloor \frac{r}{2} \rfloor - 1$ holds. If $\hom_i(M) = 0$ for all $i \in [s]$ then $\mathcal{A} \cdot R_{d+1} \cdot \mathcal{B} = \mathcal{A} \cdot ER_{d+1} \cdot \mathcal{B} = \ldots = \mathcal{A} \cdot E^{s-d-1} \cdot R_{d+1} \cdot \mathcal{B} = 0$.*

Proof. By using Theorem 5.2, we know that $\hom_\ell(ATB - C) = 0$ for all $\ell \in [s]$. Since $\deg(C) \leq d$, we get that $\forall i \in \{d+1, \ldots, s\}$, $\hom_i(ATB) = 0 = \mathcal{A} \cdot R_i \cdot \mathcal{B}$. Now the theorem follows by using the recursive formulation of R_i proved in Lemma 5.4. □

Notice that the $r \times r$ matrix L_d is non-singular because there is an assignment $\lambda_1, \lambda_2, \ldots, \lambda_m$ to the variables of L_d which makes $L_d(\lambda_1, \lambda_2, \ldots, \lambda_m) = I_r$. (Since Q is homogeneous of degree d, L_d equals the upper-right $r \times r$-submatrix of Q.) Therefore L_d as a matrix with polynomial entries is also non-singular. This implies that E is also non-singular because L_d is.

LEMMA 5.5. (LEMMA 5.3 IN [BJP18]) *Let $B \in \mathbb{F}^{n \times n}$ and*

$$B = \begin{array}{c} \overbrace{}^{r \text{ columns}} \\ r \text{ rows} \left\{ \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \right\} n-r \text{ rows} \\ \underbrace{}_{n-r \text{ columns}} \end{array} \tag{5.17}$$

Consider the following sequence of matrices $B_{22}, B_{21}B_{12}, B_{21}B_{11}B_{12}, \ldots, B_{21}B_{11}^jB_{12} \ldots$ If the first $k \geq 1$ elements in this sequence are equal to the zero matrix and B_{11} is non-singular, then $\text{rank}(B_{12}) \leq \frac{r}{k}$ or $\text{rank}(B_{21}) \leq \frac{r}{k}$.

THEOREM 5.4. *If the conditions in Theorem 5.3 are true, then $\text{rank}(M_1) \leq \frac{dr}{s-d+1}$ or $\text{rank}(M_2) \leq \frac{dr}{s-d+1}$.*

Proof. By using Theorem 5.3, we know that

$$\mathcal{A} \cdot R_{d+1} \cdot \mathcal{B} = \mathcal{A} \cdot ER_{d+1} \cdot \mathcal{B} = \ldots \cdots = \mathcal{A} \cdot E^{s-d-1} \cdot R_{d+1} \cdot \mathcal{B} = 0.$$

Consider the $(n + r(d-1)) \times (n + r(d-1))$ matrix $\mathcal{S}$ whose whose first rd rows and first rd columns form the matrix E. The last $n - r$ rows form the matrix $\mathcal{A}$ and the last $n - r$ columns form the matrix $R_{d+1}\mathcal{B}$ and the remaining entries are zero. Thus we have:

$$\mathcal{S} = \begin{bmatrix} E & R_{d+1}\mathcal{B} \\ \mathcal{A} & 0 \end{bmatrix}.$$

Now we apply Lemma 5.5 with $B_{11} = E$ and $B_{12} = R_{d+1}\mathcal{B}$ and $B_{21} = \mathcal{A}$. This implies that $\text{rank}(\mathcal{A}) \leq \frac{dr}{s-d+1}$ or $\text{rank}(R_{d+1}\mathcal{B}) \leq \frac{dr}{s-d+1}$. Note that R_{d+1} looks like below:

$$R_{d+1} = \begin{bmatrix} * & * & \dots & * & I_r \\ * & \vdots & * & I_r & 0 \\ \vdots & * & I_r & 0 & \vdots \\ * & I_r & 0 & \dots & 0 \\ I_r & 0 & \dots & 0 & 0 \end{bmatrix}.$$

In particular, R_{d+1} is non-singular. Thus $\text{rank}(R_{d+1}\mathcal{B}) = \text{rank}(\mathcal{B})$. Now the claim follows from Lemma 5.2. □

COROLLARY 5.2. *If the conditions in Theorem 5.3 are true then we have:*

$$\text{rank}(Q(x_1, x_2, \ldots, x_m)) \leq r + \frac{dr}{s-d+1} \leq r\left(1 + \frac{d}{s-d+1}\right).$$

Proof. Recall that

$$Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m) = \begin{bmatrix} I_r + L & B \\ A & C \end{bmatrix}.$$

By using Fact 3.3 and Theorem 5.4, we obtain that $\text{rank}(Q(x_1 + \lambda_1, x_2 + \lambda_2, \ldots., x_m + \lambda_m)) \leq r\left(1 + \frac{d}{s-d+1}\right)$. Now the claim follows by using Lemma 4.1. □

6 Final algorithm

Let us recall our strategy once again. We have shown above that at least one of the following conditions holds:

1. If $d+1 \leq s \leq \lfloor \frac{r}{2} \rfloor - 1$ and $\hom_\ell(M) \neq 0$ for some $\ell \in [s]$, then our rank increasing step succeeds.
2. Otherwise, we have $\text{rank}(Q(x_1, x_2, \ldots, x_m)) \leq r\left(1 + \frac{d}{s-d+1}\right)$ by Corollary 5.2.

Thus if we choose s large enough then our rank increasing step succeeds, otherwise r is already a good approximation of $\text{rank}(Q(x_1, x_2, \ldots, x_m))$ by Corollary 5.2. This leads to the following Algorithm 1, which is a natural greedy algorithm and it tries to increase the current rank as long as it can.

THEOREM 6.1. (THEOREM 2.1 RESTATED) *Algorithm 1 runs in time $O((mnd)^{\frac{d}{\epsilon}} + (\frac{md}{\epsilon})^{\frac{2d^2+2d}{\epsilon}}) \cdot$*

Algorithm 1 Greedy algorithm for $(1-\epsilon)$-approximating commutative rank

Input: A $n \times n$ matrix $Q(x_1, x_2, \ldots, x_m) = (q_{ij})_{n\times n}$ whose entries q_{ij} are homogeneous polynomials of degree d in the variables $\mathbf{x} = (x_1, x_2, \ldots, x_m)$. An approximation parameter $0 < \epsilon < 1$.

Output: $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{F}$ such that $\operatorname{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m)) \geq (1-\epsilon)\operatorname{rank}(Q(x_1, x_2, \ldots, x_m))$.

1: $\ell \leftarrow \lceil \frac{d}{\epsilon} - 1 \rceil$
2: $\lambda \leftarrow (\lambda_1, \lambda_2, \ldots, \lambda_m)$ such that $\operatorname{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m)) \geq 2\ell + 2$
▷ This is just to satisfy the condition $d + 1 \leq s \leq \lfloor \frac{r}{2} \rfloor - 1$ assumed in Corollary 5.2.
3: **while** Rank is increasing **do**
4: Check if there exist $(\mu_1, \mu_2, \ldots, \mu_m) \in H_{m,nd,\ell}$ such that
$\operatorname{rank} Q(\mu_1 + \lambda_1, \mu_2 + \lambda_2, \ldots, \mu_m + \lambda_m) > \operatorname{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m))$.
5: **if** $\operatorname{rank}(Q(\mu_1 + \lambda_1, \mu_2 + \lambda_2, \ldots, \mu_m + \lambda_m)) > \operatorname{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m))$ **then**
6: $\lambda \leftarrow \lambda + \mu$
7: **end if**
8: **end while**
9: **return** λ

$n \cdot M(n)$) and returns $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{F}$ such that $\operatorname{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m)) \geq (1-\epsilon)\operatorname{rank}(Q(x_1, x_2, \ldots, x_m))$. Here $M(n)$ is the time required to compute the rank of an $n \times n$ matrix over $\mathbb{F}$.

Proof. Let $(\lambda_1, \lambda_2, \ldots, \lambda_m)$ be the assignment returned by Algorithm 1 and $r = \operatorname{rank}(Q(\lambda_1, \lambda_2, \ldots, \lambda_m))$. We have $\ell = \lceil \frac{d}{\epsilon} - 1 \rceil$. We know that $\hom_i(M) = 0$ for $i \in [\ell]$, otherwise line 4 would succeed in increasing the rank of $Q(\lambda_1, \lambda_2, \ldots, \lambda_m)$. Here M is the matrix defined in Corollary 4.1. By using Corollary 5.2, we obtain that $\operatorname{rank}(Q) \leq r\left(1 + \frac{d}{\ell - d + 1}\right)$. Thus $r \geq \left(1 - \frac{d}{\ell+1}\right)\operatorname{rank}(Q)$. By using $\ell = \lceil \frac{d}{\epsilon} - 1 \rceil$, we know that $\ell + 1 \geq \frac{d}{\epsilon}$. Therefore $r \geq (1-\epsilon)\operatorname{rank}(Q)$.

The desired running time can also be proved easily. By using Lemma 3.1 on $H_{m,d(2\ell+2),d(2\ell+2)}$, the running time of line 2 is $O((md\ell)^{2d\ell+2d} \cdot M(n))$. The outer while loop runs at most n times, thus the total running time is at most n times the running time of one iteration. The running time of one iteration is at most $O((m^\ell(nd+1)^\ell M(n))$. Thus the claimed bound on total running time follows. □

7 Discussion and Open Problems

One can study the variants of Theorem 2.1 when the entries are polynomial sized formulas/circuits, however, the following lemma implies that it is at least as hard as PIT for formulas/circuits respectively.

Lemma 7.1. (PIT is a prerequisite) *For any $\epsilon > 0$, given an oracle access to approximation of the commutative rank of matrix Q, whose entries are from a circuit class $\mathcal{C}$ in time t. We can do PIT for circuit class $\mathcal{C}$ in time t.*

Proof. Suppose we want to test if $f \in \mathcal{C}$ is identically zero polynomial. Consider the following matrix Q.

$$Q = \begin{bmatrix} f \end{bmatrix}$$

If $f \neq 0$, then the rank of Q is 1 and any ϵ-approximation ($\epsilon > 0$) will output a positive number. Thus, just by checking if the output is positive or zero we can infer if f is identically zero or not. □

Since derandomizing PIT for general circuits is believed to be a hard problem and will imply circuit lower bounds, a natural restriction to this question of computing commutative rank when entries of matrix come from classes where PIT is already known. In particular, when the entries of the matrix are sparse polynomials or sum of powers of linear forms. We leave this as an open question. Stated differently, can we approximate algebraic rank of a set of sparse polynomials?

Our current techniques do not give anything on this, primarily because our technique of increasing the rank by finding a non-zero assignment of different homogeneous components breaks down. Concretely, if Q is a matrix with sparse entries then $\hom_s(M)$ might not be sparse. See Section 5 and Theorem 5.1 for more details.

Also, over finite fields, one could study the approximation of algebraic rank. The best known complexity of this is $\mathsf{AM} \cap \mathsf{co\text{-}AM}$ by Guo et al. [GSS18]. Can we do anything better than $\mathsf{AM} \cap \mathsf{co\text{-}AM}$ for approximating the algebraic rank over finite fields?

References

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, pages 781–793, 2004.

[ASS16] Manindra Agrawal, Nitin Saxena, and

Shubham Sahai Srivastava. Integer Factoring Using Small Algebraic Dependencies. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:14, 2016.

[ASSS12] M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: Hitting-sets, lower bounds for depth-D occur-k formulas & depth-3 transcendence degree-k circuits. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 599–614, 2012.

[BIZ18] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. *J. ACM*, 65(5):32:1–32:29, 2018.

[BJP18] Markus Bläser, Gorav Jindal, and Anurag Pandey. A deterministic PTAS for the commutative rank of matrix spaces. *Theory of Computing*, 14(3):1–21, 2018.

[BS83] W. Bauer and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983.

[DGW09] Zeev Dvir, Ariel Gabizon, and Avi Wigderson. Extractors and rank extractors for polynomial sources. *Computational Complexity*, 18(1):1–58, 2009.

[DL78] Richard A DeMillo and Richard J Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

[Edm67] Jack R. Edmonds. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards Sect. B*, 71:241–245, 1967.

[FGT16] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 754–763, 2016.

[FR04] Marc Fortin and Christophe Reutenauer. Commutative/noncommutative rank of linear matrices and subspaces of matrices of low rank. *Séminaire Lotharingien de Combinatoire*, 52:B52f, 2004.

[FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 243–252, Washington, DC, USA, 2013. IEEE Computer Society.

[FSS14] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 867–875, New York, NY, USA, 2014. ACM.

[GGOW15] Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. Operator scaling; theory and applications. *CoRR*, abs/1511.03730, 2015.

[GGOW16] A. Garg, L. Gurvits, R. Oliveira, and A. Wigderson. A deterministic polynomial time algorithm for non-commutative rational identity testing. In *57th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 109–117, Oct 2016.

[GKS16] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and commutative, read-once oblivious abps. In *Proceedings of the 31st Conference on Computational Complexity, CCC'16*, volume 50 of *LIPIcs*, pages 29:1–29:16, 2016.

[GS86] Harold N. Gabow and Matthias F. M. Stallmann. An augmenting path algorithm for linear matroid parity. *Combinatorica*, 6(2):123–150, 1986.

[GSS18] Zeyu Guo, Nitin Saxena, and Amit Sinhababu. Algebraic dependencies and PSPACE algorithms in approximative complexity. In *Proceedings of the 33rd Computational Complexity Conference, CCC '18*, volume 102 of *LIPIcs*, pages 10:1–10:21, 2018.

[GT17] Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-NC. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 821–830, 2017.

[Har09] Nicholas JA Harvey. Algebraic algorithms for matching and matroid problems. *SIAM Journal on Computing*, 39(2):679–702, 2009.

[Hig40] Graham Higman. The units of group-rings. *Proceedings of the London Mathematical Society*, 2(1):231–248, 1940.

[HJ12] Uffe Heide-Jørgensen. On the determinantal complexity of the 2-hook-immanant. *PhD Dissertation, Department of Mathematics, Aarhus University*, 2012.

[IQS17] Gábor Ivanyos, Youming Qiao, and K Venkata Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. *Computational Complexity*, 26:1–33, 2017.

[Jac41] C. G. J. Jacobi. De determinantibus functionalibus. *J. Reine Angew. Math.*, 22(4):319–359, 1841.

[Kal85] K. A. Kalorkoti. A Lower Bound for the Formula Size of Rational Functions. *SIAM J. Comp.*, 14(3):678–687, 1985. (Conference version in ICALP 1982).

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KS01] Adam R. Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 216–223, New York, NY, USA, 2001. ACM.

[KS16] Mrinal Kumar and Shubhangi Saraf. Arithmetic circuits with locally low algebraic rank. In *31st Conference on Computational Complexity, CCC 2016*, volume 50 of *LIPIcs*, pages 34:1–34:27, 2016.

[Kum18] Mrinal Kumar. On top fan-in vs formal degree for depth-3 arithmetic circuits. *CoRR*, abs/1804.03303, 2018.

[LMR10] Joseph M Landsberg, Laurent Manivel, and Nicolas Ressayre. Hypersurfaces with degenerate duals and the geometric complexity theory program. *arXiv preprint arXiv:1004.4802*, 2010.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *Proc. 2nd Internat. Conf. Fundamentals of Computation Theory (FCT'79)*, pages 565–574, 1979.

[MV97] Meena Mahajan and V. Vinay. A combinatorial algorithm for the determinant. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, pages 730–738, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[MVV87] Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354. ACM, 1987.

[Nis91] Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418. ACM, 1991.

[Orl08] James B. Orlin. A fast, simpler algorithm for the matroid parity problem. In *Proc. 13th Internat. Conf. on Integer Programming and Combinatorial Optimization (IPCO'08)*, volume 5035 of *Lecture Notes in Comp. Sci.*, pages 240–258. Springer, 2008.

[Oxl06] James G Oxley. *Matroid theory*, volume 3. Oxford university press, 2006.

[PSS18] Anurag Pandey, Nitin Saxena, and Amit Sinhababu. Algebraic independence over positive characteristic: New criterion and applications to locally low-algebraic-rank circuits. *Computational Complexity*, 27:617–670, 2018.

[RS05a] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *computational complexity*, 14(1):1–19, Apr 2005.

[RS05b] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[Sax08] Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *International Colloquium on Automata, Languages, and Programming*, pages 60–71. Springer, 2008.

[Sax09] N. Saxena. Progress on Polynomial Identity Testing. *BEATCS*, (90):49–79, 2009.

[Sax14] Nitin Saxena. Progress on polynomial identity testing-ii. In *Perspectives in Computational Complexity*, pages 131–146. Springer, 2014.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[SV07] F. Bruce Shepherd and Adrian Vetta. The demand-matching problem. *Math. Oper. Res.*, 32(3):563–578, 2007.

[SY10] A. Shpilka and A. Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[Tut47] William T Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947.

[Val79] L. G. Valiant. Completeness classes in algebra. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 249–261, New York, NY, USA, 1979. ACM.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. Symbolic and Algebraic Comput. (EUROSAM'79)*, volume 72 of *Lecture Notes in Comp. Sci.*, pages 216–226. Springer, 1979.

A A PTAS for general degree d polynomials

We have demonstrated a PTAS above for the rank of an $n \times n$ matrix $Q(x_1, x_2, \ldots, x_m)$ whose entries are homogeneous degree d polynomials in the variables $x_1, x_2, \ldots, x_m$. But entries being homogeneous polynomials is not a restriction. Here we show that even if the entries of Q are general degree d polynomials, we can still use our algorithm to approximate the rank of Q. For a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_m]$ of degree at most d, the homogenization f^H of f is a homogeneous polynomial of degree d in $\mathbb{F}[x_1, x_2, \ldots, x_m, y]$. More specifically, f^H is defined as $f^H \stackrel{def}{=\!=} \sum_{i=0}^{d} \hom_i(f) \cdot y^{d-i}$. We can extend this definition to matrix polynomials in the obvious way. More precisely, the homogenization $Q^H(x_1,\ x_2,\ \ldots,\ x_m, y)$ of a given matrix polynomial $Q(x_1,\ x_2,\ \ldots,\ x_m)$ is defined as $(Q^H)_{ij} \stackrel{def}{=\!=} (Q_{ij})^H$. Thus to homogenize a matrix, we just homogenize all its entries.

Lemma A.1. *If $Q(x_1, x_2, \ldots, x_m)$ is matrix with its entries being polynomials of degree at most d in the variables $x_1, x_2, \ldots, x_m$ and $|\mathbb{F}| > dn+1$ then* $\operatorname{rank}(Q) = \operatorname{rank}(Q^H)$.

Proof. It is clear that $\operatorname{rank}(Q) \leq \operatorname{rank}(Q^H)$ because $Q^H(x_1, x_2, \ldots, x_m, 1) = Q(x_1, x_2, \ldots, x_m)$. Now suppose that $\operatorname{rank}(Q^H) = r$. Thus there exists a non-zero $r \times r$ minor M_r of Q^H. Notice that M_r is a homogeneous polynomial of degree at most $rd \leq nd$ in the variables $x_1, x_2, \ldots, x_m, y$. Thus by using the Schwartz-Zippel lemma [Zip79, Sch80], there exist scalars $(\lambda_1, \lambda_2, \ldots, \lambda_m, \mu) \in \mathbb{F}^{m+1}$ with the property that $M_r(\lambda_1, \lambda_2, \ldots, \lambda_m, \mu) \neq 0$. Here μ can be assumed to be non-zero as $|\mathbb{F}| > dn + 1$. Since M_r is homogeneous, $\mu \neq 0$ and $M_r(\lambda_1, \lambda_2, \ldots, \lambda_m, \mu) \neq 0$, we get that $M_r\left(\frac{\lambda_1}{\mu}, \frac{\lambda_2}{\mu}, \ldots, \frac{\lambda_m}{\mu}, 1\right) \neq 0$. Thus M_r would be a non-zero minor in Q as well. Hence $\operatorname{rank}(Q) \geq \operatorname{rank}(Q^H)$. Therefore $\operatorname{rank}(Q) = \operatorname{rank}(Q^H)$. □