On the Power of Border Width-2 ABPs over Fields of Characteristic 2

Pranjal Dutta
National University of Singapore, Singapore

Christian Ikenmeyer
University of Warwick, UK

Balagopal Komarath
Indian Institute of Technology Gandhinagar, India

Harshil Mittal
Indian Institute of Technology Gandhinagar, India

Saraswati Girish Nanoti
Indian Institute of Technology Gandhinagar, India

Dhara Thakkar
Indian Institute of Technology Gandhinagar, India

Abstract

The celebrated result by Ben-Or and Cleve [SICOMP92] showed that algebraic formulas are polynomially equivalent to width-3 algebraic branching programs (ABP) for computing polynomials. i.e., $\mathsf{VF} = \mathsf{VBP}_3$. Further, there are simple polynomials, such as $\sum_{i=1}^{8} x_i y_i$, that cannot be computed by width-2 ABPs [Allender and Wang, CC16]. Bringmann, Ikenmeyer and Zuiddam, [JACM18], on the other hand, studied these questions in the setting of approximate (i.e., border complexity) computation, and showed the universality of border width-2 ABPs, over fields of characteristic $\neq 2$. In particular, they showed that polynomials that can be approximated by formulas can also be approximated (with only a polynomial blowup in size) by width-2 ABPs, i.e., $\overline{\mathsf{VF}} = \overline{\mathsf{VBP}_2}$. The power of border width-2 algebraic branching programs when the characteristic of the field is 2 was left open.

In this paper, we show that width-2 ABPs can approximate every polynomial irrespective of the field characteristic. We show that any polynomial f with ℓ monomials and with at most t odd-power indeterminates per monomial can be approximated by $\mathcal{O}\big(\ell \cdot (\deg(f) + 2^t)\big)$-size width-2 ABPs. Since ℓ and t are finite, this proves universality of border width-2 ABPs. For univariate polynomials, we improve this upper-bound from $O(\deg(f)^2)$ to $O(\deg(f))$.

Moreover, we show that, if a polynomial f can be approximated by small formulas, then the polynomial f^d, for some small power d, can be approximated by small width-2 ABPs. Therefore, even over fields of characteristic two, border width-2 ABPs are a reasonably powerful computational model. Our construction works over any field.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic branching programs, border complexity, characteristic 2

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.31

Funding *Pranjal Dutta*: Supported by the project "Foundation of Lattice-based Cryptography", funded by NUS-NCS Joint Laboratory for Cyber Security, Singapore.
Christian Ikenmeyer: Supported by EPSRC grant EP/W014882/1.
Saraswati Girish Nanoti: Funded by CSIR NET JRF Fellowship.
Dhara Thakkar: Funded by CSIR-UGC NET JRF Fellowship.

1 Introduction

The fundamental aim in computational complexity theory is to separate computational complexity classes – classes of problems that can be solved using a bounded amount of computational resources (e.g., time, space). Despite a lot of research, separating classes has remained elusive because the general computational model, Turing machines, are surprisingly difficult to prove lower bounds against. Valiant [22] proposed a computational complexity theory for families of multivariate polynomials, now called *algebraic complexity*, where the computational models only use algebraic operations such as addition $+$, multiplication $\times$, etc. The central question in algebraic complexity is to compare the computational power of the permanent and determinant polynomials, for a symbolic matrix $\mathbf{X}_n = (x_{i,j})_{i,j\in[n]}$, defined as follows:

$$\mathsf{per}_n := \mathsf{per}_n(\mathbf{X}_n) = \sum_{\sigma\in S_n}\prod_{i=1}^{n} x_{i,\sigma(i)} ,$$

$$\mathsf{det}_n := \mathsf{det}_n(\mathbf{X}_n) = \sum_{\sigma\in S_n}\mathsf{sgn}(\sigma)\prod_{i=1}^{n} x_{i,\sigma(i)} .$$

The summations above are over all permutations on n elements. Efficient algorithms to compute the determinant of a matrix whose entries are from a suitable ring (e.g. integers) are known [3, 14]. However, efficient algorithms to compute the permanent would imply that $\#\mathsf{P} = \mathsf{FP}$, which is widely believed to be false.

A sequence $(c_n)_{n\in\mathbb{N}}$ of natural numbers is called *polynomially bounded* if there exists a polynomial q with $\forall n : c_n \le q(n)$. A p-family is a sequence of polynomials whose degree and number of variables are polynomially bounded. Usually, algebraic complexity theorists are concerned with explicit p-families (e.g., $(\mathsf{det}_n)_n, (\mathsf{per}_n)_n$) because of its intimate connections to Boolean complexity.

One can define the *determinantal complexity* of a multivariate polynomial $f \in \mathbb{F}[\mathbf{x}]$ over a field $\mathbb{F}$, denoted $\mathsf{dc}(f)$, to be the smallest n such that f can be written as the determinant of an $n \times n$ matrix with entries being affine linear forms (i.e. of the form $a_0 + a_1x_1 + \cdots + a_nx_n$, where $a_i \in \mathbb{F}$). The class VBP consists of all p-families $(f_n)_{n\in\mathbb{N}}$ for which the determinantal complexity is polynomially bounded, see e.g. [13]. Interestingly, VBP can be captured by *algebraic branching programs* (ABPs) which can be thought of as a product of $w \times w$ matrices with affine linear entries, and w is called the *width* of the ABP.

The *permanental complexity* of a polynomial f, denoted $\mathsf{pc}(f)$, is the smallest n such that f can be written as the permanent of an $n \times n$ matrix of affine linear forms. The class VNP consists of all p-families $(f_n)_{n\in\mathbb{N}}$ for which the permanental complexity is polynomially bounded.

It is known that $\mathsf{VBP} \subseteq \mathsf{VNP}$ [22, 21]. One of the central questions in algebraic complexity is Valiant's conjecture of $\mathsf{VNP} \not\subseteq \mathsf{VBP}$, or equivalently proving $\mathsf{dc}(\mathsf{per}_n) = n^{\omega(1)}$ [22]. This is often known as the *determinant vs permanent* problem. The best known bounds for $\mathsf{dc}(\mathsf{per}_n)$, over $\mathbb{F} = \mathbb{C}$ is: $n^2/2 \le \mathsf{dc}(\mathsf{per}_n) \le 2^n - 1$ [15, 10].

IMM-complexity. There are plausibly *weaker* classes than VBP, such as VF that tries to capture the *algebraic formula complexity* of polynomial families. An algebraic formula is a directed tree with a unique sink vertex. The source vertices are labelled by variables or constants from $\mathbb{F}$, and each internal node of the graph is labelled by either $+$ or $\times$. Nodes compute polynomials in the natural way by induction. The size of a formula is the number of its nodes. Finally, the *algebraic formula complexity* of a polynomial f is the minimum size of a formula computing f. Ben-Or and Cleve [2] showed a surprising result that the polynomial family constructed using an iterated product of 3×3 symbolic matrices (formally it is called IMM_3, see Definition 7) is computationally *equivalent* to algebraic formulas. And further, Valiant showed that any polynomial f with algebraic formula complexity s, has determinantal complexity at most $2s$ [22]. Therefore, separation questions like VF vs. VBP, and VF vs. VNP can be framed as whether $\mathsf{immc}_3(\mathsf{det}_n) = n^{\omega(1)}$, and $\mathsf{immc}_3(\mathsf{per}_n) = n^{\omega(1)}$; for a formal definition of IMM-complexity for 3×3 matrices (immc_3), see Definition 9.

Universality vs. impossibility. It is noteworthy that all the above-mentioned complexity measures ($\mathsf{dc}, \mathsf{pc}, \mathsf{immc}_3$) are *finite* for any polynomial $f \in \mathbb{F}[\mathbf{x}]$; in other words, the model of computation defined by these complexity measures are *"universal"*. Given the phenomenon of universality and the results of Ben-Or and Cleve and Valiant, it is natural to study the computational power of iterated multiplication of 2×2 matrices. Astonishingly, Allender and Wang [1] showed an *impossibility* result that the polynomial $\sum_{i=1}^{8} x_i y_i$ *cannot* be computed using IMM_2. In other words, the IMM_2-complexity (Definition 9) of this polynomial is infinite! However, Bringmann, Ikenmeyer, and Zuiddam [4] showed that by allowing *approximations*, the polynomial family IMM_2 becomes universal! In fact, they proved a stronger statement that the IMM_2-*approximation complexity*, which we denote by $\underline{\mathsf{immc}}_2$, is polynomially related to approximate algebraic formula complexity. However, their proofs only work over fields $\mathbb{F}$ when $\mathsf{char}(\mathbb{F}) \neq 2$. They left open the following, which sets the fundamental basis for this work.

▶ **Question 1** ([4])**.** *Determine the computational power of* IMM_2 *with approximations over fields of characteristic* 2.

Border complexity & GCT. The study of *border complexity* measures, by allowing approximations in the algebraic model was first introduced in [17, 5]. Given $f \in \mathbb{F}[\mathbf{x}]$ and a suitable associated complexity measure Γ, the border-Γ complexity of f (denoted $\underline{\Gamma}(f)$) is the *smallest* n such that f can be approximated arbitrarily closely by polynomials of Γ-complexity at most n. Trivially, $\underline{\Gamma}(f) \leq \Gamma(f)$, for any f. By this definition, one can talk about the border-complexity measures such as $\underline{\mathsf{immc}}, \underline{\mathsf{dc}}, \underline{\mathsf{pc}}$ etc. Replacing a complexity measure by its border measure in a complexity class, we obtain the *closure* of this class, such as $\overline{\mathsf{VF}}$, $\overline{\mathsf{VBP}}, \overline{\mathsf{VNP}}$, and so on. The operation of going to the closure is indeed a closure operator in the sense of topology (See [11]). The original Geometric Complexity Theory (GCT) papers [17, 18] propose to use representation-theoretic techniques to separate VNP from $\overline{\mathsf{VBP}}$ by studying the determinant orbit closure, but progress has been slow. Simpler models of computation are desirable to study the easier $\mathsf{VNP} \not\subseteq \overline{\mathsf{VF}}$ conjecture, for example $\underline{\mathsf{immc}}_3$, or even the much simpler $\underline{\mathsf{immc}}_2$. This was a main motivation for [4], but their result does not work in characteristic 2. This naturally leads to the following question.

▶ **Question 2.** *How is* $\underline{\mathsf{immc}}_2$ *related to* $\underline{\mathsf{immc}}_3$ *for fields of characteristic* 2*?*

Division and powering. Strassen [20] showed that we can eliminate divisions in algebraic circuits and formulas computing polynomials without loss of efficiency. The result relies on the ability to compute small powers of polynomials efficiently. This naturally leads to the following question.

▶ **Question 3.** *Given border width-*2 *computations for polynomials f and g, can we also compute $\frac{f}{g}$ (given g divides f) and f^r, for small r, efficiently?*

More generally, one can ask, given computations for f and g, what combinations of f and g are possible in the model? A known approach to produce such results is to use Waring decompositions (See [4, 12]). Given a homogeneous degree d polynomial f, the *Waring rank* of f, denoted $\mathsf{WR}(f)$, is the smallest r such that there exist homogeneous linear polynomials $\ell_1, \cdots \ell_r$ with $f = \sum_{i=1}^{r} \ell_i^d$. Border Waring rank, denoted $\underline{\mathsf{WR}}(f)$, can be defined analogously in the border setup. For example, a border Waring decomposition for xy would allow us to compute the product fg using only addition, scaling by constants, and squaring. Over fields of characteristic 2, the border Waring rank of xy is infinite and hence, this technique becomes infeasible.

1.1 Our Contributions

Our main theorem is to answer Question 1 by showing the universality of $\underline{\mathsf{immc}}_2$:

▶ **Theorem 4** (Universality of $\underline{\mathsf{immc}}_2$). $\underline{\mathsf{immc}}_2(f)$ *is finite for every polynomial f, over all fields.*

This theorem over fields of characteristic other than two was proved by Bringmann, Ikenmeyer, and Zuiddam [4]. In fact, they prove the stronger statement that any polynomial family with small algebraic formulas approximating it can also be approximated with IMM_2 with only a polynomial blow-up in complexity. Unfortunately, our construction yields an *exponential* complexity for even simple polynomial families, such as $\prod_{i=1}^{n} x_i + \prod_{i=1}^{n} y_i + \prod_{i=1}^{n} z_i$ (see Theorem 19). However, the next theorem proves that for every polynomial with small formulas approximating them, we can approximate a small power of the polynomial using IMM_2 over any field. This partially answers Question 2.

▶ **Theorem 5** (Powering is powerful). *There exists a constant k such that for any polynomial f with a size-s formula approximating f, there is a $d \leq s^k + k$ such that $\underline{\mathsf{immc}}_2(f^d) \leq s^k + k$.*

The above theorem shows that the border width-2 ABPs are a reasonably powerful computational model. Further, Theorem 4 and Theorem 5 can be seen as weak extensions of [4], over *any* field, *regardless* of its characteristic and size.

A natural question is which interesting classes of polynomials can be efficiently approximated using IMM_2. In Theorem 19, we show that every sparse polynomial family (i.e., the number of monomials is $\mathsf{poly}(n)$) where the monomials do not have a large number of variables with odd degree can be efficiently approximated. A particularly interesting subset of this class is the class of all univariate polynomials. Applying Theorem 19 to univariate polynomials, we obtain a computation of any degree-d univariate polynomial using $O(d^2)$ operations. But, Horner's rule gives a formula for any degree-d univariate polynomial that only uses $O(d)$ operations. The following theorem is a refinement of Theorem 19 to univariates where we show that every degree-d univariate can be approximated using $O(d)$ matrices. This construction is a consequence of our partial answers towards Question 3.

▶ **Theorem 6.** *For any degree-d univariate polynomial f, we have $\underline{\mathsf{immc}}_2(f) \leq \frac{9d+4}{2}$.*

We leave open the question whether $\underline{\mathsf{immc}}_2$ is polynomially related to approximate algebraic formula complexity over fields of characteristic 2.

1.2 Comparison with previous works

As mentioned before, [4] showed that any polynomial with small border algebraic formula complexity have small $\underline{\mathsf{immc}}_2$-complexity, when $\mathsf{char}(\mathbb{F}) \neq 2$. Their proof was constructive, and *fundamentally* (& inductively) used the following identity: $x \cdot y = \frac{1}{2} \cdot \left((x+y)^2 - x^2 - y^2\right)$. One could also use even a smaller representation: $x \cdot y = \left(\frac{1}{2} \cdot (x+y)\right)^2 - \left(\frac{1}{2} \cdot (x-y)\right)^2$. However both representations use the constant $\frac{1}{2}$, and one can show that one *cannot* come up with an identity which does not use $\frac{1}{2n}$, for some $n \in \mathbb{N}$. In other words, $\mathsf{WR}(x \cdot y) = \underline{\mathsf{WR}}(x \cdot y) = \infty$ over $\mathbb{F}$ with $\mathsf{char}(\mathbb{F}) = 2$. Therefore, their construction fails miserably over characteristic 2 fields.

On the other hand, Kumar [12] showed that for any $f \in \mathbb{C}[\mathbf{x}]$, a constant multiple of f can be approximated by $\prod_{i \in [m]}(1 + \ell_i) - 1$, where ℓ_i are linear polynomials in $\mathbb{C}(\epsilon)[\mathbf{x}]$. Note that, this implies that $\underline{\mathsf{immc}}_2(f) \leq m$. The representation depends on the Waring decomposition of f, and further one can show that for the minimum m: $\underline{\mathsf{WR}}(f) \leq m \leq \deg(f) \cdot \underline{\mathsf{WR}}(f)$ [8]. However, over $\mathbb{F}$ of $\mathsf{char}(\mathbb{F}) = 2$, for any $d \geq 2$, there are d-degree polynomials (e.g., $x_1 \cdots x_d$) which has infinite border Waring rank, and hence the above universality proof fails.

In this work, we come up with a *Waring-free* proof to show the universality over characteristic 2 fields, and therefore our proofs are very different (yet *simple*) from the known constructive proofs.

1.3 Proof ideas

The key building block in the proof of universality of border width-2 ABPs over fields characteristic $\neq 2$ in [4] is a Q matrix. For a polynomial f, they define $Q(f) = \begin{pmatrix} f & 1 \\ 1 & 0 \end{pmatrix}$. Given $Q(f)$ and $Q(g)$, $Q(f+g)$ can be computed as $Q(f)Q(0)Q(g)$. So, to prove universality, it suffices to show that $Q(fg)$ can also be computed from $Q(f)$ and $Q(g)$. Bringmann, Ikenmeyer and Zuiddam [4] showed that $Q(f^2)$ can be approximately computed using $Q(f)$, and then the identity $fg = (\frac{1}{2}(f+g))^2 - (\frac{1}{2}(f-g))^2$ can be used to compute the product using squaring, addition, and scaling by constants. As discussed before, such an identity does not exist over fields of characteristic two.

We overcome this block by not trying to compute the product of two arbitrary polynomials. We observe that for universality, it is enough to be able to compute $Q(fx)$ from $Q(f)$ for an arbitrary variable x. The advantage is that since x is a variable and not an arbitrary polynomial, we can use any 2×2 matrix that contains only constants and the variable x in the computation of $Q(fx)$, whereas for computing $Q(fg)$, both f and g are available to us only as Q matrices (or in any other form that have been proved inductively). This is the key idea in Lemma 15 (see Section 4).

The source of inefficiency of Lemma 15 is that $Q(f)$ is used twice to compute $Q(fx)$. Therefore, even computing a simple polynomial such as x^n using this lemma takes $\Omega(2^n)$ matrices. Compare this to the computation of $Q(fg)$ in [4] where they use $Q(f)$ and $Q(g)$ at most three times which is enough to stay within a polynomial factor of formula complexity. In Lemma 17, we show that we can compute $Q(fg^2)$ by using $Q(f)$ once and $Q(g)$ twice (see Section 4). This lemma enables efficient computation of powers of polynomials with small formulas (Theorem 20), sparse polynomials where each monomial only contains a few variables with odd power (Theorem 19), and univariate polynomials (Theorem 24). We also use this lemma to compute powers of polynomials efficiently. That is, given a computation of $Q(f)$ using s matrices, compute $Q(f^r)$ using $O(rs)$ matrices (see Section 7). We also observe that the division $Q(\frac{f}{g^2})$ from $Q(f)$ and $Q(g)$ can be performed by combining standard division elimination techniques [20] with Lemma 17 (see Section 8).

2 Preliminaries

We consider polynomial families $f = (f_n)_{n\geq 0}$ over an arbitrary field $\mathbb{F}$. The n^{th} polynomial in the family f_n is a polynomial in $\mathbb{F}[x_1, \dots, x_m]$ where $m = \mathsf{poly}(n)$. The following polynomial family is particularly important in this paper.

▶ **Definition 7.** *For any fixed, natural $k \geq 1$, we define the polynomial family $\mathsf{IMM}_k = (\mathsf{IMM}_{k,n})$ such that $\mathsf{IMM}_{k,n}$ is the $(1,1)^{th}$ entry of the product of n matrices of order $k \times k$ where each entry of each matrix is a fresh variable, i.e., the $(i,j)^{th}$ entry of the m^{th} matrix is the variable $x_{i,j}^{(m)}$ for all $1 \leq i, j \leq k$ and $1 \leq m \leq n$.*

▶ **Definition 8.** *A* weakest projection *from a set of variables X to another set of variables Y is a mapping $X \mapsto Y \cup \mathbb{F}$. A* weak projection *is a mapping from X to affine linear forms in at most one variable in $\mathbb{F}[Y]$. For polynomials f and g, we say $f \leq^{\mathsf{wst}} g$ ($f \leq^{\mathsf{w}} g$), if there is a weakest projection (resp., weak projection) that maps g to f.*

The notion of a projection is used to compare the number of algebraic operations required to compute polynomials. Note that if f_n is computable using s operations and if $g_m \leq^{\mathsf{wst}} f_n$, then g_m is also computable using s operations. The weak variant $\leq^{\mathsf{w}}$ weakens this slightly since we can only conclude that g_m can be computed using at most $\mathsf{poly}(s)$ operations.

▶ **Definition 9.** *Let $f = (f_n)$ be a polynomial family. We define the f-*complexity wrt $\leq^{\mathsf{wst}}$ (or $\leq^{\mathsf{w}}$) of a polynomial g *as the smallest m such that $g \leq^{\mathsf{wst}} f_m$ (resp., $\leq^{\mathsf{w}}$). If there is no such m, then the f-complexity of g is ∞. We define the f-*complexity of a polynomial family $g = (g_n)$ *as the sequence $s = (s_n)$ where s_n is the f-complexity of the polynomial g_n.*

We say that f computes a polynomial g wrt $\leq^{\mathsf{wst}}$ (or, $\leq^{\mathsf{w}}$) *if f-complexity of g wrt $\leq^{\mathsf{wst}}$ (resp., $\leq^{\mathsf{w}}$) is finite.*

For $f = (f_n)$, we denote f-complexity wrt $\leq^{\mathsf{wst}}$ (or, $\leq^{\mathsf{w}}$) using fc^{wst} (resp., fc^{w}). We omit the projection from the notation if it is the weakest projection. For example, we denote det-complexity, IMM_3-complexity, and IMM_2-complexity under weakest projections by dc, immc_3, and immc_2 respectively.

▶ **Definition 10.** *A polynomial family $f = (f_n)_{n\geq 0}$ is called* universal *wrt $\leq^{\mathsf{wst}}$ (or $\leq^{\mathsf{w}}$) if for any polynomial g, the f-complexity of g wrt $\leq^{\mathsf{wst}}$ (resp., $\leq^{\mathsf{w}}$) is finite.*

We can now define the approximation equivalent of $\leq^{\mathsf{wst}}$ and $\leq^{\mathsf{w}}$.

▶ **Definition 11.** *An* approximate weakest projection *is a map from X to $Y \cup \mathbb{F}(\epsilon)$. An* approximate weak projection *is a map from X to affine linear forms in at most one variable in $\mathbb{F}(\epsilon)[Y]$.*

Given $f, g \in \mathbb{F}[X]$, we say $f \trianglelefteq^{\mathsf{wst}} g$ ($f \trianglelefteq^{\mathsf{w}} g$) if there is an approximate weakest projection (resp., approximate weak projection) that maps g to some polynomial that approximates f.

We can use these to define approximate f-complexity of polynomials.

▶ **Definition 12.** *Let $f = (f_n)$ be a polynomial family. We define the* approximate f-complexity of a polynomial g *as the smallest m such that $g \trianglelefteq^{\mathsf{wst}} f_m$ (or $g \trianglelefteq^{\mathsf{w}} f_m$). If no such m exists, we define the f-complexity of g as ∞. We define the f-complexity of a polynomial family $g = (g_n)$ as the sequence $s = (s_n)$ where s_n is the f-complexity of the polynomial g_n.*

We say that f approximately computes a polynomial g wrt $\trianglelefteq^{\mathsf{wst}}$ (or, $\trianglelefteq^{\mathsf{w}}$) *if the approximate f-complexity of g wrt $\trianglelefteq^{\mathsf{wst}}$ (resp., $\trianglelefteq^{\mathsf{w}}$) is finite.*

We denote approximate f-complexity wrt $\leq^{\mathsf{wst}}$ (or, $\leq^{\mathsf{w}}$) $\underline{fc}^{\mathsf{wst}}$ (resp., $\underline{fc}^{\mathsf{w}}$). As before, we omit the projection if it is the weakest projection.

We now introduce some additional definitions that are applicable when $f = \mathsf{IMM}_2$. In this case, we can naturally consider computation of 2×2 matrices of polynomials by f.

▶ **Definition 13.** *Let* $A = \begin{pmatrix} g_1 & g_2 \\ g_3 & g_4 \end{pmatrix}$ *where* g_1, g_2, g_3, g_4 *are polynomials. We say that* A is computed wrt $\leq^{\mathsf{wst}}$ (or, $\leq^{\mathsf{w}}$) by a sequence of m matrices *if there is a sequence of* m 2×2 *matrices, where all* $4m$ *entries are variables or constants from* $\mathbb{F}$ *(resp., affine linear forms in at most one variable), such that the product of those matrices is* A.

The above definition can be naturally extended into the setting of approximate computation. Following [4], we use the notation $\mathcal{O}(\epsilon^k)$ to denote an arbitrary polynomial in the set $\epsilon^k \mathbb{F}[\epsilon, x_1, \ldots, x_n]$.

▶ **Definition 14.** *We say that* A is approximately computed wrt $\leq^{\mathsf{wst}}$ (or, $\leq^{\mathsf{w}}$) by a sequence of m matrices *if there is a sequence of* m 2×2 *matrices, where all* $4m$ *entries are variables or constants from* $\mathbb{F}(\epsilon)$ *(resp., affine linear forms over* $\mathbb{F}(\epsilon)$ *in at most one variable), such that the product of those matrices is* $\begin{pmatrix} g_1 + \mathcal{O}(\epsilon) & g_2 + \mathcal{O}(\epsilon) \\ g_3 + \mathcal{O}(\epsilon) & g_4 + \mathcal{O}(\epsilon) \end{pmatrix}$.

We omit the projection if it is the weakest projection. All results in this paper except Theorem 26 hold wrt weakest projections.

3 Approximately computing the Allender-Wang polynomial over fields of characteristic 2

Allender and Wang showed that $\mathsf{immc}_2(\mathsf{AW}) = \infty$ where $\mathsf{AW} = \sum_{i=1}^{8} x_i y_i$. Bringmann, Ikenmeyer, and Zuiddam (See Example 3.8 in [4]) constructed an approximation to the AW polynomial when $\mathsf{char}(\mathbb{F}) \neq 2$ thereby showing that $\underline{\mathsf{immc}}_2(\mathsf{AW})$ is finite when $\mathsf{char}(\mathbb{F}) \neq 2$. Here, we show that it is finite when $\mathsf{char}(\mathbb{F}) = 2$ as well.

We restate the definition of *Q-matrix computing a polynomial f* from [4].

$$Q(f) = \begin{pmatrix} f & 1 \\ 1 & 0 \end{pmatrix}$$

Observe that $Q(f+g) = Q(f)Q(0)Q(g)$. That is, if we can compute two polynomials as Q-matrices, then we can also compute their sum as a Q-matrix. Now, let

$$F(x,y) := \begin{pmatrix} \frac{1}{\epsilon} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\epsilon} & y \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & -\epsilon \end{pmatrix}.$$

Note that $F(x,y)$ computes $\begin{pmatrix} xy & 1 \\ 1+\epsilon y & 0 \end{pmatrix}$.

Finally, the following sequence approximately computes AW:

$$\begin{pmatrix} 1 & 0 \end{pmatrix} F(x_1, y_1) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} F(x_2, y_2) \cdots \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} F(x_8, y_8) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \mathsf{AW} + \mathcal{O}(\epsilon).$$

This shows that $\underline{\mathsf{immc}}_2(\mathsf{AW}) \leq 55$. The above computation works over all fields, irrespective of the characteristic.

4 Universality of IMM_2 with approximations

The key idea in [4] that allows IMM_2 to efficiently simulate formulas is a way to compute $Q(f^2)$ from $Q(f)$ (squaring). Then, the identify $fg = ((f+g)^2 - f^2 - g^2)/2$ that is valid only when $\mathsf{char}(\mathbb{F}) \neq 2$ is used to compute $Q(fg)$ from $Q(f)$ and $Q(g)$ using addition and squaring. The following lemma allows one to multiply an arbitrary polynomial with any indeterminate when $\mathsf{char}(\mathbb{F}) = 2$.

▶ **Lemma 15.** *Let f be a polynomial. Suppose that there is a sequence, say σ, of N matrices that approximately computes $Q(f)$. Then, for any indeterminate x, there is a sequence of $2N+4$ matrices that approximately computes $Q(fx)$.*

Proof. Consider the following sequence, say σ', of $2N+4$ matrices:

$$\begin{pmatrix} \frac{1}{\epsilon} & 0 \\ 0 & 1 \end{pmatrix} \sigma|_{\epsilon\to\epsilon^2} \begin{pmatrix} \epsilon & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\epsilon} & x \\ -1 & 1 \end{pmatrix} \sigma|_{\epsilon\to\epsilon^2} \begin{pmatrix} 1 & 0 \\ 1 & -\epsilon \end{pmatrix}$$

where $\sigma\big|_{\epsilon\to\epsilon^2}$ denotes the sequence obtained from σ by replacing ϵ with ϵ^2.

Note that σ' computes

$$\begin{aligned}
&\begin{pmatrix} \frac{1}{\epsilon} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} f+\mathcal{O}(\epsilon^2) & 1+\mathcal{O}(\epsilon^2) \\ 1+\mathcal{O}(\epsilon^2) & \mathcal{O}(\epsilon^2) \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\epsilon} & x \\ -1 & 1 \end{pmatrix} \begin{pmatrix} f+\mathcal{O}(\epsilon^2) & 1+\mathcal{O}(\epsilon^2) \\ 1+\mathcal{O}(\epsilon^2) & \mathcal{O}(\epsilon^2) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & -\epsilon \end{pmatrix} \\
&= \begin{pmatrix} \frac{f}{\epsilon}+\mathcal{O}(\epsilon) & \frac{1}{\epsilon}+\mathcal{O}(\epsilon) \\ 1+\mathcal{O}(\epsilon^2) & \mathcal{O}(\epsilon^2) \end{pmatrix} \begin{pmatrix} 0 & \epsilon x+1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} f+1+\mathcal{O}(\epsilon^2) & -\epsilon+\mathcal{O}(\epsilon^3) \\ 1+\mathcal{O}(\epsilon^2) & \mathcal{O}(\epsilon^3) \end{pmatrix} \\
&= \begin{pmatrix} -\frac{1}{\epsilon}+\mathcal{O}(\epsilon) & fx+\frac{f+1}{\epsilon}+\mathcal{O}(\epsilon) \\ \mathcal{O}(\epsilon^2) & \epsilon x+1+\mathcal{O}(\epsilon^2) \end{pmatrix} \begin{pmatrix} f+1+\mathcal{O}(\epsilon^2) & -\epsilon+\mathcal{O}(\epsilon^3) \\ 1+\mathcal{O}(\epsilon^2) & \mathcal{O}(\epsilon^3) \end{pmatrix} \\
&= \begin{pmatrix} fx+\mathcal{O}(\epsilon) & 1+\mathcal{O}(\epsilon^2) \\ 1+\epsilon x+\mathcal{O}(\epsilon^2) & \mathcal{O}(\epsilon^3) \end{pmatrix}.
\end{aligned}$$

◀

We also provide a Macaulay program in Appendix A to verify the construction described in the proof of Lemma 15. Although not as powerful as multiplying two arbitrary polynomials, Lemma 15 is sufficient to prove universality. Let p be a polynomial with ℓ monomials. Note that for any monomial, say m, of p, repeatedly applying Lemma 15 gives a sequence of $\mathcal{O}\big(2^{\deg(m)}\big)$ matrices that approximately computes $Q(m)$. Thus, $Q(p)$ can be approximately computed using a sequence of $\mathcal{O}\big(\ell \cdot 2^{\deg(p)}\big)$ matrices.

Although sufficient to show universality, this is *inefficient*. Even for simple polynomials such as x^n which can be computed using $n-1$ operations, we require $\mathcal{O}(2^n)$ matrices. We can improve the efficiency by using the following lemma.

▶ Remark 16. For any degree-d monomial m, we have $\mathsf{immc}_2(m) = d$. We can write $m = y_1 \cdots y_d$ where each y_i is a variable. Then, we set the $(1,1)$ entry of the ith matrix to y_i. All other entries are 0. The product now computes m at entry $(1,1)$ and 0 elsewhere. Since this construction does not compute $Q(m)$, it is not possible to use this to compute, say $\prod_{i=1}^n x_i + \prod_{i=1}^n y_i + \prod_{i=1}^n z_i$ using $\mathsf{poly}(n)$ operations.

▶ **Lemma 17.** *Let f and g be polynomials. Suppose that there is a sequence, say σ, of N matrices that approximately computes $Q(f)$, and a sequence, say π, of M matrices that approximately computes $Q(g)$. Then, there is a sequence of $N+2M+4$ matrices that approximately computes $Q(fg^2)$.*

Proof. Consider the following sequence, say σ', of $N + 2M + 4$ matrices:

$$\begin{pmatrix} -\frac{1}{\epsilon} & 0 \\ 0 & \epsilon \end{pmatrix} \pi\Big|_{\epsilon\to\epsilon^3} \begin{pmatrix} \epsilon & 0 \\ 0 & \frac{1}{\epsilon} \end{pmatrix} \sigma\Big|_{\epsilon\to\epsilon^5} \begin{pmatrix} -\epsilon & 0 \\ 0 & \frac{1}{\epsilon} \end{pmatrix} \pi\Big|_{\epsilon\to\epsilon^3} \begin{pmatrix} \frac{1}{\epsilon} & 0 \\ 0 & \epsilon \end{pmatrix}$$

where $\sigma\Big|_{\epsilon\to\epsilon^5}$ denotes the sequence obtained from σ by replacing ϵ with ϵ^5, and $\pi\Big|_{\epsilon\to\epsilon^3}$ denotes the sequence obtained from π by replacing ϵ with ϵ^3.

Note that σ' computes

$$\begin{aligned}
&\begin{pmatrix} -\frac{1}{\epsilon} & 0 \\ 0 & \epsilon \end{pmatrix} \begin{pmatrix} g + \mathcal{O}(\epsilon^3) & 1 + \mathcal{O}(\epsilon^3) \\ 1 + \mathcal{O}(\epsilon^3) & \mathcal{O}(\epsilon^3) \end{pmatrix} \begin{pmatrix} \epsilon & 0 \\ 0 & \frac{1}{\epsilon} \end{pmatrix} \begin{pmatrix} f + \mathcal{O}(\epsilon^5) & 1 + \mathcal{O}(\epsilon^5) \\ 1 + \mathcal{O}(\epsilon^5) & \mathcal{O}(\epsilon^5) \end{pmatrix} \begin{pmatrix} -\epsilon & 0 \\ 0 & \frac{1}{\epsilon} \end{pmatrix} \\
&\quad \begin{pmatrix} g + \mathcal{O}(\epsilon^3) & 1 + \mathcal{O}(\epsilon^3) \\ 1 + \mathcal{O}(\epsilon^3) & \mathcal{O}(\epsilon^3) \end{pmatrix} \begin{pmatrix} \frac{1}{\epsilon} & 0 \\ 0 & \epsilon \end{pmatrix} \\
&= \begin{pmatrix} -g + \mathcal{O}(\epsilon^3) & -\frac{1}{\epsilon^2} + \mathcal{O}(\epsilon) \\ \epsilon^2 + \mathcal{O}(\epsilon^5) & \mathcal{O}(\epsilon^3) \end{pmatrix} \begin{pmatrix} f + \mathcal{O}(\epsilon^5) & 1 + \mathcal{O}(\epsilon^5) \\ 1 + \mathcal{O}(\epsilon^5) & \mathcal{O}(\epsilon^5) \end{pmatrix} \begin{pmatrix} -g + \mathcal{O}(\epsilon^3) & -\epsilon^2 + \mathcal{O}(\epsilon^5) \\ \frac{1}{\epsilon^2} + \mathcal{O}(\epsilon) & \mathcal{O}(\epsilon^3) \end{pmatrix} \\
&= \begin{pmatrix} -fg - \frac{1}{\epsilon^2} + \mathcal{O}(\epsilon) & -g + \mathcal{O}(\epsilon^3) \\ \epsilon^2 f + \mathcal{O}(\epsilon^3) & \epsilon^2 + \mathcal{O}(\epsilon^5) \end{pmatrix} \begin{pmatrix} -g + \mathcal{O}(\epsilon^3) & -\epsilon^2 + \mathcal{O}(\epsilon^5) \\ \frac{1}{\epsilon^2} + \mathcal{O}(\epsilon) & \mathcal{O}(\epsilon^3) \end{pmatrix} \\
&= \begin{pmatrix} fg^2 + \mathcal{O}(\epsilon) & 1 + \epsilon^2 fg + \mathcal{O}(\epsilon^3) \\ 1 - \epsilon^2 fg + \mathcal{O}(\epsilon^3) & -\epsilon^4 f + \mathcal{O}(\epsilon^5) \end{pmatrix}.
\end{aligned}$$

This proves Lemma 17. ◀

We also provide a Macaulay program in Appendix A to verify the construction described in the proof of Lemma 17. The key improvement here is that instead of using σ for $Q(f)$ two times as in Lemma 15, we can compute $Q(fx^2)$ using $Q(f)$ only once. Crucially, this allows certain monomials to be computed efficiently.

▶ **Lemma 18.** *Consider a monomial, say $m = c \cdot x_1^{k_1} \cdots x_n^{k_n}$. Let λ denote the number of odd k_i's in $k_1, \ldots, k_n$. Then, $Q(m)$ can be approximately computed using a sequence of $(5 \cdot 2^\lambda - 4) + 3 \cdot \big(\deg(m) - \lambda\big)$ matrices.*

Proof. Without loss of generality, assume that $k_1, \ldots, k_\lambda$ are the λ odd k_i's. At a high level, we start with $Q(c)$, then repeatedly apply Lemma 15 to get $Q(c \cdot x_1 \cdots x_\lambda)$, then repeatedly apply Lemma 17 to get $Q(c \cdot x_1^{k_1} \cdots x_\lambda^{k_\lambda})$, and then repeatedly applying Lemma 17 to get $Q(c \cdot x_1^{k_1} \cdots x_\lambda^{k_\lambda} x_{\lambda+1}^{k_{\lambda+1}} \cdots x_n^{k_n})$. More precisely, our construction is as follows:

We begin with the sequence $Q(c)$. Using Lemma 15 $\big($with indeterminate $x_1\big)$, we get a sequence of $2 \cdot 1 + 4 = 6$ matrices that approximately computes $Q(c \cdot x_1)$. Next, using Lemma 15 $\big($with indeterminate $x_2\big)$, we get a sequence of $2 \cdot 6 + 4 = 16$ matrices that approximately computes $Q(c \cdot x_1 x_2)$. Again, using Lemma 15 $\big($with indeterminate $x_3\big)$, we get a sequence of $2 \cdot 16 + 4 = 36$ matrices that approximately computes $Q(c \cdot x_1 x_2 x_3)$. We continue this process until finally, using Lemma 15 $\big($with indeterminate $x_\lambda\big)$, we get a sequence of $2 \cdot \big(5 \cdot 2^{\lambda-1} - 4\big) + 4 = 5 \cdot 2^\lambda - 4$ matrices that approximately computes $Q(c \cdot x_1 x_2 x_3 \cdots x_\lambda)$.

Now, using Lemma 17 $\big($with $g = x_1\big)$ $\frac{k_1 - 1}{2}$ times, we get a sequence of $(5 \cdot 2^\lambda - 4) + (2 + 4) \cdot \big(\frac{k_1 - 1}{2}\big)$ matrices that approximately computes $Q(c \cdot x_1^{k_1} x_2 \cdots x_\lambda)$. Next, using Lemma 17 $\big($with $g = x_2\big)$ $\frac{k_2 - 1}{2}$ times, we get a sequence of $(5 \cdot 2^\lambda - 4) + (2 + 4) \cdot \big(\frac{k_1 - 1}{2}\big) + (2 + 4) \cdot \big(\frac{k_2 - 1}{2}\big)$ matrices that approximately computes $Q(c \cdot x_1^{k_1} x_2^{k_2} x_3 \cdots x_\lambda)$. We continue this process until finally, using Lemma 17 $\big($with $g = x_\lambda\big)$ $\frac{k_\lambda-1}{2}$ times, we get a sequence of $(5\cdot 2^\lambda - 4) + (2+4)\cdot \big(\frac{k_1-1}{2}\big) + (2+4)\cdot\big(\frac{k_2-1}{2}\big) + \ldots + (2+4)\cdot\big(\frac{k_\lambda-1}{2}\big) = (5\cdot 2^\lambda - 4) + 3\cdot\big(\sum_{i=1}^{\lambda} k_i - \lambda\big)$ matrices that approximately computes $Q(c\cdot x_1^{k_1}x_2^{k_2}x_3^{k_3}\cdots x_\lambda^{k_\lambda})$.

Now, using Lemma 17 $\big($with $g = x_{\lambda+1}\big)$ $\frac{k_{\lambda+1}}{2}$ times, we get a sequence of $(5\cdot 2^\lambda - 4) + 3\cdot\big(\sum_{i=1}^{\lambda} k_i - \lambda\big) + (2+4)\cdot\big(\frac{k_{\lambda+1}}{2}\big)$ matrices that approximately computes $Q(c\cdot x_1^{k_1}\cdots x_\lambda^{k_\lambda}x_{\lambda+1}^{k_{\lambda+1}})$. Next, using Lemma 17 $\big($with $g = x_{\lambda+2}\big)$ $\frac{k_{\lambda+2}}{2}$ times, we get a sequence of $(5\cdot 2^\lambda - 4) + 3\cdot\big(\sum_{i=1}^{\lambda} k_i - \lambda\big) + (2+4)\cdot\big(\frac{k_{\lambda+1}}{2}\big) + (2+4)\cdot\big(\frac{k_{\lambda+2}}{2}\big)$ matrices that approximately computes $Q(c\cdot x_1^{k_1}\cdots x_\lambda^{k_\lambda}x_{\lambda+1}^{k_{\lambda+1}}x_{\lambda+2}^{k_{\lambda+2}})$. We continue this process until finally, using Lemma 17 $\big($with $g = x_n\big)$ $\frac{k_n}{2}$ times, we get a sequence of $(5\cdot 2^\lambda - 4) + 3\cdot\big(\sum_{i=1}^{\lambda} k_i - \lambda\big) + (2+4)\cdot\big(\frac{k_{\lambda+1}}{2}\big) + (2+4)\cdot\big(\frac{k_{\lambda+2}}{2}\big) + \ldots + (2+4)\cdot\big(\frac{k_n}{2}\big) = (5\cdot 2^\lambda - 4) + 3\cdot\big(\sum_{i=1}^{\lambda} k_i - \lambda\big) + 3\cdot\sum_{i=\lambda+1}^{n} k_i$ matrices that approximately computes $Q(c\cdot x_1^{k_1}\cdots x_\lambda^{k_\lambda}x_{\lambda+1}^{k_{\lambda+1}}x_{\lambda+2}^{k_{\lambda+2}}\cdots x_n^{k_n})$. That is, we get a sequence of $(5\cdot 2^\lambda - 4) + 3\cdot\big(\deg(m) - \lambda\big)$ matrices that approximately computes $Q(m)$.

This proves Lemma 18. ◀

Note that Lemma 18 allows us to compute x^n using $O(n)$ matrices.

▶ **Theorem 19.** *Let p be a polynomial with ℓ monomials, each containing at most t odd-power indeterminates. Then, $Q(p)$ can be approximately computed using a sequence of at most $\ell\cdot\big(5\cdot 2^t + 3\cdot\deg(p)\big)$ matrices.*

Proof. Let $m_1, \ldots, m_\ell$ denote the ℓ monomials of p. For each $1 \le i \le \ell$, we use Lemma 18 to get a sequence, say σ_i, of at most $(5\cdot 2^t - 4) + 3\cdot\deg(m_i)$ matrices that approximately computes $Q(m_i)$. Now, the following sequence approximately computes $Q(p)$:

$$\sigma_1\cdot Q(0)\cdot\sigma_2\cdot Q(0)\cdots Q(0)\cdot\sigma_\ell$$

Note that the number of matrices in this sequence is at most

$$(\ell-1) + \sum_{i=1}^{\ell}\big((5\cdot 2^t - 4) + 3\cdot\deg(m_i)\big) \le \ell\cdot\big(5\cdot 2^t + 3\cdot\deg(p)\big)$$

This proves Theorem 19. ◀

5 Connections to Algebraic Formulas

In this section, we explore the relationship between the computational power of width-2 ABPs and algebraic formulas. Our main theorem in this section is:

▶ **Theorem 20.** *There exists a constant k such that for any polynomial f with a size-s formula approximating it, there is a $d \le s^k + k$ such that $\underline{\mathsf{immc}}_2(f^d) \le s^k + k$.*

Proof. If the field has characteristic $\neq 2$, this can be done by using the methods in [4]. We consider fields of characteristic two. It is sufficient to consider $\mathsf{IMM}_{3,n}$ for an arbitrary n as IMM_3 is a VF-complete family. We can consider without loss of generality that n is a power of two. These polynomials have polynomial-size algebraic formulas of depth $O(\log(n))$ where every path from root to leaf has the same number of product gates. We now construct a width-two algebraic branching program inductively from the formula as follows. For every polynomial p computed at a sub-formula with product depth d, we will compute $Q(p^{2^d})$. For input gates, this is trivial. Suppose f and g are sub-formulas that have product depth d.

For the formula $f+g$, notice that $(f+g)^{2^d} = f^{2^d} + g^{2^d}$ over fields of characteristic two. We can compute $Q(f^{2^d}+g^{2^d})$ from $Q(f^{2^d})$ and $Q(g^{2^d})$. For the formula $f \cdot g$, we compute $Q\big((f^{2^d})^2 (g^{2^d})^2\big) = Q((fg)^{2^{d+1}})$ using Lemma 17. Notice that since the product depth is the same on every root to leaf path, these cases are exhaustive. Since each step can at most double the size and depth is $O(\log(n))$, the size of the resulting width-two algebraic branching program is only $\mathsf{poly}(n)$. ◀

The following remarks discuss two important consequences of this theorem. First, it allows us to extend the main result of [4] to more fields.

▶ Remark 21. Over characteristic 2, it is not clear whether one can compute f from f^d, for a polynomially-bounded d, which is a power of 2, using $\underline{\mathsf{immc}}_2$. However, over large fields of characteristic $\neq 2$, one can follow the efficient *root-finding* procedure, for e.g., see [5, 6, 19], to conclude a small border width-2 complexity of f.

Second, it allows us to reduce border PIT for formulas to border PIT for width-2 ABPs.

▶ Remark 22. The border PIT problem (for definition and further connections with lower bounds, see [16, Section 2.6], [9], or [7, Section 7.1]) for a computational model is to check whether or not the polynomial computed by the given computation is approximately 0. Theorem 20 shows that border PIT for formulas reduces to border PIT for width-2 ABPs over all fields. For fields of characteristic $\neq 2$, this was already a consequence of the main result in [4]. Theorem 20 extends this to all fields. Notice that the proof of this theorem is constructive. That is, given a formula that approximately computes f, the proof of Theorem 20 can be easily modified to produce a polynomial-time algorithm to output a width-2 algebraic program approximating f^d. Now, over any field, f^d is approximately 0 if and only if f is approximately 0.

We say that a model supports efficient computation of square roots if any computation of f^2 in the model implies the existence of a computation for f where the size is polynomially related to the computation for f^2. The following corollary establishes that if we can efficiently compute square roots approximately using width-two algebraic branching programs, then all polynomial families with constant-depth, polynomial-size circuits can be approximately computed using polynomial-size width-two algebraic branching programs.

▶ **Corollary 23.** *Suppose k is a universal constant such that given any width-two algebraic branching program of size s approximately computing a polynomial f^2, we can approximately compute f using width-two algebraic branching programs of size at most $s^k + k$. Then, any polynomial family p that has constant depth algebraic circuits of size s can be approximately computed using width-two algebraic branching programs of size* $\mathsf{poly}(s)$.

Proof. Since p has polynomial-size algebraic circuits of constant depth, it also has polynomial-size algebraic formulas of constant depth where all root to leaf paths have the same product depth. We then apply Theorem 20 to obtain a width-two algebraic branching program that computes f^{2^d}, where d is the product depth of the formula. Notice that the construction in Lemma 17 can obtain a width-two algebraic branching program that approximately computes $(f_1 \cdots f_k)^2$ in size $2\sum_{i=1}^{k} s_i + O(k)$ from those of size s_i for f_i, where $1 \leq i \leq k$, even when k is unbounded. Finally, we apply the square root computation given by the hypothesis d times to obtain a width-two algebraic branching program that approximately computes f in size $O(s^{k^d})$. ◀

6 Improved bound for univariate polynomials

For univariate polynomials, a quadratic (in degree) upper bound on immc_2 over fields of characteristic 2 follows from Theorem 19. However, we can do better. In fact, we can make this asymptotically optimal by using a two-step Horner's method.

▶ **Theorem 24.** *Let p be a univariate polynomial in x. Then, $Q(p)$ can be approximately computed using a sequence of at most $\dfrac{9 \cdot \deg(p) + 4}{2}$ matrices.*

Proof. Let $d := \deg(p)$ if $\deg(p)$ is even, and $d := \deg(p) - 1$ otherwise.
If $\deg(p)$ is even, p is of the following form:

$$a_d x^d + a_{d-1}x^{d-1} + \ldots + a_1 x + a_0.$$

Otherwise, p is of the following form:

$$a_{d+1}x^{d+1} + a_d x^d + a_{d-1}x^{d-1} + \ldots + a_1 x + a_0.$$

Note that in both the cases, p can be expressed as follows:

$$\Big(\cdots\Big((ax^2 + a_{d-1}x + a_{d-2})x^2 + a_{d-3}x + a_{d-4}\Big)x^2 + \ldots + a_3 x + a_2\Big)x^2 + a_1 x + a_0,$$

where $a := a_d$ if $\deg(p)$ is even, and $a := a_{d+1}x + a_d$ otherwise.

At a high level, our construction exploits the above expression by starting with $Q(a)$, then obtaining $Q(ax^2)$ using Lemma 17, then obtaining $Q\big(ax^2 + a_{d-1}x + a_{d-2}\big)$ by appending a few matrices, then obtaining $Q\Big(\big(ax^2 + a_{d-1}x + a_{d-2}\big)x^2\Big)$ using Lemma 17, and so on, until we finally obtain $Q(p)$. More precisely, we construct the desired sequence as follows:

First, we compute $Q(a)$. When d is even, the matrix $Q(a_d)$ computes $Q(a)$. When d is odd, we could have taken $Q(a_{d+1}x)Q(0)Q(a_d)$ as a sequence of matrices computing $Q(a)$ if we were in the weak setting. However, since we are in the weakest setting, we instead use the length-2 sequence $\begin{pmatrix} a_{d+1} & a_d \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x & \frac{1}{a_{d+1}} \\ 1 & 0 \end{pmatrix}$ to compute $Q(a)$.

Next, using Lemma 17 $\big($with $g = x\big)$, we get a sequence of at most $2 + 2 + 4 = 8$ matrices that approximately computes $Q(ax^2)$. Again, if we were in the weak setting, we could have appended this sequence with $Q(0)Q(a_{d-1}x)Q(0)Q(a_{d-2})$ to get $Q(ax^2 + a_{d-1}x + a_{d-2})$. However, since we are in the weakest setting, we instead append this sequence with $Q(0)\begin{pmatrix} a_{d-1} & a_{d-2} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x & \frac{1}{a_{d-1}} \\ 1 & 0 \end{pmatrix}$ when $a_{d-1} \neq 0$, and $Q(0)Q(a_{d-2})$ when $a_{d-1} = 0$. This gives us a sequence of at most $8 + 3 = 11$ matrices that computes $Q(ax^2 + a_{d-1}x + a_{d-2})$.

Again, using Lemma 17 $\big($with $g = x\big)$, we get a sequence of at most $11 + 2 + 4 = 17$ matrices that approximately computes $Q\big((ax^2 + a_{d-1}x + a_{d-2})x^2\big)$. As before, we append it with $Q(0)\begin{pmatrix} a_{d-3} & a_{d-4} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x & \frac{1}{a_{d-3}} \\ 1 & 0 \end{pmatrix}$ when $a_{d-3} \neq 0$, and $Q(0)Q(a_{d-4})$ when $a_{d-3} = 0$. This gives us a sequence of at most $17 + 3 = 20$ matrices that approximately computes $Q\Big((ax^2 + a_{d-1}x + a_{d-2})x^2 + a_{d-3}x + a_{d-4}\Big)$.

We continue this process. Finally, we get a sequence of at most $\dfrac{9d+4}{2} \leq \dfrac{9 \cdot \deg(p) + 4}{2}$ matrices that approximately computes $Q(p)$. This proves Theorem 24. ◀

7 Powering

Efficiently computing f^r from f, or powering, is an essential ingredient in many constructions, such as division elimination.

▶ **Lemma 25.** *Let p be a polynomial. Let $r \geq 1$ be an integer. Suppose that there is a sequence of M matrices that approximately computes $Q(p)$. Then, there is a sequence of at most $rM + 2r + 1$ matrices that approximately computes $Q(p^r)$.*

Proof. At a high level, we repeatedly use Lemma 17 to get $Q(p^2)$, $Q(p^4), \dots, Q(p^r)$ when r is even, and $Q(p^3), Q(p^5), \dots, Q(p^r)$ when r is odd. More precisely, we construct the desired sequence as follows:

Case 1: r is even. Using Lemma 17 $\big($with $f = 1$ and $g = p\big)$, we get a sequence of $1 + 2M + 4 = 2M + 5$ matrices that approximately computes $Q(p^2)$. Next, using Lemma 17 $\big($with $f = p^2$ and $g = p\big)$, we get a sequence of $(2M + 5) + 2M + 4 = 4M + 9$ matrices that approximately computes $Q(p^4)$. Again, using Lemma 17 $\big($with $f = p^4$ and $g = p\big)$, we get a sequence of $(4M + 9) + 2M + 4 = 6M + 13$ matrices that approximately computes $Q(p^6)$. We continue this process until finally, using Lemma 17 $\big($with $f = p^{r-2}$ and $g = p\big)$, we get a sequence of $\big((r-2)M + 2r - 3\big) + 2M + 4 = rM + 2r + 1$ matrices that approximately computes $Q(p^r)$.

Case 2: r is odd. Using Lemma 17 $\big($with $f = p$ and $g = p\big)$, we get a sequence of $M + 2M + 4 = 3M + 4$ matrices that approximately computes $Q(p^3)$. Next, using Lemma 17 $\big($with $f = p^3$ and $g = p\big)$, we get a sequence of $(3M + 4) + 2M + 4 = 5M + 8$ matrices that approximately computes $Q(p^5)$. Again, using Lemma 17 $\big($with $f = p^5$ and $g = p\big)$, we get a sequence of $(5M + 8) + 2M + 4 = 7M + 12$ matrices that approximately computes $Q(p^7)$. We continue this process until finally, using Lemma 17 $\big($with $f = p^{r-2}$ and $g = p\big)$, we get a sequence of $\big((r-2)M + 2r - 6\big) + 2M + 4 = rM + 2r - 2$ matrices that approximately computes $Q(p^r)$. This proves Lemma 25. ◀

8 Division Elimination

We are now ready to prove a division elimination result. The usual division elimination computes f/g from f and g given that g divides f. Since we can compute $Q(fg^2)$ efficiently from $Q(f)$ and $Q(g)$. Efficient division elimination will imply that we can compute $Q(fg) = Q(fg^2/g)$ as well. In the following theorem, we prove a weaker version of division elimination, where we show how to compute f/g^2 from f and g given g^2 divides f. This is the only construction in this paper that relies on the additional power of weak projections over weakest projections.

▶ **Theorem 26.** *Let $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ be n-variate polynomials over a sufficiently large field of characteristic 2, where $\boldsymbol{x} = (x_1, \dots, x_n)$. Suppose that there are sequences, say σ and π, of N and M matrices that approximately compute $Q(f)$ and $Q(g)$ wrt weak projections respectively. Assume that g^2 divides f. Then, there is a sequence, say η, of $\mathcal{O}\big(N^4 M(M+N)\big)$ matrices that approximately computes $Q\big(\frac{f}{g^2}\big)$ wrt weak projections.*

Proof. Define $h(\mathbf{x}) := \frac{f(\mathbf{x})}{g(\mathbf{x})^2}$. Let k be the degree of $h(\mathbf{x})$. If $g(\mathbf{0}) \neq 1$, then we find α such that $g(\mathbf{x} + \alpha) = 1 + g_1(\mathbf{x})$.

Using the sequence π, we can get a new sequence of $\mathcal{O}(M)$ matrices that approximately computes $g_1(\mathbf{x})$. We have

$$h(\mathbf{x}+\alpha) = \frac{f(\mathbf{x}+\alpha)}{(g(\mathbf{x}+\alpha))^2} = \frac{f(\mathbf{x}+\alpha)}{(1+(-1+g(\mathbf{x}+\alpha)))^2} = \frac{f(\mathbf{x}+\alpha)}{(1+g_1(x))^2} = \frac{f(\mathbf{x}+\alpha)}{1+g_1^2(\mathbf{x})} = \sum_{i\geq 0} f\cdot (g_1^2)^i$$

For each $0 \leq i \leq k/2$, we get a sequence, say η_i, of $\mathcal{O}\big(k(M+N)\big)$ matrices, that approximately computes $Q(f \cdot g_1^{2i})$ using Lemma 17.

Define $\mathcal{P}(\mathbf{x}) := \sum_{i=0}^{k/2} f \cdot (g_1^2)^i$. The following sequence, say λ, of $\mathcal{O}\big(k^2(M+N)\big)$ matrices, computes $Q(\mathcal{P})$ approximately:

$$\eta_0 \cdot Q(0) \cdot \eta_1 \cdot Q(0) \cdots Q(0) \cdot \eta_{k/2}.$$

Let $\mathcal{R}(t) := \mathcal{P}(tx_1, ..., tx_n)$. Note that $\mathcal{R}(t)$ is of the form, $\mathcal{R}(t) = b_0 + b_1 t + b_2 t^2 + \ldots + b_\ell t^\ell$, where $b_0, b_1, \ldots, b_\ell$ are polynomials in $x_1, \ldots, x_n$ over $\mathbb{F}$. Let $a_0, \ldots, a_\ell \in \mathbb{F}$. Note that

$$A \cdot \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_\ell \end{bmatrix} = \begin{bmatrix} R(a_0) \\ R(a_1) \\ \vdots \\ R(a_\ell) \end{bmatrix}, \text{ where } A := \begin{bmatrix} 1 & a_0 & a_0^2 & \ldots & a_0^\ell \\ 1 & a_1 & a_1^2 & \ldots & a_1^\ell \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & a_\ell & a_\ell^2 & \ldots & a_\ell^\ell \end{bmatrix}$$

For every $0 \leq i, j \leq \ell$, let $c_{i,j}$ denote the entry at the i^{th} row and the j^{th} column of A^{-1}. Then, we have

$$b_0 = c_{0,0} \cdot R(a_0) + c_{0,1} \cdot R(a_1) + \ldots + c_{0,\ell} \cdot R(a_\ell)$$
$$b_1 = c_{1,0} \cdot R(a_0) + c_{1,1} \cdot R(a_1) + \ldots + c_{1,\ell} \cdot R(a_\ell)$$
$$\vdots$$
$$b_\ell = c_{\ell,0} \cdot R(a_0) + c_{\ell,1} \cdot R(a_1) + \ldots + c_{\ell,\ell} \cdot R(a_\ell)$$

For every $0 \leq i \leq \ell$, we obtain a sequence, say λ_i, from λ, by replacing x_r with $a_i \cdot x_r$ for every $1 \leq r \leq n$. Note that λ_i approximately computes $Q(R(a_i))$ using $\mathcal{O}\big(k^2(M+N)\big)$ matrices.

Now, for every $0 \leq i \leq k$, the following sequence, say Γ_i, approximately computes $Q(b_i)$ using $\mathcal{O}\big(k^2\ell(M+N)\big)$ matrices:

$$\begin{bmatrix} c_{i,0} & 0 \\ 0 & 1 \end{bmatrix} \lambda_0 \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{c_{i,0}} \end{bmatrix} Q(0) \begin{bmatrix} c_{i,1} & 0 \\ 0 & 1 \end{bmatrix} \lambda_1 \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{c_{i,1}} \end{bmatrix} Q(0) \ldots Q(0) \begin{bmatrix} c_{i,\ell} & 0 \\ 0 & 1 \end{bmatrix} \lambda_\ell \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{c_{i,\ell}} \end{bmatrix}.$$

Also, we have

$$\begin{aligned} h(\mathbf{x}+\alpha) &= \text{hom}_0\big(\mathcal{P}(\mathbf{x})\big) + \text{hom}_1\big(\mathcal{P}(\mathbf{x})\big) + \ldots + \text{hom}_k\big(\mathcal{P}(\mathbf{x})\big) \\ &= b_0 + b_1 + \ldots + b_k \end{aligned}$$

Therefore, the following sequence of $\mathcal{O}\big(k^3\ell(M+N)\big)$ matrices approximately computes $Q(h(\mathbf{x}+\alpha))$:

$$\Gamma_0 \cdot Q(0) \cdot \Gamma_1 \cdot Q(0) \ldots Q(0) \cdot \Gamma_k$$

Finally, we replace $\mathbf{x}$ by $\mathbf{x}+\alpha$ in the above sequence to get a sequence, say η, that approximately computes $Q(h(\mathbf{x}))$. Note that $k \leq \deg(f) \leq N$ and $\ell \leq \deg(f) + k \cdot \deg(g) \leq \mathcal{O}(MN)$. Thus, η has $\mathcal{O}\big(N^4M(M+N)\big)$ matrices. ◀

9 Conclusion

This work successfully establishes that width-2 ABPs can approximate any polynomial *regardless* of the characteristic of the field, thus resolving a weaker version of the open question from [4]. Here are some immediate questions which require rigorous investigation.

1. Let $f \in \mathbb{F}[\mathbf{x}]$, of degree d, where $\mathrm{char}(\mathbb{F}) = 2$. Further, let $\underline{\mathsf{immc}}(f^2) = s$. Can we say that $\underline{\mathsf{immc}_2}(f) = \mathsf{poly}(s, d)$?
2. Can we prove a subexponential upper bound on $\underline{\mathsf{immc}_2}(f)$, for any exponential-sparse polynomial f, of border formula-complexity $\mathsf{poly}(n)$, over fields of characteristics 2? Of course, proving a polynomial upper bound would settle the open question of [4], proving that $\overline{\mathsf{VF}} = \overline{\mathsf{VBP}_2}$, over fields of characteristics 2 (and hence, over any field!).

References

1 Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *computational complexity*, 25(1):217–253, 2016.

2 Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992.

3 Stuart J Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information processing letters*, 18(3):147–150, 1984.

4 Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. *Journal of the ACM (JACM)*, 65(5):1–29, 2018.

5 Peter Bürgisser. The complexity of factors of multivariate polynomials. *Found. Comput. Math.*, 4(4):369–396, 2004. doi:10.1007/s10208-002-0059-5.

6 Pranjal Dutta. Discovering the roots: Unifying and extending results on multivariate polynomial factoring in algebraic complexity. *Master's thesis*, 2018.

7 Pranjal Dutta. A tale of hardness, de-randomization and de-bordering in complexity theory. *PhD Thesis*, 2022.

8 Pranjal Dutta, Fulvio Gesmundo, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Border complexity via elementary symmetric polynomials. *arXiv preprint arXiv:2211.07055*, 2022.

9 Michael A Forbes and Amir Shpilka. A pspace construction of a hitting set for the closure of small algebraic circuits. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1180–1192, 2018.

10 Bruno Grenet. An upper bound for the permanent versus determinant problem. *Theory of Computing*, 2011.

11 Christian Ikenmeyer and Abhiroop Sanyal. A note on VNP-completeness and border complexity. *Information Processing Letters*, 176:106243, 2022.

12 Mrinal Kumar. On the power of border of depth-3 arithmetic circuits. *ACM Trans. Comput. Theory*, 12(1):5:1–5:8, 2020. doi:10.1145/3371506.

13 M. Mahajan. Algebraic complexity classes. *Perspectives in Comp. Compl.: The Somenath Biswas Ann. Vol.*, pages 51–75, 2014.

14 Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago Journal of Theoretical Computer Science*, 1997(5), 1997.

15 Thierry Mignon and Nicolas Ressayre. A quadratic bound for the determinant and permanent problem. *International Mathematics Research Notices*, 2004(79):4241–4253, 2004.

16 Ketan Mulmuley. Geometric complexity theory v: Efficient algorithms for noether normalization. *Journal of the American Mathematical Society*, 30(1):225–309, 2017.

17 Ketan Mulmuley and Milind A. Sohoni. Geometric complexity theory I: an approach to the P vs. NP and related problems. *SIAM J. Comput.*, 31(2):496–526, 2001. doi:10.1137/S009753970038715X.

18 Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory II: towards explicit obstructions for embeddings among class varieties. *SIAM Journal on Computing*, 38(3):1175–1206, 2008.

19 Amit Sinhababu and Thomas Thierauf. Factorization of polynomials given by arithmetic branching programs. *computational complexity*, 30:1–47, 2021.

20 Volker Strassen. Vermeidung von Divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973. URL: `http://eudml.org/doc/151394`.

21 Seinosuke Toda. Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Transactions on Information and Systems*, 75(1):116–124, 1992.

22 Leslie G Valiant. Completeness classes in algebra. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 249–261, 1979.

A Macaulay2 source code for main constructions

Listing 1 illustrates our construction of $Q(fx)$ from $Q(f)$. The code can be run using Macaulay2. The variables `O1` through `O8` in these programs represent (arbitrary) polynomials in the ring `ZZ/2[eps,x1,...,xn]` that appear as a result of the approximation.

Listing 1 $Q(fx)$ from $Q(f)$.

```
R=ZZ/2[eps];
S=frac R;
S[f,x,O1,O2,O3,O4];
M1=matrix{{1/eps,0},{0,1}};
M2=matrix{{f+eps^2*O1,1+eps^2*O2},{1+eps^2*O3,eps^2*O4}};
M3=matrix{{eps,1},{0,1}};
M4=matrix{{1/eps,x},{-1,1}};
M5=matrix{{1,0},{1,-eps}};
print(M1*M2*M3*M4*M2*M5);
```

Listing 2 illustrates our construction of $Q(fg^2)$ from $Q(f)$ and $Q(g)$.

Listing 2 $Q(fg^2)$ from $Q(f)$ and $Q(g)$.

```
R=ZZ/2[eps];
S=frac R;
S[f,g,O1,O2,O3,O4,O5,O6,O7,O8];
M1=matrix{{-1/eps,0},{0,eps}};
M2=matrix{{g+eps^3*O5,1+eps^3*O6},{1+eps^3*O7,eps^3*O8}};
M3=matrix{{eps,0},{0,1/eps}};
M4=matrix{{f+eps^5*O1,1+eps^5*O2},{1+eps^5*O3,eps^5*O4}};
M5=matrix{{-eps,0},{0,1/eps}};
M6=matrix{{1/eps,0},{0,eps}};
print(M1*M2*M3*M4*M5*M2*M6);
```