Rabbits Approximate, Cows Compute Exactly!

Balagopal Komarath ✉
IIT Gandhinagar, India

Anurag Pandey ✉
Department of Computer Science, Saarland University, Saarland Informatics Campus, Germany

Nitin Saurabh ✉
Department of Computer Science and Engineering, IIT Hyderabad, India

Abstract

Valiant, in his seminal paper in 1979, showed an efficient simulation of algebraic formulas by determinants, showing that VF, the class of polynomial families computable by polynomial-sized algebraic formulas, is contained in VDet, the class of polynomial families computable by polynomial-sized determinants. Whether this containment is strict has been a long-standing open problem. We show that algebraic formulas can in fact be efficiently simulated by the determinant of tetradiagonal matrices, transforming the open problem into a problem about determinant of general matrices versus determinant of tetradiagonal matrices with just three non-zero diagonals. This is also optimal in a sense that we cannot hope to get the same result for matrices with only two non-zero diagonals or even tridiagonal matrices, thanks to Allender and Wang (Computational Complexity'16) which showed that the determinant of tridiagonal matrices cannot even compute simple polynomials like $x_1x_2 + x_3x_4 + \cdots + x_{15}x_{16}$.

Our proof involves a structural refinement of the simulation of algebraic formulas by width-3 algebraic branching programs by Ben-Or and Cleve (SIAM Journal of Computing'92). The tetradiagonal matrices we obtain in our proof are also structurally very similar to the tridiagonal matrices of Bringmann, Ikenmeyer and Zuiddam (JACM'18) which showed that, if we allow approximations in the sense of geometric complexity theory, algebraic formulas can be efficiently simulated by the determinant of tridiagonal matrices of a very special form, namely the continuant polynomial. The continuant polynomial family is closely related to the Fibonacci sequence, which was used to model the breeding of rabbits. The determinants of our tetradiagonal matrices, in comparison, is closely related to Narayana's cows sequences, which was originally used to model the breeding of cows. Our result shows that the need for approximation can be eliminated by using Narayana's cows polynomials instead of continuant polynomials, or equivalently, shifting one of the outer diagonals of a tridiagonal matrix one place away from the center.

Conversely, we observe that the determinant (or, permanent) of band matrices can be computed by polynomial-sized algebraic formulas when the bandwidth is bounded by a constant, showing that the determinant (or, permanent) of bandwidth k matrices for all constants $k \geq 2$ yield VF-complete polynomial families. In particular, this implies that the determinant of tetradiagonal matrices in general and Narayana's cows polynomials in particular yield complete polynomial families for the class VF.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Computational complexity and cryptography; Theory of computation → Circuit complexity

Keywords and phrases Algebraic complexity theory, Algebraic complexity classes, Determinant versus permanent, Algebraic formulas, Algebraic branching programs, Band matrices, Tridiagonal matrices, Tetradiagonal matrices, Continuant, Narayana's cow sequence, Padovan sequence

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.65

Related Version *Full Version*: `https://nitinsau.github.io/pubs/cow-sequences.pdf` [13]

Acknowledgements We thank Arkadev Chattopadhyay for herding us towards this beautiful problem and sharing his insights! We also thank Meena Mahajan for bringing the reference [3] to our notice. We also thank the organizers of GCT2022 workshop for hosting a talk by Avi Wigderson which prompted us towards this investigation. We thank anonymous reviewers for helpful suggestions.

1 Introduction

Valiant in his seminal work [20] laid the foundation for investigation of algebraic analog of the P versus NP problem, the flagship problem of theoretical computer science. He introduced algebraic formulas and determinants as models for computing polynomial families and identified them as notions of efficient computation, while the permanent family, $\text{per}_n(x_{11}, \ldots, x_{nn}) := \sum_{\sigma \in \mathbb{S}_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$ was identified as a family that is highly likely to be hard to compute. He defined the complexity class VF as the set of polynomial families that can be computed by formulas of polynomially-bounded size, and VDet as the set of families that can be expressed as the determinant of a symbolic matrix of polynomially-bounded dimension. He also showed, among other things, that a polynomial computable by an algebraic formula of size s can be expressed as the determinant of a symbolic matrix of size $(s+2) \times (s+2)$, thus showing the containment $\mathsf{VF} \subseteq \mathsf{VDet}$. Conversely, the smallest known formulas for the determinant family, $\det_n(x_{11}, \ldots, x_{nn}) := \sum_{\sigma \in \mathbb{S}_n} \text{sgn}(\sigma) \prod_{i=1}^{n} x_{i,\sigma(i)}$, have size $n^{O(\log n)}$ [11, 6]. Thus the two notions of efficient computation are not known to be equivalent. It is a long standing open problem whether algebraic formulas of polynomial size exist for the determinant family.

▶ **Problem 1.** *Is the determinant family strictly more expressive than algebraic formulas? In other words, is* $\mathsf{VF} \subsetneq \mathsf{VDet}$*?*

An improved construction of a formula for the determinant family has resisted all attempts for long, which can be interpreted as an evidence to an affirmative answer to Problem 1. Though the relationship between the classes VF and VDet is poorly understood as of now, they themselves are very natural otherwise. Not only they contain many natural examples of polynomial families, there are many differing, but equivalent, ways to define them too.

For example, the class VDet is equivalently captured by the model of algebraic branching programs of polynomial size, denoted VBP. Recall, an algebraic branching program (ABP) is a directed acyclic graph G with two special nodes, say s (source node) and t (sink node), and edges labeled with variables or constants. For every s-to-t path p in G we associate a monomial m_p obtained by multiplying the edge labels on this path. The polynomial computed by the algebraic branching program G is defined to be the sum over all monomials given by s-to-t paths, i.e., $\sum_{p:\ s\text{-to-}t \text{ path } p} m_p$. Rephrasing the characterization, we know $\mathsf{VDet} = \mathsf{VBP}$. We can assume, wlog, branching programs to be layered, i.e., the vertices are topologically ordered in layers, from left to right, such that the edges only go between consecutive layers. Then the *width* of a branching program is defined to be the maximum number of vertices in any one layer.

In an influential work, Ben-Or and Cleve [5] showed that branching programs of constant width characterize formulas. In other words, they showed $\mathsf{VF} = \mathsf{VBP}_3$, where VBP_3 denotes the class of algebraic branching programs of width 3 and polynomial size. In light of this, Problem 1 can be rephrased as asking whether $\mathsf{VBP}_3 \subsetneq \mathsf{VBP}$, that is, whether algebraic branching programs of width 3 are computationally strictly weaker than algebraic branching programs of arbitrary width. This seems even more likely when phrased this way!

In a recent work, Bringmann, Ikenmeyer and Zuiddam [8] took this one step further by showing that the topological closure of VF is equivalent to the topological closure of VBP_2, i.e. $\overline{\mathsf{VF}} = \overline{\mathsf{VBP}_2}$, where VBP_2 is the class corresponding to algebraic branching programs of width 2! Stated differently, they showed that algebraic branching programs of width 2 can efficiently *approximate* all polynomials that are efficiently computed (or, approximated) by algebraic formulas. In fact, the equivalent width 2 algebraic branching programs given by the reduction have very special structure, which make them equivalent to the determinant of tridiagonal symbolic matrices of a very special form. These tridiagonal matrices have non-trivial entries, variables and constants, on the main diagonal while the other two diagonals are fixed to all ± 1s. Determinant of such tridiagonal symbolic matrices is well-studied in the literature and is known as the *continuant*, deriving its name from continued fractions since continuants are used to represent the convergents of continued fractions. They are also related to the Fibonacci sequence via the following recursive definition: $F_0 := 1$, $F_1 := x_1$, and $F_n := x_n F_{n-1} + F_{n-2}$ for all $n \geq 2$. Thus, for a positive resolution of Problem 1, it is sufficient to show that the determinant of certain family of tridiagonal matrices, namely the continuant family $\{F_n\}$, *cannot* efficiently approximate the determinant of general matrices.

The continuant is known to have rich algebraic structures [16, 10, 9, 17], which may be helpful in separating VF from VDet. Although quite promising, an additional challenge this formulation poses is that we now need to deal with approximations. In other words, we need to show a stronger separation $\overline{\mathsf{VF}} \subsetneq \mathsf{VDet}$. It would be very pleasing if we could have the result of Bringmann, Ikenmeyer and Zuiddam [8] without using approximations. That is, if the following would be true – the continuant family $\{F_n\}$ can *efficiently exactly* simulate formulas. However, such a result is an impossibility! Allender and Wang [4] showed that the simple polynomial, $x_1x_2 + x_3x_4 + \cdots + x_{15}x_{16}$, cannot even be expressed by the continuant family, irrespective of efficiency. Thus, one may wonder what is the *simplest* class of matrices whose determinants can *efficiently exactly* simulate algebraic formulas?

Motivated by this question, we study the determinant of matrices with few diagonals, also known as band matrices, and identify two polynomial families that are as simple as the continuant family $\{F_n\}$, but unlike it they simulate formulas exactly and efficiently.

The Narayana's cows polynomial. The m-th polynomial in this family, denoted $N_m(x_1, \ldots, x_m)$, is defined by the recurrence $N_0 := 1$, $N_1 := x_1$, $N_2 := x_1x_2$, and $N_m = x_m N_{m-1} + N_{m-3}$ for all $m \geq 3$. Just as the continuant polynomial is based on the Fibonacci sequence, the Narayana's cows polynomial is based on the Narayana's cows sequence [1, 21]. This sequence originated in the following problem studied by the 14-th century mathematician Narayana Pandita in his book Ganita Kaumudi [18]: *A cow produces a calf every year. Cows start producing calves from the beginning of the fourth year. Then, starting from* 1 *cow in the first year, how many cows are there after* m *years?* This sequence is given by the recurrence: $N_m = N_{m-1} + N_{m-3}$ with $N_0 = N_1 = N_2 = 1$, where N_{m-1} gives the population after m years. Thus, the sequence captures the growth in the population of cows in the same way as the Fibonacci sequence captures the growth in the population of rabbits. The Narayana's cows sequence has wide applications in combinatorics. (See, e.g., [1, 14] and references therein.)

The Padovan polynomial. The recurrences for Fibonacci and Narayana's cows sequences are similar. Exploring this similarity and considering the only remaining two-term recurrence: $P_n = P_{n-2} + P_{n-3}$, we obtain another lesser known cousin of Fibonacci, called as the Padovan sequence [2, 23, 19]. Analogously, we can define the Padovan polynomial via the recurrence $P_0 := 1$, $P_1 := 0$, $P_2 := x_1$, and $P_n = x_{n-1}P_{n-2} + P_{n-3}$ for all $n \geq 3$. This generalizes the univariate Padovan polynomial that is known in the literature [22].

Our results complement the results of Bringmann, Ikenmeyer and Zuiddam [8] by showing that the aforementioned polynomial families, namely Narayana's cows and Padovan, based on the lesser known cousins of Fibonacci, are complete for the class VF. In other words, both families can efficiently exactly simulate formulas.

$$\begin{pmatrix} x_1 & -1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & 1 & x_n \end{pmatrix}$$

(a) Continuant polynomial $F_n := x_n F_{n-1} + F_{n-2}$.

$$\begin{pmatrix} x_1 & 0 & 1 & & \\ 1 & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & \ddots & \ddots & 0 \\ & & & 1 & x_m \end{pmatrix}$$

(b) Narayana's cows polynomial $N_m := x_m N_{m-1} + N_{m-3}$.

$$\begin{pmatrix} 0 & -x_1 & 1 & & \\ 1 & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & \ddots & \ddots & -x_{n-1} \\ & & & 1 & 0 \end{pmatrix}$$

(c) Padovan polynomial $P_n := x_{n-1} P_{n-2} + P_{n-3}$.

Figure 1 Polynomial families defined by determinants of simple matrices and their recurrences.

1.1 Our findings

We discover the simplest class of matrices whose determinants characterize algebraic formulas. We find that tetradiagonal matrices of a very special form suffice for this purpose.

▶ **Theorem 2** (Informal, See Theorem 16 and Corollary 23)**.** *The determinant family of tetradiagonal symbolic matrices is polynomially equivalent to algebraic formulas.*

In fact, the tetradiagonal matrices (Figures 1b and 1c) that is sufficient for efficiently simulating algebraic formulas are remarkably similar to the tridiagonal matrices (Figure 1a) used by Bringmann, Ikenmeyer and Zuiddam [8] to efficiently approximate algebraic formulas. It follows from the above theorem and Allender and Wang's separation [4], that tetradiagonal matrices are more expressive than tridiagonal matrices, but at the same time it can also be seen (Figure 1) to be nearly as simple as tridiagonal matrices – having just one extra diagonal whose entries are all 0s! We thus have the following equivalent reformulation of Problem 1.

> *Is the minimum size of a tetradiagonal matrix whose determinant equals* $\det_n$ *superpolynomially large, where* $\det_n$ *is the determinant of a general* $n \times n$ *symbolic matrix?*

This further motivated us to investigate matrices with few non-zero diagonals. Such matrices are called *band matrices* in the literature. We say that a matrix M is a band matrix of type (k_1, k_2) if all the non-zero entries of the matrix is concentrated between k_1 diagonals below the main diagonal and k_2 diagonals above the main diagonal. That is, $M_{ij} = 0$ if $j < i - k_1$ or $j > i + k_2$. A band matrix of type (k_1, k_2) will also be referred as (k_1, k_2)-diagonal matrix. The bandwidth of such matrices are defined to be $k := \max(k_1, k_2)$.

For example, diagonal matrices are $(0, 0)$-diagonal and has bandwidth 0, tridiagonal matrices are $(1, 1)$-diagonal with bandwidth 1, tetradiagonal matrices are either $(1, 2)$-diagonal or $(2, 1)$-diagonal with bandwidth 2, and pentadiagonal matrices are $(2, 2)$-diagonal with bandwidth 2. Figures 1b and 1c are examples of $(1, 2)$-diagonal matrices.

If follows from Theorem 2 that $(1, 2)$-diagonal matrices can simulate formulas, and hence any (k_1, k_2)-diagonal matrix can simulate formulas as long as $\min(k_1, k_2) \geq 1$ and $\max(k_1, k_2) \geq 2$. It is then interesting to investigate the converse, i.e., for which (k_1, k_2)-diagonal matrices their determinants have small formulas?

We observe that determinants of bandwidth k matrices can be computed by polynomial-sized algebraic formulas when the bandwidth k is bounded by a constant. In fact, our constructions give efficient *(syntactic) multilinear* ABPs and circuits for low bandwidth matrices. These are circuits for which *every* intermediate polynomial that is computed is also multilinear. A polynomial is said to be *multilinear* if every monomial of the polynomial is multilinear, and a monomial is called *multilinear* if every variable has degree at most 1 in it. In comparison, polynomial size circuits for the determinant of general matrices given by Berkowitz [6] and polynomial size ABPs given by Mahajan and Vinay [15] are non-multilinear.

▶ **Theorem 3** (Informal, See Corollary 23)**.** *Determinants of symbolic band matrices are computable by polynomial-sized algebraic formulas when bandwidth is bounded by a constant.*

In fact, the above theorem holds for the permanent of a band matrix too. Combining Theorems 2 and 3, we get a nice characterization of algebraic formulas in terms of determinants (or, permanents) of band matrices of small bandwidth. In other words, determinants of band matrices with bounded bandwidth yield polynomial families which are complete for the complexity class VF.

▶ **Theorem 4** (Informal, See Theorem 16, Theorem 19, and Corollary 23)**.** *For all constant $k \geq 2$, the determinant (or, permanent) family of symbolic matrices of bandwidth k is* VF*-complete.*

1.2 Proof methods

Ideas for Theorem 2 (Simulating formulas via determinant of tetradiagonal matrices). We prove Theorem 2 in Section 3, where we begin with tetradiagonal matrices of type $(1, 2)$. That is, the non-zero entries are limited to one diagonal below the main diagonal, the main diagonal, and two diagonals above the main diagonal. We first show that the symbolic determinant of such tetradiagonal matrices can be written as a product of 3×3 matrices whose entries are variables (or their negations), 0, and 1, where the number of matrices in the product is linear in the size of the original matrix. This is obtained by exploiting a simple recurrence revealed while computing the determinant of these $(1, 2)$ tetradiagonal matrices using Laplace expansion, see Lemma 12. Thus, to prove Theorem 2, it is sufficient to show that algebraic formulas can be efficiently simulated by the matrix product of the 3×3 matrices obtained above. In fact, Ben-Or and Cleve, in their simulation of algebraic formulas using width 3 algebraic branching programs, showed that algebraic formulas can be efficiently simulated by the matrix product of 3×3 matrices. Thus, it might be tempting to conclude that we are already done. However, it turns out that the 3×3 matrices whose products equals the determinant of tetradiagonal matrices desire more structure than the matrices used in the proof of Ben-Or and Cleve. This is where the core technical novelty of our work lies – we show that algebraic formulas can indeed be efficiently simulated by product of 3×3 matrices of the form whose products are equivalent to the determinant of $(1, 2)$-tetradiagonal matrices. In fact, we are able to efficiently simulate formulas with even more structure on the matrices, allowing us to conclude that formulas can be efficiently simulated by tetradigonal matrices where the variable entries are only on the main diagonal, the diagonal below the main diagonal is all 1s, whereas the two diagonals above the main diagonal are all 0s and all 1s respectively, see Section 3.1 for details.

Ideas for Theorem 3 (Formulas for determinant of symbolic band matrices). Theorem 3 is relatively simpler to derive from the literature. We prove it in Section 4 taking two different constructions for computing determinants of general matrices and carefully specializing those constructions in the case of bandwidth k matrices, ensuring that the undesirable blowups are limited to parameter k, allowing us to get polynomial-sized formulas when k is bounded by a constant. In our first construction, we modify the construction of Grenet for computing permanent of an $n \times n$ matrix using algebraic branching programs. For bandwidth k matrices, we are able to get syntactic multilinear ABPs of length linear in the size of matrix and exponential in the bandwidth, see Theorem 22 for details. Applying standard conversion from ABPs to formulas yield Theorem 3. This gives us a formula of depth $O(k \log(n))$ and size $n^{O(k)}$. In our second construction, we adapt the generalized Laplace expansion to low bandwidth matrices, see Theorem 26 for details. The construction yields a syntactic multilinear arithmetic circuit of size $O(\exp(k)n)$ and depth $O(\text{poly}(k) \log(n))$, which can be converted to algebraic formulas using standard conversion from circuits to formulas, giving an alternative proof of Theorem 3.

The rest of this paper is organized as follows: Section 3 gives efficient simulations of algebraic formulas via determinant of tetradiagonal symbolic matrices. Subsections 3.1 and 3.2 show that Narayana's cows polynomials and Padovan polynomials are complete for VF. Section 4 shows that determinants of all matrices with constant bandwidth have polynomial size formulas. See the full version [13] for omitted proofs.

2 Preliminaries

We define computational models that are of interest in this paper.

▶ **Definition 5.** *An* algebraic circuit *C is a rooted directed acyclic graph where the source nodes are labeled by elements of $\mathbb{F}$ or variables $x_1, \ldots, x_n$, and the internal nodes have in-degree 2 and are labeled by $+$ or $\times$. It naturally computes a polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ in a bottom-up fashion. An* algebraic formula *is a circuit whose underlying graph is a tree. The* size *of a circuit is the number of nodes in the graph and the* depth *of a circuit is the number of edges on the longest path from the root to some source node.*

We recall the definition of ABPs.

▶ **Definition 6.** *An ABP is a layered directed acyclic graph with source node s and sink node t such that each edge is labeled by a variable or a constant. The polynomial computed by the ABP is given by $\sum_p m_p$, where p is an s to t path and m_p is the product of edge labels on the path p. The* width *of an ABP is the maximum number of nodes in any layer and the* length *is the number of layers.*

It is easy to see that width-w, length-ℓ ABPs are equivalent to product of a sequence of ℓ matrices of order $w \times w$.

We now define a notion of reduction that allows us to relate the complexity of polynomials under the above model.

▶ **Definition 7.** *A polynomial $f(x) \in \mathbb{F}[x_1, \ldots, x_n]$ is a* projection *of a polynomial $g(y) \in \mathbb{F}[y_1, \ldots, y_m]$, denoted $f \leq g$, if and only if $f(x_1, \ldots, x_n) = g(a_1, \ldots, a_m)$, where $a_i \in \mathbb{F} \cup \{x_1, x_2, \ldots, x_n\}$.*

It is easy to see that if g is computed by a formula of size s and depth d and $f \leq g$, then f is also computed by a formula of size at most s and depth at most d.

As is usually the case in algorithms and complexity, formula size or depth for fixed polynomials is rarely of interest. Instead, we look at families of polynomials and the asymptotic growth of size and depth of formulas computing them.

▶ **Definition 8.** *A* polynomial family *over a field* $\mathbb{F}$ *is a sequence* $f = (f_n)_{n\in\mathbb{N}}$ *of polynomials such that both the number of variables and the degree of* f_n *are polynomially bounded functions in* n*.*

A sequence of formulas $F = (F_n)_{n\in\mathbb{N}}$ *is said to compute a polynomial family* $f = (f_n)$ *if and only if* F_n *computes* f_n *for all* n*. The* size *and* depth *of the sequence* F *is defined as functions such that* $s(n)$ *and* $d(n)$ *are the size and depth of* F_n*.*

We now extend the definition of reductions to families.

▶ **Definition 9.** *A polynomial family* (f_n) *is a* p-projection *of another family* (g_m)*, denoted* $(f_n) \leq_p (g_m)$ *if and only if there exists a polynomially bounded function* $t : \mathbb{N} \mapsto \mathbb{N}$ *such that* $f_n \leq g_{t(n)}$ *for all* n*.*

Note that if (g_n) is computed by polynomial size formulas and $(f_n) \leq_p (g_n)$, then (f_n) is also computed by polynomial size formulas.

▶ **Definition 10.** *The algebraic class* VF *is defined as the set of all polynomial families that have polynomial size formulas.*

The class VDet *(equivalently* VBP*) is defined as the set of all polynomial families* f *such that* $f \leq_p \det$*, where* $\det$ *is the family of determinants over* $n \times n$ *symbolic matrices.*

Using the above notion of reduction, we can define completeness for classes.

▶ **Definition 11.** *A polynomial family* $f = (f_n)$ *is said to be* complete *for a class* $\mathcal{C}$ *w.r.t* p*-projections if and only if* $f \in \mathcal{C}$ *and for all* $g \in \mathcal{C}$*, we have* $g \leq_p f$*.*

Note that det is trivially complete for VDet. Ben-Or and Cleve [5] showed that the polynomial family $\mathrm{per}_{3,n}$, the $(1,1)$ entry of the product of n 3×3 symbolic matrices is complete for VF.

3 Determinant of $(1,2)$-diagonal matrix versus algebraic formulas

In this section, we show that the determinants of $(1,2)$-diagonal symbolic matrices are polynomially equivalent to algebraic formulas, thereby, proving Theorem 2. We begin with the easier direction, that is, by showing that the determinant of $(1,2)$-diagonal symbolic matrix has polynomial-sized algebraic formulas. In fact, we give a polynomial-sized algebraic branching programs for them of width-3, which can then be converted into a polynomial-sized formula using a divide and conquer algorithm.

▶ **Lemma 12.** *The determinant (or, permanent) of* $(1,2)$*-diagonal symbolic matrix of dimensions* $n \times n$ *can be computed by a width-*3 *syntactic multilinear ABP of length at most* $3n - 2$*.*

In particular, they can be computed by (syntactic) multilinear formulas of size $\mathsf{poly}(n)$*.*

Proof. Let M denote the following $(1,2)$-diagonal symbolic matrix,

$$M = \begin{pmatrix} x_{11} & x_{12} & x_{13} & & \\ x_{21} & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & x_{n-2,n} \\ & & \ddots & \ddots & x_{n-1,n} \\ & & & x_{n,n-1} & x_{n,n} \end{pmatrix}. \tag{1}$$

For $0 \leq i \leq n-1$, define $K(n-i)$ to be the determinant of the principal submatrix of M obtained by deleting both the first i rows and columns. Furthermore, set $K(0) := 1$ and $K(-1) := 0$. Note that, by definition, $K(n) = \det(M)$ and $K(1) = x_{nn}$. Then we have the following recursive formula for $K(n)$:

$$K(n) \quad = x_{11}K(n-1) - x_{12}x_{21}K(n-2) + x_{13}x_{32}x_{21}K(n-3). \tag{2}$$

The correctness of the above formula easily follows from a backward induction on i. Rewriting the recurrence in a matrix form we obtain

$$\begin{bmatrix} K(n) \\ K(n-1) \\ K(n-2) \end{bmatrix} = \begin{bmatrix} x_{11} & -x_{12}x_{21} & x_{13}x_{32}x_{21} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} K(n-1) \\ K(n-2) \\ K(n-3) \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} x_{21} & x_{11} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -x_{12} & x_{13} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & x_{32} \end{bmatrix} \begin{bmatrix} K(n-1) \\ K(n-2) \\ K(n-3) \end{bmatrix} \tag{4}$$

Unrolling Eq. (4) and using $K(1) = x_{nn}$, $K(0) = 1$, and $K(-1) = 0$ we obtain the claimed width-3 ABP for $K(n)$. ◀

We now consider a special kind of $(1,2)$-diagonal symbolic matrices where entries in both the lowermost and the uppermost diagonals are only 1. We show that the determinant (or, permanent) of such a matrix is equivalent to a special kind of width-3 ABP. These matrices would serve as the key building block in our main proofs. However, we first need a name for the special kind of $(1,2)$-diagonal matrices that we are going to be dealing with.

▶ **Definition 13.** *Let $(\alpha,\beta,\gamma,\delta) \in (\mathbb{F} \cup \{*\})^4$. A $(1,2)$-diagonal matrix is said to be of type $(\alpha,\beta,\gamma,\delta)$ if all entries on the lowermost diagonal, main diagonal, first upper diagonal and second upper diagonal equals α, β, γ and δ respectively. Furthermore, if α, β, γ, or δ equals $*$ then the entries on the respective diagonals are* ***not*** *restricted.*

For example, a general $(1,2)$-diagonal symbolic matrix, shown in Equation 1, is of type $(*,*,*,*)$ and a $(1,2)$-diagonal matrix of type $(\alpha,\beta,\gamma,\delta) \in \mathbb{F}^4$ is also a *Toeplitz* matrix. The special kind of $(1,2)$-diagonal matrices that we consider are of type $(1,*,*,1)$. We now characterize the determinant of such matrices by a restricted width-3 ABP where the interconnections between layers are given by a special 3×3 matrix.

▶ **Lemma 14.** *Let M denote the following $(1,2)$-diagonal symbolic matrix of type $(1,*,*,1)$ of dimension $n \times n$:*

$$M = \begin{pmatrix} x_{11} & x_{12} & 1 & & \\ 1 & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & \ddots & \ddots & x_{n-1,n} \\ & & & 1 & x_{n,n} \end{pmatrix}.$$

Then, $\det(M)$ *is given by the* $(1,1)$ *entry of the following iterated matrix multiplication over* 3×3 *matrices,*

$$\begin{bmatrix} x_{11} & -x_{12} & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{22} & -x_{23} & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdots\cdots \begin{bmatrix} x_{(n-1)(n-1)} & -x_{(n-1)n} & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{nn} & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Conversely, the $(1,1)$ *entry of the following iterated matrix multiplication:*

$$\begin{bmatrix}\alpha_1 & \beta_1 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\alpha_2 & \beta_2 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\cdots\cdots\begin{bmatrix}\alpha_{(n-1)} & \beta_{(n-1)} & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\alpha_n & \beta_n & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix},$$

is given by the determinant of the following $(1,2)$*-diagonal matrix of type* $(1,*,*,1)$*,*

$$M=\begin{pmatrix}\alpha_1 & -\beta_1 & 1 & & \\ 1 & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & 1\\ & & \ddots & \ddots & -\beta_{n-1}\\ & & & 1 & \alpha_n\end{pmatrix}.$$

Proof. The equivalence follows from observing that in this special case the recurrence of (3) becomes

$$\begin{bmatrix}K(n)\\ K(n-1)\\ K(n-2)\end{bmatrix}=\begin{bmatrix}x_{11} & -x_{12} & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}K(n-1)\\ K(n-2)\\ K(n-3)\end{bmatrix},$$

$$=\begin{bmatrix}x_{11} & -x_{12} & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\cdots\begin{bmatrix}x_{(n-1)(n-1)} & -x_{(n-1)n} & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}K(1)\\ K(0)\\ K(-1)\end{bmatrix},$$

where $K(i), -1 \le i \le n$, as defined in the proof of Lemma 12, is the determinant of the principal submatrix of M obtained by deleting both the first $n-i$ rows and columns with $K(0)=1$ and $K(-1)=0$. ◀

3.1 Narayana's cows polynomial is VF-complete

In this section, we simulate algebraic formulas with tetradiagonal matrices of type (1,$*$,0,1). The determinant of such matrices follow the same recurrence as that of Narayana's cows polynomial described in Section 1. This simulation along with Lemma 12 finishes the proof of completeness of Narayana's cows polynomial families for the class VF.

We know from Lemma 14 that the determinant (or, permanent) of $(1,2)$-diagonal matrices of type $(1,*,0,1)$ is equivalent to the $(1,1)$ entry of an iterated matrix multiplication where the base matrices are of the form: $\begin{bmatrix}* & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$. For notational convenience, let us denote the base matrix $\begin{bmatrix}z & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$ by $A(z)$. In the following we will only work with iterated matrix multiplication over the base matrix $A(*)$ and use the equivalence given by Lemma 14 to represent the matrix product as the determinant (or, permanent) of $(1,2)$-diagonal matrix of type $(1,*,0,1)$.

For a better understanding of the algorithm we will present the algorithm in a recursive way. In particular we will have intermediate computations where the matrices will be of the form $\begin{bmatrix}0 & f & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$. We will denote such matrices by $B(f)$. Note that $A(0)=B(0)$. We now state and prove our simulation of formulas as a product of base matrices $A(z)$.

▶ **Lemma 15.** *Let p be a polynomial computed by a formula of depth d. Then, both $A(p)$ and $A(-p)$ can be expressed as an iterated matrix multiplication of length at most $30 \cdot 4^d - 29$ over the base matrices $A(z)$, where z is either a field constant, a variable, or a negated variable.*

Proof. The proof is by induction on depth.

Base case: $d = 0$. Then it computes either a field constant, a variable or a negated variable which can be represented by a single base matrix $A(z)$, where z is the label of the node.

Induction step: $d = m$. There are two cases to be considered depending on whether the node at depth m is an addition or a multiplication node.

Case 1: (Addition). Suppose $p = f + g$, where f and g are computable by depth $m - 1$ formulas. By induction hypothesis, we can express both $A(f)$ and $A(g)$. Then, $A(p) = A(f) \cdot A(0) \cdot A(0) \cdot A(g)$. In other words,

$$\begin{bmatrix} f+g & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} g & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Similarly one can express $A(-p)$.

Case 2: (Multiplication). Suppose $p = f \cdot g$, where f and g are computable by depth $m - 1$ formulas. We will use the following equation to compute $f \cdot g$.

$$A(f \cdot g) = A(0) \cdot A(0) \cdot B(-g) \cdot B(f) \cdot A(0) \cdot B(g) \cdot B(-f). \tag{5}$$

We will now show how to compute $B(f)$ using matrices of type $A(\cdot)$ which will complete the recursive algorithm. Similar to Eq. (5), the following equation computes $B(f \cdot g)$ using matrices of type $A(\cdot)$.

$$B(f \cdot g) = A(0) \cdot A(-f) \cdot A(0) \cdot A(g) \cdot A(f) \cdot A(0) \cdot A(-g). \tag{6}$$

We can thus use appropriate substitutions in Eq. (6) to get $B(f)$, $B(g)$, $B(-f)$, and $B(-g)$ and complete the algorithm. However, note that to compute $B(f)$ we need to make two calls to f as $A(-f)$ and $A(f)$. This would result in a total length of $O(8^d)$. To bring down the length to $O(4^d)$, we now show how to compute $B(f)$ using a single call to $A(f)$. Consider the following equation:

$$B(f) = A(0) \cdot B(-1) \cdot A(0) \cdot A(1) \cdot A(f) \cdot A(0) \cdot A(-1) \cdot B(1) \cdot A(0) \cdot A(0). \tag{7}$$

We can use Eq. (6) to obtain $B(-1)$ and $B(1)$, thus completing the algorithm to compute $B(f)$ with a single call to $A(f)$. We can now use equations (5) and (7) to compute $A(f \cdot g)$. Similarly one can express $A(-p)$.

The upper bound on the length of the iterated matrix multiplication follows from the following recurrence: $T(d) \leq 4 \cdot T(d-1) + 87$ and $T(0) = 1$. ◀

As a corollary to Lemmas 12 and 15, we obtain the following characterization of formulas.

▶ **Theorem 16.** *Let M_n denote the following $(1, 2)$-diagonal symbolic matrix of type $(1, *, 0, 1)$ of dimension $n \times n$:*

$$M_n = \begin{pmatrix} x_1 & 0 & 1 & & \\ 1 & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & \ddots & \ddots & 0 \\ & & & 1 & x_n \end{pmatrix}.$$

Then the sequences of polynomials $\{\det(M_n)\}_{n \geq 1}$ and $\{\text{per}(M_n)\}_{n \geq 1}$ are VF*-complete with respect to p-projections.*

Proof. Follows from Lemmas 12 and 15, and depth reduction of formulas [7]. ◀

We are now all set to deduce the completeness of the Narayana's Cows polynomial family for class VF.

▶ **Theorem 17.** *Narayana's cows polynomial family is* VF*-complete.*

Proof. We observe that the determinants of the sequence of $(1, 2)$-diagonal symbolic matrices of type $(1, *, 0, 1)$ follow the recurrence $N_m = x_m N_{m-1} + N_{m-3}$ for all $m \geq 3$, which is precisely the recurrence defining the Narayana's cows polynomials as described in Section 1. ◀

3.2 Padovan polynomial is VF-complete

In this section, we simulate algebraic formulas with tetradiagonal matrices of type $(1, 0, *, 1)$ instead. This time, the determinant of such matrices follow the same recurrence as that of Padovan polynomial described in Section 1. This simulation along with Lemma 12 finishes the proof of completeness of Padovan polynomial families for the class VF.

Again from Lemma 14 we know that the determinant (or, permanent) of $(1, 2)$-diagonal matrices of type $(1, 0, *, 1)$ is equivalent to the $(1, 1)$ entry of an iterated matrix multiplication where the base matrices are of the form: $\begin{bmatrix} 0 & * & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$. Recall we denote base matrices of the form $\begin{bmatrix} 0 & z & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ by $B(z)$. In the following we will only work with iterated matrix multiplication over the base matrix $B(*)$ and use the equivalence given by Lemma 14 to represent the matrix product as the determinant (or, permanent) of $(1, 2)$-diagonal matrix of type $(1, 0, *, 1)$.

▶ **Lemma 18.** *Let p be a polynomial computed by a formula of depth d. Then, both $B(p)$ and $B(-p)$ can be expressed as an iterated matrix multiplication of length at most $30 \cdot 4^d - 29$ over the base matrices $B(z)$, where z is either a field constant, a variable, or a negated variable.*

The proof is analogous to the proof of Lemma 15. (See the full version [13].)

As a corollary to Lemmas 12 and 18, we obtain another characterization of formulas.

▶ **Theorem 19.** *Let M_n denote the following $(1, 2)$-diagonal symbolic matrix of type $(1, 0, *, 1)$ of dimension $n \times n$:*

$$M_n = \begin{pmatrix} 0 & x_1 & 1 & & \\ 1 & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & \ddots & \ddots & x_{n-1} \\ & & & 1 & 0 \end{pmatrix}.$$

Then the sequences of polynomials $\{\det(M_n)\}_{n \geq 2}$ and $\{\text{per}(M_n)\}_{n \geq 2}$ are VF*-complete with respect to p-projections.*

Proof. The containment in VF follows from Lemma 12. While the hardness follows by translating the iterated product in Lemma 18 to a $(1, 2)$-diagonal symbolic matrix of type $(1, 0, *, 1)$ using Lemma 14. Note that to apply Lemma 14 one has to multiply the iterated product on the right by $B(0)$ (to move the polynomial to $(1, 1)$ entry). However, this only increases the length by 1. Finally using the depth reduction of formulas [7] completes the proof. ◀

We are now all set to deduce the completeness of Padovan polynomial family for class VF.

▶ **Theorem 20.** *Padovan polynomial family is* VF*-complete.*

Proof. We observe that the determinants of the sequence of $(1, 2)$-diagonal symbolic matrices of type $(1, 0, *, 1)$ in Theorem 19 follow the recurrence $P_n = x_{n-1}P_{n-2} + P_{n-3}$, for all $n \geq 3$, if we negate all variables in the matrix, which is precisely the recurrence for the Padovan polynomials as described in Section 1. ◀

4 Matrices of small bandwidth

Our main goal in this section is to prove that for all fixed k, the determinant of matrices of bandwidth k can be computed by polynomial sized formulas. Along with the results in Section 3, this gives a complete characterization of the algebraic complexity of the determinant of constant bandwidth matrices (Theorem 24). Following the spirit of parameterized algorithms, we consider the bandwidth k as a parameter, and show that we can construct efficient syntactic multilinear ABPs (Theorem 22) and circuits (Theorem 26) for computing the determinant where the undesirable blowup (exponential for size, polynomial for depth) is limited to the parameter k.

Our parameterized constructions are derived from Grenet's syntactic multilinear ABP construction for the $n \times n$ permanent [12] and the generalized Laplace expansion that constructs syntactic multilinear circuits for the $n \times n$ determinant and permanent. We state the bounds given by those constructions below:

▶ **Lemma 21.** *The determinant (or, permanent) of an $n \times n$ symbolic matrix can be computed by a syntactic multilinear circuit of size $O(n2^n)$ and depth $O(n)$. Moreover, it can be computed by a syntactic multilinear ABP of length at most $n + 2$ and width at most $\binom{n}{n/2}$.*

Notice that the ABP in Lemma 21 has width that is exponential in n. Our construction for matrices of bandwidth k shows that this exponential blowup can be limited to k.

▶ **Theorem 22.** *The determinant (permanent) of a (k, k)-diagonal symbolic matrix of dimension $n \times n$ can be computed using a syntactic multilinear ABP of length $n + 2$ and width $\binom{2k}{k}$.*

Proof. We begin with a high-level recall of Grenet's construction [12]. In his construction, the start node is in layer 0. All monomials computed at layer i correspond to some permutation that maps rows $[i]$ to some set of i columns. Further, a node in a particular layer keeps track of the subset of columns in the monomials computed at that node. This means that in layer $n/2$, it has to keep track of $\binom{n}{n/2}$ distinct sets resulting in exponential (in n) width. The edges between layers are specified such that these invariants are preserved.

We now build a layered ABP for small bandwidth matrices that is a modification of Grenet's construction.

For matrices of bandwidth k, we can make use of the fact that rows that are separated by at least $2k$ rows have no common non-zero columns. Therefore, instead of keeping track of a subset of all columns, we can keep track of a subset of only a few columns. More specifically, any monomial computed at layer i (assume $k \leq i \leq n - k$ for simplicity, the rest of the rows are handled similarly) must pick i columns from $[i + k]$ since all columns further to the right are zero for these rows. Moreover, the columns $[i - k]$ have to be picked by the first i rows since these columns are zero from row $i + 1$. Therefore, rows up to i must pick exactly k columns from the $2k$ sized set of columns $[i - k + 1, i + k]$. In layer i, we have exactly one node for each k sized subset of this $2k$ sized set. This ABP has $n+2$ layers and each layer has at most $\binom{2k}{k}$ nodes. This is precisely where we improve over Grenet's construction when specialized to matrices of bandwidth k. We refer to the full version [13] for details. ◀

By using standard conversion from ABP to formula, we obtain the following corollary.

▶ **Corollary 23.** *For all fixed k, the determinant (or, permanent) of symbolic matrices of bandwidth k can be computed using polynomial sized formulas.*

Along with the results in Section 3, the above corollary gives a complete characterization of the algebraic complexity of determinant (or, permanent) of constant bandwidth matrices.

▶ **Theorem 24.** *For all constant $k \geq 2$, the determinant (or, permanent) family of symbolic matrices of bandwidth k is* VF*-complete.*

▶ Remark 25. For completeness, we add that for $k = 0$ (symbolic diagonal matrices), the determinant (or, permanent) family is complete for width-1 ABPs, and for $k = 1$, the determinant (or, permanent) family is complete for width-2 ABPs.

The ABP given by Theorem 22 has depth n. On the other hand, converting it to a formula makes the depth $O(k\log(n))$ but the size $n^{O(k)}$. If we are interested in arithmetic circuits, we can eliminate the dependence of k in the exponent of n while keeping the depth logarithmic in n. Compared to Lemma 21, our construction, which is an adaption of the generalized Laplace expansion to low bandwidth matrices, limits the exponential blowup in size *and* the polynomial blowup in depth to the parameter k.

▶ **Theorem 26.** *The determinant (or, permanent) of an $n \times n$ (k, k)-diagonal symbolic matrix can be computed using a syntactic multilinear circuit of size $O(exp(k)n)$ and depth $O(k\log(n))$.*

We refer to the full version [13] for details.

References

1 OEIS Foundation Inc. (2022). Narayana's Cows Sequence, Entry A000930 in the On-Line Encyclopedia of Integer Sequences. `https://oeis.org/A000930`, 1964. Accessed: 2022-04-26.
2 OEIS Foundation Inc. (2022). Padovan Sequence, Entry A000931 in the On-Line Encyclopedia of Integer Sequences. `https://oeis.org/A000931`, 1964. Accessed: 2022-04-26.
3 E. Allender, V. Arvind, and M. Mahajan. Arithmetic complexity, kleene closure, and formal power series. *Theory Comput. Syst.*, 36(4):303–328, 2003.
4 E. Allender and F. Wang. On the power of algebraic branching programs of width two. *Comput. Complex.*, 25(1):217–253, 2016.
5 M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992.
6 S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984.
7 R. P. Brent. The parallel evaluation of general arithmetic expressions. *J. ACM*, 21(2):201–206, 1974.
8 K. Bringmann, C. Ikenmeyer, and J. Zuiddam. On algebraic branching programs of small width. *J. ACM*, 65(5):32:1–32:29, 2018.
9 C. Conley and V. Ovsienko. Lagrangian Configurations and Symplectic Cross-Ratios. *Mathematische Annalen*, pages 1105–1145, 2018.
10 C. Conley and V. Ovsienko. Rotundus: Triangulations, Chebyshev Polynomials, and Pfaffians. *Math Intelligencer*, 40:45–50, 2018.

11 L. Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976.
12 B. Grenet. An Upper Bound for the Permanent versus Determinant Problem. Manuscript, 2011.
13 B. Komarath, A. Pandey, and N. Saurabh. Rabbits approximate, cows compute exactly! Manuscript, 2022. URL: https://nitinsau.github.io/pubs/cow-sequences.pdf.
14 X. Lin. On the Recurrence Properties of Narayana's Cows Sequence. *Symmetry*, 13(1), 2021. URL: https://www.mdpi.com/2073-8994/13/1/149.
15 M. Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Electron. Colloquium Comput. Complex.*, 4, 1997.
16 S. Morier-Genoud. Coxeter's Frieze Patterns at the Crossroads of Algebra, Geometry and Combinatorics. *Bulletin of the London Mathematical Society*, 47(6):895–938, 2015.
17 S. Morier-Genoud and V. Ovsienko. Farey Boat: Continued Fractions and Triangulations, Modular Group and Polygon Dissections. *Jahresber. Dtsch. Math. Ver.*, 121:91–136, 2019.
18 Narayana Pandita. Ganita Kaumudi, 1356. India.
19 I. Stewart. Tales of a neglected number. *Scientific American*, 274(6):102–103, 1996. URL: http://www.jstor.org/stable/24989576.
20 L. G. Valiant. Completeness Classes in Algebra. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 249–261, 1979.
21 Wikipedia contributors. Narayana Pandita (mathematician). https://en.wikipedia.org/w/index.php?title=Narayana_Pandita_(mathematician)&oldid=1071293682, 2022. [Online; accessed 26-April-2022].
22 Wikipedia contributors. Padovan Polynomials. https://en.wikipedia.org/w/index.php?title=Padovan_polynomials&oldid=1080802324, 2022. [Online; accessed 27-April-2022].
23 Wikipedia contributors. Padovan Sequence. https://en.wikipedia.org/w/index.php?title=Padovan_sequence&oldid=1078995920, 2022. [Online; accessed 27-April-2022].