Reconstruction Algorithms for Sums of Affine Powers

Ignacio García-Marco
LIF, Aix-Marseille Université

Pascal Koiran
LIP, Ecole Normale Supérieure de Lyon
Université de Lyon

Timothée Pecatte
LIP, Ecole Normale Supérieure de Lyon
Université de Lyon

ABSTRACT

A sum of affine powers is an expression of the form

$$f(x) = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i}.$$

Although quite simple, this model is a generalization of two well-studied models: Waring decomposition and Sparsest Shift. For these three models there are natural extensions to several variables, but this paper is mostly focused on univariate polynomials. We propose algorithms that find the smallest decomposition of f in the first model (sums of affine powers) for an input polynomial f given in dense representation. Our algorithms only work in situations where the smallest decomposition is unique, and we provide conditions that guarantee the uniqueness of the smallest decomposition.

1. INTRODUCTION

Let $\mathbb{F}$ be any characteristic zero field and let $f \in \mathbb{F}[x]$ be a univariate polynomial. This work concerns the study of expressions of f as a linear combination of powers of affine forms.

MODEL 1.1. *We consider expressions of f of the form:*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i}$$

with $\alpha_i, a_i \in \mathbb{F}$, $e_i \in \mathbb{N}$. We denote by $AffPow_{\mathbb{F}}(f)$ the minimum value s such that there exists a representation of the previous form with s terms.

This model was already studied in [8], where we gave explicit examples of polynomials of degree d requiring $\mathrm{AffPow}_{\mathbb{R}}(f) = \Omega(d)$ terms for the field $\mathbb{F} = \mathbb{R}$.

The main goal of this work is to design algorithms that reconstruct the optimal representation of polynomials in this model, i.e., algorithms that receive as input $f \in \mathbb{F}[x]$ and compute the exact value $s = \mathrm{AffPow}_{\mathbb{F}}(f)$ and a set of triplets of coefficients, nodes and exponents $\{(\alpha_i, a_i, e_i) \,|\, 1 \leq i \leq s\} \subseteq \mathbb{F} \times \mathbb{F} \times \mathbb{N}$ such that $f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i}$. We assume that f is given in dense representation, i.e., as a tuple of $\deg(f) + 1$ elements of $\mathbb{F}$.

ISSAC '17, July 25-28, 2017, Kaiserslautern, Germany
© 2017 ACM. ISBN 978-1-4503-5064-8/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3087604.3087605

Model 1.1 extends two already well-studied models. The first one is the Waring model, where all the exponents are equal to the degree of the polynomial, i.e., $e_i = \deg(f)$ for all i.

MODEL 1.2. *For a polynomial f of degree d, we consider expressions of f of the form:*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{d}$$

with $\alpha_i, a_i \in \mathbb{F}$. We denote by $Waring_{\mathbb{F}}(f)$ the Waring rank *of f, which is the minimum value s such that there exists a representation of the previous form with s terms.*

Waring rank has been studied by algebraists and geometers since the 19th century. The algorithmic study of Model 1.2 is usually attributed to Sylvester. We refer to [17] for the historical background and to section 1.3 of that book for a description of the algorithm (see also Kleppe [21] and Proposition 46 of Kayal [19]). Most of the subsequent work was devoted to the multivariate generalization[1] of Model 1.2, with much of the 20th century work focused on the determination of the Waring rank of generic polynomials [1, 6, 17]. A few recent papers [23, 4] have begun to investigate the Waring rank of specific polynomials such as monomials, sums of coprime monomials, the permanent and the determinant.

The second model that we generalize is the Sparsest Shift model, where all the shifts a_i are required to be equal.

MODEL 1.3. *For a polynomial f, we consider expressions of f of the form:*

$$f = \sum_{i=1}^{s} \alpha_i (x - a)^{e_i}$$

with $\alpha_i, a \in \mathbb{F}, e_i \in \mathbb{N}$. We denote by $Sparsest_{\mathbb{F}}(f)$ the minimum value s such that there exists a representation of the previous form with s terms.

This model and its variations have been studied in the computer science literature at least since Borodin and Tiwari [5]. Some of these papers deal with multivariate generalizations [14, 16, 10], with "supersparse" polynomials[2] [12] or establish condition for the uniqueness of the sparsest shift [22]. A much more general model is studied in [15], namely, arithmetic circuits with addition, multiplication and powering gates (rational exponents are allowed in

[1]In the literature, Waring rank is usually defined for homogeneous polynomials. After homogenization, the univariate model 1.2 becomes bivariate and the "multivariate generalization" therefore deals with homogeneous polynomials in 3 variables or more.

[2]In that model, the size of the monomial x^d is defined to be $\log d$ instead of d as in the usual dense encoding.

that paper). It is suggested at the end of [10] to allow "multiple shifts" instead of a single shift, and this is just what we do in this paper. More precisely, as is apparent from Model 1.1, we do not place any constraint on the number of distinct shifts: it can be as high as the number s of affine powers. It would also make sense to place an upper bound k on the number of distinct shifts. This would provide a smooth interpolation between the sparsest shift model (where $k = 1$) and Model 1.1, where $k = s$.

1.1 Our results

We provide both structural and algorithmic results. Our structural results are presented in Section 3. We study the uniqueness of the optimal representation as a sum of affine powers. It turns out that our reconstruction algorithms only work in a regime where the uniqueness of optimal representations is guaranteed. We provide more structural results in [9]: we compare the expressive power of our three models, and study specifically the case where $\mathbb{F}$ is the field of real numbers (it turns out that we have sharper structural results for $\mathbb{R}$ than for other fields).

As already explained, we present algorithms that find the optimal representation of an input polynomial f. We achieve this goal in several cases, but we do not solve the problem in its full generality. One typical result is as follows (see Theorem 4.4 in Section 4 for a more detailed statement which includes a description of the algorithm).

THEOREM 1.4. *Let $f \in \mathbb{F}[x]$ be a polynomial that can be written as*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i},$$

where the constants $a_i \in \mathbb{F}$ are all distinct, $\alpha_i \in \mathbb{F} \setminus \{0\}$, and $e_i \in \mathbb{N}$. Assume moreover that $n_i \le (3i/4)^{1/3} - 1$ for all $i \ge 2$, where n_i denotes the number of indices j such that $e_j \le i$.

Then, $\mathrm{AffPow}_{\mathbb{F}}(f) = s$. Moreover, there is a polynomial time algorithm that receives $f = \sum_{i=0}^{d} f_i x^i \in \mathbb{F}[x]$ as input and computes the s-tuples of coefficients $C(f) = (\alpha_1, \dots, \alpha_s)$, of nodes $N(f) = (a_1, \dots, a_s)$ and exponents $E(f) = (e_1, \dots, e_s)$.

From the point of view of the optimality of representations, it is quite natural to assume an upper bound on the numbers n_i. Indeed, if there is an index j such that $n_j > j+1$ then the powers $(x-a_i)^{e_i}$ are linearly dependent, and there would be a smaller expression of f as a linear combination of these polynomials.[3] We would therefore have $\mathrm{AffPow}_{\mathbb{F}}(f) < s$ instead of $\mathrm{AffPow}_{\mathbb{F}}(f) = s$. It would nonetheless be interesting to relax the assumption $n_i \le (3i/4)^{1/3} - 1$ in this theorem. Another restriction is the assumption that the shifts a_i are all distinct. We relax that assumption in [9] but we still need to assume that all the exponents e_i corresponding to a given shift $a_i = a$ belong to a "small" interval (see [9, Theorem 5.3] for a precise statement). Alternatively, we can assume instead that there is a large gap between the exponents in two consecutive occurences of the same shift as in [9, Theorem 5.8].

In [9, Section 6] we extend the model of sums of affine powers to several variables by considering expressions of the form

$$f(x_1, \dots, x_n) = \sum_{i=1}^{s} \alpha_i \ell_i(x_1, \dots, x_n)^{e_i},$$

where $e_i \in \mathbb{N}$, $\alpha_i \in \mathbb{F}$ and ℓ_i is a (non constant) linear form for all i. We begin a study of reconstruction algorithms for this multivariate model.

[3] It is hardly more difficult to show that one must have $n_j \le \lceil \frac{j+1}{2} \rceil$ for any optimal expression, see [8, Proposition 18].

1.2 Main tools

Most of our results hinge on the study of certain differential equations satisfied by the input polynomial f. We consider differential equations of the form

$$\sum_{i=0}^{k} P_i(x) f^{(i)} = 0 \tag{1}$$

where the P_i's are polynomials. If the degree of P_i is bounded by i for every i, we say that (1) is a *Shifted Differential Equation (SDE)* of order k. The name *SDE* is a reference to the method of shifted partial derivatives [26]. Our algorithmic results can be viewed as an application of this lower bound method, and more precisely of its univariate version [20]. Section 2 recalls some (mostly standard) background on differential equations and the Wronskian determinant.

When f is a polynomial with an expression of size s in Model 1.1 we prove in Proposition 2.6 that f satisfies a "small" SDE, of order $2s - 1$. The basic idea behind our algorithms is to look for one of these "small" SDEs satisfied by f, and hope that the powers $(x - a_i)^{e_i}$ in an optimal decomposition of f satisfy the same SDE. This isn't just wishful thinking because the SDE from Proposition 2.6 is satisfied not only by f but also by the powers $(x - a_i)^{e_i}$.

Unfortunately, this basic idea by itself does not yield efficient algorithms. The main difficulty is that f could satisfy several SDE of order $2s - 1$. We can efficiently find such a SDE by linear algebra (see Remark 2.7), but what if we don't find the "right" SDE, i.e., the SDE which (by Proposition 2.6) is guaranteed to be satisfied by f *and* by the powers $(x - a_i)^{e_i}$? One way around this difficulty is to assume that the exponents e_i are all sufficiently large compared to s. In this case we can show that every SDE of order $2s - 1$ which is satisfied by f is also satisfied by $(x - a_i)^{e_i}$. This fact is established in Corollary 4.2, and yields the following result (see Theorem 4.3 in Section 4 for a more detailed statement which includes a description of the algorithm).

THEOREM 1.5 (BIG EXPONENTS). *Let $f \in \mathbb{F}[x]$ be a polynomial that can be written as*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i},$$

where the constants $a_i \in \mathbb{F}$ are all distinct, $\alpha_i \in \mathbb{F} \setminus \{0\}$ and $e_i > 5s^2/2$. Then, $\mathrm{AffPow}_{\mathbb{F}}(f) = s$. Moreover, there is a polynomial time algorithm that receives $f = \sum_{i=0}^{d} f_i x^i \in \mathbb{F}[x]$ as input and computes the s-tuples of coefficients $C(f) = (\alpha_1, \dots, \alpha_s)$, of nodes $N(f) = (a_1, \dots, a_s)$ and exponents $E(f) = (e_1, \dots, e_s)$.

The algorithm of Theorem 1.4 is more involved: contrary to Theorem 1.5, we cannot determine all the terms $(x - a_i)^{e_i}$ in a single pass. Solving the SDE only allows the determination of some (high degree) terms. We must then subtract these terms from f, and iterate.

1.3 Models of computation

Our algorithms take as inputs polynomials with coefficients in an arbitrary field $\mathbb{K}$ of characteristic 0. At this level of generality, we need to be able to perform arithmetic operations (additions, multiplications) and equality tests between elements of $\mathbb{K}$. When we write that an algorithm runs in polynomial time, we mean that the number of such steps is polynomial in the input size. This is a fairly standard setup for algebraic algorithms (it is also interesting to analyze the bit complexity of our algorithms for some specific fields such as the field of rational numbers; more on this at the end of this subsection and in Section 1.4). An input polynomial of degree d is represented simply by the list of coefficients of its $d+1$ monomials, and its size thus equals $d+1$. In addition to arithmetic operations and equality tests, we need to be able to compute roots of polynomials with coefficients in $\mathbb{K}$. This is in general unavoidable: for an optimal decomposition of $f \in \mathbb{K}[x]$ in Model 1.1, the coefficients α_i, a_i may lie in an extension field $\mathbb{F}$ of $\mathbb{K}$ (see Section 3 and more precisely [9, Example 3.3] for the case $\mathbb{K} = \mathbb{R}, \mathbb{F} = \mathbb{C}$). If the optimal decomposition has size s, we need to compute roots of polynomials of degree at most $2s-1$. As a rule, root finding is used only to output the nodes a_i of the optimal decomposition,[4] but the "internal working" of our algorithms remains purely rational (i.e., requires only arithmetic operations and comparisons). This is similar to the symbolic algorithm for univariate sparsest shifts of Giesbrecht, Kaltofen and Lee ([10], p. 408 of the journal paper), which also needs access to a polynomial root finder.

The one exception to this rule is the algorithm of Theorem 1.4. As mentioned at the end of Section 1.2, this is an iterative algorithm. At each step of the iteration we have to compute roots of polynomials (which may lie outside $\mathbb{K}$), and we keep computing with these roots in the subsequent iterations. For more details see Theorem 4.4 and the discussion after that theorem.

We also take some steps toward the analysis of our algorithms in the bit model of computation. We focus on the algorithm of Theorem 1.4 since it is the most difficult to analyze due to its iterative nature. We show in Proposition 4.5 that for polynomials with integer coefficients, this algorithm can be implemented in the bit model to run in time polynomial in the bit size of the *output*. We do not have a polynomial running time bound as a function of the input size (more on this in Section 1.4).

1.4 Future work

One could try to extend the results of this paper in several directions. For instance, one could try to handle "supersparse" polynomials like in the Sparsest Shift algorithm of [12]. We begin a study of the multivariate case in the full version of this paper [9], but these developments are completely omitted here due to space limits. In [9] we proceed by reduction to the univariate case, but one could try to design more "genuinely multivariate" algorithms. For Waring decomposition, such an algorithm is proposed in "case 2" of [19, Theorem 5]. Its analysis relies on a randomness assumption for the input f (our multivariate algorithm is randomized, but in this paper we never assume that the input polynomial is randomized).

One should also keep in mind, however, that the basic univariate problem studied in the present paper is far from completely solved: our algorithms all rely on some assumptions for the exponents e_i in a decomposition of f, and some algorithms also rely on a distinctness assumption for the shifts a_i. It would be very interesting to weaken these assumptions, or even to remove them entirely. With a view toward this question, one could first try to improve the lower bounds from [20]. Indeed, the same tools (Wronskians, shifted differential equations) turn out to be useful for the two problems (lower bounds and reconstruction algorithms) but the lower bound problem appears to be easier. For real polynomials we have already obtained optimal $\Omega(d)$ lower bounds in [8] using Birkhoff interpolation, but it remains to give an algorithmic application of this lower bound method.

Another issue that we have only begun to address is the analysis of the bit complexity of our algorithms. It would be straightforward to give a polynomial bit size bound for, e.g., the algorithm of Theorem 4.3 but this issue seems to be more subtle for Theorem 1.4 due to the iterative nature of our algorithm. It is in fact not clear that there exists a solution of size polynomially bounded in the input size (i.e., in the bit size of f given as a sum of monomials). More precisely, we ask the following question.

QUESTION 1.6. *We define the dense size of a polynomial $f = \sum_{i=0}^{d} f_i x^i \in \mathbb{Z}[x]$ as $\sum_{i=0}^{d}[1 + \log_2(1 + |f_i|)]$. Assume that f can be written as*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i}$$

with $a_i \in \mathbb{Z}$, $\alpha_i \in \mathbb{Z} \setminus \{0\}$, and that this decomposition satisfies the conditions of Theorem 1.4: the constants a_i are all distinct, and $n_i \leq (3i/4)^{1/3} - 1$ for all $i \geq 2$, where n_i denotes the number of indices j such that $e_j \leq i$.

Is it possible to bound the bit size of the constants α_i, a_i by a polynomial function of the dense size of f?

As explained at the end of Section 1.3, under the same conditions we have a decomposition algorithm that runs in time polynomial in the bit size of the *output*. It follows that the above question has a positive answer if and only if there is a decomposition algorithm that runs in time polynomial in the bit size of the input (i.e., in time polynomial in the dense size of f).

One could also ask similar questions in the case where the conditions of Theorem 1.4 do not hold. For instance, assuming that f has an optimal decomposition with integer coefficients, is there such a decomposition where the coefficients α_i, a_i are of size polynomial in the size of f?

2. PRELIMINARIES

In this section we present some tools that are useful for their algorithmic applications in Sections 4. Section 3 can be read independently, except for the proof of Proposition 3.1 which uses the Wronskian.

2.1 The Wronskian

In mathematics the *Wronskian* is a tool mainly used in the study of differential equations, where it can be used to show that a set of solutions is linearly independent.

DEFINITION 2.1 (WRONSKIAN). *For n univariate functions $f_1, \ldots, f_n$, which are $n-1$ times differentiable, the Wronskian $Wr(f_1, \ldots, f_n)$ is defined as*

$$Wr(f_1, \ldots, f_n)(x) = \begin{vmatrix} f_1(x) & f_2(x) & \ldots & f_n(x) \\ f_1'(x) & f_2'(x) & \ldots & f_n'(x) \\ \vdots & \vdots & \ddots & \vdots \\ f_1^{(n-1)} & f_2^{(n-1)} & \ldots & f_n^{(n-1)} \end{vmatrix}$$

It is a classical result, going back at least to [3], that the Wronskian captures the linear dependence of polynomials in $\mathbb{F}[x]$.

PROPOSITION 2.2. *For $f_1, \ldots, f_n \in \mathbb{F}[x]$, the polynomials are linearly dependent if and only if the Wronskian $Wr(f_1, \ldots, f_n)$ vanishes everywhere.*

For every $f \in \mathbb{F}[x]$ and every $a \in \mathbb{F}$ we denote by $\mathrm{M}_a(f)$ the multiplicity of a as a root of f, i.e., $\mathrm{M}_a(f)$ is the maximum $t \in \mathbb{N}$ such that $(x-a)^t$ divides f. The following result from [28] gives a Wronskian-based bound on the multiplicity of a root in a sum of polynomials.

[4]Once the a_i's have been determined, we also need to do some linear algebra computations with these nodes to determine the coefficients α_i.

LEMMA 2.3. *Let $f_1, \ldots, f_n$ be some linearly independent polynomials and $a \in \mathbb{F}$, and let $f(x) = \sum_{j=1}^n f_j(x)$. Then:*

$$\mathrm{M}_a(f) \leq n - 1 + \mathrm{M}_a(\mathit{Wr}(f_1, \ldots, f_n)),$$

where $\mathrm{M}_a(f)$ is finite since $\mathit{Wr}(f_1, \ldots, f_n) \not\equiv 0$.

In [25] one can find several properties concerning the Wronskian (and which have been known since the 19^{th} century). In this work we will use the following properties, which can be easily derived from those of [25].

PROPOSITION 2.4. *Let $f_1, \ldots, f_n \in \mathbb{F}[x]$ be linearly independent polynomials and let $a_1, \ldots, a_n \in \mathbb{F}$. If $f_j = (x-a_j)^{d_j} g_j$ for all j, then $Q := \prod_{d_j \geq n}(x-a_j)^{d_j-n+1}$ divides $\mathit{Wr}(f_1, \ldots, f_n)$. Moreover, if $Q(a) \neq 0$, then*

$$\mathrm{M}_a(\mathit{Wr}(f_1, \ldots, f_n)) \leq \sum_{j=1}^n \deg(g_j) + \frac{n(n-1)}{2}.$$

Hence, if we set $f := \sum_{j=1}^n f_j$, then

$$\mathrm{M}_a(f) \leq \sum_{j=1}^n \deg(g_j) + \frac{(n+2)(n-1)}{2}.$$

PROOF. Consider the $n \times n$ Wronskian matrix W whose $(i+1, j)$-th entry is $f_j^{(i)}(x)$ with $0 \leq i \leq n-1$, $1 \leq j \leq n$. Assume that $d_j \geq n$. Since $(x-a_j)^{d_j}$ divides f_j, then $f_j^{(i)} = (x-a_j)^{d_j-i} g_{i,j} = (x-a_j)^{d_j-n+1}(x-a_j)^{n-1-i} g_{i,j}$, for some $g_{i,j} \in \mathbb{F}[x]$ of degree $\deg(g_j)$. Since $(x-a_j)^{d_j-n+1}$ divides every element in the j-th column of W, we can factor it out from the Wronskian. This proves that Q divides $\mathrm{Wr}(f_1, \ldots, f_n)$. Once we have factored out $(x-a_j)^{d_j-n+1}$ for all $d_j \geq n$, we observe that $\mathrm{Wr}(f_1, \ldots, f_n) = Q(x)\,h(x)$, where $h(x)$ is the determinant of a matrix whose $(i+1, j)$-th entry has degree at most $\deg(g_j) + (n-i-1)$ for all $0 \leq i \leq n-1$ and $1 \leq j \leq n$. Hence, $\deg(h) \leq \sum_{j=1}^n [\deg(g_j) + (n-1)] - \binom{n}{2} = \sum_{j=1}^n \deg(g_j) + \frac{n(n-1)}{2}$. Finally, we observe that if $Q(a) \neq 0$:

$$\mathrm{M}_a(\mathrm{Wr}(f_1, \ldots, f_n)) = \mathrm{M}_a(Q) + \mathrm{M}_a(h) = \mathrm{M}_a(h) \leq \deg(h).$$

For $f = \sum_{j=1}^n f_j$, the upper bound for $\mathrm{M}_a(f)$ follows directly from Lemma 2.3. □

2.2 Shifted Differential Equations

DEFINITION 2.5. *A* Shifted Differential Equation *(SDE) is a differential equation of the form*

$$\sum_{i=0}^k P_i(x) f^{(i)}(x) = 0$$

where f is the unknown function and the P_i are polynomials in $\mathbb{F}[x]$ with $\deg(P_i) \leq i$.
The quantity k is called the order *of the equation. We will usually denote such a differential equation by SDE(k).*

One of the key ingredients for our results is that if AffPow(f) is small, then f satisfies a "small" SDE. More precisely:

PROPOSITION 2.6. *Let $f \in \mathbb{F}[x]$ be written as*

$$f = \sum_{i=1}^t \alpha_i (x-a_i)^{e_i}.$$

where $a_i \in \mathbb{F}$, $e_i \in \mathbb{N}$ for all i.
Then, f satisfies a SDE$(2t-1)$ which is also satisfied by the t terms $f_i(x) = (x-a_i)^{e_i}$.

PROOF. If we can find a SDE$(2t-1)$ which is satisfied by all the f_i, by linearity the same SDE is satisfied by f and the result follows. The existence of this common SDE is equivalent to the existence of a nonzero solution for the following linear system in the unknowns $a_{j,k}$:

$$\sum_{j,k} a_{j,k} x^j f_i^{(k)}(x) = 0,$$

where $1 \leq i \leq t$, $0 \leq k \leq 2t-1$ and $0 \leq j \leq k$. There are $1 + 2 + \cdots + 2t = (2t+1)t$ unknowns, so we need to show that the matrix of this linear system has rank smaller than $(2t+1)t$. It suffices to show that for each fixed value of $i \in \{1, \ldots, t\}$, the subsystem:

$$\sum_{j,k} a_{j,k} x^j f_i^{(k)}(x) = 0 \ (0 \leq k \leq 2t-1, 0 \leq j \leq k)$$

has a matrix of rank $< 2t+1$. In other words, we have to show that the subspace V_i spanned by the polynomials $x^j f_i^{(k)}(x)$ has dimension less than $2t+1$. But V_i is included in the subspace spanned by the polynomials

$$\{(x-a_i)^{e_i+j};\ -(2t-1) \leq j \leq 0, e_i + j \geq 0\}.$$

This is due to the fact that the polynomials x^j belongs to the span of the polynomials $\{(x-a_i)^\ell \mid 0 \leq \ell \leq j\}$,. We conclude that $\dim V_i \leq 2t < 2t+1$. □

REMARK 2.7. *A polynomial f satisfies a SDE(k) if and only if the polynomials $(x^j f^{(i)}(x))_{0 \leq i \leq k, 0 \leq j \leq i}$ are linearly dependent over $\mathbb{F}$. The existence of such a SDE can therefore be decided efficiently by linear algebra, and when a SDE(k) exists it can be found explicitly by solving the corresponding linear system (see, e.g., [27, Corollary 3.3a] for an analysis of linear system solving in the bit model of computation). Moreover, given a polynomial f, one can find a SDE(k) satisfied by f with smallest k in time polynomial in the degree of f. We use this fact repeatedly in the algorithms of Section 4.*

Notice that an SDE is a linear homogeneous differential equation, hence we can derive the following result from classic property about sets of solutions of linear differential equations.

LEMMA 2.8. *The set of polynomial solutions of a SDE of order k is a vector space of dimension at most k.*

3. STRUCTURAL RESULTS

In this section we study the uniqueness of optimal representations. It turns out that the algorithms of Section 4 only work in a regime where the uniqueness of optimal representations is guaranteed.

PROPOSITION 3.1. *Consider a polynomial identity of the form:*

$$\sum_{i=1}^k \alpha_i (x-a_i)^d = \sum_{i=1}^l \beta_i (x-b_i)^{e_i}$$

where the $a_i \in \mathbb{F}$ are distinct, the $\alpha_i \in \mathbb{F}$ are not all zero, $\beta_i, b_i \in \mathbb{F}$ are arbitrary, and $e_i < d$ for every i. Then we must have $k + l > \sqrt{2(d+1)}$.

One can find in [9, Theorem 3.1] a sharper version of this result for the field of real numbers.

PROOF. We assume $\alpha_1 \neq 0$ and we have the following equality:

$$\alpha_1(x-a_1)^d = -\sum_{i=2}^{k} \alpha_i(x-a_i)^d + \sum_{i=1}^{l} \beta_i(x-b_i)^{e_i}$$

Consider an independent subfamily on the right hand side of this equality. We obtain a new identity of the form: $g = \sum_{i=1}^{p} \lambda_i \ell_i^{r_i}$, with $\ell_i(x) = x - c_i$, $g(x) = \alpha_1(x-a_1)^d$, and $p \leq k+l-1$. Since $\deg(g) = d$ and $e_i < d$ for all i; then there exists i such that $r_i = d$. We assume without loss of generality that $\ell_1 = x - a_2$ and $r_1 = d$.

We take the derivatives of this equality to obtain the following system of equations for $j = 0 \ldots p-1$:

$$g^{(j)} = \sum_{i=1}^{p} \lambda_i \left[\ell_i^{r_i}\right]^{(j)}$$

Using Cramer's rule, we obtain:

$$\lambda_1 = \frac{\mathrm{Wr}(g, \ell_2^{r_2}, \ldots, \ell_p^{r_p})}{\mathrm{Wr}(\ell_1^{r_1}, \ell_2^{r_2}, \ldots, \ell_p^{r_p})}$$

We define $\Delta = \{i : 2 \leq i \leq p, r_i \geq p\}$ and, following Proposition 2.4, we factorise the Wronskians:

$$\lambda_1 = \frac{(x-a_1)^{d-(p-1)} \prod_{i\in\Delta} \ell_i^{r_i-(p-1)} \cdot W_1}{(x-a_2)^{d-(p-1)} \prod_{i\in\Delta} \ell_i^{r_i-(p-1)} \cdot W_2}$$

where W_1, W_2 are the remaining determinants.
After some simplifications, we obtain the following identity:

$$\lambda_1(x-a_2)^{d-(p-1)} W_2 = (x-a_1)^{d-(p-1)} W_1$$

Notice now that since we have factorised the large r_i's, the i^{th} row of W_1 and W_2 contains polynomials with degree bounded by $p-i$, thus $\deg W_1, \deg W_2 \leq p(p-1)/2$.

Moreover, since $a_1 \neq a_2$, we compute the multiplicity of a_1 on both sides of the identity and obtain that

$$\begin{aligned}\mathrm{M}_{a_1}\left((x-a_1)^{d-(p-1)} W_1\right) &= \mathrm{M}_{a_1}\left(\lambda_1(x-a_2)^{d-(p-1)} W_2\right) \\ &= \mathrm{M}_{a_1}(W_2) \leq \deg(W_2)\end{aligned}$$

The previous remark on the degree of W_2 therefore implies that $d-(p-1) \leq \frac{p(p-1)}{2}$. Finally, we set $s = l+k$ and we use the fact that $p \leq s-1$ to obtain the desired lower bound: $2(d+1) < s^2$. □

REMARK 3.2. *The order of this bound is tight when $\mathbb{F} = \mathbb{C}$, the field of complex numbers. Indeed, the following equality was exhibited in [8, Proposition 19]:*

$$\sum_{j=1}^{k} \xi^j (x+\xi^j)^d = \sum_{\substack{0\leq i\leq d \\ i\equiv -1 \pmod{k}}} k\binom{d}{i} x^{d-i}$$

where $k \in \mathbb{N}$ and $\xi \in \mathbb{C}$ is a k-th primitive root of unity. In particular, choosing $k = \sqrt{d}$ leads to an equality which has $2\sqrt{d}$ terms.

As a consequence of Proposition 3.1 we obtain that whenever $\mathrm{AffPow}_{\mathbb{F}}(f)$ is sufficiently small, the terms of highest degree in an optimal expression of f as $f = \sum_{i=1}^{s} \alpha_i(x-a_i)^{e_i}$ are uniquely determined.

COROLLARY 3.3. *Let $f \in \mathbb{F}[x]$ be a polynomial of the form :*

$$f = \sum_{i=1}^{k} \alpha_i(x-a_i)^d + \sum_{j=1}^{l} \beta_j(x-b_j)^{e_j}$$

with $e_j < d$. If $k+l \leq \sqrt{\frac{d+1}{2}}$, then the highest degree terms are unique. In other words, for every expression of f as

$$f = \sum_{i=1}^{k'} \alpha'_i(x-a'_i)^d + \sum_{j=1}^{l'} \beta'_j(x-b'_j)^{e'_j}$$

with $e'_j < d$ and $k'+l' \leq \sqrt{\frac{d+1}{2}}$, then $\{(\alpha_i, a_i)\} = \{(\alpha'_i, a'_i)\}$.

PROOF. Let us assume that we have another different decomposition for f:

$$f = \sum_{i=1}^{k'} \alpha'_i(x-a'_i)^d + \sum_{j=1}^{l'} \beta'_j(x-b'_j)^{e'_j}$$

with $k'+l' \leq \sqrt{(d+1)/2}$. Hence, we have the following equality:

$$\sum_{i=1}^{k}\alpha_i(x-a_i)^d - \sum_{i=1}^{k'}\alpha'_i(x-a'_i)^d = \sum_{j=1}^{l}\beta_j(x-b_j)^{e_j} - \sum_{j=1}^{l'}\beta'_j(x-b'_j)^{e'_j}$$

Since $k+k'+l+l' \leq \sqrt{2(d+1)}$, the result follows from Proposition 3.1. □

Finally, as a direct consequence of Corollary 3.3, we obtain a a sufficient condition for a polynomial to have a unique optimal expression in the AffPow model:

COROLLARY 3.4. *Let $f \in \mathbb{F}[x]$ be a polynomial of the form:*

$$f = \sum_{i=1}^{s} \alpha_i(x-a_i)^{e_i}$$

For every $e \in \mathbb{N}$ we denote by n_e the number of exponents smaller than e, i.e., $n_e = \#\{i : e_i \leq e\}$. If $n_e \leq \sqrt{\frac{e+1}{2}}$ for all $e \in \mathbb{N}$, then $\mathrm{AffPow}_{\mathbb{F}}(f) = s$ and the optimal representation of f is unique.

Another consequence of Proposition 3.1 is the following upper bound on the degree of the terms involved in an optimal expression of f in the model $\mathrm{AffPow}_{\mathbb{F}}$.

COROLLARY 3.5. *Let $f \in \mathbb{F}[x]$ be a polynomial of degree d written as*

$$f = \sum_{i=1}^{s} \alpha_i(x-a_i)^{e_i}$$

with $\alpha_i, a_i \in \mathbb{F}$, $e_i \in \mathbb{N}$ and $s = \mathrm{AffPow}_{\mathbb{F}}(f)$. We set $e := \max\{e_i : 1 \leq i \leq s\}$, then $e < d + \frac{s^2}{2}$ and, if $\mathbb{F} = \mathbb{R}$, then $e \leq d + 2s - 2$. In particular, we have that $e < d + \frac{(d+2)^2}{8}$ and, if $\mathbb{F} = \mathbb{R}$, then $e \leq 2d$.

PROOF. If $e = d$, then the result is trivial. Assume therefore that $e > d$. Now, we differentiate $d+1$ times the expression for f to obtain the identity:

$$0 = f^{(d+1)} = \sum_{e_i > d} \alpha_i \frac{e_i!}{(e_i-d-1)!}(x-a_i)^{e_i-d-1}.$$

By Proposition 3.1 we have $s > \sqrt{2(e-d)}$ and we conclude that $e < d + \frac{s^2}{2}$. When $\mathbb{F} = \mathbb{R}$, by Theorem [8, Theorem 13] we have $s \geq (e-d+2)/2$ and we conclude that $e \leq d+2s-2$. To finish the proof it suffices to recall that $s = \mathrm{AffPow}_{\mathbb{F}}(f) \leq \lceil (d+1)/2 \rceil \leq (d+2)/2$; see [8, Proposition 18]. □

REMARK 3.6. *On can find examples that are close to the bound of Corollary 3.5. Indeed, if we take $k = \sqrt{d}$ in Remark 3.2, we get an expression of the 0 polynomial with $2k$ terms. If we integrate this expression $7d$ times we get a polynomial f of degree $< 7d$ with $s := AffPow_{\mathbb{F}}(f) = 2k$ (by Corollary 3.4) and whose maximum exponent in the optimal expression is $8d = 7d + d = \deg(f) + (s^2/4)$.*

REMARK 3.7. *As a byproduct of Corollary 3.5, we obtain a naive brute force algorithm to find one optimal expression for any polynomial f. Indeed, for a fixed integer s, there are only a finite number of sequences of exponents $(e_1, \ldots, e_s)$ with $e_i \leq d + s^2/2$. For one sequence, one can try to find an expression with these exponents by solving a system of polynomial equations in $2s$ variables. The smallest s with a solution gives the value of $AffPow_{\mathbb{F}}(f)$.*

4. ALGORITHMS FOR DISTINCT NODES

The goal of this section is to provide algorithms that receive as input a polynomial f and computes $s = \mathrm{AffPow}_{\mathbb{F}}(f)$ and the triplets (α_i, a_i, e_i) for $i \in \{1, \ldots, s\}$ such that $f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i}$. We will not able to solve the problem in all its generality but under certain hypotheses. This section concerns the case where the nodes a_i in the optimal expression of f are all distinct (we study repeated nodes in [9, Section 5]). In this setting, our main result is Theorem 4.4 where we solve the problem when the number n_e of exponents in the optimal expression that are $\leq e$ is 'small'. A key point to obtain the algorithms is given by the following Proposition. Roughly speaking, this result says that if f satisfies a SDE, then every term in the optimal expression of f with exponent e_i big enough also satisfies the same SDE.

PROPOSITION 4.1. *Let $f \in \mathbb{F}[x]$ be written as*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i},$$

with $\alpha_i \in \mathbb{F}$ nonzero, the $a_i \in \mathbb{F}$ are all distinct, and $e_i \in \mathbb{N}$. Whenever f satisfies a SDE(k), then for all $e_i \geq ks + \binom{s}{2}$ we have that $(x - a_i)^{e_i}$ satisfies the same SDE.

PROOF. We assume that $e_1 \geq ks + \binom{s}{2}$ and that f satisfies the following SDE(k) in the unknown g:

$$\sum_{i=0}^{k} P_i(x)\, g^{(i)}(x) = 0,$$

with $\deg(P_i) \leq i$. By contradiction, we assume that $(x - a_1)^{e_1}$ does not satisfy this equation. For every $j \in \{1, \ldots, s\}$, we denote by f_j and R_j the polynomials such that

$$f_j = \sum_{i=0}^{k} P_i(x)\, ((x - a_j)^{e_j})^{(i)} = R_j(x)\, (x - a_j)^{d_j},$$

where $d_j := \max\{e_j - k, 0\}$. We observe that $\deg(f_j) \leq e_j$, so $\deg(R_j) \leq k$, and that $-f_1 = \sum_{j=2}^{s} f_j \neq 0$. We consider a linearly independent subfamily of $f_2, \ldots, f_s$, namely $\{f_j \mid j \in J\}$ with $J = \{j_1, \ldots, j_p\} \subseteq \{2, \ldots, s\}$. Then by Proposition 2.4 we have that

$$e_1 - k = d_1 \leq \mathrm{M}_{a_1}(f_1) \quad \leq \textstyle\sum_{j=1}^{p} \deg(R_j) + \frac{(p+2)(p-1)}{2}$$

Since $p \leq s - 1$, we get that $e_1 \leq k + k(s-1) + \frac{(s+1)(s-2)}{2} < ks + \binom{s}{2}$, a contradiction. $\square$

As a consequence of Proposition 4.1, we get Corollary 4.2 and Theorem 4.3. They provide an effective method to obtain the optimal expression of a polynomial f in the Affine Power model whenever all the terms involved have big exponents and all the nodes are different.

COROLLARY 4.2. *Let $f \in \mathbb{F}[x]$ be written as $f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i}$, with $\alpha_i \in \mathbb{F} \setminus \{0\}$, $a_i \in \mathbb{F}$ all distinct, and $e_i \geq 5s^2/2$ for all i. Then,*

a) $\{(x - a_i)^{e_i} \mid 1 \leq i \leq s\}$ are linearly independent,

b) If $f = \sum_{i=1}^{t} \beta_i (x - b_i)^{d_i}$ with $t \leq s$, then $t = s$ and we have the equality $\{(\alpha_i, a_i, e_i) \mid 1 \leq i \leq s\} = \{(\beta_i, b_i, d_i) \mid 1 \leq i \leq s\}$; in particular, $AffPow_{\mathbb{F}}(f) = s$,

c) f satisfies a SDE$(2s - 1)$,

d) if f satisfies a SDE(k) with $k \leq 2s - 1$ then $(x - a_i)^{e_i}$ also satisfies it for all $i \in \{1, \ldots, s\}$, and

e) f does not satisfy any SDE(k) with $k < s$.

PROOF. Notice first that (b) implies (a). Assume now that (b) does not hold, we can write f as $f = \sum_{i=1}^{t} \beta_i (x - b_i)^{d_i}$ with $t \leq s$. Hence, by Proposition 3.1, we get that

$$2s \geq t + s > \sqrt{2(\min(\{e_1, \ldots, e_s\}) + 1)} \geq \sqrt{5s^2},$$

a contradiction. From Proposition 2.6 we get (c). If f satisfies a SDE(k) with $k \leq 2s - 1$, then for all $i \in \{1, \ldots, s\}$ we have that

$$e_i \geq 5s^2/2 \geq (2s-1)s + \binom{s}{2} \geq ks + \binom{s}{2}.$$

Hence, Proposition 4.1 yields that $(x - a_i)^{e_i}$ is also a solution of this equation for all i, proving (d). Finally, f cannot satisfy a SDE(k) with $k < s$; otherwise by (a) and (d), the vector space of solutions to this equation has dimension $\geq s$, which contradicts Lemma 2.8. $\square$

THEOREM 4.3 (BIG EXPONENTS). *Let $f \in \mathbb{F}[x]$ be a polynomial that can be written as*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i},$$

where the constants $a_i \in \mathbb{F}$ are all distinct, $\alpha_i \in \mathbb{F} \setminus \{0\}$ and $e_i > 5s^2/2$. Then, $AffPow_{\mathbb{F}}(f) = s$. Moreover, there is a polynomial time algorithm `Build`(f) *that receives $f = \sum_{i=0}^{d} f_i x^i \in \mathbb{F}[x]$ as input and computes the s-tuples of coefficients $C(f) = (\alpha_1, \ldots, \alpha_s)$, of nodes $N(f) = (a_1, \ldots, a_s)$ and exponents $E(f) = (e_1, \ldots, e_s)$. The algorithm* `Build`$(f)$ *works as follows:*

Step 1. *Take r the minimum value such that f satisfies a SDE(r) and compute explicitly one of these SDE.*

Step 2. *Compute $B = \{(x - b_i)^{d_i} \mid 1 \leq i \leq r\}$, the set of all the solutions of the SDE of the form $(x - b)^e$ with $(r+1)^2/2 \leq e \leq \deg(f) + (r^2/2)$.*

Step 3. *Determine $\beta_1, \ldots, \beta_r$ such that $f = \sum_{i=1}^{r} \beta_i (x - b_i)^{d_i}$*

Step 4. *Set $I := \{i \mid \beta_i \neq 0\}$ and output the sets $C(f) = (\beta_i \mid i \in I)$, $N(f) = (b_i \mid i \in I)$ and $E(f) = (d_i \mid i \in I)$.*

PROOF. Corollary 4.2 proves the correctness of this algorithm. Indeed, by Corollary 4.2.(c) and (e), the value r computed in **Step 1** satisfies that $s \leq r \leq 2s - 1$. We claim that the set B computed in **Step 2** satisfies that:

(1) it contains the set $\{(x-a_i)^{e_i} \mid 1 \leq i \leq s)\}$,

(2) it has at most r elements, and

(3) all its elements are $\mathbb{F}$-linearly independent.

The first claim follows from Corollary 4.2.(d), the fact that $(r+1)^2/2 \leq (2s)^2/2 < 5s^2/2$, and from Corollary 3.5, since $e_i \leq \deg(f) + (s^2/2) \leq \deg(f) + (r^2/2)$ for all i. To prove the second claim assume that B has more than r elements, then we take $t_1, \ldots, t_{r+1} \in B$. To reach a contradiction, by Lemma 2.8 it suffices to prove that $t_1, \ldots, t_{r+1}$ are linearly independent. If this were not the case, by Proposition 3.1, we would have that $r+1 > \sqrt{(r+1)^2+2}$, which is not possible. A similar argument and the fact that B has at most r elements proves the third claim. By (1) and (3), the expression of f as a combination of the elements of B is unique and is the desired one.

Finally, the four steps can be perfomed in polynomial time. Only the first two steps require a justification. See Remark 2.7 in Section 2 regarding Step 1. In Step 2 we substitute for each value of e the polynomial $(x-b)^e$ in the SDE. This yields a polynomial $g(x)$ whose coefficients are polynomials in b of degree at most $r \leq 2s-1$. We are looking for the values of b which make g identically 0, so we find b as a root of the gcd of the coefficients of g. □

Now, we can proceed with the main result of this section:

THEOREM 4.4 (DIFFERENT NODES). *Let $f \in \mathbb{F}[x]$ be a polynomial that can be written as*

$$f = \sum_{i=1}^{s} \alpha_i (x-a_i)^{e_i},$$

where the constants $a_i \in \mathbb{F}$ are all distinct, $\alpha_i \in \mathbb{F} \setminus \{0\}$, and $e_i \in \mathbb{N}$. Assume moreover that $n_i \leq (3i/4)^{1/3} - 1$ for all $i \geq 2$, where n_i denotes the number of indices j such that $e_j \leq i$.

Then, AffPow$_{\mathbb{F}}(f) = s$. Moreover, there is a polynomial time algorithm `Build`(f) *that receives $f = \sum_{i=0}^{d} f_i x^i \in \mathbb{F}[x]$ as input and computes the s-tuples of coefficients $C(f) = (\alpha_1, \ldots, \alpha_s)$, of nodes $N(f) = (a_1, \ldots, a_s)$ and exponents $E(f) = (e_1, \ldots, e_s)$. The algorithm* `Build`$(f)$ *works as follows:*

Step 1. *We take t the minimum value such that f satisfies a SDE(t) and compute explicitly one of these SDE.*

Step 2. *Consider $B := \{(x-b_i)^{d_i} \mid 1 \leq i \leq l\}$, the set of all the solutions of the SDE of the form $(x-b)^e$ with $(t+1)^2/2 \leq e \leq \deg(f) + \frac{(\deg(f)+2)^2}{8}$ and assume that $d_1 \geq d_2 \geq \cdots \geq d_l \geq d_{l+1} := (t+1)^2/2$.*

Step 3. *We take $r \in \{1, \ldots, l\}$ such that $d_r - d_{r+1} > r^2/2$ and $d_{r+1} < \deg(f)$.*

Step 4. *We set $j := d_r - (r^2/2)$ and express $f^{(j)}$ as $f^{(j)} = \sum_{i=1}^{r} \beta_i \frac{d_i!}{(d_i-j)!}(x-b_i)^{d_i-j}$ with $\beta_1, \ldots, \beta_r \in \mathbb{F}$. We set $I := \{i \mid \beta_i \neq 0\}$.*

Step 5. *We set $\widetilde{f} := \sum_{i=1}^{r} \beta_i (x-b_i)^{d_i}$ and $h := f - \widetilde{f}$.*

If $h = 0$, then $C(f) = (\beta_i \mid i \in I)$, $N(f) = (b_i \mid i \in I)$ and $E(f) = (d_i \mid i \in I)$.

Otherwise, we set $h := f - \widetilde{f}$ and we have that $C(f) = (\beta_i \mid i \in I) \cup C(h)$, $N(f) = (b_i \mid i \in I) \cup N(h)$ and $E(f) = (d_i \mid i \in I) \cup E(h)$, where the triplet $(C(h), N(h), E(h))$ is the output of `Build`(h).

PROOF. By Corollary 3.4 we have that AffPow$_{\mathbb{F}}(f) = s$. Concerning the algorithm, first we observe that the value t computed in **Step 1** is $\leq 2s-1$ by Proposition 2.6. Moreover, we claim that the set B computed in **Step 2** has $l \leq t$ elements. Otherwise, by Lemma 2.8, there exists a set $I \subseteq \{1, \ldots, l\}$ of size $\leq t+1$ and there exist $\{\gamma_i \mid i \in I\} \subseteq \mathbb{F} \setminus \{0\}$ such that $\sum_{i \in I} \gamma_i (x-b_i)^{d_i} = 0$. Setting $m := \max\{d_i \mid i \in I\} \geq (t+1)^2/2$, Proposition 3.1 yields that $t+1 \geq |I| > \sqrt{2(m+1)} > t+1$, a contradiction.

Now we set $L := 5s^2/2$ and consider the set $C := \{(x-a_i)^{e_i} \mid e_i \geq L\}$ where the a_i's are the nodes in the optimal expression of f. We have that $C \neq \emptyset$; indeed, if we set $e_{\max} := \max\{e_i \mid 1 \leq i \leq s\}$, then $s = n_{e_{\max}} \leq (3e_{\max}/4)^{1/3} - 1$ and $L \leq 4(s+1)^3/3 \leq e_{\max}$.

Since

$$ts + \binom{s}{2} \leq (2s-1)s + \binom{s}{2} \leq 5s^2/2,$$

Proposition 4.1 yields that all the elements of C are solution of the SDE and, by Corollary 3.5 we know that $e_i \leq \deg(f) + \frac{(\deg(f)+2)^2}{8}$ for all $i \in \{1, \ldots, s\}$. Hence $C \subseteq B$. In particular, there exists a $\tau \in \{1, \ldots, l\}$ such that $d_1 \geq d_\tau = e_{\max} \geq \frac{4}{3}(s+1)^3$.

Now we take $k := \max\{i \mid d_i > L\}$ (we have that $1 \leq k \leq l \leq t \leq 2s-1$) and we are going to prove that

- there exists $r \in \{\tau, \ldots, k-1\}$ such that $d_r - d_{r+1} > r^2/2$, or
- $d_k - L > k^2/2$.

Indeed, if this is not the case, then we get the following contradiction:

$$\begin{array}{rcl} \frac{4s^3}{3} & \leq & \frac{4(s+1)^3}{3} - L \leq e_{\max} - L = d_\tau - L = \\ & = & \sum_{i=\tau}^{k-1}(d_i - d_{i+1}) + d_k - L \leq \frac{1}{2}\sum_{i=\tau}^{k} i^2 \leq \\ & \leq & \frac{1}{2}\sum_{i=1}^{k} i^2 = \frac{k(k+1)(2k+1)}{12} \leq \frac{(2s-1)2s(4s-1)}{12} < \frac{4s^3}{3}. \end{array}$$

We take $r \in \{1, \ldots, k-1\}$ such that $d_r - d_{r+1} > r^2/2$, or $r = k$ if such a value does not exist (and $d_k - L > k^2/2$). We claim that $e_{\max} \geq d_r$ if and only if $d_{r+1} < \deg(f)$ and, thus, the r described in **Step 3** always exists. If $d_{r+1} < \deg(f)$, since $\deg(f) \leq e$ and $C \subseteq B$, then $d_r \leq e_{\max}$ (since $e_{\max} = d_\tau$, it cannot be sandwiched between two consecutive elements d_r, d_{r+1} of this sequence).

Conversely, assume now that $e_{\max} \geq d_r$ and let us prove that $d_{r+1} < \deg(f)$. To prove this we first observe that setting $j := d_r - (r^2/2)$, then $f^{(j)}$ can be uniquely expressed as a linear combination of $B' := \{(x-b_i)^{d_i-j} \mid 1 \leq j \leq r\}$. Indeed, $f^{(j)} = \sum_{e_i \geq j} \alpha_i \frac{e_i!}{(e_i-j)!}(x-a_i)^{e_i-j}$ with $\alpha_i \neq 0$ and $(x-a_i)^{e_i-j} \in B'$ for all $e_i \geq j$, and if there is another way of expressing $f^{(j)}$ as a linear combination of B', then by Proposition 3.1 we get that $r > \sqrt{2(\min\{d_i \mid 1 \leq i \leq r\} - j + 1)} \geq \sqrt{r^2+2} > r$, a contradiction. So, if $d_{r+1} \geq \deg(f)$, then $f^{(j)} = 0$ and the only expression of $f^{(j)}$ as a linear combination of B' would be the one in which every coefficient is 0, a contradiction. Hence, the value r computed in **Step 3** exists.

We have seen that $f^{(j)}$ can be uniquely expressed as a linear combination of B' as $f^{(j)} = \sum_{e_i \geq j} \alpha_i \frac{e_i!}{(e_i-j)!}(x-a_i)^{e_i-j}$. Hence, in **Step 4**, one finds all the (α_i, a_i, e_i) such that $e_i \geq j$. In **Step 5**, either $h = 0$ and we have finished or $h = \sum_{e_i < j} \alpha_i (x-a_i)^{e_i}$ is written as a linear combination of strictly less than s terms and satisfies the hypotheses of the Theorem, so by induction we are done. □

We define the size of the set of triplets $\{(\alpha_i, a_i, e_i) \mid 1 \leq i \leq s\} \subset \mathbb{Z} \times \mathbb{Z} \times \mathbb{N}$ as $\sum_{i=1}^{s} [1 + \log_2(1 + |a_i|) + \log_2(1 + |\alpha_i|) + e_i]$. As mentioned in the introduction, it is not clear that the size of the output of the algorithm proposed in Theorem 4.4 is polynomially bounded in the input size (i.e., in the bit size of f given as a sum of monomials). However, since the exponents are encoded in unary, it is straightforward to check that the input size is polynomially bounded by the output size. Indeed, the degree of f is upper bounded by the maximum value of the e_i and every coefficient of f can be seen as the evaluation of a small polynomial in the α_i, a_i's. In the following result we prove that the algorithm works in polynomial time in the size of the output. Hence, a positive answer to Question 1.6 together with Corollary 3.5 would directly yield that the algorithm works in polynomial time (in the size of the input).

PROPOSITION 4.5. *Let $f \in \mathbb{Z}[x]$ be written as*

$$f = \sum_{i=1}^{s} \alpha_i (x - a_i)^{e_i}$$

with $a_i \in \mathbb{Z}$, $\alpha_i \in \mathbb{Z} \setminus \{0\}$, $e_i \in \mathbb{N}$ and assume that this decomposition satisfies the conditions of Theorem 4.4: the constants a_i are all distinct, and $n_i \leq (3i/4)^{1/3} - 1$ for all $i \geq 2$, where n_i denotes the number of indices j such that $e_j \leq i$.

Then, the algorithm in Theorem 4.4 works in polynomial time in the size of the output $\{(\alpha_i, a_i, e_i) \mid 1 \leq i \leq s\}$.

PROOF. We write $f = \sum_{j=0}^{d} f_j x^j$ with $f_j \in \mathbb{Z}$ and $d = \deg(f) \leq \max\{e_1, \ldots, e_s\}$. Since $f_j = \sum_{e_i \geq j} \alpha_i \binom{e_i}{j} a_i^{e_i - j}$ for all $j \in \{0, \ldots, d\}$, the size of f is polynomially bounded by the size of the output. To perform **Step 1** we follow Remark 2.7. We note that the coefficients of the polynomials appearing in the SDE are polynomially bounded by the size of f. In **Step 2** we have to compute the integral roots of polynomials of degree $t \leq s$ with integral coefficients, which can also be done in polynomial time (see, e.g., [24]). **Step 4** can also be performed in polynomial time by solving a linear system of equations (see, e.g., [27, Corollary 3.3a]) . The result follows from the fact that the polynomial h defined in **Step 5** can be written as $h = \sum_{j \in J} \alpha_j (x - a_j)^{e_j}$ for some set $J \subset \{1, \ldots, s\}$ of at most $s - 1$ elements. After the first iteration, the algorithm is therefore called recursively on polynomials h with an output size bounded by the output size of the original f. $\square$

Acknowledgments

Work supported by ANR project CompA (code ANR-13-BS02-0001-01). The first author was also partially supported by project MTM2016-78881-P. The reconstruction problem for sums of affine powers was suggested to one of us (P.K.) by Erich Kaltofen at a Dagstuhl workshop where P.K. gave a talk on lower bounds for this model. We thank the anonymous referees for several useful suggestions.

5. REFERENCES

[1] J. Alexander, A. Hirschowitz. Polynomial interpolation in several variables. *Journal of Algebraic Geometry*, 4(2):201–222, 1995.

[2] M. Ben-Or, P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proc. 20th annual ACM Symposium on Theory of Computing*. ACM, 1988.

[3] M. Bocher. The theory of linear dependence. *Annals of Mathematics*, 2(1/4): 81–96, 1900-1901.

[4] M. Boij, E. Carlini, A. V. Geramita. Monomials as sums of powers: the real binary case. *Proc. Amer. Math. Soc.* 139(9):3039–3043, 2011.

[5] A. Borodin, P. Tiwari. On the decidability of sparse univariate polynomial interpolation. *Computational Complexity*, 1(1):67-90, 1991.

[6] M. C. Brambilla, G. Ottaviani. On the Alexander –Hirschowitz theorem. *Journal of Pure and Applied Algebra*, 212(5):1229–1251, 2008.

[7] D. A. Cox, Galois theory. Second edition. Pure and Applied Mathematics (Hoboken). John Wiley & Sons, Inc., 2012.

[8] I. García-Marco, P. Koiran. Lower bounds by Birkhoff interpolation. *Journal of Complexity* 39:38-50, 2017.

[9] I. García-Marco, P. Koiran, T. Pecatte. Reconstruction Algorithms for Sums of Affine Powers, arXiv:1607.05420v2, 2016.

[10] M. Giesbrecht, E. Kaltofen, W.-S. Lee. Algorithms for computing sparsest shifts of polynomials in power, Chebyshev and Pochhammer bases. *International Symposium on Symbolic and Algebraic Computation* (ISSAC'2002) (Lille). *Journal of Symbolic Computation* 36(3-4):401–424, 2003.

[11] M. Giesbrecht, G. Labahn, W.-S. Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. *Journal of Symbolic Computation* 44(8):943–959, 2009.

[12] M. Giesbrecht, D. S. Roche. Interpolation of shifted-lacunary polynomials. *Computational Complexity* 19(3):333–354, 2010.

[13] J. H. Grace, A. Young. The algebra of invariants. Cambridge University Press, 1903.

[14] D. Grigoriev, M. Karpinski. A zero-test and an interpolation algorithm for the shifted sparse polynomials. In *Proc. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 10th International Symposium (AAECC-10)*. LNCS 673, pp. 162-169, Springer, 1993.

[15] D. Grigoriev, M. Karpinski. Computability of the additive complexity of algebraic circuits with root extracting. *Theoretical computer science* 157(1):91-99, 1996.

[16] D. Grigoriev, Y. Lakshman. Algorithms for computing sparse shifts for multivariate polynomials. *Applicable Algebra in Engineering, Communication and Computing* 11(1):43-67, 2000.

[17] A. Iarrobino, V. Kanev. Power sums, Gorenstein algebras, and determinantal loci. Appendix C by Iarrobino and Steven L. Kleiman. Lecture Notes in Mathematics, 1721. Springer-Verlag, Berlin, 1999.

[18] E. Kaltofen, B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation* 9(3):301-320, 1990.

[19] N. Kayal. Affine projections of polynomials. In *Proc. 44th annual ACM Symposium on Theory of Computing (STOC 2012)*, pp. 643-662. ACM, 2012.

[20] N. Kayal, P. Koiran, T. Pecatte, C. Saha. Lower bounds for sums of powers of low degree univariates. In *Proc. 42nd International Colloquium on Automata, Languages and Programming (ICALP 2015), part I*, LNCS 9134, pages 810–821. Springer, 2015. Available from http://perso.ens-lyon.fr/pascal.koiran.

[21] J. Kleppe. Representing a Homogenous Polynomial as a Sum of Powers of Linear Forms. *Thesis for the degree of Candidatus Scientiarum (University of Oslo)*, 1999. Available at http://folk.uio.no/johannkl/kleppe-master.pdf.

[22] Y. N. Lakshman, B. D. Saunders. Sparse shifts for univariate polynomials. *Appl. Algebra Engrg. Comm. Comput.* 7 (1996), no. 5, 351–364.

[23] J. M. Landsberg, Z. Teitler. On the ranks and border ranks of symmetric tensors. *Foundations of Computational Mathematics*, 10(3):339–366, 2010.

[24] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Math. Ann. 261 (1982), no. 4, 515–534.

[25] G. Pólya, G. Szegö. Problems and theorems in analysis. Vol. II. Theory of functions, zeros, polynomials, determinants, number theory, geometry. Revised and enlarged translation by C. E. Billigheimer of the fourth German edition. Springer Study Edition. Springer-Verlag, New York-Heidelberg, 1976. xi+391 pp.

[26] R. Saptharishi. A survey of lower bounds in arithmetic circuit complexity. github.com/dasarpmar/lowerbounds-survey/releases.

[27] A. Schrijver. Theory of linear and integer programming. John Wiley & Sons, 1986. xii+471 pp.

[28] M. Voorhoeve, A.J. Van Der Poorten. Wronskian determinants and the zeros of certain functions. Indagationes Mathematicae, 37 (1975), no. 5, 417–424.