Solving structured linear systems with large displacement rank

Alin Bostan, Claude-Pierre Jeannerod, Éric Schost

Solving structured linear systems with large displacement rank

Alin Bostan [a,1] Claude-Pierre Jeannerod [b] Éric Schost [c,2]

[a] *Algorithms Project, INRIA Paris-Rocquencourt, 78153 Le Chesnay Cedex, France*

[b] *Arénaire Project, INRIA Rhônes-Alpes, Laboratoire LIP (CNRS, ENSL, INRIA, UCBL), ENS Lyon, France*

[c] *Computer Science Department, The University of Western Ontario, London, Ontario, Canada*

Abstract

Linear systems with structures such as Toeplitz, Vandermonde or Cauchy-likeness can be solved in $O^{\sim}(\alpha^2 n)$ operations, where n is the matrix size, α is its displacement rank, and $O^{\sim}$ denotes the omission of logarithmic factors. We show that for such matrices, this cost can be reduced to $O^{\sim}(\alpha^{\omega-1} n)$, where ω is a feasible exponent for matrix multiplication over the base field. The best known estimate for ω is $\omega < 2.38$, resulting in costs of order $O^{\sim}(\alpha^{1.38} n)$. We present consequences for Hermite-Padé approximation and bivariate interpolation.

1 Introduction

Structured linear algebra techniques are a versatile set of tools. They enable one to deal at once with matrices with features such as Toeplitz, Vandermonde or Cauchy-likeness, which arise in various problems, from interpolation to reconstruction of rational or algebraic functions, etc.

Following [26], the usual way of measuring to what extent a matrix possesses one such structure is through its *displacement rank*, that is, the rank of its

Email addresses: Alin.Bostan@inria.fr (Alin Bostan), claude-pierre.jeannerod@ens-lyon.fr (Claude-Pierre Jeannerod), eschost@uwo.ca (Éric Schost).

[1] Supported by the French National Agency for Research (ANR Project "Gecko") and the Microsoft Research-INRIA Joint Centre

[2] Supported by NSERC (Canada) and by the Canada Research Chair Program

image through a suitable *displacement operator*. For P and Q in respectively $\mathbb{K}^{n\times n}$ and $\mathbb{K}^{m\times m}$, where $\mathbb{K}$ is our base field, we will use the displacement operator

$$\begin{aligned}\Delta[\mathsf{P},\mathsf{Q}] : \mathbb{K}^{n\times m} &\to \quad \mathbb{K}^{n\times m}\\ \mathsf{A} &\mapsto \mathsf{A} - \mathsf{P}\,\mathsf{A}\,\mathsf{Q}.\end{aligned}$$

Two matrices $(\mathsf{Y},\ \mathsf{Z})$ in $\mathbb{K}^{n\times\alpha}\times\mathbb{K}^{m\times\alpha}$ will be called a P,Q-*generator of length* α for A if $\Delta[\mathsf{P},\mathsf{Q}](\mathsf{A}) = \mathsf{Y}\,\mathsf{Z}^t$. The main idea behind algorithms for structured matrices is to use such generators as a compact data structure, in cases when $\Delta[\mathsf{P},\mathsf{Q}](\mathsf{A})$ has low rank. Even though these definitions hold for rectangular A, in most of this article, we have $n = m$.

Usual choices for P or Q are either diagonal matrices or cyclic down-shift matrices of size n, defined for φ in $\mathbb{K}$ by

$$\mathbb{Z}_{n,\varphi} = \begin{bmatrix} 0 & & & \varphi \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \in \mathbb{K}^{n\times n}.$$

The Toeplitz structure corresponds to $\mathsf{P} = \mathbb{Z}_{n,0}$ and $\mathsf{Q} = \mathbb{Z}^t_{m,0}$, so that $\Delta[\mathbb{Z}_{n,0},\mathbb{Z}^t_{m,0}](\mathsf{A})$ equals

$$\mathsf{A} - \ (\mathsf{A} \text{ shifted down and right by one unit}).$$

The Vandermonde structure is obtained by taking P diagonal and Q a cyclic right-shift matrix $\mathbb{Z}^t_{m,\varphi}$. For the Cauchy structure, both P and Q are diagonal. In particular, in all these cases, storing P and Q requires no more than $O(n)$ elements.

In this paper, we consider the following task:

$\mathsf{LinearSystem}(\mathsf{P},\mathsf{Q},\alpha)$: *Given a* P, Q*-generator of length* α *for a matrix* $\mathsf{A} \in \mathbb{K}^{n\times n}$*, with* $\alpha \le n$*, and given* $\mathsf{v} \in \mathbb{K}^n$*, find a solution to the equation* $\mathsf{Au} = \mathsf{v}$*, or determine that none exists.*

This problem makes sense only when the operator $\Delta[\mathsf{P},\mathsf{Q}]$ is invertible: this will be the case in our three cases of focus, Toeplitz-like, Vandermonde-like and Cauchy-like matrices (however, non-invertible operators can be dealt with if some extra information about A is known, such as suitable columns, see [46] and [45, Section 4.5]). Previous work then yielded the following kind of results: for these three structures, one can solve the problem $\mathsf{LinearSystem}$ using $O\tilde{}(\alpha^2 n)$ operations in $\mathbb{K}$, where the $O\tilde{}$ notation hides logarithmic factors.

When α is constant, such estimates are optimal up to logarithmic factors. However, there are several situations where α is not bounded *a priori* (see examples below). In the extreme case of very loosely structured matrices, when α goes up to $\alpha \simeq n$, the cost above becomes $O^{\sim}(n^3)$.

On the other side of the spectrum, we find *dense* linear algebra methods. Let $\omega \leq 3$ be such that $n \times n$ matrices over $\mathbb{K}$ can be multiplied in $O(n^\omega)$ operations (the current record estimate is $\omega < 2.38$ [13]). As in many other references, we will assume that $\omega > 2$ (in a potential situation with $\omega = 2$, logarithmic factors would appear in some estimates). Linear systems of size n can then be solved in time $O(n^\omega)$, using for example LSP factorization [25]; with $\omega < 3$, this is better than the above $O^{\sim}(n^3)$ estimate.

Our contribution bridges a gap between the approaches of structured and dense linear algebra by providing algorithms of cost $O^{\sim}(\alpha^{\omega-1}n)$ in the case of Toeplitz-like, Vandermonde-like and Cauchy-like matrices.

Model of computation. Our underlying computational model is the *algebraic RAM* over the field $\mathbb{K}$; a complete definition is given by Kaltofen in [27]. Concretely, the cost estimates count two kinds of operations:

- Operations in $\mathbb{K}$ (sums, products, equality tests, inversions) have unit cost; the generation of a random element in $\mathbb{K}$ has unit cost as well.
- We use integer arithmetic for handling indices in arrays. With a view towards simplicity, we see all these operations at unit cost.

Regarding the last point, in all our algorithms, the indices we will work with will be polynomial in n (when dealing with matrices of size n); hence, if desired, the translation to actual binary cost estimates would only induce a polylogarithmic overhead in n.

Our algorithms rely on *polynomial multiplication*; we will thus denote by $\mathsf{M} : \mathbb{N}_{>0} \to \mathbb{R}_{>0}$ a function such that polynomials in $\mathbb{K}[x]$ of degree less than d can be multiplied using at most $\mathsf{M}(d)$ operations. Using Fast Fourier Transform algorithms, $\mathsf{M}(d)$ can be taken in $O(d \log(d) \log\log(d))$, see Section 2 for more details.

Our algorithms are *probabilistic*. For simplicity, we say that an algorithm has type $P(s, d)$ if it chooses $r \leq s$ elements in $\mathbb{K}$ and, if these elements are chosen uniformly at random in a finite subset S of $\mathbb{K}$, the probability of success is at least $1 - d/|S|$. In particular, $\mathbb{K}$ should have cardinality more than d to guarantee that this probability can be made positive.

Main results. The first result covers matrices having Toeplitz-like structure, with $\mathsf{P} = \mathbb{Z}_{n,0}$ and $\mathsf{Q} = \mathbb{Z}^t_{n,0}$. We obtain a complexity in $O^{\sim}(\alpha^{\omega-1}n) \subset O^{\sim}(\alpha^{1.38}n)$, to be compared with an optimal cost of $O(\alpha\, n)$. For α constant,

our result is quasi-linear in n; when $\alpha \simeq n$, we recover the $O(n^\omega)$ behaviour of dense methods, up to logarithmic factors.

Theorem 1 *The problem* $\mathsf{LinearSystem}(\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^t, \alpha)$ *can be solved in time* $O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$*, by a probabilistic algorithm of type* $P(3n-2, n^2+n)$*.*

A fundamental application is the solution of approximation problems: given a *master polynomial* M and polynomials $f_1, \ldots, f_s$, one seeks a combination of the f_i, with polynomial coefficients of prescribed degrees, which vanishes modulo M. This includes in particular Padé and Hermite-Padé approximation (taking $M = x^n$), with applications to e.g. recovering the minimal polynomial of an algebraic power series f (taking $f_i = f^{i-1}$).

Corollary 1 *Let* $M \in \mathbb{K}[x]$ *be of degree* n*, let* $f_1, \ldots, f_s \in \mathbb{K}[x]$ *be of degrees less than* n*, and let* $\nu_1, \ldots, \nu_s \in \mathbb{N}$ *be such that* $\sum_{i \le s} \nu_i = n+1$*. One can find* $g_1, \ldots, g_s \in \mathbb{K}[x]$*, not all zero, of respective degrees less than* $\nu_1, \ldots, \nu_s$*, such that*

$$g_1 f_1 + \cdots + g_s f_s = 0 \bmod M,$$

in time $O(s^{\omega-1}\mathsf{M}(n)\log^2(n))$*. The algorithm is probabilistic of type* $P(3n-2, n^2+n)$*.*

Our second result addresses the Vandermonde-like case, where $\mathsf{P} = \mathbb{D}(\mathsf{x})$ is the diagonal matrix defined by a vector $\mathsf{x} = [x_1, \ldots, x_n]^t \in \mathbb{K}^n$ and Q is $\mathbb{Z}_{n,\psi}^t$. We assume that x has the form

$$\mathsf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \vdots \\ \mathsf{x}_s \end{bmatrix}, \text{ with } \mathsf{x}_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,\nu_i} \end{bmatrix} \in \mathbb{K}^{\nu_i}, \tag{1}$$

and with the following conditions: for all j, all entries of x_j are pairwise distinct; for $j < s$, x_{j+1} is a *prefix* of x_j, in the sense that $0 < \nu_{j+1} \le \nu_j$ and $x_{j+1,k} = x_{j,k}$ for $k = 1, \ldots, \nu_{j+1}$. We will call s the *multiplicity* of x (see Section 2). Note also that, given x as in Equation (1), we can deduce $\nu_1, \ldots, \nu_s$ in time $O(n)$.

Assuming that Equation (1) is satisfied is a mild assumption: any vector x can be put into this form after permuting its entries. However, finding the permutation has a cost: this is particularly easy if there is an order on $\mathbb{K}$, using sorting algorithms; without this assumption, though, the question seems harder: we mention an algorithm of complexity $O(\mathsf{M}(n)\log^3(n))$ in appendix. Hence, we rather stick to the above assumption for simplicity.

We make a second assumption, which is necessary to ensure that our problem is well-posed:

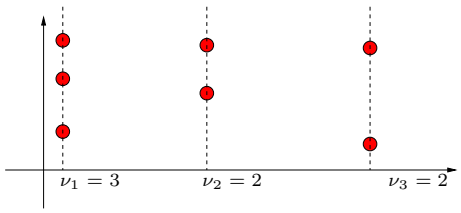
$$\psi\, x_i^n \neq 1 \quad \text{for} \quad i \le n. \tag{2}$$

Theorem 2 *Given* x *and* ψ *as in Equations* (1) *and* (2)*, one can solve the problem* $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}, \alpha)$ *in time* $O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$ *by a probabilistic algorithm of type* $P(4n-2, 4n^2+n)$.

An application of the latter theorem is polynomial interpolation. The approach applies to any number of variables, but we discuss only the bivariate case for simplicity. Consider n pairwise distinct interpolation points in $\mathbb{K}^2$; without loss of generality, we assume that they are given through the following lists, where the points with the same abscissas are grouped together:

$$\begin{aligned} P_1 &= [\,p_{1,1} = (x_1, y_{1,1}), \quad \ldots, \quad p_{1,\nu_1} = (x_1, y_{1,\nu_1})\,], \\ &\vdots \\ P_s &= [\,p_{s,1} = (x_s, y_{s,1}), \quad \ldots, \quad p_{s,\nu_s} = (x_s, y_{s,\nu_s})\,], \end{aligned} \tag{3}$$

with $\nu_1 \geq \cdots \geq \nu_s > 0$ and $n = \nu_1 + \cdots + \nu_s$. The following figure illustrates the case $s = 3$, $n = 7$ and $(\nu_1, \nu_2, \nu_3) = (3, 2, 2)$.



In general, it is difficult to state *a priori* that a multivariate interpolation problem is well-defined. Here, however, given values $[v_{i,j}] \in \mathbb{K}^n$, with $1 \leq i \leq s$ and $1 \leq j \leq \nu_i$, Theorem 1 in [35] (see also [19]) implies that there exists a unique $F \in \mathbb{K}[x, y]$ of the form

$$F = \sum_{0 \leq i < s,\ 0 \leq j < \nu_{i+1}} f_{i,j} x^i y^j$$

such that $F(p_{i,j}) = v_{i,j}$ for all i, j (hence the monomial support of the polynomial F corresponds to the index set of the sample points). In our previous example, F thus has the form

$$F = f_{0,0} + f_{0,1}y + f_{0,2}y^2 + f_{1,0}x + f_{1,1}xy + f_{2,0}x^2 + f_{2,1}x^2y.$$

Finding the coefficients $f_{i,j}$ of F is a linear problem with Vandermonde-like structure; in Section 5.4 we will prove the following corollary of Theorem 2.

Corollary 2 *Given the lists of points* $P_1, \ldots, P_s$ *and the lists of values* $[F(p) : p \in P_1], \ldots, [F(p) : p \in P_s]$*, the coefficients* $f_{i,j}$ *of* F *can be computed in time* $O(\nu_1^{\omega-1}\mathsf{M}(n)\log^2(n))$*. The algorithm is probabilistic of type* $P(4n-2, 4n^2+n)$.

Suppose for instance that $\nu_1 = s, \nu_2 = s-1, \ldots, \nu_s = 1$, so that we are interpolating on the simplex of monomials of degree less than s; here, $n = s(s+1)/2$. Then, our algorithm has subquadratic complexity $O^{\sim}(n^{(\omega+1)/2}) \subset O^{\sim}(n^{1.69})$.

Our third result deals with the Cauchy-like case, where $\mathsf{P} = \mathbb{D}(\mathsf{x})$ and $\mathsf{Q} = \mathbb{D}(\mathsf{y})$ are diagonal matrices defined by some vectors x and y in $\mathbb{K}^n$. We assume as before that x satisfies (1) and, in addition, make a similar assumption on y:

$$\mathsf{y} = \begin{bmatrix} \mathsf{y}_1 \\ \vdots \\ \mathsf{y}_t \end{bmatrix}, \text{ with } \mathsf{y}_i = \begin{bmatrix} y_{i,1} \\ \vdots \\ y_{i,\delta_i} \end{bmatrix} \in \mathbb{K}^{\delta_i}, \tag{4}$$

and with the following conditions: for all k, all entries of y_k are pairwise distinct; for $k < t$, y_{k+1} is a *prefix* of y_k. Again, given x as in Equation (1) and y as in Equation (4), one can deduce all the ν_j and δ_k in time $O(n)$.

For our Cauchy-like problem to be well-posed, we now replace the assumption in Equation (2) with the following one:

$$x_i y_j \neq 1 \text{ for } 1 \leq i, j \leq n. \tag{5}$$

The complexity result we obtain in this case is then essentially the same as for the Toeplitz-like and Vandermonde-like cases:

Theorem 3 *Given* x *and* y *as in Equations* (1), (4) *and* (5), *one can solve the problem* $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{x}), \mathbb{D}(\mathsf{y}), \alpha)$ *in time* $O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$, *by a probabilistic algorithm of type* $P(5n-2, 7n^2+n)$.

Previous work on structured systems. The notions of displacement rank and displacement operator originate from the work of Kailath, Kung and Morf [26]. Since then, the literature has vastly developed; see [45] for a list of references.

The basis for Theorem 1 is the seminal algorithm of Bitmead and Anderson [7] and Morf [37,38], which nontrivially adapts Strassen's divide-and-conquer approach [53] to structured computations. This algorithm requires several invertibility conditions to hold. Kaltofen [28,29] extended this idea to arbitrary matrices (see also [6, p. 204] for some related ideas), obtaining a cost of $O(\alpha^2\,\mathsf{M}(n)\log(n))$; for small α, this is better than our result in Theorem 1 by a factor of $\log(n)$.

We follow these previous approaches; our main technical contribution is an algorithm for the fast multiplication of a Toeplitz-like matrix (given by a generator of length α) by α vectors: we replace the separate computation of

these products (which has cost quadratic in α) by simultaneous computations, where using polynomial matrix multiplication yields a subquadratic cost in α.

To prove Theorem 2, we transform a Vandermonde-like system into a Toeplitz-like one, following Pan's idea [41,42]. We use a transformation from [24], generalizing it by taking into account the possibility of repetitions in the diagonal component of the operator. Similarly, the proof of Theorem 3 uses a reduction from Cauchy-like systems to Vandermonde-like ones.

In both the Vandermonde and Cauchy cases, a direct approach (not relying on the reduction to Toeplitz systems) is possible, see [48,44,47,12] and [45, Chapter 5]. As in the Toeplitz case, the bottleneck of these algorithms is the multiplication of a structured matrix (given by a generator of length α) by α vectors. However, following this approach, we are not able to obtain algorithms whose costs would match in all cases those of Theorems 2 and 3.

Applications. An important example of Toeplitz-like system solving is the approximation problem of Corollary 1. In the particular case of Hermite-Padé approximation, with $M = x^n$, a central reference is Beckermann-Labahn's algorithm [2], that has complexity $O(s^\omega \mathsf{M}(n)\log(n))$ for computing a σ-basis of order n of the input system (and thus a solution to the approximation problem); see [23]. In generic cases, an unpublished result of Lecerf reduces the cost to $O(s^{\omega-1}\mathsf{M}(n)\log n)$ and Storjohann [52] subsequently obtained a deterministic algorithm of similar complexity, applying in all cases. However, to our knowledge, these results do not extend to an arbitrary choice of M. Following notably [1,54], Beckermann and Labahn study that general case in [3] under the angle of fraction-free algorithms, with however a complexity more than linear in n.

Another example of a Toeplitz-like system that occurs frequently is when the matrix is block-Toeplitz, a block-size equal to α giving a displacement rank in $O(\alpha)$. Although Theorem 1 applies to any such system, a deterministic cost of $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$ can be obtained in the particular case where the matrix is invertible. As described for example in [16,17], this cost follows from combining an inversion formula of [33] with σ-basis computations as in [23].

Multivariate polynomial interpolation has been studied extensively (see [20] for a survey and [4,10,56] for algorithms relevant to sparse techniques). However, to our knowledge, previous references either do not cover the problems we deal with, or have higher complexity (typically, quadratic). Regarding the converse evaluation problem, let us mention the subquadratic complexity result of [40], which however deals with more general situations than ours.

Practical issues. Few practical algorithms are currently known for matrix multiplication with complexity better than cubic (see [53,34] and [32] for an exponent 2.776). However, even when using algorithms of cubic complexity,

the re-introduction of dense matrix arithmetic in our algorithms means that we can rely on extremely optimized implementations of matrix multiplication, such as the ones relying on BLAS libraries for finite field arithmetic [15].

Hence, besides theoretical estimates, the re-introduction of dense matrix multiplication in algorithms for structured matrices may lead to practical improvements. However, extra work may be required: useful refinements would for instance consist in removing the undesired logarithmic factors that appear in Theorems 1, 2 and 3.

Organization of the paper. After introducing basic notation and results in Section 2, we present in Section 3 the bases of our technical improvement, which can be stated in terms of polynomial operations only. These results are then applied, first to the Toeplitz case in Section 4, then to the Vandermonde case in Section 5 and the Cauchy case in Section 6. The appendix presents an algorithm for putting an arbitrary vector into the form of Equation (1), without relying on sorting algorithms.

Acknowledgments and notes. We thank E. Kaltofen, G. Labahn, M. Morf, V. Y. Pan, B. Salvy, A. Storjohann and G. Villard for useful discussions and comments. An earlier version of this article (lacking in particular the discussion of the Cauchy case) was published in [8].

2 Notation and preliminaries

We gather here all needed notation for vectors and matrices as well as some basic identities and complexity results on structured and dense matrices.

Matrices, vectors and polynomials. In what follows, we consider matrices and vectors over a field $\mathbb{K}$; vectors in $\mathbb{K}^n$ are identified with column-matrices in $\mathbb{K}^{n\times 1}$.

Matrices (resp. vectors) are written in upper-case (resp. lower-case) sans-serif font. If A (resp. B, C, ...) is a matrix, a_i (resp. b_i, c_i, ...) is its ith column. If x (resp. y, z, ...) is a vector, its ith entry is written x_i (resp. $y_i, z_i, \ldots$). Special matrices (diagonal, Vandermonde, Toeplitz, ...) will be written with blackboard bold letters ($\mathbb{D}$, $\mathbb{V}$, ...).

For n a positive integer and $i \leq n$, we let $\mathsf{e}_{n,i}$ be the ith unit vector in $\mathbb{K}^n$, so that $\mathsf{e}_{n,i}$ is zero, except at the ith entry, which is 1. The identity matrix in $\mathbb{K}^{n\times n}$, whose ith column equals $\mathsf{e}_{n,i}$, will be written $\mathbb{I}_n$.

Regarding univariate polynomials, we use the following notation:

- The degree of $F \in \mathbb{K}[x]$ is written $\deg(F)$.
- For $n \in \mathbb{N}$, we let $\mathbb{K}[x]_n$ be the n-dimensional vector space of polynomials of degree less than n.
- For $n \in \mathbb{N}$ and $F = f_0 + \cdots + f_{n-1}x^{n-1} \in \mathbb{K}[x]_n$, we will write $\mathsf{Rev}_n(F)$ for the reversal $f_{n-1} + \cdots + f_0 x^{n-1}$ of F.
- For $F \in \mathbb{K}[x]$ and $G \in \mathbb{K}[x]$ nonzero, F div G and F mod G are the quotient and the remainder in the division of F by G.

  Notation such as AB mod C or AB div C must be interpreted as (AB) mod C or (AB) div C, respectively.

We say that a vector is *repetition-free* when its entries are pairwise distinct. Besides, we use the following notation for vectors and polynomials derived from a vector $\mathsf{a} = [a_1, \ldots, a_n]^t$ in $\mathbb{K}^n$:

- $\mathsf{Flip}(\mathsf{a})$ is the vector $[0, a_n, \ldots, a_2]^t \in \mathbb{K}^n$.
- $\mathsf{Pol}(\mathsf{a})$ is the polynomial $\sum_{i=0}^{n-1} a_{i+1}x^i \in \mathbb{K}[x]_n$.
- If F is a function on $\mathbb{K}$, then $F(\mathsf{a})$ is the vector $[F(a_1), \ldots, F(a_n)]^t \in \mathbb{K}^n$.
- Conversely, if a is repetition-free and y is in $\mathbb{K}^n$, then $\mathsf{Interp}(\mathsf{a}, \mathsf{y})$ is the unique polynomial F in $\mathbb{K}[x]_n$ such that $y_i = F(a_i)$ for $1 \le i \le n$.

Families of structured matrices. Several families of structured matrices will be used along this paper. First, we associate matrices to a vector $\mathsf{x} = [x_1, \ldots, x_n]^t$ in $\mathbb{K}^n$:

- $\mathbb{D}(\mathsf{x}) \in \mathbb{K}^{n \times n}$ is the diagonal matrix whose ith diagonal entry equals x_i.
- $\mathbb{L}(\mathsf{x})$ is the lower-triangular Toeplitz matrix with first column x:

$$\mathbb{L}(\mathsf{x}) = \begin{bmatrix} x_1 & & & \\ x_2 & x_1 & & \\ \vdots & \vdots & \ddots & \\ x_n & x_{n-1} & \cdots & x_1 \end{bmatrix} \in \mathbb{K}^{n \times n}.$$

- $\mathbb{U}(\mathsf{x}) = \mathbb{L}(\mathsf{x})^t$ is the upper-triangular Toeplitz matrix with first row x^t:

$$\mathbb{U}(\mathsf{x}) = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ & x_1 & \cdots & x_{n-1} \\ & & \ddots & \vdots \\ & & & x_1 \end{bmatrix} \in \mathbb{K}^{n \times n}.$$

- For φ in $\mathbb{K}$, we denote by $\mathbb{T}(\mathsf{x}, \varphi)$ the φ-circulant matrix with first column x; that is, $\mathbb{T}(\mathsf{x}, \varphi) = \mathbb{L}(\mathsf{x}) + \varphi\, \mathbb{U}(\mathsf{Flip}(\mathsf{x}))$:

$$\mathbb{T}(\mathsf{x}, \varphi) = \begin{bmatrix} x_1 & \varphi\, x_n & \cdots & \varphi\, x_2 \\ x_2 & x_1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \varphi\, x_n \\ x_n & x_{n-1} & \cdots & x_1 \end{bmatrix} \in \mathbb{K}^{n \times n}.$$

- For $m \in \mathbb{N}$, the $n \times m$ Vandermonde matrix $\mathbb{V}(\mathsf{x}, m)$ is given by:

$$\mathbb{V}(\mathsf{x}, m) = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{m-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^{m-1} \end{bmatrix} \in \mathbb{K}^{n \times m}.$$

- Given $\mathsf{y} = [y_1, \ldots, y_m]^t$ in $\mathbb{K}^m$ such that $x_i y_j \neq 1$ for all i, j, the $n \times m$ Cauchy matrix $\mathbb{C}(\mathsf{x}, \mathsf{y})$ is defined by

$$\mathbb{C}(\mathsf{x}, \mathsf{y}) = \begin{bmatrix} \frac{1}{1-x_1 y_1} & \cdots & \frac{1}{1-x_1 y_m} \\ \vdots & & \vdots \\ \frac{1}{1-x_n y_1} & \cdots & \frac{1}{1-x_n y_m} \end{bmatrix} \in \mathbb{K}^{n \times m}.$$

We also associate families of matrices to univariate polynomials:

- Given $F = f_0 + \cdots + f_{n-1} x^{n-1} \in \mathbb{K}[x]_n$ and $m \in \mathbb{N}$, we denote by $\mathbb{M}(F, m)$ the matrix of the map $\mathbb{K}[x]_m \to \mathbb{K}[x]_{n+m-1}$ of multiplication by F:

$$\mathbb{M}(F, m) = \begin{bmatrix} f_0 & & \\ \vdots & \ddots & \\ \vdots & \ddots & f_0 \\ f_{n-1} & \ddots & \vdots \\ & \ddots & \vdots \\ & & f_{n-1} \end{bmatrix} \in \mathbb{K}^{(n+m-1) \times m}.$$

- Given the monic polynomial $F = f_0 + \cdots + f_{n-1} x^{n-1} + x^n$, we denote by

$\mathbb{X}(F)$ the matrix of multiplication by x in $\mathbb{K}[x]/\langle F\rangle$:

$$\mathbb{X}(F) = \begin{bmatrix} & & & -f_0 \\ 1 & & & -f_1 \\ & \ddots & & \vdots \\ & & 1 & -f_{n-1} \end{bmatrix} \in \mathbb{K}^{n\times n}.$$

Structured matrix identities. The families of structured matrices shown above satisfy many identities. We present now the ones we will need below; we start by showing how to rewrite products (diagonal matrix)×(Vandermonde matrix) using polynomial multiplication matrices.

Lemma 1 *Let* x *be repetition-free in* $\mathbb{K}^n$*, let* y *be in* $\mathbb{K}^n$*, and let* $P \in \mathbb{K}[x]_n$ *be defined by* $P = \mathsf{Interp}(\mathsf{x}, \mathsf{y})$*. Then, for* m *in* $\mathbb{N}$*, we have*

$$\mathbb{D}(\mathsf{y})\,\mathbb{V}(\mathsf{x}, m) = \mathbb{V}(\mathsf{x}, n+m-1)\,\mathbb{M}(P, m).$$

Proof. Let f be in $\mathbb{K}^m$ and let $F = \mathsf{Pol}(\mathsf{f}) \in \mathbb{K}[x]_m$. Then, both vectors $\mathbb{D}(\mathsf{y})\,\mathbb{V}(\mathsf{x}, m)\,\mathsf{f}$ and $\mathbb{V}(\mathsf{x}, n+m-1)\,\mathbb{M}(P, m)\,\mathsf{f}$ have entries $y_i F(x_i)$. □

The above identity is used in Subsections 5.1 and 6.1 to solve multiplication problems associated to, respectively, Vandermonde and Cauchy-like matrices. In fact, for Cauchy-like matrices, we will use it in combination with another identity, shown in Lemma 2 below: it factors $n \times m$ Cauchy matrices by means of Vandermonde and transposed Vandermonde matrices; when $n = m$, this is [18, Proposition 3.2] (see also [45, Exercise 3.10(b)]).

Lemma 2 *Let* x *in* $\mathbb{K}^n$ *and* y *in* $\mathbb{K}^m$ *be such that* $x_i y_j \neq 1$ *for all* i, j*. Let* $F \in \mathbb{K}[x]_{m+1}$ *be given by* $F = \prod_{j\leq m}(1 - xy_j)$ *and let* f *in* $\mathbb{K}^m$ *be such that* $\mathsf{Pol}(\mathsf{f}) = F \bmod x^m$*. Then*

$$\mathbb{C}(\mathsf{x}, \mathsf{y}) = \mathbb{D}(F(\mathsf{x}))^{-1}\,\mathbb{V}(\mathsf{x}, m)\,\mathbb{L}(\mathsf{f})\,\mathbb{V}(\mathsf{y}, m)^t.$$

Proof. Since $F(x_i) \neq 0$ for all i, we have $(1 - x_i y_j)^{-1} = F(x_i)^{-1} G_j(x_i)$, with $G_j = F/(1 - xy_j)$. Therefore, $\mathbb{C}(\mathsf{x}, \mathsf{y}) = \mathbb{D}(F(\mathsf{x}))^{-1}\,[G_1(\mathsf{x}) \ \cdots \ G_m(\mathsf{x})]$. Now, $G_j \in \mathbb{K}[x]_m$ and thus

$$G_j = (F \bmod x^m)(1 + y_j x + \cdots + y_j^{m-1} x^{m-1}) \bmod x^m.$$

Writing g_j for the vector in $\mathbb{K}^m$ such that $\mathsf{Pol}(\mathsf{g}_j) = G_j$, we obtain

$$G_j(\mathsf{x}) = \mathbb{V}(\mathsf{x}, m)\,\mathsf{g}_j = \mathbb{V}(\mathsf{x}, m)\,\mathbb{L}(\mathsf{f})\begin{bmatrix}1 & y_j & \cdots & y_j^{m-1}\end{bmatrix}^t$$

for $1 \leq j \leq m$, which concludes the proof. $\square$

Our final expression, already seen e.g. in [24], shows that some companion matrices and diagonal matrices are similar. We will use it in Subsection 5.2 for reducing Vandermonde-like systems to Toeplitz-like ones, and then again in Subsection 6.2 for reducing Cauchy-like systems to Vandermonde-like ones.

Lemma 3 *Let* x *be a repetition-free vector in* $\mathbb{K}^n$ *and let* F *be the monic polynomial defined by* $F = \prod_{i \leq n}(x - x_i)$*. Then we have the equality*

$$\mathbb{D}(\mathsf{x}) = \mathbb{V}(\mathsf{x}, n)\, \mathbb{X}(F)\, \mathbb{V}(\mathsf{x}, n)^{-1}.$$

Proof. For y in $\mathbb{K}^n$, we claim that the vectors $\mathbb{D}(\mathsf{x})\mathsf{y}$ and $\mathbb{V}(\mathsf{x}, n)\mathbb{X}(\mathbb{F})\mathbb{V}(\mathsf{x}, n)^{-1}\mathsf{y}$ have entries $x_i y_i$. This is readily seen for the former one. Defining $P = \mathsf{Interp}(\mathsf{x}, \mathsf{y}) \in \mathbb{K}[x]_n$, computing the latter vector amounts to evaluate the polynomial $(xP \bmod F)$ at x, yielding the values $x_i y_i$ as well. $\square$

Polynomial expressions. Multiplication by the various families of structured matrices seen before can be reinterpreted in terms of polynomial operations. This is foremost the case for multiplication by lower or upper-triangular Toeplitz matrices, as seen in the following well-known lemma.

Lemma 4 *Let* y *and* z *be in* $\mathbb{K}^n$ *and let* $\mathsf{u} = \mathbb{L}(\mathsf{y})\, \mathsf{z}$ *and* $\mathsf{v} = \mathbb{U}(\mathsf{y})\, \mathsf{z}$*. We have*

$$\mathsf{Pol}(\mathsf{u}) = \mathsf{Pol}(\mathsf{y})\ \mathsf{Pol}(\mathsf{z}) \bmod x^n,$$
$$\mathsf{Pol}(\mathsf{v}) = \mathsf{Rev}_n(\mathsf{Pol}(\mathsf{y}))\ \mathsf{Pol}(\mathsf{z}) \operatorname{div} x^{n-1}.$$

We gather here some further expressions of a similar spirit, that are needed later. First, we show how to rewrite transposed polynomial multiplication in terms of plain multiplication; the following result is from [9, Section 4.1].

Lemma 5 *Let* P *be in* $\mathbb{K}[x]_m$*,* f *be in* $\mathbb{K}^{n+m-1}$ *and let* $\mathsf{u} = \mathbb{M}(P, n)^t\, \mathsf{f} \in \mathbb{K}^n$*. Then we have*

$$\mathsf{Pol}(\mathsf{u}) = (\mathsf{Rev}_m(P)\, \mathsf{Pol}(\mathsf{f}) \operatorname{div} x^{m-1}) \bmod x^n.$$

The next lemma describes a more complex operation, needed in Section 5.1 for handling Vandermonde-like matrices.

Lemma 6 *Let* z *be in* $\mathbb{K}^n$*,* φ *be in* $\mathbb{K}$*,* P *be in* $\mathbb{K}[x]_m$*,* f *be in* $\mathbb{K}^{n+m-1}$*, and define* $\mathsf{g} \in \mathbb{K}^n$ *by*

$$\mathsf{g} = \mathbb{T}(\mathsf{z}, \varphi)\ \mathbb{M}(P, n)^t\ \mathsf{f}.$$

Define further

- $\mathsf{z}' = \mathsf{Flip}(\mathsf{z})$ *in* $\mathbb{K}^n$,
- $Z = \mathsf{Pol}(\mathsf{z})$ *and* $Z' = \mathsf{Pol}(\mathsf{z}')$ *in* $\mathbb{K}[x]_n$,
- $F = \mathsf{Pol}(\mathsf{f})$ *in* $\mathbb{K}[x]_{n+m-1}$.

Then we have the equality

$$\mathsf{Pol}(\mathsf{g}) = Z\,\Big(\mathsf{Rev}_m(P)\,F \text{ div } x^{m-1}\Big) \bmod x^n \\ + \varphi\,\mathsf{Rev}_n\Big(Z'\,(P\,\mathsf{Rev}_{n+m-1}(F) \text{ div } x^{m-1}) \bmod x^n\Big).$$

Proof. Since by definition $\mathbb{T}(\mathsf{z},\varphi)$ equals $\mathbb{L}(\mathsf{z}) + \varphi\,\mathbb{U}(\mathsf{z}')$, we see that $\mathsf{Pol}(\mathsf{g})$ equals $\mathsf{Pol}(\mathsf{g}')+\varphi\,\mathsf{Pol}(\mathsf{g}'')$, where $\mathsf{g}' = \mathbb{L}(\mathsf{z})\,\mathbb{M}(P,n)^t\,\mathsf{f}$ and $\mathsf{g}'' = \mathbb{U}(\mathsf{z}')\,\mathbb{M}(P,n)^t\,\mathsf{f}$. Thus, by applying successively Lemma 4 and Lemma 5 to g', we first obtain

$$\mathsf{Pol}(\mathsf{g}') = Z\,\Big(\mathsf{Rev}_m(P)\,F \text{ div } x^{m-1}\Big) \bmod x^n.$$

To deal with g'', notice that $\mathbb{U}(\mathsf{z}') = \mathbb{J}\,\mathbb{L}(\mathsf{z}')\,\mathbb{J}$, where the *reversal* matrix $\mathbb{J}$ is zero, except on the anti-diagonal, whose entries are 1's. Then, doing as before,

$$\mathsf{Pol}(\mathsf{g}'') = \mathsf{Rev}_n\Big(Z'\,\mathsf{Rev}_n\Big((\mathsf{Rev}_m(P)\,F \text{ div } x^{m-1}) \bmod x^n\Big) \bmod x^n\Big);$$

observe the presence of the two extra Rev_n operations, due to the conjugation by the $\mathbb{J}$ matrix. To get rid of the second reversal, one checks that

$$\mathsf{Rev}_n\Big((\mathsf{Rev}_m(P)\,F \text{ div } x^{m-1}) \bmod x^n\Big) = \Big(P\,\mathsf{Rev}_{n+m-1}(F) \text{ div } x^{m-1}\Big) \bmod x^n,$$

which gives the required result. □

Algorithms for univariate polynomials. As mentioned before, M denotes a function from $\mathbb{N}_{>0}$ to $\mathbb{R}_{>0}$ such that over any ring, polynomials of degree less than d can be multiplied using at most $\mathsf{M}(d)$ ring operations.

Following [21, Chapter 8], we make the assumption that the function $d \mapsto \mathsf{M}(d)/d$ is non-decreasing; this implies in particular that the super-linearity condition $\mathsf{M}(d) + \mathsf{M}(d') \le \mathsf{M}(d+d')$ holds for all d, d'. Using the results of [49,11], one can take $\mathsf{M}(d) \in O(d\log(d)\log\log(d))$.

Several fast algorithms are based on fast multiplication. Let thus $\mathsf{x} = [x_1, \ldots, x_n]^t$ be in $\mathbb{K}^n$; we will use the following well-known results.

Construction from roots. The polynomials $\prod_{i\le n}(x - x_i)$ and $\prod_{i\le n}(1 - x_i x)$ can be computed in $O(\mathsf{M}(n)\log(n))$ operations [21, Lemma 10.4].

Evaluation. Given $\mathsf{f} \in \mathbb{K}^n$, one can compute $\mathbb{V}(\mathsf{x},n)\,\mathsf{f}$ (equivalently, evaluate any polynomial $F \in \mathbb{K}[x]_n$ at x) in $O(\mathsf{M}(n)\log(n))$ operations [21, Corollary 10.8].

Transposed evaluation. Given $\mathsf{f} \in \mathbb{K}^n$, one can compute the vector $\mathbb{V}(\mathsf{x}, n)^t \mathsf{f}$ in $O(\mathsf{M}(n)\log(n))$ operations [22, Theorem 10.4].

Interpolation. If x is repetition-free and f is in $\mathbb{K}^n$, one can compute the vector $\mathbb{V}(\mathsf{x}, n)^{-1} \mathsf{f}$ (equivalently, interpolate any polynomial $F \in \mathbb{K}[x]_n$ at x) in $O(\mathsf{M}(n)\log(n))$ operations [21, Corollary 10.12].

Transposed interpolation. If x is repetition-free and f is in $\mathbb{K}^n$, one can compute the vector $\mathbb{V}(\mathsf{x}, n)^{-t} \mathsf{f}$ in $O(\mathsf{M}(n)\log(n))$ operations [30, Section 5] (a similar result can be found in [41, Section 10]).

Algorithms for dense matrices. As said earlier, we let $2 < \omega \leq 3$ be such that $n \times n$ matrices over $\mathbb{K}$ can be multiplied in $O(n^\omega)$ operations in $\mathbb{K}$.

Dense matrix operations will be used for several purposes. First, since we require that our fast polynomial multiplication algorithms apply over any ring, we deduce that polynomial matrices over $\mathbb{K}$ having degree less than d and size n can be multiplied in $O(n^\omega \mathsf{M}(d))$ operations. In Section 3, we will actually use the following two results, which deal more precisely with some specific rectangular polynomial matrix products.

Lemma 7 *Let $\mathsf{A}, \mathsf{B}, \mathsf{C}$ be matrices of respective sizes $(n \times p)$, $(p \times n)$ and $(n \times p)$, with entries in $\mathbb{K}[x]_d$. Then one can compute the product $\mathsf{A}\,\mathsf{B}\,\mathsf{C}$ using $O(p^{\omega-1} n \mathsf{M}(d))$ operations.*

Proof. Suppose first that $p \leq n$. Up to bordering the matrices by less than p zero rows or columns, we can suppose that p divides n. Let then $\ell = n/p$. We can rewrite the product $\mathsf{A}\,\mathsf{B}\,\mathsf{C}$ as a product of matrices of size $(\ell \times 1) \times (1 \times \ell) \times (\ell \times 1)$, having blocks of size $p \times p$ as entries. We then compute this product as $\mathsf{A}\,(\mathsf{B}\,\mathsf{C})$; the cost is $O(p^\omega \ell \mathsf{M}(d))$, which is $O(p^{\omega-1} n \mathsf{M}(d))$.

Suppose now that $n \leq p$; as above, we suppose that n divides p and let $\ell = p/n$. Then, the product $\mathsf{A}\,\mathsf{B}\,\mathsf{C}$ is rewritten as a product of matrices of size $(1 \times \ell) \times (\ell \times 1) \times (1 \times \ell)$, having blocks of size $n \times n$ as entries. We then compute this product as $(\mathsf{A}\,\mathsf{B})\,\mathsf{C}$; the cost is $O(n^\omega \ell \mathsf{M}(d))$, that is, in $O(n^{\omega-1} p \mathsf{M}(d))$. This is also in $O(p^{\omega-1} n \mathsf{M}(d))$, since $0 \leq n \leq p$ and $\omega \geq 2$ imply $n^{\omega-1} p \leq p^{\omega-1} n$. □

Lemma 8 *Let A and B be matrices of respective sizes $(n \times p)$ and $(p \times m)$, with entries in $\mathbb{K}[x]_d$. If $n \leq p$, then one can compute the product $\mathsf{A}\,\mathsf{B}$ using $O(p^{\omega-1} \max\{n, m\} \mathsf{M}(d))$ operations.*

Proof. Suppose first that $n \leq m$. As in the previous lemma, up to bordering A by less than n zero columns and B by less than n zero columns and rows, we can suppose that n divides p and m. Let $k = p/n$ and $\ell = m/n$. Then, the product $\mathsf{A}\,\mathsf{B}$ is rewritten as a product of matrices of size $(1 \times k) \times (k \times \ell)$,

having blocks of size $n \times n$ as entries. The cost for computing this product is thus $O(n^\omega\, k\, \ell\, \mathsf{M}(d))$ operations, which is $O(n^{\omega-2}\, p\, m\, \mathsf{M}(d))$. Since $0 \leq n \leq p$ and $\omega - 2 \geq 0$, this is in $O(p^{\omega-1}\, m\, \mathsf{M}(d))$, as required.

Assuming $m \leq n$, we obtain a similar estimate of $O(m^{\omega-2}\, p\, n\, \mathsf{M}(d))$. Since then $0 \leq m \leq p$ and $\omega - 2 \geq 0$, this is in $O(p^{\omega-1}\, n\, \mathsf{M}(d))$, as required. □

Besides fast (polynomial) matrix multiplication in the above two particular cases, we shall also use fast elimination on dense matrices. We thus give the following result on isolating a basis of the row-span of a matrix. This is a simple consequence of Proposition 2.15 in [51].

Lemma 9 *Let A be in $\mathbb{K}^{n\times p}$. One can compute in time $O(r^{\omega-2}np)$ a quadruple $(r, J, \mathsf{G}, \mathsf{P})$ such that r is the rank of A, J is a subset of $\{1, \ldots, n\}$ of length r, G is a matrix in $\mathbb{K}^{(n-r)\times r}$, P is a permutation matrix of order n, and*

$$\mathsf{E}\,\mathsf{A} = \begin{bmatrix} \mathsf{A}' \\ 0 \end{bmatrix}, \quad \textit{where} \quad \mathsf{E} = \begin{bmatrix} \mathbb{I}_r & 0 \\ \mathsf{G} & \mathbb{I}_{n-r} \end{bmatrix} \mathsf{P},$$

and where $\mathsf{A}' \in \mathbb{K}^{r\times p}$ consists of the rows of A indexed by J.

Proof. Using Algorithm 2.14 in [51], one can compute in time $O(r^{\omega-2}np)$ the rank r of A along with a permutation matrix P, and a matrix U of the form

$$\mathsf{U} = \begin{bmatrix} \mathsf{F} & 0 \\ \mathsf{G} & \mathbb{I}_{n-r} \end{bmatrix}, \quad \text{with } \mathsf{F} \in \mathbb{K}^{r\times r},$$

such that the last $n - r$ rows of the product $\mathsf{U}\,\mathsf{P}\,\mathsf{A}$ are zero. Then let $J = \{i_1, \ldots, i_r\}$, where i_j is the index of the non-zero entry of the jth row of P. Define E as above, replacing F by the identity $\mathbb{I}_r$ in U; then, the first r rows of $\mathsf{P}\,\mathsf{A}$, and thus of $\mathsf{E}\,\mathsf{A}$, are the rows of A indexed by J. □

The above lemma will be used in Subsections 5.3 and 6.3 for introducing zeros in some generators of Vandermonde and Cauchy-like matrices, and thus handling the high multiplicities that may arise in those cases. Another application of this lemma is the fast computation of generators of minimal length (or, equivalently, of maximal rank), an operation already needed for the Toeplitz case. The proof of the result below, which is Remark 4.6.7 in [45], shows this.

Lemma 10 *Let $\mathsf{P}, \mathsf{Q} \in \mathbb{K}^{n\times n}$. Given a P, Q-generator of length $\alpha \leq n$ for $\mathsf{A} \in \mathbb{K}^{n\times n}$, one can compute a P, Q-generator for A of minimal length in time $O(\alpha^{\omega-1}n)$.*

Proof. Denoting by (Y, Z) the given generator, we have $\mathsf{A} - \mathsf{P}\,\mathsf{A}\,\mathsf{Q} = \mathsf{Y}\,\mathsf{Z}^t$. Let V be an invertible matrix in $\mathbb{K}^{\alpha\times\alpha}$ such that $\mathsf{Y}\,\mathsf{V} = [\mathsf{Y}'\ 0]$, with $\mathsf{Y}' \in \mathbb{K}^{n\times r}$

and $r = \mathrm{rank}(\mathsf{Y})$. Let also $\mathsf{Z}' \in \mathbb{K}^{n\times r}$ consist of the first r columns of $\mathsf{Z}\,\mathsf{V}^{-t}$. We have $\mathsf{Y}\,\mathsf{Z}^t = \mathsf{Y}'\,\mathsf{Z}'^t$ and thus $(\mathsf{Y}', \mathsf{Z}')$ is a P, Q-generator for A of length r.

Although Y' has full rank, Z' may have rank less than r. Therefore, let further V' be an invertible matrix in $\mathbb{K}^{r\times r}$ such that $\mathsf{Z}'\,\mathsf{V}' = [\mathsf{H}\ 0]$, with $\mathsf{H} \in \mathbb{K}^{n\times r'}$ and $r' = \mathrm{rank}(\mathsf{Z}')$. Defining G as the matrix in $\mathbb{K}^{n\times r'}$ that consists of the first r' columns of $\mathsf{Y}'\,\mathsf{V}'^{-t}$, we obtain the equality $\mathsf{Y}'\,\mathsf{Z}'^t = \mathsf{G}\,\mathsf{H}^t$. Since now both G and H have full rank, they form a P, Q-generator for A of minimal length.

By Lemma 9, the cost of computing r and V is in $O(\alpha^{\omega-1}n)$. One then obtains Y' in time $O(\alpha^{\omega-1}n)$ by multiplication with V. To get Z', one can first compute V^{-t} in time $O(\alpha^{\omega})$ and then deduce $\mathsf{Z}\,\mathsf{V}^{-t}$ in time $O(\alpha^{\omega-1}n)$. Similarly, deducing (G, H) from $(\mathsf{Y}', \mathsf{Z}')$ can be done in time $O(\alpha^{\omega-1}n)$, as required. □

Partitions and multiplicities. Our algorithms for Vandermonde and Cauchy-like systems may require to perform rearrangements of the diagonal matrices appearing in the corresponding displacement operators. We detail this process here, pointing out in particular that performing the required permutations is inexpensive.

We start by integer partitions. A partition ν of $n \in \mathbb{N}_{>0}$ is a sequence of positive integers $\nu_1 \geq \cdots \geq \nu_s$ such that $\nu_1 + \cdots + \nu_s = n$. It will be useful to associate to ν the sequence $\nu_0^* = 0, \nu_1^* = \nu_1, \ldots, \nu_i^* = \nu_1 + \cdots + \nu_i, \ldots$.

The *conjugate* of ν is another partition $\mu = \mu_1 \geq \cdots \geq \mu_t$ of n, where μ_j is the number of elements i in $\{1, \ldots, s\}$ such that $\nu_i \geq j$; in particular, the conjugate of μ is ν and we have $t = \nu_1$ and $s = \mu_1$. Remark that given ν, one deduces μ in n integer additions: after initializing μ at $0, \ldots, 0$ (ν_1 repetitions), we obtain μ by incrementing μ_j, for $i = 1, \ldots, s$ and $j = 1, \ldots, \nu_i$,

Next, we discuss partitions of vectors. Let thus x be in $\mathbb{K}^n$. We suppose that x has the form of Equation (1):

$$\mathsf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \vdots \\ \mathsf{x}_s \end{bmatrix}, \text{ with } \mathsf{x}_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,\nu_i} \end{bmatrix} \in \mathbb{K}^{\nu_i},$$

where each x_j is a repetition-free vector of size ν_j and where for $j < s$, x_{j+1} is a prefix of x_j. Since $\nu = \nu_1 \geq \cdots \geq \nu_s$ is a partition of n, we let $\mu = \mu_1 \geq \cdots \geq \mu_t$ be the conjugate partition. The number of occurrences of $x_{1,j}$ in x is thus μ_j; we call it its *multiplicity*, and call $\mu_1 = s$ the multiplicity of x.

It follows that, after permuting its entries, x can be rewritten as

$$\mathsf{x}' = \begin{bmatrix} x_{\sigma(1)} \\ \vdots \\ x_{\sigma(n)} \end{bmatrix} = \begin{bmatrix} \mathsf{x}'_1 \\ \vdots \\ \mathsf{x}'_t \end{bmatrix}, \tag{6}$$

where x'_j is a vector consisting of μ_j repetitions of $x_{1,j}$. For example, if $\mathsf{x} = [1, 2, 4, 1, 2, 1, 2]^t$, we have

$$\mathsf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}, \mathsf{x}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathsf{x}_3 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{and} \quad \mathsf{x}'_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \mathsf{x}'_2 = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}, \mathsf{x}'_3 = \begin{bmatrix} 4 \end{bmatrix},$$

with $s = t = 3$, $(\nu_1, \nu_2, \nu_3) = (3, 2, 2)$, $(\mu_1, \mu_2, \mu_3) = (3, 3, 1)$, $\sigma = [1, 4, 6, 2, 5, 7, 3]$ and $\sigma^{-1} = [1, 4, 7, 2, 5, 3, 6]$.

Lemma 11 *Given x as in Equation (1), one can deduce a permutation σ as in Equation (6) in time $O(n)$; the converse operation can be done within the same complexity.*

Proof. The partition ν is obtained by scanning x for occurrences of $x_{1,1}$, in time $O(n)$. One obtains μ as the conjugate of ν for a similar cost; the permutation σ can then be made explicit as

$$\sigma(1), \ldots, \sigma(n) = 1, 1 + \nu^*_1, \ \ldots, \ 1 + \nu^*_{\mu_1 - 1}, \ \ldots, \ \nu_1, \nu_1 + \nu^*_1, \ \ldots, \ \nu_1 + \nu^*_{\mu_t - 1},$$

the first block giving the indices of $x_{1,1} = x_{2,1} = \cdots = x_{s,1}$ in x and so on. Conversely, given x', μ is obtained by scanning x' for indices where two consecutive entries differ. Then, one recovers ν and σ^{-1} in the same way as before, changing the roles of ν and μ. In any case, the cost fits into the required bound. □

By analogy with the squarefree decomposition of polynomials, we call the representation in Equation (6) a *repetition-free decomposition* of x.

3 Polynomial operations

We discuss now two problems involving polynomials. Both boil down to suitably using polynomial matrix multiplication to speed up the simultaneous computation of several bilinear or trilinear expressions; as it turns out, these questions are the main ingredients of the algorithms in the following sections.

In the following, some integers n and $\alpha \leq n$ are fixed. Let

$$(Y_i)_{i\leq\alpha}, \quad (Z_i)_{i\leq\alpha} \quad \text{and} \quad (F_j)_{j\leq\alpha}$$

be in $\mathbb{K}[x]_n$. The next proposition will be used in Section 4, as the key to the proof of Theorem 1 for Toeplitz-like systems. The cost reported in this proposition is actually the bottleneck of the main algorithm: any reduction on the following estimate would entail a reduction of the overall cost.

Proposition 1 *One can compute the polynomials*

$$G_j = \sum_{i=1}^{\alpha} Y_i \, (Z_i \, F_j \bmod x^n), \quad j = 1, \ldots, \alpha$$

in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$.

The direct approach, used in previous work, consists in computing all polynomials G_j independently, for a cost of $O(\alpha^2\,\mathsf{M}(n))$ operations. Our contribution shows how to compute the polynomials G_j simultaneously using polynomial matrix multiplication.

Proof. Up to replacing n with $\bar{n} = 2^{\lceil \log(n) \rceil}$ and F_j with $x^{\bar{n}-n}F_j$, we can (and will) suppose that n is a power of 2.

We first show how to rewrite truncated products using non-truncated ones, using ideas reminiscent of *short products* [39]. Let $k \geq 1$ be a power of 2 and let ℓ be in $\mathbb{N}$. For $P = p_0 + p_1 x + \cdots$, we define $P^{(\ell,k)} \in \mathbb{K}[x]$ as follows:

$$P^{(\ell,1)} = p_\ell \quad \text{and} \quad P^{(\ell,k)} = \sum_{i=\ell k}^{\ell k + k/2 - 1} p_i x^{i-\ell k} \quad \text{for } k \geq 2.$$

In all cases, $P^{(\ell,k)}$ is a polynomial of degree less than $k/2$. Using this subdivision enables us to rewrite a truncated product $PQ \bmod x^n$ as a sum of non-truncated ones.

Lemma 12 *For P and Q in $\mathbb{K}[x]$ and m a power of 2,*

$$PQ \bmod x^m = \sum_{k=1,2,4,\ldots,m} x^{m-k} \sum_{\ell=0}^{m/k-1} P^{(\ell,k)} Q^{(m/k-1-\ell,k)},$$

where the sum is taken on all $k \leq m$ that are powers of 2.

Proof. We proceed by induction on $m \geq 1$, for m a power of 2. If $m = 1$ the result is clear, so, assuming that the property holds at index $m/2$, we prove it

at index m. Let us write

$$P \bmod x^m = P_0 + x^{m/2} P_1 \text{ and } Q \bmod x^m = Q_0 + x^{m/2} Q_1,$$

with P_0, P_1, Q_0, Q_1 of degree less than $m/2$. Then we have

$$P_0^{(\ell,k)} = P^{(\ell,k)} \quad \text{and} \quad P_1^{(\ell,k)} = P^{(\ell+m/2k,k)}$$

for any $k \geq 1$ and $\ell \geq 0$ such that $\ell k + k/2 \leq m/2$. Analogous equalities hold for Q, Q_0 and Q_1. Now, by definition, the following equality holds:

$$PQ \bmod x^m = P_0 Q_0 + x^{m/2}(P_0 Q_1 + P_1 Q_0 \bmod x^{m/2}). \tag{7}$$

Observe first that $P_0 Q_0$ equals $P^{(0,m)} Q^{(0,m)}$, which corresponds to the term $k = m$ in the right-hand side of the formula we wish to establish. Next, the induction assumption shows that $P_0 Q_1 \bmod x^{m/2}$ is given by

$$\sum_{k=1,2,\ldots,m/2} x^{m/2-k} \sum_{\ell=0}^{m/2k-1} P_0^{(\ell,k)} Q_1^{(m/2k-1-\ell,k)},$$

which we rewrite as

$$\sum_{k=1,2,\ldots,m/2} x^{m/2-k} \sum_{\ell=0}^{m/2k-1} P^{(\ell,k)} Q^{(m/k-1-\ell,k)}.$$

Similarly, $P_1 Q_0 \bmod x^{m/2}$ equals

$$\sum_{k=1,2,\ldots,m/2} x^{m/2-k} \sum_{\ell=m/2k}^{m/k-1} P^{(\ell,k)} Q^{(m/k-1-\ell,k)}.$$

Putting these equalities in Equation (7) ends the proof. □

We can now prove the proposition. By Lemma 12, we have for all i and j

$$Y_i (Z_i F_j \bmod x^n) = \sum_{k=1,2,4,\ldots,n} x^{n-k} \sum_{\ell=0}^{n/k-1} Y_i \; Z_i^{(\ell,k)} F_j^{(n/k-1-\ell,k)}.$$

Thus for $j \leq \alpha$, we have $G_j = \sum_{k=1,2,4,\ldots,n} x^{n-k} \, G_{j,k}$, with

$$G_{j,k} = \sum_{i=1}^{\alpha} \sum_{\ell=0}^{n/k-1} Y_i \; Z_i^{(\ell,k)} \; F_j^{(n/k-1-\ell,k)}.$$

We next show how to compute all polynomials $G_{1,k}, \ldots, G_{\alpha,k}$, for a fixed k.

Lemma 13 *Let $k \leq n$ be a power of 2. Then one can compute $G_{1,k}, \ldots, G_{\alpha,k}$ in time $O(\alpha^{\omega-1} \mathsf{M}(n))$.*

PROOF. Let $k' = n/k$, and let Z and F be the $(\alpha \times k')$ and $(k' \times \alpha)$ polynomial matrices

$$\mathsf{Z} = \begin{bmatrix} Z_1^{(0,k)} & \cdots & Z_1^{(k'-1,k)} \\ \vdots & & \vdots \\ Z_\alpha^{(0,k)} & \cdots & Z_\alpha^{(k'-1,k)} \end{bmatrix}, \quad \mathsf{F} = \begin{bmatrix} F_1^{(k'-1,k)} & \cdots & F_\alpha^{(k'-1,k)} \\ \vdots & & \vdots \\ F_1^{(0,k)} & \cdots & F_\alpha^{(0,k)} \end{bmatrix}.$$

Then we have the equality

$$\begin{bmatrix} G_{1,k} & \cdots & G_{\alpha,k} \end{bmatrix} = \begin{bmatrix} Y_1 & \cdots & Y_\alpha \end{bmatrix} \mathsf{Z}\,\mathsf{F}.$$

All entries of Z and F have degree less than $k/2$, whereas the polynomials Y_i have degree less than n. To balance the degrees, for $i \leq \alpha$, we write $Y_i = \sum_{\ell=0}^{k'-1} Y_{i,\ell} x^{k\ell}$, with $Y_{i,\ell}$ of degree less than k. We then define the $(k' \times \alpha)$ matrix

$$\mathsf{Y} = \begin{bmatrix} Y_{1,0} & \cdots & Y_{\alpha,0} \\ \vdots & & \vdots \\ Y_{1,k'-1} & \cdots & Y_{\alpha,k'-1} \end{bmatrix}$$

with polynomial entries of degree less than k, such that

$$\begin{bmatrix} Y_1 & \cdots & Y_\alpha \end{bmatrix} = \begin{bmatrix} 1 & x^k & x^{2k} & \cdots & x^{(k'-1)k} \end{bmatrix} \mathsf{Y}. \tag{8}$$

Using Lemma 7, we can compute the product $\mathsf{Y}\,\mathsf{Z}\,\mathsf{F}$ in $O(\alpha^{\omega-1}\, k'\, \mathsf{M}(k))$ operations. Since $k'\,\mathsf{M}(k) \leq \mathsf{M}(n)$, this cost is in $O(\alpha^{\omega-1}\mathsf{M}(n))$. Finally, by Equation (8), we deduce $G_{1,k}, \ldots, G_{\alpha,k}$ from the product $\mathsf{Y}\,\mathsf{Z}\,\mathsf{F}$ in time $O(\alpha\, k'\, k)$, which is in $O(\alpha\, n)$. □

To conclude the proof of Proposition 1, we apply Lemma 13 to $k = 1, 2, 4, \ldots, n$, for a total cost of $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$. The cost of deducing $G_1, \ldots, G_\alpha$ is a negligible $O(\alpha\, n \log(n))$. □

3.2 Second problem

As above, some integers $n \in \mathbb{N}$ and $\alpha \leq n$ are fixed. Let also s and t be positive integers bounded by α and let $\nu_1, \ldots, \nu_s$ and $\delta_1, \ldots, \delta_t$ be positive integers such that

$$n = \nu_1 + \cdots + \nu_s \quad \text{and} \quad n = \delta_1 + \cdots + \delta_t.$$

Let then

$$(Q_{i,j})_{i \leq \alpha,\, j \leq s} \quad \text{and} \quad (R_{i,k})_{i \leq \alpha,\, k \leq t}$$

be in $\mathbb{K}[x]$, with $\deg(Q_{i,j}) < \nu_j$ and $\deg(R_{i,k}) < \delta_k$. The following result is used in Section 6 to reduce the solution of Cauchy-like systems to that of Vandermonde-like systems.

Proposition 2 *One can compute the polynomials*

$$P_{j,k} = \sum_{i=1}^{\alpha} Q_{i,j} R_{i,k}, \quad j = 1, \ldots, s, \quad k = 1, \ldots, t$$

in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$.

Remark that computing all polynomials $P_{j,k}$ independently leads to a cost in $O(\alpha^2\,\mathsf{M}(n))$, which is quadratic in α. As we did in the previous subsection, we are going to use polynomial matrix multiplication to compute the polynomials $P_{j,k}$ simultaneously and obtain a cost subquadratic in α.

Proof. Let $L = \{0, 1, \ldots, \lfloor \log(n) \rfloor\}$. For $\beta, \gamma \in L$, define

$$S_\beta = \{j : 2^\beta \leq \nu_j < 2^{\beta+1}\} \quad \text{and} \quad T_\gamma = \{k : 2^\gamma \leq \delta_k < 2^{\gamma+1}\};$$

we then let $q_\beta = |S_\beta|$ and $r_\gamma = |T_\gamma|$. Rewrite the equality $n = \sum_{j=1}^{s} \nu_j$ as

$$n = \sum_{\beta \in L} \sum_{j \in S_\beta} \nu_j.$$

Since for $j \in S_\beta$, all ν_j are at least 2^β, we deduce the following inequality, together with its analogue for the subsets T_γ:

$$\sum_{\beta \in L} 2^\beta q_\beta \leq n \quad \text{and} \quad \sum_{\gamma \in L} 2^\gamma r_\gamma \leq n. \tag{9}$$

Given the integers $\nu_1, \ldots, \nu_s$ and $\delta_1, \ldots, \delta_t$, all subsets S_β and T_γ can be obtained in time $O(n)$. Then, for a given pair (β, γ), the polynomials $\{P_{j,k} : j \in S_\beta, k \in T_\gamma\}$ are the entries of the matrix product $\mathsf{Q}\,\mathsf{R}$, with

$$\mathsf{Q} = \begin{bmatrix} \vdots & & \vdots \\ Q_{1,j} & \cdots & Q_{\alpha,j} \\ \vdots & & \vdots \end{bmatrix}_{j \in S_\beta}, \quad \mathsf{R} = \begin{bmatrix} \cdots & R_{1,k} & \cdots \\ & \vdots & \\ \cdots & R_{\alpha,k} & \cdots \end{bmatrix}_{k \in T_\gamma}.$$

The polynomial matrix Q has dimensions $(q_\beta \times \alpha)$ and degree less than $2^{\beta+1}$; R has dimensions $(\alpha \times r_\gamma)$ and degree less than $2^{\gamma+1}$.

Let us first deal with the case where $\beta \leq \gamma$. Then, the polynomials in R have potentially larger degrees than the ones in Q. To balance these degrees, we write $\mathsf{R} = \sum_{\ell=0}^{2^{\gamma-\beta}-1} \mathsf{R}_\ell\, x^{2^{\beta+1}\ell}$, where R_ℓ is an $(\alpha \times r_\gamma)$ polynomial matrix with entries of degree less than $2^{\beta+1}$.

From the knowledge of the products $\{\mathsf{Q}\,\mathsf{R}_\ell : 0 \le \ell < 2^{\gamma-\beta}\}$, one can deduce $\mathsf{Q}\,\mathsf{R}$ using $O(\alpha\, 2^\gamma\, r_\gamma)$ extra additions; summing over all β, γ with $\beta \le \gamma$, and using Equation (9), this amounts to $O(\alpha\, n \log(n))$ additions. Hence, we focus on computing the former products. Concatenating all matrices R_ℓ, let us define

$$\mathsf{R}' = \left[\mathsf{R}_0 \ \cdots \ \mathsf{R}_{2^{\gamma-\beta}-1}\right];$$

it is then enough to compute $\mathsf{Q}\,\mathsf{R}'$, since the $\mathsf{Q}\,\mathsf{R}_\ell$ can be read off this product.

The matrix R' has dimensions $(\alpha \times r'_\gamma)$, with $r'_\gamma = 2^{\gamma-\beta} r_\gamma$, and entries of degree less than $2^{\beta+1}$. Since $q_\beta \le \alpha$, Lemma 8 shows that the cost of computing the product $\mathsf{Q}\,\mathsf{R}'$ is in

$$O(\alpha^{\omega-1} \max\{q_\beta, r'_\gamma\}\, \mathsf{M}(2^{\beta+1})).$$

Recall that by assumption on the function M, we have $\mathsf{M}(2^{\beta+1}) \le 2^{\beta+1}\mathsf{M}(n)/n$. Bounding the max by a sum and substituting r'_γ by its value, we deduce the upper bound

$$O\left(\alpha^{\omega-1}\, 2^\beta\, q_\beta\, \frac{\mathsf{M}(n)}{n} + \alpha^{\omega-1}\, 2^\gamma r_\gamma\, \frac{\mathsf{M}(n)}{n}\right).$$

Summing over all β, γ with $\beta \le \gamma$, and using Equation (9), we obtain an upper bound in $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$. The case where $\beta > \gamma$ is handled similarly. □

We will need in Section 5 the following corollary of the previous proposition, to reduce the solution of Vandermonde-like systems to that of Toeplitz-like systems. Let $s \le \alpha$ and $\nu_1, \ldots, \nu_s$ be as above and let

$$(Z_i)_{i\le\alpha}, \quad (Y_{i,j})_{i\le\alpha,\, j\le s}, \quad \text{and} \quad (F_j)_{j\le s}$$

be in $\mathbb{K}[x]$, with $\deg(Z_i) < n$, $\deg(Y_{i,j}) < \nu_j$ and $\deg(F_j) < n + \nu_j$.

Proposition 3 *One can compute the polynomials*

$$P_j = \sum_{i=1}^{\alpha} Z_i \left(Y_{i,j}F_j \ \text{div}\ x^{\nu_j-1}\right), \quad j = 1, \ldots, s$$

in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$.

The remarks we did before still apply: a direct approach consists in computing all polynomials P_j independently, for a cost in $O(\alpha^2\, \mathsf{M}(n))$. Using polynomial matrix multiplication to share computations, we get a cost subquadratic in α.

Proof. For $i \le \alpha$ and $j \le s$, let $G_{i,j} = Y_{i,j}F_j \bmod x^{\nu_j-1}$. Then, for $j \le s$, we can define

$$Q_j = \sum_{i=1}^{\alpha} Z_i G_{i,j} \quad \text{and} \quad R_j = \sum_{i=1}^{\alpha} Z_i Y_{i,j}.$$

It follows that P_j is given by $(R_jF_j - Q_j) \ \text{div}\ x^{\nu_j-1}$.

For given i and j, the polynomial $G_{i,j}$ can be computed in $\mathsf{M}(\nu_j)$ operations; hence, the whole time for computing all these polynomials is at most $\alpha\,\mathsf{M}(n)$. Applying Proposition 2 with $t = 1$ and $\delta_1 = n$, we deduce that we can compute all polynomials Q_j and R_j using $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$ operations. Finally, we recover the polynomials P_j in time $O(s\,\mathsf{M}(n)) \subset O(\alpha\,\mathsf{M}(n))$. □

4 The Toeplitz case

The operator associated with the Toeplitz structure is

$$\Delta[\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^t](\mathsf{A}) = \mathsf{A} - \mathbb{Z}_{n,0}\,\mathsf{A}\,\mathbb{Z}_{n,0}^t, \quad \mathsf{A} \in \mathbb{K}^{n\times n}. \tag{10}$$

This operator is invertible: given (Y, Z) in $\mathbb{K}^{n\times\alpha} \times \mathbb{K}^{n\times\alpha}$, there exists a unique A such that $\Delta[\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^t](\mathsf{A}) = \mathsf{Y}\mathsf{Z}^t$; it is given by the representation

$$\mathsf{A} = \sum_{i=1}^{\alpha} \mathbb{L}(\mathsf{y}_i)\mathbb{U}(\mathsf{z}_i),$$

called in [28,29] a ΣLU *representation of length* α *for* A. Using Lemma 4, this representation allows one to compute a matrix-vector product $\mathsf{A}\mathsf{u}$ in $O(\alpha\,\mathsf{M}(n))$ operations. Our problem in this section is the converse one: given v in $\mathbb{K}^n$, find u such that $\mathsf{A}\mathsf{u} = \mathsf{v}$ (or conclude that no such vector exists).

For large α, we improve previous algorithms, reducing their complexity from $O(\alpha^2\,\mathsf{M}(n)\log(n))$ to $O(\alpha^{\omega-1}\,\mathsf{M}(n)\log^2(n))$. The structure of our algorithm is similar to those initiated by Bitmead and Anderson [7] and Morf [37,38]; the key difference consists in using the results of the previous section to perform efficiently the following operation: given Y, Z as above and $\mathsf{u}_1, \ldots, \mathsf{u}_\alpha$ in $\mathbb{K}^n$, compute the α products $\mathsf{v}_j = \mathsf{A}\mathsf{u}_j$.

4.1 Preliminaries

In addition to the operator in Equation (10) we will use the operator

$$\Delta[\mathbb{Z}_{n,0}^t, \mathbb{Z}_{m,0}](\mathsf{A}) = \mathsf{A} - \mathbb{Z}_{n,0}^t\,\mathsf{A}\,\mathbb{Z}_{m,0}, \quad \mathsf{A} \in \mathbb{K}^{n\times m}.$$

Regardless of dimensions, the operators $\Delta[\mathbb{Z}_{n,0}^t, \mathbb{Z}_{m,0}]$ and $\Delta[\mathbb{Z}_{n,0}, \mathbb{Z}_{m,0}^t]$ are called respectively ϕ_- and ϕ_+ in [43,28,29]; their generators are called ϕ_--generators and ϕ_+-generators. From now on, we use this simplifying notation.

We give in this subsection some useful results on generators for submatrices, sums, products, ... Our contribution is Proposition 4 below, which is a faster

version of [43, Proposition A.3] for generating matrix products; as in [28,29] we extend the result to rectangular matrices. Proofs not given here can be found in e.g. [7,38,43,28,45].

A first key feature of the operators ϕ_+ and ϕ_- is that when A is invertible, the ranks of $\phi_+(\mathsf{A})$ and $\phi_-(\mathsf{A}^{-1})$ coincide. Secondly, when A is square, the ranks of $\phi_+(\mathsf{A})$ and $\phi_-(\mathsf{A})$ differ by at most 2.

The next lemma gives the complexity of converting from ϕ_-- to ϕ_+-generators; the same holds for converting back.

Lemma 14 *Given a ϕ_--generator of length α for the matrix $\mathsf{A} \in \mathbb{K}^{n\times n}$, one can compute a ϕ_+-generator of length $\alpha + 2$ for A in time $O(\alpha\,\mathsf{M}(n))$.*

Assuming that $n = m$, partition A into blocks as

$$\mathsf{A} = \begin{bmatrix} \mathsf{A}_{1,1} & \mathsf{A}_{1,2} \\ \mathsf{A}_{2,1} & \mathsf{A}_{2,2} \end{bmatrix}, \tag{11}$$

with $\mathsf{A}_{i,j} \in \mathbb{K}^{n_i\times n_j}$, and $n_1 + n_2 = n$. Then the rank of $\phi_+(\mathsf{A}_{1,1})$ is at most that of $\phi_+(\mathsf{A})$; if $\mathsf{A}_{1,1}$ is invertible and has its upper-left entry non-zero then the same bound holds for the Schur complement $\mathsf{A}_{2,2} - \mathsf{A}_{2,1}\mathsf{A}_{1,1}^{-1}\mathsf{A}_{1,2}$.

Next, we consider the cost of deducing generators for the blocks $\mathsf{A}_{i,j}$ from generators for A, and conversely.

Lemma 15 *Given a ϕ_+-generator of length α for A, one can find ϕ_+-generators of length $O(\alpha)$ for all $\mathsf{A}_{i,j}$ in time $O(\alpha\,\mathsf{M}(n))$. Conversely, given ϕ_+-generators of length at most α for all $\mathsf{A}_{i,j}$, one can find a ϕ_+-generator of length $O(\alpha)$ for A in time $O(\alpha\,\mathsf{M}(n))$.*

Adding matrices given by their generators is straightforward (even though the generators thus obtained may not be minimal).

Lemma 16 *If (T,U) and (Y,Z) are ϕ_+-generators for some matrices A and B of the same dimensions, then $([\mathsf{T}\ \mathsf{Y}],\ [\mathsf{U}\ \mathsf{Z}])$ is a ϕ_+-generator for $\mathsf{A}+\mathsf{B}$.*

We conclude with the key novelty of this subsection, which concerns the complexity of computing a generator for the product of two matrices.

Proposition 4 *Let (T,U) and (Y,Z) be ϕ_+-generators of lengths α and β for some matrices $\mathsf{A} \in \mathbb{K}^{n\times m}$ and $\mathsf{B} \in \mathbb{K}^{m\times p}$. Then one can find a ϕ_+-generator of length $\alpha+\beta+1$ for the product AB in time $O(\gamma^{\omega-1}\mathsf{M}(q)\log(q))$, with $\gamma = \max\{\alpha,\beta\}$ and $q = \max\{n,m,p\}$.*

Proof. Let $\mathsf{V} = \mathsf{B}^t\mathsf{U}$ and $\mathsf{W} = \mathbb{Z}_{n,0}\,\mathsf{A}\,\mathbb{Z}_{m,0}^t\mathsf{Y}$; let also a (resp. b) be the lower

shift of the last column of A (resp. B^t). Then, following [43, Proposition A.3], the proof of [28, Proposition 2] shows that $[\mathsf{T}\ \mathsf{W}\ \mathsf{a}]$ and $[\mathsf{V}\ \mathsf{Z}\ -\mathsf{b}]$ form a ϕ_+-generator of length $\alpha+\beta+1$ for AB. Hence, it suffices to bound the cost of computing V, W, a and b.

Let us detail the computation of $\mathsf{V} = \mathsf{B}^t\mathsf{U}$ when $m \geq p$. We reduce to the square case by defining $\mathsf{B}' = [0\ \mathsf{B}] \in \mathbb{K}^{m\times m}$ and $\mathsf{V}' = \mathsf{B}'^t\mathsf{U} \in \mathbb{K}^{m\times\alpha}$. Since V can be read off V', we focus on computing the latter matrix.

Remark that $\phi_+(\mathsf{B}') = \mathsf{Y}\mathsf{Z}'^t$ with $\mathsf{Z}'^t = [0\ \mathsf{Z}^t] \in \mathbb{K}^{\beta\times m}$. Since B' is square, the Gohberg-Semencul formula then shows that

$$\mathsf{B}' = \sum_{i=1}^{\beta} \mathbb{L}(\mathsf{y}_i)\mathbb{U}(\mathsf{z}'_i),$$

where y_i is the ith column of Y and z'_i the ith column of Z'. Thus, recalling that $\mathbb{J}$ denotes the reversal matrix of order m, the transpose of B' is given by

$$\mathsf{B}'^t = \sum_{i=1}^{\beta} \mathbb{L}(\mathsf{z}'_i)\mathbb{U}(\mathsf{y}_i) = \sum_{i=1}^{\beta} \mathbb{J}\,\mathbb{U}(\mathsf{z}'_i)\,\mathbb{L}(\mathsf{y}_i)\,\mathbb{J}.$$

Now, let u_j (resp. v'_j) be the jth column of U (resp. V'). The previous formula for B'^t and the equation $\mathsf{V}' = \mathsf{B}'^t\mathsf{U}$ thus give

$$\mathbb{J}\,\mathsf{v}'_j = \sum_{i=1}^{\beta} \mathbb{U}(\mathsf{z}'_i)\mathbb{L}(\mathsf{y}_i)\,\mathbb{J}\,\mathsf{u}_j.$$

In polynomial terms, in view of Lemma 4, this reads

$$\mathsf{Rev}_m(\mathsf{Pol}(\mathsf{v}'_j)) = \left[\sum_{i=1}^{\beta} Z'_i\,(Y_iU_j \bmod x^m)\right] \text{ div } x^{m-1},$$

with $Z'_i = \mathsf{Rev}_m(\mathsf{Pol}(\mathsf{z}'_i))$, $Y_i = \mathsf{Pol}(\mathsf{y}_i)$ and $U_j = \mathsf{Rev}_m(\mathsf{Pol}(\mathsf{u}_j))$. By Proposition 1, we can compute the polynomials

$$\sum_{i=1}^{\beta} Z'_i\,(Y_iU_j \bmod x^m), \quad j = 1,\ldots,\alpha$$

in time $O(\max\{\alpha,\beta\}^{\omega-1}\,\mathsf{M}(m)\log(m))$; the vectors v'_j are then deduced by coefficient extraction. The case $p > m$ is treated similarly, padding B with $p-m$ zero rows, and giving a cost of $O(\max\{\alpha,\beta\}^{\omega-1}\,\mathsf{M}(p)\log(p))$. Hence, in any case, V can be obtained in time $O(\gamma^{\omega-1}\,\mathsf{M}(q)\log(q))$, as claimed.

The computation of W is done similarly too, by multiplying A on the right by $\mathbb{Z}^t_{m,0}\mathsf{Y}$, and has a similar cost estimate. Computing a and b is faster: it suffices to multiply A and B^t by a single vector, so the cost is merely $O(\gamma\,\mathsf{M}(q))$. □

We now prove Theorem 1. Let $(\mathsf{T}, \mathsf{U}, \mathsf{w}) \in \mathbb{K}^{n\times\alpha} \times \mathbb{K}^{n\times\alpha} \times \mathbb{K}^n$ be the input of problem $\mathsf{LinearSystem}(\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^t, \alpha)$. As in [28,29] we will reduce by randomization to a similar problem but with a regularized input $(\mathsf{Y}, \mathsf{Z}, \mathsf{v})$.

Let B be (implicitly) defined by $\phi_+(\mathsf{B}) = \mathsf{T}\mathsf{U}^t$, so that the system we want to solve is $\mathsf{B}\mathsf{t} = \mathsf{w}$. Define $\mathsf{A} = \mathbb{U}(\mathsf{y})\,\mathsf{B}\,\mathbb{L}(\mathsf{z})$ and $\mathsf{v} = \mathbb{U}(\mathsf{y})\,\mathsf{w}$, where y, z are random vectors in $\mathbb{K}^n$ with first entry equal to 1. Then, a vector t satisfies $\mathsf{B}\mathsf{t} = \mathsf{w}$ if and only if the vector $\mathsf{u} = \mathbb{L}(\mathsf{z})^{-1}\mathsf{t}$ satisfies $\mathsf{A}\mathsf{u} = \mathsf{v}$.

Using the proof of Proposition 4 and the simple structure of the matrices $\mathbb{U}(\mathsf{y})$ and $\mathbb{L}(\mathsf{z})$, one may check that a ϕ_+-generator (Y, Z) of length $\alpha + O(1)$ for A can be computed in time $O(\alpha\,\mathsf{M}(n))$. Besides, the vector t can be recovered from u in time $\mathsf{M}(n)$. Hence, we can focus on solving the latter problem $\mathsf{A}\mathsf{u} = \mathsf{v}$.

By Theorem 2 in [31], there exists a non-zero polynomial Γ of $2n-2$ variables and degree n^2+n, such that if $\Gamma(y_2, \ldots, y_n, z_2, \ldots, z_n) \neq 0$, the matrix A has generic rank profile.

Supposing that this condition is satisfied, let r be the rank of A and let $\mathsf{A}_r \in \mathbb{K}^{r\times r}$ be the largest non-singular leading principal submatrix of A. The next proposition bounds the cost of finding a ϕ_--generator of length α for A_r^{-1}. Then, using a third random vector of size n, Theorem 4 in [31] (see also [28, Proposition 3]) shows how to find a random solution to the system $\mathsf{A}\mathsf{u} = \mathsf{v}$, if one exists, in $O(\alpha\,\mathsf{M}(n))$ operations. This concludes the cost analysis of Theorem 1.

The probability analysis follows from the previous discussion: the Zippel-Schwartz lemma [14,55,50] shows that this algorithm has type $P(3n-2, n^2+n)$.

Proposition 5 *Assume that* A *has generic rank profile. Given a* ϕ_+*-generator of length* α *for* $\mathsf{A} \in \mathbb{K}^{n\times n}$*, one can compute its rank* r *as well as a* ϕ_-*-generator of length at most* α *for* A_r^{-1} *in time* $O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$.

Proof. We use Kaltofen's *Leading Principal Inverse* algorithm [28,29], which follows the ideas of Morf [37,38] and Bitmead-Anderson [7]; with Lemma 10, this algorithm becomes deterministic, as noted in [47, §7]. The proof of Theorem 3 in [28] shows that its cost is $T(\alpha, n) = O(\alpha^\omega)$ if $n \leq \alpha$ and otherwise

$$T(\alpha, n) = T(\alpha, \lceil n/2 \rceil) + T(\alpha, \lfloor n/2 \rfloor) + T_1(\alpha, n) + T_2(\alpha, n) + O(\alpha^{\omega-1}n + \alpha\mathsf{M}(n)). \quad (12)$$

Here the term in $O(\alpha^{\omega-1}n + \alpha\mathsf{M}(n))$ bounds the cost of some conversions between ϕ_+- and ϕ_--generators (Lemma 14) and the cost of finding generators

of minimal length (Lemma 10); the terms $T_1(\alpha, n)$ and $T_2(\alpha, n)$ are the costs of two tasks we shall describe now, after recalling some notation from [28].

With $n_1 = \lceil n/2 \rceil$, partition A as in Equation (11) and A_r as

$$\mathsf{A}_r = \begin{bmatrix} \mathsf{A}_{1,1} & \mathsf{A}'_{1,2} \\ \mathsf{A}'_{2,1} & \mathsf{A}'_{2,2} \end{bmatrix}.$$

Assume that $\mathsf{A}_{1,1}$ is non-singular (else, the cost is smaller) and let

$$\Delta = \mathsf{A}_{2,2} - \mathsf{A}_{2,1}\,\mathsf{A}_{1,1}^{-1}\,\mathsf{A}_{1,2} \quad \text{and} \quad \Delta' = \mathsf{A}'_{2,2} - \mathsf{A}'_{2,1}\,\mathsf{A}_{1,1}^{-1}\,\mathsf{A}'_{1,2}.$$

Given ϕ_+-generators of length $O(\alpha)$ for A and $\mathsf{A}_{1,1}^{-1}$, the first task is to compute a ϕ_+-generator for Δ. Using Lemmas 15 and 16 and Proposition 4, its cost is thus $T_1(\alpha, n) = O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$.

Given ϕ_+-generators of length $O(\alpha)$ for A, $\mathsf{A}_{1,1}^{-1}$ and Δ'^{-1}, the second task consists in computing a ϕ_+-generator for A_r^{-1}. Recall first that (see for example Theorem 5.2.3 in [45])

$$\mathsf{A}_r^{-1} = \begin{bmatrix} \mathsf{B}'_{1,1} & \mathsf{B}'_{1,2} \\ \mathsf{B}'_{2,1} & \Delta'^{-1} \end{bmatrix} \quad \text{with} \quad \begin{aligned} \mathsf{B}'_{1,2} &= -\mathsf{A}_{1,1}^{-1}\mathsf{A}'_{1,2}\Delta'^{-1} \\ \mathsf{B}'_{2,1} &= -\Delta'^{-1}\mathsf{A}'_{2,1}\mathsf{A}_{1,1}^{-1} \\ \mathsf{B}'_{1,1} &= \mathsf{A}_{1,1}^{-1} - \mathsf{B}'_{1,2}\mathsf{A}'_{2,1}\mathsf{A}_{1,1}^{-1}. \end{aligned}$$

Using again Lemmas 15 and 16 and Proposition 4, we get the same estimate as before: $T_2(\alpha, n) = O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$. In view of the recurrence relation in Equation (12), this implies that $T(\alpha, n) = O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$. □

4.3 Application: Padé-type approximation

We conclude by proving Corollary 1 on polynomial approximation. Let $M \in \mathbb{K}[x]$ be of degree n, let $f_1, \ldots, f_s \in \mathbb{K}[x]$ be of degrees less than n and let $\nu_1, \ldots, \nu_s$ be positive integers such that $\sum_{i \le s} \nu_i = n + 1$. We look for approximants $g_1, \ldots, g_s \in \mathbb{K}[x]$, not all zero, with $\deg(g_i) < \nu_i$ and such that $g_1 f_1 + \cdots + g_s f_s = 0 \bmod M$.

Write $M = \sum_{i=0}^{n} m_i x^i$, with $m_n = 1$ and let $\mathbb{X} = \mathbb{X}(M) \in \mathbb{K}^{n \times n}$ be the matrix of multiplication by x modulo M. For $i \le s$, define A_i as the Krylov matrix

$$\mathsf{A}_i = \begin{bmatrix} \mathsf{f}_i & \mathbb{X}\,\mathsf{f}_i & \cdots & \mathbb{X}^{\nu_i - 1}\,\mathsf{f}_i \end{bmatrix} \in \mathbb{K}^{n \times \nu_i},$$

where $\mathsf{f}_i = [f_{i,0} \ \cdots \ f_{i,n-1}]^t$ is the vector of coefficients of f_i. Let finally $\mathsf{A} = [\mathsf{A}_1 \ \cdots \ \mathsf{A}_s] \in \mathbb{K}^{n\times(n+1)}$ and $\mathsf{A}' \in \mathbb{K}^{(n+1)\times(n+1)}$ be the matrix obtained by padding A with an $(n+1)$st row full of 1's.

Since the right null space of A is non-trivial, the square system $\mathsf{A}'\mathsf{u} = [0 \cdots 0\, 1]^t$ admits a solution, and any such solution solves our problem. The following lemma shows the Toeplitz-like structure of the matrix A'; combining it with Theorem 1 proves Corollary 1.

Lemma 17 *Given M, $f_1, \ldots, f_s$ and $\nu_1, \ldots, \nu_s$ as above, one can compute a ϕ_+-generator of length $s+2$ for the matrix A' in time $O(s\,\mathsf{M}(n))$.*

Proof. Remark that $\mathbb{X} = \mathbb{Z}_{n,0} - \mathsf{m}\,\mathsf{e}_{n,n}^t$, with

$$\mathsf{m} = \begin{bmatrix} m_0 \\ \vdots \\ m_{n-1} \end{bmatrix} \in \mathbb{K}^n \quad \text{and} \quad \mathsf{e}_{n,n} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbb{K}^n;$$

since A has dimensions $n \times (n+1)$, it follows that

$$\phi_+(\mathsf{A}) = \mathsf{A} - \mathbb{X}\,\mathsf{A}\,\mathbb{Z}_{n+1,0}^t - \mathsf{m}\,\mathsf{e}_{n,n}^t\,\mathsf{A}\,\mathbb{Z}_{n+1,0}^t.$$

Let $\mathsf{a} = \mathsf{A}^t\,\mathsf{e}_{n,n} \in \mathbb{K}^{n+1}$ be the transpose of the last row of A and let $\mathsf{b} = \mathbb{Z}_{n+1,0}\,\mathsf{a}$. Then, the previous formula becomes $\phi_+(\mathsf{A}) = \mathsf{A} - \mathbb{X}\,\mathsf{A}\,\mathbb{Z}_{n+1,0}^t - \mathsf{m}\,\mathsf{b}^t$.

Taking $\mathsf{f}_0 = 0$ and $\nu_0 = 0$, we can write $\mathsf{A} - \mathbb{X}\,\mathsf{A}\,\mathbb{Z}_{n+1,0}^t$ as $\mathsf{Y}\,\mathsf{Z}^t$, with Y and Z of dimensions $n \times s$ and $(n+1) \times s$ given as follows: the ith column y_i of Y is $\mathsf{f}_i - \mathbb{X}^{\nu_{i-1}}\mathsf{f}_{i-1}$; the ith column z_i of Z is zero, except for a 1 at row $1 + \nu_1 + \cdots + \nu_{i-1}$. Since $\mathbb{X}^{\nu_{i-1}}\mathsf{f}_{i-1}$ is the coefficient vector of $x^{\nu_{i-1}} f_{i-1} \bmod M$, it can be computed in time $O(\mathsf{M}(n))$, so Y and Z can be computed in time $O(s\,\mathsf{M}(n))$.

To determine a ϕ_+-generator of A, it remains to determine the vector a, from which b follows easily. For $i \le s$, let $\mathsf{a}_i \in \mathbb{K}^{1\times\nu_i}$ be the last row of A_i, so that

$$\mathsf{a}_i = \left[\operatorname{coeff}(f_i, x^{n-1}) \ \cdots \ \operatorname{coeff}(x^{\nu_i-1} f_i \bmod M, x^{n-1})\right].$$

Define next

$$\mathsf{m}_i = \begin{bmatrix} m_n \\ \vdots \\ m_{n-\nu_i+1} \end{bmatrix} \in \mathbb{K}^{\nu_i} \quad \text{and} \quad \mathsf{f}_i' = \begin{bmatrix} f_{i,n-1} \\ \vdots \\ f_{i,n-\nu_i} \end{bmatrix} \in \mathbb{K}^{\nu_i}.$$

Noticing that $\mathbb{L}(\mathsf{m}_i)\,\mathsf{a}_i^t = \mathsf{f}_i'$, we see that the entries of a_i can be computed in

time $O(\mathsf{M}(\nu_i))$. Since $\sum_{i\le s}\nu_i = n+1$, the vector $\mathsf{a} = [\mathsf{a}_1\ \ldots\ \mathsf{a}_s]^t \in \mathbb{K}^{n+1}$ can thus be computed in time $O(\mathsf{M}(n))$.

In conclusion, one can compute $\mathsf{G} = [\,\mathsf{Y}\ \ -\mathsf{m}\,] \in \mathbb{K}^{n\times(s+1)}$ and $\mathsf{H} = [\,\mathsf{Z}\ \mathsf{b}\,] \in \mathbb{K}^{(n+1)\times(s+1)}$ in time $O(s\,\mathsf{M}(n))$, such that (G,H) is a ϕ_+-generator of length $s+1$ for A. One then obtains a ϕ_+-generator of length $s+2$ for A' by adjoining a last row of zeros to G and the columns

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbb{K}^{n+1} \quad \text{and} \quad \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} - \mathsf{b} \in \mathbb{K}^{n+1}$$

to respectively G and H. □

5 The Vandermonde case

In this section, $\mathsf{x} \in \mathbb{K}^n$ and $\psi \in \mathbb{K}$ are as in Equations (1) and (2). The operator associated with the Vandermonde structure is

$$\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}](\mathsf{A}) = \mathsf{A} - \mathbb{D}(\mathsf{x})\,\mathsf{A}\,\mathbb{Z}^t_{n,\psi}.$$

With our choice of ψ, Theorem 4.3.2 in [45] shows that this operator is invertible. Moreover, given Y, Z in $\mathbb{K}^{n\times\alpha}$, Example 4.4.6 in [45] shows that the unique matrix $\mathsf{A} \in \mathbb{K}^{n\times n}$ such that $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}](\mathsf{A}) = \mathsf{Y}\,\mathsf{Z}^t$ is

$$\mathsf{A} = \mathbb{D}\Big((1-\psi\,\mathsf{x}^n)^{-1}\Big) \sum_{i=1}^{\alpha} \mathbb{D}(\mathsf{y}_i)\,\mathbb{V}(\mathsf{x},n)\,\mathbb{T}(\mathsf{z}_i,\psi)^t. \tag{13}$$

Following Pan's idea [41,42], we will prove Theorem 2 on the complexity of solving Vandermonde-like systems of the form $\mathsf{Au} = \mathsf{v}$ by turning them into Toeplitz-like ones.

We will use the same kind of reduction as in [24]. However, that approach requires the entries of x to be pairwise distinct, *i.e.*, that $\mathbb{V}(\mathsf{x},n)$ be invertible; else, the preprocessing step in [24, Section 2] fails. Similarly, the reduction in [45, Example 4.8.4] does not solve the problem when $\mathbb{V}(\mathsf{x},n)$ is singular (in the application of Subsection 5.4, this invertibility assumption does not hold).

The partition of x of Equation (1) will be used to solve this problem in cases when x has low multiplicity: this is exposed in Subsections 5.1 and 5.2. When the multiplicity becomes too large, some extra work is needed, presented in Subsection 5.3.

We start by solving a preliminary subproblem. Let Y and Z be in $\mathbb{K}^{n\times\alpha}$, and let A be the unique $n \times n$ matrix such that $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}](\mathsf{A}) = \mathsf{Y}\,\mathsf{Z}^t$. Splitting A along its rows according to the given partition of x, we thus write

$$\mathsf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \vdots \\ \mathsf{x}_s \end{bmatrix}, \text{ with } \mathsf{x}_j \in \mathbb{K}^{\nu_j} \quad \text{and} \quad \mathsf{A} = \begin{bmatrix} \mathsf{A}_1 \\ \vdots \\ \mathsf{A}_s \end{bmatrix}, \quad \text{with } \mathsf{A}_j \text{ in } \mathbb{K}^{\nu_j \times n}, \tag{14}$$

where all x_j are repetition-free. Given vectors $\mathsf{w}_1, \ldots, \mathsf{w}_s$, with w_j in $\mathbb{K}^{\nu_j}$, we study in this subsection the cost of computing all products $\mathsf{A}_j^t\mathsf{w}_j \in \mathbb{K}^n$, using the results of Subsection 3.2. This will be the key in our reduction of Vandermonde-like systems to Toeplitz-like ones.

Proposition 6 *Given* x, ψ, Y, Z *and* $\mathsf{w}_1, \ldots, \mathsf{w}_s$ *as above, and assuming that* $s \leq \alpha$, *one can compute all products* $\mathsf{A}_j^t\,\mathsf{w}_j$ *in time* $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$.

Proof. Let y_i and z_i be the columns of Y and Z. We adapt the partition of x and A to the vectors y_i, writing

$$\mathsf{y}_i = \begin{bmatrix} \mathsf{y}_{i,1} \\ \vdots \\ \mathsf{y}_{i,s} \end{bmatrix}, \quad \text{with } \mathsf{y}_{i,j} \text{ in } \mathbb{K}^{\nu_j}.$$

Since A is given by Equation (13), its submatrices A_j are given by $\mathsf{A}_j = \mathbb{D}\Big((1-\psi\,\mathsf{x}_j^n)^{-1}\Big)\,\mathsf{B}_j$, with

$$\mathsf{B}_j = \sum_{i=1}^{\alpha} \mathbb{D}(\mathsf{y}_{i,j})\,\mathbb{V}(\mathsf{x}_j, n)\,\mathbb{T}(\mathsf{z}_i, \psi)^t$$

and thus

$$\mathsf{B}_j^t = \sum_{i=1}^{\alpha} \mathbb{T}(\mathsf{z}_i, \psi)\,\mathbb{V}(\mathsf{x}_j, n)^t\,\mathbb{D}(\mathsf{y}_{i,j}). \tag{15}$$

For $j \leq s$, let $\mathsf{f}_j = \mathbb{D}((1-\psi\,\mathsf{x}_j^n)^{-1})\,\mathsf{w}_j$. The vectors f_j can be deduced from x, ψ and the w_j in $O(n\log(n))$ operations. Since we have $\mathsf{A}_j^t\,\mathsf{w}_j = \mathsf{B}_j^t\,\mathsf{f}_j$, we are thus left with computing all the products $\mathsf{B}_j^t\,\mathsf{f}_j$.

Recalling that $\mathsf{x}_j \in \mathbb{K}^{\nu_j}$ is repetition-free, we let $Y_{i,j} = \mathsf{Interp}(\mathsf{x}_j, \mathsf{y}_{i,j}) \in \mathbb{K}[x]_{\nu_j}$. Applying Lemma 1 to the right hand side of Equation (15) then gives

$$\mathsf{B}_j^t = \sum_{i=1}^{\alpha} \mathbb{T}(\mathsf{z}_i, \psi)\,\mathbb{M}(Y_{i,j}, n)^t\,\mathbb{V}(\mathsf{x}_j, n+\nu_j-1)^t.$$

We can now factor out the rightmost transposed Vandermonde matrices, which do not depend on the summation index i. Defining

$$\mathsf{f}'_j = \mathbb{V}(\mathsf{x}_j, n+\nu_j-1)^t\,\mathsf{f}_j,$$

we deduce that

$$\mathsf{B}_j^t\,\mathsf{f}_j = \sum_{i=1}^{\alpha} \mathbb{T}(\mathsf{z}_i,\psi)\,\mathbb{M}(Y_{i,j},n)^t\,\mathsf{f}'_j.$$

By fast application of a transposed Vandermonde matrix, each vector f'_j can be computed in time $O(\mathsf{M}(n)\log(n))$; hence, the total time for their computation is $O(\alpha\,\mathsf{M}(n)\log(n))$.

For $i \le \alpha$ and $j \le s$, define the vector $\mathsf{g}_{i,j}$ in $\mathbb{K}^n$ by

$$\mathsf{g}_{i,j} = \mathbb{T}(\mathsf{z}_i,\psi)\,\mathbb{M}(Y_{i,j},n)^t\,\mathsf{f}'_j,$$

so that $\mathsf{B}_j^t\mathsf{f}_j = \sum_{i=1}^{\alpha}\mathsf{g}_{i,j}$. We will obtain the vectors $\mathsf{g}_{i,j}$ by means of Lemma 6. To do so, define the polynomials

$$Y'_{i,j} = \mathsf{Rev}_{\nu_j}(Y_{i,j}),\quad F_j = \mathsf{Pol}(\mathsf{f}'_j)\quad\text{and}\quad F'_j = \mathsf{Rev}_{n+\nu_j-1}(F_j).$$

Let also $Z_i = \mathsf{Pol}(\mathsf{z}_i)$ and $Z'_i = \mathsf{Pol}(\mathsf{Flip}(\mathsf{z}_i))$. Lemma 6 then gives

$$\begin{aligned}\mathsf{Pol}(\mathsf{g}_{i,j}) = Z_i\,(Y'_{i,j}\,F_j \text{ div } x^{\nu_j-1}) \bmod x^n \\ + \psi\,\mathsf{Rev}_n\Big(Z'_i\,(Y_{i,j}\,F'_j \text{ div } x^{\nu_j-1}) \bmod x^n\Big).\end{aligned}$$

Summing over i eventually yields our output

$$\begin{aligned}\mathsf{Pol}\Big(\mathsf{B}_j^t\mathsf{f}_j\Big) = \sum_{i=1}^{\alpha}\mathsf{Pol}(\mathsf{g}_{i,j}) = \sum_{i=1}^{\alpha} Z_i\,(Y'_{i,j}\,F_j \text{ div } x^{\nu_j-1}) \bmod x^n \\ + \psi\,\mathsf{Rev}_n\Big(\sum_{i=1}^{\alpha} Z'_i\,(Y_{i,j}\,F'_j \text{ div } x^{\nu_j-1}) \bmod x^n\Big). \quad (16)\end{aligned}$$

It remains to perform the cost estimate, using the results recalled in Section 2.

- Using fast interpolation, we compute each $Y_{i,j}$ in time $O(\mathsf{M}(\nu_j)\log(\nu_j))$ and thus all $Y_{i,j}$ and $Y'_{i,j}$ in time $O(\alpha\,\mathsf{M}(n)\log(n))$.
- Applying Proposition 3 to both summands in Equation (16) shows that all polynomials $\mathsf{Pol}(\mathsf{B}_j^t\mathsf{f}_j)$ can be computed in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$, which concludes the proof. □

5.2 The case of low multiplicities

In this subsection, we reduce the resolution of Vandermonde-like systems to that of Toeplitz-like systems. We adapt the reduction of [24], allowing now for

repetitions in x. For the moment, we work in the case where the multiplicity of x is bounded by α.

Proposition 7 *Let x and ψ be as in Equations (1) and (2). If the multiplicity s of x satisfies $s \leq \alpha$, then the problem $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}, \alpha)$ can be solved in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$. The algorithm is probabilistic of type $P(3n-2, n^2+n)$.*

Proof. Given Y and Z in $\mathbb{K}^{n\times\alpha}$ and v in $\mathbb{K}^n$, we are looking for solutions u to the system $\mathsf{Au} = \mathsf{v}$, where A is such that $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}](\mathsf{A}) = \mathsf{Y}\,\mathsf{Z}^t$.

For $j \leq s$, let M_j be the monic polynomial $M_j = \prod_{a\in\mathsf{x}_j}(x-a)$. Since x_j is repetition-free, Lemma 3 shows that

$$\mathbb{D}(\mathsf{x}_j) = \mathbb{V}(\mathsf{x}_j, \nu_j)\,\mathbb{X}(M_j)\,\mathbb{V}(\mathsf{x}_j, \nu_j)^{-1}.$$

It follows that

$$\mathbb{D}(\mathsf{x}) = \mathbb{V}\,\mathbb{X}\,\mathbb{V}^{-1}, \tag{17}$$

where $\mathbb{V}$ and $\mathbb{X}$ are block-diagonal with respective blocks $\mathbb{V}(\mathsf{x}_j, \nu_j)$ and $\mathbb{X}(M_j)$.

For $j \leq s$, let $\mathsf{m}_j \in \mathbb{K}^{\nu_j}$ be the coefficient vector of $-M_j \bmod x^{\nu_j}$, so that $\mathbb{X}(M_j) = \mathbb{Z}_{\nu_j,0} + \mathsf{m}_j\mathsf{e}^t_{\nu_j,\nu_j}$. Hence, writing $\nu^*_j = \nu_1 + \cdots + \nu_j$ as in Section 2,

$$\mathbb{X} = \mathbb{Z}_{n,0} + \sum_{j=1}^{s} \mathsf{g}_j\mathsf{e}^t_{n,\nu^*_j}, \tag{18}$$

where for $j \leq s$, $\mathsf{g}_j \in \mathbb{K}^n$ is obtained by padding m_j with ν^*_{j-1} zeros on the top and, if $j \neq s$, with -1 followed by $n - \nu^*_j - 1$ zeros on the bottom.

Defining $\mathsf{B} = \mathbb{V}^{-1}\mathsf{A}$ and $\mathsf{v}' = \mathbb{V}^{-1}\mathsf{v}$, solving $\mathsf{Au} = \mathsf{v}$ amounts to solve $\mathsf{Bu} = \mathsf{v}'$. To do so in the claimed complexity, we exhibit the Toeplitz-like structure of B and bound the cost of computing v' and a generator for B. Pre-multiplying by $\mathbb{V}^{-1}$ the relation

$$\mathsf{A} - \mathbb{D}(\mathsf{x})\,\mathsf{A}\,\mathbb{Z}^t_{n,\psi} = \mathsf{Y}\,\mathsf{Z}^t,$$

we get

$$\mathsf{B} - \mathbb{V}^{-1}\,\mathbb{D}(\mathsf{x})\,\mathbb{V}\,\mathsf{B}\,\mathbb{Z}^t_{n,\psi} = \mathbb{V}^{-1}\,\mathsf{Y}\,\mathsf{Z}^t;$$

using Equation (17), we rewrite this as

$$\mathsf{B} - \mathbb{X}\,\mathsf{B}\,\mathbb{Z}^t_{n,\psi} = \mathsf{Y}'\,\mathsf{Z}^t, \quad \text{with } \mathsf{Y}' = \mathbb{V}^{-1}\,\mathsf{Y}.$$

Then, from (18) and the relation $\mathbb{Z}_{n,\psi} = \mathbb{Z}_{n,0} + \psi\,\mathsf{e}_{n,1}\mathsf{e}^t_{n,n}$, we deduce that

$$\mathsf{B} - \mathbb{Z}_{n,0}\,\mathsf{B}\,\mathbb{Z}^t_{n,0} \;=\; \psi\,\mathbb{Z}_{n,0}\,\mathsf{B}\,\mathsf{e}_{n,n}\mathsf{e}^t_{n,1} \;+\; \Big(\sum_{j=1}^{s} \mathsf{g}_j\mathsf{e}^t_{n,\nu^*_j}\Big)\mathsf{B}\,\mathbb{Z}^t_{n,\psi} \;+\; \mathsf{Y}'\,\mathsf{Z}^t.$$

Define the vectors $\mathsf{f}_1 = \psi\,\mathbb{Z}_{n,0}\,\mathsf{B}\,\mathsf{e}_{n,n}$ and, for $j \leq s$, $\mathsf{h}_j = \mathsf{B}^t\mathsf{e}_{n,\nu_j^*}$ and $\mathsf{h}'_j = \mathbb{Z}_{n,\psi}\mathsf{h}_j$. The above formula then becomes

$$\Delta[\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^t](\mathsf{B}) = \mathsf{f}_1\mathsf{e}_{n,1}^t + \mathsf{G}\mathsf{H}'^t + \mathsf{Y}'\mathsf{Z}^t,$$

where G (resp. H') has columns g_j (resp. h'_j). The matrices $[\mathsf{f}_1\ \mathsf{G}\ \mathsf{Y}']$ and $[\mathsf{e}_{n,1}\ \mathsf{H}'\ \mathsf{Z}]$ thus form a $\mathbb{Z}_{n,0}, \mathbb{Z}_{n,0}^t$-generator of length $\alpha + s + 1 \leq 2\alpha + 1$ for B. Once this generator and v' are known, the system $\mathsf{B}\mathsf{u} = \mathsf{v}'$ can be solved within the prescribed complexity by the probabilistic algorithm of Theorem 1.

It remains to estimate the cost of computing $\mathsf{v}', \mathsf{f}_1, \mathsf{Y}', \mathsf{G}$ and H'. We will do so using Proposition 6 as well as the results recalled in Section 2 on the complexity of polynomial operations.

As a first step, though, we detail further the structure of the vectors h_j. For $j \leq s$, one has $\mathsf{h}_j = \mathsf{B}^t\mathsf{e}_{n,\nu_j^*} = \mathsf{A}^t\,\mathbb{V}^{-t}\,\mathsf{e}_{n,\nu_j^*}$. In view of the block structures of A^t and $\mathbb{V}^{-1}$, this can be rewritten as $\mathsf{h}_j = \mathsf{A}_j^t\,\mathsf{w}_j$, where $\mathsf{w}_j = \mathbb{V}(\mathsf{x}_j, \nu_j)^{-t}\,\mathsf{e}_{\nu_j,\nu_j}$ is the last column of $\mathbb{V}(\mathsf{x}_j, \nu_j)^{-t}$.

- All polynomials M_j (and thus all vectors g_j and the matrix G) can be constructed from their roots in $O(\mathsf{M}(n)\log(n))$ operations.
- In view of Equation (13), one can multiply A by the vector $\mathsf{e}_{n,n}$ in time $O(\alpha\,\mathsf{M}(n)\log(n))$, see e.g. [24, Section 2].
- Since multiplication by $\mathbb{V}(\mathsf{x}_j, \nu_j)^{-1}$ has cost $O(\mathsf{M}(\nu_j)\log(\nu_j))$, multiplication by $\mathbb{V}^{-1}$ has cost $O(\mathsf{M}(n)\log(n))$. This implies that $\mathsf{f}_1 = \psi\,\mathbb{Z}_{n,0}\,\mathbb{V}^{-1}\,\mathsf{A}\,\mathsf{e}_{n,n}$ can be deduced from $\mathsf{A}\,\mathsf{e}_{n,n}$ in time $O(\mathsf{M}(n)\log(n))$. Similarly, one can compute $\mathsf{Y}' = \mathbb{V}^{-1}\mathsf{Y}$ in time $O(\alpha\,\mathsf{M}(n)\log(n))$ and $\mathsf{v}' = \mathbb{V}^{-1}\mathsf{v}$ in time $O(\mathsf{M}(n)\log(n))$.
- Computing the last column of $\mathbb{V}(\mathsf{x}_j, \nu_j)^{-t}$ takes time $O(\mathsf{M}(\nu_j)\log(\nu_j))$, which induces a cost of $O(\mathsf{M}(n)\log(n))$ for finding all vectors w_j.
- Knowing all w_j, Proposition 6 shows that all vectors h_j can be computed in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$. Deducing the vectors h'_j takes time $O(\alpha\,n)$, which concludes the proof. □

5.3 The case of high multiplicities

We conclude the proof of Theorem 2 by considering the case of high multiplicities ($s > \alpha$), reducing it to the case of low multiplicities ($s \leq \alpha$) seen in Subsection 5.2. Our reduction has cost $O(\alpha^{\omega-1}\,n)$, which fits in the requested bound. It however introduces an extra probabilistic aspect; combined with the one of Proposition 7, it yields the overall probability estimate of Theorem 2.

Proposition 8 *Let* x *and* ψ *be as in Equations* (1) *and* (2). *If* x *has multi-*

plicity larger than α, one can reduce the problem $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}, \alpha)$ *to the problem* $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{y}), \mathbb{Z}^t_{n,\psi}, \alpha)$*, where* $\mathsf{y} \in \mathbb{K}^n$ *has multiplicity at most* α *and satisfies the constraints of Equations* (1) *and* (2)*. The reduction can be done in time* $O(\alpha^{\omega-1}n)$ *by a probabilistic algorithm of type* $P(n, 3n^2)$*.*

Proof. Given Y and Z in $\mathbb{K}^{n\times\alpha}$ and v in $\mathbb{K}^n$, we are looking for solutions u to the system $\mathsf{Au} = \mathsf{v}$, where A is such that $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}](\mathsf{A}) = \mathsf{Y}\,\mathsf{Z}^t$. We assume that Y and Z have full rank; if this is not the case, we can replace (Y, Z) by a minimal-length generator, whose two matrices then have full rank.

We start by reordering the entries of x to obtain a repetition-free decomposition as in Equation (6) of Section 2:

$$\mathsf{x}' = \begin{bmatrix} x_{\sigma(1)} \\ \vdots \\ x_{\sigma(n)} \end{bmatrix} = \begin{bmatrix} \mathsf{x}'_1 \\ \vdots \\ \mathsf{x}'_t \end{bmatrix},$$

where x'_i is a vector consisting of μ_i repetitions of the same element ξ_i, so that $n = \mu_1 + \cdots + \mu_t$, with $\xi_i \neq \xi_j$ for $i \neq j$ and $\mu_1 \geq \cdots \geq \mu_t > 0$.

Let v', A' and Y' be obtained by applying the same reordering to the entries of v and to the rows of A and Y; hence, $(\mathsf{Y}', \mathsf{Z})$ is a $\mathbb{D}(\mathsf{x}'), \mathbb{Z}^t_{n,\psi}$-generator for A'. Since the solution sets of $\mathsf{Au} = \mathsf{v}$ and $\mathsf{A}'\mathsf{u} = \mathsf{v}'$ are the same, we focus on the latter problem.

By construction, the matrices A' and Y' admit the following decompositions:

$$\mathsf{A}' = \begin{bmatrix} \mathsf{A}'_1 \\ \vdots \\ \mathsf{A}'_t \end{bmatrix} \quad \text{and} \quad \mathsf{Y}' = \begin{bmatrix} \mathsf{Y}'_1 \\ \vdots \\ \mathsf{Y}'_t \end{bmatrix},$$

with A'_i in $\mathbb{K}^{\mu_i\times n}$ and Y'_i in $\mathbb{K}^{\mu_i\times\alpha}$. Remark then that $\mathsf{A}'_i - \mathbb{D}(\mathsf{x}'_i)\,\mathsf{A}'_i\,\mathbb{Z}^t_{n,\psi} = \mathsf{Y}'_i\,\mathsf{Z}^t$. Now, $\mathbb{D}(\mathsf{x}'_i)$ equals $\xi_i\mathbb{I}_{\mu_i}$ and, since $\psi{\xi_i}^n \neq 1$ for all i, all matrices $\mathbb{I}_n - \xi_i\mathbb{Z}^t_{n,\psi}$ are invertible. We thus obtain the equalities

$$\mathsf{A}'_i = \mathsf{Y}'_i\mathsf{Z}^t(\mathbb{I}_n - \xi_i\mathbb{Z}^t_{n,\psi})^{-1} \quad \text{for } 1 \leq i \leq t. \tag{19}$$

We will use dense matrix methods to reduce the number of non-zero entries in A' and Y', while maintaining a Vandermonde-like structure. Let thus τ be such that $\mu_\tau > \alpha \geq \mu_{\tau+1}$.

- For $1 \leq i \leq \tau$, we have $\mu_i > \alpha$. We let $J_i \subset \{1, \ldots, \mu_i\}$ and $\mathsf{E}_i \in \mathbb{K}^{\mu_i\times\mu_i}$ be the index set and the matrix obtained by applying Lemma 9 to Y'_i. Let also

$r_i = |J_i| = \mathrm{rank}(\mathsf{Y}'_i)$, so that $r_i \leq \alpha$. Lemma 9 and Equation (19) then give

$$\mathsf{E}_i \, \mathsf{Y}'_i = \begin{bmatrix} \mathsf{Y}''_i \\ 0 \end{bmatrix} \quad \text{and} \quad \mathsf{E}_i \, \mathsf{A}'_i = \begin{bmatrix} \mathsf{A}''_i \\ 0 \end{bmatrix},$$

where Y''_i and A''_i have respective sizes $(r_i \times \alpha)$ and $(r_i \times n)$, and Y''_i consists of the rows of Y'_i indexed by J_i.

- For $\tau < i \leq t$, we have $\mu_i \leq \alpha$. We let $\mathsf{E}_i = \mathbb{I}_{\mu_i}$, $\mathsf{Y}''_i = \mathsf{Y}'_i$ and $\mathsf{A}''_i = \mathsf{A}'_i$.

Let E be the block-diagonal matrix having $\mathsf{E}_1, \ldots, \mathsf{E}_t$ on the diagonal. Hence, $\mathsf{Y}'' = \mathsf{E}\,\mathsf{Y}'$ consists of the matrices Y''_i, interleaved by blocks of zeros when $i \leq \tau$; the same holds for $\mathsf{A}'' = \mathsf{E}\,\mathsf{A}'$. Besides, since each $\mathbb{D}(\mathsf{x}'_i)$ is a homothety matrix, it commutes with E_i; we deduce that $(\mathsf{Y}'', \mathsf{Z})$ is a $\mathbb{D}(\mathsf{x}'), \mathbb{Z}^t_{n,\psi}$-generator for A''. Finally, since E is invertible, the solution sets of $\mathsf{A}'\mathsf{u} = \mathsf{v}'$ and $\mathsf{A}''\mathsf{u} = \mathsf{v}''$ coincide, where we wrote $\mathsf{v}'' = \mathsf{E}\,\mathsf{v}'$.

We solve the latter problem, by exhibiting the Vandermonde-like structure of A'' for a modified displacement operator with lower multiplicity. Define a vector $\mathsf{x}'' \in \mathbb{K}^n$ by replacing, for $i \leq \tau$, the last $\mu_i - r_i$ entries of x'_i by new values taken from $\mathbb{K}$. Due to the presence of corresponding blocks of zeros in Y'' and A'', the matrices Y'' and Z now form a $\mathbb{D}(\mathsf{x}''), \mathbb{Z}^t_{n,\psi}$-generator for A''.

With $r = \sum_{i \leq \tau}(\mu_i - r_i)$, suppose that the new values $y_1, \ldots, y_r$ inserted in x'' are pairwise distinct and that none of them belongs to x' or satisfies $\psi \, y_i^n = 1$. Then, x'' satisfies the constraint of Equation (2) and has multiplicity at most α (since all r_i are at most α). It remains to reorder the entries of x'' to obtain a vector y that also satisfies Equation (1), and let Σ be the permutation matrix such that $\mathsf{y} = \Sigma\,\mathsf{x}''$. Defining $\mathsf{U} = \Sigma\,\mathsf{Y}''$, it follows that (U, Z) is a $\mathbb{D}(\mathsf{y}), \mathbb{Z}^t_{n,\psi}$-generator for the matrix $\mathsf{B} = \Sigma\,\mathsf{A}''$. To conclude, let $\mathsf{w} = \Sigma\,\mathsf{v}''$; then, the solution sets of $\mathsf{A}''\mathsf{u} = \mathsf{v}''$ and $\mathsf{B}\mathsf{u} = \mathsf{w}$ coincide.

We have thus reduced solving the system $\mathsf{A}\mathsf{u} = \mathsf{v}$ to solving $\mathsf{B}\mathsf{u} = \mathsf{w}$, while providing a generator for B with respect to the Vandermonde-like structure $\mathbb{D}(\mathsf{y}), \mathbb{Z}^t_{n,\psi}$. It only remains to perform the complexity analysis.

- By Lemma 10, the cost of making the input generator minimal is $O(\alpha^{\omega-1}n)$.
- The permutation σ that gives x' can be computed in time $O(n)$ by Lemma 11.
- For $i \leq \tau$, since Y'_i is in $\mathbb{K}^{\mu_i \times \alpha}$ with $\mu_i \geq \alpha$, one can compute E_i in time $O(\alpha^{\omega-1}\mu_i)$ by Lemma 9. Since $\sum_{i=1}^{\tau} \mu_i \leq n$, the total cost is in $O(\alpha^{\omega-1}n)$.
- Since each matrix E_i is given in the form $(r_i, J_i, \mathsf{G}_i, \mathsf{P}_i)$ of Lemma 9, and thus has $O(\alpha\mu_i)$ non-zero entries, v'' can be deduced from v' in $O(\alpha n)$ operations.
- To put x'' into the form of Equation (1), we first put it into the repetition-free form of Equation (6). Since the repeated entries are already grouped

together (and thus the multiplicities are known), all we have to do is to sort the multiplicities in decreasing order; this can be done in time $O(n)$ using bucket sorting. Then, Lemma 11 puts the result into the form of Equation (1) in time $O(n)$.

- All other operations amount to apply permutations to the entries of matrices and vectors and have negligible cost.

The probability analysis comes by remarking that the values $y_1, \ldots, y_r$ satisfy our requirements if they do not cancel the polynomial

$$\delta = \prod_{i<j\leq r} (Y_i - Y_j) \;\times\; \prod_{i\leq r,\, j\leq t} (Y_i - \xi_j) \;\times\; \prod_{i\leq r} (Y_i^n \psi - 1), \tag{20}$$

which, since r and t are bounded by n, has degree at most $3n^2$. The Zippel-Schwartz lemma gives the required probability estimate. □

5.4 Application: bivariate interpolation

We conclude by proving Corollary 2 on the complexity of bivariate interpolation. Let $P_1, \ldots, P_s$ be the lists of sample points

$$\begin{aligned} P_1 &= [\, p_{1,1} = (x_1, y_{1,1}), \quad \ldots, \quad p_{1,\nu_1} = (x_1, y_{1,\nu_1}) \,], \\ &\vdots \\ P_s &= [\, p_{s,1} = (x_s, y_{s,1}), \quad \ldots, \quad p_{s,\nu_s} = (x_s, y_{s,\nu_s}) \,], \end{aligned}$$

with $\nu_1 \geq \cdots \geq \nu_s > 0$. Given a vector of values $\mathsf{v} = [v_{i,j}] \in \mathbb{K}^n$, with $1 \leq i \leq s$ and $1 \leq j \leq \nu_i$, there exists a unique polynomial F in $\mathbb{K}[x, y]$ of the form

$$F = \sum_{0\leq i<s,\, 0\leq j<\nu_{i+1}} f_{i,j} x^i y^j$$

such that $F(p_{i,j}) = v_{i,j}$ for all i, j. To recover F, we will use the Vandermonde-like structure of the corresponding linear system.

The support of F is thus the set of all monomials $x^i y^j$, with $0 \leq i < s$ and $0 \leq j < \nu_{i+1}$. To arrange these monomials in a suitable order, we let $\mu = \mu_1 \geq \cdots \geq \mu_t$ be the conjugate partition of ν, with $s = \mu_1$ and $t = \nu_1$; for $1 \leq j \leq t$, we define

$$B_j = [x^{i-1} y^{j-1} \mid 1 \leq i \leq \mu_j]$$

and let B be the concatenation of the lists B_j. Thus, for a fixed j, the entries of B_j have the same degree in y and increasing degrees in x.

It will be convenient to rearrange the lists of sample points, by "transposing" the input lists $P_1, \ldots, P_s$ to obtain

$$\begin{aligned} Q_1 &= [\,p_{1,1}, \quad p_{2,1}, \quad \ldots, \quad p_{\mu_1,1}\,], \\ Q_2 &= [\,p_{1,2}, \quad p_{2,2}, \quad \ldots, \quad p_{\mu_2,2}\,], \\ &\vdots \\ Q_t &= [\,p_{1,t}, \quad p_{2,t}, \quad \ldots, \quad p_{\mu_t,t}\,]; \end{aligned}$$

we then let Q be the concatenation of $Q_1, \ldots, Q_t$. Taking the x-coordinates of the elements in Q, we obtain a vector x in $\mathbb{K}^n$; by construction, x is in the form of Equation (1). Let further $\mathsf{w} \in \mathbb{K}^n$ be obtained by rearranging v in the same way. By Lemma 11, one can deduce Q and w from P and v in time $O(n)$.

Let $\mathsf{Span}(B) \subset \mathbb{K}[x, y]$ be the vector space generated by B; we are thus concerned by the evaluation map $F \in \mathsf{Span}(B) \mapsto [F(p)]_{p \in Q}$ and its inverse. Let

$$\mathsf{A} = \Big[b(p)\Big]_{p \in Q,\ b \in B} \in \mathbb{K}^{n \times n}$$

be the matrix of this map, with rows indexed by Q and columns by B. Hence,

$$\mathsf{A} = \Big[\mathsf{A}_1 \ \cdots \ \mathsf{A}_t\Big], \quad \text{with} \quad \mathsf{A}_j = \Big[b(p)\Big]_{p \in Q,\ b \in B_j} \in \mathbb{K}^{n \times \mu_j}.$$

Then $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,0}](\mathsf{A})$ can be written $\mathsf{G}\mathsf{H}^t$, with G and H in $\mathbb{K}^{n \times t}$; the jth column of H is zero, except for a 1 at row $1 + \mu^*_{j-1}$; the jth column of G is

$$\Big[1 \ \cdots \ 1\Big]^t \ \text{(for } j = 1) \quad \text{or} \quad \Big[y(p)^{j-1} \ - \ y(p)^{j-2}\, x(p)^{\mu_{j-1}}\Big]^t_{p \in Q} \ \text{(for } j > 1),$$

with $x(p)$ the x-coordinate of $p \in Q$ and $y(p)$ its y-coordinate. Given Q, the matrices G and H can be computed in time $O(t\, n \log(n))$; Theorem 2 then shows that the system $\mathsf{A}\,\mathsf{f} = \mathsf{w}$ can be solved in time $O(t^{\omega-1}\mathsf{M}(n) \log^2(n))$, where f is the coefficient vector of the polynomial to interpolate and w is the rearranged value vector. Remembering that $t = \nu_1$ concludes the proof.

6 The Cauchy case

In this section, $\mathsf{x}, \mathsf{y} \in \mathbb{K}^n$ are as in Equations (1), (4) and (5). The operator associated with the Cauchy structure that we consider here is

$$\Delta[\mathbb{D}(\mathsf{x}), \mathbb{D}(\mathsf{y})](\mathsf{A}) = \mathsf{A} - \mathbb{D}(\mathsf{x})\, \mathsf{A}\, \mathbb{D}(\mathsf{y}).$$

It follows from Equation (5) and [45, Theorem 4.3.2] that such an operator is invertible. Moreover, given G and H in $\mathbb{K}^{n\times\alpha}$, Example 4.4.7 in [45] shows that the unique matrix $\mathsf{A} \in \mathbb{K}^{n\times n}$ such that $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{D}(\mathsf{y})](\mathsf{A}) = \mathsf{G}\mathsf{H}^t$ is

$$\mathsf{A} = \sum_{i=1}^{\alpha} \mathbb{D}(\mathsf{g}_i)\, \mathbb{C}(\mathsf{x}, \mathsf{y})\, \mathbb{D}(\mathsf{h}_i). \tag{21}$$

We will prove Theorem 3 on the complexity of solving Cauchy-like systems of the form $\mathsf{A}\mathsf{u} = \mathsf{v}$ by turning them into Vandermonde-like ones with low multiplicity, and then using the result of Proposition 7. Our reduction follows the one in [24, Section 3] but adds the possibility of handling repetitions among the entries of the vectors x and y above.

The organization of this section is very similar to that of the previous one; the technical arguments are in the same vein as well. We shall start in Subsection 6.1 with solving another multiplication problem, in the spirit of the one used for the Vandermonde case. Together with the partitions of x and y given in Equations (1) and (4), this allows to handle in Subsection 6.2 the cases where multiplicities are small. Large multiplicities are treated in Subsection 6.3 with the same tools as for the Vandermonde case.

6.1 A multiplication problem

Let G and H be in $\mathbb{K}^{n\times\alpha}$, and let A be the unique $n \times n$ matrix such that $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{D}(\mathsf{y})](\mathsf{A}) = \mathsf{G}\mathsf{H}^t$. Splitting A along its rows according to the partition of x and along its columns according to the partition of y, we thus write

$$\mathsf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \vdots \\ \mathsf{x}_s \end{bmatrix}, \text{ with } \mathsf{x}_j \in \mathbb{K}^{\nu_j}, \qquad \mathsf{y} = \begin{bmatrix} \mathsf{y}_1 \\ \vdots \\ \mathsf{y}_t \end{bmatrix}, \text{ with } \mathsf{y}_k \in \mathbb{K}^{\delta_k},$$

and

$$\mathsf{A} = \begin{bmatrix} \mathsf{A}_{1,1} & \cdots & \mathsf{A}_{1,t} \\ \vdots & & \vdots \\ \mathsf{A}_{s,1} & \cdots & \mathsf{A}_{s,t} \end{bmatrix}, \quad \text{with } \mathsf{A}_{j,k} \text{ in } \mathbb{K}^{\nu_j\times\delta_k}. \tag{22}$$

Given vectors $\mathsf{w}_1, \ldots, \mathsf{w}_t$, with $\mathsf{w}_k \in \mathbb{K}^{\delta_k}$, we study in this subsection the cost of computing all the products $\mathsf{A}_{j,k}\,\mathsf{w}_k \in \mathbb{K}^{\nu_j}$ when both s and t are bounded by α. The result below, which relies on Proposition 2 of Section 3, will be the key for reducing Cauchy-like systems to Vandermonde-like ones.

Proposition 9 *Given* x, y, G, H *and* $\mathsf{w}_1, \ldots, \mathsf{w}_t$ *as above, and assuming that* $\max\{s, t\} \le \alpha$, *one can compute all the products* $\mathsf{A}_{j,k}\,\mathsf{w}_k$ *using* $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$ *operations in* $\mathbb{K}$.

PROOF. Let g_i and h_i be the columns of G and H. We adapt the partition of x to g_i and the one of y to h_i, writing

$$\mathsf{g}_i = \begin{bmatrix} \mathsf{g}_{i,1} \\ \vdots \\ \mathsf{g}_{i,s} \end{bmatrix}, \quad \text{with } \mathsf{g}_{i,j} \text{ in } \mathbb{K}^{\nu_j}, \qquad \mathsf{h}_i = \begin{bmatrix} \mathsf{h}_{i,1} \\ \vdots \\ \mathsf{h}_{i,t} \end{bmatrix}, \quad \text{with } \mathsf{h}_{i,k} \text{ in } \mathbb{K}^{\delta_k}.$$

For $k \leq t$, let $F_k = \prod_{\gamma \in \mathsf{y}_k}(1 - x\gamma)$ and let $\mathsf{v}_k \in \mathbb{K}^{\delta_k}$ be such that $\mathsf{Pol}(\mathsf{v}_k) = F_k \bmod x^{\delta_k}$. Using Equation (21) and Lemma 2, it follows that the submatrices $\mathsf{A}_{j,k}$ of A are given by $\mathsf{A}_{j,k} = \mathbb{D}(F_k(\mathsf{x}_j))^{-1}\,\mathsf{B}_{j,k}$, with

$$\mathsf{B}_{j,k} = \sum_{i=1}^{\alpha} \mathbb{D}(\mathsf{g}_{i,j})\,\mathbb{V}(\mathsf{x}_j, \delta_k)\,\mathbb{L}(\mathsf{v}_k)\,\mathbb{V}(\mathsf{y}_k, \delta_k)^t\,\mathbb{D}(\mathsf{h}_{i,k}). \tag{23}$$

Remark first that the matrices $\mathbb{D}(F_k(\mathsf{x}_j))$ can easily be obtained, using two polynomial operations of Section 2:

- Each polynomial F_k can be constructed from its roots in time $O(\mathsf{M}(\delta_k)\log(\delta_k))$. Since $\sum_{k \leq t} \delta_k = n$, the cost of getting all of them is in $O(\mathsf{M}(n)\log(n))$.
- By fast evaluation, each $F_k(\mathsf{x}) \in \mathbb{K}^n$ can be obtained in $O(\mathsf{M}(n)\log(n))$ operations, and thus all of them in time $O(t\,\mathsf{M}(n)\log(n)) \subset O(\alpha\,\mathsf{M}(n)\log(n))$.

We now turn to the computation of the vectors $\mathsf{B}_{j,k}\,\mathsf{w}_k$. Since x_j and y_k are repetition-free, we let $Q_{i,j} = \mathsf{Interp}(\mathsf{x}_j, \mathsf{g}_{i,j})$ and $S_{i,k} = \mathsf{Interp}(\mathsf{y}_k, \mathsf{h}_{i,k})$. Applying Lemma 1 twice to the right hand side of Equation (23) then gives

$$\mathsf{B}_{j,k} = \sum_{i=1}^{\alpha} \mathbb{V}(\mathsf{x}_j, \nu_j + \delta_k - 1)\,\mathbb{M}(Q_{i,j}, \delta_k)\,\mathbb{L}(\mathsf{v}_k)\,\mathbb{M}(S_{i,k}, \delta_k)^t\,\mathbb{V}(\mathsf{y}_k, 2\delta_k - 1)^t.$$

We can now factor out the leftmost and rightmost (transposed) Vandermonde matrices, which do not depend on the summation index i. Defining

$$\mathsf{f}_k = \mathbb{V}(\mathsf{y}_k, 2\delta_k - 1)^t\,\mathsf{w}_k \tag{24}$$

and

$$\mathsf{w}'_{j,k} = \sum_{i=1}^{\alpha} \mathbb{M}(Q_{i,j}, \delta_k)\,\mathbb{L}(\mathsf{v}_k)\,\mathbb{M}(S_{i,k}, \delta_k)^t\,\mathsf{f}_k, \tag{25}$$

we deduce from the previous equation that

$$\mathsf{B}_{j,k}\,\mathsf{w}_k = \mathbb{V}(\mathsf{x}_j, \nu_j + \delta_k - 1)\,\mathsf{w}'_{j,k}. \tag{26}$$

It remains to estimate the cost of computing all vectors $\mathsf{B}_{j,k}\,\mathsf{w}_k$ by means of Equations (24), (25) and (26), and deducing the desired vectors $\mathsf{A}_{j,k}\,\mathsf{w}_k$. The costs of the first and last steps follow directly from the reminders of Section 2:

- By fast transposed evaluation, each vector f_k is obtained in $O(\mathsf{M}(\delta_k)\log(\delta_k))$ operations. Since $\sum_{k\leq t}\delta_k = n$, this gives a total cost of $O(\mathsf{M}(n)\log(n))$.
- Assuming that we know $\mathsf{w}'_{j,k}$, we deduce $\mathsf{B}_{j,k}\,\mathsf{w}_k$ by fast evaluation, in

$$O(\mathsf{M}(\nu_j+\delta_k)\log(\nu_j+\delta_k))$$

  operations. A total cost of $O(\alpha\,\mathsf{M}(n)\log(n))$ then follows from the facts that

$$\mathsf{M}(\nu_j+\delta_k) \leq (\nu_j+\delta_k)\,\mathsf{M}(2n)/2n \quad \text{and} \quad \sum_{j\leq s}\sum_{k\leq t}(\nu_j+\delta_k) = (s+t)\,n.$$

- One recovers the vectors $\mathsf{A}_{j,k}\,\mathsf{w}_k$ with $t\,n \in O(\alpha\,n)$ divisions.

It only remains to bound the cost of deducing the vectors $\mathsf{w}'_{j,k}$ in Equation (25) from the vectors f_k. Recall that $\mathsf{Pol}(\mathsf{v}_k) = F_k \bmod x^{\delta_k}$ and let $S'_{i,k} = \mathsf{Rev}_{\delta_k}(S_{i,k})$ and $F'_k = \mathsf{Pol}(\mathsf{f}_k)$. Then, using Lemma 4 and Lemma 5, we obtain

$$\mathsf{Pol}(\mathsf{w}'_{j,k}) = \textstyle\sum_{i=1}^{\alpha} Q_{i,j}R_{i,k}, \tag{27}$$

where $Q_{i,j} \in \mathbb{K}[x]_{\nu_j}$ is as above and where $R_{i,k} \in \mathbb{K}[x]_{\delta_k}$ is given by

$$R_{i,k} = F_k\,(S'_{i,k}\,F'_k \text{ div } x^{\delta_k-1}) \bmod x^{\delta_k}.$$

The cost then follows from Section 2 as well as Proposition 2 in Section 3:

- By fast interpolation, each $Q_{i,j}$ can be computed in $O(\mathsf{M}(\nu_j)\log(\nu_j))$ operations. Since $\sum_{j\leq s}\nu_j = n$, this gives a total of $O(\alpha\,\mathsf{M}(n)\log(n))$.
- By fast interpolation, each $S_{i,k}$ can be computed in $O(\mathsf{M}(\delta_k)\log(\delta_k))$ operations. Since $\sum_{k\leq t}\delta_k = n$, the total cost for all $S_{i,k}$ and $S'_{i,k}$ is $O(\alpha\,\mathsf{M}(n)\log(n))$.
- Since $S'_{i,k} \in \mathbb{K}[x]_{\delta_k}$ and $F'_k \in \mathbb{K}[x]_{2\delta_k-1}$, a full product $S'_{i,k}\,F'_k$ has cost $O(\mathsf{M}(\delta_k))$, and thus all $R_{i,k}$ follow from F_k, $S'_{i,k}$, F'_k in time $O(\alpha\,\mathsf{M}(n))$.
- Applying Proposition 2 to Equation (27) shows that all the polynomials $\mathsf{Pol}(\mathsf{w}'_{j,k})$ can be obtained in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$. □

6.2 The case of low multiplicities

Here we reduce the Cauchy case to the Vandermonde case, assuming that the multiplicities of x and y are bounded by α. As in Subsection 5.2, we adapt the reduction of [24], now allowing for repetitions in both x and y.

Proposition 10 *Let $\mathsf{x},\mathsf{y} \in \mathbb{K}^n$ be as in Equations (1), (4) and (5). If the multiplicity s of x and the multiplicity t of y satisfy $\max\{s,t\} \leq \alpha$, then the problem $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{x}),\mathbb{D}(\mathsf{y}),\alpha)$ can be solved in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$. The algorithm is probabilistic of type $P(3n-2, n^2+n)$.*

PROOF. Given $\mathsf{G}, \mathsf{H} \in \mathbb{K}^{n\times\alpha}$ and $\mathsf{v} \in \mathbb{K}^n$, we are looking for solutions u to the system $\mathsf{Au} = \mathsf{v}$, where A is the $n \times n$ matrix such that $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{D}(\mathsf{y})](\mathsf{A}) = \mathsf{GH}^t$.

For $k \leq t$, let $M_k = \prod_{\gamma\in\mathsf{y}_k}(x - \gamma)$. Since y_j is repetition-free, Lemma 3 gives

$$\mathbb{D}(\mathsf{y}_k) = \mathbb{V}(\mathsf{y}_k, \delta_k)\, \mathbb{X}(M_k)\, \mathbb{V}(\mathsf{y}_k, \delta_k)^{-1}.$$

Hence

$$\mathbb{D}(\mathsf{y}) = \mathbb{V}\,\mathbb{X}\,\mathbb{V}^{-1}, \tag{28}$$

where $\mathbb{V}$ and $\mathbb{X}$ are block-diagonal with respective blocks $\mathbb{V}(\mathsf{y}_k, \delta_k)$ and $\mathbb{X}(M_k)$. For $k \leq t$, let $\mathsf{m}_k \in \mathbb{K}^{\delta_k}$ be the coefficient vector of $-M_k \bmod x^{\delta_k}$. Then $\mathbb{X}(M_k) = \mathbb{Z}_{\delta_k,0} + \mathsf{m}_k\, \mathsf{e}^t_{\delta_k,\delta_k}$ and, recalling that $\delta^*_k = \delta_1 + \cdots + \delta_k$,

$$\mathbb{X} = \mathbb{Z}_{n,0} + \sum_{k=1}^{t} \mathsf{d}_k \mathsf{e}^t_{n,\delta^*_k}, \tag{29}$$

where for $k \leq t$, $\mathsf{d}_k \in \mathbb{K}^n$ is obtained by padding m_k with δ^*_{k-1} zeros on the top and, if $k \neq t$, with -1 followed by $n - \delta^*_k - 1$ zeros on the bottom.

Defining $\mathsf{B} = \mathsf{A}\mathbb{V}^{-t}$, solving $\mathsf{Au} = \mathsf{v}$ amounts to solve $\mathsf{Bu}' = \mathsf{v}$ and then, if a solution exists, to compute $\mathsf{u} = \mathbb{V}^{-t}\mathsf{u}'$. To do so in the claimed complexity, we exhibit the Vandermonde-like structure of B and bound the cost of computing u and a generator for B. By transposing both sides of Equation (28), we obtain

$$\mathsf{B} - \mathbb{D}(\mathsf{x})\,\mathsf{B}\,\mathbb{X}^t = \mathsf{G}\,\mathsf{H}'^t, \quad \text{with } \mathsf{H}' = \mathbb{V}^{-1}\mathsf{H}.$$

Now, for $k \leq t$, define the vectors $\mathsf{c}_k = \mathsf{B}\,\mathsf{e}_{n,\delta^*_k}$ and $\mathsf{c}'_k = \mathbb{D}(\mathsf{x})\,\mathsf{c}_k$. Using Equation (29), the previous identity then becomes

$$\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,0}](\mathsf{B}) = \mathsf{C}'\,\mathsf{D}^t + \mathsf{G}\,\mathsf{H}'^t, \tag{30}$$

where C' has columns c'_k and D has columns d_k. The matrices $[\mathsf{C}'\ \mathsf{G}]$ and $[\mathsf{D}\ \mathsf{H}']$ thus form a $\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,0}$-generator of length $t + \alpha$ for B. Since the operator in Equation (30) is $\Delta[\mathbb{D}(\mathsf{x}), \mathbb{Z}^t_{n,\psi}]$ with $\psi = 0$, the condition in Equation (2) is clearly satisfied. Moreover, the multiplicity s of x is bounded by the length $t+\alpha$ of the above generator. Therefore, once this generator is known, the system $\mathsf{Bu}' = \mathsf{v}$ can be solved by the probabilistic algorithm of type $P(3n-2, n^2+n)$ in Proposition 7; since $t \leq \alpha$, the cost is still in $O(\alpha^{\omega-1}\mathsf{M}(n)\log^2(n))$.

It remains to bound the cost computing u, C', D and H'. To do so, we will use the result in Proposition 9 and the reminders of Section 2 on the complexity of polynomial operations:

- By fast transposed interpolation and using the block structure of the matrix $\mathbb{V}$, we can deduce the vector u from y and u' in time $O(\mathsf{M}(n)\log(n))$.

- By fast interpolation and using the block structure of the matrix $\mathbb{V}$, we can deduce the matrix H' from y and H in time $O(\alpha\,\mathsf{M}(n)\log(n))$.
- All polynomials M_k (and thus all vectors d_k and the matrix D) can be constructed from their roots in time $O(\mathsf{M}(n)\log(n))$.

In order to bound the cost of computing C', note that, due to the block structure of A and $\mathbb{V}^{-t}$, each vector c_k is in fact given by $\mathsf{c}_k = \mathsf{A}_{j,k}\mathsf{w}_k$, where $\mathsf{w}_k \in \mathbb{K}^{\delta_k}$ is the last column of $\mathbb{V}(\mathsf{y}_k, \delta_k)^{-t}$. Thus, we can compute the vectors $\mathsf{w}_1, \ldots, \mathsf{w}_t$ and then solve the multiplication problem of Section 6.1:

- Computing the last column of $\mathbb{V}(\mathsf{y}_k, \delta_k)^{-t}$ takes time $O(\mathsf{M}(\delta_k)\log(\delta_k))$, which induces a cost of $O(\mathsf{M}(n)\log(n))$ for finding all vectors w_k.
- Knowing all w_k, Proposition 9 shows that all vectors c_k can be computed in time $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$. Deducing the vectors c'_k by t multiplications with $\mathbb{D}(\mathsf{x})$ takes time $O(\alpha\,n)$, which concludes the proof. □

6.3 The case of high multiplicities

We eventually consider the case of high multiplicities ($\max\{s,t\} > \alpha$), reducing it to the case of low multiplicities ($\max\{s,t\} \leq \alpha$) seen in Subsection 6.2. Our reduction has cost $O(\alpha^{\omega-1}\,\mathsf{M}(n))$ and thus fits in the requested bound; however, like in the Vandermonde case, randomization is used. Combining the complexity and probability results in the next proposition with those of Proposition 10 concludes the proof of Theorem 3.

Proposition 11 *Let* x *and* y *in* $\mathbb{K}^n$ *be as in Equations* (1), (4), *and* (5). *If* x *or* y *has multiplicity larger than* α, *one can reduce the problem* $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{x}), \mathbb{D}(\mathsf{y}), \alpha)$ *to the problem* $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{b}), \alpha)$, *where* $\mathsf{a}, \mathsf{b} \in \mathbb{K}^n$ *both have multiplicity at most* α *and satisfy the constraints of Equations* (1), (4) *and* (5). *The reduction can be done in time* $O(\alpha^{\omega-1}n)$ *by a probabilistic algorithm of type* $P(2n, 6n^2)$.

Proof. If x has multiplicity $s > \alpha$, we start by reducing the problem $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{x}), \mathbb{D}(\mathsf{y}), \alpha)$ to the problem $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{y}), \alpha)$, where $\mathsf{a} \in \mathbb{K}^n$ has multiplicity at most α and satisfies the constraints of Equations (1) and (5).

We can do this first reduction in time $O(\alpha^{\omega-1}n)$ by a probabilistic algorithm of type $P(n, 3n^2)$, using the same technique as in the proof of Proposition 8. The only modifications to do there are to replace $\mathbb{Z}^t_{n,\psi}$ by $\mathbb{D}(\mathsf{y})$, and to replace the polynomial δ in Equation (20) by one of the form

$$\prod_{i<j\leq r}(Y_i - Y_j) \;\times\; \prod_{i\leq r,\, j\leq r'}(Y_i - \xi_j) \;\times\; \prod_{i\leq r,\, j\leq n}(Y_i\, y_j - 1), \quad \text{with } r, r' \leq n, \tag{31}$$

which still has degree bounded by $3n^2$. We do not give more details on this part of the algorithm.

If y has multiplicity $t > \alpha$, it remains to reduce $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{y}), \alpha)$ to $\mathsf{LinearSystem}(\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{b}), \alpha)$, where b has multiplicity at most α and satisfies the constraints of Equations (4) and (5).

To do so, let G and H in $\mathbb{K}^{n\times\alpha}$ and v in $\mathbb{K}^n$ be given, and recall that we are looking for solutions to the system $\mathsf{Au} = \mathsf{v}$, where A is such that $\Delta[\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{y})](\mathsf{A}) = \mathsf{GH}^t$. We assume that G and H have full rank, for otherwise one can replace (G, H) by a minimal-length generator, whose matrices then have full rank.

We start by reordering the entries of y to obtain a repetition-free decomposition as in Equation (6) of Section 2:

$$\mathsf{y}' = \begin{bmatrix} y_{\sigma(1)} \\ \vdots \\ y_{\sigma(n)} \end{bmatrix} = \begin{bmatrix} \mathsf{y}'_1 \\ \vdots \\ \mathsf{y}'_{t'} \end{bmatrix},$$

where y'_i is a vector consisting of μ_i repetitions of the same element ξ_i, so that $n = \mu_1 + \cdots + \mu_{t'}$, with $\xi_i \neq \xi_j$ for $i \neq j$ and $\mu_1 \geq \cdots \geq \mu_{t'} > 0$.

Let A' and H' be obtained by applying the same reordering to, respectively, the columns of A and the rows of H; hence, $(\mathsf{G}, \mathsf{H}')$ is a $\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{y}')$-generator for A'. Writing Π for the permutation matrix such that $\mathsf{A}' = \mathsf{A}\,\Pi$, we see that every solution to the system $\mathsf{A}\,\mathsf{u} = \mathsf{v}$ corresponds to a solution $\Pi^t\,\mathsf{u}$ to the system $\mathsf{A}'\mathsf{u}' = \mathsf{v}$, and conversely. Therefore we focus on solving $\mathsf{A}'\mathsf{u}' = \mathsf{v}$.

By construction, the matrices A' and H' admit the following decompositions:

$$\mathsf{A}' = \begin{bmatrix} \mathsf{A}'_1 & \cdots & \mathsf{A}'_{t'} \end{bmatrix} \quad \text{and} \quad \mathsf{H}' = \begin{bmatrix} \mathsf{H}'_1 \\ \vdots \\ \mathsf{H}'_{t'} \end{bmatrix},$$

with A'_i in $\mathbb{K}^{n\times\mu_i}$ and H'_i in $\mathbb{K}^{\mu_i\times\alpha}$. Remark then that $\mathsf{A}'_i - \mathbb{D}(\mathsf{a})\,\mathsf{A}'_i\,\mathbb{D}(\mathsf{y}'_i) = \mathsf{G}\,\mathsf{H}'^t_i$. Now, $\mathbb{D}(\mathsf{y}'_i) = \xi_i\mathbb{I}_{\mu_i}$ and, since $\xi_i a_j \neq 1$ for all $i \leq t'$ and all $j \leq n$, all matrices $\mathbb{I}_n - \xi_i\mathbb{D}(\mathsf{a})$ are invertible. We thus obtain the equalities

$$\mathsf{A}'_i = (\mathbb{I}_n - \xi_i\mathbb{D}(\mathsf{a}))^{-1}\,\mathsf{G}\,\mathsf{H}'^t_i \quad \text{for } 1 \leq i \leq t'. \tag{32}$$

We will use dense matrix methods to reduce the number of non-zero entries in A' and H', while maintaining a Cauchy-like structure. Let thus τ be such that $\mu_\tau > \alpha \geq \mu_{\tau+1}$.

- For $1 \leq i \leq \tau$, we have $\mu_i > \alpha$. We let $J_i \subset \{1, \ldots, \mu_i\}$ and $\mathsf{E}_i \in \mathbb{K}^{\mu_i\times\mu_i}$ be

the index set and the matrix obtained by applying Lemma 9 to H'_i. Let also $r_i = |J_i| = \mathrm{rank}(\mathsf{H}'_i)$, so that $r_i \leq \alpha$. Equation (32) then gives

$$\mathsf{E}_i\,\mathsf{H}'_i = \begin{bmatrix} \mathsf{H}''_i \\ 0 \end{bmatrix} \quad \text{and} \quad \mathsf{A}'_i\,\mathsf{E}_i^t = \begin{bmatrix} \mathsf{A}''_i & 0 \end{bmatrix},$$

where H''_i and A''_i have respective sizes $(r_i \times \alpha)$ and $(n \times r_i)$, and H''_i consists of the rows of H'_i indexed by J_i.

- For $\tau < i \leq t'$, we have $\mu_i \leq \alpha$. We let $\mathsf{E}_i = \mathbb{I}_{\mu_i}$, $\mathsf{H}''_i = \mathsf{H}'_i$ and $\mathsf{A}''_i = \mathsf{A}'_i$.

Let E be the block-diagonal matrix having $\mathsf{E}_1, \ldots, \mathsf{E}_{t'}$ on the diagonal. Hence, $\mathsf{H}'' = \mathsf{E}\,\mathsf{H}'$ consists of the matrices H''_i, interleaved by blocks of zeros when $i \leq \tau$; the same holds for $\mathsf{A}'' = \mathsf{A}'\,\mathsf{E}^t$, considering columns instead of rows. Besides, since each $\mathbb{D}(\mathsf{y}'_i)$ is a homothety matrix, it commutes with E_i^t; we deduce that $(\mathsf{G}, \mathsf{H}'')$ is a $\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{y}')$-generator for A''. Finally, since E is invertible, every solution to the system $\mathsf{A}'\,\mathsf{u}' = \mathsf{v}$ corresponds to a solution $\mathsf{E}^{-t}\,\mathsf{u}'$ to the system $\mathsf{A}''\mathsf{u}'' = \mathsf{v}$, and conversely. Therefore, solving the problem $\mathsf{A}''\mathsf{u}'' = \mathsf{v}$ is enough.

We do so by exhibiting the Cauchy-like structure of A'' for a modified displacement operator whose second diagonal component has lower multiplicity. Define $\mathsf{y}'' \in \mathbb{K}^n$ by replacing, for $i \leq \tau$, the last $\mu_i - r_i$ entries of y'_i by new values taken from $\mathbb{K}$. Due to the presence of corresponding blocks of zeros in H'' and A'', the matrices G and H'' now form a $\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{y}'')$-generator for A''.

With $r = \sum_{i \leq \tau}(\mu_i - r_i)$, suppose that the new values $z_1, \ldots, z_r$ inserted in y'' are pairwise distinct and that none of them belongs to y' or satisfies $a_i z_j = 1$. Then, y'' satisfies the constraint of Equation (5) and has multiplicity at most α (since all r_i are at most α).

It remains to reorder the entries of y'' to obtain a vector b that also satisfies Equation (1), and let Σ be the permutation matrix such that $\mathsf{b} = \Sigma\,\mathsf{y}''$. Defining $\mathsf{F} = \Sigma\,\mathsf{H}''$, it follows that (G, F) is a $\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{b})$-generator for the matrix $\mathsf{B} = \mathsf{A}''\,\Sigma^t$. To conclude, every solution to $\mathsf{B}\mathsf{w} = \mathsf{v}$ corresponds to a solution $\mathsf{u}'' = \Sigma^t\mathsf{w}$ to $\mathsf{A}''\mathsf{u}'' = \mathsf{v}$, and conversely.

We have thus reduced solving the system $\mathsf{A}\mathsf{u} = \mathsf{v}$ to solving $\mathsf{B}\mathsf{w} = \mathsf{v}$ (and then, if a solution exists, to computing $\mathsf{u} = \Pi\,\mathsf{E}^t\,\Sigma^t\,\mathsf{w}$), while providing a generator for B with respect to the Cauchy-like structure $\mathbb{D}(\mathsf{a}), \mathbb{D}(\mathsf{b})$.

The complexity analysis of this second reduction is the same as in the proof of Proposition 8 and yields a cost in $O(\alpha^{\omega-1}n)$ as required.

For the probability analysis, remark that the values $z_1, \ldots, z_r$ satisfy our requirements if they do not cancel the polynomial

$$\gamma = \prod_{i<j\leq r} (Z_i - Z_j) \times \prod_{i\leq r,\, j\leq t'} (Z_i - \xi_j) \times \prod_{i\leq n,\, j\leq r} (a_i Z_j - 1),$$

which has degree at most $3n^2$, since r and t' are bounded by n. Hence, the algorithm for the second reduction has type $P(n, 3n^2)$. Recalling that the first reduction was also by an algorithm of type $P(n, 3n^2)$ concludes the proof. □

References

[1] B. Beckermann. A reliable method for computing M-Padé approximants on arbitrary staircases. *J. Comput. Appl. Math.*, 40(1):19–42, 1992.

[2] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM J. Matrix Anal. Appl.*, 15(3):804–823, 1994.

[3] B. Beckermann and G. Labahn. Fraction-free computation of matrix rational interpolants and matrix GCDs. *SIAM J. Matrix Anal. Appl.*, 22(1):114–144, 2000.

[4] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *20th Annual ACM Symp. Theory Comp.*, pages 301–309d. ACM Press, 1988.

[5] D. J. Bernstein. Factoring into coprimes in essentially linear time. *J. Algorithms* 54(1):1–30, 2005.

[6] D. Bini and V. Y. Pan. *Polynomial and Matrix Computations, volume 1: Fundamental Algorithms*. Birkhäuser, 1994.

[7] R. R. Bitmead and B. D. O. Anderson. Asymptotically fast solution of Toeplitz and related systems of linear equations. *Linear Algebra Appl.*, 34:103–116, 1980.

[8] A. Bostan, C.-P. Jeannerod and É. Schost. Solving Toeplitz- and Vandermonde-like linear systems with large displacement rank. In *ISSAC'07*, pages 33–40. ACM Press, 2007.

[9] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC'03*, pages 37–44. ACM Press, 2003.

[10] J. Canny, E. Kaltofen, and Y. Lakshman. Solving systems of non-linear polynomial equations faster. In *ISSAC'89*, pages 121–128. ACM Press, 1989.

[11] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.

[12] Z. Chen and V. Y. Pan. An efficient solution for Cauchy-like systems of linear equations. *Computers and Mathematics with Applications*, 48:529-537, 2004.

[13] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[14] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[15] J.-G. Dumas, T. Gautier, and C. Pernet. Finite field linear algebra subroutines. In *ISSAC'02*, pages 63–74. ACM Press, 2002.

[16] W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann, and G. Villard. Solving sparse rational linear systems. In *ISSAC'06*, pages 63–70. ACM Press, 2006.

[17] W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann, and G. Villard. Faster inversion and other black box matrix computations using efficient block projections. In *ISSAC'07*, pages 143–150. ACM Press, 2007.

[18] T. Finck, G. Heinig, and K. Rost. An inversion formula and fast algorithms for Cauchy-Vandermonde matrices. *Linear Algebra and its Appl.*, 183(1):179–191, 1993.

[19] S. Gao, V. M. Rodrigues, and J. Stroomer. Gröbner basis structure of finite sets of points, preprint, 2003.

[20] M. Gasca and T. Sauer. Polynomial interpolation in several variables. *Adv. Comput. Math.*, 12(4):377–410, 2000.

[21] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition, 2003.

[22] J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Comput. Complexity*, 2(3):187–224, 1992.

[23] P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *ISSAC'03*, pages 135–142. ACM Press, 2003.

[24] I. C. Gohberg and V. Olshevsky. Complexity of multiplication with vectors for structured matrices. *Linear Algebra Appl.*, 202:163–192, 1994.

[25] O. H. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *J. Algorithms*, 3(1):45–56, 1982.

[26] T. Kailath, S. Y. Kung, and M. Morf. Displacement ranks of matrices and linear equations. *J. Math. Anal. Appl.*, 68(2):395–407, 1979.

[27] E. Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *J. ACM*, 35(1):231–264. 1988.

[28] E. Kaltofen. Asymptotically fast solution of Toeplitz-like singular linear systems. In *ISSAC'94*, pages 297–304. ACM Press, 1994.

[29] E. Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Math. Comp.*, 64(210):777–806, 1995.

[30] E. Kaltofen and Y. Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In *ISSAC'88*, number 358 of LNCS, pages 467–474. Springer Verlag, 1988.

[31] E. Kaltofen and D. Saunders. On Wiedemann's method of solving sparse linear systems. In *AAECC-9*, number 539 of LNCS, pages 29–38. Springer Verlag, 1991.

[32] I. Kaporin. The aggregation and cancellation techniques as a practical tool for faster matrix multiplication. *Theor. Comput. Sci.*, 315(2-3):469–510, 2004.

[33] G. Labahn, D. K. Choi, and S. Cabay. The inverses of block Hankel and block Toeplitz matrices. *SIAM J. Comput.*, 19(1):98–123, 1990.

[34] J. Laderman, V. Y. Pan, and X.-H. Sha. On practical algorithms for accelerated matrix multiplication. *Linear Algebra Appl.*, 162-164:557–588, 1992.

[35] D. Lazard. Ideal bases and primary decomposition: the case of two variables. *J. Symb. Comput.*, 1:261–270, 1985.

[36] P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comp.*, 48(177): 243–264, 1987.

[37] M. Morf. *Fast algorithms for multivariable systems*. PhD thesis, Stanford University, 1974.

[38] M. Morf. Doubling algorithms for Toeplitz and related equations. In *IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 954–959, 1980.

[39] T. Mulders. On short multiplications and divisions. *Applic. Alg. Eng. Comm. Comp.*, 11(1):69–88, 2000.

[40] M. Nüsken and M. Ziegler. Fast multipoint evaluation of bivariate polynomials. In *ESA 2004*, number 3222 in LNCS, pages 544–555. Springer, 2004.

[41] V. Y. Pan. On some computations with dense structured matrices. In *ISSAC'89*, pages 34–42. ACM Press, 1989.

[42] V. Y. Pan. On computations with dense structured matrices. *Math. Comp.*, 55(191):179–190, 1990.

[43] V. Y. Pan. Parametrization of Newton's iteration for computations with structured matrices and applications. *Computers Math. Applic.*, 24(3):61–75, 1992.

[44] V. Y. Pan. Nearly optimal computations with structured matrices. In *SODA'00*, pages 953–962. ACM Press, 2000.

[45] V. Y. Pan. *Structured Matrices and Polynomials*. Birkhäuser Boston Inc., 2001.

[46] V. Y. Pan and X. Wang. Inversion of displacement operators. *SIAM J. Matrix Anal. Appl.*, 24(3): 660–677, 2003.

[47] V. Y. Pan and A. Zheng. Superfast algorithms for Cauchy-like matrix computations and extensions. *Linear Algebra Appl.*, 310:83–108, 2000.

[48] V. Y. Pan, A. Zheng, M. Abu Tabanjeh, Z. Chen and S. Providence. Superfast computations with singular structured matrices over abstract fields. In *Computer algebra in scientific computing: CASC'99*, pages 323–338. Springer, 1999.

[49] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

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

[51] A. Storjohann. *Algorithms for matrix canonical forms*. PhD thesis, ETH, Zürich, 2000.

[52] A. Storjohann. Notes on computing minimal approximant bases. Technical report, Symbolic Computation Group, University of Waterloo, 2006.

[53] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[54] M. Van Barel and A. Bultheel. A general module theoretic framework for vector M-Padé and matrix rational interpolation. *Numer. Algorithms*, 3:451–461, 1992.

[55] R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM' 79*, number 72 of *LNCS*. Springer Verlag, 1979.

[56] R. Zippel. Interpolating polynomials from their values. *J. Symb. Comp.*, 9(3):375–403, 1990.

Appendix

All along this paper, for matrices of Vandermonde or Cauchy type, we assumed that the diagonal matrices used in the operators are as in Equation (1). One can always reduce to this situation by permuting the entries of these diagonal matrices. We discuss here the cost of this operation.

Given $\mathsf{a} = [a_1, \ldots, a_n]^t$ in $\mathbb{K}^n$, we actually study the cost of finding a repetition-free decomposition of a, that is, a permutation that puts a into the form of Equation (6) of Section 2. As seen in Lemma 11, the further cost of converting to the representation of Equation (1) is then a mere $O(n)$.

If the base field $\mathbb{K}$ is endowed with an order $<$, a sorting algorithm solves the problem in $O(n \log(n))$ comparisons. However, if we do not allow ordering the elements of $\mathbb{K}$, it is far less clear how to achieve a similar complexity. Our solution involves polynomial arithmetic, and bears similarities with techniques used in both elliptic curve factorization algorithms [36] or factorization into coprimes [5].

Proposition 12 *Let* $\mathsf{a} = [a_1, \ldots, a_n]^t$ *be in* $\mathbb{K}^n$*. One can compute a repetition-free decomposition of* a *in* $O(\mathsf{M}(n)\log^3(n))$ *operations.*

Before giving the proof of this proposition, we start by two lemmas on sequence manipulations. As it turns out, one of the main difficulties is the following (see Lemma 19): given two repetition-free sequences $[c_1, \ldots, c_r]$ and $[d_1, \ldots, d_r]$, such that $d_i = c_{\sigma(i)}$ for some permutation σ of $\{1, \ldots, r\}$, how to recover σ, if ordering elements is not allowed?

Lemma 18 *Let* $\mathsf{c} = [c_1, \ldots, c_s]$ *and* $\mathsf{d} = [d_1, \ldots, d_t]$ *be repetition-free vectors with entries in* $\mathbb{K}$*. One can compute sequences of integers* $\ell_1 < \cdots < \ell_u$ *and* $m_1 < \cdots < m_u$ *such that*

$$\{c_1, \ldots, c_s\} \cap \{d_1, \ldots, d_t\} = \{c_{\ell_1}, \ldots, c_{\ell_u}\} = \{d_{m_1}, \ldots, d_{m_u}\}$$

in $O(\mathsf{M}(q)\log(q))$ *operations, with* $q = \max\{s, t\}$*.*

Proof. Let $P = \prod_{i \le s}(x - c_i)$. Remark that $\alpha \in \mathbb{K}$ belongs to $\{c_1, \ldots, c_s\}$ if and only if $P(\alpha) = 0$; hence, evaluating P at d, we obtain the indices m_i as those for which $P(d_{m_i}) = 0$. Similarly, one obtains the indices ℓ_i by evaluating $Q = \prod_{i \le t}(x - d_i)$ at c. All operations fit into the $O(\mathsf{M}(q)\log(q))$ bound. □

Lemma 19 *Let* $\mathsf{c} = [c_1, \ldots, c_r]$ *and* $\mathsf{d} = [d_1, \ldots, d_r]$ *be repetition-free vectors with entries in* $\mathbb{K}$*, such that* $\{c_1, \ldots, c_r\} = \{d_1, \ldots, d_r\}$ *(as sets). One can compute the unique permutation* σ *of* $\{1, \ldots, r\}$ *such that* $d_i = c_{\sigma(i)}$ *in* $O(\mathsf{M}(n)\log^2(n))$ *operations.*

Proof. Defining $s = \lceil r/2 \rceil$ and $t = r - s$, we split d into the subsequences $\mathsf{d}' = [d_1, \ldots, d_s]$ and $\mathsf{d}'' = [d_{s+1}, \ldots, d_r]$ of respective lengths s and t. We then compute the polynomial $P = \prod_{i \le s}(x - d_i)$ and evaluate it at c. Let $\ell_1 < \cdots < \ell_s$ be the indices of the entries of c where P vanishes and $m_1 < \cdots < m_t$ be those where P is non-zero. We deduce that

$$\{c_{\ell_1}, \ldots, c_{\ell_s}\} = \{d_1, \ldots, d_s\} \quad \text{and} \quad \{c_{m_1}, \ldots, c_{m_t}\} = \{d_{s+1}, \ldots, d_r\}.$$

We can then proceed recursively, on $[c_{\ell_1}, \ldots, c_{\ell_s}]$ and d' on one hand, and $[c_{m_1}, \ldots, c_{m_t}]$ and d'' on the other hand. This gives us permutations ρ of $\{1, \ldots, s\}$ and τ of $\{1, \ldots, t\}$, such that $d_i = c_{\ell_{\rho(i)}}$ and $d_{j+s} = c_{m_{\tau(j)}}$ hold for $i \le s$ and $j \le t$. We deduce σ, as $\sigma(i) = \ell_{\rho(i)}$ for $i \le s$ and $\sigma(i) = m_{\tau(i-s)}$ for $i > s$.

The cost of computing P and re-evaluating it is in $O(\mathsf{M}(r)\log(r))$ operations in $\mathbb{K}$; the extra cost is $O(r)$ bookkeeping operations. Our claim follows from the super-additivity of the function M. □

We can now prove our claim on the cost of finding a repetition-free decomposition. We give a recursive algorithm that, given the vector $\mathsf{a} = [a_1, \ldots, a_n]$,

computes a permutation σ of $\{1, \ldots, n\}$ such that

$$\begin{bmatrix} a_{\sigma(1)} \\ \vdots \\ a_{\sigma(n)} \end{bmatrix} = \begin{bmatrix} \mathsf{a}_1 \\ \vdots \\ \mathsf{a}_u \end{bmatrix},$$

where the vector a_i consists of λ_i repetitions of an element α_i, the α_i being pairwise distinct. Remark that this specification is a slight relaxation of the definition given in Section 2, as we do not require that the multiplicities be sorted. The extra cost for sorting them is $O(n)$ integer operations, using bucket sorting.

Given $\mathsf{a} = [a_1, \ldots, a_n]$, we define $\ell = \lceil n/2 \rceil$ and $m = n - \ell$ and recursively call the algorithm on $[a_1, \ldots, a_\ell]$ and $[a_{\ell+1}, \ldots, a_n]$: we obtain permutations ρ' of $\{1, \ldots, \ell\}$ and ρ'' of $\{1, \ldots, m\}$ such that

$$\begin{bmatrix} a_{\rho'(1)} \\ \vdots \\ a_{\rho'(\ell)} \end{bmatrix} = \begin{bmatrix} \mathsf{a}'_1 \\ \vdots \\ \mathsf{a}'_s \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} a_{\ell+\rho''(1)} \\ \vdots \\ a_{\ell+\rho''(m)} \end{bmatrix} = \begin{bmatrix} \mathsf{a}''_1 \\ \vdots \\ \mathsf{a}''_t \end{bmatrix}, \tag{33}$$

where the vector a'_i (resp. a''_j) consists of ν_i (resp. μ_j) repetitions of an element α'_i (resp. α''_j), the α'_i and α''_j being pairwise distinct.

We now have to determine the common elements between the lists $[\alpha'_1, \ldots, \alpha'_s]$ and $[\alpha''_1, \ldots, \alpha''_t]$. Using Lemma 18, we compute in $O(\mathsf{M}(n)\log(n))$ operations the sequences of indices $\ell_1 < \cdots < \ell_r$ and $m_1 < \cdots < m_r$ such that

$$\{\alpha'_1, \ldots, \alpha'_s\} \cap \{\alpha''_1, \ldots, \alpha''_t\} = \{\alpha'_{\ell_1}, \ldots, \alpha'_{\ell_r}\} = \{\alpha''_{m_1}, \ldots, \alpha''_{m_r}\}.$$

Using $O(n)$ extra operations, we then obtain the complementary sequences $\ell'_1 < \cdots < \ell'_{s-r}$ and $m'_1 < \cdots < m'_{t-r}$, such that

$$\{\ell'_1, \ldots, \ell'_{s-r}\} = \{1, \ldots, s\} - \{\ell_1, \ldots, \ell_r\}$$
$$\text{and} \quad \{m'_1, \ldots, m'_{t-r}\} = \{1, \ldots, t\} - \{m_1, \ldots, m_r\}.$$

Since the sequences $[\alpha'_{\ell_1}, \ldots, \alpha'_{\ell_r}]$ and $[\alpha''_{m_1}, \ldots, \alpha''_{m_r}]$ are repetition-free, there exists a permutation σ of $\{1, \ldots, r\}$ such that $\alpha'_{\ell_i} = \alpha''_{m_{\sigma(i)}}$ for $i \leq r$; by Lemma 19, we can compute σ in $O(\mathsf{M}(n)\log^2(n))$ operations.

Knowing these subsequences, it remains to reorder the vectors in Equation (33). We first interleave the entries appearing in both vectors

$$\alpha'_{\ell_1} \ (\nu_{\ell_1} \text{ times}), \alpha''_{m_{\sigma(1)}} \ (\mu_{m_{\sigma(1)}} \text{ times}), \ \ldots, \ \alpha'_{\ell_r} \ (\nu_{\ell_r} \text{ times}), \alpha''_{m_{\sigma(r)}} \ (\mu_{m_{\sigma(r)}} \text{ times}),$$

followed by the entries appearing only in the first vector,

$$\alpha'_{\ell'_1}\ (\nu_{\ell'_1}\text{ times}),\ \ldots,\ \alpha'_{\ell'_{s-r}}\ (\nu_{\ell'_{s-r}}\text{ times}),$$

and by those appearing only in the first vector,

$$\alpha''_{m'_1}\ (\mu_{m'_1}\text{ times})\ \ldots,\ \alpha''_{m_{t-r}}\ (\mu_{m'_{t-r}}\text{ times}).$$

The permutation σ that actually puts a into the above order is readily deduced for $O(n)$ operations. The overall cost is thus $O(\mathsf{M}(n)\log^2(n))$, plus that of the two recursive calls; the estimate given in Proposition 12 follows.