Using structured linear algebra to compute Hermite-Padé approximants

bla blu
bli
blo@ble

ABSTRACT

bla bla

1. INTRODUCTION

In this paper, we discuss linear algebra algorithms inspired by the following kind of question: given polynomials such as

$$t_0 = 8x^4 - 8x^2 + 1$$
$$t_1 = 16x^5 - 20x^3 + 5x$$
$$t_2 = 32x^6 - 48x^4 + 18x^2 - 1,$$

find that the relation $t_0 - 2xt_1 + t_2 = 0$ holds (this is the recurrence between Chebyshev polynomials of the first kind).

More precisely, given input polynomials $(t_0, \ldots, t_{s-1})$ over a field $\mathbb{K}$, together with integers $(n_0, \ldots, n_{s-1})$ and σ, the *Hermite-Padé* approximation problem asks to compute polynomials $(p_0, \ldots, p_{s-1})$, not all zero, such that $\deg(p_i) < n_i$ holds for all i, and such that we have $p_0t_0 + \cdots + p_{s-1}t_{s-1} = O(x^\sigma)$. There exist numerous applications to this type of question, very important particular cases being *algebraic approximants* (with $t_i = f^i$, for some given f) or *differential approximants* (with $t_i = d^i f/dx^i$, for some given f); see for instance [2, Chapitre 7].

Expressed in the canonical monomial bases, the matrix of a Hermite-Padé problem has size $\sigma \times (n_0 + \cdots + n_{s-1})$, and consists of s lower triangular Toeplitz blocks. More generally, our goal in this paper is to compute efficiently elements in the kernel of *mosaic Toeplitz* matrices [11], that is, $m \times n$ matrices $\mathsf{A} = (\mathsf{A}_{i,j})_{1\le i\le p, 1\le j\le q}$ with a $p \times q$ block structure, each of the pq blocks $\mathsf{A}_{i,j}$ being Toeplitz.

An $m \times n$ Toeplitz matrix $\mathsf{A} = (a_{i-j})_{1\le i\le m, 1\le j\le n}$ can be succinctly represented by the polynomial $P_\mathsf{A} = a_{-n+1} + a_{-n+2}x + \cdots + a_{m-1}x^{m+n-2}$; multiplication of A by a vector $\mathsf{b} = [b_0 \ \cdots \ b_{n-1}]^t$ amounts to computing $P_\mathsf{A}P_\mathsf{b}$ mod x^{m+n-1} and keeping the coefficients of degrees $n-1, \ldots, m+n-2$, where we write $P_\mathsf{b} = b_0 + \cdots + b_{n-1}x^{n-1}$. More generally, a mosaic Toeplitz $\mathsf{A} = (\mathsf{A}_{i,j})_{1\le i\le p, 1\le j\le q}$ can be described by a sequence of pq such polynomials $\mathscr{P} = (P_{\mathsf{A}_{i,j}})_{1\le i\le p, 1\le j\le q}$, together with the sequences $I = (i_1, \ldots, i_p)$ and $J = (j_1, \ldots, j_q)$ giving the row-sizes, resp. column-sizes of the blocks. Then, our main problem is the following.

PROBLEM A. *Given polynomials $\mathscr{P}$ and integers I, J as above, defining a mosaic Toeplitz matrix A, find a non-zero vector in the kernel of A.*

We consider two situations, first over an arbitrary field $\mathbb{K}$, counting all operations in $\mathbb{K}$ at unit cost (this is a good model for computations over finite fields), then specifically over $\mathbb{Q}$, taking bit-size into account. In all this paper, we closely follow the existing formalism of *structured matrix computations*. In addition to two new algorithms that complement previous work on the subject, our main contribution is a discussion on the design and practical performance of a C++ implementation of some of these structured matrix algorithms. Our goal being to obtain an implementation with the best possible performance, a significant part of the paper is dedicated to e.g. constant time improvements that are crucial in practice.

2. BASICS ON STRUCTURED LINEAR ALGEBRA

This section reviews background material on structured matrices; for a more comprehensive treatment, we refer the reader to [22].

Overview. Initially developed in [14], the *displacement operator* associates to matrix A its displacement $\phi(\mathsf{A})$, that is, the image of A under a *displacement operator* ϕ. Then, we say that A is structured with respect to ϕ if $\phi(\mathsf{A})$ has a small rank compared to its size; the rank of $\phi(\mathsf{A})$ is called the *displacement rank* of A with respect to ϕ. A prominent example is the family of so-called *Toeplitz-like* matrices, which are structured for the Toeplitz displacement operator

$$\phi^+ : \mathsf{A} \mapsto \mathsf{A} - (\mathsf{A} \text{ shifted down and right by one place}).$$

The displacement rank of a Toeplitz matrix for this operator is at most two; if A is a mosaic Toeplitz with a $p \times q$ block structure, its Toeplitz displacement rank is at most $p + q$.

The key idea of most algorithms for structured matrices is summarized by Pan's motto [22]: compress, operate, decompress. Indeed, for A of size $m \times n$ over a field $\mathbb{K}$, if $\phi(\mathsf{A})$ has rank α, it can be represented using few elements through *ϕ-generators*, that is, two matrices (G, H) in $\mathbb{K}^{m\times\alpha} \times \mathbb{K}^{n\times\alpha}$, with $\phi(\mathsf{A}) = \mathsf{G}\mathsf{H}^t$. The main idea behind algorithms for structured matrices is to use generators as a compact data structure, involving $\alpha(m + n)$ field elements instead of mn.

Cauchy-like matrices. Beyond the Toeplitz structure (and the directly related Hankel one), two other important cases are so-called Vandermonde and Cauchy structures. While the case of Toeplitz-like matrices was the first one to be studied in detail, we will actually focus on Cauchy-like matrices, as we will see that this particular structure is quite convenient to work with.

For a sequence $\mathsf{u} = (u_1, \ldots, u_m)$ in $\mathbb{K}^m$, let $\mathsf{D}_\mathsf{u} \in \mathbb{K}^{m\times m}$ be the diagonal matrix with diagonal entries $u_1, \ldots, u_m$. Then, given u as above and v in $\mathbb{K}^n$, we will consider the operator $\nabla_{\mathsf{u},\mathsf{v}} : \mathsf{A} \in \mathbb{K}^{m\times n} \mapsto \mathsf{D}_\mathsf{u}\mathsf{A} - \mathsf{A}\mathsf{D}_\mathsf{v}$; *Cauchy-like* matrices (with respect to the choice of u and v) are those matrices A for which $\nabla_{\mathsf{u},\mathsf{v}}(\mathsf{A})$ has small rank.

Let u, v be given and suppose that $u_i \neq v_j$ holds for all i, j. Then, the operator $\nabla_{\mathsf{u},\mathsf{v}}$ is invertible: given $\nabla_{\mathsf{u},\mathsf{v}}$-generators (G, H) of length α for A, we can reconstruct A as

$$\mathsf{A} = \sum_{i=1}^{\alpha} \mathsf{D}_{\mathsf{g}_i} \mathsf{C}_{\mathsf{u},\mathsf{v}} \mathsf{D}_{\mathsf{h}_i}, \quad \mathsf{C}_{\mathsf{u},\mathsf{v}} = \begin{bmatrix} \frac{1}{u_1-v_1} & \cdots & \frac{1}{u_1-v_n} \\ \vdots & & \vdots \\ \frac{1}{u_m-v_1} & \cdots & \frac{1}{u_m-v_n} \end{bmatrix}, \quad (1)$$

where g_i and h_i are the ith columns of respectively G and H, and matrix $\mathsf{C}_{\mathsf{u},\mathsf{v}}$ is known as a *Cauchy matrix*. Remark that we can equivalently rewrite A as

$$\mathsf{A} = (\mathsf{G}\mathsf{H}^t) \odot \mathsf{C}_{\mathsf{u},\mathsf{v}}, \quad (2)$$

where $\odot$ denotes the entrywise product.

We will have to handle submatrices of A through their generators. The fact that D_u and D_v are diagonal matrices makes this easy (this is one of the aspects in which the Cauchy structure behaves more simply than the Toeplitz one). Suppose that (G, H) are generators for A, with respect to the operator $\nabla_{\mathsf{u},\mathsf{v}}$, and let $\mathsf{u}_I = (u_i)_{i\in I}$ and $\mathsf{v}_J = (v_j)_{j\in J}$ be subsequences of respectively u and v, corresponding to entries of indices I and J. Let $\mathsf{A}_{I,J}$ be the submatrix of A obtained by keeping rows and columns of indices respectively in I and J, and let $(\mathsf{G}_I, \mathsf{H}_J)$ be the matrices obtained from (G, H) by respectively keeping rows of G of indices in I, and rows of H of indices in J. Then, $(\mathsf{G}_I, \mathsf{H}_J)$ is a $\nabla_{\mathsf{u}_I,\mathsf{v}_J}$-generator for $\mathsf{A}_{I,J}$.

Another useful property relates to inverses of Cauchy-like matrices. If a matrix $\mathsf{A} \in \mathbb{K}^{n\times n}$ is invertible, and is structured with respect to an operator $\nabla_{\mathsf{u},\mathsf{v}}$, its inverse is structured with respect to $\nabla_{\mathsf{v},\mathsf{u}}$: if $\mathsf{D}_\mathsf{u}\mathsf{A} - \mathsf{A}\mathsf{D}_\mathsf{v} = \mathsf{G}\mathsf{H}^t$, one easily deduces that $\mathsf{D}_\mathsf{v}\mathsf{A}^{-1} - \mathsf{A}^{-1}\mathsf{D}_\mathsf{u} = -(\mathsf{A}^{-1}\mathsf{G})(\mathsf{A}^{-t}\mathsf{H})^t$.

Algorithms. In most the paper, we give costs in an algebraic model, counting base field operations at unit cost (in Section 4, we work over $\mathbb{Q}$ and use a boolean model).

We let $\mathscr{M}$ be such that over any ring, polynomials of degree at most d can be multiplied in $\mathscr{M}(d)$ base ring operations; we also assume that the super-linearity assumptions of [9, Chapter 8] hold. Using the Cantor-Kaltofen algorithm [5], we can take $\mathscr{M}(d) \in O(d\log(d)\log\log(d))$. We let ω be a feasible exponent for linear algebra, in the sense that matrices of size n can be multiplied in $O(n^\omega)$ base ring operations over any ring; the best bound to date is $\omega < 2.38$ [7, 17]. The notation $O^\sim(\,)$ indicates that we omit polylogarithmic terms.

Matrix-vector multiplication with $\mathsf{C}_{\mathsf{u},\mathsf{v}}$ reduces to degree n polynomial interpolation at the points v and evaluation at the points u. Using fast polynomial evaluation and interpolation, this can be done in time $O(\mathscr{M}(p')\log(p'))$, with $p' = \max(m, n)$; thus, we can multiply matrix A of (1) by a vector in time $O(\alpha\mathscr{M}(p')\log(p')) \subset O^\sim(\alpha p')$.

In [22, Theorem 4.7.3], Pan shows that if the entries of both u and v are in *geometric progression*, one can reduce the cost of the matrix-vector multiplication by $\mathsf{C}_{\mathsf{u},\mathsf{v}}$ to $O(\mathscr{M}(p'))$, since polynomial evaluation or interpolation at n points in geometric progression can be done in time $O(\mathscr{M}(n))$; similarly, multiplication by A as above takes time $O(\alpha\mathscr{M}(p'))$.

We propose here a refinement of this idea, that allows us to save a constant factor in runtime: we require that u and v be geometric progressions with *the same ratio* τ. Then, the cauchy matrix $\mathsf{C}_{\mathsf{u},\mathsf{v}}$ has entries $1/(u_i - v_j) = 1/(u_1\tau^{i-1} - v_1\tau^{j-1})$, so it can be factored as

$$\mathsf{C}_{\mathsf{u},\mathsf{v}} = \mathsf{D}_{\tau^{-1}} \begin{bmatrix} \frac{1}{u_1-v_1} & \frac{1}{u_1-v_1\tau} & \cdots & \frac{1}{u_1-v_1\tau^{n-1}} \\ \frac{1}{u_1-v_1\tau^{-1}} & \frac{1}{u_1-v_1} & \cdots & \frac{1}{u_1-v_1\tau^{n-2}} \\ \vdots & & \vdots & \\ \frac{1}{u_1-v_1\tau^{1-m}} & \frac{1}{u_1-v_1\tau^{2-m}} & \cdots & \frac{1}{u_1-v_1\tau^{n-m}} \end{bmatrix},$$

where $\mathsf{D}_{\tau^{-1}}$ is diagonal with entries $(1, \tau^{-1}, \tau^{-2}, \ldots, \tau^{1-m})$, and where the left-hand matrix is Toeplitz. In the reconstruction formula (1), the diagonal matrix D_τ commutes with all matrices D_{g_i}, so we can take it out of the sum. Hence, we replaced α evaluations / interpolations at geometric progressions by α product by Toeplitz matrices, each of which can be done in a single polynomial multiplication. The cost for a matrix-vector product by A remains $O(\alpha\mathscr{M}(p'))$, but the constant in the big-O is lower: for $m = n$, using middle product techniques [10, 3, 4], the cost goes down from $3\alpha\mathscr{M}(p') + O(\alpha p')$ to $\alpha\mathscr{M}(p') + O(\alpha p')$.

If one needs to multiply A as above by *several* vectors, further improvements are possible: we mention (without giving details) an algorithm from [**?**], which makes it possible to multiply A by α vectors in time $O(\alpha^{\omega-1}\mathscr{M}(p'))$ instead of $O(\alpha^2\mathscr{M}(p'))$, by reduction to a sequence of polynomial matrix multiplications.

Reduction from mosaic Toeplitz to Cauchy structure. We said that our primary interest lies in mosaic Toeplitz matrices. An important insight of Pan [21] shows that one can reduce questions about Toeplitz-like matrices to ones about Cauchy-like matrices (and conversely, if one wishes to), for a moderate cost overhead.

To a vector u in $\mathbb{K}^m$, let us now associate the corresponding Vandermonde matrix $\mathsf{V}_\mathsf{u} \in \mathbb{K}^{m\times m}$, together with W_u, obtained by reversing the order of the rows in V_u^t. Given u as above, v in $\mathbb{K}^n$, and a mosaic Toeplitz $\mathsf{A} = (\mathsf{A}_{i,j})_{1\leq i\leq p, 1\leq j\leq q}$ of size $m \times n$, with $\mathsf{A}_{i,j}$ of size $m_i \times n_j$, we let $\mathsf{A}' = \mathsf{V}_\mathsf{u}\,\mathsf{A}\,\mathsf{W}_\mathsf{v}$, and we define the following.

- For $1 \leq i \leq p$, let $\mathsf{A}_{i,*}$ be the block matrix $[\mathsf{A}_{i,1} \cdots \mathsf{A}_{i,q}]$. Let η_i be the first row of $\mathsf{A}_{i,*}$, shifted left once, let η'_i be the last row of $\mathsf{A}_{i,*}$, and finally let $\mathsf{h}_i = (\eta_i - \eta'_{i-1})\mathsf{W}_\mathsf{v}$ (for $i = 1$, η'_0 is the zero vector); this is a row-vector of size n. We also let g_i be the column of V_u of index $m_1 + \cdots + m_{i-1} + 1$.
- For $1 \leq j \leq q$, define $\mathsf{A}_{*,j}$ similarly to $\mathsf{A}_{i,*}$; γ_j and γ'_j are its first and last columns, with now a downward shift for γ'_j; in addition, their entry of index $m_1 + \cdots + m_j$ are to zero. Then, we define $\mathsf{g}_{p+j} = \mathsf{V}_\mathsf{u}(\gamma'_j - \gamma_{j+1})$ (as above, undefined entries are set to zero). We also let h_{p+j} be the row of index $n_1 + \cdots + n_j$ in W_v.
- We define g_{p+q+1} as the column vector with entries u_i^m,

$1 \le i \le m$, and h_{p+q+1} as the last row of $\mathsf{AW_v}$; we also define g_{p+q+2} as the first column of $-\mathsf{V_uA}$, and h_{p+q+2} as the column vector with entries v_j^n, $1 \le j \le n$.

Then, a straightforward verification shows that the matrix G' with columns $\mathsf{g}_1, \ldots, \mathsf{g}_{p+q+2}$, and the matrix H' with columns $\mathsf{h}_1^t, \ldots, \mathsf{h}_{p+q+2}^t$, are $\nabla_{\mathsf{u},\mathsf{v}}$-generators of A'.

The bottleneck in the computation of G' and H' are p left-products by $\mathsf{W_v}$ and p products by $\mathsf{V_u}$. If all entries of u and v are in geometric progression this takes time $O(p\mathscr{M}(n) + q\mathscr{M}(m))$; whereas A has ϕ^+-generators of length $p+q$, the generators $(\mathsf{G}', \mathsf{H}')$ have length $p+q+2$.

Regularization. In our algorithms for Cauchy-like matrices, we will assume that input matrix has *generic rank profile*, that is, that its leading principal minors of size up to its rank are invertible. Regularization for structured matrices was introduced for this purpose by Kaltofen [16], for the Toeplitz structure. In our Cauchy context, one could apply to A' as defined above the regularization procedure from [22, Section 5.6], which consists in replacing A' by $\mathsf{A}'' = \mathsf{D_x C_{a,u} A' C_{v,b} D_y}$, for some new vectors $\mathsf{x}, \mathsf{a} \in \mathbb{K}^m$ and $\mathsf{y}, \mathsf{b} \in \mathbb{K}^n$. Theorem 5.6.2 in [22] shows that if a, u consist of $2m$ distinct scalars, and b, v consist of $2n$ distinct scalars, then there exists a non-zero polynomial Δ in the entries of x and y, of degree at most $p = \min(m,n)$ in each block of variables, such that the non-vanishing of Δ implies that A'' has generic rank profile (that theorem is stated for square matrices, but the result holds in the rectangular case as well). The downside of this contruction is that it requires to compute a pair of generators of length $p+q+4$ for A'', involving the multiplication of G' and H' by $\mathsf{C_{a,u}}$ and $\mathsf{C_{v,b}}$.

We now point out a simpler construction. Instead of the previous definition, let us take $\mathsf{A}'' = \mathsf{D_x A' D_y} = \mathsf{D_x V_u\, A\, W_v D_y}$, for x and y as above. Following the proof of [22, Theorem 5.6.2], we can use the Cauchy-Binet formula to express the minors of A'' in terms of those of $\mathsf{V_u}$, A and $\mathsf{W_v}$ as follows. Let $i \le \mathrm{rank}(\mathsf{A})$ and let $I = \{1, \ldots, i\}$. The determinant δ_i of the ith leading principal minor of A'' is the sum over all $J = \{j_1, \ldots, j_i\} \subset \{1, \ldots, m\}$ and $K = \{k_1, \ldots, k_i\} \subset \{1, \ldots, n\}$ of the terms $\alpha_{I,J}\, \beta_{J,K}\, \gamma_{K,I}\, x_{j_1} \cdots x_{j_i}\, y_{k_1} \cdots y_{k_i}$, where $\alpha_{I,J}$ is the determinant of $(\mathsf{V_u})_{I,J}$, $\beta_{J,K}$ is the determinant of $\mathsf{A}_{J,K}$, and $\gamma_{K,I}$ is the determinant of $(\mathsf{W_v})_{K,I}$. In particular, δ_i is a non-zero polynomial in the entries of $\mathsf{u}, \mathsf{v}, \mathsf{x}, \mathsf{y}$, of degree at most $pm, pm, 1, 1$ in these respective blocks of variables. Taking all $i \le \mathrm{rank}(\mathsf{A}) \le p$ into account, we see that there exists a non-zero polynomial Δ' in variables as above, with degree at most p^2m, p^2m, p, p in these variables, such that the non-vanishing of Δ' implies that A'' has generic rank profile. (The original statement of [22, Theorem 5.6.2] uses Cauchy premultipliers of the form $\mathsf{C_{a,u}}$ and $\mathsf{C_{v,b}}$ instead of our Vandermonde ones, for which all submatrice are invertible).

The advantage of this construction is that given generators $(\mathsf{G}', \mathsf{H}')$ of A', we deduce those of A'' simply as $\mathsf{G}'' = \mathsf{D_x G}'$ and $\mathsf{H}'' = \mathsf{D_y H}'$; in particular, they still have length $p+q+2$.

3. OVER AN ABSTRACT FIELD

In this section, we work over an arbitrary field $\mathbb{K}$, and we explain how to solve Problem A by using Pan's reduction to the Cauchy-like Problem B below; this latter task, involving a Cauchy-like system, will occupy most of the section.

Even though our goal is to solve a linear system, the algorithms do slightly more: they compute the inverse of a given matrix (or of a maximal minor thereof); this is similar to what happens for dense matrices, where we do not know how to solve linear systems with an exponent better than ω.

We can then state the main question of this section; remark that the assumptions on the vectors u, v are slightly stronger than the one required for $\nabla_{\mathsf{u},\mathsf{v}}$ to be invertible.

PROBLEM B. *Consider vectors $\mathsf{u} = (u_1, \ldots, u_m)$ and $\mathsf{v} = (v_1, \ldots, v_n)$, with $u_i \ne v_j$, $u_i \ne u_{i'}$, $v_j \ne v_{j'}$ for all i, i', j, j'. Given $\nabla_{\mathsf{u},\mathsf{v}}$-generators of length α for a matrix A in $\mathbb{K}^{m\times n}$, with $\alpha \le \min(m,n)$, do the following:*

- *if A does not have generic rank profile, raise an error,*
- *else, return $\nabla_{\mathsf{v}',\mathsf{u}'}$-generators for the inverse of the leading principal minor of A, with $\mathsf{v}' = (v_1, \ldots, v_r)$ and $\mathsf{u}' = (u_1, \ldots, u_r)$, where $r = \mathrm{rank}(\mathsf{A})$.*

There exist two major classes of algorithms for handling such problems, iterative ones, of cost that grows like mn (for fixed α), and algorithms using divide-and-conquer techniques, of quasi-linear cost in $m+n$. We stress that having a fast quadratic-time algorithm is actually crucial in practice: as is the case for the Half-GCD, fast linear algebra algorithms, etc, the quasi-linear algorithm will fall back on the quadratic one for input sizes under a certain threshold, and the performance of the latter will be an important factor in the overall runtime.

3.1 Previous results

Iterative algorithms that solve a size n Toeplitz system in time $O(n^2)$ have been known for decades [18, 8, 23]; extensions to structured matrices were later given, as for instance in [13]. For the particular form of our Problem B, we give in Section 3.2 an algorithm inspired by [20, Algorithme 4]. In this reference, Mouilleron sketches an algorithm that solves Problem B in time $O(\alpha n^2)$, in the case where $m = n$ and A is invertible (but without the rank profile assumption); he credits the origin of this algorithm to Kailath [15, §1.10], who dealt with symmetric matrices. Our algorithm follows the same pattern, but reduces the cost to $O(\alpha^{\omega-2} n^2)$.

In Section 3.3, we review divide-and-conquer techniques. Kaltofen [16] gave a divide-and-conquer algorithm that solves the analogue of Problem B for Toeplitz-like matrices, lifting assumptions of (strong) non-singularity needed in the original Morf and Bitmead-Anderson algorithm [19, 1]; a generalization of his algorithm to most usual structures is in [22]. A further improvement due to Jeannerod and Mouilleron [12], following Cardinal's work [6], allows one to bypass costly "compression" stages that were needed in Kaltofen's algorithm and its extensions, by predicting the shape of the generators we have to compute. For the case of square Cauchy-like matrices of size n, this results in an algorithm of cost $O(\alpha^2 \mathscr{M}(n) \log(n)^2)$; as pointed out in [**?**], the cost can be reduced to $O(\alpha^2 \mathscr{M}(n) \log(n))$ by choosing vectors u and v adequately.

3.2 A faster quadratic algorithm

Let $(\mathsf{G}, \mathsf{H}) \in \mathbb{K}^{m\times\alpha} \times \mathbb{K}^{n\times\alpha}$ be $\nabla_{\mathsf{u},\mathsf{v}}$-generators of a matrix A with respect to the operator $\nabla_{\mathsf{u},\mathsf{v}}$, with $\mathsf{u} = (u_1, \ldots, u_m)$ and $\mathsf{v} = (v_1, \ldots, v_n)$. Let further r be the rank of A. Our goal is to decide if A has generic rank profile, and if so, to return generators $(\mathsf{Y}, \mathsf{Z}) \in \mathbb{K}^{r\times\alpha} \times \mathbb{K}^{r\times\alpha}$ of the inverse of the leading principal minor of A.

Let $p = \min(m, n)$. For i in $\{0, \ldots, p\}$, write A as

$$\mathsf{A} = \begin{bmatrix} \mathsf{A}_{0,0}^{(i)} & \mathsf{A}_{0,1}^{(i)} \\ \mathsf{A}_{1,0}^{(i)} & \mathsf{A}_{1,1}^{(i)} \end{bmatrix}.$$

If the principal minor $\mathsf{A}_{0,0}^{(i)}$ is invertible, we define as in [6]

$$\mathsf{S}^{(i)} = \begin{bmatrix} \mathsf{A}_{0,0}^{(i)\,-1} & -\mathsf{A}_{0,0}^{(i)\,-1}\mathsf{A}_{0,1}^{(i)} \\ \mathsf{A}_{1,0}^{(i)}\mathsf{A}_{0,0}^{(i)\,-1} & \mathsf{A}_{1,1}^{(i)} - \mathsf{A}_{1,0}^{(i)}\mathsf{A}_{0,0}^{(i)\,-1}\mathsf{A}_{0,1}^{(i)} \end{bmatrix}.$$

We next write the decompositions $\mathsf{G} = \begin{bmatrix} \mathsf{G}_0^{(i)} \\ \mathsf{G}_1^{(i)} \end{bmatrix}$, $\mathsf{H} = \begin{bmatrix} \mathsf{H}_0^{(i)} \\ \mathsf{H}_1^{(i)} \end{bmatrix}$, with $\mathsf{G}_0^{(i)}$ and $\mathsf{H}_0^{(i)}$ of size $i \times \alpha$ and we define

$$\mathsf{Y}^{(i)} = \begin{bmatrix} \mathsf{Y}_0^{(i)} \\ \mathsf{Y}_1^{(i)} \end{bmatrix}, \quad \mathsf{Z}^{(i)} = \begin{bmatrix} \mathsf{Z}_0^{(i)} \\ \mathsf{Z}_1^{(i)} \end{bmatrix}, \tag{3}$$

with

$$\mathsf{Y}_0^{(i)} = -\mathsf{A}_{0,0}^{(i)\,-1}\mathsf{G}_0^{(i)}, \quad \mathsf{Y}_1^{(i)} = -\mathsf{A}_{1,0}^{(i)}\mathsf{A}_{0,0}^{(i)\,-1}\mathsf{G}_0^{(i)} + \mathsf{G}_0^{(i)}, \tag{4}$$

$$\mathsf{Z}_0^{(i)} = \mathsf{A}_{0,0}^{(i)\,-t}\mathsf{H}_0^{(i)}, \quad \mathsf{Z}_1^{(i)} = -\mathsf{A}_{0,1}^{(i)\,t}\mathsf{A}_{0,0}^{(i)\,-t}\mathsf{H}_0^{(i)} + \mathsf{H}_1^{(i)}. \tag{5}$$

Given integers a, b, $\mathsf{u}_{a:b}$ denotes the sequence $(u_a, \ldots, u_b)$ (and similarly for $\mathsf{v}_{a:b}$); we then define $\mathsf{e}^{(i)} = (\mathsf{v}_{1:i}, \mathsf{u}_{[i+1:m)}$ and $\mathsf{f}^{(i)} = (\mathsf{u}_{1:i}, \mathsf{v}_{i+1:n})$. Then, a key result for the sequel is the following, which is [6, Proposition 1]; remark that the operator $\nabla_{\mathsf{e}^{(i)},\mathsf{f}^{(i)}}$ is invertible, in view of our assumption on u and v.

LEMMA 1. *$(\mathsf{Y}^{(i)}, \mathsf{Z}^{(i)})$ are $\nabla_{\mathsf{e}^{(i)},\mathsf{f}^{(i)}}$-generators for $\mathsf{S}^{(i)}$.*

For $i = 0$, we simply have $\mathsf{S}^{(0)} = \mathsf{A}$ and $(\mathsf{Y}^{(0)}, \mathsf{Z}^{(0)}) = (\mathsf{G}, \mathsf{H})$. If A has generic rank profile, then the above lemma shows that for $i = r$, $(\mathsf{Y}_0^{(r)}, \mathsf{Z}_0^{(r)})$ are $\nabla_{\mathsf{v}',\mathsf{u}'}$-generators for $\mathsf{A}_{0,0}^{(r)\,-1}$, for u', v' as in Problem B, so they solve our problem. These facts will allow us to devise an iterative algorithm that starts from $(\mathsf{Y}^{(0)}, \mathsf{Z}^{(0)})$ and computes $(\mathsf{Y}^{(i_1)}, \mathsf{Z}^{(i_1)})$, $(\mathsf{Y}^{(i_2)}, \mathsf{Z}^{(i_2)})$, ... for some sequence of indices $0 < i_1 < i_2 < \ldots$, until we finally reach $(\mathsf{Y}^{(r)}, \mathsf{Z}^{(r)})$, on which we read off our output; if A does not have generic rank profile, we will detect it. The basis of the algorithm is Lemma 2 below, which uses the following notation.

Let i, j be non-negative integers with $0 \le i + j \le p$, and $\mathsf{A}_{0,0}^{(i)}$ invertible. Write a block decomposition of $\mathsf{S}^{(i)}$ as

$$\mathsf{S}^{(i)} = \begin{bmatrix} \mathsf{S}_{0,0}^{(i,j)} & \mathsf{S}_{0,1}^{(i,j)} & \mathsf{S}_{0,2}^{(i,j)} \\ \mathsf{S}_{1,0}^{(i,j)} & \mathsf{S}_{1,1}^{(i,j)} & \mathsf{S}_{1,2}^{(i,j)} \\ \mathsf{S}_{2,0}^{(i,j)} & \mathsf{S}_{2,1}^{(i,j)} & \mathsf{S}_{2,2}^{(i,j)} \end{bmatrix}, \tag{6}$$

with $\mathsf{S}_{0,0}^{(i,j)} = \mathsf{A}_{0,0}^{(i)\,-1}$ of size $i \times i$ and $\mathsf{S}_{1,1}^{(i,j)}$ of size $j \times j$ (from this, the sizes of all blocks can be deduced). Similarly, we can refine our decompositions of $\mathsf{Y}^{(i)}$ and $\mathsf{Z}^{(i)}$ as

$$\mathsf{Y}^{(i)} = \begin{bmatrix} \mathsf{Y}_0^{(i)} \\ \mathsf{Y}_1^{(i,j)} \\ \mathsf{Y}_2^{(i,j)} \end{bmatrix}, \quad \mathsf{Z}^{(i)} = \begin{bmatrix} \mathsf{Z}_0^{(i)} \\ \mathsf{Z}_1^{(i,j)} \\ \mathsf{Z}_2^{(i,j)} \end{bmatrix},$$

with $\mathsf{Y}_1^{(i,j)}$ and $\mathsf{Z}_2^{(i,j)}$ of size $j \times \alpha$.

LEMMA 2. *$\mathsf{A}_{0,0}^{(i+j)}$ has generic rank profile if and only if $\mathsf{S}_{1,1}^{(i,j)}$ does, and $\operatorname{rank}(\mathsf{A}_{0,0}^{(i+j)}) = i + \operatorname{rank}(\mathsf{S}_{1,1}^{(i,j)})$. If these matrices are invertible, we have the equalities*

$$\mathsf{Y}^{(i+j)} = \begin{bmatrix} \mathsf{Y}_0^{(i)} - \mathsf{S}_{0,1}^{(i,j)}\mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{Y}_1^{(i,j)} \\ -\mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{Y}_1^{(i,j)} \\ \mathsf{Y}_2^{(i,j)} - \mathsf{S}_{2,1}^{(i,j)}\mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{Y}_1^{(i,j)} \end{bmatrix}$$

and

$$\mathsf{Z}^{(i+j)} = \begin{bmatrix} \mathsf{Z}_0^{(i)} - \mathsf{S}_{1,0}^{(i,j)t}\mathsf{S}_{1,1}^{(i,j)\,-t}\mathsf{Z}_1^{(i,j)} \\ \mathsf{S}_{1,1}^{(i,j)\,-t}\mathsf{Z}_1^{(i,j)} \\ \mathsf{Z}_2^{(i,j)} - \mathsf{S}_{1,2}^{(i,j)t}\mathsf{S}_{1,1}^{(i,j)\,-t}\mathsf{Z}_1^{(i,j)} \end{bmatrix}.$$

PROOF. By construction, $\mathsf{S}_{1,1}^{(i,j)}$ is the Schur complement of $\mathsf{A}_{0,0}^{(i)}$, seen as a submatrix of $\mathsf{A}_{0,0}^{(i+j)}$; this proves our first claim. From the definition of $\mathsf{S}^{(i)}$ in (6), a direct calculation shows that S_{i+j} is given by

$$\mathsf{S}_{i+j} = \begin{bmatrix} \mathsf{J}_{0,0}^{(i,j)} & \mathsf{S}_{0,1}^{(i,j)}\mathsf{S}_{1,1}^{(i,j)\,-1} & \mathsf{J}_{0,2}^{(i,j)} \\ -\mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{S}_{1,0}^{(i,j)} & \mathsf{S}_{1,1}^{(i,j)\,-1} & -\mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{S}_{1,2}^{(i,j)} \\ \mathsf{J}_{2,0}^{(i,j)} & \mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{S}_{2,1}^{(i,j)} & \mathsf{J}_{2,0}^{(i,j)} \end{bmatrix},$$

with $\mathsf{J}_{a,b}^{(i,j)} = \mathsf{S}_{a,b}^{(i,j)} - \mathsf{S}_{a,1}^{(i,j)}\mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{S}_{1,b}^{(i,j)}$ for all a, b. In other words, up to permuting the first two blocks of rows, and the first two blocks of columns, $\mathsf{S}^{(i+j)}$ is derived from $\mathsf{S}^{(i)}$ in the same manner that $\mathsf{S}^{(i)}$ is derived from A. We may thus apply the rule that we used in Lemma 1 to derive $(\mathsf{Y}^{(i)}, \mathsf{Z}^{(i)})$ from (G, H) in order to derive $(\mathsf{Y}^{(i+j)}, \mathsf{Z}^{(i+j)})$ from $(\mathsf{Y}^{(i)}, \mathsf{Z}^{(i)})$. Taking into account the permutations we applied, we obtain the formulas given above. □

We can then describe the basic iterative step of our algorithm; we will use a step size $\beta \in \{1, \ldots, \alpha\}$, given as a parameter. Suppose that we have found that $\mathsf{A}_{0,0}^{(i)}$ has rank i, with generic rank profile, and that we have computed $(\mathsf{Y}^{(i)}, \mathsf{Z}^{(i)})$, for some index i in $\{0, \ldots, p\}$.

0. If $i = p$, return $(\mathsf{Y}_0^{(p)}, \mathsf{Z}_0^{(p)})$.

1. Let $j = \min(\beta, p - i) \ge 1$, and compute the matrices $\mathsf{S}_{1,0}^{(i,j)}$, $\mathsf{S}_{1,1}^{(i,j)}$, $\mathsf{S}_{1,2}^{(i,j)}$, $\mathsf{S}_{0,1}^{(i,j)}$ and $\mathsf{S}_{0,2}^{(i,j)}$ of (6). Since $(\mathsf{Y}^{(i)}, \mathsf{Z}^{(i)})$ are $\nabla_{\mathsf{e}^{(i)},\mathsf{f}^{(i)}}$-generators of $\mathsf{S}^{(i)}$, $(\mathsf{Y}_1^{(i,j)}, \mathsf{Z}^{(i)})$ and $(\mathsf{Y}^{(i)}, \mathsf{Z}_1^{(i,j)})$ are respectively $\nabla_{\mathsf{u}_{i+1:i+j},\mathsf{f}^{(i)}}$-generators and $\nabla_{\mathsf{e}^{(i)},\mathsf{v}_{i+1:i+j}}$-generators of its submatrices

$$\begin{bmatrix} \mathsf{S}_{1,0}^{(i,j)} & \mathsf{S}_{1,1}^{(i,j)} & \mathsf{S}_{1,2}^{(i,j)} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathsf{S}_{0,1}^{(i,j)} \\ \mathsf{S}_{1,1}^{(i,j)} \\ \mathsf{S}_{2,1}^{(i,j)} \end{bmatrix}.$$

Using block matrix multiplication in Eq. (2), since $j \le \beta \le \alpha$, we see that we can recover all these matrices in time $O(\beta^{\omega-2}\alpha(m+n))$.

2. If $\mathsf{S}_{1,1}^{(i,j)}$ does not have generic rank profile, raise an error. Else, compute its rank ρ. This costs $O(\beta^\omega)$.

3. If $\rho = j$, $\mathsf{A}_{0,0}^{(i+j)}$ has generic rank profile and rank $i + j$. Compute $(\mathsf{Y}^{(i+j)}, \mathsf{Z}^{(i+j)})$ by means of Lemma 2 and re-enter Step **0** with index $i+j$. To compute $\mathsf{Y}^{(i+j)}$, we first compute $\mathsf{S}_{1,1}^{(i,j)\,-1}\mathsf{Y}_1^{(i,j)}$ in time $O(\beta^{\omega-1}\alpha)$; then all other operations take time $O(\beta^{\omega-2}\alpha m)$. Similarly, computing $\mathsf{Z}^{(i+j)}$ takes time $O(\beta^{\omega-2}\alpha n)$.

4. If $\rho < j$, $\mathsf{A}_{0,0}^{(i+\rho)}$ has generic rank profile and rank $i + \rho$, but $\mathsf{A}_{0,0}^{(i+\rho+1)}$ still has rank $i + \rho$. Thus, either A has rank

$i+\rho$, in which case we are done, or it has rank greater than $i+\rho$, in which case it does not have generic rank profile.

As a result, we go over Step **1** again, with index ρ instead of j, and we now deduce the generators $(\mathsf{Y}^{(i+\rho)}, \mathsf{Z}^{(i+\rho)})$, as we did in the previous item (the cost remains of the same order). This allows us to write down $\mathsf{S}^{(i+\rho)}$, on which we can read off the Schur complement $\mathsf{A}_{1,1}^{(i+\rho)} - \mathsf{A}_{1,0}^{(i+\rho)}\mathsf{A}_{0,0}^{(i+\rho)\,-1}\mathsf{A}_{0,1}^{(i+\rho)}$. If it vanishes, we return $(\mathsf{Y}_0^{(i+\rho)}, \mathsf{Z}_0^{(i+\rho)})$; else, we raise an error. Computing $\mathsf{S}^{(i+\rho)}$ from its $\nabla_{\mathsf{e}^{(i+\rho)},\mathsf{f}^{(i+\rho)}}$-generators $(\mathsf{Y}^{(i+\rho)}, \mathsf{Z}^{(i+\rho)})$ is done by means of (2), using block matrix multiplication, with a cost $O(\alpha^{\omega-2}mn)$.

The cost of Steps **0-3** is $O(\beta^{\omega-2}\alpha(m+n))$; we enter these steps $O(p/\beta)$ times, for a total cost of $O(\beta^{\omega-3}\alpha p(m+n))$, which is $O(\beta^{\omega-3}\alpha mn)$. This dominates the cost of Step **4** (which we enter once, at most), so that the whole running time is $O(\beta^{\omega-3}\alpha mn)$. The algorithm of [20] uses $\beta = 1$, for which the cost is $O(\alpha mn)$; choosing $\beta = \alpha$, we benefit from fast matrix multiplication, as the cost drops to $O(\alpha^{\omega-2}mn)$, as claimed previously.

3.3 The divide-and-conquer algorithm

We now review the divide-and-conquer approach to solving Problem B, and discuss a constant factor improvement for the case of certain Cauchy-like matrices.

Let $\mathsf{A}, \mathsf{G}, \mathsf{H}, p, r$ be as in the previous subsection, and for i in $\{0, \ldots, p\}$, define $\mathsf{A}_{0,0}^{(i)}, \mathsf{A}_{0,1}^{(i)}, \mathsf{A}_{1,0}^{(i)}, \mathsf{A}_{1,1}^{(i)}, \ldots$ as before. The algorithm will compute the rank r of A, together with specific generators for the inverse of the leading principal minor $\mathsf{A}_{r,0,0}$, namely $\mathsf{Y}_0^{(r)} = -\mathsf{A}_{0,0}^{(r)\,-1}\mathsf{G}_0^{(r)}$ and $\mathsf{Z}_0^{(r)} = \mathsf{A}_{0,0}^{(r)\,-t}\mathsf{H}_0^{(r)}$, as did the algorithm in the previous subsection.

We choose i in $\{0, \ldots, p\}$, and we proceed as follows, essentially following [12], with the minor difference that do not assume A invertible, and that we explicitly check if A satisfies the generic rank profile assumption.

0. If i is less than a certain fixed threshold, return the output of the algorithm of the previous section.

1. Call the algorithm recursively for the submatrix $\mathsf{A}_{0,0}^{(i)}$, given by its $\nabla_{\mathsf{u}_{1:i},\mathsf{v}_{1:i}}$-generators $(\mathsf{G}_0^{(i)}, \mathsf{H}_0^{(i)})$. This will raise an error if $\mathsf{A}_{0,0}^{(i)}$ does not have generic rank profile, so we assume we are not in that case, and that we know $\rho = \mathrm{rank}(\mathsf{A}_{0,0}^{(i)})$, $\mathsf{Y}_0^{(\rho)} = -\mathsf{A}_{0,0}^{(\rho)\,-1}\mathsf{G}_0^{(\rho)}$ and $\mathsf{Z}_0^{(\rho)} = \mathsf{A}_{0,0}^{(\rho)\,-t}\mathsf{H}_0^{(\rho)}$.

2. Compute $(\mathsf{Y}_1^{(\rho)}, \mathsf{Z}_1^{(\rho)})$ using (4) and (5). The dominant cost is that of computing the products $\mathsf{A}_{1,0}^{(\rho)}\mathsf{Y}_0^{(\rho)}$ and $\mathsf{A}_{0,1}^{(\rho)t}\mathsf{Z}_0^{(\rho)}$, given the $\nabla_{\mathsf{u}_{\rho+1:m},\mathsf{v}_{1:\rho}}$-generators $(\mathsf{G}_1^{(\rho)}, \mathsf{H}_0^{(\rho)})$ of $\mathsf{A}_{1,0}^{(\rho)}$, and the $\nabla_{\mathsf{u}_{1:\rho},\mathsf{v}_{\rho+1:n}}$-generators $(\mathsf{G}_0^{(\rho)}, \mathsf{H}_1^{(\rho)})$ of $\mathsf{A}_{0,1}^{(\rho)}$. We examine this cost below.

3. If $\rho < i$, test whether the Schur complement $\mathsf{A}_{1,1}^{(\rho)} - \mathsf{A}_{1,0}^{(\rho)}\mathsf{A}_{0,0}^{(\rho)\,-1}\mathsf{A}_{0,1}^{(\rho)}$ vanishes, or equivalently whether $\mathsf{Y}_1^{(\rho)}\,\mathsf{Z}_1^{(\rho)t} = 0$: if so, return $(\mathsf{Y}_0^{(\rho)}, \mathsf{Z}_0^{(\rho)})$; else, raise an error. This is done by finding a minimal set of independent rows in $\mathsf{Z}_1^{(\rho)}$ (this takes time $O(\alpha^{\omega-1}n)$) and multiplying their transposes by $\mathsf{Y}_1^{(\rho)}$ (there are at most α such rows, so this takes time $O(\alpha^{\omega-1}n)$ as well).

4. If $\rho = i$, call the algorithm recursively for the Schur complement $\mathsf{A}_{1,1}^{(i)} - \mathsf{A}_{1,0}^{(i)}\mathsf{A}_{0,0}^{(i)\,-1}\mathsf{A}_{0,1}^{(i)}$, given by its $\nabla_{\mathsf{u}_{i+1:m},\mathsf{v}_{i+1:n}}$-generators $(\mathsf{Y}_1^{(i)}, \mathsf{Z}_1^{(i)})$. This will raise an error if A does not have generic rank profile. Else, we obtain the rank σ of the Schur complement, as well as $\nabla_{\mathsf{v}_{i+1:i+\sigma},\mathsf{u}_{i+1:i+\sigma}}$-generators for its leading principal minor; with our notation, they are $-\mathsf{S}_{1,1}^{(i,\sigma)\,-1}\mathsf{Y}_1^{(i,\sigma)}$ and $\mathsf{S}_{1,1}^{(i,\sigma)\,-t}\mathsf{Z}_1^{(i,\sigma)}$.

5. We compute and return $(\mathsf{Y}_0^{(i+\sigma)}, \mathsf{Z}_0^{(i+\sigma)})$ using the formulas of Lemma 2. This is done by multiplying the above matrices by respectively $\mathsf{S}_{0,1}^{(i,\sigma)}$, given through its $\nabla_{\mathsf{u}_{i+1:i+\sigma},\mathsf{u}_{1:i}}$-generator $(\mathsf{Y}_0^{(i)}, \mathsf{Z}_1^{(i,\sigma)})$, and $\mathsf{S}_{1,0}^{(i,\sigma)t}$, given by its $\nabla_{\mathsf{v}_{1:i},\mathsf{v}_{i+1:i+\sigma}}$-generator $(\mathsf{Y}_1^{(i,\sigma)}, \mathsf{Z}_0^{(i)})$.

The bottleneck in this algorithm, at Steps **2** and **4**, is the multiplication of a Cauchy-like matrix of size roughly $m \times n$, given by generators of lengh α, by α vectors, either on the left or on the right. As explained in Section 2, u and v are chosen as geometric progressions *with the same ratio*; if this is the case at the top-level, this will remain the case for all recursive calls; this allows us to reduce the matrix-vector products by all Cauchy matrices involved here to Toeplitz matrix-vector products. Choosing $i = \lceil p/2 \rceil$ balances the costs of the two recursive calls; as a result, the overall runtime of the algorithm is $O(\alpha^2 \mathscr{M}(p')\log(p))$.

4. MODULAR TECHNIQUES

We now address Problem A in the particular case where $\mathbb{K} = \mathbb{Q}$; without loss of generality, we assume that the entries of T are integers.

Several *modular algorithms* are available to solve dense linear systems over $\mathbb{Q}$. A first option relies on the Chinese Remainder Theorem, solving the system modulo several primes $p_1, p_2, \ldots$ before reconstructing the solution, making sure to ensure consistency of the modular solutions when $\ker(\mathsf{T})$ has dimension more than 1. Other approaches such as Newton iteration or divide-and-conquer algorithms use one prime p, and lift the solution modulo powers of p.

4.1 Reduction to nullity one

Many of these techniques rely on the knowledge of a maximal invertible minor of A. However, if A is mosaic Toeplitz, there is no guarantee that it possesses such a minor that would be mosaic Toeplitz as well. Hence, we are going to rely on the transformation to the Cauchy structure / regularization technique of 2. Working over $\mathbb{Q}$, the solution we used before has a certain shortcoming. Our output is a vector b of the form $\mathsf{b} = \mathsf{W}_\mathsf{v}[(-\mathsf{A}_r^{-1}\mathsf{c})^t\ \mathsf{c}^t]^t$, with $\mathsf{A} = \mathsf{V}_\mathsf{u}\mathsf{T}\mathsf{W}_\mathsf{v}$, A_r^{-1} a maximal minor of it and c a random vector. Due to the preconditionning, the entries of b are expected of large height (the upper bounds one can naively derive from the formula above seem however to be overly crude).

When $\ker(\mathsf{T})$ has dimension 1, we are not affected by this issue. Indeed, in this case, all solutions are of the form $\lambda\mathsf{b}_0$, for some vector $\mathsf{b}_0 \in \mathbb{Z}^n$ whose bit-size can be bounded only in terms of T, by an expression in $O\tilde{}(nh)$, if h is a bound on the bit-size of the entries of T (for instance by means of Siegel's lemma). Hence, it suffices to compute the solution b of the regularized system modulo a large enough integer N and normalize it by setting one of its entries to 1; this gives us the normalization of b_0 mod N.

For an arbitrary T, our suggestion is to reduce to nullity 1 by adding extra rows to T

In the important particular case of algebraic approximation, we suggest another method.

4.2 The divide-and-conquer algorithm

4.3 The divide-and-conquer algorithm

5. REFERENCES

[1] 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.

[2] A. Bostan, F. Chyzak, M. Giusti, R. Lebreton, G. Lecerf, B. Salvy, and É. Schost. Algorithmes efficaces en calcul formel. https://specfun.inria.fr/chyzak/mpri/poly.pdf.

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

[4] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *J. Complexity*, 21(4):420–446, 2005.

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

[6] J.-P. Cardinal. On a property of Cauchy-like matrices. *C. R. Acad. Sci. Paris Série I*, 388:1089–1093, 1999.

[7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, March 1990.

[8] J. Durbin. The fitting of time series models. *Rev. Inst. Int. Stat.*, 28:233–243, 1960.

[9] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, third edition, 2013.

[10] G. Hanrot, M. Quercia, and P. Zimmermann. The Middle Product Algorithm, I. *Appl. Algebra Engrg. Comm. Comp.*, 14(6):415–438, 2004.

[11] G. Heinig and T. Amdeberhan. On the inverses of Hankel and Toeplitz mosaic matrices. In *Seminar Analysis (Berlin, 1987/1988)*, pages 53–65. Akademie-Verlag, Berlin, 1988.

[12] C.-P. Jeannerod and C. Mouilleron. Computing specified generators of structured matrix inverses. In *ISSAC'10*, pages 281–288, 2010.

[13] T. Kailath, I. Gohberg, and V. Olshevsky. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. *Math. Comp.*, 64(212):1557–1576, 1995.

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

[15] T. Kailath and A. H. Sayed. *Fast Reliable Algorithms for Matrices with Structure*. SIAM, 1999.

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

[17] F. Le Gall. Powers of tensors and fast matrix multiplication. In *ISSAC'14*, pages 296–303. ACM, 2014.

[18] N. Levinson. The Wiener RMS error criterion in filter design and prediction. *J. Math. Phys.*, 25:261–278, 1947.

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

[20] C. Mouilleron. Algorithmes rapides pour la résolution de problèmes algébriques structurés. Master's thesis, ENS Lyon, 2008.

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

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

[23] W. F. Trench. An algorithm for the inversion of finite Toeplitz matrices. *J. Soc. Indust. Appl. Math.*, 12:515–522, 1964.