A Two-pronged Progress in Structured Dense Matrix Vector Multiplication

Christopher De Sa* Albert Gu† Rohan Puttagunta‡ Christopher Ré§ Atri Rudra¶

Abstract

Matrix-vector multiplication is one of the most fundamental computing primitives. Given a matrix $\mathbf{A} \in \mathbb{F}^{N \times N}$ and a vector $\mathbf{b} \in \mathbb{F}^N$, it is known that in the worst case $\Theta(N^2)$ operations over $\mathbb{F}$ are needed to compute $\mathbf{Ab}$. Many types of structured matrices do admit faster multiplication. However, even given a matrix $\mathbf{A}$ that is known to have this property, it is hard in general to recover a representation of $\mathbf{A}$ exposing the actual fast multiplication algorithm. Additionally, it is not known in general whether the inverses of such structured matrices can be computed or multiplied quickly. A broad question is thus to identify classes of *structured dense* matrices that can be represented with $O(N)$ parameters, and for which matrix-vector multiplication (and ideally other operations such as solvers) can be performed in a sub-quadratic number of operations.

One such class of structured matrices that admit near-linear matrix-vector multiplication are the *orthogonal polynomial transforms* whose rows correspond to a family of orthogonal polynomials. Other well known classes include the Toeplitz, Hankel, Vandermonde, Cauchy matrices and their extensions (e.g. confluent Cauchy-like matrices) that are all special cases of a low *displacement rank* property.

In this paper, we make progress on two fronts:

1. We introduce the notion of *recurrence width* of matrices. For matrices $\mathbf{A}$ with constant recurrence width, we design algorithms to compute both $\mathbf{Ab}$ and $\mathbf{A}^T\mathbf{b}$ with a near-linear number of operations. This notion of width is finer than all the above classes of structured matrices and thus we can compute near-linear matrix-vector multiplication for all of them using the same core algorithm. Furthermore, we show that it is possible to solve the harder problems of *recovering* the structured parameterization of a matrix with low recurrence width, and computing matrix-vector product with its *inverse* in near-linear time.
2. We additionally adapt our algorithm to a matrix-vector multiplication algorithm for a much more general class of matrices with displacement structure: those with low displacement rank with respect to quasiseparable matrices. This result is a novel connection between matrices with displacement structure and those with rank structure, two large but previously separate classes of structured matrices. This class includes Toeplitz-plus-Hankel-like matrices, the Discrete Trigonometric Transforms, and more, and captures all previously known matrices with displacement structure under a unified parameterization and algorithm.

Our work unifies, generalizes, and simplifies existing state-of-the-art results in structured matrix-vector multiplication. Finally, we show how applications in areas such as multipoint evaluations of multivariate polynomials can be reduced to problems involving low recurrence width matrices.

1 Introduction

Given a generic matrix $\mathbf{A} \in \mathbb{F}^{N \times N}$ over any field $\mathbb{F}$, the problem of matrix-vector multiplication by $\mathbf{A}$ has a clear $\Omega(N^2)$ lower bound in general.[1] Many special classes of matrices, however, admit multiplication algorithms that only require near linear (in N) operations. In general, any matrix $\mathbf{A}$ can be identified with the smallest linear circuit that computes the linear function induced by $\mathbf{A}$. This is a tight characterization of the best possible arithmetic complexity of any matrix-vector multiplication algorithm for $\mathbf{A}$ that uses linear operations[2] and captures all known structured matrix vector multiplication algorithms. Additionally, it implies the classical *transposition principle* [10, 14], which states that the number of linear operations needed for matrix-vector multiplication by $\mathbf{A}$ and $\mathbf{A}^T$ are within constant factors of each other. Thus, this quantity is a very general characterization of the complexity of a matrix. However, it has several shortcomings. Most importantly, given a matrix, the problem of finding the minimum circuit size is APX-hard [13], and the best known upper bound on the approximation ratio is only $O(N/\log N)$ [35]. Finally, this characterization does not say anything about the inverse of a structured matrix, even though $\mathbf{A}^{-1}$ is often also structured if $\mathbf{A}$ is. Thus, much work in the structured matrix vector multiplication literature has focused on the following problem:

> *Identify the most general class of structured matrices* $\mathbf{A}$ *so that one can in near-linear operations compute both* $\mathbf{Ab}$ *and* $\mathbf{A}^{-1}\mathbf{b}$. *In addition given an arbitrary matrix* $\mathbf{A}$, *we would*

*Computer Science Department, Cornell University.

†Computer Science Department, Stanford University.

‡Instagram, Inc.

§Computer Science Department, Stanford University.

¶Department of Computer Science and Engineering, University at Buffalo, SUNY.

[1] This is in terms of operations over $\mathbb{F}$ in exact arithmetic, which will be our primary focus throughout this paper. Also we will focus exclusively on computing the matrix vector product exactly as opposed to approximately. We leave the study of approximation and numerical stability (which are important for computation over real/complex numbers) for future work.

[2] Furthermore over any infinite field $\mathbb{F}$, non-linear operations do not help, i.e. the smallest linear circuit is within a constant factor size of the smallest arithmetic circuit [14].

like to efficiently recover the representation of the matrix in the chosen class.

One of most general classes studied so far is the structure of low *displacement rank*. The notion of displacement rank, which was introduced in the seminal work of Kailath et al. [25], is defined as follows. Given any pair of matrices $(\mathbf{L}, \mathbf{R})$, the displacement rank of $\mathbf{A}$ with respect to $(\mathbf{L}, \mathbf{R})$ is the rank of the *error matrix* [3]:

$$\mathbf{E} = \mathbf{LA} - \mathbf{AR}. \tag{1.1}$$

Until very recently, the most general structure in this framework was studied by Olshevsky and Shokrollahi [42], who show that any matrix with a displacement rank of r with respect to *Jordan form* matrices $\mathbf{L}$ and $\mathbf{R}$ supports near-linear operations matrix-vector multiplication. A very recent pre-print extended this result to the case when both $\mathbf{L}$ and $\mathbf{R}$ are *block companion matrices* [11]. In our first main result,

we substantially generalize these results to the case when $\mathbf{L}$ *and* $\mathbf{R}$ *are* quasiseparable *matrices.*

Quasiseparable matrices are a type of *rank-structured* matrix, defined by imposing low-rank constraints on certain submatrices, which have become widely used in efficient numerical computations [55]. This result represents a new unification of two large, important, and previously separate classes of structured matrices, namely those with displacement structure and those with rank structure.

Another general class of matrices are *orthogonal polynomial transforms* [1–3, 15]. We first observe that known results on such polynomials [12, 18, 49] can be easily extended to polynomial recurrences of bounded width. However, these results and those for displacement rank matrices tend to be proved using seemingly disparate algorithms and techniques.

In our second main result,

we introduce the notion of recurrence width*, which captures the class of orthogonal polynomial transforms as well as matrices of low displacement rank with respect to Jordan form matrices.*

We design a simple and general near-linear-operation matrix-vector multiplication algorithm for low recurrence width matrices, hence capturing these previously disparate classes. Moreover, we observe that we can solve the harder problems of recovery (i.e. recovering the recurrence width parameterization given a matrix) and inverse for the polynomials recurrences of bounded width in polynomial time. Figure 1 shows the relationship between the various classes of matrices that we have discussed, and collects the relevant known results and our new results. (All the algorithms result in linear circuits and hence by the transposition principle, we get algorithms for $\mathbf{A}^T\mathbf{b}$ and $\mathbf{A}^{-T}\mathbf{b}$ with the same complexity as $\mathbf{Ab}$ and $\mathbf{A}^{-1}\mathbf{b}$ respectively.)

We now focus on the two strands of work that inspired much of our work, and describe how we capture (and generalize) previous results in these areas. These strands have been well-studied and have rich and varied applications, which we summarize at the end of the section. Throughout this section we describe square matrices for simplicity, but we emphasize that the recurrence width concept applies to general matrices (see Definition 3.2).

Displacement rank. The approach of expressing structured matrices via their displacements and using this to define unified algorithms for multiplication and inversion has traditionally been limited to the four most popular classes of Toeplitz-like, Hankel-like, Vandermonde-like, and Cauchy-like matrices [22, 40]. These classic results on displacement rank constrain $\mathbf{L}, \mathbf{R}$ to diagonal or shift matrices. Until very recently, the most powerful results on matrix-vector multiplication for matrices with low displacement rank are in the work of Olshevsky and Shokrollahi [42], who show that any matrix with a displacement rank of r with respect to Jordan form matrices $\mathbf{L}$ and $\mathbf{R}$ can be multiplied by an arbitrary vector with $\tilde{O}(rN)$ operations. (They use these results and matrices to solve the Nevalinna-Pick problem as well as solve the interpolation step in some list decoding algorithms in a previous work [41].) Recall that Jordan normal form matrices are a special case of 2-*band upper triangular matrices*. In this work, by applying the recurrence width concept (Definition 1.2), we show that when both $\mathbf{L}$ and $\mathbf{R}$ are any triangular t-band matrices, matrices with displacement rank of r with respect to them admit fast multiplication. Furthermore, in the restricted case of $t = 2$ our result matches the previous bound [42].

In our following theorem and for the rest of the paper, let $\mathcal{M}(N)$ denote the cost of multiplying two polynomials of degree N over $\mathbb{F}$ (ranging from $N \log N$ to $N \log N \log\log N$) and ω be the matrix-matrix multiplication exponent.

THEOREM 1.1. *Let* $\mathbf{L}$ *and* $\mathbf{R}$ *be triangular* t*-band matrices sharing no eigenvalues, and let* $\mathbf{A}$ *be a matrix such that* $\mathbf{LA} - \mathbf{AR}$ *has rank* r*. Then with* $O(t^\omega \mathcal{M}(N) \log N)$

[3] This defines the Sylvester type displacement operator. Our algorithms work equally well for the Stein type displacement operator $\mathbf{A} - \mathbf{LAR}$. We also note that the terminology of error matrix to talk about the displacement operator is non-standard: we make this change to be consistent with our general framework where this terminology makes sense.

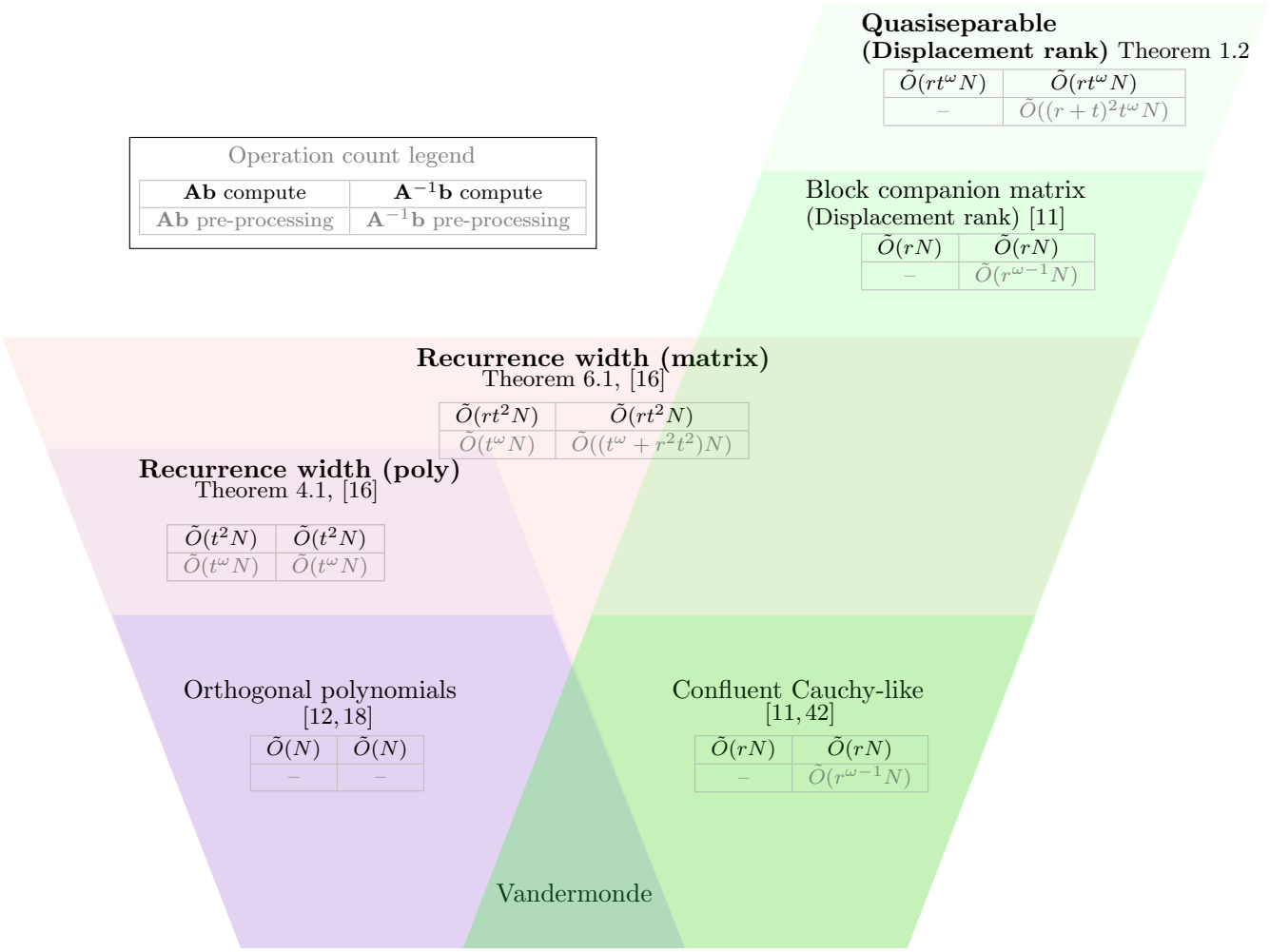

Figure 1: Overview of results and hierarchy of matrix classes. Operation count includes pre-processing (on $\mathbf{A}$ only) and computation (on $\mathbf{A}$ and $\mathbf{b}$) where $\tilde{O}(\cdot)$ hides polylog factors in N. Each class has a parameter controlling the degree of structure: t refers to the recurrence width or quasiseparability degree, and r indicates the displacement rank. ω denotes the exponent of matrix-matrix multiplication runtime.

pre-processing operations, $\mathbf{A}$ *and* $\mathbf{A}^T$ *can be multiplied by any vector* $\mathbf{b}$ *in* $O(rt^2\mathcal{M}(N)\log N)$ *operations. With* $O(t^\omega\mathcal{M}(N)\log N + r^2t^2\mathcal{M}(N)\log^2 N)$ *pre-processing operations,* $\mathbf{A}^{-1}$ *and* $\mathbf{A}^{-T}$ *can be multiplied by any vector in* $O(rt^2\mathcal{M}(N)\log N)$ *operations.*

In March 2017, a pre-print [11] generalized the result of [42] in a different direction, to the case where $\mathbf{L}$ and $\mathbf{R}$ are block companion matrices. We show an alternative technique that is slower than Theorem 1.1 by polylog factors, but works for even more general $\mathbf{L}$ and $\mathbf{R}$ that subsumes the classes of both Theorem 1.1 and [11]:

THEOREM 1.2. *Let* $\mathbf{L}$ *and* $\mathbf{R}$ *be* t*-quasiseparable matrices and* $\mathbf{E}$ *be rank* r *such that* $\mathbf{A}$ *is uniquely defined by* $\mathbf{LA} - \mathbf{AR} = \mathbf{E}$*. Then* $\mathbf{A}$ *and* $\mathbf{A}^T$ *admit matrix-vector multiplication in* $O(rt^\omega\mathcal{M}(N)\log^2 N + rt^2\mathcal{M}(N)\log^3 N)$ *operations. Furthermore, letting this cost be denoted* $M_t(N,r)$*, then with* $O((r+t\log N)M_t(N, r+t\log N)\log N)$ *operations,* $\mathbf{A}^{-1}$ *and* $\mathbf{A}^{-T}$ *also admit matrix-vector multiplication in* $M_t(N,r)$ *operations.*

The notion of t-quasiseparability refers to a type of rank-structured matrix where every submatrix not intersecting the diagonal has rank at most t [19] (see Definition 6.2). This class includes both the triangular banded matrices in Theorem 1.1 which are t-quasiseparable, and the block companion matrices of [11] which are 1-quasiseparable. Other well-known classes of matrices with displacement structure include Toeplitz-plus-Hankel-like matrices, all vari-

ants of Discrete Cosine/Sine Transforms, Chebyshev-Vandermonde-like matrices, and so on [39], and the displacement operators for all of these are tridiagonal which are 1-quasiseparable.[4] To the best of our knowledge, Theorem 1.2 is the most general result on near-linear time matrix vector multiplication for matrices with displacement rank structure, and in particular captures all currently known matrices with such structure.

Aside from strongly extending these previous important classes of structured matrices, this class of matrices is well-behaved and nice in its own right. They naturally inherit traits of displacement rank such as certain multiplicative and inversion closure properties [45]. Furthermore, the class of quasiseparable matrices, which was introduced relatively recently and are still under active study [19], generalize band matrices and their inverses and satisfy properties such as gradation under addition and multiplication[5] and closure under inversion. As one consequence, the Sylvester and Stein displacement classes with respect to quasiseparable operators are essentially identical.[6]

Orthogonal polynomial transforms. The second strand of work that inspired our results relates to *orthogonal polynomial transforms* [1–3,15]. Any matrix $\mathbf{A}$ can be represented as a polynomial transform, defined as follows.

DEFINITION 1.1. *Let $a_0(X), \ldots, a_{N-1}(X)$ be a collection of polynomials over a field $\mathbb{F}$ and $z_0, \ldots, z_{N-1}$ be a set of points. The discrete polynomial transform matrix $\mathbf{A}$ is defined by $\mathbf{A}[i,j] = a_i(z_j)$. The discrete polynomial transform of a vector $\mathbf{b}$ with respect to the a_i and z_j is given by the product $\mathbf{Ab}$.*

When the a_i are a family of orthogonal polynomials [15], we are left with an orthogonal polynomial transform. Orthogonal polynomials can be characterized by the following two term recurrence[7]

$$a_i(X) = (\alpha_i X + \beta_i)a_{i-1}(X) + \gamma_i a_{i-2}(X), \tag{1.2}$$

where $\alpha_i, \beta_i, \gamma_i \in \mathbb{F}$. Driscoll, Healy and Rockmore present an algorithm to perform the orthogonal polynomial transform over $\mathbb{F} = \mathbb{R}$ in $O(N \log^2 N)$ operations [18]. Later it was shown how to perform the transposed transform and inverse transform (i.e. multiplication by $\mathbf{A}^T$ and $\mathbf{A}^{-1}$) with the same complexity [12,49].

We observe that this class of transforms can be extended to polynomials that satisfy a more general recurrence. We introduce the notion of recurrence width which describes the degree of this type of structure in a matrix:

DEFINITION 1.2. *An $N \times N$ matrix $\mathbf{A}$ has recurrence width t if the polynomials $a_i(X) = \sum_{j=0}^{N-1} \mathbf{A}[i,j]X^j$ satisfy $\deg(a_i) \leq i$ for $i < t$, and*

$$a_i(X) = \sum_{j=1}^{t} g_{i,j}(X)a_{i-j}(X) \tag{1.3}$$

for $i \geq t$, where the polynomials $g_{i,j} \in \mathbb{F}[X]$ have degree at most j.

Note that under this definition, an orthogonal polynomial transform matrix has recurrence width 2.[8] The recurrence structure allows us to generate matrix vector multiplication algorithms for matrices $\mathbf{A}^T$ and $\mathbf{A}^{-1}$ (and for $\mathbf{A}$ and $\mathbf{A}^{-T}$ by the transposition principle) in a simple and general way.

THEOREM 1.3. *Let $\mathbf{A} \in \mathbb{F}^{N \times N}$ be a matrix with recurrence width t. With $O(t^\omega \mathcal{M}(N) \log N)$ pre-processing operations, the products $\mathbf{A}^T\mathbf{b}$ and $\mathbf{Ab}$ can be computed in $O(t^2 \mathcal{M}(N) \log N)$ operations for any vector $\mathbf{b}$, and the products $\mathbf{A}^{-1}\mathbf{b}$ and $\mathbf{A}^{-T}\mathbf{b}$ can be computed in $O(t^2 \mathcal{M}(N) \log^2 N)$ operations for any vector $\mathbf{b}$.*

In Section 4 we provide an algorithm for $\mathbf{A}^T\mathbf{b}$, and bound its complexity in Theorem 4.1. The statement for $\mathbf{Ab}$ then follows from the transposition principle (and more detail is provided in the full version of the paper [16]). The statement for $\mathbf{A}^{-1}\mathbf{b}$ (hence $\mathbf{A}^{-T}\mathbf{b}$) is proved in the full version of the paper [16]. Our algorithms are optimal in the sense that their complexity is equal to the worst-case input size of $\Theta(t^2 N)$ for a matrix with recurrence width t, up to log factors (and if $\omega = 2$, so is the pre-processing). In particular, we recover the bounds of Driscoll et al. of $O(\mathcal{M}(N) \log N)$ in the orthogonal polynomial case [18].

The connection. Theorem 1.1 and 1.3 are special cases of our results on the most general notion of recurrence width. This connection is compelling because the set of matrices with low recurrence width and those with low displacement rank seem to be widely different. Indeed the existing algorithms for the class of orthogonal polynomials [18] and low displacement rank [42] look very different. Specifically, the algorithm of Driscoll et

[4] Some variants allow the top right and bottom left corner elements to be non-zero; these are still 2-quasiseparable.

[5] For example, the sum of a p-quasiseparable matrix and q-quasiseparable matrix is $(p+q)$-quasiseparable.

[6] A matrix has low Sylvester displacement with respect to $\mathbf{L}, \mathbf{R}$ if and only if it has low Stein displacement with respect to $\mathbf{L}^{-1}, \mathbf{R}$.

[7] This is often called a three-term recurrence, but we will consider the number of terms on the RHS to be the recurrence length.

[8] There is a subtlety in that the orthogonal polynomial transform in Definition 1.1 can be factored as the product of the matrix in Definition 1.2 times a Vandermonde matrix on the z_j. This is covered by a generalization of Definition 1.2, see Definition 3.2 and Remark 3.1.

al. [18] is a divide and conquer algorithm, while that of Olshevsky and Shokrollahi [42] (and other works) heavily exploits structural algebraic properties on matrices with low displacement rank. Despite this, we show that both of these classes of matrices can be captured by a more abstract recurrence and handled with a single class of superfast algorithms. Arguably our definition is more natural since it makes polynomials front and center while existing matrices (definitons) only use polynomials in their algorithms. Thus, our definition makes this connection more explicit.

Similarly, both classic matrices with displacement structure [44] and those with rank structure [20] have large but separate bodies of literature, and Theorem 1.2 shows that their structure can be combined in a way that still admits efficient algorithms. We believe that unifying these existing threads of disparate work is interesting in its own right and our most important contribution. In Section 8, we provide a more detailed history of these threads.

Some Applications. Orthogonal polynomials and low displacement rank matrices have applications in numerous areas from signal processing to machine learning. Indeed orthogonal polynomials have their own dedicated conference [4]. For matrices with low displacement rank, the survey by Kailath and Sayed [26] covers more details and applications, which our matrices naturally inherit. The matrix structure we discuss is also related to notions in control theory arising from different motivations. The notion of displacement rank is a special case of the Sylvester equation which arises in the stability analysis of linear dynamical systems [50]. Additionally, the more general type of recurrence width matrices we study can be viewed as satisfying a type of higher-order Sylvester equation which has become increasingly more important in this area. Such equations are very difficult to analyze in full generality [50], and our notions represent one line of attack for efficient solutions to these problems.

As a more concrete application, there has been recent interest in the use of structured matrices as components of neural networks, intended to replace the expensive fully-connected layers (which are linear transformations represented as matrix-vector products) with fast and compressible structures. Many ad-hoc paradigms exist for representing structured matrices in neural networks [36], and it turns out several of them are instances of a displacement rank structure. More explicitly, a recent paper by Sindhwani et al. found that Toeplitz-like matrices (which have low displacement rank with respect to shift matrices) were much more effective than standard low-rank encodings for mobile speech recognition [51]. Additionally, the theoretical guarantees of using displacement rank to compress neural networks are beginning to be understood [59]. For the same number of parameters, our generalized classes of dense, full-rank structured matrices can be even more expressive than the standard structures explored so far, so they may be suitable for these applications.

Paper organization. Our paper is organized as follows. In Section 2, we provide an overview of our techniques. In Section 3, we formally define the most general notion of recurrence width and describe the main structural properties needed for our algorithms. In Section 4, we describe the full algorithm for $\mathbf{A}^T\mathbf{b}$ for a restricted class of recurrence width; the algorithm for this subclass contains all the core ideas and in particular is already more general than Definition 1.2. In Section 5, we describe the modifications to the $\mathbf{A}^T\mathbf{b}$ algorithm needed to handle the fully general definition of recurrence width. We also elaborate on related multiplication operations, including multiplication by $\mathbf{A}$, and multiplication by Krylov matrices. In Section 6 we provide details on the multiplication algorithms for matrices with low displacement rank. We first prove Theorem 1.1 by showing that those matrices have low recurrence width, and then adapt the techniques to cover the more general displacement structure of Theorem 1.2. In Section 7, we show that our new generalizations capture matrices from coding theory that were not captured by the earlier classes. In particular, we show that the matrix corresponding to multipoint multivariate polynomial evaluation has small recurrence width. We use this connection to show a *barrier* result: if one could design algorithms that are efficient enough in terms of sparsity of the input, then we would improve the state-of-the-art results in multipoint multivariate polynomial evaluation.

All missing details and additional results can be found in the full version of the paper [16].

2 Technical Overview

For ease of exposition, we start off with an overview of our techniques for low recurrence width matrices. We describe the basic recurrence width and a natural generalization that captures displacement rank with respect to triangular band matrices. Finally we show how to adapt these techniques to handle matrices with more general displacement structure.

As in all algorithms for structured matrices, the main challenge is in manipulating the alternate compact representation of the matrix directly. Although the recurrence (1.3) is represented with only $O(N)$ total values (for a fixed t), the polynomials $a_i(X)$ it produces are large: unlike conventional linear recurrences on scalars, the total output size of this recurrence is

quadratic because the polynomial degrees grow linearly (hence the matrix produced is dense). The first hurdle we clear is compressing this output using the recurrence structure (1.3), which lets us characterize the polynomials $a_i(X)$ in a simple way and leads to an intuitive algorithm for transpose multiplication. At its core, the algorithms exploit the self-similar structure of the recurrence (1.3). By definition, all the polynomials $a_i(X)$ are generated from $a_0(X), \ldots, a_{t-1}(X)$. But due to the recurrence structure, the higher-degree polynomials $a_{N/2}(X)$ through $a_{N-1}(X)$ can be thought of as being generated from $a_{N/2}(X), \ldots, a_{N/2+t-1}(X)$ (and the dependence on these generators is now lower-degree). This forms the basis of the divide-and-conquer algorithms.

We then consider a natural generalization consisting of recurrences on vectors, where the transitions involve multiplying by polynomial functions of a fixed matrix. We show a reduction to the standard polynomial case; this allows us to interpret structures such as the displacement rank equation (1.1), which does not seem to involve polynomials, as defining a polynomial recurrence. This reduction is represented through *Krylov matrices*, which appear in independently interesting settings on numerical methods such as the Lanczos eigenvalue algorithm and Wiedemann's kernel vector algorithm [28, 52]. For the cases we are interested in, these Krylov matrices themselves turn out to have low recurrence width.

Transpose multiplication. We will consider a more general form of the recurrence (1.3) that allows some generators to be added at every step. It turns out that this generalization does not increase the asymptotic cost of the algorithm but will be necessary to capture structure such as displacement rank (1.1). Suppose that the following recurrence holds for all i

$$a_i(X) = \sum_{j=1}^{\min(i,t)} g_{i,j}(X) a_{i-j}(X) + \sum_{k=0}^{r-1} c_{i,k} d_k(X), \tag{2.4}$$

for some fixed *generators* $d_0(X), \ldots, d_{r-1}(X)$. Notice that equation (1.3) is a special case with $r = t$, $d_i(X) = a_i(X)$ for $0 \le i < t$, and $c_{i,k} = \delta_{ik}$ for $0 \le i < n, 0 \le k < r$ (i.e. the t initial polynomials are generators and are never re-added).

Consider the simplified case when $r = 1$ and $d_0(X) = 1$, so that the recurrence becomes $a_i(X) = \sum_{j=1}^{\min(i,t)} g_{i,j}(X) a_{i-j}(X) + c_i$; this simplification captures the main difficulties. This can be written as

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -g_{1,1}(X) & 1 & 0 & \cdots & 0 \\ -g_{2,2}(X) & -g_{2,1}(X) & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_0(X) \\ a_1(X) \\ a_2(X) \\ \vdots \\ a_{N-1}(X) \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{bmatrix}$$

Let this matrix of recurrence coefficients be $\mathbf{G}$. Notice that computing $\mathbf{A}^T\mathbf{b}$ is equivalent to computing the coefficient vector of $\sum \mathbf{b}[i] a_i(X)$. By the equation above, it suffices to compute $\mathbf{b}^T\mathbf{G}^{-1}\mathbf{c}$.

Thus, we convert the main challenge of understanding the structure of the matrix $\mathbf{A}$ into that of understanding the structure of $\mathbf{G}^{-1}$, which is easier to grasp. Notice that $\mathbf{G}$ is triangular and t-banded. The inverse of it is also structured[9] and has the property that every submatrix below the diagonal has rank t. Thus we can partition $\mathbf{G}^{-1}$ into $O(\log N)$ structured submatrices; for any such submatrix $\mathbf{G}'$, we only need to be able to compute $\mathbf{b}'^T\mathbf{G}'\mathbf{c}'$ for vectors $\mathbf{b}', \mathbf{c}'$. We provide a more explicit formula for the entries of $\mathbf{G}^{-1}$ that enables us to do this. The pre-computation step of our algorithms, as mentioned in Theorem 1.3, essentially corresponds to computing a representation of $\mathbf{G}^{-1}$ via generators of its low-rank submatrices. This is formalized in Section 3.3. We note that this algorithm automatically induces an algorithm for $\mathbf{Ab}$ with the same time complexity, by the transposition principle.

Matrix Recurrences, Krylov Efficiency and Displacement Rank. Equation (1.3) can be thought of as equipping the vector space $\mathbb{F}^N$ with a $\mathbb{F}[X]$-module structure and defining a recurrence on vectors $\mathbf{a}_i = \sum g_{i,j}(X)\mathbf{a}_{i-j}$. In general, a $\mathbb{F}[X]$-module structure is defined by the action of X, which can be an arbitrary linear map. This motivates our most general recurrence:

$$\mathbf{a}_i = \sum_{j=1}^{\min(i,t)} g_{i,j}(\mathbf{R})\mathbf{a}_{i-j} + \sum_{k=0}^{r-1} c_{i,k}\mathbf{d}_k, \tag{2.5}$$

where the row vectors are governed by a *matrix recurrence*. This structure naturally captures polynomial recurrence structure (1.3) and (2.4) (when $\mathbf{R}$ is the shift matrix, i.e. 1 on the main subdiagonal), low rank matrices (when the recurrence is degenerate, i.e. $t = 0$), and displacement rank, which we show next.

Consider a matrix $\mathbf{A}$ satisfying equation (1.1) for lower triangular $(t+1)$-banded $\mathbf{L}$ and $\mathbf{R}$ and $\text{rank}(\mathbf{E}) = r$. By the rank condition, each row of $\mathbf{E}$ can be expanded as a linear combination of some r fixed vectors $\mathbf{d}_k$, so the ith row of this equation can be rewritten as $\sum_{j=0}^{t} \mathbf{L}[i, i-j]\mathbf{A}[i-j, :] - \mathbf{A}[i, :]\mathbf{R} = \mathbf{E}[i, :]$, or

$$\begin{aligned} &\mathbf{A}[i, :](\mathbf{L}[i, i]\mathbf{I} - \mathbf{R}) = \\ &\qquad \sum_{j=1}^{t} -\mathbf{L}[i, i-j]\mathbf{A}[i-j, :] + \sum_{k=0}^{r-1} c_{i,k}\mathbf{d}_k. \end{aligned} \tag{2.6}$$

Notice the similarity to equation (2.4) if $\mathbf{A}[i, :]$ is replaced with $a_i(X)$ and $\mathbf{R}$ is X. In fact, we show that

[9] Inverses of banded matrices are called semiseparable [54].

the matrix $\mathbf{A}$ can be decomposed as $\sum_1^r \mathbf{A}_i\mathbf{K}_i$, where $\mathbf{A}_i$ are matrices of recurrence width t and $\mathbf{K}_i$ are *Krylov matrices on* $\mathbf{R}$.[10] We remark that this form generalizes the well-known *ΣLU representation* of Toeplitz-like matrices, which is used in multiplication and inversion algorithms for them [25].

Recurrence (2.5) involves evaluating functions at a matrix $\mathbf{R}$. Classical ways of computing these matrix functions use natural decompositions of the matrix, such as its singular value/eigendecomposition, or Jordan normal form $\mathbf{R} = \mathbf{AJA}^{-1}$. In the full version of the paper we show that it is possible to compute the Jordan decomposition quickly for several special subclasses of the matrices we are interested in, using techniques involving low-width recurrences [16].

However, (2.5) has more structure and only requires multiplication by the aforementioned Krylov matrices. In Section 5.4, we show that the Krylov matrix itself satisfies a recurrence of type (2.4) of width t. Using our established results, this gives a single $O(t^2\mathcal{M}(N)\log N)$ algorithm that unifies the aforementioned subclasses with Jordan decompositions, and implies all triangular banded matrices are *Krylov efficient*.

When $\mathbf{R}$ is not banded, the Krylov matrices do not have low recurrence width, but they can still be structured. The techniques of Section 5.4 show that in general, multiplying by a Krylov matrix on $\mathbf{R}$ is can be reduced to a computation on the *resolvent* of $\mathbf{R}$. Specifically, it is enough to be able to compute the rational function $\mathbf{b}^T(\mathbf{I} - X\mathbf{R})^{-1}\mathbf{c}$ for any $\mathbf{b}, \mathbf{c}$. This reduction bears similarity to results from control theory about the Sylvester equation (1.1) implying that manipulating $\mathbf{A}$ can be done through operating on the resolvents of $\mathbf{L}$ and $\mathbf{R}$ [50]. In the case of multiplication by $\mathbf{A}$, it suffices to be able to solve the above resolvent multiplication problem. In Section 6.1, we show how to solve it when $\mathbf{R}$ is quasiseparable, by using a recursive low-rank decomposition of $\mathbf{I} - X\mathbf{R}$.

3 Problem Definition

3.1 Notation We will use $\mathbb{F}$ to denote a field and use $\mathbb{R}$ and $\mathbb{C}$ to denote the field of real and complex numbers respectively.

Polynomials. For polynomials $p(X), q(X), s(X) \in \mathbb{F}[X]$, we use the notation $p(X) \equiv q(X) \pmod{s(X)}$ to indicate equivalence modulo $s(X)$, i.e. $s(X)|(p(X) - q(X))$, and $p(X) = q(X) \pmod{s(X)}$ to specify $q(X)$ as the unique element of $p(X)$'s equivalence class with degree less than $\deg s(X)$. We use $\mathcal{M}(N)$ to denote the time complexity of multiplying two polynomials of degree N over the field $\mathbb{F}$ (and is known to be $O(N \log N \log\log N)$ in the worst-case). We will use $\tilde{O}(T(N))$ to denote $O\left(T(N) \cdot (\log T(N))^{O(1)}\right)$.

Vectors and Indexing. For any integer $m \geq 1$, we will use $[m]$ to denote the set $\{0, \ldots, m-1\}$. Unless specified otherwise, indices in the paper start from 0. Vectors are boldset like $\mathbf{x}$ and are column vectors unless specified otherwise. We will denote the ith element in $\mathbf{x}$ by $\mathbf{x}[i]$ and the vector between the positions $[\ell, r) : \ell \leq r$ by $\mathbf{x}[\ell : r]$. For any subset $T \subseteq [N]$, $\mathbf{e}_T$ denotes the characteristic vector of T. We will shorten $\mathbf{e}_{\{i\}}$ by $\mathbf{e}_i$. Sometimes a vector $\mathbf{b}$ is associated with a polynomial $b(X)$ and this mapping is always $b(X) = \sum_{i \geq 0} \mathbf{b}[i]X^i$ unless specified otherwise.

Matrices. Matrices will be boldset like $\mathbf{M}$ and by default $\mathbf{M} \in \mathbb{F}^{N \times N}$. We will denote the element in the ith row and jth column by $\mathbf{M}[i, j]$ ($\mathbf{M}[0, 0]$ denotes the 'top-left' element of $\mathbf{M}$). $\mathbf{M}[\ell_1 : r_1, \ell_2 : r_2]$ denotes the sub-matrix $\{\mathbf{M}[i, j]\}_{\ell_1 \leq i < r_1, \ell_2 \leq j < r_2}$. In particular we will use $\mathbf{M}[i, :]$ and $\mathbf{M}[:, j]$ to denote the ith row and jth column of $\mathbf{M}$ respectively. We will use $\mathbf{S}$ to denote the shift matrix (i.e. $\mathbf{S}[i, j] = 1$ if $i = j + 1$ and 0 otherwise) and $\mathbf{I}$ to denote the identity matrix. We let $c_{\mathbf{M}}(X) = \det(X\mathbf{I} - \mathbf{M})$ denote the characteristic polynomial of $\mathbf{M}$. Given a matrix $\mathbf{A}$, we denote its transpose and inverse (assuming it exists) by $\mathbf{A}^T$ (so that $\mathbf{A}^T[i, j] = \mathbf{A}[j, i]$) and $\mathbf{A}^{-1}$ (so that $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{I}$). We will denote $(\mathbf{A}^T)^{-1}$ by $\mathbf{A}^{-T}$.

Finally, we define the notion of Krylov efficiency which will be used throughout and addressed in Section 5.4.

DEFINITION 3.1. *Given a matrix $\mathbf{M} \in \mathbb{F}^{N \times N}$ and a vector $\mathbf{y} \in \mathbb{F}^N$, the* Krylov matrix of $\mathbf{M}$ generated by $\mathbf{y}$ *(denoted by $\mathcal{K}(\mathbf{M}, \mathbf{y})$) is the $N \times N$ matrix whose ith column for $0 \leq i < N$ is $\mathbf{M}^i \cdot \mathbf{y}$. We say that $\mathbf{M}$ is* (α, β)-Krylov efficient *if for every $\mathbf{y} \in \mathbb{F}^N$, we have that $\mathbf{K} = \mathcal{K}(\mathbf{M}, \mathbf{y})$ admits the operations $\mathbf{Kx}$ and $\mathbf{K}^T\mathbf{x}$ (for any $\mathbf{x} \in \mathbb{F}^N$) with $O(\beta)$ many operations (with $O(\alpha)$ pre-processing operations on $\mathbf{M}$).*

3.2 Our Problem We address structured matrices satisfying the following property.

DEFINITION 3.2. *A matrix $\mathbf{A} \in \mathbb{F}^{M \times N}$ satisfies a* $\mathbf{R}$-recurrence *of* width (t, r) *if its row vectors $\mathbf{a}_i = \mathbf{A}[i, :]^T$ satisfy*

$$g_{i,0}(\mathbf{R})\mathbf{a}_i = \sum_{j=1}^{\min(t,i)} g_{i,j}(\mathbf{R})\mathbf{a}_{i-j} + \mathbf{f}_i \tag{3.7}$$

for some polynomials $g_{i,j}(X) \in \mathbb{F}[X]$, and the matrix $\mathbf{F} \in \mathbb{F}^{M \times N}$ formed by stacking the $\mathbf{f}_i$ (as row vectors) has rank r. We sometimes call the $\mathbf{f}_i$ error terms *and $\mathbf{F}$ the* error matrix*, reflecting how they modify the base recurrence.*

[10] The ith column of the Krylov matrix on $\mathbf{R}$ and $\mathbf{x}$ is $\mathbf{R}^i\mathbf{x}$.

Note that $\mathbf{R} \in \mathbb{F}^{N\times N}$ *and we assume* $g_{i,0}(\mathbf{R})$ *is invertible for all* i.

The recurrence has degree $(d, \bar{d})$ *if* $\deg(g_{i,j}) \le dj + \bar{d}$.

For convenience in describing the algorithms, we assume that M, N and t are powers of 2 throughout this paper, since it does not affect the asymptotics.

REMARK 3.1. *We point out some specific cases of importance, and typical assumptions.*

1. *For shorthand, we sometimes say the width is* t *instead of* (t, r) *if* $r \le t$. *In particular, the basic polynomial recurrence* (1.3) *has width* t. *In Section 5.2, we show that every rank* r *up to* t *has the same complexity for our multiplication algorithm.*

2. *When* $\bar{d} = 0$, *we assume that* $g_{i,0} = 1$ *for all* i *and omit it from the recurrence equation.*

3. *The orthogonal polynomial transform in Definition 1.1 has width* $(2, 1)$ *and degree* $(1, 0)$. *More specifically,* $\mathbf{R} = \operatorname{diag}(z_0, \ldots, z_{N-1})$ *and* $\mathbf{F} = \left[\begin{array}{c|c|c|c} \mathbf{e}_1 & \cdots & \mathbf{e}_1 \end{array}\right]$. *In Section 5.3 we show that a Krylov matrix* $\mathcal{K}(\mathbf{R}, \mathbf{1})$ *can be factored out, leaving behind a matrix satisfying a polynomial recurrence* (1.3). *Thus we think of Definition 1.2 as the prototypical example of recurrence width.*

4. *When* $\mathbf{R}$ *is a companion matrix*

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & m_0 \\ 1 & 0 & \cdots & 0 & m_1 \\ 0 & 1 & \cdots & 0 & m_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & m_{N-1} \end{bmatrix}$$

corresponding to the characteristic polynomial $c_{\mathbf{R}}(X) = X^N - m_{N-1}X^{N-1} - \cdots - m_0$, *the recurrence is equivalent to interpreting* $\mathbf{a}_i, \mathbf{f}_i$ *as the coefficient vector of a polynomial and following a polynomial recurrence* $\pmod{c_{\mathbf{R}}(X)}$. *That is, if* $a_i(X) = \sum_{j=0}^{N-1} \mathbf{a}_i[j]X^j = \sum_{j=0}^{N-1} \mathbf{A}[i, j]X^j$ *(and analogously for* $f_i(X)$*), then the* $a_i(X)$ *satisfy the recurrence*

$$g_{i,0}(X)a_i(X) = \sum_{j=1}^{\min(t,i)} g_{i,j}(X)a_{i-j}(X) + f_i(X) \pmod{c_{\mathbf{R}}(X)} \tag{3.8}$$

We sometimes call this a modular recurrence.

As an even more special case, the basic polynomial recurrence (1.3) *can be considered an instance of* (3.7) *with* $\mathbf{R} = \mathbf{S}$, *the shift matrix.*

5. *If* $t = 0$, *then the recurrence is degenerate and* $\mathbf{A} = \mathbf{F}$. *In other words, low rank matrices are degenerate recurrence width matrices.*

A matrix defined by Definition 3.2 can be compactly represented as follows. Let $\mathbf{G} \in \mathbb{F}[X]^{M\times M}$ be given by $\mathbf{G}_{ii} = g_{i,0}(X)$ and $\mathbf{G}_{ij} = -g_{i,i-j}(X)$ (thus, $\mathbf{G}$ is zero outside of the main diagonal and t subdiagonals). Let $\mathbf{F} \in \mathbb{F}^{M\times N}$ be the matrix formed by stacking the $\mathbf{f}_i$ (as row vectors). Then $\mathbf{G}, \mathbf{F}$, and $\mathbf{R}$ fully specify the recurrence and we sometimes refer to $\mathbf{A}$ as a $(\mathbf{G}, \mathbf{F}, \mathbf{R})$-recurrence matrix.

3.3 The Structure Lemma In this section, we provide a characterization of the elements $\mathbf{a}_i$ generated by the recurrence in terms of its parameterization $\mathbf{G}, \mathbf{F}, \mathbf{R}$. We further provide a characterization of the entries of this matrix. We assume for now that we are working with a recurrence of degree $(1, 0)$.

LEMMA 3.1. (STRUCTURE LEMMA, (I)) *Let* $\mathbf{H} = (h_{i,j}(X))_{i,j} = \mathbf{G}^{-1} \pmod{c_{\mathbf{R}}(X)} \in \mathbb{F}[X]^{M\times M}$. *Then*

$$\mathbf{a}_i = \sum_{j=0}^{M-1} h_{i,j}(\mathbf{R})\mathbf{f}_j$$

Proof. Recall that $\mathbf{g}_{i,0}(\mathbf{R})$ is invertible, so $\mathbf{g}_{i,0}(X)$ shares no roots with $c_{\mathbf{R}}(X)$. Thus $\mathbf{G}^{-1} \pmod{c_{\mathbf{R}}(X)}$ is well-defined, since $\mathbf{G}$ is triangular and its diagonal elements are invertible $\pmod{c_{\mathbf{R}}(X)}$.

Given a matrix $\mathbf{M} = (m_{ij}(X))_{i,j}$, let $\mathbf{M}(\mathbf{R})$ denote element-wise evaluation: $\mathbf{M}(\mathbf{R}) = (m_{ij}(\mathbf{R}))$. Note that Definition 3.2 is equivalent to the equation

$$\mathbf{G}(\mathbf{R}) \begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_{M-1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_0 \\ \vdots \\ \mathbf{f}_{M-1} \end{bmatrix}.$$

However, $\mathbf{H}(\mathbf{R})\mathbf{G}(\mathbf{R}) = (\mathbf{H}\mathbf{G})(\mathbf{R}) = \mathbf{I}$ by the Cayley-Hamilton Theorem. This implies

$$\begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_{M-1} \end{bmatrix} = \mathbf{H}(\mathbf{R}) \begin{bmatrix} \mathbf{f}_0 \\ \vdots \\ \mathbf{f}_{M-1} \end{bmatrix}.$$

Our algorithms involve a pre-processing step that computes a compact representation of $\mathbf{H}$. We use the following characterization of the elements of $\mathbf{H}$.

First, for any $0 \le i < M$, we will define the $t \times t$ transition matrix:

$$\mathbf{T}_i = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \\ g_{i+1,t}(X) & g_{i+1,t-1}(X) & \cdots & g_{i+1,1}(X) \end{pmatrix}$$

(let $g_{i,j}(X) = 0$ for $j > i$). And for any $\ell \le r$, define $\mathbf{T}_{[\ell:r]} = \mathbf{T}_{r-1} \times \cdots \times \mathbf{T}_\ell$ (note that $\mathbf{T}_{[\ell:\ell]} = \mathbf{I}_t$).

LEMMA 3.2. (STRUCTURE LEMMA, (II)) *$h_{i,j}(X)$ is the bottom right element of $\mathbf{T}_{[j:i]} \pmod{c_{\mathbf{R}}(X)}$.*

Intuitively, $\mathbf{T}_i$ describes, using the recurrence, how to compute $\mathbf{a}_{i+1}$ in terms of $\mathbf{a}_{i-t+1}, \ldots, \mathbf{a}_i$, and the ranged transition $\mathbf{T}_{[j:i]}$ describes the dependence of $\mathbf{a}_i$ on $\mathbf{a}_{j-t+1}, \ldots, \mathbf{a}_j$. The following lemma about the sizes of entries in $\mathbf{T}_{[\ell:r]}$ will help bound the cost of computing them.

LEMMA 3.3. *Let the recurrence in (3.7) have degree $(1,0)$. Then for any $0 \le \ell \le r < N$ and $0 \le i, j < t$,*

$$\deg(\mathbf{T}_{[\ell:r]}[i,j]) \le \max((r - \ell + i - j), 0).$$

In particular, it is helpful to keep in mind that $\mathbf{H}$ satisfies the same degree condition as $\mathbf{G}$: $\deg(\mathbf{H}[i,j]) \le \max(i-j, 0)$. Statements of Lemmas 3.2 and 3.1 for general $(d, \vec{d})$-degree recurrences, and their proofs (as well as the proof of Lemma 3.3), are presented in the full version of the paper [16].

4 Core Multiplication Algorithm

We provide an algorithm for computing $\mathbf{A}^T\mathbf{b}$ for a simplified setting that contains all the core ideas. Assume $\mathbf{A} \in \mathbb{F}^{N \times N}$ is a modular recurrence of width $(t,1)$ and degree $(1,0)$. In other words, $\mathbf{A}$ is square with three independent simplifications on its generators: $\mathbf{G}$ has degree $(1,0)$, $\mathbf{F}$ has rank 1, and $\mathbf{R}$ is a companion matrix.[11] We can express this as

$$a_i(X) = \sum_{j=1}^{\min(t,i)} g_{i,j}(X) a_{i-j}(X) + \mathbf{c}[i] d(X) \pmod{c_{\mathbf{R}}(X)} \tag{4.9}$$

for some $\mathbf{c} \in \mathbb{F}^N$, $d(X) \in \mathbb{F}[X]$, $\deg(g_{i,j}) \le j$ and recall $c_{\mathbf{R}}(X)$ is the characteristic polynomial of $\mathbf{R}$.

[11] These assumptions are already more general than the setting in Driscoll et al. [18]; their setting corresponds to $t = 2$, $\mathbf{R} = \mathbf{S}$, $\mathbf{c} = \mathbf{e}_0$, $d(X) = 1$ and $c_{\mathbf{R}}(X) = X^N$.

In this context, Lemma 3.1 states that $a_i(X) = \sum_{j=0}^{N-1} h_{i,j}(X) \cdot \mathbf{c}[j] d(X) \pmod{c_{\mathbf{R}}(X)}$. Therefore the desired vector $\mathbf{A}^T\mathbf{b}$ is the coefficient vector of

$$\begin{aligned} \sum_{i=0}^{N-1} \mathbf{b}[i] a_i(X) &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{b}[i] h_{i,j}(X) \cdot \mathbf{c}[j] d(X) \\ &= \mathbf{b}^T \mathbf{H} \mathbf{c} \cdot d(X) \pmod{c_{\mathbf{R}}(X)} \end{aligned} \tag{4.10}$$

So it suffices to compute $\mathbf{b}^T\mathbf{H}\mathbf{c}$, and perform the multiplication by $d(X) \pmod{c_{\mathbf{R}}(X)}$ at the end. By the Lemma 3.2, $\mathbf{b}^T\mathbf{H}\mathbf{c}$ is the bottom right element of the following $t \times t$ matrix:

$$\sum_{0 \le j \le i < N} \mathbf{b}[i] \mathbf{T}_{[j:i]} \mathbf{c}[j] = \sum_{j \le i < \frac{N}{2}} \mathbf{b}[i] \mathbf{T}_{[j:i]} \mathbf{c}[j] + \sum_{i \ge \frac{N}{2}} \mathbf{b}[i] \mathbf{T}_{[\frac{N}{2}:i]} \sum_{j < \frac{N}{2}} \mathbf{T}_{[j:\frac{N}{2}]} \mathbf{c}[j] + \sum_{\frac{N}{2} \le j \le i} \mathbf{b}[i] \mathbf{T}_{[j:i]} \mathbf{c}[j]$$

The first and third sums have the same form as the original. Recursively applying this decomposition, we see that it suffices to compute the last row of $\sum_{i \in [\ell:r]} \mathbf{b}[i] \mathbf{T}_{[\ell:i]}$ and the last column of $\sum_{j \in [\ell:r]} \mathbf{c}[j] \mathbf{T}_{[j:r]}$ for all dyadic intervals $[\ell : r] = \left[\frac{b}{2^d} N : \frac{b+1}{2^d} N\right]$. By symmetry, we can focus only on the former. Denoting this row vector (i.e. $1 \times t$ matrix) $\mathbf{P}_{\ell,r} = \sum_{i \in [\ell:r]} \mathbf{B}_i \mathbf{T}_{[\ell:i]}$ (where $\mathbf{B}_i = \mathbf{b}[i]\mathbf{e}_{t-1}^T = \begin{bmatrix} 0 & \cdots & \mathbf{b}[i] \end{bmatrix}$), it satisfies a simple relation $\mathbf{P}_{\ell,r} = \mathbf{P}_{\ell,m} + \mathbf{P}_{m,r} \mathbf{T}_{[\ell:m]}$ for any $\ell \le m < r$, and thus can be computed with two recursive calls and a matrix-vector multiply over $\mathbb{F}[X]^t$.

Also, note that the $\mathbf{T}_{[\ell:r]}$ are independent of $\mathbf{b}$, so the relevant products will be pre-computed.

Symmetrically, the $t \times 1$ matrices $\mathbf{Q}_{\ell,r} = \sum_{j \in [\ell:r]} \mathbf{T}_{[j:r]} \mathbf{C}_j$ (where $\mathbf{C}_j = \mathbf{c}[j]\mathbf{e}_{t-1}$) satisfy $\mathbf{Q}_{\ell,r} = \mathbf{T}_{[m:r]} \mathbf{Q}_{\ell,m} + \mathbf{Q}_{m,r}$ and can be computed in the same way.

We present the full details in Algorithm 1. The main subroutine $\mathrm{P}(\ell, r)$ computes $\mathbf{P}_{\ell,r}$. Subroutine $\mathrm{H}(\ell, r)$ computes $\sum_{\ell \le j \le i < r} \mathbf{B}[i] \mathbf{T}_{[j:i]} \mathbf{C}[j]$. We omit the procedure Q computing $\mathbf{Q}_{\ell,r} = \sum_{j \in [\ell:r]} \mathbf{T}_{[j:r]} \mathbf{C}_j$, which is identical to procedure P up to a transformation of the indexing.

4.1 Pre-processing time

The pre-processing step is computing a compact representation of $\mathbf{H}$, which we represent through the matrices $\mathbf{T}_{[bN/2^d : bN/2^d + N/2^{d+1}]} \in \mathbb{F}[X]^{t \times t}$ for $0 \le d < m$, $0 \le b < 2^d$. Since we have assumed that N is a power of 2, these ranges can be expressed in terms of dyadic strings; we need to precompute $\mathbf{T}_{[s]}$ for all strings $s \in \{0,1\}^*, |s| \le \lg N$ (where we interpret $[s]$ as the corresponding dyadic interval in $[0, N-1]$). All the required matrices can be computed in a natural bottom-up fashion:

Algorithm 1 TRANSPOSEMULT

Input: $\mathbf{G}, \mathbf{b}, \mathbf{c}, \mathbf{T}_{[\ell:r]}$ for all dyadic intervals: $[\ell : r] = \left[\frac{b}{2^d}N : \frac{b+1}{2^d}N\right]$, $d \in [m], b \in \left[2^d\right]$
Output: $\mathbf{b}^T\mathbf{H}\mathbf{c}$

function $\mathrm{P}(\mathbf{b}, \ell, r)$
  If $r - \ell \le t$ **then**
    $\mathbf{P}_{\ell,r} \leftarrow \sum_{i\in[\ell:r]} \mathbf{B}_i\mathbf{T}_{[\ell:i]}$
  else
    $m \leftarrow (\ell + r)/2$
    $\mathbf{P}_{\ell,r} \leftarrow \mathrm{P}(\mathbf{b}, \ell, m) + \mathrm{P}(\mathbf{b}, m, r)\mathbf{T}_{[\ell:m]}$
    Return $\mathbf{P}_{\ell,r}$
function $\mathrm{H}(\ell, r)$
  If $r - \ell \le t$ **then**
    Return $\sum_{\ell\le j\le i<r} \mathbf{B}[i]\mathbf{T}_{[j:i]}\mathbf{C}[j]$
  else
    $m \leftarrow (\ell + r)/2$
    Return $\mathrm{H}(\ell, m) + \mathbf{P}_{\ell,m}\mathbf{Q}_{m,r} + \mathrm{H}(m, r)$
function TRANSPOSEMULT$(\mathbf{b}, \mathbf{c})$
  $\mathrm{P}(\mathbf{b}, 0, N)$
  $\mathrm{Q}(\mathbf{c}, 0, N)$
  Return $\mathrm{H}(0, N)$

LEMMA 4.1. *We can pre-compute* $\mathbf{T}_{[s]}$ *for all strings* $s \in \{0,1\}^*, |s| \le \lg N$ *with* $O(t^\omega\mathcal{M}(N)\log N)$ *operations.*

Proof. Fix an arbitrary s^* of length $\ell < \lg N$. We can compute $\mathbf{T}_{[s^*]} = \mathbf{T}_{[s^*0]} \cdot \mathbf{T}_{[s^*1]}$. Using the matrix multiplication algorithm, we have $O(t^\omega)$ polynomial multiplications to compute, where the polynomials are of degree at most $\frac{N}{2^\ell} + t$ by Lemma 3.3. So computing $\mathbf{T}_{[s^*]}$ takes $O(t^\omega\mathcal{M}(N/2^\ell + t))$ operations. Thus computing $\mathbf{T}_{[s]}$ for all $|s| = \ell$ is $O(t^\omega\mathcal{M}(N + t2^\ell))$, and computing all $\mathbf{T}_{[s]}$ is $O(t^\omega\mathcal{M}(N)\log N)$, as desired.[12]

COROLLARY 4.1. *Pre-processing for Algorithm 1 requires* $O(t^\omega\mathcal{M}(N)\log N)$ *operations over* $\mathbb{F}$.

4.2 Runtime analysis Assuming that the pre-processing step is already done, the runtime of Algorithm 1 can be bounded with a straightforward recursive analysis (which can be found in the full version of the paper [16]).

LEMMA 4.2. *After pre-processing, Algorithm 1 needs* $O(t^2\mathcal{M}(N)\log N)$ *operations over* $\mathbb{F}$.

We remark that the input vectors $\mathbf{b}$ and the error coefficients $\mathbf{c}$ are duals in a sense: they affect Algorithm 1 symmetrically and independently (through the computations of $\mathbf{P}_{\ell,r}$ and $\mathbf{Q}_{\ell,r}$ respectively). The main difference is that $\mathbf{c}$ affects the pre-processing steps and $\mathbf{b}$ affects the main runtime. We will make use of this observation later in Section 5.2 which generalizes $\mathbf{c}$ from vectors to matrices. The same generalization works for $\mathbf{b}$ and corresponds to matrix-matrix multiplication $\mathbf{A}^T\mathbf{B}$ [16].

Corollary 4.1 and Lemma 4.2 imply the following result:

THEOREM 4.1. *For any matrix* $\mathbf{A}$ *satisfying a modular recurrence* (3.8) *of width* $(t, 1)$ *and degree* $(1, 0)$*, with* $O(t^\omega\mathcal{M}(N)\log N)$ *pre-processing operations, the product* $\mathbf{A}^T\mathbf{b}$ *can be computed for any* $\mathbf{b}$ *with* $O(t^2\mathcal{M}(N)\log N)$ *operations over* $\mathbb{F}$.

5 The General Recurrence and Related Operations

In Section 4, we introduced a multiplication algorithm for matrices $\mathbf{A}$ satisfying a $(\mathbf{G}, \mathbf{F}, \mathbf{R})$-recurrence (see Definition 3.2) with simplified assumptions on $\mathbf{G}, \mathbf{F}, \mathbf{R}$. In this section, we first finish the details of the matrix-vector multiplication algorithm for matrices satisfying the general recurrence. Then we cover several operations that are simple consequences or extensions of the algorithm, such as the case when $\mathbf{A}$ is rectangular.

There are three modifications to recurrence (4.9) needed to recover the general case (3.7). First, we generalize from degree-$(1, 0)$ to degree-$(d, \bar{d})$ recurrences. Second, we generalize the error matrix $\mathbf{F}$ from rank 1 to rank r. Finally, we relax the constraint that $\mathbf{R}$ is a companion matrix.

5.1 General G First, it is easy to see that if $\mathbf{G}$ satisfies a degree $(d, 0)$ recurrence, the companion matrices $\mathbf{T}_i$ can still be defined and the degree bounds in Lemma 3.3 are scaled by d. Since polynomials involved have degrees scaled by d, the cost of Algorithm 1 increases by a factor of d.[13]

Next, consider a matrix $\mathbf{G}$ for a $(d, \bar{d})$-recurrence (i.e. $\mathbf{G}$ lower triangular banded, $\deg(\mathbf{G}[i, j]) \le d(i - j) + \bar{d}$). By multiplying $\mathbf{G}$ on the left and right by suitable diagonal matrices (which depend on the $g_{i,0}(X)$'s), it can be converted into a matrix $\mathbf{G}'$ satisfying $\deg(\mathbf{G}'[i, j]) \le (d + \bar{d})(i - j)$, i.e. corresponding to a $(d + \bar{d}, 0)$-recurrence. The details of this transformation are shown in the full version of the paper [16].

Therefore algorithms for a $(d, \bar{d})$-recurrence have runtimes scaled by a factor of $(d + \bar{d})$. This generaliza-

[12] In the last estimate we note that $|s^*| \le \log_2(t + 1)$, so the $t^{\omega+1}2^\ell$ term only gets added up to $\ell = \log(N/t)$.

[13] The runtime is actually slightly better because the degrees don't grow beyond N (due to the modulus). The $\log N$ factor can be replaced by $\log N - \log d$, but the asymptotics are the same since we usually assume d is fixed.

tion is also independent of the other generalizations - the algorithm does not change, only the operation count. We henceforth assume again that $(d, \bar{d}) = (1, 0)$ for simplicity.

5.2 General F Now consider a modular recurrence of width (t, r). As usual convert the vectors to polynomials; write $\mathbf{F} = \mathbf{CD}$ and let $d_i(X) = \sum_{j=0}^{N-1} \mathbf{D}[i,j]X^j$. By a small modification of the derivation in Section 4, we deduce that the desired quantity $\mathbf{A}^T\mathbf{b}$ is the coefficient vector of the polynomial

$$\mathbf{b}^T\mathbf{HC} \cdot \begin{bmatrix} d_0(X) \\ \vdots \\ d_{r-1}(X) \end{bmatrix} \pmod{c_{\mathbf{R}}(X)}$$

(compare to equation (4.10) in the case $r = 1$). Again the multiplications by $d_0(X), \ldots, d_{r-1}(X)$ can be postponed to the end and incur a cost of $r\mathcal{M}(N)$ operations, so we focus on computing $\mathbf{b}^T\mathbf{HC} \pmod{c_{\mathbf{R}}(X)}$. Recall that $\mathbf{b} \in \mathbb{F}^N, \mathbf{H} \in \mathbb{F}[X]^{N\times N}, \mathbf{C} \in \mathbb{F}^{N\times r}$. We can still apply Algorithm 1 directly. Procedure P still computes $\mathbf{P}_{\ell,s} = \sum_{\ell}^{s-1} \mathbf{B}_i\mathbf{T}_{[\ell:i]}$, but procedure Q now computes $\mathbf{Q}_{\ell,s} = \sum_{\ell}^{s-1} \mathbf{T}_{[i:s]}\mathbf{C}_i$ where

$$\mathbf{C}_i = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \mathbf{C}[i,0] & \cdots & \mathbf{C}[i,s-1] \end{bmatrix},$$

with the only difference being that it now has dimensions $t \times r$ instead of $t \times 1$.

We analyze the change in runtime compared to Lemma 4.2. The call to P does not change; its complexity is still $O(t^2\mathcal{M}(N)\log N)$.

In procedure Q, the change is that the recursive computation $\mathbf{Q}_{\ell,r} = \mathbf{T}_{[m:r]}\mathbf{Q}_{\ell,m} + \mathbf{Q}_{m,r}$ is now a multiplication of a $t \times t$ and $t \times r$ matrix instead of $t \times t$ by $t \times 1$. The runtime coefficient changes from t^2 to $\alpha_{t,r}$, where $\alpha_{t,r}$ is the cost of performing a $t \times t$ by $t \times r$ matrix multiplication. Also note that since this does not depend on $\mathbf{b}$, this can be counted as part of the pre-processing step. The total pre-processing step is now $O\left((t^\omega + \alpha_{t,r})\mathcal{M}(N)\log N\right)$.

In procedure H, we are now performing a $1 \times t$ by $t \times r$ multiplication. The runtime of $\mathrm{H}(0, N)$ is now $O(tr\mathcal{M}(N)\log N)$. The total runtime (the calls to P and H) is $O(t(t+r)\mathcal{M}(N)\log N)$.

Finally, we note that $\alpha_{t,t} = t^\omega$, and in general $\alpha_{t,r} \le \min(rt^2, (1 + r/t)t^\omega)$. For large enough r we use the latter bound, whence the pre-processing coefficient $O(t^\omega + \alpha_{t,r})$ above becomes $O((t+r)t^{\omega-1})$.

In particular, the pre-processing time is still $O(t^\omega\mathcal{M}(N)\log N)$ and the runtime is still $O(t^2\mathcal{M}(N)\log N)$ when $r = O(t)$. This fully captures the original polynomial recurrence (1.3), and in particular the orthogonal polynomial transforms [18].

5.3 General R Aside from the reason that it is necessary to handle the displacement rank recurrence (2.6), we provide an intrinsic reason for considering the general matrix-recurrence (3.7), as a natural continuation of polynomial recurrences (1.3). Equation (1.3) is written in terms of polynomials, but it is defining vectors. Another way of writing it is as follows: we are actually defining a recurrence on vectors $\mathbf{a}_i$ (the rows of $\mathbf{A}$) satisfying $\mathbf{a}_i = \sum g_{ij}(X) \cdot \mathbf{a}_{i-j}$, where the bilinear operator $(\cdot : \mathbb{F}[X] \times \mathbb{F}^N \to \mathbb{F}^N)$ is defined for $g(X)\cdot\mathbf{a}$ by converting $\mathbf{a}$ to a polynomial, multiplying by $g(X) \pmod{c_{\mathbf{R}}(X)}$, and converting back to a vector. This is just an instance of equipping the vector space $\mathbb{F}^N$ with a $\mathbb{F}[X]$-module structure. Thus it is natural to consider what happens in general when we define $\mathbf{a}_i = \sum g_{ij}(X) \cdot \mathbf{a}_{i-j}$ for any $\mathbb{F}[X]$-module structure on $V = \mathbb{F}^N$. In general, this is uniquely defined by the action of X; this is a linear map on V, hence equivalent to multiplication by a matrix $\mathbf{R}$. This leads to the matrix recurrence (3.7).

By Lemma 3.1, $\mathbf{a}_i = \sum_{j=0}^{N-1} h_{i,j}(\mathbf{R})\mathbf{f}_j$, which can be simplified:

$$\begin{aligned}
\mathbf{a}_i &= \sum_{j=0}^{N-1} h_{i,j}(\mathbf{R}) \left(\sum_{k=0}^{r-1} c_{jk}\mathbf{d}_k \right) \\
&= \sum_{j=0}^{N-1} \sum_{k=0}^{r-1} c_{jk} h_{i,j}(\mathbf{R})\mathbf{d}_k \\
&= \sum_{j=0}^{N-1} \sum_{k=0}^{r-1} c_{jk} \left(\sum_{\ell=0}^{N-1} h_{i,j}[\ell]\mathbf{R}^\ell \right) \mathbf{d}_k \\
&= \sum_{j=0}^{N-1} \sum_{k=0}^{r-1} c_{jk} (\mathcal{K}(\mathbf{R}, \mathbf{d}_k)\mathbf{h}_{ij}),
\end{aligned}$$

where $\mathbf{h}_{ij}$ is the coefficient vector of $h_{i,j}(X)$.

Thus the desired answer to $\mathbf{A}^T\mathbf{b}$ is

$$\begin{aligned}
\sum_{i=0}^{N-1} \mathbf{b}[i]\mathbf{a}_i &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{r-1} \mathbf{b}[i] c_{jk} \mathcal{K}(\mathbf{R}, \mathbf{d}_k)\mathbf{h}_{ij} \\
&= \sum_{k=0}^{r-1} \mathcal{K}(\mathbf{R}, \mathbf{d}_k) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{b}[i] c_{jk} \mathbf{h}_{ij} \qquad (5.11)
\end{aligned}$$

Finally, recall that $\mathbf{b}^T\mathbf{HC}$ is a $1 \times r$ vector of polynomials, and $\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{b}[i] c_{jk} \mathbf{h}_{ij}$ is the coefficient array of its kth entry. Thus we compute $\mathbf{b}^T\mathbf{HC}$ as before, and then perform r matrix-vector multiplications by the Krylov matrices $\mathcal{K}(\mathbf{R}, \mathbf{d}_k)$.

With fully general $\mathbf{G}, \mathbf{F}, \mathbf{R}$, we get the following matrix-vector multiplication runtime.

THEOREM 5.1. *If $\mathbf{A} \in \mathbb{F}^{N\times N}$ is a matrix satisfying a $\mathbf{R}$-matrix recurrence* (3.7) *(with known characteristic polynomial $c_{\mathbf{R}}(X)$) of width (t, r) and degree $(d, \bar{d})$, and $\mathbf{R}$ is (α, β)-Krylov efficient, then with $O((d+\bar{d})t^{\omega-1}(t+r)\mathcal{M}(N)\log N + \alpha)$ pre-processing, the products $\mathbf{A}^T\mathbf{b}$ and $\mathbf{Ab}$ for any vector $\mathbf{b}$ can be computed with $O((d+\bar{d})t(t+r)\mathcal{M}(N)\log N + r\beta)$ operations.*

We also note that for the case of rectangular matrices $\mathbf{A} \in \mathbb{F}^{M\times N}$, Algorithm 1 and the modifications in this section still apply. The runtime is easy to analyze; the sizes of the pre-processing and recursion parameters (e.g. degree of polynomials in $\mathbf{T}_{[\ell:r]}$) depend on M, and the sizes of the post-recursion steps (e.g. multiplication by Krylov matrices) depend on N [16].

5.4 Krylov Efficiency In Section 5.3, we showed how to factor matrix recurrences into the product of a polynomial/modular recurrence and a Krylov matrix, thus reducing the runtime of a $\mathbf{R}$-matrix recurrence to the Krylov efficiency (Definition 3.1) of $\mathbf{R}$. In this section we show Krylov efficiency for a particular class of matrices that is necessary to prove Theorem 1.1.

We note that a natural approach to Krylov efficiency is utilizing the Jordan normal form. In the full version of the paper [16], we show how knowing the Jordan form $\mathbf{M} = \mathbf{AJA}^{-1}$ implies a particularly simple algorithm for Krylov efficiency, and provide cases for which we can compute this Jordan decomposition efficiently.

However, this reduction is clearly one way– finding a Jordan decomposition is stronger than Krylov efficiency, but the latter problem has more structure that we can take advantage of. Now we will show that the class of banded triangular matrices are Krylov efficient by showing that the Krylov matrix itself has low recurrence width (equal to the bandwidth).

We remark that the Krylov efficiency concept does not apply only to our problem. If $\mathbf{K}$ is the Krylov matrix on $\mathbf{A}$ and $\mathbf{b}$, then $\mathbf{Kb} = \sum \mathbf{b}[i]\mathbf{A}^i\mathbf{x}$ is naturally related to contexts involving Krylov subspaces, matrix polynomials, and so on. The product $\mathbf{K}^T\mathbf{b} = [\mathbf{b}\cdot\mathbf{x}, \mathbf{b}\cdot\mathbf{Ax}, \mathbf{b}\cdot\mathbf{A}^2\mathbf{x}, \dots]$ is also useful; it is the first step in the Wiedemann algorithm for computing the minimal polynomial or kernel vectors of a matrix $\mathbf{A}$ [28].

5.4.1 Krylov Efficiency of triangular banded matrices Let $\mathbf{M}$ be a lower triangular $(\Delta+1)$-band matrix, i.e. all values other than $\mathbf{M}[i,\ell]$ for $i-\Delta \le \ell \le i$ are zero. Let $\mathbf{y}$ be an arbitrary vector and let $\mathbf{K}$ denote the Krylov matrix of $\mathbf{M}$ with respect to $\mathbf{y}$.

We will show that $\mathbf{K}$ satisfies a modular recurrence of width $(\Delta, 1)$. The same results also hold for upper triangular matrices.

Define polynomials $f_i(X) = \sum_{j=0}^{N-1} \mathbf{K}[i,j]\cdot X^j$ and let $\mathbf{F} = \begin{bmatrix} f_0(X) \\ \vdots \\ f_{N-1}(X) \end{bmatrix}$. We can alternatively express it as

$$\mathbf{F} = \sum_{j=0}^{N-1} \mathbf{K}[:,j]X^j = \sum_{j=0}^{N-1} (\mathbf{M}^j\mathbf{y})X^j = \left(\sum_{j=0}^{N-1} (\mathbf{M}X)^j\right)\mathbf{y}.$$

Multiplying by $\mathbf{I}-\mathbf{M}X$, we get the equation

$$(\mathbf{I}-\mathbf{M}X)\mathbf{F} = \mathbf{y} - (\mathbf{M}X)^N\mathbf{y} \tag{5.12}$$

Therefore it is true that

$$(\mathbf{I}-\mathbf{M}X)\mathbf{F} \equiv \mathbf{y} \pmod{X^N} \tag{5.13}$$

and furthermore, $\mathbf{F}$ can be defined as the unique solution of equation (5.13) because $\mathbf{I}-\mathbf{M}X$ is invertible in $\mathbb{F}[X]/(X^N)$ (since it is triangular and its diagonal is comprised of invertible elements $(1-\mathbf{M}[i,i]X)$).

But equation (5.13) exactly defines a modular recurrence (3.8) of degree $(0,1)$ and width $(\Delta,1)$. Theorem 5.1 implies

THEOREM 5.2. *Any triangular Δ-band matrix is $(\Delta^\omega\mathcal{M}(N)\log N, \Delta^2\mathcal{M}(N)\log N)$-Krylov efficient.*

6 Displacement Rank

Recall that the *displacement rank* of a matrix A with respect to matrices $\mathbf{L}, \mathbf{R}$ is defined as the rank of the *error matrix*

$$\mathbf{E} = \mathbf{LA} - \mathbf{AR}.$$

The concept of displacement rank has been used to generalize and unify common structured matrices such as Hankel, Toeplitz, Vandermonde, and Cauchy matrices; these matrices all have low displacement ranks with respect to diagonal or shift matrices being $\mathbf{L}$ and $\mathbf{R}$. Olshevsky and Shokrollahi [42] defined the confluent Cauchy-like matrices to be the class of matrices with low displacement rank with respect to Jordan form matrices; this class of matrices generalized and unified the previously mentioned common structured matrices. Our class of structured matrices extends the results of [42] to a more general form for $\mathbf{L}$ and $\mathbf{R}$ while matching the complexity bound in their setting.

As usual in the displacement rank approach, we wish to work with a matrix $\mathbf{A}$ defined by a compressed displacement representation. We consider square matrices for simplicity, although the techniques work for rectangular matrices as well [16]. Definition 6.1 formally lays out the representation of $\mathbf{A}$.

DEFINITION 6.1. *Suppose we are given the following:*

- $\mathbf{R} \in \mathbb{F}^{N\times N}$ *that is* (α, β)*-Krylov efficient and such that we know its characteristic polynomial* $c_{\mathbf{R}}(X)$*,*
- $\mathbf{L} \in \mathbb{F}^{N\times N}$ *that is triangular (throughout this section, we will assume it is lower triangular) and* $(\Delta+1)$*-band,*
- $\mathbf{C} \in \mathbb{F}^{N\times r}$ *and* $\mathbf{D} \in \mathbb{F}^{r\times N}$*, generators for a low rank matrix*
- *a displacement operator* $D_{\mathbf{L},\mathbf{R}} \in \{\nabla_{\mathbf{L},\mathbf{R}}, \Delta_{\mathbf{L},\mathbf{R}}\}$*. The Sylvester operator is defined as* $\nabla_{\mathbf{L},\mathbf{R}} : \mathbf{A} \mapsto \mathbf{LA} - \mathbf{AR}$*, and the Stein operator is* $\Delta_{\mathbf{L},\mathbf{R}} : \mathbf{A} \mapsto \mathbf{A} - \mathbf{LAR}$*.*

Assume that $\mathbf{A} \in \mathbb{F}^{N\times N}$ *is implicitly and uniquely defined by the equation* $D_{\mathbf{L},\mathbf{R}}(\mathbf{A}) = \mathbf{CD}$*.*[14]

Suppose that $\mathbf{A}$ is defined according to a Sylvester displacement equation; the Stein displacement case is very similar [16]. We will show that the rows of $\mathbf{A}$ satisfy a standard $\mathbf{R}$-matrix recurrence as in (3.7). Let $\mathbf{d}_0, \ldots, \mathbf{d}_{r-1} \in \mathbb{F}^N$ be the rows of $\mathbf{D}$ (vectorized as a column vector), so that every row of $D_{\mathbf{L},\mathbf{R}}(\mathbf{A})$ (vectorized) can be written as a linear combination of the basis $\mathbf{d}_0, \ldots, \mathbf{d}_{r-1}$.

As in equation (2.6), expanding and rearranging the ith row of $\mathbf{LA} - \mathbf{AR} = \mathbf{E}$ yields

$$\mathbf{A}[i,:](\mathbf{L}[i,i]\mathbf{I}-\mathbf{R}) = \sum_{j=1}^{t} -\mathbf{L}[i,i-j]\mathbf{A}[i-j,:] + \sum_{k=0}^{r-1} c_{i,k}\mathbf{d}_k.$$

By Definition 3.2, this exactly defines a $\mathbf{R}^T$-matrix recurrence of width (t, r) and degree $(0, 1)$. Note that the disjoint eigenvalue assumption means $\mathbf{L}[i,i]\mathbf{I} - \mathbf{R}$ is invertible for all i. In this case $g_{i,0}(X) = \mathbf{L}[i,i] - X$ and $g_{i,j}(X) = -\mathbf{L}[i, i-j]$.

Theorem 5.1 gives the complexity of superfast matrix-vector multiplication by $\mathbf{A}$ and $\mathbf{A}^T$ in terms of the Krylov efficiency of $\mathbf{R}$. Furthermore, when $\mathbf{R}$ is also triangular and Δ-band, its characteristic polynomial can be computed in $O(\mathcal{M}(N)\log N)$ time and it is $(\Delta^\omega \mathcal{M}(N)\log N, \Delta^2\mathcal{M}(N)\log N)$-Krylov efficient by Theorem 5.2.

Again, we stated the reduction for lower triangular matrices, but upper triangular $\mathbf{L}$ defines a similar recurrence. Finally, it is known that $\mathbf{A}^T$ also has low displacement rank, with respect to $\mathbf{R}^T$ and $\mathbf{L}^T$, so the same reduction and algorithm works for $\mathbf{A}^T$.

The above discussion and applying Theorem 5.1 implies

THEOREM 6.1. *Suppose we are given* $\mathbf{L}, \mathbf{R}, \mathbf{C}, \mathbf{D}$ *that define a matrix* $\mathbf{A}$ *according to Definition 6.1. Then we can compute* $\mathbf{Ab}$ *and* $\mathbf{A}^T\mathbf{b}$ *for any vector* $\mathbf{b}$ *in* $O((\Delta + r)\Delta\mathcal{M}(N)\log N + r\beta)$ *operations with* $O(\alpha_{\Delta,r}\mathcal{M}(N)\log N + \alpha)$ *preprocessing.*

COROLLARY 6.1. *Suppose we are given* $\mathbf{L}, \mathbf{R}, \mathbf{C}, \mathbf{D}$ *that define a matrix* $\mathbf{A}$ *according to Definition 6.1, and additionally suppose that* $\mathbf{L}$ *and* $\mathbf{R}$ *are both* Δ*-band. Then we can compute* $\mathbf{Ab}$ *and* $\mathbf{A}^T\mathbf{b}$ *for any vector* $\mathbf{b}$ *in* $O(\Delta^2 r\mathcal{M}(N)\log N)$ *operations with* $O((\Delta^{\omega-1}(\Delta + r))\mathcal{M}(N)\log N)$ *preprocessing.*

This finishes the proof of the first part of Theorem 1.1.

We remark that this captures the previous displacement results in the literature before the very recent results of Bostan et al. [11]. For the four classic types, Toeplitz- and Hankel-like matrices are defined with the Stein operator and $\mathbf{L} = \mathbf{S}$, $\mathbf{R} = \mathbf{S}^T$; Vandermonde-like matrices are defined with the Sylvester operator and $\mathbf{L}$ diagonal, $\mathbf{R} = \mathbf{S}^T$; and Cauchy-like matrices are defined with the Sylvester operator and $\mathbf{L}, \mathbf{R}$ diagonal. Until recently, the most general previous displacement rank results in literature had $\mathbf{L}$ and $\mathbf{R}$ in Jordan normal form, which were handled by Olshovsky and Shokrollahi [42]. The results in this section cover all $\mathbf{L}$ and $\mathbf{R}$ in Jordan normal form that have distinct eigenvalues. We note that it is not possible to have a single efficient algorithm for matrices with low displacement rank with respect to arbitrary $\mathbf{L}, \mathbf{R}$ in Jordan normal form. In particular, every matrix has low displacement rank with respect to $\mathbf{L} = \mathbf{R} = \mathbf{I}$. In general, when $\mathbf{L}$ and $\mathbf{R}$ share eigenvalues, the equation $\mathbf{LA} - \mathbf{AR} = \mathbf{E}$ does not uniquely specify $\mathbf{A}$, and we hypothesize that a general algorithm will incur an extra factor roughly corresponding to the complexity of fully specifying $\mathbf{A}$.

6.1 Quasiseparable L and R We now show how to adapt the above to more general $\mathbf{L}$ and $\mathbf{R}$, which in particular includes both the triangular band matrices of Corollary 6.1 and the block companion matrices of Bostan et al. [11]. For concreteness, we focus on the Sylvester displacement $\mathbf{LA} - \mathbf{AR} = \mathbf{CD}^T$, but the Stein displacement case is similar. Let us trace through the execution of the full algorithm, paying special attention to equation (5.11) which we re-write here for convenience.

$$\mathbf{A}^T\mathbf{b} = \sum_{k=0}^{r-1} \mathcal{K}(\mathbf{R}, \mathbf{d}_k) \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{b}[i]c_{jk}\mathbf{h}_{ij}$$

(where $\mathbf{H}$ is from the Structure Lemma 3.1 and $\mathbf{h}_{ij}$ is the coefficient vector of $\mathbf{H}_{ij} \in \mathbb{F}[X]$).

[14] E.g. In the Sylvester case, the last condition is equivalent to $\mathbf{L}$ and $\mathbf{R}$ not sharing eigenvalues. [50].

The full algorithm for computing $\mathbf{A}^T\mathbf{b}$ can be summarized as follows. Let $\mathbf{H} = (\mathbf{L} - X\mathbf{I})^{-1} \pmod{c_{\mathbf{R}}(X)} \in \mathbb{F}[X]^{N\times N}$. Compute $\mathbf{F} = \mathbf{b}^T\mathbf{H}\mathbf{C} \in \mathbb{F}[X]^{1\times r}$. Let $\mathbf{f}_k \in \mathbb{F}^N$ be the coefficient vector of the kth element of $\mathbf{F}$. The answer is $\sum_{k=0}^{r-1} \mathcal{K}(\mathbf{R}, \mathbf{d}_k)\mathbf{f}_k$. Finally, to perform the Krylov multiplications, recall that in Section 5.4 we showed that $\mathcal{K}(\mathbf{R}, \mathbf{d})^T\mathbf{f}$ is the coefficient vector of the polynomial $\mathbf{f}^T(\mathbf{I} - \mathbf{R}X)^{-1}\mathbf{d} \pmod{X^N}$ (analogous to the $\mathbf{b}^T\mathbf{H}\mathbf{c}$ step of Algorithm 1). The multiplication $\mathcal{K}(\mathbf{R}, \mathbf{d})\mathbf{f}$ has the same complexity by the transposition principle, and an explicit algorithm can be found by using the same techniques to convert the recurrence width transpose multiplication algorithm $\mathbf{A}^T\mathbf{b}$ to the algorithm for $\mathbf{A}\mathbf{b}$ [16].

Thus the multiplication $\mathbf{A}^T\mathbf{b}$ can be reduced to performing $O(r)$ computations of the form $\mathbf{b}^T(X\mathbf{I} - \mathbf{R})^{-1}\mathbf{c} \pmod{M(X)}$ (or $\mathbf{b}^T(\mathbf{I}-\mathbf{R}X)^{-1}\mathbf{c} \pmod{M(X)}$, but these are algorithmically equivalent, so we focus on the former) for some $M(X)$ of degree N (note that $M(X)$ will be equal to either $c_{\mathbf{R}}(X)$ or X^N).[15] It is enough to find $\mathbf{b}^T(X\mathbf{I} - \mathbf{R})^{-1}\mathbf{c}$, which is a rational function of the form $f(X)/g(X)$ - then reducing it $\pmod{M(X)}$ requires $\mathcal{M}(N)\log N$ steps for inverting $g(X) \pmod{M(X)}$ [57] and $\mathcal{M}(N)$ for multiplying by $f(X) \pmod{M(X)}$. We call computing $\mathbf{b}^T(X\mathbf{I}-\mathbf{R})^{-1}\mathbf{c}$ the resolvent problem on $\mathbf{R}$. Henceforth we also let $X - \mathbf{R}$ denote $X\mathbf{I} - \mathbf{R}$.

6.1.1 Resolvent computation The most general useful class of matrices for which we know how to solve the resolvent problem in soft-linear time are the *quasiseparable* matrices, introduced by Eidelman and Gohberg [19].

DEFINITION 6.2. *A matrix $\mathbf{R} \in \mathbb{F}^{N\times N}$ is (p, q)-quasiseparable if*

- *Every submatrix contained strictly below (above) the diagonal has rank at most p (q).*

A (q, q)-quasiseparable matrix is also called q-quasiseparable.[16]

The problem we now address is given t-quasiseparable $\mathbf{R}$, to compute the rational function $\mathbf{b}^T(X - \mathbf{R})^{-1}\mathbf{c}$ for any vectors $\mathbf{b}, \mathbf{c}$.

[15] The above reduction from the Sylvester equation to resolvents is similar to various known formulae for $\mathbf{A}$ based on the Sylvester equation [34, 50].

[16] Given a q-quasiseparable matrix $\mathbf{R}$ satisfying Definition 6.2, we will assume that we have access to a factorization of any rank-q sub-matrix (or can compute one in time equal to the size of this factorization, i.e. $q(k + \ell)$ for a $k \times \ell$ sub-matrix). There are many efficient representations of quasiseparable matrices that allow this [17, 19], and even without one, simple randomized approaches still allow efficient computation of the generators [27].

The idea here is that quasiseparable matrices are recursively "self-similar", in that the leading and trailing principal submatrices are also quasiseparable, which leads to a divide-and-conquer algorithm. Consider a quasiseparable matrix $\mathbf{R}$ for which we want to compute the resolvent $(X - \mathbf{R})^{-1}$. The top left and bottom right blocks of $X - \mathbf{R}$ are self-similar to $X - \mathbf{R}$ itself by definition of quasiseparability. Suppose through recursion we can invert each of them, in other words compute $\mathrm{diag}\{X - \mathbf{R}_{11}, X - \mathbf{R}_{22}\}$. But by quasiseparability, $X - \mathbf{R}$ is simply a low-rank perturbation of $\mathrm{diag}\{X-\mathbf{R}_{11}, X-\mathbf{R}_{22}\}$, and so by standard techniques we can compute $(X - \mathbf{R})^{-1}$ [56]:

PROPOSITION 6.1. (WOODBURY MATRIX IDENTITY) *Over a commutative ring $\mathcal{R}$, let $\mathbf{A} \in \mathcal{R}^{N\times N}$ and $\mathbf{U}, \mathbf{V} \in \mathcal{R}^{N\times p}$. Suppose $\mathbf{A}$ and $\mathbf{A} + \mathbf{U}\mathbf{V}^T$ are invertible. Then $\mathbf{I}_p + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U}$ is invertible and*

$$(\mathbf{A}+\mathbf{U}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{U}(\mathbf{I}_p+\mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1}$$

For our purposes, $\mathcal{R}$ will be the ring of rational functions over $\mathbb{F}$. Now we can prove the following.

LEMMA 6.1. *Let $\mathbf{R}$ be a t-quasiseparable matrix. Then $\mathbf{b}^T(X - \mathbf{R})^{-1}\mathbf{c}$ for any scalar vectors $\mathbf{b}, \mathbf{c}$ can be computed in $O(t^\omega\mathcal{M}(N)\log^2 N + t^2\mathcal{M}(N)\log^3 N)$ operations.*

Proof. More generally, we will consider computing $\mathbf{B}^T(X-\mathbf{R})^{-1}\mathbf{C}$ for matrices $\mathbf{B} \in \mathbb{F}^{N\times k}$ and $\mathbf{C} \in \mathbb{F}^{N\times k}$. Note that the result is a $k \times k$ matrix of rational functions of degree at most N on top and bottom.

Let $\mathbf{R}$ be partitioned into submatrices $\mathbf{R}_{11}, \mathbf{R}_{12}, \mathbf{R}_{21}, \mathbf{R}_{22} \in (\mathbb{F}(X))^{N/2\times N/2}$ in the usual way. Since $\mathbf{R}$ is t-quasiseparable, we can write $\mathbf{R}_{21} = \mathbf{U}_L\mathbf{V}_L^T$ and $\mathbf{R}_{12} = \mathbf{U}_U\mathbf{V}_U^T$ where $\mathbf{U}_\cdot, \mathbf{V}_\cdot \in \mathbb{F}^{N\times t}$. Notice that we can write $X - \mathbf{R}$ as

$$\begin{bmatrix} X - \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & X - \mathbf{R}_{22} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{U}_U \\ \mathbf{U}_L & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_U \end{bmatrix}^T.$$

Suppose we know the expansions of each of

$$\mathbf{M}_1 = \mathbf{B}^T \begin{bmatrix} X - \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & X - \mathbf{R}_{22} \end{bmatrix}^{-1} \mathbf{C}$$

$$\mathbf{M}_2 = \mathbf{B}^T \begin{bmatrix} X - \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & X - \mathbf{R}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} & \mathbf{U}_U \\ \mathbf{U}_L & \mathbf{0} \end{bmatrix}$$

$$\mathbf{M}_3 = \begin{bmatrix} \mathbf{V}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_U \end{bmatrix}^T \begin{bmatrix} X - \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & X - \mathbf{R}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} & \mathbf{U}_U \\ \mathbf{U}_L & \mathbf{0} \end{bmatrix}$$

$$\mathbf{M}_4 = \begin{bmatrix} \mathbf{V}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_U \end{bmatrix}^T \begin{bmatrix} X - \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & X - \mathbf{R}_{22} \end{bmatrix}^{-1} \mathbf{C}.$$

These have dimensions $k \times k, k \times 2t, 2t \times 2t, 2t \times k$ respectively (and entries bounded by degree $N/2$ on the

top and bottom). Also note that all the above inverses exist because a matrix of the form $\mathbf{X}-\mathbf{R}$ for $\mathbf{R}\in\mathbb{F}$ has non-zero determinant.

By Proposition 6.1, the desired answer is

$$\mathbf{B}^T(X-\mathbf{R})^{-1}\mathbf{C}=\mathbf{M}_1-\mathbf{M}_2(\mathbf{I}_{2t}+\mathbf{M}_3)^{-1}\mathbf{M}_4.$$

Then the final result can be computed by inverting $\mathbf{I}_{2t}+\mathbf{M}_3$ ($O(t^\omega\mathcal{M}(N))$ operations), multiplying by $\mathbf{M}_2,\mathbf{M}_4$ ($O(k/t\cdot t^\omega\mathcal{M}(N))$ operations each[17]), and subtracting from $\mathbf{M}_1$ ($O(k^2\mathcal{M}(N))$ operations). This is a total of $O((t^\omega+kt^{\omega-1}+k^2)\mathcal{M}(N))$ operations. Note that when $k=O(t\log N)$, this becomes $O(t^\omega\mathcal{M}(N)\log N+t^2\mathcal{M}(N)\log^2 N)$; we will use this in the analysis shortly.

To compute the $\mathbf{M}_i$, it suffices to compute:

$$\begin{aligned}
&\mathbf{B}_1^T(X-\mathbf{R}_{11})^{-1}\mathbf{C}_1 && \mathbf{B}_2^T(X-\mathbf{R}_{22})^{-1}\mathbf{C}_2\\
&\mathbf{B}_1^T(X-\mathbf{R}_{11})^{-1}\mathbf{U}_U && \mathbf{B}_2^T(X-\mathbf{R}_{22})^{-1}\mathbf{U}_L\\
&\mathbf{V}_L^T(X-\mathbf{R}_{11})^{-1}\mathbf{U}_U && \mathbf{V}_U^T(X-\mathbf{R}_{22})^{-1}\mathbf{U}_L\\
&\mathbf{V}_L^T(X-\mathbf{R}_{11})^{-1}\mathbf{C}_1 && \mathbf{V}_U^T(X-\mathbf{R}_{22})^{-1}\mathbf{C}_2.
\end{aligned}$$

But to compute those, it suffices to compute the following $(k+t)\times(k+t)$ matrices:

$$\begin{bmatrix}\mathbf{B}_1 & \mathbf{V}_L\end{bmatrix}^T(X-\mathbf{R}_{11})^{-1}\begin{bmatrix}\mathbf{C}_1 & \mathbf{U}_U\end{bmatrix}$$
$$\begin{bmatrix}\mathbf{B}_2 & \mathbf{V}_U\end{bmatrix}^T(X-\mathbf{R}_{22})^{-1}\begin{bmatrix}\mathbf{C}_2 & \mathbf{U}_L\end{bmatrix}$$

Since $\mathbf{R}_{11}$ and $\mathbf{R}_{22}$ have the same form as $\mathbf{R}$, this is two recursive calls of half the size. Notice that the size of the other input (dimensions of $\mathbf{B},\mathbf{C}$) is growing, but when the initial input is $k=1$, it never exceeds $1+t\log N$ (since they increase by t every time we go down a level). Earlier, we noticed that when $k=O(t\log N)$, the reduction step has complexity $O(t^\omega\mathcal{M}(N)\log N+t^2\mathcal{M}(N)\log^2 N)$ for any recursive call. As usual, the complete runtime is a $\log N$ multiplicative factor on top of this.

Combining the aforementioned reduction from the Sylvester equation to resolvents with this algorithm proves the multiplication part of Theorem 1.2.

We note that the bounds in Lemma 6.1 are slightly worse in the exponent of t and the number of $\log N$ factors, compared to the bounds derived from the recurrence width algorithm as in Corollary 6.1. A more detailed analysis that isolates operations independent of $\mathbf{b}$ as pre-computations should be possible to bridge the gap between these bounds, and is left for future work.

[17] When $k<t$ this term is subsumed by the previous one anyways

7 Succinct Representations and Multivariate Polynomials

The goal of this section is two fold. The first goal is to present matrices that have low recurrence width in our sense but were not captured by previous notions of widths of structured matrices. The second goal is to show that substantially improving upon the efficiency of our algorithms with respect to sharper notions of input size will lead to improvements in the state-of-the-art algorithms for multipoint evaluation of multivariate polynomials. Our initial interest in these matrices arose from their connections to coding theory, which we will also highlight as we deal with the corresponding matrices.

We consider the following problem.

Definition 7.1. *Given an m-variate polynomial $f(X_1,\dots,X_m)$ such that each variable has degree at most $d-1$ and $N=d^m$ distinct points $\mathbf{x}(i)=(x(i)_1,\dots,x(i)_m)$ for $1\le i\le N$, output the vector $(f(\mathbf{x}(i)))_{i=1}^N$.*

The best runtime for an algorithm that solves the above problem (over an arbitrary field) takes time $O(d^{\omega_2(m-1)/2+1})$ [16], where an $n\times n$ matrix can be multiplied with an $n\times n^2$ matrix with $O(n^{\omega_2})$ operations [29, 38]. We remark on three points. First in the multipoint evaluation problem we do not assume any structure on the N points: e.g. if the points form an m-dimensional grid, then the problem can be solved in $\tilde{O}(N)$ many operations using standard FFT techniques. Second, if we are fine with solving the problem over *finite* fields, then the breakthrough result of Kedlaya and Umans [29] solves this problem with $N^{1+o(1)}$ operations (but for arbitrary N evaluation points). In other words, the problem is not fully solved only if we do not have any structure in the evaluation points and we want our algorithms to work over arbitrary fields (or even $\mathbb{R}$ or $\mathbb{C}$). Finally, from a coding theory perspective, this problem (over finite fields) corresponds to encoding of arbitrary puncturings of Reed-Muller codes.

While we do not prove any new upper bounds for these problems, it turns out that the connection to multipoint evaluation of multivariate polynomials has some interesting complexity implications for our result. In particular, recall that the worst-case input size of a matrix with recurrence width t is $\Theta(t^2N)$ and our algorithms are optimal with respect to this input measure (assuming $\omega=2$). However, it is natural to wonder if one can have faster algorithms with respect to a more per-instance input size.

Next, we aim to show that if we can improve our algorithms in certain settings then it would imply a fast multipoint evaluation of multivariate polynomials. In

particular, we consider the following two more succinct ways of representing the input. For a given polynomial $f(X) \in \mathbb{F}[X]$, let $\|f\|_0$ denote the size of the support of f. Finally, consider a matrix $\mathbf{A}$ defined by a recurrence in (3.7). Define

$$\|\mathbf{A}\|_0 = \sum_{i=0}^{N-1} \sum_{j=0}^{t} \|g_{i,j}\|_0 + rN,$$

i.e. the size of sum of the sizes of supports of $g_{i,j}$'s plus the size of the rank r-representation of the error matrix in (3.7).

The second more succinct representation where we have an extra bound that $\|g_{i,j}\| \leq D$ (for potentially $D < t$) for the recurrence in (3.7). Then note that the corresponding matrix $\mathbf{A}$ can be represented with size $\Theta(tDN + rN)$ elements. In this case, we will explore if one can improve upon the dependence on r in Theorem 5.1.

We would like to point out that in all of the above the way we argue that the error matrix $\mathbf{E}$ has rank at most r is by showing it has at most r non-zero columns. Thus, for our reductions rN is also an upper bound on $\|\mathbf{E}\|_0$, so there is no hope of getting improved results in terms of the sparsity of the error matrix instead of its rank without improving upon the state-of-the-art results in multipoint evaluation of multivariate polynomials.

7.1 Multipoint evaluation of bivariate polynomials We begin with the bivariate case (i.e. $m = 2$) since that is enough to connect improvements over our results to improving the state-of-the-art results in multipoint evaluation of bivariate polynomials.

For notational simplicity we assume that the polynomial is $f(X, Y) = \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} f_{i,j} X^i Y^j$ and the evaluation points are $(x_1, y_1), \ldots, (x_N, y_N)$. Now consider the $N \times N$ matrix in (7.14).

Note that to solve the multipoint evaluation problem we just need to solve $\mathbf{A}^{(2)} \cdot \mathbf{f}$, where $\mathbf{f}$ contains the coefficients of $f(X, Y)$. Let $\mathbf{D}_X$ and $\mathbf{D}_Y$ denote the diagonal matrices with $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{y} = (y_1, \ldots, y_N)$ on their diagonals respectively. Finally, define $\mathbf{Z} = \mathbf{S}^T$. Now consider the matrix

$$\mathbf{B}^{(2)} = \mathbf{D}_X^{-1} \mathbf{A}^{(2)} - \mathbf{A}^{(2)} \mathbf{Z}.$$

It can be checked that $\mathbf{B}^{(2)}$ has rank at most d. Indeed note that $\mathbf{B}^{(2)}$ has the structure as in (7.15).

The above was already noticed in [41]. The above is not quite enough to argue what we want so we make the following stronger observation. Consider

$$\begin{aligned} \mathbf{C}^{(2)} &= \mathbf{D}_Y^{-1} \mathbf{B}^{(2)} - \mathbf{B}^{(2)} \mathbf{Z}^d \\ &= \mathbf{D}_Y^{-1} \mathbf{D}_X^{-1} \mathbf{A}^{(2)} - \mathbf{D}_Y^{-1} \mathbf{A}^{(2)} \mathbf{Z} \\ &\quad - \mathbf{D}_X^{-1} \mathbf{A}^{(2)} \mathbf{Z}^d + \mathbf{A}^{(2)} \mathbf{Z}^{d+1}. \end{aligned} \tag{7.16}$$

One can re-write the above recurrence as follows (where $\mathbf{a}_i = \left(\mathbf{A}^{(2)}[i, :]\right)^T$ and recall $\mathbf{Z} = \mathbf{S}^T$) for any $0 \leq i < N$:

$$\left(\frac{1}{x_i y_i} - \frac{\mathbf{S}}{y_i} - \frac{\mathbf{S}^d}{x_i} + \mathbf{S}^{d+1} \right) \cdot \mathbf{a}_i = \left(\mathbf{C}^{(2)}[i, :] \right)^T.$$

We now claim that the rank of $\mathbf{C}^{(2)}$ is at most two. Indeed, note that $\mathbf{C}^{(2)}$ has structure as in (7.17).

Thus, we have a recurrence with recurrence width $(1, 2)$ and degree $(D, 1)$. Theorem 5.1 implies that we can solve the above problem with $\tilde{O}(d^3)$ operations. The algorithm of [38] uses $\tilde{O}(d^{\omega_2/2+1})$ many operations. However, note that

$$\|\mathbf{A}^{(2)}\|_0 = \Theta(d^2).$$

Thus, we have the following result:

THEOREM 7.1. *If one can solve* $\mathbf{Ab}$ *for any* $\mathbf{b}$ *with* $\tilde{O}\left((\|\mathbf{A}\|_0)^{\omega_2/4+1/2-\epsilon} \right)$ *operations, then one will have an multipoint evaluation of bivariate polynomials with* $\tilde{O}(d^{\omega_2/2+1-2\epsilon})$ *operations, which would improve upon the currently best-known algorithm for the latter.*

7.2 Multipoint evaluation of multivariate polynomials We now consider the general multivariate polynomial case. Note that we can represent the multipoint evaluation of the m-variate polynomial $f(X_1, \ldots, X_m)$ as $\mathbf{A}^{(m)} \mathbf{f}$, where $\mathbf{f}$ is the vector of coefficients and $\mathbf{A}^{(m)}$ is presented as follows.

Each of the d^m columns are indexed by tuples $\mathbf{i} \in \mathbb{Z}_d^m$ and the columns are sorted in lexicographic increasing order of the indices. The column $\mathbf{i} = (i_1, \ldots, i_m) \in \mathbb{Z}_d^m$ is represented by

$$\mathbf{A}^{(m)}[:, \mathbf{i}] = \begin{pmatrix} \prod_{j=1}^m x(1)_j^{i_j} \\ \prod_{j=1}^m x(2)_j^{i_j} \\ \vdots \\ \prod_{j=1}^m x(N)_j^{i_j} \end{pmatrix},$$

where the evaluation points are given by $\mathbf{x}(1), \ldots, \mathbf{x}(N)$.

For notational simplicity, we will assume that m is even. (The arguments below can be easily modified for odd m.) Define recursively for $0 \leq j \leq m/2$:

$$\mathbf{B}^{(j)} = \mathbf{D}_{X_{m-j}}^{-1} \mathbf{B}^{(j+1)} - \mathbf{B}^{(j+1)} \mathbf{Z}^{d^j}, \tag{7.18}$$

(7.14)
$$\mathbf{A}^{(2)} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{d-1} & y_1 & y_1x_1 & \cdots & y_1x_1^{d-1} & y_1^2 & \cdots & y_1^2x_1^{d-1} & \cdots & y_1^{d-1} & \cdots & y_1^{d-1}x_1^{d-1} \\ 1 & x_2 & \cdots & x_2^{d-1} & y_2 & y_2x_2 & \cdots & y_2x_2^{d-1} & y_2^2 & \cdots & y_2^2x_2^{d-1} & \cdots & y_2^{d-1} & \cdots & y_2^{d-1}x_2^{d-1} \\ & & & & & & \vdots & & & & & \vdots & & & \\ 1 & x_N & \cdots & x_N^{d-1} & y_N & y_Nx_N & \cdots & y_Nx_N^{d-1} & y_N^2 & \cdots & y_N^2x_N^{d-1} & \cdots & y_N^{d-1} & \cdots & y_N^{d-1}x_N^{d-1} \end{pmatrix}.$$

(7.15)
$$\mathbf{B}^{(2)} = \begin{pmatrix} \frac{1}{x_1} & 0\cdots0 & \frac{y_1}{x_1} - x_1^{d-1} & 0\cdots0 & y_1\left(\frac{y_1}{x_1} - x_1^{d-1}\right) & 0\cdots0 & \cdots & y_1^{d-2}\left(\frac{y_1}{x_1} - x_1^{d-1}\right) & 0\cdots0 \\ \frac{1}{x_2} & 0\cdots0 & \frac{y_2}{x_2} - x_2^{d-1} & 0\cdots0 & y_2\left(\frac{y_2}{x_2} - x_2^{d-1}\right) & 0\cdots0 & \cdots & y_2^{d-2}\left(\frac{y_2}{x_2} - x_2^{d-1}\right) & 0\cdots0 \\ & & & \vdots & & & & \vdots & \\ \frac{1}{x_N} & 0\cdots0 & \frac{y_N}{x_N} - x_N^{d-1} & 0\cdots0 & y_N\left(\frac{y_N}{x_N} - x_N^{d-1}\right) & 0\cdots0 & \cdots & y_N^{d-2}\left(\frac{y_N}{x_N} - x_N^{d-1}\right) & 0\cdots0 \end{pmatrix}.$$

(7.17)
$$\mathbf{C}^{(2)} = \begin{pmatrix} \frac{1}{x_1y_1} & 0 & \cdots & 0 & \frac{-x_1^{d-1}}{y_1} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \frac{1}{x_2y_2} & 0 & \cdots & 0 & \frac{-x_2^{d-1}}{y_2} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & & & & & & \vdots & & & & & & \vdots & & & & \\ \frac{1}{x_Ny_N} & 0 & \cdots & 0 & \frac{-x_N^{d-1}}{y_N} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

where $\mathbf{D}_{X_k}$ is the diagonal matrix with $(x(1)_k, \ldots, x(N)_k)$ on its diagonal. Finally, for the base case we have

$$\mathbf{B}^{(\frac{m}{2}+1)} = \mathbf{A}^{(m)}.$$

It can be verified (e.g. by induction) that the recurrence in (7.18) can be expanded out to

$$\mathbf{B}^{(0)} = \sum_{S \subseteq [m/2,m]} (-1)^{m/2+1-|S|} \left(\prod_{j \in S} \mathbf{D}_{X_j}^{-1}\right) \mathbf{A}^{(m)} \left(\prod_{j \in [m/2,m]\setminus S} \mathbf{Z}^{d^{m-j}}\right). \tag{7.19}$$

The above can be re-written as (where $\mathbf{f}_i = \left(\mathbf{A}^{(m)}[i,:]\right)^T$):

$$\left(\mathbf{B}^{(0)}[i,:]\right)^T = \left(\sum_{S \subseteq [m/2,m]} (-1)^{m/2+1-|S|} \cdot \frac{1}{\prod_{j \in S} x(i)_j} \left(\prod_{j \in [m/2,m]\setminus S} \mathbf{S}^{d^{m-j}}\right)\right) \cdot \mathbf{f}_i.$$

We argue in the full version of the paper [16] that

LEMMA 7.1. *$\mathbf{B}^{(0)}$ has rank at most $2^{m/2} \cdot d^{m/2-1}$.*

Note that the above lemma implies that the recurrence in (7.19) is a **S**-matrix recurrence with recurrence width $(1, r = 2^{m/2}d^{m/2-1})$ and degree $(D = \frac{d^{1+m/2}-1}{d-1}, 1)$. Note that in this case we have $tDN + rN = \Theta((2d)^{3m/2-1})$. Thus, we have the following result:

THEOREM 7.2. *If for an **S**-dependent recurrence we could improve the algorithm from Theorem 5.1 to run with $\tilde{O}(\text{poly}(t) \cdot DN + rN)$ operations for matrix vector multiplication, then we would be able to solve the general multipoint evaluation of multivariate polynomials in time $\tilde{O}((2d)^{3m/2-1})$, which would be a polynomial improvement over the current best algorithm (when $d = \omega(1)$), where currently we still have $\omega_2 > 3$.*

Note that the above shows that improving the dependence in r in Theorem 5.1 significantly (even to the extent of having some dependence on Dr) will improve upon the current best-known algorithms (unless $\omega_2 = 3$).

Finally, in the full version of the paper [16], we present one more example of matrices that have been studied in coding theory that satisfy our general notion of recurrence. These matrices encode multipoint evaluation of multivariate polynomials and their derivatives and correspond to (puncturing of) multivariate multiplicity codes, which have been studied recently [31–33].

However, currently this does not yield any conditional "lower bounds" along the lines of Theorem 7.1 or 7.2.

8 Related Work

Superfast structured matrix-vector multiplication has been a rich area of research. Popular classes of structured matrices such as Toeplitz, Hankel, and Vandermonde matrices and their inverses all have classical superfast multiplication algorithms that correspond to operations on polynomials [6, 8]. The four classes of matrices are all subsumed by the notion of displacement rank introduced by Kailath, Kung, and Morf in 1979 [25] (also see [21] and [23]). Kailath et al. initially used displacement rank to define Toeplitz-like matrices, generalizing Toeplitz matrices. Then Morf [37] and Bitmead and Anderson [9] developed a fast divide-and-conquer approach for solving a Toeplitz-like linear system of equations in quasilinear time, now known as the MBA algorithm. This was extended to Hankel and Vandermonde-like matrices in [43]. In 1994, Gohberg and Olshevsky further used displacement rank to define Vandermonde-like, Hankel-like, and Cauchy-like matrices and developed superfast algorithms for vector multiplication [22]. In 1998, Olshevsky and Pan extended the MBA algorithm to superfast inversion, in a unified way, of these four main classes of displacement structured matrices [40]. These matrix classes were unified and generalized by Olshevsky and Shokrollahi in 2000 [42] with a class of matrices they named confluent Cauchy-like, which have low displacement rank with respect to Jordan form matrices. In this work, we extend these results by investigating matrices with low displacement rank with respect to any triangular t-band matrices, which we define to be matrices whose non-zero elements all appear in t consecutive diagonals. General and unified algorithms for the most popular classes of matrices with displacement structure have continued to be refined for practicality and precision, such as in [24] and [46].

A second strand of research that inspired our work is the study of orthogonal polynomial transforms, especially that of Driscoll, Healy, and Rockmore [18]. Orthogonal polynomials are widely used and well worth studying in their own right: for an introduction to the area, see the classic book of Chihara [15]. We present applications for some specific orthogonal polynomials. Chebyshev polynomials are used for numerical stability (see e.g. the ChebFun package [5]) as well as approximation theory (see e.g. Chebyshev approximation [1]). Jacobi polynomials form solutions of certain differential equations [2]. Zernike polynomials have applications in optics and physics [3]. In fact, our investigation into structured matrix-vector multiplication problems started with some applied work on Zernike polynomials, and our results applied to fast Zernike transforms have been used in improved cancer imaging [58]. Driscoll et al. rely heavily on the three-term recurrence satisfied by orthogonal polynomials to devise a divide-and-conquer algorithm for computing matrix-vector multiplication. One main result of this work is a direct generalization of the recurrence, and we rely heavily on the recurrence to formulate our own divide and conquer algorithm. The basic algorithm in Section 4 is modeled after previous works on orthogonal polynomials [12,18,49], which are themselves reminiscent of classic recursive doubling techniques such as that proposed by Kogge and Stone [30].

A third significant strand of research is the study of rank-structured matrices. The most well-known example is the class of semiseparable matrices, for which we refer to an extensive survey by Vandebril, Van Barel, Golub, and Mastronardi [53] for a detailed overview of the body of work. Many variants and generalizations exist including the generator representable semiseparable matrices, sequentially separable mentions, and quasiseparable matrices, which all share the defining feature of certain sub-matrices (such as those contained above or below the diagonal) being low rank [20, 55]. These types of matrices turn out to be highly useful for devising fast practical solutions of certain structured systems of equations. Quasiseparable matrices in particular, which appear in our Theorem 1.2, are actively being researched with recent fast representations and algorithms in [47, 48]. Just as how the four main displacement structures are closely tied to polynomial operations [44], the work of Bella, Eidelman, Gohberg, and Olshevsky show deep connections between computations with rank-structured matrices and with polynomials [7]. Indeed at this point connections between structured matrices and polynomials is well established [8, 43, 44].

As discussed in Section 1, we view the connection of these many strands of work as our strongest conceptual contribution.

Acknowledgments

We thank the anonymous STOC 17, FOCS 17 and SODA 18 reviewers on drafts of this paper for their helpful comments including pointing us to the transposition principle and other existing literature. We would like to thank Swastik Kopparty, Chris Umans, and Mary Wootters for helpful discussions. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) SIMPLEX program under No. N66001-15-C-4043, the DARPA D3M program under No. FA8750-17-2-0095, the National Science Foun-

dation (NSF) CAREER Award under No. IIS-1353606, the Office of Naval Research (ONR) under awards No. N000141210041 and No. N000141310129, a Sloan Research Fellowship, the Moore Foundation, an Okawa Research Grant, Toshiba, and Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, NSF, ONR, or the U.S. government. Atri Rudra's research is supported by NSF grant CCF-1319402 and CCF-1717134.

References

[1] https://en.wikipedia.org/wiki/Approximation_theory#Chebyshev_approximation.

[2] https://en.wikipedia.org/wiki/Jacobi_polynomials#Differential_equation.

[3] https://en.wikipedia.org/wiki/Zernike_polynomials.

[4] http://wis.kuleuven.be/events/OPSFA/.

[5] http://www.chebfun.org.

[6] Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[7] Bella, T., Eidelman, Y., Gohberg, I., and Olshevsky, V. Computations with quasiseparable polynomials and matrices. *Theoretical Computer Science 409*, 2 (2008), 158 – 179. Symbolic-Numerical Computations.

[8] Bini, D., and Pan, V. Y. *Polynomial and Matrix Computations (Vol. 1): Fundamental Algorithms*. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1994.

[9] Bitmead, R. R., and Anderson, B. D. Asymptotically fast solution of Toeplitz and related systems of linear equations. *Linear Algebra and its Applications 34* (1980), 103–116.

[10] Bordewijk, J. L. Inter-reciprocity applied to electrical networks. *Applied Scientific Research B: Electrophysics, Acoustics, Optics, Mathematical Methods 6* (1956), 1–74. URL: http://cr.yp.to/bib/entries.html#1956/bordewijk.

[11] Bostan, A., Jeannerod, C.-P., Mouilleron, C., and Schost, É. On matrices with displacement structure: generalized operators and faster algorithms. *arXiv preprint arXiv:1703.03734* (2017).

[12] Bostan, A., Salvy, B., and Schost, É. Fast conversion algorithms for orthogonal polynomials. *Linear Algebra and its Applications 432*, 1 (2010), 249–258.

[13] Boyar, J., Matthews, P., and Peralta, R. Logic minimization techniques with applications to cryptology. *Journal of Cryptology 26*, 2 (2013), 280–312.

[14] Bürgisser, P., Clausen, M., and Shokrollahi, A. *Algebraic complexity theory*, vol. 315. Springer Science & Business Media, 2013.

[15] Chihara, T. *An Introduction to Orthogonal Polynomials*. Dover Books on Mathematics. Dover Publications, 2011.

[16] De Sa, C., Gu, A., Puttagunta, R., Ré, C., and Rudra, A. A two-pronged progress in structured dense matrix vector multiplication. *arXiv preprint arXiv:1611.01569* (2017).

[17] Delvaux, S., and Van Barel, M. A Givens-weight representation for rank structured matrices. *SIAM Journal on Matrix Analysis and Applications 29*, 4 (2007), 1147–1170.

[18] Driscoll, J. R., Healy, Jr., D. M., and Rockmore, D. N. Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs. *SIAM J. Comput. 26*, 4 (Aug. 1997), 1066–1099.

[19] Eidelman, Y., and Gohberg, I. On a new class of structured matrices. *Integral Equations and Operator Theory 34*, 3 (1999), 293–324.

[20] Eidelman, Y., Gohberg, I., and Haimovici, I. *Separable Type Representations of Matrices and Fast Algorithms: Volume 1 Basics. Completion Problems. Multiplication and Inversion Algorithms*, vol. 234. Springer Science & Business Media, 2013.

[21] Friedlander, B., Morf, M., Kailath, T., and Ljung, L. New inversion formulas for matrices classified in terms of their distance from toeplitz matrices. *Linear Algebra and its Applications 27* (1979), 31 – 60.

[22] Gohberg, I., and Olshevsky, V. Complexity of multiplication with vectors for structured matrices. *Linear Algebra and its Applications 202* (1994), 163 – 192.

[23] Heinig, G., and Rost, K. *Algebraic methods for Toeplitz-like matrices and operators*. Mathematical research. Akademie-Verlag, 1984.

[24] Hyun, S. G., Lebreton, R., and Schost, É. Algorithms for structured linear systems solving and their implementation. Draft at https://hal-lirmm.ccsd.cnrs.fr/lirmm-01484831, Jan. 2017.

[25] Kailath, T., Kung, S.-Y., and Morf, M. Displacement ranks of matrices and linear equations. *Journal of Mathematical Analysis and Applications 68*, 2 (1979), 395 – 407.

[26] Kailath, T., and Sayed, A. H. Displacement structure: Theory and applications. *SIAM Review 37*, 3 (1995), 297–386.

[27] Kaltofen, E. Asymptotically fast solution of Toeplitz-like singular linear systems. In *Proceedings of the international symposium on Symbolic and algebraic computation* (1994), ACM, pp. 297–304.

[28] Kaltofen, E., and Saunders, B. D. On Wiedemann's method of solving sparse linear systems. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes* (1991), Springer Berlin Heidelberg, pp. 29–38.

[29] Kedlaya, K. S., and Umans, C. Fast polynomial factorization and modular composition. *SIAM J. Comput. 40*, 6 (2011), 1767–1802.

[30] Kogge, P. M., and Stone, H. S. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE transactions on computers 100*, 8 (1973), 786–793.

[31] Kopparty, S. List-decoding multiplicity codes. *Theory of Computing 11* (2015), 149–182.

[32] Kopparty, S., Meir, O., Ron-Zewi, N., and Saraf, S. High rate locally-correctable and locally-testable codes with sub-polynomial query complexity. *CoRR abs/1504.05653* (2015).

[33] Kopparty, S., Saraf, S., and Yekhanin, S. High-rate codes with sublinear-time decoding. *J. ACM 61*, 5 (2014), 28:1–28:20.

[34] Lancaster, P., Lerer, L., and Tismenetsky, M. Factored forms for solutions of AX- XB= C and X-AXB= C in companion matrices. *Linear Algebra and Its Applications 62* (1984), 19–49.

[35] Lupanov, O. A method of circuit synthesis. *Izv. VUZ (Radiofizika) 1* (1958), 120–140.

[36] Moczulski, M., Denil, M., Appleyard, J., and de Freitas, N. ACDC: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946* (2015).

[37] Morf, M. Doubling algorithms for Toeplitz and related equations. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'80.* (1980), vol. 5, IEEE, pp. 954–959.

[38] Nüsken, M., and Ziegler, M. Fast multipoint evaluation of bivariate polynomials. In *Algorithms - ESA 2004, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings* (2004), pp. 544–555.

[39] Olshevsky, V. *Fast Algorithms for Structured Matrices: Theory and Applications: AMS-IMS-SIAM Joint Summer Research Conference on Fast Algorithms in Mathematics, Computer Science, and Engineering, August 5-9, 2001, Mount Holyoke College, South Hadley, Massachusetts*, vol. 323. American Mathematical Soc., 2003.

[40] Olshevsky, V., and Pan, V. A unified superfast algorithm for boundary rational tangential interpolation problems and for inversion and factorization of dense structured matrices. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on* (1998), IEEE, pp. 192–201.

[41] Olshevsky, V., and Shokrollahi, M. A. A displacement approach to efficient decoding of algebraic-geometric codes. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA* (1999), pp. 235–244.

[42] Olshevsky, V., and Shokrollahi, M. A. Matrix-vector product for confluent Cauchy-like matrices with application to confluent rational interpolation. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA* (2000), pp. 573–581.

[43] Pan, V. Y. On computations with dense structured matrices. *Mathematics of Computation 55*, 191 (1990), 179–190.

[44] Pan, V. Y. *Structured Matrices and Polynomials: Unified Superfast Algorithms.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[45] Pan, V. Y., Rami, Y., and Wang, X. Structured matrices and Newton's iteration: unified approach. *Linear algebra and its applications 343* (2002), 233–265.

[46] Pan, V. Y., and Tsigaridas, E. P. Nearly optimal computations with structured matrices. *Theoretical Computer Science* (2017).

[47] Pernet, C. Computing with quasiseparable matrices. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation* (2016), ACM, pp. 389–396.

[48] Pernet, C., and Storjohann, A. Time and space efficient generators for quasiseparable matrices. *Journal of Symbolic Computation 85* (2017), 224–246.

[49] Potts, D., Steidl, G., and Tasche, M. Fast algorithms for discrete polynomial transforms. *Mathematics of Computation of the American Mathematical Society 67*, 224 (1998), 1577–1590.

[50] Simoncini, V. Computational methods for linear matrix equations. *SIAM Review 58*, 3 (2016), 377–441.

[51] Sindhwani, V., Sainath, T., and Kumar, S. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems* (2015), pp. 3088–3096.

[52] Trefethen, L. N., and Bau III, D. *Numerical linear algebra*, vol. 50. Siam, 1997.

[53] Vandebril, R., Van Barel, M., Golub, G., and Mastronardi, N. A bibliography on semiseparable matrices. *Calcolo 42*, 3 (2005), 249–270.

[54] Vandebril, R., Van Barel, M., and Mastronardi, N. A note on the representation and definition of semiseparable matrices. *Numerical Linear Algebra with Applications 12*, 8 (2005), 839–858.

[55] Vandebril, R., Van Barel, M., and Mastronardi, N. *Matrix computations and semiseparable matrices: linear systems*, vol. 1. JHU Press, 2007.

[56] Woodbury, M. A. Inverting modified matrices. *Memorandum report 42* (1950), 106.

[57] Yap, C.-K. *Fundamental problems of algorithmic algebra*, vol. 49. Oxford University Press Oxford, 2000.

[58] Yu, K.-H., Zhang, C., Berry, G. J., Altman, R. B., Ré, C., Rubin, D. L., and Snyder, M. Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. *Nature Communications 7* (2016).

[59] Zhao, L., Liao, S., Wang, Y., Li, Z., Tang, J., Pan, V., and Yuan, B. Theoretical Properties for Neural Networks with Weight Matrices of Low Displacement Rank. *ArXiv e-prints* (Mar. 2017).