PARAMETRIZATION OF NEWTON'S ITERATION FOR COMPUTATIONS WITH STRUCTURED MATRICES AND APPLICATIONS

Victor Pan

Department of Computer Science
Columbia University
New York, NY 10027

Department of Mathematics and Computer Science
Lehman College
CUNY, Bronx, NY 10468

and

Department of Computer Science, SUNYA
Albany, NY 12222

(Supported by NSF Grant CCR-8805782 and by
PSC-CUNY Awards #661340, #668541 and #669290)

CUCS-032-90

Summary: We apply a new parametrized version of Newton's iteration in order to compute (over any field F of constants) the solution or a least-squares solution to a linear system B**x** = **v** with an $n\times n$ Toeplitz or Toeplitz-like matrix B, as well as the determinant of B and the coefficients of its characteristic polynomial, det ($\lambda I - B$), dramatically improving the processor efficiency of the known fast parallel algorithms. Our algorithms, together with some previously known and some recent results of p904 p814 p188 p822 p425 as well as with our new techniques for computing polynomial gcds and lcms, imply respective improvement of the known estimates for parallel arithmetic complexity of several fundamental computations with polynomials and with both structured and general matrices.

---

parm.comp.jb12/5/90

1. Introduction.

Toeplitz matrices are defined as matrices with entries invariant in their shifts in the diagonal direction, and the more general class of Toeplitz-like matrices (including, say, the products and the inverses of Toeplitz matrices, as well as the resultant and subresultant matrices for a pair of polynomials) is defined by using some natural extension of this property, in terms of their displacement ranks (see definition 3.1 below).

Toeplitz and Toeplitz-like matrices are ubiquitous in signal processing and in scientific and engineering computing (see a vast bibliography in p430 p681 p437 p431 p432 p537 and have close correlation to many fundamental computations with polynomials, rational functions and power series (such as computing polynomial gcd, Padé approximation and extended Euclidean scheme for two polynomials), as well as with the resultant and subresultant matrices, which are Toeplitz-like matrices (see, for instance, p897 p019 p155 p904 Furthermore, computations with structured matrices of several other highly important classes (such as Hankel, Vandermonde, generalized Hilbert matrices and alike) can be immediately reduced to computations with Toeplitz-like matrices p822

Now we come to the main point: computations with Toeplitz and Toeplitz-like matrices (and consequently numerous related computations) have low complexity. In particular, a nonsingular linear system with an $n \times n$ Toeplitz or Toeplitz-like coefficient matrix can be solved very fast, in $O(n \log^2 n)$ arithmetic operations p019 p154 p616 rather than in $M(n) = O(n^\omega)$, required for a general nonsingular linear system of n equations, provided that $M(n) = O(n^\omega)$ arithmetic operations suffice for an $n \times n$ matrix multiplication. In theory, $2 \le \omega < 2.376$ p500 but in the algorithms applied in practice, ω is at best about 2.8 so far (see p036 p883 p103

Our main result is a dramatic improvement of the known parallel algorithms for Toeplitz and Toeplitz-like computations, immediately translated into similar improvements of computations with other structured matrices, polynomials, power series, and, perhaps somewhat surprisingly, even general matrices. This progress is mainly due to our novel technique, which we call *parametrization of Newton's algorithm for matrix inversion* (algorithm 2.1), but some other techniques and the ideas that we used may be of independent interest too, for instance, our reductions of computing the gcd and lcm of two polynomials to simple computations with Toeplitz matrices, the reduction to a Smith-like normal form of λ-matrices (matrix polynomials), which we apply in order to decrease the length of their displacement generators (in the proof of the proposition A.6), and the use (in the appendix A) of displacement operators ϕ^+ and ϕ^- (instead of the customary ϕ_+ and ϕ_-) in order to work out our approach over finite fields. The entire appendix A may be of interest as a concise survey of the main properties of such displacement operators and of related matrix computations.

Let us next specify our results and compare them with the known results, assuming the customary PRAM arithmetic model of parallel computing p612 p632 where every processor performs at most one arithmetic operation in every step. We will follow Brent's scheduling principle p632 that allows to save processors by slowing down the computations, so that $O_A(t, p)$ will denote the simultaneous upper bounds $O(ts)$ on the parallel arithmetic time and $\lceil p/s \rceil$ on the number of processors involved, where any $s \geq 1$ can be assumed.

The known best parallel algorithms for nonsingular Toeplitz linear systems over any field F of constants support the parallel complexity bounds either $O_A(n, \log^2 n)$ p616 p154 where the time complexity bound n is too high, or $O_A(\log^2 n, n^{\omega+1})$, with ω defined above p007 p826 where the processor bound is too high. The latter bounds can be improved to $O_A(\log^2 n, n^{\omega+0.5-\delta})$, for a positive $\delta = \delta(\omega)$, $\omega + 0.5 - \delta < 2.851$, if F allows division by n! p117 p032 and to $O_A(\log^2 n, n^2)$, if the input Toeplitz matrix is filled with integers or rationals p634 p585 p880 The algorithms of the latter papers support the sequential complexity bound $O_A(n^2 \log^2 n, 1)$, which is already close to the computational cost $O(n^2)$ of Durbin-Levinson's algorithm, widely used in practice for solving nonsingular Toeplitz systems; moreover, the algorithm of p880 also computes, for the cost $O_A(\log^2 n, n^2)$, the least-squares solution to a singular and even to a rank deficient Toeplitz linear system, and for this problem, the algorithm supports the record sequential time bound.

Substantial weakness of these algorithms of p634 p585 and p880 however, is due to the involvement of the auxiliary numerical approximations, which excludes any chance for applying the modular reduction techniques, accompanied with p-adic lifting and/or Chinese remainder computations, a customary means of bounding the precision of algebraic computations, so that the latter algorithms are prone to the numerical stability problems, known to be severe p430 for the Toeplitz and related computations, such as, say, the evaluation of the polynomial greatest common divisors (gcds). As usual, the numerical stability problems severely inhibit the practical application of the algorithms and imply their high Boolean cost, which motivates a further work on devising algorithms with a similarly low parallel cost but with no numerical approximation stage, so that they can be applied over any field of constants.

Since the complexity of Toeplitz-like computations has been long and intensively studied and has well-known applications to some fundamental computations with polynomials and general matrices p155 p814 p425 (both areas enjoying great attention of the researchers), our new progress, reported below, should seem surprising.

Indeed, our novel and completely algebraic approach works over any field of constants and improves all the previous parallel complexity bounds, even the estimates of p880 over integer matrices. Specifically, we reached the bounds $O_A(\log^2 n, n\, p_F(n) q_F(n) / \log n)$, over any field F, where

$$p_F(n) = \begin{cases} n \text{ if F supports FFT at } 2^h > n \text{ points} & (1.1) \\ n \log\log n \text{ otherwise,} \end{cases} \qquad (1.2)$$

$$q_F(n) = \begin{cases} 1 \text{ if F allows division by } n! & (1.3) \\ n \text{ otherwise.} \end{cases} \qquad (1.4)$$

These bounds support the evaluation of the determinant and the characteristic polynomial of a Toeplitz or Toeplitz-like matrix and the solution or a least-squares solution to a Toeplitz or Toeplitz-like linear system; they can be extended to computations with dense structured matrices of other classes (see above or p822 and can be applied to various further computational areas.

In particular, combining our results with the recent results of p188 p904 or, alternatively, with our simple but novel application of Padé approximations to computing the gcds and the lcms of two polynomials (section 5 below) dramatically improves the previous record estimate of $O_A(\log^2 n, n^{\omega+1})$ (of p814 for computing (over any field of constants F) the gcd of two polynomials of degrees at most n, to the bounds $O_A(\log^2 n, n\, p_F(n)/\log n)$ if F is infinite, $O_A(\log^2 n, n^2\, p_F(n)/\log n)$ if F is finite, and also leads to a similar dramatic improvement of the known parallel complexity bounds for other fundamental algebraic computations, such as computing all the entries of the extended Euclidean scheme for two polynomials, Padé approximation and the Berlekamp-Massey minimum span of a linear recurrence. Finally, by combining our results with the reductions due to p814 p425 p913 we arrive at new record estimates for the Las-Vegas type probabilistic parallel complexity of computing (over a field F) the solution $\mathbf{x} = A^{-1}\mathbf{v}$ to a linear system, with an $n\times n$ general coefficient matrix A, as well as of computing det A and A^{-1}, that is, at the estimates $O_A(\log^2 n, n^{\omega})$ if F is the field of real or complex numbers (which is within the logarithmic factor from *the optimum bounds* on the parallel complexity of this problem), $O_A(\log^3 n, n^2\, p_F(n)/\log n)$ otherwise. To be fair, the previous record bounds $O_A(\log^2 n, n^{\omega+1})$ of p007 p826 over any field, and $O_A(\log^2 n, n^{\omega+0.5-\delta})$ of p032 over the fields allowing divisions by n!, were deterministic.)

We will organize our presentation as follows: In the next section and in section 3, we will present our algorithms for computations with Toeplitz and Toeplitz-like matrices, respectively, over the fields allowing division by n!. In section 4, we will show an extension to any field. In section 5, we will comment on some further applications to computations with polynomials and general matrices. In the appendix A, we will review the relevant (old and new) techniques and results for computations with Toeplitz-like matrices. In the Appendix B, we will recall an expression for a least-squares solution to a linear system.

Acknowledgements: I wish to thank E. Kaltofen for preprints and also Joan Bentley, Pat Keller and Denise Rosenthal for their assistance with typing this paper.

2. Toeplitz Matrix Computations

Let us first consider computations over a field F of constants that allows division by $2, 3, \ldots, n$, that is, by (n!), and let us compute (over F) the characteristic polynomial, the inverse B^{-1} and, if F is infinite, also the Moore-Penrose generalized inverse B^{+} of a given $n\times n$ matrix B, by using Csanky's algorithm p168 and its extension to computing B^{+} (see p880 or appendix B below). The computation is reduced to computing the coefficients $c_0, \ldots, c_{n-1}$ of the characteristic polynomial of A,

$c(\lambda) = \det(\lambda I - B) = \lambda^n + \sum_{i=0}^{n-1} c_i \lambda^i$, $c_0 = (-1)^n \det B$, and this may in turn be reduced p120 see also p880 appendix A), for the cost $O_A(\log^2 n, p_F(n)/\log n)$, to computing the traces of the matrix powers $B, B^2, \ldots, B^{n-1}$.

We now propose a novelty, that is, we will compute the powers of B by means of Newton's algorithm for inverting the auxiliary matrix $A = I - \lambda B$, for the auxiliary parameter λ.

Algorithm 2.1, Parametrization of Newton's Iteration.

Input: natural n and k and an $n \times n$ matrix B.

Output: powers $I, B, B^2, \ldots, B^k$ of B, given by the k+1 coefficients of the matrix polynomial $X_d \bmod \lambda^{k+1}$, defined below.

Initialize: $X_0 := I$, $A := I - \lambda B$, $d := \lceil \log_2 (k+1) \rceil$.

Stage i, $i = 0, 1, \ldots, d-1$:

$$X_{i+1} := X_i (2I - AX_i). \tag{2.1}$$

The correctness of this simple algorithm over any ring of constants follows from the matrix equations, $I - AX_0 = \lambda B$, $I - AX_i = (I - AX_{i-1})^2 = (I - AX_0)^{2^i} = (\lambda B)^{2^i}$, for all i, so that

$$X_i = A^{-1} \bmod \lambda^{2^i}, \text{ for all i,} \tag{2.2}$$

and since $A^{-1} = (I - \lambda B)^{-1} = \sum_{j=0}^{\infty} (\lambda B)^j$, it follows that

$$X_i = \sum_{j=0}^{2^i - 1} (\lambda B)^j \bmod \lambda^{2^i}, \text{ for all i.} \tag{2.3}$$

[In fact, (2.1) implies that the degree of X_i in λ is at most $2^i - 1$, so that $X_i = \sum_{j=0}^{2^i-1} (\lambda B)^j$.]

For a general input matrix B, algorithm 2.1 is less effective than, say, algorithm of p016 p.128, for the same problem, but this comparison is dramatically reversed if B is a Toeplitz matrix.

We will rely on the following well-known result:

Fact 2.1.
An $n \times n$ *Toeplitz matrix* $T = [t_{ij}]$
[whose entries $t_{ij} = t_{i-j}$ *are invariant in their shift (displacement) in the down-right (diagonal) direction] has at most* $2n - 1$ *distinct entries and can be multiplied by a vector over a field* F *for the cost* $O_A(\log n, p_F(n))$ [see (1.1)-(1.2)] *of multiplication over* F *of two polynomials of degrees at most* $2n - 2$
and $n - 1$.

The inverse T^{-1} of an $n \times n$ nonsingular Toeplitz matrix may have the order of n^2 distinct entries, but it is usually suffices to store at most $2n - 1$ of them that form two columns of T^{-1}, the first, $\mathbf{x}$, and the last, $\mathbf{y}$ (see proposition 2.1 below).

Definition 2.1. $J = [\delta_{g,\, n-g}]$, $Z = [\delta_{i+1,\, j}]$ are the $n \times n$ matrices of reversion and lower shift, respectively, $\delta_{u,w}$ is Kronecker's symbol, $\delta_{u,u} = 1$, $\delta_{u,w} = 0$ if $u \neq w$, so that

$$J\,\mathbf{v} = [v_n, \ldots, v_1]^T,\ Z\mathbf{v} = [0, v_1, \ldots, v_{n-1}]^T,$$

for a vector $\mathbf{v} = [v_1, \ldots, v_n]^T$. $L(\mathbf{v})$ is the lower triangular Toeplitz matrix with the first column $\mathbf{v}$.

Proposition 2.1 (see p910 p433 p911 and p912 for proofs and extensions). *Let* $X = T^{-1}$ *be the inverse of a Toeplitz matrix,* $\mathbf{x}$ *be the first column and* $\mathbf{y}$ *be the last column of* X, $x_0 \neq 0$ *be the first component of* $\mathbf{x}$. *Then, over any field of constants,*

$$X = \frac{1}{x_0}(L(\mathbf{x})L^T(J\,\mathbf{y}) - L(Z\mathbf{y})L^T(Z^T\,J\mathbf{x})). \tag{2.4}$$

Now, let us revisit algorithm 2.1 where B and, consequently, $A = I - \lambda B$ are Toeplitz matrices, and therefore, due to (2.2), so are $X_i^{-1} \bmod \lambda^{2^i}$, $i = 0, 1, \ldots$. Due to proposition 2.1, it suffices to compute two columns of X_i (the first, $\mathbf{x}_i$, and the last, $\mathbf{y}_i$), for each i, so that the right side of (2.4), with $\mathbf{x} = \mathbf{x}_i$ and $\mathbf{y} = \mathbf{y}_i$, will equal $X_i \bmod \lambda^{2^i}$. [Since $X_i = I \bmod \lambda$, the northwestern entry, (1,1), of X_i equals $1 \bmod \lambda$, and thus has a reciprocal, so that proposition 2.1 can be applied to $X = X_i$, for all i.] We shall bound the degrees of all the polynomials in λ involved in the evaluation of X_{i+1} according to (2.1) (and therefore, the complexity of operating with such polynomials), by reducing them modulo λ^s, $s = 2^{i+1}$.

We may apply (2.4) to $X = X_i \bmod \lambda^{2^i}$, but generally not to $X = X_i \bmod \lambda^{2^{i+1}}$, whose inverse may not be a Toeplitz matrix, but already (2.4), for $X = X_i \bmod \lambda^{2^i}$, suffices for our purpose, because we only need to use $X_i \bmod \lambda^{2^i}$ in order to arrive at $X_{i+1} \bmod \lambda^{2^{i+1}}$, by means of (2.1) [since $I - AX_{i+1} = (I - AX_i)^2$], and thus we will always replace X_i in (2.1) by the right side expression of its representations according to (2.4) (for $X = X_i$).

Dealing with Toeplitz matrix polynomials modulo λ^s (that is, with Toeplitz matrices filled with polynomials modulo λ^s), we shall change the cost bounds of fact 2.1 into the bounds of p869

$$c_A(F) = O_A(\log n,\ sp_F(n)), \tag{2.5}$$

on the cost of multiplication of two bivariate polynomials of degrees at most 2s and 2n in their two variables, respectively.

Due to proposition 2.1, each step (2.1) essentially reduces to 2 multiplications of each of the matrices A and X_i by vectors, that is, to 10 multiplications of $n \times n$ Toeplitz matrices by vectors, whose entries are polynomials modulo λ^s, $s = 2^{i+1}$, and therefore, each step (2.1) has the complexity bounds $O_A(\log n,\ sp_F(n))$.

We slow down the computation to save processors and arrive at the estimates $O_A((\log^2 n)(s/n),\ n\,p_F(n)/\log n)$, for $s > n/\log n$, then sum the time bounds over all $i, i=1,\ldots,d$, $d = \lceil \log_2(n+1) \rceil$, and thus, estimate the overall cost of algorithm 2.1 (with $k = n$) as $O_A(\log^2 n,\ n\,p_F(n)/\log n)$ provided that the output is represented by two columns (the first and the last) of $X_d \bmod \lambda^{n+1}$.

We then need to compute the trace of $A^{-1} \bmod \lambda^{n+1} = X_d \bmod \lambda^{n+1} = \sum_{i=0}^{n} (\lambda B)^i$. Applying proposition 2.1, we reduce this problem essentially to two stages, each consisting in computing n inner products, of the k-th row of a (lower triangular) Toeplitz matrix polynomial modulo λ^{n+1} by the k-th column of an (upper triangular) Toeplitz matrix polynomial modulo λ^{n+1}, for $k=1,\ldots,n$. Due to the Toeplitz structure of the input matrices, each of these two stages is reduced to n concurrent polynomial multiplications modulo λ^{n+1} and to computing the sum and the $n-1$ partial sums of the resulting polynomials.

The complexity of these computations is surely within the overall bounds $O_A(\log^2 n,\ n\,p_F(n)/\log n)$ (we use the parallel prefix computation algorithm for the summation, see p612 p632 as also is the complexity of the already cited transition from trace($A^{-1} \bmod \lambda^{n+1}$) (which gives us the traces of $B, B^2, \ldots, B^n$) to the coefficients of $c(\lambda) = \det(\lambda I - A)$, as well as the cost of computing $\mathbf{x} = B^{-1}\mathbf{v}$ and /or $\mathbf{x} = B^{+}\mathbf{v}$, given such coefficients, a vector $\mathbf{v}$ and the two columns (the first and the last) of $A^{-1} \bmod \lambda^{n+1}$. [Indeed, we have already commented on the transition from the traces to the coefficients; for computing $B^{-1}\mathbf{v}$ or $B^{+}\mathbf{v}$, we first apply proposition 2.1 to compute $A^{-1}\mathbf{v}$, which gives us the vectors $B^k\mathbf{v}$, for $k=1,\ldots,n$, and then recover $B^{+}\mathbf{v}$ as their linear combination $\sum_k g_k B^k \mathbf{v}$, with the scalars g_k defined by $c(\lambda)$, (compare the appendix B below). For a nonsingular matrix B, we obtain $B^{-1}\mathbf{v} = B^{+}\mathbf{v}$.]

We thus arrive at the following result:

Proposition 2.2.

Given a positive integer n, *a field* F *allowing division by* n!, *an* $n \times n$ *Toeplitz matrix* B, *and an n-dimensional vector* $\mathbf{v}$, *it is possible to compute over* F, *for the cost* $O_A(\log^2 n,\ n\,p_F(n)/\log n)$:

a) *the coefficients* $c_0, \ldots, c_{n-1}$ *of the characteristic polynomial of* B, $\sum_{i=0}^{n} c_i \lambda^i = \det(\lambda I - B)$, which also gives us $\det B = (-1)^n c_0$; *if* F *is infinite, then* $\operatorname{rank} B = n - \min(i{:}c_i \neq 0) = \operatorname{trace}(B^{+}B)$;

b) *the solution* $\mathbf{x} = B^{-1}\mathbf{v}$ *to the linear system* $B\mathbf{x} = \mathbf{v}$ *if* B is nonsingular;

c) *the least-squares solution* $\mathbf{x} = B^{+}\mathbf{v}$ *to* $B\mathbf{x} = \mathbf{v}$ *if* F is infinite.

Remark 2.1. Due to proposition 2.1 and fact 2.1, we may extend the estimates of proposition 2.2 to computing the inverse B^{-1} of any $n \times n$ nonsingular Toeplitz matrix B, provided that the (1,1) entry of B^{-1} has a reciprocal. The latter assumption about the reciprocal can be removed by using proposition A.7 below, instead of proposition 2.1 above.

3. Extension to Other Classes of Dense Structured Matrices.

Let us extend the estimates of proposition 2.2 to the important case where B is a dense and structured but non Toeplitz matrix, whose study can be found in p426 p165 p806 and p897

Definition 3.1 p426 p165 A pair of $n \times r$ matrices G and H is a *generator of length* r for an $n \times n$ matrix $A = GH^T$. The *rank* of A equals the minimum length of generators for A. For a linear operator ϕ defined on the space of $n \times n$ matrices, a generator and the rank of $\phi(A)$ are called an ϕ - *generator* and the ϕ - *rank* of A.

Following and extending p426 p165 we will first define four *displacement operators,* naturally associated with Toeplitz matrices:

$$\phi_+(A) = A - ZAZ^T, \tag{3.1}$$

$$\phi_-(A) = A - Z^T AZ, \tag{3.2}$$

$$\phi^+(A) = AZ - ZA, \tag{3.3}$$

$$\phi^-(A) = AZ^T - Z^T A, \tag{3.4}$$

and then will define the *displacement ranks* and *displacement generators* of matrices as their ϕ-ranks and ϕ-generators, for $\phi = \phi_+$, $\phi = \phi_-$, $\phi = \phi^+$ and $\phi = \phi^-$ or, equivalently, as the ranks and the generators of $\phi_+(A)$, $\phi_-(A)$, $\phi^+(A)$ and $\phi_-(A)$.

The displacement ranks are at most 2 for all Toeplitz matrices and for their inverses (if there exist the inverses), at most $m + n$ for all $m \times n$ block matrices with Toeplitz blocks, in particular, at most $m + n = 3$ for the resultant and subresultant matrices, and at most 4 for the product of two Toeplitz matrices (see Appendix A below, p897 p426 p432 p165 and p537 on some basic properties and applications of the displacement ranks and generators). The matrices having smaller displacement ranks are sometimes called *Toeplitz-like matrices* .

Hereafter, we will use the displacement ranks and generators for $n \times n$ matrices and matrix polynomials in λ modulo λ^s, over a field F, for $s<2^n$. We will next prove the following extension of proposition 2.2:

Proposition 3.1.
Given an $n \times n$ *matrix* B *with its displacement generator of length* r, *over an infinite field of constants* F, *the complexity estimates of proposition 2.2 can be extended to*

$$c_A = O_A(r \log^2 n,\ r\, n\, p_F(n)/\log n)$$

[compare (1.1)-(1.4)].

The basis for the latter extensions, as well as for many other effective algorithms for computations with various classes of dense structured matrices, is their representation by means of their ϕ-generators of smaller length, for an associated operator ϕ, so that all the operations with matrices are replaced by the operations with their ϕ-generators.

In the appendix A we will list and prove some basis results for such computations (see propositions A.1-A.7).

Now, let us apply these results instead of proposition 2.1, and otherwise let us follow the line of the proof of proposition 2.2 in order to prove proposition 3.1. To be certain, let us be given a matrix B with its ϕ_+-generator of length $r, 1 \le r$ (similarly, for ϕ^+, ϕ^- or ϕ_--generators), and let us apply algorithm 2.1, for k=n. Then we deduce from (2.1)-(2.3) and proposition A.4 below that rank ϕ_- $(X_i \bmod \lambda^{2^i}) \le r$. We surely have an ϕ_--generator of length 1 for $X_0 = I$; we will apply induction on i, assuming for each $i \ge 0$ that we are given an ϕ_--generator of length at most r for X_i modulo λ^{2^i}, and will compute, for the cost $O_A(r\log n, r\, n\, p_F(x))$, an ϕ_--generator of length at most r, for X_{i+1} modulo $\lambda^{2^{i+1}}$.

Specifically, we first apply proposition A.5, for $s=2^{i+1}$, and compute an ϕ_--generator of length at most $R = 3r + c(\phi_-) = 3r + 2$, for X_{i+1} mod λ^s. The cost of this stage is $O_A(\log n,\ r^2 s\, p_F(n))$, or after a slowdown, $O_A(r \log n,\ rs\, p_F(n))$. Then for the cost satisfying the same bounds, we compute an ϕ_--generator of length r for X_{i+1} mod $\lambda^{2^{i+1}}$, by using proposition A.6. Thus, the cost of the transition from X_i mod λ^{2^i} to X_{i+1} mod $\lambda^{2^{i+1}}$ (where the matrices are represented by their ϕ_--generators of length r) is $O_A(r \log n,\ rs\, p_F(n))$, so that the overall cost (for all i) is bounded by $O_A(r \log^2 n,\ rs\, p_F(n)/\log n)$, as we need. (Here again, we use appropriate slowdown, to save processors.) The transition to computing the coefficients of $c(\lambda) = \det(\lambda I - B)$ and the vector $B^+\mathbf{v}$ is now performed as in the proof of proposition 2.2, but with using proposition A.1 instead of proposition 2.1. $\square$

Remark 3.1. Proposition 2.2 is a special case of proposition 3.1, where $r \le 2$. Our algorithm supporting proposition 2.2, however, is a little simpler (with by a constant factor decrease of the cost bounds) than our algorithm supporting proposition 3.1.

Remark 3.2. Based on proposition A.1 below, the complexity bounds of proposition 3.1 can be extended to the evaluation of an ϕ_--generator of length r, for the inverse of a nonsingular $n\times n$ matrix B given with its ϕ_+-generator of length r, provided that we are also given a pair of $n\times r$ matrices S and R^T such that $C = R\phi_-(B^{-1})S$ is an $r\times r$ nonsingular matrix, since in this case, $\phi_-(B^{-1}) = GH^T$, $G = \phi_-(B^{-1})S$, $H^T = C^{-1}R\ \phi_-(B^{-1})$ (compare p154 For an appropriate random choice of R and S, the nonsingularity of the matrix C can be ensured with high a probability even over the finite fields (see p425 Similar observations can be made given another displacement generator of B, rather than ϕ_+.

Remark 3.3. The results for computing the determinant and the inverse of Toeplitz-like matrices and for solving linear systems defined by such matrices can be extended to the case of all Vandermonde-like, Hankel-like, and Hilbert-like matrices by means of the techniques of p822

4. Extension to Any Field of Constants.

In this section, we will combine our algorithms of the previous sections with the algorithm of p826 in order to extend our results to computations over any field, where the division by n! is not generally allowed. Similar extension can be based on the algorithm of p007 rather than of p826 and in both cases, the extension requires to use by n times more processors to support the same time bound $O(\log^2 n)$, but as a by-product, the characteristic polynomials of all the $k \times k$ leading principal submatrices B_k of B are also computed (for the same cost), $k=1,\ldots,n$.

The algorithm of p826 relies on the following equations for the reverse characteristic polynomials of B_k:

$$y_k(\lambda) = \det(I_k - \lambda\, B_k) = \sum_{i=0}^{k} c_{i,k}\, \lambda^{k-1} = 1/\prod_{j=1}^{k}((I_j - \lambda B_j^{-1}))_{j,\,j} \bmod \lambda^{k+1}, \tag{4.1}$$

$k=1,\ldots,n$, where I_j denotes the $j \times j$ identity matrix and $W_{i,\,j}$ denotes the entry (i,j) of a matrix W.

Our extension of algorithm 2.1 to the case of any field F follows.

Algorithm 4.1.

Input: an $n \times n$ matrix B.

Output: the coefficients $c_{i,k}$, $i = 0,1,\ldots,k-1$, of the characteristic polynomials of B_k, the $k \times k$ leading principal submatrices of B, for $k = 1,2,\ldots,n$,

$$c_k(\lambda) = \det(\lambda I_k - B_k) = \sum_{i=0}^{k} c_{i,\,k}\, \lambda^i,\ c_{k,\,k} = 1,\ c_{0,\,k} = (-1)^k \det B_k, \tag{4.2}$$

where I_k is the $k \times k$ identity matrix.

Computations:

1) *n* times call algorithm 2.1, for $B = B_j$, to compute the polynomials

$$b_j(\lambda) = ((I - \lambda B_j)^{-1})_{j,\,j} \bmod \lambda^{n+1},$$

for $j = 1,2,\ldots,n$;

2) apply the parallel prefix algorithm p612 p632 to compute modulo λ^{k+1} the products

$$p_k(\lambda) = \prod_{j=1}^{k} b_j(\lambda) \bmod \lambda^{k+1},\ k = 1,\ldots,n;$$

each of the $\lceil \log_2 n \rceil$ steps of this algorithm amounts to $\lceil \frac{n}{\log_2 n} \rceil$ polynomial multiplications modulo λ^s, for $s \le n+1$;

3) for every $k, k = 1, \ldots, n$, apply $g(k) = \lceil \log_2 (k+1) \rceil$ steps of Newton's iteration for the equation $\frac{1}{y_k(\lambda)} - p_k(\lambda) = 0$:

$$y_{0,\,k}(\lambda) = 1,\ y_{i+1,\,k}(\lambda) = y_{i,\,k}(\lambda)\,(2 - p_k(\lambda) y_{i,\,k}(\lambda)) \bmod \lambda^{2^{i+1}},\ i = 0, \ldots, g(k) - 1, \tag{4.3}$$

in order to compute and output the coefficients of the reverse characteristic polynomial $y_{g(k),\,k}(\lambda) = (1/\,p_k(\lambda)) \bmod \lambda^{k+1} = \det(I_k - \lambda B_k)$.

The correctness of algorithm 2.2 immediately follows from the equations (4.1), from the observation that

$$p_k(\lambda) = 1 \bmod \lambda, \text{ for all } k, \tag{4.4}$$

and from the equations (4.3), which imply that

$$1 - y_{i,\,k+1}(\lambda) p_k(\lambda) = (1 - y_{i,\,k}\ (\lambda) p_k(\lambda))^2,$$

and therefore, due to (4.4), that

$$1 - y_{i,k}(\lambda) p_k(\lambda) = 0 \bmod \lambda^{2^i},\ i = 0, 1, \ldots \qquad \square$$

Algorithm 4.1 enables us to extend our results of sections 2 and 3 to computations over any field of constants, but the overall complexity bounds increase to $O_A(r\log^2 n,\ r\,n^2 p_F(n)/\log n)$ (for any $n \times n$ input matrix B given with its displacement generator of length r), since we need to involve the submatrices B_k, for $k = 1, 2, \ldots, n$,

5. Some Further Extensions.

Let us extend our comments given in the introduction on further applications of our results (see more in p897 The techniques of p904 and p188 reduce the evaluation of the polynomial gcds and all other entries of the extended Euclidean scheme for two polynomials of degrees at most n over any field F to some computations with Toeplitz and Hankel matrices, in particular, to their inversion and the evaluation of their ranks and/or determinants. By using our algorithm at the latter stages, we arrive at the new record complexity estimates for these computations, reaching, for the gcd, the bounds of the introduction, and for the Euclidean scheme, over infinite fields F, the bounds $O_A(\log^3 n,\ n\,p_F(n)/\log^2 n)$ [compare (1.1)-(1.2)].

These bounds, with n replaced by m+n, can be extended to computing the (m,n) Padé approximation of any analytic function; this computation can alternatively be reduced to solving a consistent Toeplitz system $B\mathbf{x} = \mathbf{v}$ of n linear equations with n unknowns and to multiplying an $m \times n$ Toeplitz matrix by a vector p019 Moreover, due to parts d) and e) of theorem 2 of p019 (reproduced in p019 from p437 even if this system is singular, we may compute the rank r of its $n \times n$ coefficient matrix B and then conclude that the $r \times r$ northwestern (that is, leading principal) submatrix of B is nonsingular. Thus, the overall complexity of computing the (m,n) Padé approximation is bounded by $O_A(\log^2 n + \log m,\ p_F(m) + n\, p_F(n)/\log n)$, over an infinite field F. Over the finite fields, we apply Chistov's algorithm and obtain $r = \max\{k\colon \det B_k = O\}$, so the overall cost of the solution is $O_A(\log^2 n + \log m,\ p_F(m) + n^2\, p_F(n)/\log n)$.

Let us next show an application of our algorithms for Toeplitz computations to computing $m(x) = \mathrm{lcm}(p(x), q(x))$, the least common multiple (lcm) of two polynomials p(x) and q(x). This also gives us $d(x) = \gcd(p(x), q(x))$, the greatest common divisor (gcd) of these polynomials, since $d(x) = p(x)\, q(x)/ m(x)$. Conversely, $m(x) = p(x)\, q(x)/ d(x)$.

Computing m(x), we assume (with no loss of generality) that $p(0) = q(0) = 1$. Let $m = \deg(p(x)\, q(x))$, $n = \deg(p(x) + q(x))$, $N = m + n + 1$, and apply the following algorithm:

Algorithm 5.1, computing polynomial lcms.

1. Compute the first N Taylor's coefficients of the analytic function $a(x) = \left[\frac{1}{p(x)} + \frac{1}{q(x)}\right]^{-1} = \sum_{j=0}^{+\infty} a_j x^j$, that is, compute the coefficients of the polynomial $(1/p(x) + 1/q(x))^{-1} \bmod x^N = \sum_{j=0}^{N-1} a_j x^j$.
2. Compute the rank r of an $n \times n$ Toeplitz matrix with the first row $[a_m, a_{m+1}, \ldots, a_{m+n-1}]$ and with the first column $[a_m, a_{m-1}, \ldots, a_{m-n+1}]^T$, where $a_s = 0$ for $s < 0$. (For such a matrix, its $r \times r$ leading principal submatrix is nonsingular.)
3. Compute the $(m-r, n-r)$ Padé approximation $[u(x), v(x)]$ to the function a(x) and output $u(x) = \mathrm{lcm}(p(x), q(x))$.

The correctness of this algorithm immediately follows from the parts d) and e) of theorem 2 of p019 (reproduced from p437

The complexity of this algorithm is upper bounded by the complexity of computing the rank and the $(m-r, n-r)$ Padé approximation. Thus we arrive at an alternate derivation of the results of p904 for computing the gcd and the lcm of two polynomials.

Algorithm 5.1 can be modified in order to output $u(x) = \gcd(p(x), q(x))$ if we set $m = \deg p(x)$, $n = \deg q(x)$, $a(x) = p(x)/ q(x)$.

Computing the minimum span for a (2n)-term linear recurrence sequence can be reduced to computing the $(n-1, n)$ [or alternatively, the (n,n)] Padé approximation, whose complexity estimates are thus extended to computing the minimum span. As this was earlier observed in p814 based on p425 such estimates were the only remaining stage for proving the record complexity bounds $O_A(\log^2 n, n^{\omega})$, $\omega < 2.376$, for randomized parallel computations with general $n \times n$ matrices over infinite fields (that is, for computing the determinant of a matrix and solving a linear system of equations). Specifically, the two latter problems are first reduced in p425 to computing the minimum polynomial of B (or of RBS, for random matrices R and S of appropriate sizes), and then to two stages [repeated O(1) times]:

a) compute the Krylov sequence of vectors $\mathbf{w}_i = B^i \mathbf{v}$ [or $(RBS^T)^i \mathbf{v}$] and then the (2n)-term sequence of scalars $\mathbf{u}^T B^i \mathbf{v}$ [or $\mathbf{u}^T (RBS^T)^i \mathbf{v}$], $i = 1, \ldots, 2n-1$, for two random vectors $\mathbf{u}$ and $\mathbf{v}$ [an algorithm of p042 (compare p016 p. 128) performs this stage for the cost $O_A(\log^2 n, n^{\omega})$];

b) find the minimum span of the latter sequence of scalars (and here we show the desired improvement).

Computing the inverse is then reduced to computing the determinant, for the same parallel cost, within a constant factor (see p913

Over the finite fields F, the same algorithms for general matrices have the cost bounds $O_A(\log^2 n, n^2 \, p_F(n) / \log n)$ dominated by the cost bounds for Padé approximation.

Appendix A. Some Properties of Displacement Generators.

All the results of this appendix hold over any field of constants, and the input matrices and vectors have entries being polynomials in λ modulo λ^s, $s = 2^i$ for i of (2.1)-(2.3). The reader may compare our exposition with previous ones, such as p426 p152 p165 The first proposition and corollary immediately follow from definition 3.1.

Proposition A.1 p426 p165
A pair (G, H) of $n \times r$ *matrices* $G = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_r]$ *and* $H = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_r]$
is a generator of length r *for an* $n \times n$
matrix $B - ZBZ^T$ *if and only if*

$$B = \sum_{i=1}^{r} L(\mathbf{g}_i) \, L^T(\mathbf{h}_i)$$

and for the matrix $B - Z^T B Z$ *if and only if*

$$B = \sum_{i=1}^{r} L^T(\mathbf{g}_i) L(\mathbf{h}_i).$$

Corollary A.1 (see p154 Lemma 5). *For any pair of vectors* $\mathbf{g}$ *and* $\mathbf{h}$
of the same dimension,

$$L(\mathbf{g}) L^T(\mathbf{h}) = L(\mathbf{a}) + L^T(\mathbf{b}) - L^T(ZJ\mathbf{g}) L(ZJ\mathbf{h}),$$

$$L^T(\mathbf{g}) L(\mathbf{h}) = L^T(\mathbf{c}) + L(\mathbf{d}) - L(ZJ\mathbf{g}) L^T(ZJ\mathbf{h})$$

where J *and* Z *are the matrices of Definition 2.1,* $\mathbf{a}^T J$ *is the last row and* $J\mathbf{b}$ *is the last column of* $L(\mathbf{g})L^T(\mathbf{h})$, $\mathbf{c}^T$ *is the first row and* $\mathbf{d}$ *is the first column of* $L^T(\mathbf{g})L(\mathbf{h})$.

Due to corollary A.1, we may immediately define an ϕ_+ (respectively, an ϕ_-)-generator of length $r + 2$ for a matrix, given its ϕ_- (respectively, its ϕ_+)-generator of length r. Let us next show some simple correlations among the representations (3.1)-(3.4).

Proposition A.2.
Let $\mathbf{i}_1 = [1,0,...,0]^T$, $\mathbf{i}_n = [0,...,0,1]^T$. *Then*

$$\phi^+(A)Z^T = \phi_+(A) - A\mathbf{i}_1\mathbf{i}_1^T,$$

$$Z^T\phi^+(A) = \mathbf{i}_n\mathbf{i}_n^T A - \phi_-(A),$$

$$\phi^-(A)Z = \phi_-(A) - A\ \mathbf{i}_n\mathbf{i}_n^T,$$

$$Z\ \phi^-(A) = \mathbf{i}_1\mathbf{i}_1^T A - \phi_+(A),$$

$$\phi_+(A)Z = \phi^+(A) + ZA\mathbf{i}_n\mathbf{i}_n^T,$$

$$Z^T\phi_+(A) = \mathbf{i}_n\mathbf{i}_n^T AZ^T - \phi^-(A),$$

$$\phi_-(A)Z^T = \phi^-(A) + Z^T A\ \mathbf{i}_1\mathbf{i}_1^T,$$

$$Z\phi_-(A) = \mathbf{i}_1\mathbf{i}_1^T AZ - \phi^+(A).$$

Proof. Observe that

$$Z^T Z = I - \mathbf{i}_n\ \mathbf{i}_n^T,\ ZZ^T = I - \mathbf{i}_1\ \mathbf{i}_1^T, \tag{A.1}$$

pre- and postmultiply each of the matrix equations (3.3) by Z^T, (3.4) by Z, substitute (3.1), (3.2) and (A.1) and arrive at the first four equations of proposition A.2. Then postmultiply (3.1) by Z, (3.2) by Z^T, premultiply (3.1) by Z^T, (3.2) by Z, substitute (3.3), (3.4) and (A.1) and deduce the last four equations of proposition A.2. □

The eight equations of proposition A.2 enable us to compute the ϕ_+- and ϕ_--generators of length at most r+1 for the matrix A given its ϕ^+- or its ϕ^--generator of length r and to compute the ϕ^+- and ϕ^--generators of length at most r+1 for A given its ϕ_+- or its ϕ_--generator of length r.

We will modify the original proofs of the two following results, so as to deduce them over any field of constants.

Proposition A.3.
Given a displacement operator ϕ
and a pair of ϕ- *generators of lengths* a *and* b, *for a pair of* $n\times n$ *matrices* A
and B, *we may immediately obtain an* ϕ- *generator of length at most* $a + b$ for $A + \alpha B$ *(for any fixed scalar* α); *furthermore, we may also compute [over a field F, for the cost of* $O_A(\log n, ab\ p_F(n))$] ϕ- *generators of lengths at most* $a + b + 1$ *for* AB (see p165 p822

The latter length bound decreases by 1, to $a+b$, if $\phi = \phi^{+}$ or $\phi = \phi^{-}$.

Proof. We only need to prove the part about computing AB, and we will only consider the cases $\phi = \phi^{+}$ and $\phi = \phi_{+}$, since the cases $\phi = \phi^{-}$ and $\phi = \phi_{-}$ are treated similarly.

First let $\phi = \phi^{+}$ and observe that

$$\phi^{+}(AB) = ABZ - ZAB = A\,(BZ - ZB) + (AZ - ZA)B =$$

$$A\phi^{+}(B) + \phi^{+}(A)\,B = AG_{B}^{+}(H_{B}^{+})^{T} + G_{A}^{+}(H_{A}^{+})^{T}B = G_{AB}^{+}(H_{AB}^{+})^{T}$$

provided that $\phi^{+}(C) = G_{C}^{+}(H_{C}^{+})^{T}$, for $C = A$ and for $C = B$, $G_{AB}^{+} = [AG_{B}^{+},\ G_{A}^{+}]$, $H_{AB}^{+} = [H_{B}^{+},\ B^{T}\ H_{A}^{+}]$. To compute AG_{B}^{+} and $B^{T}H_{A}^{+}$ remaining within the required complexity bounds, we just rely on the representation of the matrices A and B according to proposition A.1. This way we settle the case where $\phi = \phi^{+}$.

Next let $\phi = \phi_{+}$, recall (A.1), denote $\mathbf{u} = ZA\mathbf{i}_{n}$, $\mathbf{v}^{T} = \mathbf{i}_{n}^{T}BZ^{T}$ and deduce that

$$\phi_{+}(AB) = AB - ZAIBZ^{T} = AB - (ZAZ^{T})\,(ZBZ^{T}) +$$

$$ZA\mathbf{i}_{n}\mathbf{i}_{n}^{T}BZ^{T} = (A - ZAZ^{T})\,B + ZAZ^{T}(B - ZBZ^{T}) + \mathbf{u}\mathbf{v}^{T} =$$

$$\phi_{+}(A)\,B + ZAZ^{+}\phi_{+}(B) + \mathbf{u}\mathbf{v}^{T},$$

and this settles the case of $\phi = \phi_{+}$. □

Proposition A.4 p426

If A is a nonsingular matrix, then

$$\text{rank}\ \phi^{+}(A^{-1}) = \text{rank}\ \phi^{+}(A),$$

$$\text{rank}\ \phi^{-}(A^{-1}) = \text{rank}\ \phi^{-}(A),$$

$$\text{rank}\ \phi_{+}(A^{-1}) = \text{rank}\ \phi_{-}(A).$$

Proof. The first two equations are immediately obtained from the equations (3.3) and (3.4), respectively, by pre- and postmultiplying both (3.3) and (3.4) by A^{-1}.

To arrive at the last equation of proposition A.4, A^{-1} deduce that $\text{rank}\ \phi_{-}(A) = \text{rank}\ (A - Z^{T}AZ) =_{(\text{premultiply by } A^{-1})} \text{rank}\ (I - A^{-1}Z^{T}AZ)$.

At this point, observe that $\text{rank}\ (I - BZ) = \text{rank}\ (I - ZB) = 1 + \text{rank}\ (I_{n-1} - B_{n,1})$, for any $n \times n$ matrix B and its $(n-1)\times(n-1)$ submatrix $B_{n,1}$ obtained by deleting the last row and the first column of B. [Here, I_{n-1} denotes the $(n-1)\times(n-1)$ identity matrix.] In particular, for $B = A^{-1}Z^{T}A$, we obtain that

$$\text{rank}\ \phi_{-}(A) = \text{rank}\ (I - A^{-1}Z^{T}AZ) = \text{rank}\ (I - ZA^{-1}Z^{T}A) =_{\text{postmultiply by } A^{-1}}$$

$$\text{rank}\ (A^{-1} - ZA^{-1}Z^{T}) = \text{rank}\ \phi_{+}(A^{-1}).$$ □

Note that proposition A.4 expresses through each other the displacement ranks, but not the displacement generators, of A and A^{-1}.

Proposition A.5.
For the cost $O_A(\log n,\ r^2 s\, p_F(n))$,
an ϕ*-generator of length at most* $3r + c(\phi)$,
for $X_{i+1} \bmod \lambda^s$, $s=2^{i+1}$, *can be computed over any field* F given an ϕ- *generator of length* r, for $X_i \bmod \lambda^{2^i}$, *and* ϕ^**- generator of length at most* r *for* A,
provided that (2.1) holds and that one of the four cases takes place:

a) $\phi = \phi^* = \phi^+$, $c(\phi) = 0$,

b) $\phi = \phi^* = \phi^-$, $c(\phi) = 0$,

c) $\phi = \phi_+$, $\phi^* = \phi_-$, $c(\phi) = 2$,

d) $\phi = \phi_-$, $\phi^* = \phi_+$, $c(\phi) = 2$.

Proof. Proposition A.5, with $c(\phi)$ increased to 1 in the cases a) and b) and to 5 in the cases c) and d) can be immediately deduced by combining propositions A.1-A.3, corollary A.1 and fact 2.1. (This would still suffice for the proof of all our main results of this paper.) We will, however, also give a direct proof for the smaller $c(\phi)$ in the cases a) and c) [the cases b) and d) can be treated similarly].

Case a). Observe that

$$\phi^+(X_{i+1}) = \phi^+(X_i(2I - AX_i)) =$$

$$X_i(2I - AX_i)\, Z - ZX_i(2I - AX_i) =$$

$$2(X_i\, Z - ZX_i) - (X_iAX_i\, Z - ZX_iAX_i) =$$

$$2(X_i\, Z - ZX_i) - X_iA(X_i\, Z - ZX_i) - (X_iAZ - ZX_iA)X_i =$$

$$2\phi^+(X_i) - X_iA\, \phi^+(X_i) - X_i(AZ - ZA)\, X_i - (X_iZ - ZX_i)\, AX_i =$$

$$(I - X_iA)\, \phi^+(X_i) + \phi^+(X_i)\, (I - AX_i) - X_i\phi^+(A)\, X_i. \tag{A.2}$$

Since we are given ϕ^+-generators of lengths at most r for X_i and A, that is,

$$\phi^+(X_i) = G^+(i)\, (H^+(i))^T, \tag{A.3}$$

$$\phi^+(A) = G^+(H^+)^T, \tag{A.4}$$

it remains to substitute (A.3) and (A.4) into (A.2), to deduce that

$$\phi^+(X_{i+1}) = (I - X_iA)\, G^+(i)\, (H^+(i))^T + G^+(i)\, (H^+(i))^T\, (I - AX_i) + X_iG^+\, (H^+)^T X_i = G^+(i{+}1)\, (H^+(i{+}1))^T,$$

and to evaluate modulo λ^{n+1} the $n\times(3r)$ matrices,

$$G^+(i{+}1) = [(I - X_iA)\, G^+(i),\ G^+(i),\ X_iG^+],$$

$$H^+(i+1) = [H^+(i), (I - AX_i)^T H^+(i), X_i^T H_i^+].$$

To perform the latter step within the cost bound $O_A(\log n, r^2 s\, p_F(n))$, it suffices to decompose A and X_i according to proposition A.1 and to reduce each multiplication modulo λ^{n+1} of A, A^T, X_i or X^T by an $n\times r$ matrix to $O(r^2)$ multiplications, each of an $n\times n$ Toeplitz matrix by a vector, for the cost bounded by (2.5). This settles the case a).

Case c). Observe that

$$\phi_+(X_{i+1}) = \phi_+(X_i(2I - AX_i)) =$$

$$X_i(2I - AX_i) - ZX_i\,(2I - AX_i)\,Z^T =$$

$$2(X_i - ZX_iZ^T) - (X_iAX_i - ZX_iAX_iZ^T) =$$

$$2\phi_+\,(X_i) - (X_i - ZX_iZ^T)AX_i -$$

$$ZX_i(Z^TAX_i - AX_iZ^T) = \phi_+(X_i)\,(2I - AX_i) -$$

$$-ZX_i[(Z^TA - AZ^T)\,X_i + A(Z^TX_i - X_iZ^T)] =$$

$$\phi_+(X_i)\,(2I - AX_i) + ZX_i(\phi^-(A)\,X_i + A\,\phi^-(X_i)) =$$

$$\phi_+(X_i)\,(2I - AX_i) + ZX_i(\phi_-(A)Z^TX_i - Z^TA\mathbf{i}_1\mathbf{i}_1^T -$$

$$AZ^T\phi_+(X_i) + A\mathbf{i}_n\mathbf{i}_nX_iZ^T)$$

[compare proposition A.1] and arrive at the desired generator for $\phi_+(X_{i+1})$ by representing the matrices A and X_i according to proposition A.1, by using generators of lengths at most r for the matrices $\phi_-(A)$ and $\phi_+(X_i)$ and by bounding, by means of (2.5), the cost of multiplication modulo λ^s of the matrices A and X_i by the vectors and by $n\times r$ matrices. This settles the case c). □

The following result is needed in section 3:

Proposition A.6.

Given a displacement operator ϕ, *four integers* n, s, r *and* R, *such that* $1\le r<R\le n$, $s\ge 1$, *an* $n\times n$ *matrix polynomial* $W = W(\lambda) \bmod \lambda^s$
having ϕ- *rank* r *over a field* F *and a pair of* $R\times n$ *matrices* G_R and H_R,
forming an ϕ- *generator of length* R *for* $W \bmod \lambda^s$, *so that* $\phi(W) = G_RH_R^T \bmod \lambda^s$,
it is possible to compute, for the cost $O_A(R\log s, nR\,p_F(s))$, *an* ϕ- *generator* G_r, H_r of *length* r, *for* $W \bmod \lambda^s$,
such that $G_rH_r^T = W \bmod \lambda^s$.

Proof. First apply the Gauss elimination process with pivoting in order to factorize $G_R \bmod \lambda^s$. In each elimination stage k, k=1,...,R, check if all the entries of column k vanish, and if so, remove this column and append the null column vector at the last, n-th position in the matrix. Otherwise, among all the diagonal and subdiagonal entries of the column k, choose one, (i,k), having nonzero term of the lowest degree and move this entry to the pivot position (k,k), by interchanging rows i and k. Let $G^{(k)} = [g_{i,j}^{(k)}(\lambda)]$ denote the matrix polynomial entering the k-th elimination stage after such a row interchange. Let us denote

$$g_{i,j}^{(k)}(\lambda) = \sum_{u=u(i,j,k)}^{s-1} g_{i,j,u}^{(k)}\lambda^u,\ g_{i,j,u(i,j,k)}^{(k)} \neq 0,\ u(i,j,k) \geq 0,$$

so that $u(k,k,k) = \min_{k \leq i \leq n} u(i,k,k)$.

Now, to perform the k-th elimination stage, first compute, for the cost $O_A(\log s,\ s \log\log s)$, over the fields F that support FFT, and $O_A(\log s,\ s^2 \log\log s)$, over other fields, the polynomial $h_{k,k}^{(k)}(\lambda) = (\lambda^{u(k,k,k)}/g_{k,k}^{(k)}(\lambda)) \bmod \lambda^s$, which is the reciprocal modulo λ^s of the polynomial $g_{k,k}^{(k)}(\lambda)/\lambda^{u(k,k,k)}$. [This polynomial has the λ-free term $g_{k,k,u(k,k,k)}^{(k)} \neq 0$.]

To support (and actually, to improve) the cost bounds $O_A(\log s,\ s \log\log s)$, provided that the field F supports discrete Fourier transform at O(s) points for the cost $O_A(\log s,\ s)$, we just apply the algorithms of p889 p928

Over any F, we may perform DFT at k = O(s) points for the cost bounded by $O_A(\log s,\ s)$ in an extension $\bar{F}$ of F, such that every operation in $\tilde{F}$ involved in DFT is reduced either to addition/subtraction of two polynomials in x modulo a polynomial of degree k = O(s) or to their multiplication by some power x^i, $i \leq k$ [both operations have cost $O_A(1,\ s)$], thus implying the overall cost bound $O_A(\log s,\ s^2 \log\log s)$ (see p869 (This bound can be further improved, but here it suffices for us as it is.) Then compute, for the cost $O_A(\log s,\ np_F(s))$, the n−k polynomials $h_{i,k}^{(k)}(\lambda) = g_{i,k}^{(k)}(\lambda)\, h_{k,k}^{(k)}(\lambda) \bmod \lambda^s$, for all i from k+1 to n. Then, for all i > k, multiply modulo λ^s the k-th row of $G^{(k)}$ by $h_{i,k}^{(k)}(\lambda)$ and subtract the resulting row from the i-th row of $G^{(k)}$. By recursively applying this process, for k=1,...,R, for the overall cost $O_A(R \log s,\ nRp_F(s))$, we factorize the matrix polynomial G_R as follows:

$$G_R = P^* L^* U^* \bmod \lambda^s$$

where U^* is an R×R upper triangular matrix polynomial, P^* is an n×n permutation matrix, and L^* is an n×R unit lower triangular matrix polynomial, that is, all its diagonal entries equal to 1 and all its superdiagonal entries vanish.

We now similarly represent H_R as $H_R = \tilde{P}\tilde{L}\tilde{U} \bmod \lambda^s$, so that

$$\phi(W) = G_R H_R^T \bmod \lambda^s = P^* L^* U^* \tilde{U}^T \tilde{L}^T \tilde{P}^T \bmod \lambda^s \qquad (A.3)$$

where $\tilde{L}$ is an n×R unit lower triangular matrix polynomial, $\tilde{U}$ is an R×R upper triangular matrix polynomial, and $\tilde{P}$ is an n×n permutation matrix.

We next compute the R×R matrix polynomial $U^*\tilde{U}^T$ and reduce it to Smith's normal form, $U^*\tilde{U}^T = \hat{P}\hat{M}\, D\, M^T P^T$ where $\hat{M}$ and M^T are unit triangular matrix polynomials, $\hat{P}$ and P are permutation matrices, and D is a diagonal matrix polynomial. Due to the uniqueness property of Smith's normal forms of $U^* \tilde{U}^T$ and of $\phi(W)$, we have that rank $D = \text{rank}(U^* \tilde{U}^T \bmod \lambda^s) = \text{rank}(\phi(W) \bmod \lambda^s) = r$, and since D is a diagonal matrix polynomial, it ought to have exactly r nonzero entries. Deleting the zero rows and columns of D, together with the corresponding columns of the matrix polynomials $P^* L^* \hat{P}\hat{M}$ and $(M^T P^T \tilde{L}^T \tilde{P}^T)^T$, we turn these matrix polynomials, as well as D, into the matrix polynomials G and $\tilde{H}$, of size $n \times r$, and $\tilde{D}$ of size $r \times r$, respectively, so that

$$\phi(W) = G\, \tilde{D}\, \tilde{H}^T \bmod \lambda^s,$$

which defines the desired ϕ-generator G, $H = \tilde{H}\, \tilde{D}$ of length r, for W mod λ^s. It remains to observe that the cost of the computation of G and H is dominated by the cost of computing the factorization (A.3). □

Finally, we will recall the following extension of proposition 2.1, due to p605 (see also p911 p912 and which we cited in remark 2.1.

Proposition A.7.

Let $A = [a_{ij}]$ *be an* $n \times n$ *nonsingular Toeplitz matrix,* $a_{ij} = a_{i-j}$, $i,j = 0,\ldots,n-1$; $\mathbf{a} = [b, a_{1-n}, a_{2-n}, \ldots, a_{-1}]^T$, *for a fixed scalar* b; $\mathbf{y} = [y_0, \ldots, y_{n-1}]^T = A^{-1}\mathbf{a}$; $\mathbf{x} = A^{-1}[1,0,\ldots,0]^T$; $\mathbf{u} = [-1, y_{n-1}, \ldots, y_1]^T$; $\mathbf{v} = ZJ\mathbf{x}$. *Then*

$$A^{-1} = L(\mathbf{y})\, L^T(\mathbf{v}) - L(\mathbf{x})\, L^T(\mathbf{u}).$$

Appendix B.

Let us extend the well-known expression $B^{-1} = -\sum_{i=1}^{n} (c_i / c_0)\, B^{i-1}$, $c_n = 1$, for the inverse of a nonsingular matrix, to the case of the Moore-Penrose generalized inverse B^+. It suffices to consider the symmetric (or Hermitian) case since $B^+ = (B^T B)^+ B^+$ and $\begin{bmatrix} O & B^T \\ B & O \end{bmatrix}^+ = \begin{bmatrix} O & B^+ \\ (B^T)^+ & O \end{bmatrix}$.

Proposition B.1 p880

Let $c(\lambda) = \det(\lambda I - B) = \sum_{i=n-r}^{n} c_i \lambda^i$, $c_{n-r} \neq 0$, *for an* $n \times n$ *Hermitian matrix B. Then*

$$c_{n-r}\, B^+\, B = -\sum_{i=n-r+1}^{n} c_i B^{i-n+r}, \qquad \text{(B.1)}$$

$$c_{n-r} B^+ = -c_{n-r+1}\, B^+\, B - \sum_{i=n-r+1}^{n-1} c_{i+1}\, B^{i-n+r} \qquad \text{(B.2)}$$

$$= \sum_{i=n-r+1}^{n-1} ((c_{n-r+1}/c_{n-r})\, c_i - c_{i+1})\, B^{i-n+r} + (c_{n-r+1}/c_{n-r}) B^r.$$

Multiplying the equations (B.1) and (B.2) by a vector **v** , we arrive at similar expressions for $B^{+}\mathbf{v}$.

$LIST$