Nearly Optimal Computations with Structured Matrices

Victor Y. Pan
Depts. of Mathematics and Computer Science
Lehman College and Graduate Center
of the City University of New York
Bronx, NY 10468 USA
victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

Elias P. Tsigaridas
PolSys Project
INRIA, Paris-Rocquencourt Center
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
Paris, France
elias.tsigaridas@inria.fr

ABSTRACT

We estimate the Boolean complexity of multiplication of structured matrices by a vector and the solution of nonsingular linear systems of equations with these matrices. We study four basic most popular classes, that is, Toeplitz, Hankel, Cauchy and Vandermonde matrices, for which the cited computational problems are equivalent to the task of polynomial multiplication and division and polynomial and rational multipoint evaluation and interpolation. The Boolean cost estimates for the latter problems have been obtained by Kirrinnis in [11], except for rational interpolation, which we supply now. All known Boolean cost estimates for these problems rely on using Kronecker product. This implies the d-fold precision increase for the d-th degree output, but we avoid such an increase by relying on distinct techniques based on employing FFT. Furthermore we simplify the analysis and make it more transparent by combining the representation of our tasks and algorithms in terms of both structured matrices and polynomials and rational functions. This also enables further extensions of our estimates to cover Trummer's important problem and computations with the popular classes of structured matrices that generalize the four cited basic matrix classes.

1. INTRODUCTION

Table 1 displays four classes of most popular structured matrices, which are omnipresent in modern computations for Sciences, Engineering, and Signal and Image Processing. These basic classes have been naturally extended to the four larger classes of matrices, $\mathcal{T}$, $\mathcal{H}$, $\mathcal{V}$, and $\mathcal{C}$, that have structures of Toeplitz, Hankel, Vandermonde and Cauchy types, respectively. They include many other important classes of structured matrices such as the products and inverses of the matrices of these four basic classes, as well as the companion, Sylvester, subresultant, Loewner, and Pick matrices. All these matrices can be readily expressed via their displacements of small ranks [16, Chapter 4], which implies their further attractive properties:

- Compressed representation of matrices as well as their products and inverses through a small number of parameters.
- Multiplication by a vector in nearly linear arithmetic time.
- Solution of nonsingular linear systems of equations with these matrices in quadratic or nearly linear arithmetic time.

These properties enable efficient computations, closely linked and frequently equivalent to fundamental computations with polynomials and rational polynomial functions, in particular to the multiplication, division, multipoint evaluation and interpolation [17]. Low arithmetic cost is surely attractive, but substantial growth of the computational precision quite frequently affects the known algorithms having low arithmetic cost (see, e.g., [5]). So the estimation of the complexity under the Boolean model is more informative, although technically more demanding.

To the best of our knowledge, the first Boolean complexity bounds for multipoint evaluation are due to Ritzmann [19]. We also wish to cite the papers [25] and [13], although their results have been superceded in the advanced work of 1998 by Kirrinnis, [11], apparently still not sufficiently well known. Namely in the process of studying approximate partial fraction decomposition he has estimated the Boolean complexity of the multipoint evaluation, interpolation, and the summation of rational functions. He required the input polynomials to be normalized, but actually this was not restrictive at all. We generalize his estimates. For simplicity we assume the evaluation at the points of small magnitude, but our estimates can be rather easily extended to the case of general input. Kirrinnis' study as well as all previous estimates of the Boolean complexity of these computational problems rely on multiplying polynomials as integers, by using Kronecker's product, aka binary segmentation, as proposed in [8]. This implies the d-fold increase of the computational precision for the d-th degree output. The results that we present rely on FFT algorithms for multiplying univariate polynomials and avoid this precision growth.

We represent our FFT-based estimates and algorithms in terms of operations with both structured matrices and polynomial and rational functions. In both representations the computational tasks and the solution algorithms are equivalent, and so the results of [11] for partial fraction decomposition can be extended to most although not all of these tasks. By using both representations, however, we make our analysis more transparent. Furthermore in Section 7 we extend

Table 1. Four classes of structured matrices

.

Toeplitz matrices $T = (t_{i-j})_{i,j=0}^{n-1}$	Hankel matrices $H = (h_{i+j})_{i,j=0}^{n-1}$
$\begin{pmatrix} t_0 & t_{-1} & \cdots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{pmatrix}$	$\begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \iddots & h_n \\ \vdots & \iddots & \iddots & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix}$
Vandermonde matrices $V = V_{\mathbf{s}} = \left(s_i^j\right)_{i,j=0}^{n-1}$	Cauchy matrices $C = C_{\mathbf{s},\mathbf{t}} = \left(\frac{1}{s_i - t_j}\right)_{i,j=0}^{n-1}$
$\begin{pmatrix} 1 & s_1 & \cdots & s_1^{n-1} \\ 1 & s_2 & \cdots & s_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & s_n & \cdots & s_n^{n-1} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{s_1-t_1} & \cdots & \frac{1}{s_1-t_n} \\ \frac{1}{s_2-t_1} & \cdots & \frac{1}{s_2-t_n} \\ \vdots & & \vdots \\ \frac{1}{s_n-t_1} & \cdots & \frac{1}{s_n-t_n} \end{pmatrix}$

Kirrinnis' results to the solution of a Cauchy linear system of equations (which unlike [11] covers rational interpolation) and in Section 7.2 to the solution of Trummer's celebrated problem [9], [10], [6], having important applications to mechanics (e.g., to particle simulation) and representing the secular equation, which is the basis for the MPSolve, the most efficient package of subroutines for polynomial root-finding [3].

Our estimates cover multiplication of the matrices of the four basic classes of Table 1 by a vector and solving Vandermonde and Cauchy linear systems of equations. (These tasks are equivalent to the listed tasks of the multiplication, division, multipoint evaluation and interpolation of polynomials and rational functions.) Expressing the solution of these problems in terms of matrices has a major advantage: it can be extended to matrices from the four larger matrix classes $\mathcal{T}$, $\mathcal{H}$, $\mathcal{V}$, and $\mathcal{C}$. Actually the algorithms for multiplication by vector can be extended quite readily, as we explain in Section 7. There we also briefly discuss the solution of linear systems of equations with the matrices of the cited classes, which can be a natural subject of our further study.

Notation. In what follows $\mathcal{O}_B$, resp. $\mathcal{O}$, means bit, resp. arithmetic, complexity and $\widetilde{\mathcal{O}}_B$, resp. $\widetilde{\mathcal{O}}$, means that we are ignoring logarithmic factors. "Ops" stands for "arithmetic operations". For a polynomial $A = \sum_{i=0}^{d} a_i\, x^i \in \mathbb{Z}[x]$, $\mathsf{deg}(A) = d$ denotes its degree and $\mathcal{L}(A) = \tau$ the maximum bitsize of its coefficients, including a bit for the sign. For $a \in \mathbb{Q}$, $\mathcal{L}(a) \geq 1$ is the maximum bitsize of the numerator and the denominator. $\mu(\lambda)$ denotes the bit complexity of multiplying two integers of size λ; we have $\mu(\lambda) = \widetilde{\mathcal{O}}_B(\lambda)$. 2^{Γ} is an upper bound on the magnitude of the roots of A. We write $\Delta_\alpha(A)$ or just Δ_α to denote the minimum distance between a root α of a polynomial A and any other root. We call this quantity *local separation bound*. We also write Δ_i instead of Δ_{α_i}. $\Delta(A) = \min_\alpha \Delta_\alpha(A)$ or just Δ denotes the *separation bound*, that is the minimum distance between all the roots of A. The Mahler bound (or measure) of A is $\mathcal{M}(A) = a_d \prod_{|\alpha|\geq 1} |\alpha|$, where α runs through the complex roots of A, e.g. [14, 26]. If $A \in \mathbb{Z}[x]$ and $\mathcal{L}(A) = \tau$, then $\mathcal{M}(A) \leq \|A\|_2 \leq \sqrt{d+1}\|A\|_\infty = 2^\tau \sqrt{d+1}$. If we evaluate a function F (e.g. $F = A$) at a number c using interval arithmetic, then we denote the resulting interval by $[F(c)]$, provided that we fix the evaluation algorithm and the precision of computing. We write $D(c,r) = \{x : |x-c| \leq r\}$. $\widetilde{f} \in \mathbb{C}[x]$ denotes a λ-approximation to a polynomial $f \in \mathbb{C}[x]$, such that $\|f - \widetilde{f}\|_\infty \leq 2^{-\lambda}$. In particular $\widetilde{a} \in \mathbb{C}$ denotes a λ-approximation to a constant $a \in \mathbb{C}$ such that $|a - \widetilde{a}| \leq 2^{-\lambda}$. lg stands for log.

2. PRELIMINARIES

2.1 Univariate Separation Bounds

The following proposition provides upper and aggregate bounds for the roots of a univariate polynomial. There are various version of these bounds. We use the one presented in [23], to which we also refer the reader for further details and a discussion of the literature. For multivariate separation bounds we refer the reader to [7].

Proposition 1. *Let $f = \sum_{i=0}^{d} a_i x^i \in \mathbb{C}[x]$ be a square-free univariate polynomial of a degree d such that $a_d a_0 \neq 0$. Let Ω be any set of k pairs of indices (i,j) such that $1 \leq i < j \leq d$, let the complex roots of A be $0 < |\gamma_1| \leq |\gamma_2| \leq \cdots \leq |\gamma_d|$, and let $\mathsf{disc}(f)$ be the discriminant of f. Then*

$$\frac{|a_0|}{\|f\|_2} \leq |\gamma_i| \leq \frac{\|f\|_2}{|a_d|}\ , \tag{1}$$

$$\prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \geq 2^{k-d-\frac{d(d-1)}{2}}\, |a_0|^k\, \mathcal{M}(f)^{1-d-k} \sqrt{|\mathsf{disc}(f)|}$$

$$\geq 2^{k-d-\frac{d(d-1)}{2}}\, |a_0|^k\, \|f\|_2^{1-d-k} \sqrt{|\mathsf{disc}(f)|}. \tag{2}$$

If $f \in \mathbb{Z}[x]$ and the maximum coefficient bitsize is τ then

$$2^{-\tau-1} \leq |\gamma| \leq 2^{\tau+1}\ , \tag{3}$$

$$-\lg \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \leq 3d^2 + 3d\tau + 4d\lg d. \tag{4}$$

The following lemma from [24] provides a lower bound on the evaluation of a polynomial that depends on the closest root and on aggregate separation bounds.

Lemma 2. *Suppose $L \in \mathbb{C}$, f is a square-free polynomial, and its root γ_1 is closest to L. Then*

$$|f(L)| \geq |a_d|^7\, |L - \gamma_1|^6\, \mathcal{M}(f)^{-6}\, 2^{\lg \prod_i \Delta_i - 6}\ .$$

2.2 Complex Interval arithmetic

We also need the following bounds for the width of complex intervals when we perform computations with interval arithmetic. We will use them to bound the error when we perform basic computation with complex (floating point) numbers. We refer the reader to [20] for further details.

Proposition 3 (Complex intervals). *Given complex intervals I and J, where $|I|$, resp. $|J|$, denotes the modulus of any complex number in the complex interval I, resp. J. If $2^{-\nu} \leq |I| \leq 2^{\tau}$ and $|J| \leq 2^{\sigma}$, then $\mathtt{wid}(I+J) \leq 2\,\mathtt{wid}(I) + 2\,\mathtt{wid}(J)$, $\mathtt{wid}(I\,J) \leq 2^{\tau+1}\,\mathtt{wid}(J) + 2^{\sigma+1}\,\mathtt{wid}(I)$, and $\mathtt{wid}(1/I) \leq 2^{4\nu+2\tau+3}\,\mathtt{wid}(I)$.*

2.3 Approximate multiplication of two polynomials

We need the following two lemmas from [17] on the evaluation of a polynomial at the powers of a root of unity and on polynomial multiplication. A result similar to the first lemma appeared in [21, Section 3] where Bluestein's technique from [4] is applied (see also [12, Chapter 4.3.3, Exercise 16]). We use that lemma to provide a bound on the Boolean complexity of multiplying two univariate polynomials when their coefficients are known up to a fixed precision. An algorithm for this problem appeared in [21, Theorem 2.2] based on employing Kronecker's product, but instead we rely on FFT and the estimates of Corollary 4.1 from [2, Chapter 3].

Lemma 4. *Suppose $A \in \mathbb{C}[x]$ of a degree at most d such that $\|A\|_\infty \leq 2^{\tau}$. Let $K = 2^k \geq d$ for a positive integer k. Assume that we know the coefficients of A up to the precision $-\ell - \tau - \lg K - 3$; that is the input is assumed to be a polynomial $\widetilde{A}$ such that $\|A - \widetilde{A}\|_\infty \leq 2^{-\ell-\tau-\lg K-3} \geq 10$. Let $\omega = \exp(\frac{2\pi}{K}\sqrt{-1})$ denote a K-th root of unity. Then we can evaluate the polynomial A at $1, \omega, \ldots, \omega^{K-1}$ in $\widetilde{\mathcal{O}}_B(K \lg K\, \mu(\ell + \tau + \lg K))$ such that $\max_{0\leq i \leq K-1}|A(\omega^i) - \widetilde{A(\omega^i)}| \leq 2^{-\ell}$. Moreover, $|A(\omega^i)| \leq K\,\|A\|_\infty \leq 2^{\tau+\lg K}$, for all $0 \leq i \leq K-1$.*

Lemma 5. *Let $A, B \in \mathbb{C}[x]$ of degree at most d, such that $\|A\|_\infty \leq 2^{\tau_1}$ and $\|B\|_\infty \leq 2^{\tau_2}$. Let C denote the product AB and let $K = 2^k \geq 2d+1$ for a positive integer k. Write $\lambda = \ell + 2\tau_1 + 2\tau_2 + 5.1 \lg K + 4$. Assume that we know the coefficients of A and B up to the precision λ, that is that the input includes two polynomials $\widetilde{A}$ and $\widetilde{B}$ such that $\|A - \widetilde{A}\|_\infty \leq 2^{-\lambda}$ and $\|B - \widetilde{B}\|_\infty \leq 2^{-\lambda}$. Then we can compute in $\mathcal{O}_B(d \lg d\, \mu(\ell + \tau_1 + \tau_2 + \lg d))$ a polynomial $\widetilde{C}$ such that $\|C - \widetilde{C}\|_\infty \leq 2^{-\ell}$. Moreover, $\|C\|_\infty \leq 2^{\tau_1+\tau_2+2\lg K}$ for all i.*

Remark 6. *In the sequel, for simplicity we occasionally replace the value $\lambda = \ell + 2\tau_1 + 2\tau_2 + 5.1 \lg(2d+1) + 4$ by its simple upper bound $\ell + 2\tau_1 + 2\tau_2 + 6 \lg d + 15$.*

3. APPROXIMATE FFT-BASED POLYNOMIAL DIVISION

In this section we present an efficient algorithm and its complexity analysis for dividing univariate polynomials approximately. This result is the main ingredient of the fast algorithms for multipoint evaluation and interpolation. The evaluation is involved into our record fast real root-refinement, but all these results are also interesting on their own right because, unlike the previous papers such as [21], [22] and [11], we keep the Boolean cost bounds of these computations at the record level by employing FFT rather than the Kronecker product and thus decreasing the precision of computing dramatically.

Assume two polynomials $s(x) = \sum_{i=0}^{m} s_i x^i$ and $t(x) = \sum_{i=0}^{n} t_i x^i$ such that $s_m t_n \neq 0$, $m \geq n$, and seek the quotient $q(x) = \sum_{i=0}^{m-n} q_i x^i$ and the remainder $r(x) = \sum_{i=0}^{n-1} r_i x^i$ of their division such that $s(x) = t(x)\,q(x) + r(x)$ and $\deg(r) < \deg(t)$. Further assume that $t_n = 1$. This is no loss of generality because we can divide the polynomial t by its nonzero leading coefficient. We narrow our task to computing the quotient $q(x)$ because as soon as the quotient is available, we can compute the remainder $r(x) = s(x) - t(x)\,q(x)$ at the dominated cost by multiplying $t(x)$ by $q(x)$ and subtracting the result from $s(x)$.

The complexity analysis that we present relies on root bounds of $t(x)$, contrary to [17] where it relies on bounds on the infinity norm of $t(x)$. To keep the presentation self-contained we copy from [17] the matrix representation of the algorithm, which occupies the next two pages, up to to Lemma 9.

We begin with an algorithm for the exact evaluation of the quotient. Represent division with a remainder by the vector equation

$$\begin{bmatrix} 1 & & & \\ t_{n-1} & 1 & & \\ \vdots & \vdots & & \\ t_1 & & & \\ t_0 & t_1 & \cdots & 1 \\ & t_0 & t_1 & \vdots \\ & & & t_1 \\ & & & t_0 \end{bmatrix} \begin{bmatrix} q_{m-n} \\ q_{m-n-1} \\ \vdots \\ q_1 \\ q_0 \end{bmatrix} + \begin{bmatrix} \\ \\ \\ r_{n-1} \\ r_{n-2} \\ \vdots \\ r_0 \end{bmatrix} = \begin{bmatrix} s_m \\ s_{m-1} \\ \vdots \\ s_n \\ s_{n-1} \\ \vdots \\ s_0 \end{bmatrix}.$$

The first $m - n + 1$ equations form the following vector equation,

$$\begin{bmatrix} 1 & & & \\ t_{n-1} & 1 & & \\ \vdots & \vdots & & \\ t_1 & & & \\ t_0 & t_1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} q_{m-n} \\ q_{m-n-1} \\ \vdots \\ q_1 \\ q_0 \end{bmatrix} = \begin{bmatrix} s_m \\ s_{m-1} \\ \vdots \\ s_{n+1} \\ s_n \\ \vdots \\ s_{m-n-1} \end{bmatrix} \Leftrightarrow T\,\mathbf{q} = \mathbf{s}, \tag{5}$$

where $\mathbf{q} = (q_i)_{i=0}^{m-n}$, $\mathbf{s} = (s_i)_{i=m-n+1}^{m}$, and T is the nonsingular lower triangular Toeplitz matrix, defined by its first column vector $\mathbf{t} = (t_i)_{i=0}^{n})$, $t_n = 1$. Write $T = Z(\mathbf{t})$ and $Z = Z(\mathbf{e}_2)$ where $\mathbf{e}_2 = (0, 1, 0 \ldots, 0)^T$ is the second coordinate vector, and express the matrix T as a polynomial in a generator matrix $Z = Z_{n+1}$ of size $(n+1) \times (n+1)$ as follows,

$$Z = \begin{pmatrix} 0 & & \cdots & & 0 \\ 1 & \ddots & & & \\ \vdots & \ddots & \ddots & & \vdots \\ & & \ddots & 0 & \\ 0 & & \cdots & 1 & 0 \end{pmatrix}, \quad T = Z(\mathbf{t}) = t(Z) = \sum_{i=0}^{n} t_i Z^i, \quad Z^{n+1} = O.$$

The matrix T is nonsingular because $t_n \neq 0$, and the latter equations imply that the inverse matrix $T^{-1} = t(Z)^{-1} \bmod Z^{n+1}$ is again a polynomial in Z, that is again a lower triangular Toeplitz matrix defined by its first column. We compute this column by applying a divide and conquer algorithm. Assume that $n+1=\gamma=2^k$ is a power of two, for a positive integer k. If this is not the case, embed the matrix T into a lower triangular Toeplitz $\gamma\times\gamma$ matrix $\bar{t}(Z_\gamma)$ for $\gamma=2^k$ and $k=\lceil\lg(n+1)\rceil$ with the leading (that is northwestern) block $T=t(Z_\gamma)$, such that $t(Z_\gamma)=\bar{t}(Z_\gamma) \bmod Z_\gamma^{n+1}$, compute the inverse matrix and output its leading $(n+1)\times(n+1)$ block T^{-1}.

Now represent T as the 2×2 block matrix, $T=\begin{bmatrix} T_0 & O \\ T_1 & T_0\end{bmatrix}$ where T_0 and T_1 are $\frac{\gamma}{2}\times\frac{\gamma}{2}$ Toeplitz submatrices of the Toeplitz matrix T, T_0 is invertible, and observe that

$$T^{-1}=\begin{bmatrix} T_0 & O \\ T_1 & T_0\end{bmatrix}^{-1}=\begin{bmatrix} T_0^{-1} & O \\ -T_0^{-1}\,T_1\,T_0^{-1} & T_0^{-1}\end{bmatrix}\ . \qquad (6)$$

We only seek the first column of the matrix T^{-1}. Its computation amounts to solving the same problem for the half-size triangular Toeplitz matrix T_0 and to multiplication of each of the $\frac{\gamma}{2}\times\frac{\gamma}{2}$ Toeplitz matrices T_1 and T_0^{-1} by a vector. Let $TTI(s)$ and $TM(s)$ denote the arithmetic cost of $s\times s$ triangular Toeplitz matrix inversion and multiplying an $s\times s$ Toeplitz matrix by a vector, respectively. Then the above analysis implies that $TTI(\gamma)\le TTI(\gamma/2)+2TM(\gamma/2)$. Recursively apply this bound to $TTI(\gamma/2^g)$ for $g=1,2,\ldots$, and deduce that $TTI(\gamma)\le\sum_{g=1}^{h}TM(\gamma/2^g)$. The following simple lemma (cf. [16, equations (2.4.3) and (2.4.4)]) reduce Toeplitz-by-vector multiplication to polynomial multiplication and the extraction of a subvector of the coefficient vector of the product, thus implying that $TM(s)\le cs\lg s$ for a constant c and consequently $TTI(\gamma)<2c\gamma\lg\gamma$.

Lemma 7. *The vector equation*

$$\begin{pmatrix} u_0 & & O \\ \vdots & \ddots & \\ \vdots & \ddots & u_0 \\ u_m & \ddots & \vdots \\ & \ddots & \vdots \\ O & & u_m \end{pmatrix}\begin{pmatrix} v_0 \\ \vdots \\ v_n\end{pmatrix}=\begin{pmatrix} p_0 \\ \vdots \\ \vdots \\ p_m \\ \vdots \\ p_{m+n}\end{pmatrix} \qquad (7)$$

is equivalent to the polynomial equation

$$\left(\sum_{i=0}^{m}u_ix^i\right)\left(\sum_{i=0}^{n}v_ix^i\right)=\sum_{i=0}^{m+n}p_ix^i. \qquad (8)$$

We wish to estimate the Boolean (rather than arithmetic) cost of inverting a triangular Toeplitz matrix T and then extend this to the Boolean cost bound of computing the vector $T^{-1}\mathbf{s}$ and of polynomial division. So next we assume that the input polynomials are known up to some precision $2^{-\lambda}$ and employ the above reduction of the problem to recursive (approximate) polynomial multiplications.

To study the Boolean complexity of this procedure, we need the following corollary, which is a direct consequence of Lemma 5 and the inequality $\lg(2d+1)\le 2+\lg d$.

Corollary 8 (Bounds for the product $P_0^2\,P_1$). *Let a polynomial $P_0\in\mathbb{C}[x]$ have a degree d, let its coefficients be known up to a precision $2^{-\nu}$, and let $\|P_0\|_\infty\le 2^{\tau_0}$. Similarly, let $P_1\in\mathbb{C}[x]$ have the degree $2d$, let its coefficients be known up to a precision $2^{-\nu}$, and let $\|P_1\|_\infty\le 2^{\tau_1}$. Then the polynomial $P=P_0^2P_1$ has degree $4d$, its coefficients are known up to the precision $2^{-\nu+8\tau_0+2\tau_1+15\lg d+40}$, and $\|P_0^2P_1\|_\infty\le 2^{2\tau_0+\tau_1+6\lg d+8}$.*

The following lemma is a normalized version of Lemma 4.4 in [11].

Lemma 9. *Let $F,G\in\mathbb{C}[x]$ such that $\deg(F)=m\ge n=\deg(G)\ge 1$, let 2^ρ be an upper bound on the magnitude of roots of G, and let $F=GQ+R$ with $\deg(Q)=m-n$ and $\deg(R)=n-1$. Then*

$$\|Q\|_\infty\le 2^{m+\lg m+m\rho}\|F\|_\infty \ \textit{ and } \ \|R\|_\infty\le 2^{m+n+\lg m+m\rho}\|F\|_\infty\ .$$

Proof: To bring the roots inside the unit circle, transform the polynomials by scaling the variable x as follows, $f(x)=F(x\,2^\rho)$, $g(x)=G(x\,2^\rho)$, $q(x)=Q(x\,2^\rho)$, and $r(x)=R(x\,2^\rho)$. Now apply [11, Lemma 4.4] to the equation $f=gq+r$ to obtain $\|q\|_\infty\le\|q\|_1\le 2^{m-1}\|f\|_1\le 2^{m+\lg m}\|f\|_\infty$ and $\|r\|_\infty\le\|r\|_1\le\frac{3}{4}2^{m+n}\|f\|_1\le 2^{m+n+\lg m}\|f\|_\infty$.
Combine these inequalities with the equation $\|f\|_\infty=2^{m\rho}\|F\|_\infty$ and the inequalities $\|Q\|_\infty\le\|q\|_\infty$ and $\|R\|_\infty\le\|r\|_\infty$ to deduce the claimed bounds. $\square$

We will estimate by induction the cost of inverting the matrix T, by using Eq. (6) recursively. The proof of the following lemma could be found in the Appendix.

Lemma 10. *Let $n+1=2^k$ for a positive integer k and let T be a lower triangular Toeplitz $(n+1)\times(n+1)$ matrix of. Eq. (5), having ones on the diagonal. Let its subdiagonal entries be complex numbers of magnitude at most 2^τ known up to a precision $2^{-\lambda}$. Let 2^ρ be an upper bounds on the magnitude of the roots of the univariate polynomial $t(x)$ associated with T. Write $T^{-1}=(T_{i,j}^{-1})_{i,j=0}^{n}$. Then*

$$\max_{i,j}|T_{i,j}^{-1}|\le 2^{(\rho+1)n+\lg(n)+1}\ .$$

Furthermore, to compute the entries of T^{-1} up to the precision of ℓ bits, that is to compute a matrix $\widetilde{T}^{-1}=(\widetilde{T}_{i,j}^{-1})_{i,j=0}^{n}$ such that $\max_{i,j}|T_{i,j}^{-1}-\widetilde{T}_{i,j}^{-1}|\le 2^{-\ell}$, it is sufficient to know the entries of T up to the precision of
$\ell+10\tau\lg n+70\lg^2 n+8(\rho+1)n\lg n$
or $\mathcal{O}(\ell+(\tau+\lg n+n\rho)\lg n)=\widetilde{\mathcal{O}}(\ell+\tau+n\rho)$ bits.
The computation of $\widetilde{T}^{-1}$ costs $\mathcal{O}_B(n\lg^2(n)\,\mu(\ell+(\tau+\lg n+n\rho)\lg n))$ or $\widetilde{\mathcal{O}}_B(n\ell+n\tau+n^2\rho)$.

As usual in estimating the complexity of approximate division we assume that $m=2n$ to simplify our presentation. Recall that $s(x)=t(x)\,q(x)+r(x)$.

Theorem 11. *Assume $s,t\in\mathbb{C}[x]$ of degree at most $2n$ and n, such that $\|s\|_\infty\le 2^{\tau_1}$, $\|t\|_\infty\le 2^{\tau_2}$, and 2^ρ is an upper bound on the magnitude of the coefficients of $t(x)$. Assume that we know the coefficients of s and t up to a precision λ, that is that the input includes two polynomials $\widetilde{s}$ and $\widetilde{t}$ such that $\|s-\widetilde{s}\|_\infty\le 2^{-\lambda}$ and $\|t-\widetilde{t}\|_\infty\le 2^{-\lambda}$, where $\lambda=\ell+\tau_1+12\tau_2\lg n+80\lg^2 n+10(\rho+1)n\lg n+30$ or $\lambda=\mathcal{O}(\ell+\tau_1+\tau_2\lg n+n\,\rho\,\lg n)$. Let q denote the quotient and let r denote remainder of the division of the polynomials s by t, that is $s=t\cdot q+r$ where $\deg r<\deg t$.*
Then we can compute in $\mathcal{O}_B(n\lg^2(n)\,\mu(\ell+\tau_1+(\tau_2+n\rho)\lg n))$ or $\widetilde{\mathcal{O}}_B(n\ell+n\tau_1+n\tau_2+n^2\rho)$ two polynomials $\widetilde{q}$ and $\widetilde{r}$ such that $\|q-\widetilde{q}\|_\infty\le 2^{-\ell}$ and $\|r-\widetilde{r}\|_\infty\le 2^{-\ell}$, $\|q\|_\infty\le 2^{n+\lg n+1+n\rho+\tau_1}$ and $\|r\|_\infty\le 2^{3n+\lg n+1+n\rho+\tau_1}$.

Proof: We compute the coefficients of $q(x)$ using Eq. (5), ie $q = T^{-1}s$. Each coefficient of the polynomial q comes as the inner product of two vectors, ie $q_i = \sum_{j=0}^{n} T^{-1}_{i,j} s_j$.

From Lemma 10 we know that $\lg|T^{-1}_{i,j}| \leq n(\rho+1) + \lg n + 1 = N$ and $\lg|T^{-1}_{i,j} - \widetilde{T}^{-1}_{i,j}| \leq -\lambda + l_2$ for $l_2 = 10\tau_2 \lg n + 70\lg^2 n + 8(\rho+1)n\lg n$.

For the coefficients of the polynomials $s = \sum_{j=0}^{2n} s_j x^j$ and $t = \sum_{j=0}^{n} t_j x^j$, we have assumed the following bounds, $\lg|s_j| \leq \tau_1$, $\lg|s_j - \widetilde{s}_j| \leq -\lambda$, $\lg|t_j| \leq \tau_2$, $\lg|t_j - \widetilde{t}_j| \leq -\lambda$, $\lg|T^{-1}_{i,j}s_j| \leq \tau_1 + N$, and $\lg|T^{-1}_{i,j}s_j - \widetilde{T}^{-1}_{i,j}\widetilde{s}_j| \leq -\lambda + \ell_2 + \tau_1$ for all i and j. Therefore

$$\begin{aligned}\lg\|q - \widetilde{q}\|_\infty &\leq \lg|\sum_j T^{-1}_{i,j}s_j - \sum_j \widetilde{T}^{-1}_{i,j}\widetilde{s}_j| \leq -\lambda + \ell_2 + \tau_1 + \lg n \\ &\leq -\lambda + 10\tau_2\lg n + 70\lg^2 n + 8(\rho+1)n\lg n + \tau_1 + \lg n\end{aligned}$$

To compute the remainder we apply the formula $r(x) = s(x) - t(x)q(x)$. It involves an approximate polynomial multiplication and a subtraction. For the former we use Lemma 5 and obtain the inequality $\lg\|t\,q - \widetilde{t}\widetilde{q}\|_\infty \leq -\lambda + 2\tau_2 + 6\lg n + 26 + \ell_2 + 2N$.

Let us also cover the impact of the subtraction. After some calculations and simplifications that make the bounds less scary (albeit less accurate wrt the constant involved), we obtain

$$\begin{aligned}\lg\|r - \widetilde{r}\|_\infty &\leq -\lambda + \tau_1 + 2\tau_2 + 6\lg n + 26 + \ell_2 + 2N \\ &\leq -\lambda + \tau_1 + 2\tau_2 + 6\lg n + 26 + 10\tau_2\lg n \\ &\quad + 70\lg^2 n + 8(\rho+1)n\lg n + 2(n(\rho+1) + \lg n + 1) \\ &\leq -\lambda + \tau_1 + 12\tau_2\lg n + 80\lg^2 n + 10(\rho+1)n\lg n + 30\ .\end{aligned}$$

By using Lemma 9 we bound the norms of the quotient and the remainder as follows: $\lg\|r\|_\infty \leq 3n + \lg n + 1 + n\rho + \tau_1$ and $\lg\|q\|_\infty \leq n + \lg n + 1 + n\rho + \tau_1$.

The maximum number of bits that we need to compute with is $\ell + \tau_1 + 12\tau_2\lg n + 80\lg^2 n + 10(\rho+1)n\lg n + 30$ or $\mathcal{O}(\ell + \tau_1 + \tau_2\lg n + \lg^2 n + n\,\rho\,\lg n)$.

The complexity of computing $\widetilde{T}^{-1}_{i,j}$ is $\mathcal{O}_B(n\,\lg^2(n)\,\mu(\ell + \tau_1 + \tau_2\lg n + \lg^2 n + n\,\rho\,\lg n))$ or $\widetilde{\mathcal{O}}_B(n\ell + n\tau_1 + n\tau_2 + n^2\rho)$.

According to Lemma 5 the complexity of computing the product $\widetilde{t}\,\widetilde{q}$ is $\mathcal{O}_B(n\lg(n)\,\mu(\ell + \tau_1 + \tau_2\lg n + \lg^2 n + n\,\rho\,\lg n))$ or $\widetilde{\mathcal{O}}_B(n\ell + n\tau_1 + n\tau_2 + n^2\rho)$. □

Remark 12. *We can eliminate the dependence of the bounds of Theorem 11 on τ_2 by applying Vieta's formulae and the following inequality, $|t_k| \leq \binom{n}{k}(2^\rho)^k \leq 2^{2n+n\rho}$, where t_k is the k-th coefficient of $t(x)$. In this way, after some further simplifications, the required precision is $\ell + \tau_1 + 150(\rho+1)n\lg n$ and the complexity bound becomes $\mathcal{O}_B(n\,\lg^2(n)\,\mu(\ell + \tau_1 + \rho\lg n))$ or $\widetilde{\mathcal{O}}_B(n\ell + n\tau_1 + n^2\rho)$.*

4. MULTIPOINT POLYNOMIAL EVALUATION

Problem 1. Multipoint polynomial evaluation. *Given the coefficients of a polynomial $p(x) = \sum_{i=0}^{n-1} p_i x^i$ and a set of knots $t_0, \ldots, t_{n-1}$, compute the values $r_0 = p(t_0), \ldots, r_{n-1} = p(t_{n-1})$ or equivalently compute the vector $\mathbf{r} = V\mathbf{p}$ where $\mathbf{r} = (r_i)_{i=0}^{n-1}$, $\mathbf{p} = (p_i)_{i=0}^{n-1}$, and $V = (x_i^j)_{i,j=0}^{n-1}$.*

In the case where the knots $t_i = \omega^i$ are the n-th roots of 1 for all i, $\omega = \exp(2\pi\sqrt{-1})/\text{n}$, and $V = \Omega = (\omega^{ij})_{i,j=0}^{n-1}$, Problem 1 turns into the problem of the DFT($\mathbf{v}$) computation.

Solution:. The Moenck–Borodin algorithm of [15] solves Problem 1 in $O(M(n)\log n)$ ops for $M(n)$ in (2.4.1), (2.4.2) based on the two following simple observations.

Fact 13. $p(a) = p(x) \bmod (x-a)$ *for any polynomial $p(x)$ and any scalar a.*

Fact 14. $w(x) \bmod p(x) = (w(x) \bmod (u(x)p(x))) \bmod p(x)$ *for any triple of polynomials $u(x)$, $p(x)$, and $w(x)$.*

Algorithm 4: the Moenck–Borodin algorithm for multipoint polynomial evaluation.

INITIALIZATION:. Write $k = \lceil\log_2 n\rceil$, $m_j^{(0)} = x - x_j$, $j = 0, 1, \ldots, n-1$; $m_j^{(0)} = 1$ for $j = n, \ldots, 2^k - 1$ (that is, pad the set of the moduli $m_j^{(0)} = x - x_j$ with ones, to make up a total of 2^k moduli). Write $r_0^{(k)} = p(x)$.

COMPUTATION:

1. *Fan-in process* (see Figure 1). Compute recursively the "supermoduli" $m_j^{(h+1)} = m_{2j}^{(h)} m_{2j+1}^{(h)}$, $j = 0, 1, \ldots, 2^{k-h} - 1$; $h = 0, 1, \ldots, k-2$.
2. *Fan-out process* (see Figure 2). Compute recursively the remainders $r_j^{(h)} = r_{\lfloor j/2 \rfloor}^{(h+1)} \bmod m_j^{(h)}$, $j = 0, 1, \ldots, \min\{n, \lceil n/2^h \rceil - 1\}$; $h = k-1, k-2, \ldots, 0$.

OUTPUT:. $p(x_i) = r_i^{(0)}$, $i = 0, 1, \ldots, n-1$.

Let us include a brief outline of the analysis of the algorithm (cf. [15]). To prove its *correctness*, first apply Fact 14 recursively to obtain that $r_j^{(h)} = v(x) \bmod m_j^{(h)}$ for all j and h. Now, correctness of the output $p(x_i) = r_i^{(0)}$ follows from Fact 13.

To estimate the *computational cost* of the algorithm, represent its two stages by the same binary tree (see Figures 3.1 and 3.2), whose nodes are the "supermoduli" $m_j^{(h)}$ at the *fan-in* stage 1, but turn into the remainders $r_j^{(h)}$ at the *fan-out* stage 2.

At each level h of the tree, the algorithm computes 2^{k-h} products of pairs of polynomials of degree 2^h at stage 1 and 2^{k-h} remainders of the division of polynomials of degree of at most 2^{h+1} by "supermoduli" of degree 2^h. Each time multiplication/division uses $O(M(2^h))$ ops for $M(n)$ in (2.4.1), (2.4.2). So we use $O(2^{k-h}M(2^h))$ ops at the h-th level and $O(\sum_{h=0}^{k-1} 2^{k-h}M(2^h)) = O(M(2^k)k)$ ops at all levels. Recall that $n \leq 2^k < 2n$ and obtain the claimed bound of $O(M(n)\log n)$ ops. □

Remark 15. *The fan-in computation at stage 1 depends only on the set $\{t_0, \ldots, t_{n-1}\}$ and can be viewed as (cost-free)* preprocessing *if the knot set is fixed and only the polynomial $p(x)$ varies. Similar observations hold for the solution of many other problems in this chapter.*

Remark 16. *Problem 1 and its solution algorithms are immediately extended to the case where we have m points $t_0, \ldots, t_{m-1}$ for $m > n$ or $m < n$. The solution requires $O(E(l)r/l)$ ops provided $l = \min\{m, n\}$, $r = \max\{m, n\}$, and $E(l)$ ops are sufficient for the solution where $n = l$. $E(l) = O(M(l)\log l)$ for a general set $\{t_i\}$ but decreases to*

$O(M(l))$, *where* $t_i = at^{2i} + bt^i + c$ *for fixed scalars* a, b, c, *and* t *and for all* i. *This also leads to a similar improvement of the estimates for the Boolean complexity [1].*

4.1 Boolean complexity estimates

In the following two lemmata we present the bit complexity of the fan-in and the fan-out process. These results are of independent interest. We do not estimate the accuracy needed and the bit complexity bound of the algorithm for multipoint evaluation because in Lemma 21 we cover a more general algorithm. Multipoint evaluation is its special case.

Lemma 17 (Complexity of Fan-in process). *Assume that we are given* n *complex numbers* x_i *known up to a precision* $\lambda = \ell + (4n-4)\tau + 32n - (\lg n + 5)^2 - 7$, *that is* $|x_i - \widetilde{x}_i| \le 2^{-\lambda}$, *and that* $|x_i| \le 2^\tau$ *for a positive integer* τ. *At the cost* $\widetilde{\mathcal{O}}_B(n \lg^2 n\, \mu(\ell + n\tau + \lg n))$ *the Fan-in process of the Moenck–Borodin algorithm approximates the "supermoduli"* $\widetilde{m}_j^{(i)}$ *within the bounds* $\|m_j^{(i)} - \widetilde{m}_j^{(i)}\|_\infty \le 2^{-\ell}$ *for all* i *and* j. *Moreover,* $\lg\|m_j^{(i)}\|_\infty \le n\tau + 8n - 2\lg n - 8$ *for all* i *and* j.

Proof: Assume that $n = 2^k$. The proof is by induction on k. Write $m_i^{(0)} = x - x_i$ and $\widetilde{m}_i^{(0)} = x - \widetilde{x}_i$. Wlog we provide the estimates just in the case where $j = 0$.

Consider the case where $k = 1$. Apply Lemma 5 for $A = m_0^{(0)}$ and $B = m_1^{(0)}$. Verify that $\lg\|m_0^{(1)}\|_\infty \le 2\tau \le 2\tau + 6$ and $\lg\|m_0^{(1)} - \widetilde{m}_0^{(1)}\|_\infty \le -\lambda + 4\tau + 14.2$. This proves the induction basis.

Now assume that the claimed bounds hold for $k-1$, that is $\deg(m_i^{(k-1)}) = 2^{k-1}$, $\lg\|m_i^{(k-1)}\|_\infty \le 2^{k-1}\tau + 2^{k+2} - 2k - 6$, and $\lg\|m_i^{(k-1)} - \widetilde{m}_i^{(k-1)}\|_\infty \le -\lambda + (2^{k+1} - 4)\tau + 2^{k+4} - (k+4)^2 - 7$ for $i \in \{0, 1\}$.

Since $m_0^{(k)} = m_0^{(k-1)}\, m_1^{(k-1)}$, it follows that $\deg(m_0^{(k)}) = 2^k$. By applying Lemma 5 we deduce that

$$\lg\|m_i^{(k)}\|_\infty = \lg\|m_0^{(k-1)}\, m_1^{(k-1)}\|_\infty \le 2^k\tau + 2^{k+3} - 2k - 8$$

and

$$\begin{aligned}\lg\|m_0^{(k)} - \widetilde{m}_0^{(k)}\|_\infty &= \lg\|m_0^{(k-1)}\, m_1^{(k-1)} - \widetilde{m}_0^{(k-1)}\, \widetilde{m}_1^{(k-1)}\|_\infty \\ &\le -\lambda + (2^{k+2} - 4)\tau + 2^{k+5} - (k+5)^2 - 7\end{aligned}$$

as claimed.

To estimate the overall complexity, note that at the hth level of the tree we perform $n/2^h$ multiplications of polynomials of degrees at most 2^{h-1} for $h = 2, \ldots, k-1$. We can assume that we perform all the computation with precision $\mathcal{O}(\ell + n\tau + \lg n)$, and so the overall cost of the algorithm is $\sum_k \frac{n}{2^k}\widetilde{\mathcal{O}}_B(2^{k-1}\lg 2^{k-1}\mu(\ell + n\tau + \lg n)) = \widetilde{\mathcal{O}}_B(n \lg^2 n\, \mu(\ell + n\tau + \lg n))$. □

Lemma 18 (Complexity of Fan-out process). *Let* $v(x) \in \mathbb{C}[x]$ *of degree* $n-1$ *and* $\|v\|_\infty \le 2^{\tau_1}$, *and let* $\widetilde{v}$ *be a* λ-*approximation. Let* $m_j^{(k)}$ *be the supermoduli of the* fan-in *process and* $\widetilde{m}_j^{(k)}$ *their* λ-*approximations.*

We can compute an ℓ-*approximation of the* fan-out *process in* $\mathcal{O}_B(n \lg^2 n\, \mu(\ell + \tau_1 \lg n + \rho\, n \lg n))$ *provided that* $\lambda = \ell + 2\tau_1 \lg n + 300(\rho + 1)n \lg n$.

Proof: We keep assuming for simplicity that $n = 2^k$ and proceed as in the proof of Lemma 17.

Recall that $|x_i| \le 2^\rho$ for all the subscripts i, and so 2^ρ bounds the roots of all polynomials $m_j^{(k)}$. We can prove by induction, by using the bounds of Theorem 11 and the simplifications of Remark 12, that the precision of $\lambda = \ell + 2\tau_1 \lg n + 300(\rho + 1)n \lg n$ bits is sufficient.

At the hth step of the algorithm, for each h, we perform 2^h approximate polynomial divisions of polynomials of degree $\frac{n}{2^h}$ using Theorem 11. We assume performing all the operations with the maximum precision, and bound the overall complexity by

$$\begin{aligned}&\sum_{h=0}^{\lg n} 2^h\, \mathcal{O}_B\Big(\frac{n}{2^k}\,(\lg \frac{n}{2^h})^2\, \mu(\ell + \tau_1 \lg n + \rho\, n \lg n)\Big) \\ &\quad = \mathcal{O}_B(n\, \lg^2 n\, \mu(\ell + \tau_1 \lg n + \rho\, n \lg n))\end{aligned}$$

□

5. BOUNDS ON THE COMPLEXITY OF BASIC ALGORITHMS

Lemma 19 (Multiplication of m polynomials). *Let* $P_j \in \mathbb{C}[x]$ *of degree* n *and* $\|P_j\|_\infty \le 2^\tau$. *Let* $\widetilde{P}_j$ *be* λ-*approximation of* P_j *with* $\lambda = \ell + (4m-4)\tau + (4m + 2\lg m - 4)\lg n + 32m$, *where* $1 \le j \le m$. *We can compute* $\prod_j \widetilde{P}_j$ *such that* $\|\prod_j P_j - \prod_j \widetilde{P}_j\|_\infty \le 2^{-\ell}$ *in* $\mathcal{O}_B(m\, n\, \lg m \lg(m\, n)\, \mu(\lambda))$ *or* $\widetilde{\mathcal{O}}_B(m\, n\, (\ell + m\tau))$. *Moreover,* $\lg\|\prod_j P_j\|_\infty \le m\tau + (m-1)\lg n + 4m - \lg m - 4$.

Proof: The algorithm is similar to the Fan-in process of Moenck-Borodin algorithm. Let $p_j^{(0)} = P_j$ and compute recursively the polynomials $p_j^{(h+1)} = p_{2j}^{(h)} p_{2j+1}^{(h)}$, for $0 \le j \le 2^{k-h}$, $h = 0, \ldots, k-2$. Let $m = 2^h$. We prove the bounds on the infinite norm and the approximation using induction on h.

For $h = 1$, we compute the polynomials $p_j^{(1)} = p_{2j}^{(0)} p_{2j+1}^{(0)}$. Wlog assume that $j = 0$. Then $\lg\|p_j^{(0)}\|_\infty \le \tau$ and $\lg\|p_j^{(0)} - \widetilde{p}_j^{(0)}\|_\infty \le -\lambda$. Apply Lemma 5 (and Remark 6) for $K = n$ and $\tau_1 = \tau_2 = \tau$ to deduce that

$$\lg\|p_0^{(1)}\|_\infty \le 2\tau + \lg n + 3\ ,$$

which agrees with our formula, and

$$\lg\|p_j^{(1)} - \widetilde{p}_j^{(1)}\|_\infty \le -\lambda + 4\tau + 5.1\lg n + 4 \le -\lambda + 4\tau + 6\lg n + 64\ ,$$

where the right hand-side represents the claimed bound for $k = 1$.

Assume the claimed bounds for $h - 1$, that is

$$\lg\|p_j^{(h-1)}\|_\infty \le 2^{h-1}\tau + (2^{h-1} - 1)\lg n + 4 \cdot 2^{h-1} - \lg 2^{h-1} - 4$$

and $\lg\|p_j^{(h-1)} - \widetilde{p}_j^{(h-1)}\|_\infty \le -\lambda + (4 \cdot 2^{h-1} - 4)\tau + (4 \cdot 2^{h-1} + 2\lg 2^{h-1} - 4)\lg n + 32 \cdot 2^{h-1}$ for $j = 0, 1$. By applying Lemma 5 for $2lg(K) \le \lg(n)$, deduce the following bounds,

$$\lg\|p_j^{(h)}\|_\infty \le 2^h\tau + (2^h - 1)\lg n + 4 \cdot 2^h - \lg 2^h - 4,$$

which agrees with the claimed norm bound, and $\lg\|p_j^{(h)} - \widetilde{p}_j^{(h)}\|_\infty \le -\lambda + (4 \cdot 2^h - 4)\tau + (4 \cdot 2^h + 2\lg 2^h - 4)\lg n + 24 \cdot 2^h + 2h - 12$ which is smaller than the claimed bound on the precision.

To estimate the overall complexity note that at each level, h, of the tree we have to perform $m/2^h$ multiplications of polynomials of degrees at most $2^{h-1}n$. We can assume that we perform all the computations with the precision $\lambda + (4m-4)\tau + (4m + 2\lg m - 4)\lg n + 32m$, or $\mathcal{O}(\ell + m\tau + m\lg n)$, and so the overall Boolean cost of performing the algorithm is $\sum_h \frac{m}{2^h}\mathcal{O}_B(2^{h-1}\,n\lg(2^{h-1}\,n)\mu(\ell+n\tau+m\lg n)) = \mathcal{O}_B(m\,n\,\lg m\lg(m\,n)\,\mu(\ell+n\tau+m\lg n))$, which concludes the proof. $\square$

If the degrees of P_j vary as j varies, then we can apply a more pedantic analysis based on Huffman trees, see [11].

The problem of computing (approximately) the sum of rational functions reduces to the problem on multiplying polynomials, which admits the same asymptotic complexity bounds. To estimate the overhead constants, we should also take into account the polynomial additions involved.

We have the following lemma.

Lemma 20 (Sum of rational functions). *Suppose $P_j \in \mathbb{C}[x]$ has degree n, $Q_j \in \mathbb{C}[x]$ has a smaller degree, $\|P_j\|_\infty \le 2^{\tau_2}$, and $\|Q_j\|_\infty \le 2^{\tau_1}$.*

Assume λ-approximations of P_j by $\widetilde{P}_j$ and of Q_j by $\widetilde{Q}_j$ where $\lambda = \ell + \tau_1 + (4m-4)\tau_2 + (5m + 2\lg m - 4)\lg n + 32m$ and $1 \le j \le m$.

Let $\frac{Q}{P} = \sum_j \frac{Q_j}{P_j}$. We can compute an ℓ-approximation of Q/P, in $\mathcal{O}_B(m\,n\,\lg m\lg(m\,n)\,\mu(\lambda))$ or $\widetilde{\mathcal{O}}_B(m\,n\,(\ell + +\tau_1 + m\tau_2))$. Moreover, $\lg\|Q\|_\infty \le \tau_2 + (m-1)(\tau_2 + \lg n) + 5m - \lg m - 4$ and $\lg\|P\|_\infty \le m\,\tau_2 + (m-1)\lg n + 4m - \lg m - 4$.

Lemma 21 (Modular representation). *Let $F \in \mathbb{C}[x]$ of degree $2mn$ and $\|F\|_\infty \le 2^{\tau_1}$. Let $P_j \in \mathbb{C}[x]$ of degree n, for $1 \le j \le m$. Moreover, 2^ρ be an upper bound on the magnitude of the roots of all P_j, for all j. Assume λ-approximations of F by $\widetilde{F}$ and of P_j by $\widetilde{P}_j$ such that $\|F - \widetilde{F}\|_\infty \le 2^{-\lambda}$ and $\|P_j - \widetilde{P}_j\|_\infty \le 2^{-\lambda}$.*

Furthermore assume that $\lambda = \ell + \tau_1\lg m + 60\,n\,m(\rho + 3)\,lg(m\,n) + 60\lg m\lg^2(m+n)$ or $\lambda = \ell + \mathcal{O}(\tau_1\lg m + m\,n\,\rho)$. Then we can compute an ℓ-approximation $\widetilde{F}_j$ of $F_j = F \mod P_j$ such that $\|F_j - \widetilde{F}_j\|_\infty \le 2^{-\ell}$ in

$$\mathcal{O}_B(m\,n\,\lg n\,\lg^2(m+n)\,\mu(\ell + \tau_1\lg m + m\,n\,\rho))$$

or $\widetilde{\mathcal{O}}_B(m\,n\,(\ell + \tau_1 + m\,n\,\rho))$.

Moreover, $\lg\|F_j\|_\infty \le \tau_1 + (\rho+1)\,m\,n + n + \lg(m\,n)$.

Proof: First we perform the Fan-in process with polynomials P_j using the algorithm of Lemma 19. Assume that $\|P_j\|_\infty \le 2^{\tau_2}$ and that we are given λ_1-approximations. Following Lemma 19 we compute all the supermoduli, $P_j^{(i)}$ so that

$$\lg\|P_j^{(i)} - \widetilde{P}_j^{(i)}\|_\infty \le -\lambda_1 + (4m-4)\tau_2 + (4m + 2\lg m - 4)\lg n + 32m\,.$$

Remark 12 implies that $\tau_2 \le 2n + n\rho$, and so $\lg\|P_j^{(i)} - \widetilde{P}_j^{(i)}\|_\infty \le -\lambda + (4m-4)(2n + \lg n + n\rho) + 2\lg m\lg n + 32m = -\lambda + \mathcal{O}(m\,n\,\rho)$.

For computing ℓ-approximations of $F_j = F \mod P_j$ we mimic the procedure of the Fan-out process. This means that we apply repeatedly Theorem 11, which we can refine by following Remark 12. The bounds accumulate at each step, and so

$$\begin{aligned}\lg\|F_j\|_\infty &\le \tau_1 + \textstyle\sum_h 3\,n\,2^h + n + h + 2^h n\rho + 1 \\ &\le (mn - n)(\rho+3) + m(m+n).\end{aligned}$$

We assume that we are given λ_2-approximation of F and all the supermoduli. For the required precision we have

$$\begin{aligned}\lg\|F_j - \widetilde{F}_j\|_\infty &\le -\lambda_2 + \sum_h 25(\rho+2)(h + \lg n)n2^h + 80(h+\lg n)^2 + \tau_1 + 30 \\ &\le -\lambda_2 + \tau_1\lg m + 25\,n\,m(\rho+2)\,lg(m\,n) + 40\lg m\lg^2(m+n).\end{aligned}$$

To ensure an ℓ-approximation for F_j we require $\lambda = \ell + \tau_1\lg m + 60\,n\,m(\rho+3)\,lg(m\,n) + 60\lg m\lg^2(m+n) = \ell + \mathcal{O}(\tau_1\lg m + m\,n\,\rho)$ approximations of the input to ensure the validity of both the Fan-in and Fan-out process.

We assume that we perform all the computations with maximum accuracy. The complexity of computing the supermoduli is $\mathcal{O}_B(m\,n\,\lg m\lg(m\,n)\,\mu(\ell + \tau_1\lg m + m\,n\,\rho))$ or $\widetilde{\mathcal{O}}_B(m\,n\,(\ell + \tau_1 + m\,n\,\rho))$.

For the complexity of the Fan-out process we proceed as follows. At each step, h, of the algorithm we perform 2^h approximate polynomial divisions of polynomials of degree $\frac{m\,n}{2^h}$ using Theorem 11. The overall complexity is
$\sum_{h=0}^{\lg m} 2^h\,\mathcal{O}_B(\frac{mn}{2^h}\,(\lg\frac{mn}{2^h})^2\,\mu(\ell + \tau_1\lg m + m\,n\,\rho))$
$= \mathcal{O}_B(m\,n\,\lg n\,\lg^2(m+n)\,\mu(\ell + \tau_1\lg m + m\,n\,\rho))$ or $\widetilde{\mathcal{O}}_B(m\,n\,(\ell + \tau_1 + m\,n\,\rho))$.

$\square$

6. LAGRANGE INTERPOLATION

Problem 2. Lagrange polynomial interpolation. *Given the knot set (or vector) $\{x_i\}_{i=0}^{n-1}$ of n distinct points $x_0, \ldots, x_{n-1}$ and the set (or vector) of values $\{y_i\}_{i=0}^{n-1}$, compute a set (or vector) $\{a_j\}_{j=0}^{n-1}$ such that $\sum_{j=0}^{n-1} a_j x_i^j = y_i$, $i = 0, 1, \ldots, n-1$, that is, recover the coefficients of a polynomial $A(X) = \sum_{j=0}^{n-1} a_j X^j$ from its values at n distinct points $x_0, \ldots, x_{n-1}$.*

We follow the approach presented in [16, Section 3.3], to which we also refer for a detailed presentation.

Lemma 22. *Let $|x_i| \le 2^{\tau_1}$, $|y_i| \le 2^{\tau_2}$, and $\Delta_i(x) = \min_j |x_i - x_j|$, for all $0 \le i \le n-1$. Assume λ-approximations of x_i and y_i, where $\lambda = \ell + 68n(\tau_1 + 3)\lg n + 4n\tau_2 - 6\lg\prod_i \Delta_i(x) + 50n + 60\lg^3 n + 20$ or $\lambda = \ell + \mathcal{O}(n\tau_1\lg n + n\tau_2 - \lg\prod_i \Delta_i(x) + \lg^3 n)$. Then we can compute an ℓ-approximation of the Lagrange polynomial interpolation in $\mathcal{O}_B(n\,\lg^2 n\,\mu(\lambda))$ or $\widetilde{\mathcal{O}}_B(n^2\tau_1 + n^2\tau_2 - n\lg\prod_i \Delta_i(x)))$.*

Proof: The input is given as λ-approximations, where λ is to be specified in the sequel. We track the loss of accuracy at each step of the algorithm.

1. Compute $B(X) = \prod_i (X - x_i) = X^n + \sum_{k=0}^{n-1} b_k X^k$.

   For this task we apply Lemma 17. In this case the infinite norm of B is bounded as follows, $\lg\|B\|_\infty \le n\tau_1 + 8n - 2\lg n - 4$, and the computed approximation, $\widetilde{B}$, is such that $\lg\|B - \widetilde{B}\|_\infty \le -\lambda_0 + (4n-4)\tau_1 + 16n + 20$. As by-product we can compute the "supermoduli" $\prod_j (x - x_j)$ and then reuse them at stage L5.

2. Compute $B'(X)$.

   This operation increases the norm and the precision bounds by a factor of n in the worst case. That is $\lg\|B'\|_\infty \le n\tau_1 + 8n - \lg n - 4$ and $\lg\|B' - \widetilde{B}'\|_\infty \le -\lambda_0 + (4n-4)\tau_1 + 16n + \lg n + 20 = -\lambda_1$.

3. Evaluate B' at all points x_i.

   Perform this task using Lemma 21. This lemma implies that $\lg|B'(x_i)| \leq (n-1)(\tau_1+3)+n(n+1)$. However, in this special case we can decrease the bound as follows, $|B'(x_i)| \leq \sum_j \|B'\|_\infty |x_i|^{n-1} \leq \sum_j 2^{n\tau_1+8n-\lg n-4} 2^{(n-1)\tau_1} \leq 2^{(2n-1)\tau_1+8n-4}$.

   We achieve the accuracy $\lg|B'(x_i)-\widetilde{B}'(\widetilde{x}_i)| \leq -\lambda_1+(n\tau_1+n-\lg n-1)\lg n+60n(\tau_1+3)\lg n+60\lg^3 n = -\lambda_2$.

4. Consider the rational functions $A_i(X) = \frac{A_{i,0}(X)}{A_{i,1}(X)} = \frac{y_i/B'(x_i)}{(X-x_i)}$.

   Deduce that $\lg\|A_{i,1}\|_\infty \leq \tau_1$, and so the approximation bound matches that of x_i.

   To compute the relevant quantities of the numerator(s) we need a lower bound for $B'(x_i)$, for all i. We notice that $B'(X) = \sum_{i=1}^n \prod_{j\neq i}(X-x_j)$. Thus $B'(x_i) = \prod_{j\neq i}(x_i-x_j)$ and so $|B'(x_i)| \geq \prod_{j\neq i}\Delta_j(x)$, and $\lg\|A_{i,0}\|_\infty \leq \tau_2 - \lg\prod_{j\neq i}\Delta_j(x) \leq \tau_2 - \lg\prod_j \Delta_j(x)$.

   For computing an approximation of the denominator we rely on (complex) interval arithmetic. That is $|A_{i,0} - \widetilde{A}_{i,0}| \leq \mathtt{wid}([A_{i,0}]) = \mathtt{wid}([y_i/B'(x_i)])$.

   We compute $\mathtt{wid}([y_i/B'(x_i)])$ based on Prop. 3, and so $\lg\mathtt{wid}([1/B'(x_i)]) \leq -\lambda_2 - 4\lg\prod_j\Delta_j(x) + 2(2n-1)\tau_2 + 2n - 8 + 3 = -\lambda_3$. Finally

$$\begin{aligned}\mathtt{wid}([y_i/B'(x_i)]) &\leq 2^{\tau_2}\,\mathtt{wid}([1/B'(x_i)]) + 2^{-\lg\prod_j\Delta_j(x)}\,\mathtt{wid}([y_i])\\ &\leq 2^{-\lambda_3+\tau_2-\lg\prod_j\Delta_j(x)} \leq 2^{-\lambda_4}\end{aligned}$$

5. Compute the sum of the rational functions ie $\frac{A_0(X)}{A_1(X)} = \sum_i \frac{A_{i,0}(X)}{A_{i,1}(X)}$.

   Using Lemma 20 we get $\lg\|A_0\|_\infty \leq \tau_2 - \lg\prod_j\Delta_j(x) + (n-1)\tau_1 + 4n - \lg n - 4$ and $\lg\|A_1\|_\infty \leq n\tau_1 + 4n - \lg n - 4$.

   For the approximation we have that $\lg\|A_0 - \widetilde{A}_0\|_\infty \leq -\lambda_4 + \tau_2 - \lg\prod_j\Delta_j(x) + (4n-4)\tau_1 + 32n$. If we substitute the various values for λ_i we have $\lg\|A_0 - \widetilde{A}_0\|_\infty \leq -\lambda + 68n(\tau_1+3)\lg n + 4n\tau_2 - 6\lg\prod_j\Delta_j(x) + 50n + 60\lg^3 n + 20$.

The numerator, A_0, is the required polynomial $A(X)$. To achieve an ℓ-approximation of $A(x)$ we assume that we perform all the computations using the maximum precision, that is $\ell + 68n(\tau_1+3)\lg n + 4n\tau_2 - 6\lg\prod_j\Delta_j(x) + 50n + 60\lg^3 n + 20$ or $\lambda = \ell + \mathcal{O}(n\tau_1\lg n + n\tau_2 - \lg\prod_j\Delta_j(x) + \lg^3 n)$.

The overall complexity is $\mathcal{O}_B(n\lg^2 n\,\mu(\lambda))$ or $\widetilde{\mathcal{O}}_B(n^2\tau_1 + n^2\tau_2 - n\lg\prod_j\Delta_j(x)))$. □

Remark 23 (The hidden costs). *In the previous lemma we have assumed bounds on the minimum distance between the x_i's, which we denote by $\Delta_i(x)$. The complexity results depend on this quantity, as it is very important in the computation of the number of bits that we need to certify the result to a desired accuracy. It is reasonable to assume that such bounds are part of the input.*

However, how do we handle the case where such bounds are not known? As the precision required for the computations depends on these bounds we should be able to compute them, given the points x_i.

We consider the numbers $x_i \in \mathbb{C}$ as points on $\mathbb{R}^2$ and we compute their Voronoi diagram. This costs $\mathcal{O}(n\lg n)$ operations, e.g. [18]. Then for each point x_i we find its closest in $\mathcal{O}(\lg n)$ operations. Therefore, we can compute the quantities $\Delta_i(x)$ in $\mathcal{O}(n\lg n)$ operations. But what about the required precision? What are the required primitive operations for these computations? We only need to evaluate the signs of 3×3 determinants. The precision of Lemma 22 is sufficient for these operations.

7. SOLUTION OF A CAUCHY LINEAR SYSTEM OF EQUATIONS

7.1 Multiplication of a Cauchy matrix by a vector

We consider the problem of computing the matrix vector product $C\mathbf{v}$, where $C = C(s,t) = (\frac{1}{s_i-t_j})_{i,j}^{n-1}$ is a Cauchy matrix and $\mathbf{v} = (v_i)_{i=0}^{n-1}$.

We refer the reader to [16, Problem 3.6.1] for further details of the algorithm.

Let $|s_i| \leq 2^{\tau_1}$, $|t_i| \leq 2^{\tau_2}$, $|v_i| \leq 2^{\tau_3}$, and $\Delta_i(t) = \min_j|t_i - t_j|$, for all i.

The following quantities are also useful $\Delta_j(s,t) = \min_i|s_j - t_i|$ and $\Delta(s,t) = \min_j \Delta_j(s,t)$.

Lemma 24. *If the input is given as a λ-approximation, where $\lambda = \ell + 90n(\tau_1+3)\lg n + 32(n-1)\tau_2\lg n + 30\tau_3\lg n - 35 - 24\lg\Delta(s,t) - 4\lg\prod_k\Delta_k(t)$, then we can compute an ℓ-approximation of the vector $C\mathbf{v}$ in $\mathcal{O}_B(n\lg^2 n\,\mu(\lambda))$ or $\widetilde{\mathcal{O}}_B(n^2\tau_1 + n^2\tau_2 + n\tau_3 - n\lg\Delta(s,t) - n\lg\prod_k\Delta_k(t))$.*

Proof: The input is given as λ-approximations, where λ is to be specified in the sequel. We track the loss of accuracy at each step of the algorithm.

1. Consider the rational functions $\frac{v_k}{x-t_k} = \frac{P_k}{Q_k}$.

   For the numerators it holds $\mathsf{deg}(P_k) = 0$, $\|P_k\|_\infty \leq 2^{\tau_3}$, and $\lg\|P_k - \widetilde{P}_k\|_\infty \leq -\lambda$. For the denominators it holds $\mathsf{deg}(Q_k) = 1$, $\|Q_k\|_\infty \leq 2^{\tau_2}$, and $\lg\|Q_k - \widetilde{P}_k\|_\infty \leq -\lambda$.

2. Compute the sum $\frac{P}{Q} = \sum_k \frac{P_k}{Q_k}$. For this computation we rely on Lemma 20.

   For the numerator of the result we have $\mathsf{deg}(P) \leq n-1$, $\lg\|P\|_\infty \leq \tau_3 + (n-1)\tau_2 + 5n - \lg n - 4$, and $\lg\|P - \widetilde{P}\|_\infty \leq -\lambda + \tau_3 + (4n-4)\tau_2 + 32n = -\lambda_1$.

   For the denominator of the result we have $\mathsf{deg}(Q) \leq n$, $\lg\|Q\|_\infty \leq n\tau_2 + 4n - \lg n - 4$, and $\lg\|Q - \widetilde{Q}\|_\infty \leq -\lambda + \tau_3 + (4n-4)\tau_2 + 32n = -\lambda_1$.

3. Compute $P(s_i)$ and $Q(s_i)$ for all i.

   For this multipoint evaluation we use Lemma 21 (with $m = n$, $n = 1$, $\tau_1 = \lg\|P\|_\infty$, $\rho = \tau_1$).

   We have $\lg|P(s_i)| \leq (\tau_1+1)n + (n-1)\tau_2 + \tau_3 + 4n - 4$ and $\lg|P(s_i) - \widetilde{P}(\widetilde{s}_i)| \leq -\lambda_1 + (\tau_3 + (n-1)\tau_2 + 4n - \lg n - 4)\lg n + 60n(\tau_1+3)\lg n + 60\lg^3 n = -\lambda_2$.

   Similar bounds hold for $Q(s_i)$.

4. Compute the fractions $\frac{P(s_i)}{Q(s_i)}$. These are the elements of the result of the matrix-vector multiplication $C\mathbf{v}$.

   For this task we need to perform n (complex) divisions. We use complex interval arithmetic to compute the loss of precision, as we did for deriving the bounds for Lagrange interpolation. To compute a lower bound for $|Q(s_i)|$ we use Lemma 2, and so

$$\begin{aligned}|Q(s_i)| &\geq (\Delta_i(s,t))^6\ 2^{-6\,\lg\|Q\|_\infty - 6\lg n}\ 2^{\lg\prod_k \Delta_k(t) - 6} \\ &\geq (\Delta_i(s,t))^6\ 2^{\lg\prod_k \Delta_k(t)}\ 2^{-6n\tau_2 - 24n + 20} \\ &\geq 2^{-\nu}\end{aligned}$$

Let $|Q(s_i)| \leq 2^T$, where $T = n\tau_1 + n\tau + 2 + 4n - 4$. Using Prop. 3, $\mathtt{wid}[1/Q(s_i)] \leq 2^{4\nu+2T+3}\, 2^{-\lambda_2} \leq 2^{-\lambda_3}$ and $\mathtt{wid}[P(s_i)/Q(s_i)] \leq 2\,|P(s_i)|\, 2^{-\lambda_3} + 2\,|1/Q(s_i)|\, 2^{-\lambda_2} \leq 2^{-\lambda_3+T+2}$ which is also the accuracy of the result.

Putting together the various values of λ_i, ν, and T, we achieve an ℓ approximation by choosing $\lambda = \ell + 90n(\tau_1 + 3)\lg n + 32(n-1)\tau_2 \lg n + 30\tau_3 \lg n - 35 - 24\lg\Delta(s,t) - 4\lg\prod_k \Delta_k(t)$.

We perform all the computations with maximum required accuracy, and so the overall complexity is $\mathcal{O}_B(n\,\lg^2 n\,\mu(\lambda))$ or $\widetilde{\mathcal{O}}_B(n^2\tau_1 + n^2\tau_2 + n\tau_3 - n\lg\Delta(s,t) - n\lg\prod_k \Delta_k(t))$. □

7.2 Trummer's problem

This is the important special case where $s = t$ and the diagonal entries of the Cauchy matrix are set to zero. In this case we compute the matrix vector product by using the following formula,

$$(C\,v)_{i=0}^{n-1} = \left(\frac{2\,P'(s_i) - v_i\,Q''(s_i)}{2\,Q'(s_i)}\right)_{i=0}^{n-1} = \left(\frac{A_{0,i}}{A_{1,i}}\right)_{i=0}^{n-1} \quad (9)$$

We refer the reader to [16, Problem 3.6.3] for further details.

Corollary 25. *Using the notation of Lemma 24, we can solve Trummer's problem in $\mathcal{O}_B(n\,\lg^2 n\,\mu(\lambda))$, where $\lambda = \ell + 70(\tau_1+3)n\lg n + 4\tau_3 \lg n - 4\lg\prod_j \Delta(s)$.*

Proof: First we compute bounds for the numerator of Eq. (9). Following the proof of Lemma 24 we have $\lg|P'(s_i)| \leq (\tau_1 + 1)n + (n-1)\tau_2 + \tau_3 + 4n - 4 + \lg n$ and $\lg|P(s_i) - \widetilde{P}(\widetilde{s}_i)| \leq -\lambda_2 + \log n$. Similarly for $\lg|Q''(s_i)| \leq (\tau_1+1)n + (n-1)\tau_2 + \tau_3 + 4n - 4 + 2\lg n$ and $\lg|Q''(s_i) - \widetilde{Q}''(\widetilde{s}_i)| \leq -\lambda_2 + 2\log n$. For the first derivative we add to the logarithm of the norm a term $\lg n$ and for the second a term $2\lg n$. Moreover, $\tau_1 = \tau_2$, as $s = t$.

Taking into account that $|v_i| \leq 2^{\tau_3}$ we deduce that

$$\lg|A_{0,i}| \leq (\tau_1+1)n + (n-1)\tau_2 + 2\tau_3 + 4n - 4 + 2\lg n$$

and $\lg|A_{0,i} - \widetilde{A}_{0,i}| \leq -\lambda_2 + \tau_3 + 2\log n + 2$.

Regarding the denominator we have that $\lg|Q'(s_i)| \leq (\tau_1 + 1)n + (n-1)\tau_2 + \tau_3 + 4n - 4 + \lg n = T$. In addition $|Q'(s_i)| = \prod_{j\neq i}|s_i - s_j| \geq \prod_{j\neq i}\Delta_j(s)$ and so $|1/A_{1,i}| = |1/2Q'(s_i)| \leq \prod_{j\neq i}(\Delta_j(s))^{-1}$. Prop. 3 leads to $\lg\mathtt{wid}([1/A_{1,i}]) = \lg\mathtt{wid}[1/Q'(s_i)] \leq -\lambda_2 + \lg n - 4\lg\prod_{j\neq i}\Delta_j(s) + 2T + 3$ where λ_2 is defined at the (C3) step of the proof of Lemma 24.

Putting all the pieces together, we have

$$\begin{aligned}\mathtt{wid}([\tfrac{A_{0,i}}{A_{1,i}}]) \leq\ & 2\,|A_{0,i}|\, 2^{-\lambda_2 + \lg n - 4\prod_{j\neq i}\Delta_j(s) + 2T + 3} \\ & + 2\,|1/A_{1,i}|\, 2^{-\lambda_2 + \tau_3 + 2\log n + 2}\end{aligned}$$

and after many simplifications and overestimations

$$\begin{aligned}\lg\left|\frac{A_{0,i}}{A_{1,i}} - \frac{\widetilde{A}_{0,i}}{\widetilde{A}_{1,i}}\right| &\leq \mathtt{wid}([\tfrac{A_{0,i}}{A_{1,i}}]) \\ &\leq -\lambda + 70(\tau_1+3)n\lg n + 4\tau_3\lg n - 4\lg\textstyle\prod_j \Delta(s).\end{aligned}$$

To achieve an ℓ-approximation we need the input to be a λ-approximation, where $\lambda = \ell + 70(\tau_1+3)n\lg n + 4\tau_3\lg n - 4\lg\prod_j \Delta(s)$, and we perform all the computations with this number of bits. The overall complexity is $\mathcal{O}_B(n\,\lg^2 n\,\mu(\lambda))$. □

7.3 Solving a (Cauchy) linear system

We consider the following problem.

Problem 3 (Cauchy linear system of equations). *Solve a non-singular Cauchy linear system of n equations, $C(\mathbf{s},\mathbf{t})\,\mathbf{v} = \mathbf{r}$ for an unknown vector $\mathbf{v}$ and 3 given vectors $\mathbf{r}, \mathbf{s}$, and $\mathbf{t}$.*

Theorem 26. *Let $|s_i| \leq 2^{\tau_1}$, $|t_i| \leq 2^{\tau_2}$, $|r_i| \leq 2^{\tau_3}$, and $\Delta_i(\mathbf{s}) = \min_j |s_i - s_j|$, $\Delta_i(\mathbf{t}) = \min_j |t_i - t_j|$, for all i. Let also $\Delta(\mathbf{s},\mathbf{t}) = \min_{i,j}|s_i - t_j|$.*

If the input is given λ-approximations, where $\lambda = \ell + 630(\tau_1+\tau_2)n\lg n + 32\tau_3\lg n - 35 - 35\lg n\lg\prod\Delta_j(\mathbf{s}) - 5\lg\prod\Delta_j(\mathbf{t}) - 25\lg\Delta(\mathbf{s},\mathbf{t})$, then an ℓ-approximation solution to Problem 3 could be obtained in $\mathcal{O}_B(n\,\lg^2 n\,\mu(\lambda))$, or $\widetilde{\mathcal{O}}_B(n\ell + n^2(\tau_1 + \tau_2) + n\tau_3 - n\lg\prod\Delta_j(\mathbf{s}) - n\lg\prod\Delta_j(\mathbf{t}) - n\lg\Delta(\mathbf{s},\mathbf{t}))$.

Proof: Following [16, Eq. 3.6.10] then inverse of $C(\mathbf{s},\mathbf{t})$ could be obtained as follows

$$\begin{aligned}C^{-1}(\mathbf{s},\mathbf{t}) &= \mathsf{diag}(p_{\mathbf{s}}(t_i)/p'_{\mathbf{t}}(t_i))_{i=0}^{n-1}\, C(\mathbf{t},\mathbf{s})\,\mathsf{diag}(p_{\mathbf{t}}(s_i)/p'_{\mathbf{s}}(s_i))_{i=0}^{n-1} \\ &= D_1 \cdot C(\mathbf{t},\mathbf{s}) \cdot D_2\end{aligned}, \quad (10)$$

where $p_{\mathbf{t}}(X) = \prod_i (X - t_i)$ and $p_{\mathbf{s}}(X) = \prod_i (X - s_i)$.

Using this formula we can solve the linear system as

$$\mathbf{v} = C^{-1}(\mathbf{s},\mathbf{t})\,\mathbf{r} = D_1 \cdot C(\mathbf{t},\mathbf{s}) \cdot D_2\,\mathbf{r}$$

We apply Lemma 17 to analyze the computation of the polynomials $p_s(x)$ and $p_t(x)$. Then we compute the two diagonal matrices D_1 and D_2 by applying the Moenck–Borodin algorithm for multipoint evaluation (see Section 4). At first we perform n ops to compute the vector $\mathbf{r_1} = D_2\,\mathbf{r}$. Then we use Lemma 24 to obtain the vector $\mathbf{r_2} = C(\mathbf{t},\mathbf{s})\mathbf{r_1}$. Finally we multiply the matrix D_1 by $\mathbf{r_2}$ to obtain the vector $\mathbf{v}$. We track the loss of precision at each of these three steps.

1. Computation of the matrices D_1 and D_2.

   For this task we need to compute $p_{\mathbf{s}}(t_i)$, $p'_{\mathbf{s}}(s_i)$, $p_{\mathbf{t}}(s_i)$, and $p'_{\mathbf{t}}(t_i)$.

   It holds $\lg\|p_{\mathbf{s}}\|_\infty \leq n\tau_1 + 4n - \lg n - 4$ and $\lg\|p_{\mathbf{s}} - \widetilde{p}_{\mathbf{s}}\|_\infty \leq -\lambda + (4n-4)\tau_1 + 32n$, using Lemma 19. By applying Lemma 21 we get $\lg\|p_{\mathbf{s}}(t_i)\|_\infty \leq n\tau_1 + n\tau_2 + 5n - 3$. and $\lg\|p_{\mathbf{s}}(t_i) - \widetilde{p}_{\mathbf{s}}(\widetilde{t}_i)\|_\infty \leq -\lambda + (n\lg n + 4\,n - 4)\tau_1 + 60\,n\tau_2\lg n - 4\,\lg n + 184\,n\lg n + 32\,n - (\lg n)^2$.

   However, to simplify the calculations we consider the inferior bounds $\lg\|p_{\mathbf{s}}(t_i)\|_\infty \leq 7n(\tau_1+\tau_2)$. and $\lg\|p_{\mathbf{s}}(t_i) - \widetilde{p}_{\mathbf{s}}(\widetilde{t}_i)\|_\infty \leq -\lambda + 300(\tau_1+\tau_2)n\lg n$ for all the involved quantities.

   We also need lower bounds for $|p'_{\mathbf{t}}(t_i)|$ and $|p'_{\mathbf{s}}(s_i)|$.

   It holds $|p'_{\mathbf{t}}(t_i)| \geq \prod_{j\neq i}|t_i - t_j| \geq \prod_{j\neq i}\Delta_j(\mathbf{t})$. Similarly $|p'_{\mathbf{s}}(s_i)| \geq \prod_{j\neq i}|s_i - s_j| \geq \prod_{j\neq i}\Delta_j(\mathbf{s})$.

   This leads to the bounds: $\left|\frac{p_{\mathbf{s}}(t_i)}{p'_{\mathbf{t}}(t_i)}\right| \leq 2^{7n(\tau_1+\tau_2) - \lg\prod_{j\neq i}\Delta_j(\mathbf{t})}$ and $\left|\frac{p_{\mathbf{t}}(s_i)}{p'_{\mathbf{s}}(s_i)}\right| \leq 2^{7n(\tau_1+\tau_2) - \lg\prod_{j\neq i}\Delta_j(\mathbf{s})}$.

   By combining the previous bounds with the complex interval arithmetic of Prop. 1 we obtain the following estimation for the approximation:

$$\lg\left|\frac{p_{\mathbf{t}}(s_i)}{p'_{\mathbf{s}}(s_i)} - \widetilde{\frac{p_{\mathbf{t}}(s_i)}{p'_{\mathbf{s}}(s_i)}}\right| \leq -\lambda + 315(\tau_1+\tau_2)n\lg n - 4\lg\prod_{j\neq i}\Delta_j(\mathbf{s})$$

$$\lg\left|\frac{p_{\mathbf{s}}(t_i)}{p'_{\mathbf{t}}(t_i)} - \widetilde{\frac{p_{\mathbf{s}}(t_i)}{p'_{\mathbf{t}}(t_i)}}\right| \leq -\lambda + 315(\tau_1+\tau_2)n\lg n - 4\lg\prod_{j\neq i}\Delta_j(\mathbf{t})$$

2. $\mathbf{r_1} = D_2\,\mathbf{r}$.

   This computation increases the bounds by a factor of τ_3. To be more specific, for the elements of $\mathbf{r_1}$, $r_{1,i}$ we have $|r_{1,i}| \leq 2^{7n(\tau_1+\tau_2)+\tau_3-\lg\prod_{j\neq i}\Delta_j(\mathbf{s})}$ and

$$\lg|r_{1,i} - \widetilde{r}_{i,1}| \leq -\lambda + 316(\tau_1+\tau_2)n\lg n + \tau_3 - 4\lg\prod_{j\neq i}\Delta_j(\mathbf{s})$$

3. $\mathbf{r_2} = C(\mathbf{t},\mathbf{s})\,\mathbf{r_1}$.
   For this computation we need to apply Lemma 24. We obtain

$$\lg|r_{2,i}| \leq 7n(\tau_1+\tau_2) + \tau_3 - \lg\prod\Delta_j(\mathbf{s}) - \lg\Delta(\mathbf{s},\mathbf{t}),$$

$$\begin{aligned}\lg|r_{2,i} - \widetilde{r}_{2,i}| \leq & -\lambda + 616(\tau_1+\tau_2)n\lg n + 31\tau_3\lg n - 35\\ & - 34\lg n\lg\prod\Delta_j(\mathbf{s}) - 4\lg\prod\Delta_j(\mathbf{t}) - 24\lg\Delta(\mathbf{s},\mathbf{t})\end{aligned}$$

4. $\mathbf{v} = D_1\,\mathbf{r_2}$.

   This computations leads to the following bounds

$$\lg|v_i| \leq 14n(\tau_1+\tau_2) + \tau_3 - \lg\prod\Delta_j(\mathbf{s}) - \lg\prod\Delta_j(\mathbf{t}) - \lg\Delta(\mathbf{s},\mathbf{t})$$

$$\begin{aligned}\lg|v_i - \widetilde{v}_i| \leq & -\lambda + 630(\tau_1+\tau_2)n\lg n + 32\tau_3\lg n - 35\\ & - 35\lg n\lg\prod\Delta_j(\mathbf{s}) - 5\lg\prod\Delta_j(\mathbf{t}) - 25\lg\Delta(\mathbf{s},\mathbf{t})\end{aligned}$$

To achieve an ℓ-approximation of the output we should require a λ-approximation of the input, where $\lambda = \ell + 630(\tau_1+\tau_2)n\lg n + 32\tau_3\lg n - 35 - 35\lg n\lg\prod\Delta_j(\mathbf{s}) - 5\lg\prod\Delta_j(\mathbf{t}) - 25\lg\Delta(\mathbf{s},\mathbf{t})$.

As in all the previous sections we perform all the computations using the maximum precision. The overall complexity of the algorithm is $\mathcal{O}_B(n\lg^2 n\,\mu(\lambda))$, or $\widetilde{\mathcal{O}}_B(n\ell + n^2(\tau_1+\tau_2) + n\tau_3 - n\lg\prod\Delta_j(\mathbf{s}) - n\lg\prod\Delta_j(\mathbf{t}) - n\lg\Delta(\mathbf{s},\mathbf{t}))$. $\square$

Acknowledgments. VP is supported by NSF Grant CCF–1116736 and PSC CUNY Awards 64512–0042 and 65792–0043. ET is partially supported by GeoLMI (ANR 2011 BS03 011 06), HPAC (ANR ANR-11-BS02-013) and an FP7 Marie Curie Career Integration Grant.

8. REFERENCES

[1] A. V. Aho, K. Steiglitz, and J. D. Ullman. Evaluating polynomials at fixed sets of points. *SIAM Journal on Computing*, 4(4):533–539, 1975.

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

[3] D. A. Bini and L. Robol. Solving secular and polynomial equations: A multiprecision algorithm. *Journal of Computational and Applied Mathematics*, 2013. (to appear).

[4] L. Bluestein. A linear filtering approach to the computation of discrete fourier transform. *Audio and Electroacoustics, IEEE Transactions on*, 18(4):451–455, 1970.

[5] J. R. Bunch. Stability of methods for solving toeplitz systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 6(2):349–364, 1985.

[6] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing*, 9(4):669–686, 1988.

[7] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. The DMM bound: Multivariate (aggregate) separation bounds. In S. Watt, editor, *ISSAC*, pages 243–250, Munich, Germany, July 2010. ACM.

[8] M. J. Fischer and M. S. Paterson. String-matching and other products. In R. Karp, editor, *Complexity of Computation*, volume 7, pages 113–125. SIAM-AMS Proc., 1974.

[9] A. Gerasoulis, M. D. Grigoriadis, and L. Sun. A fast algorithm for trummer's problem. *SIAM journal on Scientific and Statistical Computing*, 8(1):s135–s138, 1987.

[10] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[11] P. Kirrinnis. Partial fraction decomposition in $C\langle z\rangle$ and simultaneous Newton iteration for factorization in $C[z]$. *Journal of Complexity*, 14(3):378–444, 1998.

[12] D. E. Knuth. *The art of computer programming, volume 2 (2nd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[13] A. Kobel and M. Sagraloff. Fast approximate polynomial multipoint evaluation and applications. *arXiv preprint arXiv:1304.8069*, 2013.

[14] M. Mignotte. *Mathematics for Computer Algebra*. Springer-Verlag, New York, 1991.

[15] R. Moenck and A. Borodin. Fast modular transforms via division. In *Proc. of the 13th Annual Symposium on Switching and Automata Theory (SWAT)*, pages 90–96, Washington, DC, 1972. IEEE Computer Society.

[16] V. Pan. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Birkhäuser / Springer, Boston / New York, 2001.

[17] V. Y. Pan and E. Tsigaridas. Nearly Optimal Refinement of Real Roots of a Univariate Polynomial. 2014.

[18] F. P. Preparata and M. I. Shamos. Computational geometry. texts and monographs in computer science, 1985.

[19] P. Ritzmann. A fast numerical algorithm for the composition of power series with complex coefficients. *TCS*, 44:1–16, 1986.

[20] J. Rokne and P. Lancaster. Complex interval arithmetic. *Communications of the ACM*, 14(2):111–112, 1971.

[21] A. Schönhage. Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coeficients. In J. Calmet, editor, *EUROCAM*, volume 144 of *LNCS*, pages 3–15, 1982.

[22] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany, 1982.
URL: http://www.iai.uni-bonn.de/~schoe/fdthmrep.ps.gz.

[23] A. Strzeboński and E. P. Tsigaridas. Univariate real root isolation in an extension field. In A. Leykin, editor, *Proc. 36th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 321–328, San Jose, CA, USA, June 2011. ACM.

[24] A. Strzeboński and E. P. Tsigaridas. Univariate real root isolation in presence of logarithms. *Arxiv*, Jan 2014.

[25] J. van der Hoeven. Fast composition of numeric power series. Technical Report 2008-09, Université Paris-Sud, Orsay, France, 2008.

[26] C. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.

Appendix

Lemma 10. *Let $n+1=2^k$ for a positive integer k and let T be a lower triangular Toeplitz $(n+1)\times(n+1)$ matrix of. Eq. (5), having ones on the diagonal. Let its subdiagonal entries be complex numbers of magnitude at most 2^τ known up to a precision $2^{-\lambda}$. Let 2^ρ be an upper bounds on the magnitude of the roots of the univariate polynomial $t(x)$ associated with T. Write $T^{-1}=(T^{-1}_{i,j})^n_{i,j=0}$. Then*

$$\max_{i,j}|T^{-1}_{i,j}| \le 2^{(\rho+1)n+\lg(n)+1} \ .$$

Furthermore, to compute the entries of T^{-1} up to the precision of ℓ bits, that is to compute a matrix $\widetilde{T}^{-1}=(\widetilde{T}^{-1}_{i,j})^n_{i,j=0}$ such that $\max_{i,j}|T^{-1}_{i,j}-\widetilde{T}^{-1}_{i,j}|\le 2^{-\ell}$, it is sufficient to know the entries of T up to the precision of $\ell+10\tau\lg n+70\lg^2 n+8(\rho+1)n\lg n$ or $\mathcal{O}(\ell+(\tau+\lg n+n\rho)\lg n)=\widetilde{\mathcal{O}}(\ell+\tau+n\rho)$ bits.

The computation of $\widetilde{T}^{-1}$ costs $\mathcal{O}_B(n\lg^2(n)\,\mu(\ell+(\tau+\lg n+n\rho)\lg n))$ or $\widetilde{\mathcal{O}}_B(n\ell+n\tau+n^2\rho)$.

Proof (of Lemma 10): We will prove the claimed estimates by reducing the inversion to recursive multiplication of polynomials defined by equation (6) and Lemma 7.

Consider the $\frac{n+1}{2}\times\frac{n+1}{2}$ Toeplitz matrices T_0^{-1} and T_1 of equation (6). The Toeplitz matrix T_0^{-1} is triangular, and so its first column, $\mathbf{p}=(p_i)_{i=0}^{(n-1)/2}$, with $p_0=1$, defines this matrix and the polynomial $p(x)=\sum_{i=0}^{(n-1)/2}p_ix^i$ of degree $(n-1)/2$. Likewise the vector $\mathbf{t}=(t_{n-i})_{i=1}^n$ (made up of two overlapping vectors, that is, the reversed first row, $(t_{n-1},t_{n-2},\dots,t_{(n-1)/2})$ of the matrix T_1 and its first column, $(t_{(n-1)/2},t_{(n-3)/2},\dots,t_0)^T$) defines this matrix and the polynomial $\tilde{t}(x)=\sum_{i=1}^n t_{n-i}x^{i-1}$ of degree $n-1$. The first column of the $\frac{n+1}{2}\times\frac{n+1}{2}$ Toeplitz matrix $T_1T_0^{-1}$ is a subvector, $\mathbf{v}$, of dimension $(n+1)/2$ of the coefficient vector of the polynomial product $\tilde{t}(x)p(x)$, having degree $3(n-1)/2$. Likewise the first column of the $\frac{n+1}{2}\times\frac{n+1}{2}$ matrix $-T_0^{-1}T_1T_0^{-1}$ is the vector $\mathbf{q}=-T_0^{-1}\mathbf{v}$, which is a subvector of dimension $(n+1)/2$ of the coefficient vector of the polynomial product $p(x)v(x)$ of degree $n-1$, where the polynomial $v(x)$ of degree $(n-1)/2$ is defined by its coefficient vector $\mathbf{v}$. In sum the vector $\mathbf{q}$ is the coefficient vector of a polynomial $\tilde{q}(x)$ obtained by two successive multiplications of polynomials, each followed by the truncation of the coefficient vectors. Namely we first compute the polynomial $\tilde{t}(x)p(x)$, then truncate it to obtain the polynomial $v(x)$, then compute the polynomial $-p(x)v(x)$, and finally truncate it to obtain the polynomial $\tilde{q}(x)$.

The truncation can only decrease the degree of a polynomial and the maximum length of its coefficients, and so we can bound the precision and the cost of computing the matrix product $-T_0^{-1}T_1T_0^{-1}$ by the bounds on the precision and the cost of computing the polynomial product $P_0^2P_1$, estimated in Corollary 8 for $P_0=p(x)$, $P_1=\tilde{t}(x)$, and $d=(n-1)/2$.

First we prove the upper bound on the elements of T^{-1}. Consider the division $\phi_{i,k}(x)=t(x)q(x)+r(x)$, where $t(x)$ is the univariate polynomial of degree n associated with $\phi_{i,k}(x)=\operatorname{sign}(T^{-1}_{i,1})x^n+\operatorname{sign}(T^{-1}_{i,k})x^k$, $2\le k\le n$ and $1\le i\le n$. Then $\|\phi_{i,k}\|_\infty\le 1$ and $\|\phi_{i,k}\|_2\le\sqrt{2}$. By abusing notation we also write $\phi_{i,k}$ and q to denote the coefficient vectors of these polynomials. Using Eq. (5) we can compute the elements of q from the equation $q=T^{-1}\phi_{i,k}$. In this way $|q_i|=|\operatorname{sign}(T^{-1}_{i,1})T^{-1}_{i,1}+\operatorname{sign}(T^{-1}_{i,k})T^{-1}_{i,k}|=|T^{-1}_{i,1}|+|T^{-1}_{i,k}|$.

Using Lemma 9 we obtain the inequality $|q_i|\le\|q\|_\infty\le 2^{n+\lg n+n\rho}\|\phi_{i,k}\|_\infty$, which in turn implies

$$|T^{-1}_{i,k}|\le|T^{-1}_{i,1}|+|T^{-1}_{i,k}|=|q_i|\le 2^{n+\lg n+n\rho}\,\|\phi_{\nu,k}\|_\infty\le 2^{n+\lg n+n\rho} \ . \tag{11}$$

Let P_0 and P_1 denote the two polynomials defined earlier in the proof such that $\max_{i,j}\{|T^{-1}_{i,j}|\}\le\|P_0^2P_1\|_\infty$.

Then Eq. (11) implies the inequality

$$\lg\left\|P_0^2\,P_1\right\|_\infty\le n+\lg n+n\rho=N \ . \tag{12}$$

Recall that $n+1=2^k$ by assumption and that the polynomial $P_0^2\,P_1$ has degree $2n-2$. Write $h=\lceil\lg(2n-2)\rceil$. Under the assumption that the input elements are known up to the precision of λ bits, we compute a polynomial $\widetilde{P_0^2\,P_1}$ such that

$$\begin{aligned}&\lg\left\|P_0^2P_1-\widetilde{P_0^2\,P_1}\right\|_\infty\le\\ &\le-\lambda+(2\lg n+8)\tau+2\lg n(4\lg n+27)+8(\lg n-1)N\\ &\le-\lambda+(2k+8)\tau+2k(4k+27)+8(k-1)N \ .\end{aligned}\tag{13}$$

We proceed by induction. Let the base case be $n+1=4$; then $k=2$. We need to invert the following matrix

$$T=\begin{bmatrix}1&0&0&0\\ t_2&1&0&0\\ t_1&t_2&1&0\\ t_0&t_1&t_2&1\end{bmatrix}=\begin{bmatrix}T_0&0\\ T_1&T_0\end{bmatrix}\ , \text{ where}$$

$$T_0=\begin{bmatrix}1&0\\ t_2&1\end{bmatrix},\ T_0^{-1}=\begin{bmatrix}1&0\\ -t_2&1\end{bmatrix},\ T_1=\begin{bmatrix}t_1&t_2\\ t_0&t_1\end{bmatrix}\ ,$$

and $|t_i|\le 2^\tau$. The associated polynomials are $P_0(x)=1-t_2x$, for T_0^{-1}, and $P_1(x)=t_2+t_1x+t_0x^2$, for T_1. Therefore $P_1P_0=(1-t_2x)(t_2+t_1x+t_0x^2)=t_2+(t_1-t_2^2)x+(t_0-t_1t_2)x^2-t_0t_2x^3$. The subvector $\mathbf{v}=(t_1-t_2^2,t_0-t_1t_2)^T$ of the coefficient vector of the polynomial product P_1P_0 is the first column of the matrix product $T_1T_0^{-1}$. Furthermore the vector $-T_0^{-1}\mathbf{v}=(t_2^2-t_1,2t_1t_2-t_2^3-t_0)^T$ is a subvector of the coefficient vector of the polynomial product $-P_0v=-(1-t_2x)(t_1-t_2^2+(t_0-t_1t_2)x)$ where v denotes the polynomial $t_1-t_2^2+(t_0-t_1t_2)x$ with the coefficient vector $\mathbf{v}$. As we proved earlier, the subvector is the first column of the matrix

$$-T_0^{-1}\,T_1\,T_0^{-1}=\begin{bmatrix}{t_2}^2-t_1&-t_2\\ -{t_2}^3+2\,t_1t_2-t_0&{t_2}^2-t_1\end{bmatrix}\ .$$

(This is not a subvector of the coefficient vector of the polynomial

$$\begin{aligned}&P_0^2P_1=(1-t_2x)^2(t_2+t_1x+t_0x^2)\\ &=t_2+(t_1-2t_2^2)x+(t_0-2t_1t_2+t_2^3)x^2+(t_2^2t_1-2t_2t_0)x^3+t_2^2t_0x^4\end{aligned}$$

because multiplication of polynomials and the truncation of their coefficient vectors are not commutative operations, but as we observed, we still can reduce our study of the product $T_0^{-1}T_1T_0^{-1}$ to the study of $p_0^2p_1$.)

We perform the multiplications by using the algorithm of Lemma 5 and the bounds from Cor. 8, to obtain a polynomial $\widetilde{P_0^2\,P_1}$ such that

$$\lg\|P_0^2P_1-\widetilde{P_0^2\,P_1}\|_\infty\le-\lambda+10\tau+15\lg 1+40\le-\lambda+12\tau+140+8N\ ,$$

where the last inequality is obtained by substituting $k=2$ in Eq. (13).

It remains to prove the induction. Assume that the claimed bounds are true for $n+1 = 2^k$ and extend them to $n+1 = 2^{k+1}$. In our case P_0 is a polynomial of degree $2^{k-1}-1$. By induction hypothesis we know the coefficient of P_0 within $2^{-\ell}$ for ℓ defined by (13), and we can apply Eq. (12) to obtain $\|P_0\|_\infty \le 2^N$.

The polynomial P_1 has degree $2^k - 2$, $\|P_1\|_\infty \le 2^\tau$. Its coefficients are the entries of the matrix T and we know them within $2^{-\lambda}$.

We apply Cor. 8 for $d = 2^{k-1}-1$, $\tau_0 = N$, $\tau_1 = \tau$, and $-\nu = -\lambda + (2(k-1)+8)\tau + 2(k-1)(4(k-1)+27) + 8(k-2)N$, substitute $k \ge 3$, and obtain the following approximation bound,

$$\begin{aligned}\lg \left\| P_0^2 P_1 - \widetilde{P_0^2 P_1} \right\|_\infty &\le -\nu + 8\tau_0 + 2\tau_1 + 14k + 42\\ &\le -\lambda + (2k+8)\tau + 8k^2 - 2k + 104 + 8(k-1)N\end{aligned}$$

These inequalities imply that all our computations require a precision bound of at most $\lambda' = \ell + (2k+8)\tau + 2k(4k+27) + 8(k-1)N$. To simplify the formula, we substitute $k = \lg(n)$ rather than $k = \lg(n+1)$ and then rewrite λ' as follows, $\ell + (2\lg n + 8)\tau + 2\lg n(4\lg n + 27) + 8(\lg n - 1)(n + \lg n + n\rho + 1)$ bits, which we simplify to the bound of $\ell + 10\tau\lg n + 70\lg^2 n + 8(\rho+1)n\lg n$ or $\mathcal{O}(\ell + (\tau + \lg n + n\rho)\lg n)$ bits.

To estimate the overall complexity of approximating the first column of the inverse matrix T^{-1}, we notice that we perform k steps overall, for $k = \lg(n+1)$, but we will keep writing $k = \lg(n)$ to simplify the notation.

At each step we perform two multiplications of polynomials of degrees at most 2^k with the coefficients having absolute values less than $2^{\mathcal{O}(n\tau)}$, and we use the precision of $\ell + 10\tau\lg n + 70\lg^2 n + 8(\rho+1)n\lg n = \mathcal{O}(\ell + (\tau + \lg n + n\rho)\lg n)$ bits. All these bounds together imply that

$$\begin{aligned}&\textstyle\sum_{k=1}^{\lg(n+1)} k \cdot 2 \cdot \mathcal{O}_B(2^k \cdot \lg 2^k \cdot \mu(\ell + (\tau + \lg n + n\rho)\lg n)) =\\ &= \mathcal{O}_B(n\ \lg^2(n)\,\mu(\ell + (\tau + \lg n + n\rho)\lg n))\ ,\end{aligned}$$

which concludes the proof. □

9. A NORMALIZATION OF KIRRINNIS' RESULTS

The following results express the bounds of Kirrinnis [11] for polynomials of arbitrary norms; for this an appropriate scaling is applied. The complexity bounds depends on polynomial multiplication algorithms that rely on Kronecker substitution and not on FFT as our results that we presented earlier. The bounds are asymptotically the same as ours, up to logarithmic factors.

The following is from [11, Theorem 3.7 and Algorithm 5.1].

Lemma 27 (Multiplication of polynomials). *Let $P_j \in \mathbb{C}[x]$ of degree n_j and $\|P_j\|_\infty \le 2^\tau$ and $\widetilde{P}_j$ be an approximation of P_j such that $\|P_j - \widetilde{P}_j\|_\infty \le 2^{-\lambda}$, with $\lambda = \ell n(\tau + 2) + n\lg n$., where $1 \le j \le m$, $n_1 \le \cdots \le n_m$, and $\sum_{n_j} = n$. We can compute $\prod_j \widetilde{P}_j$ such that $\|\prod_j P_j - \prod_j \widetilde{P}_j\|_\infty \le 2^{-\ell}$ in $\mathcal{O}_B(\mu(n \cdot \lg n \cdot (\ell + n\tau + \sum_j \lg n_j)))$ or $\widetilde{\mathcal{O}}_B(n(\ell + n\tau))$.*

Proof: Let $p_j = 2^{-\tau - \lg n_j + n_j} P_j$. Then

$$\|P_j\|_\infty \le 2^\tau \Rightarrow \|P_j\|_1 \le 2^{\tau + \lg n_j}$$

$$\Rightarrow 2^{-\tau - \lg n_j + n_j}\|P_j\|_1 \le 2^{n_j} \Rightarrow \|p_j\|_1 \le 2^{n_j}\ .$$

Moreover,

$$\|P_j - \widetilde{P}_j\|_\infty \le 2^{-\lambda} \Rightarrow \|P_j - \widetilde{P}_j\|_1 \le 2^{-\lambda + \lg n_j}$$

$$\Rightarrow \|p_j - \widetilde{p}_j\|_1 \le 2^{-\tau - \lg n_j + n_j} 2^{-\lambda + \lg n_j} = 2^{-\lambda - \tau + n_j}\ .$$

The algorithm of Kirrinnis [11, Theorem 3.7 and Algorithm 5.1] computes and approximation of $\prod_j p_j$ such that $\left\|\prod_j p_j - \prod_j \widetilde{p}_j\right\|_1 \le 2^{-s+\tau+3n}$, which implies

$$\left\| \prod_j 2^{-\tau - \lg n_j + n_j} P_j - \prod_j 2^{-\tau - \lg n_j + n_j} \widetilde{P}_j \right\|_1 \le 2^{-\lambda + \tau + 3n}$$

$$\Rightarrow 2^{-n\tau - \sum_j \lg n_j + n} \left\| \prod_j P_j - \prod_j \widetilde{P}_j \right\|_1 \le 2^{-\lambda + \tau + 3n}\ ,$$

and so

$$\left\| \prod_j P_j - \prod_j \widetilde{P}_j \right\|_1 \le 2^{-\lambda + n(\tau+2) + n\lg n}\ .$$

If we want the error to be less than $2^{-\ell}$, then we need to choose initial precision $\lambda \ge \ell n(\tau+2) + n\lg n$. The total cost is $\mathcal{O}_B(\mu(n \cdot \lg n \cdot (\ell + n\tau + \sum_j \lg n_j)))$ or $\widetilde{\mathcal{O}}_B(n(\ell + n\tau))$. □

The algorithm for computing the sum of rational functions [11, Theorem 3.8 and Algorithm 5.2] relies on Lemma 27 and the bounds are similar, so we do not elaborate further.

The following lemma relies on [11, Theorem 3.9 and Algorithm 5.3].

Lemma 28 (Modular representation). *Let $F \in \mathbb{C}[x]$ of degree m and $\|F\|_\infty \le 2^{\tau_1}$. Let $P_j \in \mathbb{C}[x]$ of degree n_j, for $1 \le j \le \nu$, such that $\sum_j n_j = n$ and $m \ge n$. Moreover, 2^ρ be an upper bound on the magnitude of the roots of all P_j. Let $\widetilde{F}$, resp. $\widetilde{P}_j$, be an approximation of F, resp. P_j, such that $\|F - \widetilde{F}\|_\infty \le 2^{-\lambda}$, resp. $\|P_j - \widetilde{P}_j\|_\infty \le 2^{-\lambda}$.*

If $\lambda = \ell + \tau_1 + 2(n+m)\rho$ then we compute approximations $\widetilde{F}_j$ of $F_j = F \mod P_j$, such that $\|F_j - \widetilde{F}_j\|_\infty \le 2^{-\ell}$ in $\mathcal{O}_B(\mu((m + n\lg n)(\ell + \tau_1 + (m+n)\rho)))$.

Proof: The polynomials $P_j(2^\rho\, x)$ have all their roots inside the unit disc. We make them monic and then denote them p_j. It holds:

$$\|P_j - \widetilde{P}_j\|_\infty \le 2^{-\lambda} \Rightarrow \|P_j(2^\rho\, x) - \widetilde{P}_j(2^\rho\, x)\|_1 \le 2^{-\lambda + n_j\rho + \lg n_j}$$

$$\Rightarrow \|p_j - \widetilde{p}_j\|_1 \le 2^{-\lambda + n_j\rho + \lg n_j + 1}\ .$$

We apply the same transformation to F.

Let $f = 2^{-\tau_1 - m\rho - \lg m} F(2^\rho\, x)$, then

$$\|F\|_\infty \le 2^{\tau_1} \Rightarrow \|F\|_1 \le 2^{\tau_1 + \lg m} \Rightarrow \|F(2^\rho\, x)\|_1 \le 2^{\tau_1 + m\rho + \lg m}$$

$$\Rightarrow 2^{-\tau_1 - m\rho + \lg m}\|F(2^\rho\, x)\|_1 \le 1 \Rightarrow \|f\|_1 \le 1\ .$$

Now we can use [11, Theorem 3.9] choosing $\lambda = \ell + \tau_1 + 2(n+m)\rho$ to guarantee $\|f \mod p_j - \widetilde{f}_j\|_1 = \|f_j - \widetilde{f}_j\|_1 \le 2^{-\ell}$. The complexity of the procedure is $\mathcal{O}_B(\mu((m + n\lg n)(\ell + \tau_1 + (m+n)\rho)))$. □

Remark 29. *In the case where $m = n$ the latter bound becomes $\widetilde{\mathcal{O}}_B(n(\ell + \tau_1 + n\rho))$.*

Figure 1. Fan-in process, $m_j^{(h+1)} = m_{2j}^{(h)} m_{2j+1}^{(h)}$

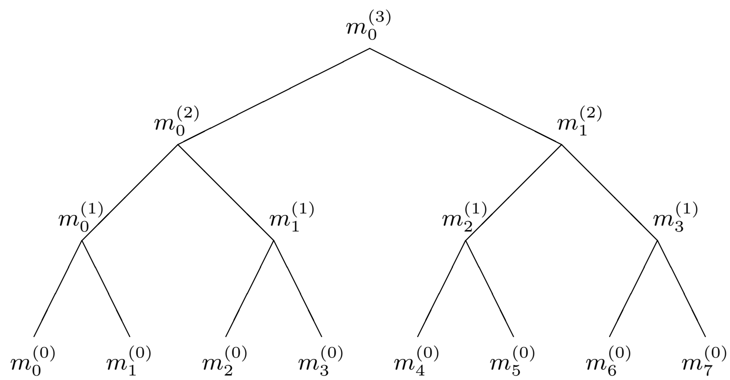

Figure 2. Fan-out process, $r_j^{(h)} = r_{\lfloor j/2 \rfloor}^{(h+1)} \bmod m_j^{(h)} = v(x) \bmod m_j^{(h)}$