Polynomial and Rational Evaluation and Interpolation (with Structured Matrices) [⋆]

Vadim Olshevsky and Victor Y. Pan

Dept. of Math. & Comp. Sci.
Georgia State University,
Atlanta, GA 30303
volshevsky@cs.gsu.edu

Dept. of Math. & Comp. Sci.
Lehman College, CUNY,
Bronx, NY 10468
vpan@alpha.lehman.cuny.edu

Summary: Polynomial and rational interpolation and multipoint evaluation are classical subjects, which remain central for the theory and practice of algebraic and numerical computing and have extensive applications to sciences, engineering and signal and image processing. In spite of long and intensive study of these subjects, several major problems remained open. We achieve substantial progress, based on our new matrix representations of the problems with the use of node polynomials. In particular:

1. We show strong correlation between rational and polynomial problems as well as between evaluation and interpolation, which enables our unified algorithmic treatment of all these subjects.

2. In applications of real polynomial evaluation and interpolation to sciences, engineering, and signal and image processing, most important is the case where input/output is represented in Chebyshev bases. In this case we rely on fast cosine/sine transforms (FCT/FST) to decrease the arithmetic cost of the known solutions from order of n to $O(\log^2 n)$ per node point.

3. In the general complex case, we devise new effective approximation algorithms for polynomial and rational evaluation and interpolation, for all input polynomials of degree $n-1$ and all sets of n nodes. The algorithms support the arithmetic complexity bounds of $O(\log n)$ per node point for approximate solution of these classical problems within the output error bound $\epsilon = 2^{-b}$, $\log b = O(\log n)$, taken relative to the specified input parameters. This substantially improved the known estimate of $O(\log^2 n)$ per point.

Our algorithms supporting the cited complexity bounds allow their NC and work optimal parallelization. Our results also include new exact solution algorithms with arithmetic cost $O(\log^2 n)$ per node point for a) Trummer's problem of rational evaluation, which (unlike Gerasoulis algorithm) is interpolation-free, and b) rational interpolation with unknown poles, which exploits the matrix structure instead of the customary reduction to Euclidean algorithm. Technically, we exploit correlation among various problems of polynomial and rational interpolation, their matrix representations and transformations (mappings) into each other of both problems and the associated structured matrices.

[⋆] This work was supported by NSF grants CCR 973235, CCR 9625344, CCR 9732206, GSU ORSP Award, and PSC CUNY Award 669363

J. Wiedermann et al. (Eds.): ICALP'99, LNCS 1644, pp. 585–594, 1999.

1 Introduction

Our subjects are polynomial and rational interpolation and multipoint evaluation, to which we will refer hereafter as *p.i., r.i., p.e.* and *r.e.*, respectively. Our main results are stated in the summary as well as in our Theorems 3.1, 5.2, 6.1 and Corollaries 5.1, 5.2, and 6.1. This includes nontrivial matrix equations (5.2) and (5.3), which extend a result of [BKO,a] and serve as the basic tools for our transformations of problems of p.i., p.e., r.i. and r.e. Our techniques involved in particular in the derivation of these equations and the cited transformations may be of independent interest. Our present study of r.i. has no overlap with [OP98] because, unlike [OP98], we assume neither the so called passivity conditions nor any boundary conditions.

In the next sections we will state the problems formally, give some background, including motivation and known difficulties, specify our contribution stated in the summary and recall the known results for comparison. Our complexity estimates are given in terms of the number of arithmetic operations and comparisons involved, to all of which we will refer as *ops*. (We will assume the customary arithmetic models of RAM (PRAM) or circuits [AHU74], [KR90], [G86]. The computations of sections 2-6 can be assumed over any field, except that we rely on the cost bound $O(n \log n)$ for polynomial multiplication modulo x^n, ignoring the extra factor $\log \log n$, required over abstract fields (cf. [BP94], pp. 12, 13, 56).) Since the basic blocks of our algorithms are FFT, FCT, FST, and linear operations with vectors, parallel versions of our algorithms can be also implemented efficiently under some more realistic models such as the hypercube and butterfly network models [Le92].

2 Classical Polynomial Evaluation and Interpolation

We first define a vector $\mathbf{x} = (x_i)_{i=0}^{n-1}$ of n distinct nodes, a polynomial $c(x) = \sum_{j=0}^{n-1} c_j x^j$, the vectors $\mathbf{c} = (c_j)_{j=0}^{n-1}$ of its coefficients and $\mathbf{v} = (v_i)_{i=0}^{n-1}$, $v_i = c(x_i)$, of its values at the nodes x_i, as well as the *node polynomial* $h_{\mathbf{x}}(x) = \prod_{i=0}^{n-1}(x - x_j)$ (having the coefficient vector $\mathbf{h}_{\mathbf{x}}$) and its derivative $h'_{\mathbf{x}}(x) = \frac{dh_{\mathbf{x}}(x)}{dx} = \sum_{j=0}^{n-1} \prod_{j \neq i}(x - x_j)$. Having these definitions, we state the problems of (multipoint) polynomial evaluation (p.e.) and interpolation (p.i.).

Problem 2.1, *p.e.*

Input: two vectors $\mathbf{c}$ and $\mathbf{x}$ and the coefficient vector $\mathbf{h}_{\mathbf{x}}$ of the node polynomial $h_{\mathbf{x}}(x)$.

Output: the vector $\mathbf{v}$.

Problem 2.2, *p.i.*

Input: two vectors $\mathbf{x}$ and $\mathbf{v}$ and the vector $\mathbf{h}_{\mathbf{x}}$.

Output: the vector $\mathbf{c}$.

Problems 2.1 and 2.2 reverse each other. P.e. and its solution can be trivially extended to the evaluation at n points of polynomials of degree $m - 1 \neq n - 1$. Given $\mathbf{x}$, one may compute the vector $\mathbf{h}_{\mathbf{x}}$ by a fan-out algorithm in $O(\log^2 n)$ ops per node point [AHU74], [BM75], [BP94], p.25.

Remark 2.1. We assume $\mathbf{h_x}$ being a part of the input (and similarly with $\mathbf{h_y}$ for Problem 3.1 of section 3) for in many practical computations $\mathbf{x}$ is fixed while $\mathbf{c}$ and $\mathbf{v}$ vary. We will use such assumptions only in section 7, where we decrease the complexity estimates below the level of $\log^2 n$ per node.

The classical numerically stable algorithms solve Problems 2.1 and 2.2 by using order of n ops per node [CdB80]. More recent algorithms solve both problems by using order of $\log^2 n$ ops per node [AHU74], [BM75], [BP94], page 25, which is within factor $\log n$ from a lower bound of [Str73]. Recursive application of polynomial division, however, makes these algorithms *numerically unstable*, that is, their output may be easily contaminated completely by small input or roundoff errors (cf. [PZHY97]). This motivates the search for numerically stable algorithms for p.e./p.i. that use $o(n)$ ops per node point, which will be one of our subjects in section 7.

3 Problems of Rational Evaluation and Interpolation

In rational evaluation (r.e.) and interpolation (r.i.) the input/output polynomial $c(x)$ of p.e./p.i. is replaced by a ratio

$$r(x) = u(x)/v(x),\ u(x) = \sum_{j=0}^{n-1} u_j x^j,\ v(x) = x^n + \sum_{i=0}^{n-1} v_i x^i. \tag{3.1}$$

If the poles of $r(x)$ are simple and available, it is convenient to state the problems of r.e./r.i. based on the following customary decomposition:

$$r(x) = \sum_{j=0}^{n-1} \frac{s_j}{x - y_j}. \tag{3.2}$$

Problem 3.1, *polarized r.e.* (*Trummer's problem*).

Input: a *node vector* $\mathbf{x} = (x_i)_{i=0}^{n-1}$ and a *pole vector* $\mathbf{y} = (y_j)_{j=0}^{n-1}$, with $2n$ distinct entries; the vector $\mathbf{h_y}$ and a vector $\mathbf{s} = (s_j)_{j=0}^{n-1}$.

Output: the vector $\mathbf{t} = (t_i)_{i=0}^{n-1}$, where $t_i = r(x_i) = \sum_{j=0}^{n-1} s_j/(x_i - y_j)$, $i = 0, \ldots, n-1$.

Problem 3.2, *polarized r.i.*

Input: two vectors $\mathbf{x}$ and $\mathbf{y}$ (as in Problem 3.1) and a vector $\mathbf{t} = (t_i)_{i=0}^{n-1}$.

Output: the vector $\mathbf{s} = (s_j)_{j=0}^{n-1}$ satisfying $t_i = r(x_i) = \sum_{j=0}^{n-1} s_j/(x_i - y_j)$, $i = 0, \ldots, n-1$.

Problems 3.1 and 3.2 represent approximation of complex and real functions by rational functions and allow fast and numerically stable solution of r.e./r.i. problems as well as their natural and widely used matrix generalizations and extensions. Problems 3.1 and 3.2 are also most customary forms for representation of r.e./r.i. used in the study of particle simulation, the n-body problem of celestial mechanics, integral equations, evaluation of Riemann's zeta function and conformal maps (see some bibliography in [PACLS98]).

Numerically stable solution of each of Problems 3.1 and 3.2 requires $2n-1$ ops per node. Order of $\log^2 n$ ops per node can be achieved at the expense of losing numerical stability [BP94], pp. 129-131.

In the rest of this section, we will study r.e. and r.i. based on (3.1).

Problem 3.3, *r.e.*

Input: three vectors, $\mathbf{u} = (u_j)_{j=0}^{n-1}$, $\mathbf{v} = (v_j)_{j=0}^{n-1}$ and $\overline{\mathbf{x}} = (x_i)_{i=0}^{2n-1}$.

Output: the vector $\mathbf{t} = (t_i)_{i=0}^{2n-1}$, where $t_i = r(x_i)$, $i = 0, \ldots, 2n-1$, for $r(x)$ of (3.1).

Problem 3.4, *r.i.*

Input: two vectors, $\mathbf{t}$ and $\overline{\mathbf{x}}$.

Output: the vectors $\mathbf{u}$ and $\mathbf{v}$ such that $t_i = r(x_i)$, $i = 0, \ldots, 2n-1$, for $r(x)$ of (3.1).

We evaluate $u(x_0)$, $v(x_0), \ldots, u(x_{2n-1}), v(x_{2n-1})$ (cf. Problem 2.1) to obtain the next result.

Theorem 3.1. *Problem 3.3 can be solved in $O(\log^2 n)$ ops per node.*

The next simple algorithm uses $O(\log^2 n)$ ops per node to reduce Problem 3.1 to Problem 3.3 (w.l.o.g. assume that $n = 2^h$).

Algorithm 3.1,

Input: two vectors $\mathbf{s}$ and $\mathbf{y}$.

Output: the vectors $\mathbf{u}$ and $\mathbf{v}$ such that (3.1) and (3.2) define the same rational function $r(x)$.

Computations:

Initialization: write $u_j^{(0)}(x) = s_j$, $v_j^{(0)}(x) = x - y_j$, $r_j^{(0)}(x) = u_j^{(0)}(x)/v_j^{(0)}(x)$, $j = 0, \ldots, n-1$.

Stage g, g=1, ..., h: compute the coefficients of the polynomials $u_j^{(g)}(x)$, $v_j^{(g)}(x)$, whose ratios $r_j^{(g)}(x) = u_j^{(g)}(x)/v_j^{(g)}(x)$ satisfy the equations $r_j^{(g)}(x) = r_{2j-1}^{(g-1)}(x) + r_{2j}^{(g-1)}(x)$, $j = 0, 1, \ldots, n/2^g$. Finally, write $r(x) = r_0^{(h)}(x) = u(x)/v(x)$, where the polynomials $u(x)$ and $v(x)$ satisfy (3.1) and (3.2), and output the vectors $\mathbf{u}$ and $\mathbf{v}$ of their coefficients.

For the converse transition from Problem 3.3 to Problem 3.1, we need to have the vector $\mathbf{y}$ of n distinct roots of the polynomial $v(x)$ of (3.1). The computational cost of the known best polynomial rootfinders is a little higher than one for r.e./r.i. [P95a], [P96], but, as mentioned, in many applications of r.e./r.i., the vector $\mathbf{y}$ is given. In this case the transition is simply via p.e. because $s_j = u(y_j)/v'(y_j)$, $j = 0, \ldots, n-1$. (Indeed, multiply by $v(x)$ both sides of the equation $\sum_j \frac{s_j}{x-y_j} = \frac{u(x)}{v(x)}$, to cancel the denominators, substitute $x = y_j$ into the resulting equation, and observe that $v(y_i)/(y_i - y_j)$ equals 0 for $i \neq j$ and $v'(y_j)$ for $i = j$.)

4 Modified Problems of Polynomial Evaluation and Interpolation

The format of Problems 2.1 and 2.2 for p.e. and p.i. is appropriate for computations over finite fields (polynomial factorization, coding and decoding) and where the vector $\mathbf{x}$ is formed by the (scaled) roots of 1, which turns Problems 1.1 and 1.2 into forward and inverse discrete Fourier transforms, effectively solved by FFT in $O(n \log n)$ ops, that is, in $O(\log n)$ ops per node point (see [BP94], pp. 9-10, on the cost estimates, and pp. 252-256, on the numerical stability of FFT). In many practical computations, however, one needs p.e./p.i. with real node vectors $\mathbf{x}$, and then the problems (in particular, p.i.) immediately become severely ill-conditioned. To overcome this difficulty, the customary recipe is to generalize Problems 2.1 and 2.2 by writing

$$c(x) = c_{\mathbf{P}}(x) = \sum_{j=0}^{n-1} c_j P_j(x) \tag{4.1}$$

where the n components $P_0(x), \ldots, P_{n-1}(x)$ of the *bases vector* $\mathbf{P} = (P_j(x))_{j=0}^{n-1}$ form a bases in the linear space of polynomials of degree at most $n-1$ such as ones made of Chebyshev polynomials of the first or the second kind,

$$P_j(x) = T_j = T_j(x) = 2^{-j} \cos(j \ \mathrm{arc\cos}\ x), \tag{4.2}$$

$$P_j(x) = U_j = U_j(x) = \sin((j+1) \ \mathrm{arc\cos}\, x) / \sin(\mathrm{arc\cos}\, x). \tag{4.3}$$

(4.2), (4.3) are the most customary choices for p.e./p.i. on real intervals where the real functions $\sum_{j=0}^{\infty} c_j P_j(x)$ are most effectively approximated by partial sums (4.1) under these choices of $P_j(x)$ [SB80], [Ri90].

Fast cosine/sine transforms (hereafter, referred to as *FCT/FST*) replace FFT wherever bases (4.2) and (4.3) replace the monomial bases, and this is reflected by rapid growth of the current library of algorithms based on FCT/FST [Po96], [SN96]. Generalized p.e./p.i. problems cover bases (4.2) and (4.3) as well as the monomial bases $\{P_j(x) = x^j\}$.

Problem 4.1, *generalized p.e.*

Input: vectors $\mathbf{P} = (P_j(x))_{j=0}^{n-1}, \mathbf{x}$, and $\mathbf{c_P} = (c_j)_{j=0}^{n-1}$.

Output: the vector $\mathbf{v} = (v_i)_{j=0}^{n-1}$, where $v_i = c_{\mathbf{P}}(x_i) = \sum_{j=0}^{n-1} c_j P_j(x_i)$, $i = 0, \ldots, n-1$.

Problem 4.2, *generalized p.i.*

Input: vectors $\mathbf{x}, \mathbf{P}$, and $\mathbf{v} = (v_i)_{i=0}^{n-1}$.

Output: the vector $\mathbf{c_P} = (c_j)_{j=0}^{n-1}$ such that $c_{\mathbf{P}}(x_i) = \sum_{j=0}^{n-1} c_j P_j(x_i) = v_i$, $i = 0, \ldots, n-1$.

5 Matrix Versions of the Problems and the Mappings of the Matrices and the Problems

Hereafter, diag $(m_i)_{i=0}^{n-1}$ denotes the $n \times n$ diagonal matrix with diagonal entries $m_0, \ldots, m_{n-1}$. $V(\mathbf{x}) = (x_i^j)_{i,j=0}^{n-1}$, $V_{\mathbf{P}}(\mathbf{x}) = (P_j(x_i))_{i,j=0}^{n-1}$ and $C(\mathbf{x}, \mathbf{y}) = (\frac{1}{x_i - y_j})_{i,j=0}^{n-1}$ denote the $n \times n$ *Vandermonde, polynomial Vandermonde* and *Cauchy matrices*, respectively, defined by vectors $\mathbf{P}, \mathbf{x}$ and $\mathbf{y}$. Problems 2.1, 2.2, 3.1, 3.2, 4.1 and 4.2 amount to computation of the vectors $\mathbf{v} = V(\mathbf{x})\mathbf{c}$, $\mathbf{c} = V^{-1}(\mathbf{x})\mathbf{v}$, $\mathbf{v} = V_{\mathbf{P}}(\mathbf{x})\mathbf{c_P}$, $\mathbf{c_P} = V_{\mathbf{P}}^{-1}(\mathbf{x})\mathbf{v}$, $\mathbf{t} = C(\mathbf{x}, \mathbf{y})\mathbf{s}$ and $\mathbf{s} = C^{-1}(\mathbf{x}, \mathbf{y})\mathbf{t}$, respectively.

Matrix representation facilitates both solution and analysis of the cited problems. Our main tool will be the mappings of structured matrices, which is a recipe first proposed for p.e. and p.i. in [PLST93], [PZHY97] as an extension of the general approach of [P90] to the unification and improvement of algorithms for various classes of structured matrices by means of mapping such classes of matrices into each other. We will rely on the next formula, due to [BKO,a] (earlier known for the special case of $\mathbf{P} = (x^j)_{j=0}^{n-1}$ [FHR93], [BP94], p.131):

Theorem 5.1. *Under the previous notation, let two vectors* $\mathbf{x}$ *and* $\mathbf{y}$*, each of dimension* n*, have* $2n$ *distinct components and let* $h_{\mathbf{y}}(x) = \prod_{j=0}^{n-1}(x-y_j)$, $h'_{\mathbf{y}}(x) = dh_{\mathbf{y}}(x)/dx = \sum_{i=0}^{n-1}\prod_{j\neq i}(x-y_j)$. *Then*

$$V_{\mathbf{P}}(\mathbf{x}) = \{\mathrm{diag}(h_{\mathbf{y}}(x_i))_{i=0}^{n-1}\}C(\mathbf{x},\mathbf{y})\{\mathrm{diag}(1/h'_{\mathbf{y}}(y_j))_{j=0}^{n-1}\}V_{\mathbf{P}}(\mathbf{y}). \tag{5.1}$$

The next corollaries are due to [PACLS98], [PACPS98]:

Corollary 5.1 *Under the assumptions of Theorem 5.1, we have*

$$V_{\mathbf{P}}^{-1}(\mathbf{x}) = V_{\mathbf{P}}^{-1}(\mathbf{y})\{\mathrm{diag}(h_{\mathbf{x}}(y_i))_{i=0}^{n-1}\}C(\mathbf{y},\mathbf{x})\{\mathrm{diag}(1/h'_{\mathbf{x}}(x_j))_{j=0}^{n-1}\}. \tag{5.2}$$

Corollary 5.2 *Let* $\mathbf{x}$*,* $\mathbf{y}$*,* $\mathbf{z}$ *be three vectors of dimension* n *having* $3n$ *distinct components. Then*

$$C(\mathbf{x},\mathbf{y}) = \{\mathrm{diag}(\frac{h_{\mathbf{z}}(x_i)}{h_{\mathbf{y}}(x_i)})_{i=0}^{n-1}\}C(\mathbf{x},\mathbf{z})\{\mathrm{diag}(\frac{h_{\mathbf{y}}(z_j)}{h'_{\mathbf{z}}(z_j)})_{j=0}^{n-1}\}C(\mathbf{z},\mathbf{y}). \tag{5.3}$$

Corollaries 5.1 and 5.2 enable various reductions of Problems 2.1, 2.2, 3.1-3.4, 4.1 and 4.2 to each other. In section 7, we will show some promising extensions of the known solution algorithms for p.e./p.i. based on such reductions. Next, we will use a matrix version of Problem 3.4, where $V(\overline{\mathbf{x}}) = (x_i^j)_{i=0,j=0}^{2n-1,n-1}$, $V(\overline{\mathbf{x}},\mathbf{t}) = (V(\overline{\mathbf{x}}), \{\mathrm{diag}(t_i)_{i=0}^{2n-1}\}V(\overline{\mathbf{x}}))$, $V(\overline{\mathbf{x}},\mathbf{t})$ is a $2n\times 2n$ matrix.

Theorem 5.2. *Problem 3.4 can be written as a linear system of equations* $V(\overline{\mathbf{x}},\mathbf{t})\binom{\mathbf{u}}{-\mathbf{v}} = (t_i x_i^n)_{i=0}^{2n-1}$. $O(n\log^2 n)$ *ops suffice to test whether the matrix* $V(\overline{\mathbf{x}},\mathbf{t})$ *is nonsingular and, if so, to compute the solution* $\mathbf{u}$, $\mathbf{v}$.

Proof. The vector equation is verified by inspection. Now, observe that the matrix $V(\overline{\mathbf{x}},\mathbf{t})$ is Vandermonde-like, having F-rank at most 2, according to the definitions of [BP94]. Apply the algorithms of [P90], [P99] to test if the matrix $V(\overline{\mathbf{x}},\mathbf{t})$ is nonsingular and, if so, to solve the above linear system in $\mathbf{u},\mathbf{v}$. *Q.E.D.*

6 Effective Choices of the Node Vectors and the Fast Exact Solution Algorithms for Generalized p.e. and p.i.

Let $\mathbf{P} = (\mathbf{x}^j)_{j=0}^{n-1}$ be the monomial bases, fix a proper scalar a, and let $\mathbf{y}$ be a scaled vector of the n-th roots of 1, $\mathbf{y} = a\mathbf{w}$, $\mathbf{w} = (w^j)_{j=0}^{n-1}$, $w = w(n) =$

$exp(2\pi\sqrt{-1}/n)$. In this case, $h_{\mathbf{y}}(x) = x^n - a^n$, $h'_{\mathbf{y}}(x) = nx^{n-1}$, and the values $h_{\mathbf{y}}(x_i) = x_i^n - a^n$ and $h'_{\mathbf{y}}(y_j) = ny_j^{n-1}$ can be immediately computed for all i and j in $O(n \log n)$ ops, $V_{\mathbf{P}}(\mathbf{y})$ and $V_{\mathbf{P}}^{-1}(\mathbf{y})$ are the matrices of the scaled forward and inverse discrete Fourier transform (DFT), which can be multiplied by a vector in $O(n \log n)$ ops (cf. [BP94], pp. 9-10).

Dealing with Chebyshev bases (4.2), (4.3), we choose node vectors $\mathbf{y}$ such that $V_{\mathbf{P}}(\mathbf{y})$ turns into one of the matrices of DCT or DST (see [Po96], [SN96], [KO96]). Such a matrix can be multiplied by a vector in $O(n \log n)$ ops. As an immediate application, we improve the known complexity bounds for Problems 4.1, 4.2 from order of n [Ri90], [SB80] to $O(\log^2 n)$ per node.

Theorem 6.1 *[OP99]. Under (4.2), (4.3), Problems 4.1 and 4.2 can be solved by using $O(n \log^2 n)$ ops, that is, $O(\log^2 n)$ per node point.*

Proof By Theorem 5.1 and Corollary 5.1, each of Problems 4.1 and 4.2 can be reduced to multiplication of the matrices $V_{\mathbf{P}}(\mathbf{y})$, $V_{\mathbf{P}}^{-1}(\mathbf{y})$, $C(\mathbf{x}, \mathbf{y})$ and/or $C(\mathbf{y}, \mathbf{x})$ by a vector and to two Problems 2.1. By choosing $\mathbf{y}$ from a table of [KO96] (up to scaling), we reduce the multiplications of $V_{\mathbf{P}}(\mathbf{y})$ and $V_{\mathbf{P}}^{-1}(\mathbf{y})$ by vectors to DCTs/DSTs and perform them in $O(n \log n)$ ops. The scaling of vector $\mathbf{y}$ keeps its components distinct from ones of $\mathbf{x}$, so that the matrix $C(\mathbf{x}, \mathbf{y})$ is well-defined. The known algorithms (cf. sections 2 and 3) use $O(n \log^2 n)$ ops to support the other steps of the computation. *Q.E.D.*

Corollary 6.1 *Given a polynomial $c(x)$ in monomial bases, its representations in the bases of (4.2) or (4.3) can be computed in $O(n \log^2 n)$ ops (or $O(\log^2 n)$ per node); $O(n \log^2 n)$ ops also suffice for the converse transition.*

Proof. Fix any node vector $\mathbf{x}$ and evaluate $c(x)$ on $\mathbf{x}$ by using the representation of $c(x)$ in the given bases. Then interpolate to $c(x)$ expressed in the new bases. Apply Theorem 6.1 and the known fast algorithms for p.e./p.i. in the monomial bases (cf. [BP94], section 1.4). *Q.E.D.*

7 Approximation Algorithms for the Evaluation and Interpolation

Some approximation algorithms for Problems 2.1, 2.2, and 3.1 working under certain restrictions on the input decrease the computational cost estimates versus the exact solution algorithms. [R88] uses $O(u^3)$ ops per node to approximate the output vector $\mathbf{v}$ of Problem 2.1 with the error bound $2^{-u} \sum_{i=0}^{n-1} |c_i|$. [P95] yields the cost bound $O(\log^2 u + \min\,(u, \log n))$. Both algorithms assume the node set on a fixed real interval, rely on the real approximation theorems, and extend neither to p.e. with nonreal input nodes nor to p.i.

The state of the art of the problem of approximate p.e. is reflected in [W98], where *p.e. on a set of points on a fixed circle* is viewed as an extension of DFT. Even in this simplest case further restrictions on the input are required to improve the known cost bound of $O(\log^2 n)$ ops per node.

Let us relax such restrictions by projecting the variable x onto the real line to yield a fast approximation algorithm. We assume Problem 2.1 of p.e. where all input nodes x_i lie on the unit circle, that is, $|x_i| = 1$ for all i, and where, w.l.o.g., the input coefficient vector $\mathbf{c}$ is real. Express the complex variable x as $x = y + \sqrt{y^2 - 1}$. As x ranges on the unit circle, $x = e^{2\pi i \phi}, i = \sqrt{-1}$, we have real $y = \mathrm{Re}\ x = \cos\ \phi$, ranging between -1 and 1. As x ranges among the k-th roots of 1, y ranges among the Chebyshev real nodes lying in the interval from -1 to 1. We also have

$$c(x) = c_0(y) + c_1(y)\sqrt{y^2 - 1}, \tag{7.1}$$

where $c_0(y)$ and $c_1(y)$ are two polynomials of degrees at most $n - 1$; $c_0(y) = \mathrm{Re}\ c(y), c_1(y) = \mathrm{Im}\ c(y)/|\sqrt{y^2 - 1}|$, for $-1 \leq y \leq 1$. Now, for $k \geq n$, evaluate $c(x)$ at the k-th roots of 1 (DFT). By (7.1), this defines $c_0(y)$ and $c_1(y)$ on the Chebyshev nodes. By interpolation, recover $c_0(x)$ and $c_1(x)$. The entire computation costs $O(k \log k)$ ops. Then apply the approximation algorithms of [R88] or [P95] to evaluate $c_0(y)$ and $c_1(y)$ on the set $\{y_i\} = \{\mathrm{Re}\ x_i\}$ Finally, apply (7.1) to evaluate $c(x)$ on the input set $\{x_i\}$.

Let us next follow [PZHY97] to solve both Problems 2.1 and 2.2 of p.e./p.i. approximately (for an arbitrary complex node vector $\mathbf{x}$), by means of the reduction to Trummer's problem based on (5.1) and (5.2). Choose $\mathbf{y} = a\mathbf{w}$, for a scalar a and the vector $\mathbf{w}$ of the n-th roots of 1, which turns $V(\mathbf{y})$ of (5.1) and $V^{-1}(\mathbf{y})$ of (5.2) into the matrices of scaled forward and inverse DFT, respectively. Now, the values $h_{\mathbf{y}}(x_i)$ and $h'_{\mathbf{y}}(y_j)$ can be computed in $O(n \log n)$ ops for all i and j, and, clearly, n ops suffice to multiply a diagonal matrix by a vector. Therefore, the overall cost is $O(n \log n)$ ops, that is, $O(\log n)$ ops per point (in terms of n).

We choose the scalar a to separate the components of $\mathbf{y}$ from ones of $\mathbf{x}$. Application of the above algorithm to Problem 2.2 (cf. (5.2)) involves Problem 2.1, at the stage where one computes $h'_{\mathbf{x}}(x_j)$, for $j = 0, \ldots, n - 1$.

Instead of the vector $\mathbf{y} = a\mathbf{w}$, which represents a scaled set of the n-th roots of 1, one may choose $\mathbf{y}$ ranging in a scaled Chebyshev set. In this case only real values are involved into the computations when the input is real.

The proposed transformations reduce p.e./p.i. essentially to Trummer's problem. Fast and numerically stable Multipole algorithms [GR87], [CGR88] give its approximate solution. The initial results of numerical experiments in [PZHY97] show that this approach is indeed effective in the case of p.e.

8 Discussion

The proposed techniques of matrix transformations for polynomial and rational evaluation and interpolation promise several further effective extensions, such as the extension of Theorem 6.1 to Problem 3.2 based on an appropriate matrix equation easily deduced from (5.1) in [OP99].

Among the important problems closely related to p.e./p.i., the multiplication of the transpose of a Vandermonde matrix and the inverse of such a transpose by a vector can be easily treated by extending our techniques.

As an example of possible modifications of our algorithms, one may exploit the next alternative to (5.2) free of $h'_{\mathbf{x}}(x_i)$:

$$V_{\mathbf{P}}^{-1}(\mathbf{x}) = V_{\mathbf{P}}^{-1}(\mathbf{y})\{\mathrm{diag}(\frac{h'_{\mathbf{y}}(y_i)h_{\mathbf{y}}(x_i)}{h'_{\mathbf{x}}(y_i)})_{i=0}^{n-1}\}C(\mathbf{y},\mathbf{x})\{\mathrm{diag}(\frac{h_{\mathbf{x}}(y_j)}{h_{\mathbf{y}}(x_j)h'_{\mathbf{y}}(x_j)})_{j=0}^{n-1}\}.$$

References

[AHU74] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass, 1974.

[BKO,a] T. Boros, T. Kailath, V. Olshevsky, Pivoting and Backward Stabillity of Fast Algorithms for Solving Cauchy Linear Systems (submitted).

[BM75] A. Borodin, I. Munro, *The Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, New York, 1975.

[BP94] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[CdB80] C.D. Conte, C. de Boor, *Elementary Numerical Analysis: an Algorithmic Approach*, McGraw-Hill, New York, 1980.

[CGR88] J. Carier, L. Greengard, V. Rokhlin, A Fast Adaptive Multipole Algorithm for Particle Simulation, *SIAM J. Sci. Stat. Comput.*, **9**, 669–686, 1988.

[FHR93] T. Fink, G. Heinig, K. Rost, An Inversion Formula and Fast Algorithms for Cauchy-Vandermonde Matrices, *Linear Algebra Appl.*, **183**, 179–191, 1993.

[G86] J. von zur Gathen, Parallel Arithmetic Computations: a Survey, *Lecture Notes in Computer Science*, **233**, 93-112, Springer, Berlin, 1986.

[GR87] L. Greengard, V. Rokhlin, A Fast Algorithm for Particle Simulation, *J. of Comput. Physics,* **73**, 325-348, 1987.

[KO96] T. Kailath, V. Olshevsky, Displacement Structure Approach to Discrete-Trigonometric-Transform Based Preconditioners of G. Strang and of T. Chan, *Calcolo*, **33**, 191-208, 1996.

[KR90] R. Karp, V. Ramachandran, A Survey of Parallel Algorithms for Shared Memory Machines, *Handbook for Theoretical Computer Science* (J. van Leeuwen editor), 869–941, North Holland, Amsterdam, 1990.

[Le92] F. T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees & Hypercubes, *Morgan Kaufmann*, San Mateo, CA 1992.

[OP98] V. Olshevsky, V. Y. Pan, A Unified Superfast Algorithm for Boundary Rational Tangential Interpolation Problem, *Proc. 39th Ann. IEEE Symp. Foundations of Comp. Sci.,* 192-201, IEEE Comp. Soc. Press, 1998.

[OP99] V. Olshevsky, V. Y. Pan, Polynomial and Rational Interpolation and Multipoint Evaluation with Application of Structured Matrices, preprint, 1999 (submitted).

[P90] V. Y. Pan, Computations with Dense Structured Matrices, *Math. of Computation*, **55**, **191**, 179–190, 1990.

[P95] V. Y. Pan, An Algebraic Approach to Approximate Evaluation of a Polynomial on a Set of Real Points, *Advances in Computational Mathematics*, **3**, 41–58, 1995.

[P95a] V. Y. Pan, Optimal (up to Polylog Factors) Sequential and Parallel Algorithms for Approximating Complex Polynomial Zeros, *Proc. 27th Ann. ACM Symposium on Theory of Computing,* 741–750, ACM Press, New York, 1995.

[P96] V. Y. Pan, Optimal and Nearly Optimal Algorithms for Approximating Polynomial Zeros, *Computers & Math. (with Applications)*, **31**, **12**, 97–138, 1996.

[P99] V. Y. Pan, Superfast Divide-and-Conquer Algorithm for Structured Matrices, preprint, 1999 (submitted).

[PACLS98] V. Y. Pan, M. AbuTabanjeh, Z. Chen, E. Landowne, A. Sadikou, New Transformations of Cauchy Matrices and Trummer's Problem, *Computer and Math. (with Applics.)*, **35, 12**, 1-5, 1998.

[PACPS98] V. Y. Pan, M. AbuTabanjeh, Z. Chen, S. Providence, A. Sadikou, Transformations of Cauchy Matrices for Trummer's Problem and a Cauchy-like Linear Solver, *Proc. of 5th Annual International Symposium on Solving Irregularly Structured Problems in Parallel*, *(Irregular98*), (A. Ferreira, J. Rolim, H. Simon, S.-H. Teng Editors), *Lecture Notes in Computer Science*, **1457**, 274-284, Springer, 1998.

[PLST93] V. Y. Pan, E. Landowne, A. Sadikou, O. Tiga, A New Approach to Fast Polynomial Interpolation and Multipoint Evaluation, *Computers & Math. (with Applications)*, **25, 9**, 25–30, 1993.

[Po96] B. Porat, *A Course in Digital Signal Processing,* Wiley, New York, 1996.

[PZHY97] V. Y. Pan, A. Zheng, X. Huang, Y. Yu, Fast Multipoint Polynomial Evaluation and Interpolation via Computations with Structured Matrices, *Annals of Numerical Math.*, **4**, 483–510, January 1997.

[R88] V. Rokhlin, A Fast Algorithm for the Discrete Laplace Transformation, *J. of Complexity*, **4**, 12-32, 1988.

[Ri90] T. Rivlin, *Chebyshev Polynomials*, Wiley, New York, 1990.

[SB80] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, Springer, New York, 1980.

[SN96] G. Strang, T. Nguyen, *Wavelet and Filterbanks,* Wellesley-Cambridge Press, Cambridge, Mass., 1996.

[Str73] V. Strassen, Die Berechnungskomplexetät von elementarysymmetrischen Funktionen und von Interpolationskoeffizienten, *Numerische Mathematik*, **20, 3**, 238-251.

[W98] A. F. Ware, Fast Approximate Fourier Transforms for Irregularly Spaced Data, *SIAM Review*, **40, 4**, 838-856, 1998.