Algorithms For Sparse Rational Interpolation

Dima Yu. Grigoriev *
Dept. of Computer Science
University of Bonn
5300 Bonn 1

Marek Karpinski †
Dept. of Computer Science
University of Bonn
5300 Bonn 1
and
International Computer Science Institute
Berkeley, California

Abstract

We present two algorithms for interpolating sparse rational functions. The first is the interpolation algorithm in a sense of sparse partial fraction representation of rational functions. The second is the algorithm for computing the *entier* and the *remainder* of a rational function. The first algorithm works without apriori known bound on the degree of a rational function, the second one is in the parallel class NC provided that the degree is known. The presented algorithms complement the sparse interpolation results of [GKS 90].

1 Introduction

We address a question of computational complexity of sparse rational interpolation and a connected question of algebraic manipulation of sparse rational functions. We study the most general method of representating rational functions by black boxes (cf. [KT 88, GKS 90]) and restrict ourselves in this paper to the univariate case only. For the technical developments which lead to this paper see [GKS 91, GKS 90, DG 91]. For the hardness results on sparse polynomial divisibility see also Plaisted [P 77b].

We present two algorithms. For the first one we consider the partial-fraction representation of a rational function and the corresponding notion of sparsity as the number of terms in this representation. An algorithm is designed for finding partial-fraction representation without knowing the degree. Of independent interest could be also construction of a new code (see Section 1), which is a generalization of Goppa and BCH code (cf. [MS 81]). The second algorithm finds an entier of a rational function, and a polynomial part of a partial-fraction representation. We show that finding an entier is in the parallel class NC (cf. [KR 90]) provided that the degree of a rational function is known. Here we measure the complexity in the combined sizes

*On leave from Steklov Institute of Mathematics, Soviet Academy of Sciences, Leningrad 191011, supported in part by the Max-Planck Institute of Mathematics, Bonn

†Supported in part by the Leibniz Center for Research in Computer Science, by the DFG Grant KA 673/4-1 and by the SERC Grant GR-E 68297

of the input and output. As a subroutine we apply the approximative analogue of a sparse polynomial interpolation schemes ([GK 87, BT 88]).

2 Extending BCH and Goppa-codes by involving multiple roots

We shall refer to [MS 81] for the general terminology on the coding theory. Assume that a polynomial $f \in \mathbb{Z}[Y]$ is unknown and its degree $\deg f = d$ is also unknown. In addition let $(\alpha_1, \ldots, \alpha_d)$ be an unknown vector. Denote $f = \prod_i (Y - c_i)^{\beta_i} = \sum_{0 \le i \le d} f_i Y^i$. Suppose also that we can compute the following *syndrome* expression (cf. [MS 81]) $g_k = \alpha_1 c_1^k + \alpha_2 k c_1^{k-1} + \alpha_3 k(k-1) c_1^{k-2} + \cdots + \alpha_{\beta_1} k(k-1)(k-2)\cdots(k-\beta_1+2) c_1^{k-\beta_1+1} + \alpha_{\beta_1+1} c_2^k + \alpha_{\beta_1+2} k c_2^{k-1} + \cdots$ for $k = 0, 1, \ldots$ where β_i summands correspond to c_i. The computational problem now is to recover f and the vector $(\alpha_1, \ldots, \alpha_d)$.

For an arbitrary $l \ge 0$ consider $(d+1)\times(d+1)$ Töplitz matrix

$$\bar{G}_l = \begin{pmatrix} g_l & g_{l+1} & \cdots & g_{l+d} \\ g_{l+1} & g_{l+2} & \cdots & g_{l+d+1} \\ \vdots & \vdots & & \vdots \\ g_{l+d} & g_{l+d+1} & \cdots & g_{l+2d} \end{pmatrix}$$

and by G_l denote its $d \times d$ submatrix obtained by deleting the last row and the last column. Consider also $d \times d$ matrix (being block-diagonal) $A =$

$$\begin{pmatrix} \boxed{\begin{matrix} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{\beta_1} \\ \alpha_2 & \alpha_3 & & \cdot & \\ \alpha_3 & & \cdot & & \\ \vdots & \cdot & & 0 & \\ \alpha_{\beta_1} & & & & \end{matrix}} & 0 & \\ 0 & \boxed{\begin{matrix} \alpha_{\beta_1+1} & \alpha_{\beta_1+2} \\ \alpha_{\beta_1+2} & \end{matrix}} & \\ & & \ddots \end{pmatrix}$$

and $d \times (d+1)$ matrix $\bar{C}_l =$

$$\begin{pmatrix} c_1^l & c_1^{l+1} & \cdots & c_1^{l+d} \\ l c_1^{l-1} & (l+1) c_1^l & \cdots & (l+d) c_1^{l+d-1} \\ \vdots & & & \\ l(l-1)\ldots(l-\beta_1+2) c_1^{l-\beta_1+1} & & \cdots & \\ c_2^l & c_2^{l+1} & \cdots & c_2^{l+d} \\ \vdots & & & \end{pmatrix}$$

namely, the second row is the derivative of the first one, the next is the derivative of the previous etc. β_1 times, thus β_1 rows correspond to c_1, then β_2 rows correspond to c_2 etc. Denote by C_l $d \times d$ matrix obtained from $\bar{C}_l$ by deleting the last column.

Then $\bar{C}_l \begin{pmatrix} f_0 \\ \vdots \\ f_d \end{pmatrix} = 0$, since denote $\bar{C}_l \begin{pmatrix} f_0 \\ \vdots \\ f_d \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_d \end{pmatrix}$, then $b_1 = c_1^l f(c_1) = 0, b_2 = \frac{d}{dc_1}(c_1^l f(c_1)) = l c_1^{l-1} f(c_1) + c_1^l f'(c_1) = 0, \ldots, b_{\beta_1} = \frac{d^{\beta_1 - 1}}{dc_1^{\beta_1 - 1}}(c_1^l f(c_1)) = 0$, and so on. On the other hand $\det(C_l) \ne 0$, provided that $c_1 \ne 0, c_2 \ne 0, \ldots$. Indeed, assume that $C_l \begin{pmatrix} h_0 \\ \vdots \\ h_{d-1} \end{pmatrix} = 0$, then denote $h = \sum h_i Y^i$ and $0 = h(c_1) = h'(c_1) = \cdots = h^{(\beta_1 - 1)}(c_1)$, $0 = h(c_2) = \cdots = h^{(\beta_2 - 1)}(c_2), \ldots$, contradiction. The latter arguing is known in the numerical analysis by considering Hermite's interpolation.

Then $\bar{G}_{l_1+l_2} = (\bar{C}_{l_1})^T A \bar{C}_{l_2}$, $G_{l_1+l_2} = (C_{l_1})^T A C_{l_2}$. Hence if $\alpha_{\beta_1} \ne 0$ and all other coefficients α_β which correspond to the highest derivatives, are distinct from zero, then $\det(A) \ne 0$ and therefore $\det(G_l) \ne 0$. Because of that $\mathrm{rg}(\bar{G}_l) = d$ and the linear system $\bar{G}_l Z = 0$ has a unique solution $Z = \begin{pmatrix} f_0 \\ \vdots \\ f_d \end{pmatrix}$. Thus, one can recover f by solving a linear system $\bar{G}_0 Z = 0$, hence $c_1, \ldots, \beta_1, \ldots$, by polynomial factoring [CG 82], representing $c_1,\ldots,$ as the roots of the irreducible over $\mathbb{Q}$ polynomials.

Finally, one can find $\alpha_1,\ldots,\alpha_d$ by solving a linear system

$$(\alpha_1,\ldots,\alpha_d)C_0=(g_0,\ldots,g_{d-1}).$$

Remark that for pairwise distinct $c_1,\ldots,c_d$ the described code converts into Goppa code [MS 81].

Note: If we take a Töplitz matrix

$$\begin{pmatrix} g_l & g_{l+1} & \cdots & g_{l+d_1} \\ g_{l+1} & g_{l+2} & \cdots & g_{l+d_1+1} \\ \vdots & \vdots & & \vdots \\ g_{l+d_1} & g_{l+d_1+1} & \cdots & g_{l+2d_1} \end{pmatrix}$$

for $d_1\geq d$ then its rank $=d$.

3 Partial-fraction sparsity of rational functions and finding highest terms

Let $f_1/f_2\in\mathbb{Q}[X]$ be a rational function given by a black-box. We assume that the black-box at every point (including ∞) gives a value of f_1/f_2 at this point (including ∞). And the same concerns any rational function which will appear at the intermediate calculations.

We suppose also that together with the black-box for f_1/f_2 we are supplied with a black-box for the derivative $(f_1/f_2)'$. If f_1/f_2 is given by a short straight-line program, then $(f_1/f_2)'$ can be represented also by a short straight-line program e.g. by virtue of [BS 83]. If (f_1/f_2) is given by a certain physical process, then also one can get $(f_1/f_2)'$.

With the help of $(f_1/f_2)'$ one can recover the highest term of f_1/f_2 at ∞. Namely, if $f_1/f_2=ax^m+O(x^{m-1})$, where $m\in\mathbb{Z}$, $a\neq 0$, then $x(f_1/f_2)'/(f_1/f_2)=m+O(x^{-1})$, so we recover m and then calculate in NC x^m and since $(f_1/f_2)/x^m=a+O(x^{-1})$, we recover a.

A rational function f_1/f_2 is uniquely represented as a sum of its partial fractions $f_1/f_2=P+\sum_i\frac{\alpha_{i,1}}{x-c_i}+\sum_i\frac{\alpha_{i,2}}{(x-c_i)^2}+\ldots$, where $P\in\mathbb{Q}[X]$ is a polynomial, $c_i,\alpha_{i,j}\in\overline{\mathbb{Q}}$. We call $P=[f_1/f_2]$ an entier of f_1/f_2 (see the last section). We call f_1/f_2 *t-sparse* if the number of nonzero terms in this representation is at most t. We'll assume in the sequel that f_1/f_2 is *t-sparse.* The problem we deal with is to find partial-fraction representation.

Firstly we find P term by term starting with the highest one. Thus, we can suppose that $f_1/f_2=\sum\frac{\alpha_{i,1}}{x-c_i}+\ldots$. Then $\mathrm{res}_\infty(f_1/f_2)=\sum_i\alpha_{i,1}$, and if it does not vanish then $\left(\sum_i\alpha_{i,1}\right)x^{-1}$ is the highest term. Thus, we can find $\mathrm{res}_\infty(f_1/f_2)$. Later on we'll calculate $g_k=\mathrm{res}_\infty x^k(f_1/f_2)$ for different k, we call them successive residues. Remark that $g_k=\sum_i\alpha_{i,1}c_i^k+\alpha_{i,2}kc_i^{k-1}+\cdots$, thus it coincides with the formula for g_k in the extended Goppa code (see the previous section).

Observe that if $(f_1/f_2)^{-1}$ is also *sparse* then one can recover both f_1/f_2 and f_2/f_1 by applying extended Goppa code (or even the usual Goppa code) to $(f_1/f_2)'/(f_1/f_2)=\sum\frac{m_i}{x-c_i}$ (being *sparse* by the same token) where m_i is the multiplicity of the pole c_i (when $m_i<0$) or of the root c_i (when $m_i>0$) of f_1/f_2. Thus, one can find c_i,m_i and considering expansions in the neighbourhood of c_i, to find (involving the procedure for recovering highest terms) the terms of the form $\frac{\alpha_{i,j}}{(x-c_i)^j}$.

4 A bound on the least nonzero successive residue

If $f_1/f_2=\sum\frac{\alpha_{i,k}}{(x-c_i)^k}+\cdots$ then we call k an order of f_1/f_2. Evidently $g_j=0$ for $j<k$. Let us estimate the least j_0 s.t. $g_{j_0}\neq 0$. Denote $\tilde{g}_{m-k+1}=g_m/\frac{m!}{(m-k+1)!}$ Then $\tilde{g}_{m-k+1}$ plays the role of g_{m-k+1} for the function $\sum\frac{\alpha_{i,k}}{(x-c_i)}+\sum\frac{\alpha_{i,k+1}}{(x-c_i)^2}+\cdots=\widetilde{(f_1/f_2)}=$

$\sum \frac{\tilde{\alpha}_{i,1}}{(x-c_i)} + \sum \frac{\tilde{\alpha}_{i,2}}{(x-c_i)^2} + \cdots$ in other words all the exponents in the denominators of partial fractions are diminished by $(k-1)$. Assume that $\tilde{g}_0 = \cdots = \tilde{g}_{N-1} = 0$ for some N. Consider any $N_1 \le N$. For any i denote by $d_i(N_1)$ the maximal $j < N_1$ s.t. $\tilde{\alpha}_{i,j} \ne 0$, and by $d(N_1) = \sum_i d_i(N_1)$.

We claim that $d(N_1) > N_1$. Indeed

$$\begin{pmatrix} \tilde{g}_0 \\ \vdots \\ \tilde{g}_{N_1-1} \end{pmatrix} = \tilde{C}(N_1) \cdot \begin{pmatrix} \tilde{\alpha}_{1,1} \\ \vdots \\ \tilde{\alpha}_{1,d_1(N_1)} \\ \tilde{\alpha}_{2,1} \\ \vdots \\ \tilde{\alpha}_{2,d_2(N_2)} \\ \vdots \end{pmatrix}$$

where the matrix

$$\tilde{C}(N_1) = \begin{pmatrix} 1 & & & 1 & & \\ c_1 & 1 & & c_2 & 1 & \\ c_1^2 & 2c_1 & \ddots & c_2^2 & 2c_2 & \ddots \\ \vdots & \vdots & & \vdots & \vdots & \end{pmatrix}$$

is similar to the matrix C_l (see the previous section), it has $d_1(N_1)$ columns which correspond to c_1, $d_2(N_1)$ columns which correspond to c_2, etc. If $d(N_1) \le N_1$, then the columns of the matrix $\tilde{C}(N_1)$ cannot be linearly dependent (see the previous section); that proves the claim.

Recall that the sequence $\tilde{\alpha}_{i,j}$ is t-*sparse* and let us find out how large is $N_0 = \max\{j : d(N_1) > \frac{N_1}{2}$ for any $N_1 \le j\}$, being a stronger property than is necessary in our case, but we will need it later in this stronger version. Let us prove that $N_0 \le 3^t$ by induction on t.

Assume the contrary. Then by inductive hypothesis in the segment $[0, 3^{t-1}]$ there are $t-1$ indices j such that $\tilde{\alpha}_{i,j} \ne 0$ for a suitable i and in the segment $(3^{t-1}, 3^t]$ there are no such indices. Again by inductive hypothesis for these indices $j_1 \le j_2 \le \cdots \le j_{t-1}$ holds $j_l \le 3^{l-1}$. Therefore $d(3^t) \le \frac{3^t}{2}$ that leads to the contradiction.

Thus, the order of f_1/f_2 is at least $N - 3^t$ where N is the least index for which $g_N \ne 0$, and we denote later $\tilde{g}_{s-N+3^t} = g_s / \frac{s!}{(s-N+3^t)!}$, also $\tilde{\alpha}_{i,j} = \alpha_{i,j+N-3^t}$.

5 Finding swarms of terms

We say that an integer N_2 creates a *swarm* of terms of the rational function f_1/f_2 if $0 < \tilde{d}(N_2) < \frac{N_2}{2}$, where $\tilde{d}(N_2) = \sum_i \tilde{d}_i(N_2) = \sum_i (d_i(N_2) - N + 3^t)$. In this case the rank of the matrix

$$\tilde{G}_{N_2/2} = \begin{pmatrix} \tilde{g}_0 & \cdots & \tilde{g}_{N_2/2} \\ \vdots & & \vdots \\ \tilde{g}_{N_2/2} & \cdots & \tilde{g}_{N_2} \end{pmatrix}$$

equals to $\tilde{d}(N_2)$ (see the section about codes).

A swarm means that in the segment $[1, N_2]$ there is some gap, in which there are no indices j such that $\tilde{\alpha}_{i,j} \ne 0$ for some i.

The Algorithm calculates $rk(\tilde{G}_0), rk(\tilde{G}_1), \cdots, rk(\tilde{G}_{3^k}), \cdots, rk(\tilde{G}_{3^{t^2 \log_3 t}})$. There exists a sequence $t \le l, l+1, \cdots, l + 2t\log_3 t \le 2t^2 \log_3 t$ such that in the segment $(3^l, 3^{l+2t\log_3 t})$ there are no j such that $\tilde{\alpha}_{i,j} \ne 0$ for some i. Since $\tilde{d}(3^l) \le t3^l$, then $rk(\tilde{G}_3 l + \log_3 t) = \cdots = rk(\tilde{G}_3 l + 2t\log_3 t) = \tilde{d}(3^l)$.

Conversely if $rk(\tilde{G}_3 l + \log_3 t) = \cdots = rk(\tilde{G}_3 l + 2t\log_3 t)$ for a certain l, then in the segment $(3^{l+\log_3 t}, 3^{l+2t\log_3 t})$ there are no j such that $\tilde{\alpha}_{i,j} \ne 0$ for some i. Indeed in the opposite case there would exist $j_0 < 3^{\frac{l+2t\log_3 t}{t^2}}$ in this segment such that in the segment $(j_0, t^2 j_0)$ there are no j such that $\tilde{\alpha}_{i,j} \ne 0$ for some i. Then

$$rk(\tilde{G}_{3^{l+2t\log_3 t}}) \ge rk(\tilde{G}_{tj_0}) > rk(\tilde{G}_{3^{l+\log_3 t}}) \quad ,$$

because $2tj_0$ creates a swarm. Thus, we have proved that in the segment $(3^{l+\log_3 t}, 3^{l+2t\log_3 t})$ there are no j such that $\tilde{\alpha}_{i,j} \ne 0$ for some i. Hence $3^{l+4\log_3 t}$ creates a swarm and the algorithm recovers it by means of the extension of Goppa code.

Actually, there could be different swarms and the algorithm will recover a swarm, after which there is a large gap, much larger than it is required by the definition of the swarm.

After finding a swarm of terms, we subtract it from the function f_1/f_2 and so reduce a number of terms (sparsity) and continue until exhausting.

6 Analysis of the algorithm

Let us assume that we are supplied also with a black-box for computing a factorial (as a preconditioning). Then the number of arithmetic operation necessary to fulfill is at most $3^{O(t^2 \log t)}$, and the number of parallel steps is $O(t^5 \log^2 t)$ by Mulmuley [M 86].
So, it is independent from the total degree d of the rational function. If we count bit complexity, then the time would be bounded by $(dM)^{O(1)}$, where d is the degree and M is the bit-size of the coefficients, and the parallel time $\leq \log^{O(1)}(dM)$ (again by [M 86]).

Remark about using [CG 82] for finding roots of denominator (see Section 2).

$$f = \prod_{1 \leq i \leq t} (Y - c_i)^{\beta_i} \quad , \deg f \leq 3^t .$$

The number of c_i is at most t because of t-sparsity of f_1/f_2 .
$(f/GCD(f, f') = \prod(Y - c_i)$ – apply to it [CG 82], find $c_i \longrightarrow$ find β_i in parallel time $O(t) \longrightarrow \alpha_i$)

7 Finding an entier of a sparse rational function is in NC

Let a rational function $q \in \mathbb{Q}(x)$ be given by a black-box and we assume that q can be represented in a form $q = f/g$, where polynomials $f, g \in \mathbb{Z}[X]$ are both t-sparse and form a minimal t-sparse representation of q (in the sense of a degree of denominator g) and the leading coefficient $lc(g) = 1$. We analyse the complexity in terms of arithmetic operations (cf. [BT 88], [GKS 91]). Unlike the previous sections we suppose that we know a bound d on the degrees $\deg(f), \deg(g) < d$. Under this supposition we'll show that the problem of finding the entier $[f/g] = h \in \mathbb{Z}[X]$ is in the parallel class NC (cf. [C 85, KR 90, K 89]) in the sizes of the inputs and outputs.

Denote $d_1 = \deg(f)$, $d_0 = \deg(g)$, M is a maximal of bit-sizes of the coefficients of f, g (they are not supposed to be given). Represent $q = f/g = [f/g] + \frac{\mathrm{Rem}(f,g)}{g}$. We call a rational number $0 < c \in \mathbb{Q}$ big enough if $\left|\frac{\mathrm{Rem}(f,g)}{g}(c)\right| < \frac{1}{2}$. Our next purpose is to construct explicitely a big enough number.

Take succesive primes $p_1, \ldots, p_t$ and for each p among them calculate (by black-box) $q(p), q(p^2), \cdots, q(p^{2t^2+1})$. For at least one p all these values are defined (let us fix it).

Lemma At least one of $q(p), q(p^2), \cdots, q(p^{2t^2+1})$ has an absolute value greater than $2^{M/2t}/t^{4dt^2}$.

PROOF Denote $\mathcal{N} = \max\{|q(p)|, \cdots, |q(p^{2t^2+1})|\}$. Denote $f = \sum_{1 \leq i \leq t} \alpha_i x^{j_i}$, $g = \sum_{1 \leq i \leq t} \beta_i x^{k_i}$. The homogeneous linear system in the indeterminates α_i, β_i

$$\sum \alpha_i p^{sj_i} = (\sum \beta_i p^{sk_i}) q(p^s), \quad 1 \leq s \leq 2t^2 + 1$$

has a unique solution, since the polynomials f, g provide a minimal t-sparse represntation of q, hence these equalities imply that $(\sum \alpha_i x^{j_i}) / (\sum \beta_i x^{k_i}) = q(x)$. Therefore, each α_i, β_i equals to an appropriate $(2t-1) \times (2t-1)$ minor of this system. Then $2^M \leq \max\{|\alpha_i|, |\beta_i|\} \leq (\mathcal{N} p^{2t^2 d} 2t)^{2t} \leq (\mathcal{N} t^{4dt^2})^{2t}$. Lemma is proved.

Then one can produce in NC ([BC 86]) an integer t^{4dt^2} and multiply it on $\mathcal{N}$, so we get a rational number greater than $2^{M/2t}$. Then again involving [BC 86], one can construct a rational number $N_0 > 36 \cdot 2^{3M} \cdot d^5$.

Calculate $q(N_0)$. W.l.o.g. assume that $lc(f) > 0$. Then $f(N_0) > N_0^{d_1} - dN_0^{d_1-1}2^M > \frac{1}{2}N_0^{d_1}$, $g(N_0) < N_0^{d_0} + dN_0^{d_0-1}2^M < \frac{3}{2}N_0^{d_0}$. Thus, $q(N_0) > \frac{1}{3}N_0^{d_1-d_0}$. On the other hand $f(N_0) < 2^M dN_0^{d_1}$, $g(N_0) > N_0^{d_0} - dN_0^{d_0-1}2^M$, therefore $q(N_0) < 3 \cdot 2^M dN_0^{d_1-d_0}$. Thus if $q(N_0) < \frac{1}{3}$ then $d_1 - d_0 < 0$ and $h = [f/g] = 0$, if $d_1 - d_0 \geq 0$ and $q(N_0) < \frac{1}{3}N_0$ then $d_1 - d_0 = 0$. Assume now that $d_1 - d_0 > 0$. Notice that the absolute value of each coefficient of $\text{rem}(f,g)$ is at most $\left(2^{M(d_1-d_0+2)}\right)(d_1 - d_0 + 2)^{d_1-d_0+2}$ (see [L 82]). Calculate then $N_1 = q(q(N_0)) > 3^{d_0-d_1-1}N_0^{(d_1-d_0)^2}$. We claim that N_1 is big enough. Indeed, $g(N_1) > N_1^{d_0} - 2^M d_0 N_1^{d_0-1} > \frac{1}{2}N_1^{d_0}$, $|\text{rem}(f,g)(N_1)| < \left(2^{M(d_1-d_0+2)}\right)(d_1 - d_0 + 2)^{d_1-d_0+2}d_0N_1^{d_0-1} < \frac{1}{4}N_1^{d_0}$. Take an integer $N = [N_1] + 1$, which is also a big enough number.

Having a big enough integer N, we'll find the entier $[f/g] = h$ by a method similar to [BT 88] (see also [GK 87]), which one can call an approximative polynomial interpolation. We compute $q(N), q(N^2), \ldots, q(N^{2t})$ and take the nearest integers to them, respectively, $A_1, \ldots, A_{2t}$. Then $A_i = h(N^i)$, $1 \leq i \leq 2t$, since N is big enough, and one can apply BCH-codes (as in [BT 88]) to recover the powers of X occurring in h, and also the coefficients.

Arithmetic complexity of the whole procedure for finding entier h is $(t \log d)^{O(1)}$ and the parallel time $O(\log t \log\log d)$.

Acknowlegment: We thank Mike Singer for the number of interesting discussions.

References

[BS 83] Baur, W., Strassen, V., *The Complexity of Partial Derivatives*, Theor. Comput. Sci., 1983, 22, pp. 317–330.

[BC 86] Beame, P. W., Cook, S. A., Hoover, H. J., *Log Depth Circuits for Division and Related Problems*, SIAM J. Comput., 1986, 15, pp. 994–1003.

[BT 88] Ben-Or, M., Tiwari, P., *A Deterministic Algorithm For Sparse Multivarite Polynomial Interpolation*, Proc. STOC ACM, 1988, pp. 301–309.

[CG 82] Chistov, A. L., Grigoriev, D. Yu., *Polynomial-Time Factoring Multivariable Polynomials Over a Global Field*, Preprint LOMI, E-5-82, Leningrad, 1982.

[C 85] Cook, S. A., *A Taxonomy of Problems with Fast Parallel Algorithms*, Information and Control **64** (1985), pp. 2-22.

[DG 91] Dress, A., Grabmeier, J., *The Interpolation Problem for k-sparse Polynomials and Character Sums*, in Adv. App. Math. **12** (1991), pp. 57-75.

[GK 87] Grigoriev, D. Yu., Karpinski, M., *The Matching Problem for Bipartite Graphs with Polynomially Bounded Permanents is in NC*, Proc. 28^{th} IEEE FOCS (1987), pp. 166–172.

[GKS 90] Grigoriev, D. Yu., Karpinski, M., Singer, M., *Interpolation of Sparse Rational Functions Without Knowing Bounds on Exponents*, Proc. 31^{st} IEEE FOCS 1990, pp. 840–846.

[GKS 91] Grigoriev, D. Yu., Karpinski, M., Singer, M., *The Interpolation Problem for k-Sparse Sums of Eigenfunctions of Operators*, in Adv. Appl. Math. **12** (1991), pp. 76-81.

[KT 88] Kaltofen E., Trager, B., *Computing with Polynomials Given by Black Boxes for Their*

Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators, Proc. 29^{th} IEEE FOCS 1988, pp. 296–305.

[KR 90] Karp, R. M. and Ramachandran, V. L., *A Survey of Parallel Algorithms for Shared-Memory Machines*, Research Report No. UCB/CSD 88/407, University of California, Berkeley; in Handbook of Theoretical Computer Science, MIT Press (1990), pp. 870-941.

[K 89] Karpinski, M., *Boolean Circuit Complexity of Algebraic Interpolation Problems*, Technical Report TR-89-027, International Computer Science Institute, Berkeley (1989); in Proc. CSL '88, Lecture Notes in Computer Science **385** (1989), Springer-Verlag, pp. 138-147.

[L 82] Loos, R., *Generalized Polynomial Remainder Sequences*, in: "Computer Algebra", Springer, 1982, pp. 115–137.

[MS 81] Mac Williams, F. J., Sloan, N. J. A., *The Theory of Error Correcting Codes*, North-Holland, 1981.

[M 86] Mulmuley, K., *A Fast Parallel Algorithm to Compute the Rank of a Matrix Over an Arbitrary Field*, Proc STOC ACM, 1986.

[P 77a] Plaisted, D., *Sparse Complex Polynomials and Polynomial Reducibility*, J. Comput. System Sci. 14 (1977), pp. 210–221

[P 77b] Plaisted, D., *New NP-Hard and NP-Complete Polynomial and Integer Divisibility Problems*, Proc. 18^{th} IEEE FOCS (1977), pp. 241–253.