REPRESENTATIONS OF RATIONAL FUNCTIONS

Joachim von zur Gathen

Department of Computer Science
University of Toronto

Extended Abstract

1. Introduction

This work forms part of an endeavour to understand the power of parallelism in symbolic computation: for which problems in algebraic manipulation with a polynomial-time sequential solution do fast parallel algorithms exist? Answers to this question may help to understand the power of parallelism in general: one may compare the power of different models of concurrent computation by testing on which models these algorithms can be implemented.

In this paper we present fast parallel algorithms for various problems in algebraic computation. It soon became apparent that the algorithms for all the problems considered here would follow the same pattern, namely conversion between different representations of the given rational function. So we start by introducing in section 2 the notion of representations of rational functions in a given "base" of polynomials. This notion encompasses several ways of representing rational functions, which are especially familiar if the rational function is a polynomial: the sequence of coefficients, a list of values, Taylor series, and a general list of values in "Hermite format". It turns out that if numerator and denominator degrees are correctly specified, then usually (but not always) such representations also determine a rational function uniquely.

We describe in section 3 two fast parallel algorithms that convert the coefficient representation of a rational function into a "base representation" and vice versa. Combining them we get an algorithm that converts the representation of a rational function in one base into that in another base. Section 4 discusses the existence question for representations. This turns out to be slightly less straightforward than one might expect, and we see e.g. that the rational functions required in Padé approximation or rational interpolation may fail to exist.

Section 5 presents as application of the general conversion algorithms fast parallel methods for the following problems in symbolic manipulation: Taylor expansion, partial fraction decomposition, Chinese remainder algorithm, elementary symmetric functions, Padé approximation, and various interpolation problems. As our model of parallel computation we can take a parallel algebraic computation graph, where at each node each processor can either perform an arithmetic operation (+,-,×,/,fetching a constant) or a test ($a=0$?) or a boolean operation (see von zur Gathen [83]). The algorithms can also be implemented on an algebraic PRAM. The algorithms all run in parallel time $O(\log^2 n)$ and use $n^{O(1)}$ processors, where n is the input size. They work over an arbitrary ground field.

In section 6, we generalize this approach to include representations in a "Laurent format". As expected, rational functions then always have a unique representation.

The dual relationship between evaluation at many points and interpolation has been observed for a long time; also the fact that both computational problems consist in converting the representation of a polynomial from one format to another; see e.g. Strassen [74]. However, one usually employs two quite different-looking sequential algorithms to solve the two problems. Besides the unification resulting from our approach even for polynomials, the fact of including rational functions into this framework seems to be even more interesting from a computational point of view, making bedfellows of such distinct-looking problems as Hermite interpolation and Padé approximation. We leave as an open question how well this approach translates into the sequential setting.

0272-5428/83/0000/0133$01.00 © 1983 IEEE

2. Representations of rational functions

Let F be an arbitrary field. A sequence $B=(b_1,\dots,b_p)$ of pairwise relatively prime polynomials $b_1,\dots,b_p \in F[x]$ is called a base. A sequence $N=(n_1,\dots,n_p) \in \mathbf{N}^p$ with $n_i \geq 1$ is called a precision (for B), and $n = \sum_{1\leq i\leq p} n_i deg b_i$ is the total precision of (B,N). A sequence

$$r=(r_{10},\dots,r_{1,n_1-1};\ \cdots\ ;r_{p0},\dots,r_{p,n_p-1})$$

such that $r_{ij} \in F[x]$ and $deg r_{ij} < deg b_i$ for all i,j with $1\leq i\leq p$ and $0\leq j<n_i$ is called a representative in base B with precision N, or (B,N)-representative for short. $R(B,N)$ is the set of all (B,N)-representatives. $R(B,N)$ can be identified with F^n. If $f=g/h \in F(x)$ and for $1\leq i\leq p$ we have

$$g \equiv h \sum_{0\leq j<n_i} r_{ij} b_i^j \bmod b_i^{n_i},$$

$$gcd(b_i,h) = 1,$$

then r is called a (B,N)-representation of f. In other words, if we set

$$r_i = \sum_{0\leq j<n_i} r_{ij}\, b_i^j,$$

then $f \equiv r_i \bmod b_i^{n_i}$ is given with "precision $n_i deg b_i$", and r_i is developed according to powers of b_i. Let us look at a few familiar representations that fit into this framework. As examples we will use $n=5$ and the two rational functions

$$f_1 = x^4-x^3+2x^2-3x-2 \in F[x],$$

$$f_2 = \frac{-7x^2+x+2}{x^2+x-1} \in F(x).$$

When B and N are given, we will write $r=\rho(f)$ if r is a (B,N)-representation of f.

1. *Sequence of coefficients.* In this most familiar of all representations we have $p=1$, $B=(x)$ and $N=(n)$. For a polynomial $f = \sum_{0\leq j} f_j x^j \in F[x]$, the (B,N)-representation $\rho(f)=(f_0,\dots,f_{n-1})$ is simply the sequence of the first n coefficients (irrespective of the degree of f). For a rational function f, $\rho(f)$ is an initial segment of the power series expansion of f around 0. In the example, $N=(5)$ and $\rho(f_1)=\rho(f_2)=(-2,-3,2,-1,1)$.

2. *List of values.* Given are $a_1,\dots,a_n \in F$ pairwise distinct, and $p=n$, $B=(x-a_1,\dots,x-a_n)$ and $N=(1,\dots,1)$. Then $r_i=f(a_i)$ is the value of f at a_i. For the example, let $(a_1,\dots,a_5)=(-2,-1,0,1,2)$ (and assume that $char F\neq 5$). Then $B=(x+2,x+1,x,x-1,x-2)$, $N=(1,1,1,1,1)$, $\rho(f_1)=(36,5,-2,-3,8)$, and $\rho(f_2)=(-28,6,-2,-4,\frac{-24}{5})$.

3. *Taylor expansion.* Here some $a \in F$ is given, and $p=1$, $B=(x-a)$, $N=(n)$, and $r=(r_0,\dots,r_{n-1})$ where

$$f \equiv \sum_{0\leq j<n} r_j (x-a)^j \bmod (x-a)^n,$$

so that $r_0,\dots,r_{n-1}$ are the first n coefficients of the Taylor expansion of f at a. (It is assumed that f can be written with a denominator that

$$r_j = \frac{1}{j!}\left(\frac{d^j f}{dx^j}\right)(a).$$

The sequence of coefficients is nothing but the Taylor expansion at 0. In the example, if we choose $a=2$ and $n=5$, then $\rho(f_1)=(8,25,20,7,1)$ and $\rho(f_2)=(\frac{-24}{5},\frac{-3}{5},\frac{4}{25},\frac{-1}{25},\frac{1}{125})$.

4. *General list of values.* As a common generalization of the last two cases, we get the general list of values of the rational function and some of its derivatives in "Hermite format". We have $a_1,\dots,a_p \in F$ pairwise distinct, $B=(x-a_1,\dots,x-a_p)$ and $N=(n_1,\dots,n_p)$ with $n_i \geq 1$ and $n_1+\cdots+n_p=n$. Then

$$\rho(f)=(r_{10},\dots,r_{1,n_1-1};\ \cdots\ ;r_{p0},\dots,r_{p,n_p-1}),$$

where

$$r_i = \sum_{0\leq j<n_i} r_{ij}(x-a_i)^j \equiv f \bmod (x-a_i)^{n_i}$$

is the initial segment of length n_i of the Taylor expansion of f at a_i. In the example, we choose $p=2$, $a_1=-1$, $a_2=2$, $B=(x+1,x-2)$ and $N=(2,3)$. Then $\rho(f_1)=(5,-14;8,25,20)$ and $\rho(f_2)=(6,-14;\frac{-24}{5},\frac{-3}{5},\frac{4}{24})$.

3. Conversion algorithms

We now describe two algorithms: the first one converts a rational function that is given as the quotient of two polynomials (and these polynomials are represented by their standard coefficient sequences) into its representation in base B with precision N, and the second one performs the inverse conversion. Thus the standard representation by coefficients plays a special role: The basic parts r_{ij} are given by their coefficients, and all conversions from one representation to another pass via this special representation.

We will make use of the Extended Euclidean Scheme of two polynomials $a_0,a_1 \in F[x]$:

$$\begin{array}{lll} a_0 = q_1 a_1 + a_2 & & s_2 a_0 + t_2 a_1 = a_2 \\ \vdots & & \vdots \\ a_{l-2} = q_{l-1} a_{l-1} + a_l & & s_{l-1} a_0 + t_{l-1} a_1 = a_{l-1} \\ a_{l-1} = q_l a_l & & s_l a_0 + t_l a_1 = a_l \end{array}$$

where the following conditions are satisfied for $2 \le k \le l$: $a_k, q_k, s_k, t_k \in F[x]$, $dega_k < dega_{k-1}$, $s_0=1$, $t_0=0$, $s_1=0$, $t_1=1$, $s_k = s_{k-2} - q_{k-1}s_{k-1}$ and $t_k=t_{k-2} - q_{k-1}t_{k-1}$. Thus the q's are the quotients and the a's the remainders of Euclid's algorithm, the s's and t's are the "continuants" or "convergents", and $gcd(f,g)$ is the unique monic scalar multiple of a_l. (By convention, all gcd's of polynomials in $F[x]$ are monic.)

Algorithm STATREP. (Standard-coefficients-to-representation)

Input: A base $B=(b_1,\dots,b_p)$ with a precision $N=(n_1,\dots,n_p)$ and a pair (g,h) of polynomials in $F[x]$ such that $gcd(b_1 \cdots b_p,h)=1$. All input polynomials are given by their coefficient vectors.

Output: The (B,N)-representation r of $f = g/h$.

1. For all i, $1 \le i \le p$, do steps 2, 3, 4.

2. Compute $s_i,t_i \in F[x]$ such that

$$s_i b_i^{n_i} + t_i h = 1,$$

$$degt_i < n_i\, degb_i.$$

3. Compute $r_i \in F[x]$ such that

$$r_i \equiv gt_i \bmod b_i^{n_i},$$

$$degr_i < n_i\, degb_i.$$

4. For $0 \le j < n_i$ compute $u_{ij}, v_{ij}, r_{ij} \in F[x]$ such that

$$r_i = u_{ij}\, b_i^j + v_{ij},$$

$$degv_{ij} < j\, degb_i,$$

$$r_{ij} = u_{ij} - u_{i,j+1}b_i.$$

(Division with remainder of r_i by b_i^j. Use $u_{i0} = r_i$ and $u_{i,n_i} = 0$.)

5. Return

$$r=(r_{10},\dots,r_{1,n_1-1};\ \cdots\ ;r_{p0},\dots,r_{p,n_p-1}).$$

Algorithm REPTSTA. (Representation-to-standard-coefficients)

Input: A base $B=(b_1,\dots,b_p)$ with a precision $N=(n_1,\dots,n_p)$ and total precision n, a representation $r \in R(B,N)$ and $d \in \mathbf{N}$ with $d<n$. This number d serves as a bound on the degree of the denominator polynomial. Again all input polynomials are given by their coefficients.

Output: The coefficients of two polynomials $g,h \in F[x]$ such that r is a (B,N)-representation of $\frac{g}{h}$, $degh \le d$, $degg + degh < n$, h is monic and $gcd(b_1 \cdots b_p,h)$ divides $gcd(g,h)$.

1. For $1 \le i \le p$ compute $r_i = \sum_{0 \le j < n_i} r_{ij} b_i^j$.

2. Compute $u \in F[x]$ such that

$$\forall i, 1 \le i \le r,\ u \equiv r_i \bmod b_i^{n_i},$$

$$degu < \sum n_i degb_i.$$

3. Compute the length l and the entries a_k, s_k, t_k, where $1 \le k \le l$, of the Extended Euclidean Scheme of $(a_0,a_1)=(b_1^{n_1} \cdots b_p^{n_p},u)$.

4. Determine k such that $dega_k \le d < dega_{k-1}$ and return $g=a_k, h=t_k$.

Theorem 3.1. The algorithms *STATREP* and *REPTSTA* work correctly as described. They can be performed in parallel time $O(log^2 n)$ on inputs that have total precision at most n and (for *STATREP*) $degg + degh \le n$. □

4. Representability

We now want to discuss some of the general properties of these representations.

Fix a base B and a precision N. Optimistically, one might hope that every rational function has a unique (B,N)-representation, and that algorithms *STATREP* and *REPTSTA* compute functions that are inverse to each other.

Both properties require a little care, however. If $f=g/h$ with $g,h \in F[x]$, $gcd(g,h)=1$ and $gcd(b_1,h) \ne 1$, then f has of course no representation in base B. Namely, if r were a representation, then

$$gcd(b_1,h) \mid g - hr_1,$$

and hence

$$gcd(b_1,h) \mid g,$$

contradicting the assumption. Therefore we can only expect a representation if f is in

$$S(B) = \{f \in F(x): \exists\, g,h \in F[x] \text{ such that } f=g/h \text{ and } gcd(b_1 \cdots b_p,h)=1\}.$$

(This semilocal ring $S(B)$ is the intersection in $F(x)$ of all localizations $F[x]_{(q)}$, with q running through the irreducible factors of $b_1 \cdots b_p$.) In Theorem 4.1 we show that indeed every $f \in S(B)$ has a unique (B,N)-representation.

For the second property of algorithms *REPTSTA* and *STATREP* computing inverse functions, consider the example $p=1$, $B=(x^3-x)$, $N=(1)$, $d=1$ and $r=(x^2+1)$. The output of algorithm *REPTSTA* is $(-2x,-x)$. Thus $f=-2x/(-x)=2$, and if we apply algorithm *STATREP* to f, the output is $(2) \ne r$. This example makes it clear that the second property will not hold for all $r \in R(B,N)$. But we will see that it holds for "almost all" r.

Theorem 4.1. Let B be a base, N a precision for B and $f \in F(x)$. Then f has a unique (B,N)-representation iff $f \in S(B)$. □

We now write $STATREP(B,N,f) = r$ and $REPTSTA(B,N,r,d) = f$ for the functions computed by the two algorithms *STATREP* and *REPTSTA*, and want to prove that they are inverse bijections on the following set $T(B,n,d)$ which is the "degree-bounded version" of $S(B)$:

$$T(B,n,d) = \{f \in F(x): \exists g,h \in F[x]\ f = g/h,$$
$$gcd(b_1 \cdots b_p, h) = 1, deg g < n-d, deg h \le d\}.$$

Theorem 4.2. The functions computed by *STATREP* and *REPTSTA* give inverse bijections between the set $T(B,n,d) \subset F(x)$ and its image in $R(B,N)$. □

5. Applications

We can now reap the fruits of the work spent in setting up the previous notation. By putting together algorithms *REPTSTA* and *STATREP* (using different bases) we obtain a fast parallel algorithm for conversion from one base to another. This yields fast parallel algorithms for a number of important computational problems (and also clarifies how these problems are related to each other).

INTERPOLATION(n) has as input a pair (a,r) where $a=(a_1,\ldots,a_n)$ and $r=(r_1,\ldots,r_n)$ have entries from F, and $a_i \neq a_j$ for $1 \le i < j \le n$. The output are the coefficients of the unique $f \in F[x]$ such that $deg f < n$ and $f(a_i) = r_i$ for $1 \le i \le n$. This function is nothing but the conversion from $((x-a_1,\ldots,x-a_n),(1,\ldots,1))$- representation to $((x),(n))$-representation. In order to obtain the unique monic $f \in F[x]$ of degree n such that $f(a_i) = r_i$ for all i, one uses the above interpolation for the values $r_i - a_i^n$.

TAYLOR EXPANSION(n) has as input $a \in F$ and the coefficients $(g_0,\ldots,g_{n-d-1}; h_0,\ldots,h_{d-1})$ of

$$f = \sum_{0 \le j < n-d} g_j x^j \Big/ \sum_{0 \le j < d} h_j x^j \in F(x)$$

where $\sum h_j a^j \neq 0$. The output are the Taylor coefficients $f_0,\ldots,f_{n-1} \in F$ of f at a, so that $f \equiv \sum_{0 \le j < n} f_j (x-a)^j \bmod (x-a)^n$. This is the conversion from coefficients to $((x-a),(n))$-representation.

For a polynomial f, TAYLOR EXPANSION can be computed by calculating binomial coefficients and evaluating universal Taylor coefficients of f (see e.g. von zur Gathen [83], section 6). This can be performed in parallel time $O(\log n)$, so that the statement of Theorem 5.1 below is not interesting in this case.

HERMITE INTERPOLATION(n) has an input of the form $(a,r_1,\ldots,r_p)$ where $a=(a_1,\ldots,a_p)$, $r_i=(r_{i0},\ldots,r_{i,n_i-1})$, all $a_i, r_{ij} \in F$, and $a_i \neq a_j$ for $1 \le i < j \le p$, and $n_1 + \cdots + n_p = n$. The output are the coefficients of the unique polynomial $f \in F[x]$ such that $deg f < n$ and

$$f \equiv \sum_{0 \le j < n_i} r_{ij} (x-a_i)^j \bmod (x-a_i)^{n_i}$$

for $1 \le i \le p$.

Thus the first n_i coefficients of the Taylor expansion of f at a_i are prescribed. If $\mathrm{char} F = 0$, this is equivalent to prescribing the values of the first n_i derivatives of f at a_i as

$$\left(\frac{d^j f}{dx^j}\right)(a_i) = j!\, r_{ij}.$$

HERMITE INTERPOLATION(n) is the conversion from $((x-a_1,\ldots,x-a_p),(n_1,\ldots,n_p))$-representation to $((x),(n))$-representation. Both INTERPOLATION and TAYLOR EXPANSION for polynomials are special cases of this problem. Again, one can also compute the unique monic interpolating polynomial of degree n.

CHINESE REMAINDER ALGORITHM(n) has as input a sequence $(b_1,\ldots,b_p,r_1,\ldots,r_p)$ of polynomials from $F[x]$ such that $deg(b_1 \cdots b_p) = n$, $deg r_i < deg b_i$ and $gcd(b_i,b_j) = 1$ for $1 \le i < j \le p$. and $gcd(b_i,b_j) = 1$ for $1 \le i < j \le p$. The output are the coefficients of the unique $f \in F[x]$ such that $f \equiv r_i \bmod b_i$ for $1 \le i \le p$ and $deg f < n$. This is the conversion from $((b_1,\ldots,b_p),\ (1,\ldots,1))$-representation to $((x),(n))$-representation.

ELEMENTARY SYMMETRIC FUNCTIONS(n) has as input a sequence $(c_1,\ldots,c_n)$ with $c_i \in F$. Output are the elementary symmetric functions $s_j = \sigma_j(c_1,\ldots,c_n)$ for $1 \le j \le n$. Thus

$$x^n - s_1 x^{n-1} + \cdots + (-1)^n s_n = (x-c_1) \cdots (x-c_n).$$

This can be viewed as a special case of the monic version of HERMITE INTERPOLATION: Set $C=\{c_1,\ldots,c_n\}$, $p = \#C$, $\{a_1,\ldots,a_p\} = C$, and $r_{ij} = 0$ for $0 \le j < n_i$, where a_i occurs exactly n_i times among $c_1,\ldots,c_n$. The inverse function - root-finding - cannot be computed by a rational algorithm.

PARTIAL FRACTION DECOMPOSITION(n) has as input the coefficients of polynomials $b_1,\ldots,b_p$ and q, and $n_1,\ldots,n_p \in \mathbf{N}$ such that $b_1,\ldots,b_p$ are pairwise relatively prime and $deg q < deg(b_1^{n_1} \cdots b_p^{n_p}) = n$. Output are the coefficients of the unique polynomials r_{ij} $(1 \le i \le p, 1 \le j < n_i)$ such that

$$\frac{q}{b_1^{n_1} \cdots b_p^{n_p}} = \sum_{\substack{1 \le i \le p \\ 1 \le j < n_i}} \frac{r_{ij}}{b_i^j},$$

$$deg r_{ij} < deg b_i.$$

PARTIAL FRACTION DECOMPOSITION(n) is the conversion from $((x),(n))$-representation to $((b_1,\ldots,b_p),(n_1,\ldots,n_p))$-representation.

PADE APPROXIMATION(n) has as input a polynomial $f\in F[x]$ of degree $<n$, and $d\in\mathbf{N}$ with $0\le d<n$. The output consists of the coefficients of polynomials $g,h\in F[x]$ such that $g\equiv fh \bmod x^n$, $degg<n-d$, and $degh\le d$. Thus $g/h\equiv f \bmod x^n$ is a Padé approximant to f (provided $h(0)\neq 0$). This function is computed by algorithm *REPTSTA* with input $B=(x)$, $N=(n)$ and $r=(f_0,\ldots,f_{n-1})$ where $f=\sum f_i x^i$.

CAUCHY INTERPOLATION(n) has as input $d\in\mathbf{N}$ with $0\le d<n$ and a pair (a,r) where $a=(a_1,\ldots,a_n)$ and $r=(r_1,\ldots,r_n)$ have entries from F and $a_i\neq a_j$ for $1\le i<j\le n$. The output consists of the coefficients of the unique polynomials $g,h\in F[x]$ such that $g(a_i)=h(a_i)r_i$ for $1\le i\le n$, $degg<n-d$, $degh\le d$, and h is monic. Thus $f=g/h$ is a rational function with prescribed denominator degree that interpolates the given values r_i at a_i, i.e. $f(a_i)=r_i$ (provided $h(a_i)\neq 0$). Cauchy [1821] had first considered this problem and given an explicit solution by a closed formula similar to the Lagrange interpolation formula. Algorithm *REPTSTA* with base $B=(x-a_1,\ldots,x-a_n)$ and precision $N=(1,\ldots,1)$ computes a solution, if one exists.

RATIONAL HERMITE INTERPOLATION(n) has an input of the form $(d,a,r_1,\ldots,r_p)$, where $0\le d<n$, $a=(a_1,\ldots,a_p)$, $r_i=(r_{i0},\ldots,r_{i,n_i-1})$, all $a_i,r_{ij}\in F$, $a_i\neq a_j$ for $1\le i<j\le p$, and $n_1+\cdots+n_p=n$. The output are the coefficients of the unique polynomials $g,h\in F[x]$ such that

$$g\equiv h\cdot\sum_{0\le j<n_i} r_{ij}(x-a_i)^j \bmod (x-a_i)^{n_i},$$

$degg<n-d$, $degh\le d$, and h is monic. Thus the initial segments of the Taylor expansion of the rational function $f=g/h$ at a_i are prescribed, i.e.

$$f\equiv\sum_{0\le j<n_i} r_{ij}(x-a_i)^j \bmod (x-a_i)^{n_i}$$

(provided $h(a_i)\neq 0$). This problem simultaneously generalizes HERMITE INTERPOLATION (which has $d=0$), PADE APPROXIMATION (which has $p=1$ and $a_1=0$), and CAUCHY INTERPOLATION (which has $n_1=\cdots=n_p=1$). It can be computed by algorithm *REPTSTA* with input $B=(x-a_1,\ldots,x-a_p)$, $N=(n_1,\ldots,n_p)$ and $r=(r_1,\ldots,r_p)$.

Theorem 5.1. The following nine functions can be computed in parallel time $O(\log^2 n)$ using $n^{O(1)}$ processors: INTERPOLATION, TAYLOR EXPANSION, HERMITE INTERPOLATION, CHINESE REMAINDER ALGORITHM, ELEMENTARY SYMMETRIC FUNCTIONS, PARTIAL FRACTION DECOMPOSITION, PADE APPROXIMATION, CAUCHY INTERPOLATION, RATIONAL HERMITE INTERPOLATION.

Remark 5.2. Theorem 4.1 has described the rational functions with unique representation. For a "bad case" for Padé approximation, let $n\ge 5$ and $r=x^{n-1}-x^{n-2}+1$. Then $g=h=x^2$ is the only solution of the conditions

$$g,h\in F[x],\ g\equiv hr \bmod x^n,$$

$$degg<n-2,\ degh\le 2,\ h \text{ monic}.$$

In particular, there does not exist a Padé approximant $f=g/h\in F(x)$ satsifying the above conditions and $gcd(g,h)=1$. This phenomenon of nonexistence of solutions to the Padé approximation problem and for rational interpolation was discovered by Kronecker [1881], who illustrated it with an example. In Padé's work [1892], this fundamental limitation does not appear.

Remark 5.3. A natural extension of the present notion of representation would be to allow "rational" representations, either by allowing fractions $r_i=\frac{s_i}{t_i}$, or by having r_i in a Laurent format $r_i=\sum_{d_i\le j<n_i+d_i} r_{ij}f_i^j$ with $d_i\in\mathbf{Z}$. With the latter approach, it turns out that indeed, given a base and precision, every rational function has a unique Laurent representation. Another direction would be to consider other Euclidean domains, for example the ring of integers, instead of $F[x]$.

In order to obtain good parallel algorithms, it was sufficient to consider the general case of base conversion. It might be interesting to know how well this general approach carries over to the well-understood sequential algorithms.

References

A. Cauchy, Cours d'analyse de l'école royale polytechnique (Analyse algébrique), 1821, 429-433 (in: Oeuvres complètes, IIe série, tome III).

J. von zur Gathen, Parallel algorithms for algebraic problems. Proc. 15th ACM Symp. Theory of Computing, Boston, 1983, 17-23. To appear in SIAM J. Comput.

E. Kronecker, Zur Theorie der Elimination einer Variabeln aus zwei algebraischen Gleichungen. Monatsberichte der Akademie der Wissenschaften, Berlin 1881, 535-600.

H. Padé, Sur la représentation approchée d'une fonction par des fractions rationnelles. Annales Scientifiques de l'Ecole Normale Supérieure, 3e série, 9(1892), Supplément S3-S93.

V. Strassen, Some Results in Algebraic Complexity Theory. Proc. Int. Congress of Mathematicians, Vancouver 1974, 497-501.