Fast Evaluation and Interpolation
&
Gcd and Resultant

Alin Bostan

MPRI C-2-22
October 7, 2021

M2 Internship Projects in relation to C-2-22

- *Structured linear algebra for polynomial system solving*,
  J. Berthomieu (LIP6, Sorbonne U.) + V. Neiger (LIP6, Sorbonne U.)
- *Algorithms for factoring differential equations*,
  A. Bostan (Inria, U. Paris-Saclay) + T. Rivoal (CNRS, U. Grenoble)
- *Automatic proofs of special functions inequalities*,
  A. Bostan + M. Safey (LIP6, Sorbonne U.) + B. Salvy (Inria, ENS Lyon)
- *Univariate matrices for faster polynomial system solving*,
  V. Neiger (LIP6, Sorbonne U.) + J. Berthomieu (LIP6, Sorbonne U.)

The exercises from last week

(1) Let $\mathsf{T}(n)$ be the complexity of multiplication of $n \times n$ lower triangular matrices. Show that one can multiply any two $n \times n$ matrices in $O(\mathsf{T}(n))$ ops.

(2) Let $\mathbb{K}$ be a field, let $P \in \mathbb{K}[x]$ be of degree less than n and θ be a feasible exponent for matrix multiplication in $\mathcal{M}_n(\mathbb{K})$.

(a) Find an algorithm for the simultaneous evaluation of P at $\lceil \sqrt{n} \rceil$ elements of $\mathbb{K}$ using $O(n^{\theta/2})$ operations in $\mathbb{K}$.

(b) If Q is another polynomial in $\mathbb{K}[X]$ of degree less than n, show how to compute the first n coefficients of $P \circ Q := P(Q(x))$ in $O(n^{\frac{\theta+1}{2}})$ ops. in $\mathbb{K}$.

▷ Hint: Write $P(x)$ as $\sum_i P_i(x)(x^d)^i$, where d is well-chosen and the P_i's have degrees less than d.

Ex. 1

Let $\mathsf{T}(n)$ be the complexity of multiplication of $n \times n$ lower triangular matrices. Show that one can multiply any two $n \times n$ matrices in $O(\mathsf{T}(n))$ ops.

Solution:

▷ For any $n \times n$ matrices A and B,

$$\begin{bmatrix} 0 & 0 & 0 \\ B & 0 & 0 \\ 0 & A & 0 \end{bmatrix}^2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ AB & 0 & 0 \end{bmatrix}.$$

▷ Let α be a feasible exponent for multiplication of lower triangular matrices. Then, $n^\theta \leq \mathsf{T}(3n) = O(n^\alpha)$ and thus $\theta \leq \alpha$.

Ex. 2

Let $\mathbb{K}$ be a field, let $P \in \mathbb{K}[x]$ be of degree less than n and θ be a feasible exponent for matrix multiplication in $\mathcal{M}_n(\mathbb{K})$.

(a) Find an algorithm for the simultaneous evaluation of P at $\lceil \sqrt{n} \rceil$ elements of $\mathbb{K}$ using $O(n^{\theta/2})$ operations in $\mathbb{K}$.

(b) If Q is another polynomial in $\mathbb{K}[X]$ of degree less than n, show how to compute the first n coefficients of $P \circ Q := P(Q(x))$ in $O(n^{\frac{\theta+1}{2}})$ ops. in $\mathbb{K}$.

Solution 2(a):

▷ Write $P(x)$ as $\sum_i P_i(x)(x^d)^i$, where $d = \lceil \sqrt{n} \rceil$ and the P_i's have degrees $< d$

▷ Evaluations of the P_i's at the points $x_1, \ldots, x_d$ read off the matrix product

$$\begin{bmatrix} P_0(x_1) & \ldots & P_0(x_d) \\ \vdots & & \vdots \\ P_{d-1}(x_1) & \ldots & P_{d-1}(x_d) \end{bmatrix} = \begin{bmatrix} p_{0,0} & \ldots & p_{0,d-1} \\ \vdots & & \vdots \\ p_{d-1,0} & \ldots & p_{d-1,d-1} \end{bmatrix} \times \begin{bmatrix} 1 & \ldots & 1 \\ \vdots & & \vdots \\ x_1^{d-1} & \ldots & x_d^{d-1} \end{bmatrix}$$

Ex. 2

Solution 2(b): Baby step / giant step strategy

▷ Write $P(x)$ as $\sum_i P_i(x)(x^d)^i$, where $d = \lceil \sqrt{n} \rceil$ and the P_i's have degrees $< d$

▷ Compute $Q^2, \dots, Q^d =: R$ and $R^2, \dots, R^{d-1}$ mod x^n $\quad O(d\,\mathsf{M}(n)) = O(n^{\frac{\theta+1}{2}})$

For $p_{i,j} := [x^j]P_i$ and $q_{i,j} := [x^j]Q^i$ $(j < n, i < d)$, compute $P_i(Q)$ mod x^n using the $(d \times d) \times (d \times n)$ matrix product $\quad [x^j]P_i(Q) = \sum_k p_{i,k}q_{k,j}$

$$\begin{bmatrix} p_{0,0} & \cdots & p_{0,d-1} \\ \vdots & & \vdots \\ p_{d-1,0} & \cdots & p_{d-1,d-1} \end{bmatrix} \times \begin{bmatrix} q_{0,0} & \cdots & q_{0,n-1} \\ \vdots & & \vdots \\ q_{d-1,0} & \cdots & q_{d-1,n-1} \end{bmatrix},$$

▷ Can be done using $\lceil n/d \rceil = O(d)$ products of $d \times d$ matrices $\quad O(d^{\theta+1})$

▷ Final recombination $P(Q)$ mod $x^n = \sum_{i=0}^{d-1} P_i(Q)R^i$ mod x^n $\quad O(d\,\mathsf{M}(n))$

Context

▷ Main concepts: Evaluation-interpolation paradigm and Modular algorithms

▷ Alternative representations of algebraic objects: e.g., polynomials given

- by list of coefficients: useful for fast division
- by list of values taken on given points: useful for fast multiplication (FFT)

▷ Modular algorithms based on fast conversions between representations, e.g. evaluation-interpolation, Chinese Remaindering

▷ Avoid intermediate expression swell, e.g. det of polynomial matrices

▷ Important issue: choice of the moduli (evaluation points), e.g. fast factorial

Main problems and results

Multipoint evaluation Given P in $\mathbb{A}[X]$, of degree $< n$, compute the values

$$P(a_0), \ldots, P(a_{n-1}).$$

Interpolation Given $v_0, \ldots, v_{n-1} \in \mathbb{A}$, with $a_i - a_j$ invertible in $\mathbb{A}$ if $i \neq j$, find the polynomial $P \in \mathbb{A}[X]$ of degree $< n$ such that

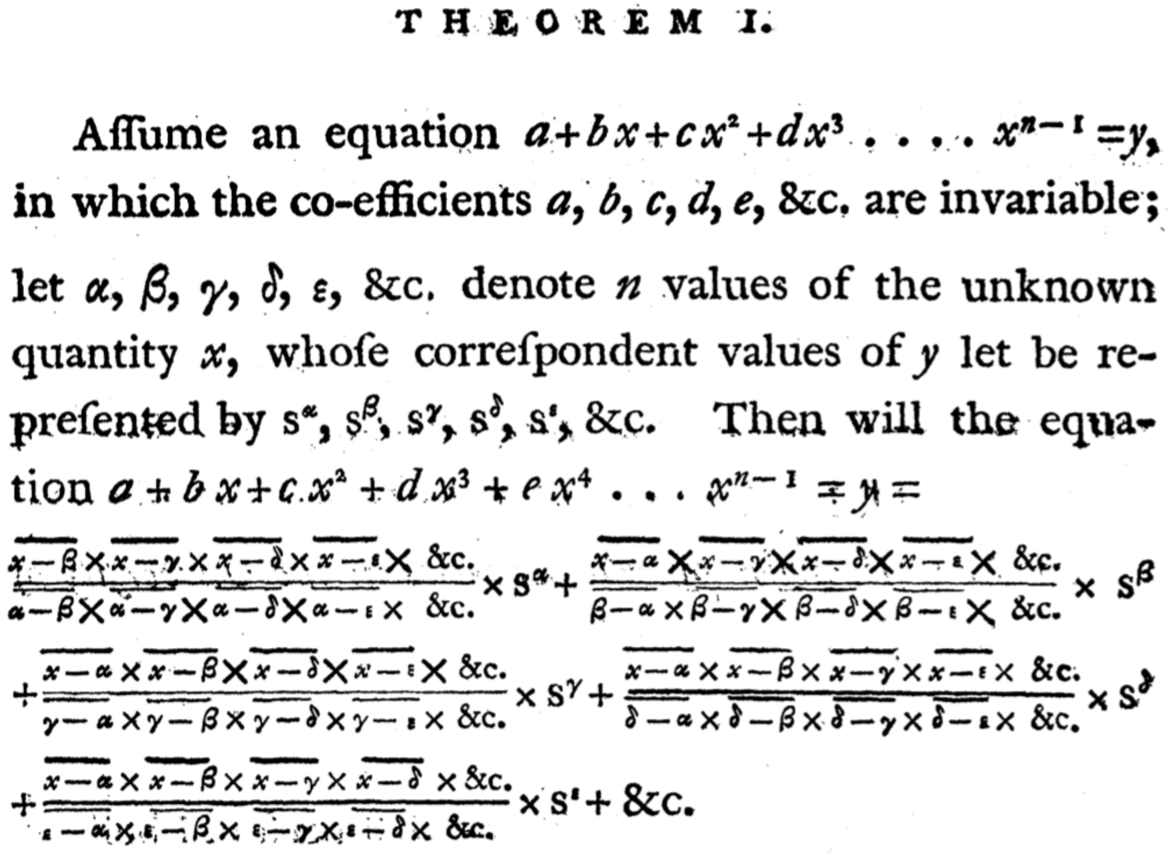
$$P(a_0) = v_0, \ldots, P(a_{n-1}) = v_{n-1}.$$

Theorem One can solve both problems in:

- $O(\mathsf{M}(n) \log n)$ ops. in $\mathbb{A}$
- $O(\mathsf{M}(n))$ ops. in $\mathbb{A}$ if the a_i's are in geometric progression

▷ Extension to fast polynomial/integer Chinese remaindering

Waring-Lagrange interpolation

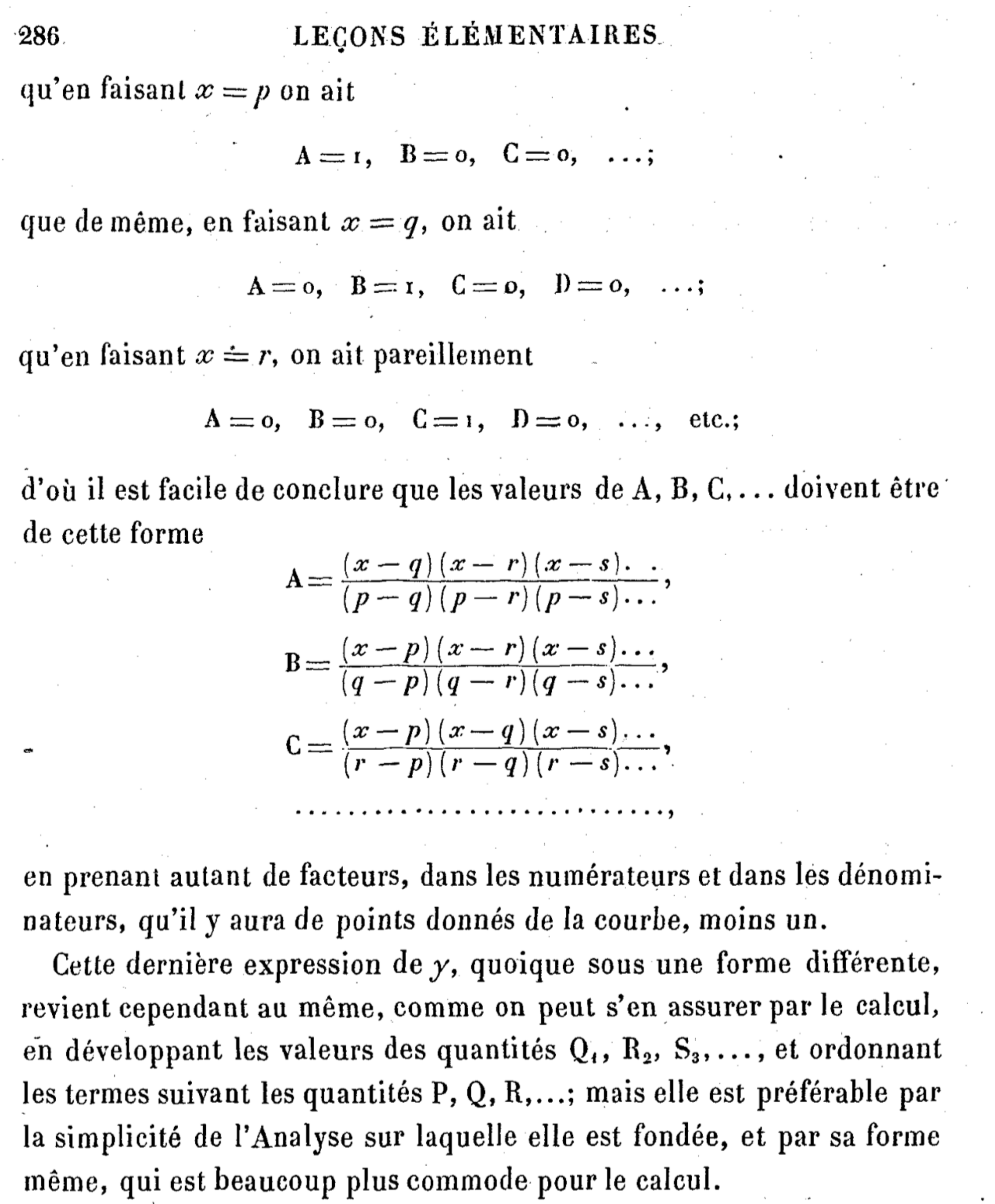


THEOREM I.

Aſſume an equation $a+bx+cx^2+dx^3 \ldots\ldots x^{n-1}=y$, in which the co-efficients a, b, c, d, e, &c. are invariable; let α, β, γ, δ, ε, &c. denote n values of the unknown quantity x, whoſe correſpondent values of y let be repreſented by s^α, s^β, s^γ, s^δ, s^ε, &c. Then will the equation $a+bx+cx^2+dx^3+ex^4 \ldots x^{n-1}=y=$

$$\frac{\overline{x-\beta}\times\overline{x-\gamma}\times\overline{x-\delta}\times\overline{x-\varepsilon}\times\ \&c.}{\overline{\alpha-\beta}\times\overline{\alpha-\gamma}\times\overline{\alpha-\delta}\times\overline{\alpha-\varepsilon}\times\ \&c.}\times s^\alpha+\frac{\overline{x-\alpha}\times\overline{x-\gamma}\times\overline{x-\delta}\times\overline{x-\varepsilon}\times\ \&c.}{\overline{\beta-\alpha}\times\overline{\beta-\gamma}\times\overline{\beta-\delta}\times\overline{\beta-\varepsilon}\times\ \&c.}\times s^\beta$$
$$+\frac{\overline{x-\alpha}\times\overline{x-\beta}\times\overline{x-\delta}\times\overline{x-\varepsilon}\times\ \&c.}{\overline{\gamma-\alpha}\times\overline{\gamma-\beta}\times\overline{\gamma-\delta}\times\overline{\gamma-\varepsilon}\times\ \&c.}\times s^\gamma+\frac{\overline{x-\alpha}\times\overline{x-\beta}\times\overline{x-\gamma}\times\overline{x-\varepsilon}\times\ \&c.}{\overline{\delta-\alpha}\times\overline{\delta-\beta}\times\overline{\delta-\gamma}\times\overline{\delta-\varepsilon}\times\ \&c.}\times s^\delta$$
$$+\frac{\overline{x-\alpha}\times\overline{x-\beta}\times\overline{x-\gamma}\times\overline{x-\delta}\times\&c.}{\overline{\varepsilon-\alpha}\times\overline{\varepsilon-\beta}\times\overline{\varepsilon-\gamma}\times\overline{\varepsilon-\delta}\times\ \&c.}\times s^\varepsilon+\ \&c.$$

[Waring, 1779 – “Problems concerning Interpolations”]



qu'en faisant $x = p$ on ait

$$A = 1, \quad B = 0, \quad C = 0, \quad \ldots;$$

que de même, en faisant $x = q$, on ait

$$A = 0, \quad B = 1, \quad C = 0, \quad D = 0, \quad \ldots;$$

qu'en faisant $x = r$, on ait pareillement

$$A = 0, \quad B = 0, \quad C = 1, \quad D = 0, \quad \ldots, \quad \text{etc.};$$

d'où il est facile de conclure que les valeurs de A, B, C, ... doivent être de cette forme

$$A = \frac{(x-q)(x-r)(x-s)\ldots}{(p-q)(p-r)(p-s)\ldots},$$

$$B = \frac{(x-p)(x-r)(x-s)\ldots}{(q-p)(q-r)(q-s)\ldots},$$

$$C = \frac{(x-p)(x-q)(x-s)\ldots}{(r-p)(r-q)(r-s)\ldots},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$

en prenant autant de facteurs, dans les numérateurs et dans les dénominateurs, qu'il y aura de points donnés de la courbe, moins un.

Cette dernière expression de y, quoique sous une forme différente, revient cependant au même, comme on peut s'en assurer par le calcul, en développant les valeurs des quantités Q_1, R_2, S_3, ..., et ordonnant les termes suivant les quantités P, Q, R,...; mais elle est préférable par la simplicité de l'Analyse sur laquelle elle est fondée, et par sa forme même, qui est beaucoup plus commode pour le calcul.

[Lagrange, 1795 – "Sur l'usage des courbes dans la solution des problèmes"]

Fast polynomial division

Euclidean division for polynomials

[Strassen, 1973]

Pb: Given $F, G \in \mathbb{K}[x]_{\leq N}$, compute (Q, R) in Euclidean division $F = QG + R$

Naive algorithm: $O(N^2)$

Idea: look at $F = QG + R$ from infinity: $Q \sim_{+\infty} F/G$

Let $N = \deg(F)$ and $n = \deg(G)$. Then $\deg(Q) = N - n$, $\deg(R) < n$ and

$$\underbrace{F(1/x)x^N}_{\text{rev}(F)} = \underbrace{G(1/x)x^n}_{\text{rev}(G)} \cdot \underbrace{Q(1/x)x^{N-n}}_{\text{rev}(Q)} + \underbrace{R(1/x)x^{\deg(R)}}_{\text{rev}(R)} \cdot x^{N-\deg(R)}$$

Algorithm:

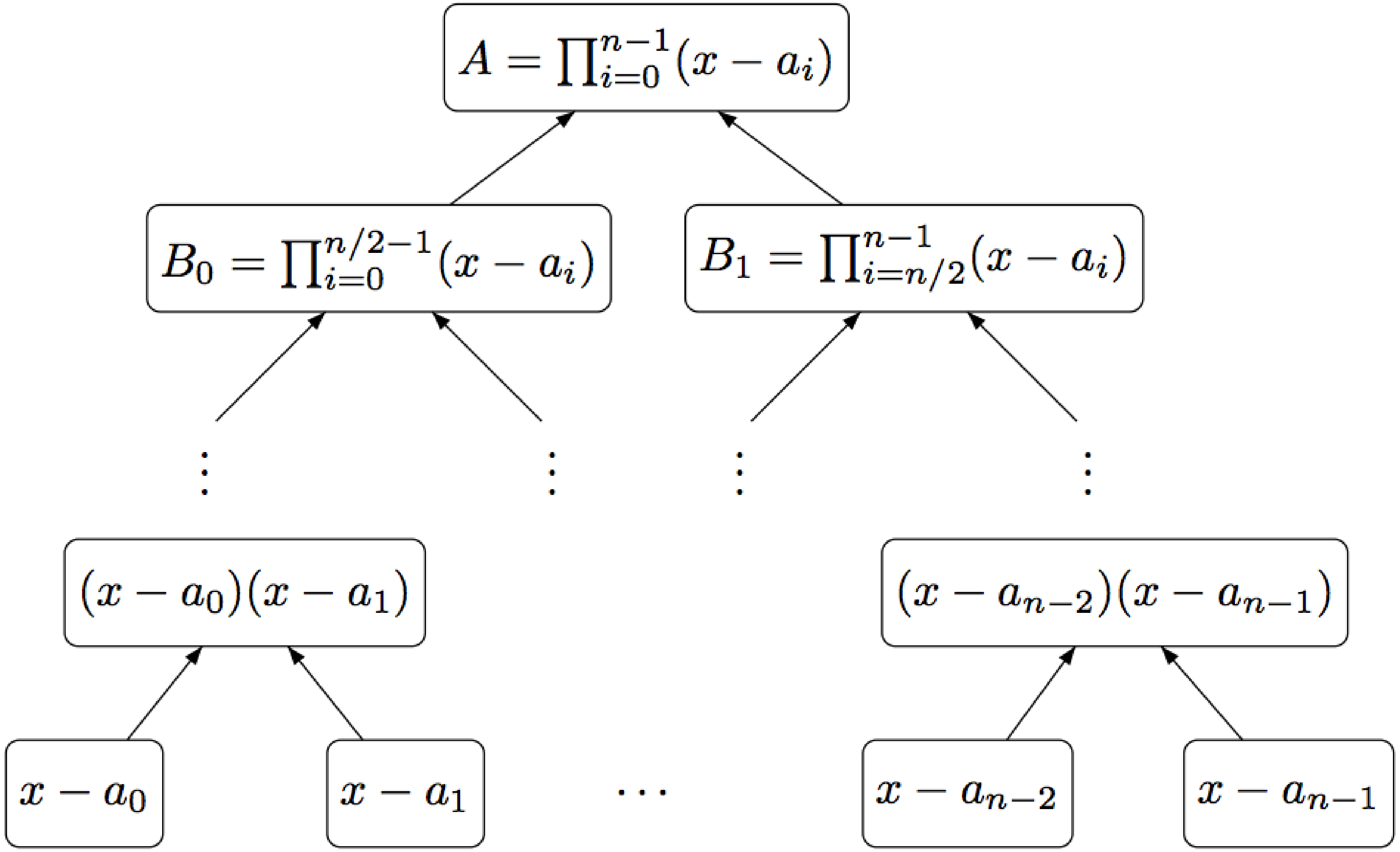
- Compute $\text{rev}(Q) = \text{rev}(F)/\text{rev}(G) \mod x^{N-n+1}$ $O(\mathsf{M}(N))$
- Recover Q $O(1)$
- Deduce $R = F - QG$ $O(\mathsf{M}(N))$

Evaluation-interpolation, general case

Subproduct tree

[Horowitz, 1972]

Problem: Given $a_0, \ldots, a_{n-1} \in \mathbb{K}$, compute $A = \prod_{i=0}^{n-1}(x - a_i)$


$A = \prod_{i=0}^{n-1}(x - a_i)$
$B_0 = \prod_{i=0}^{n/2-1}(x - a_i)$
$B_1 = \prod_{i=n/2}^{n-1}(x - a_i)$
$(x - a_0)(x - a_1)$
$(x - a_{n-2})(x - a_{n-1})$
$x - a_0$
$x - a_1$
$\cdots$
$x - a_{n-2}$
$x - a_{n-1}$

DAC Theorem: $\mathsf{S}(n) = 2 \cdot \mathsf{S}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{S}(n) = O(\mathsf{M}(n) \log n)$

Fast multipoint evaluation

[Borodin-Moenck, 1974]

Pb: Given $a_0, \dots, a_{n-1} \in \mathbb{K}$ and $P \in \mathbb{K}[x]_{<n}$, compute $P(a_0), \dots, P(a_{n-1})$

Naive algorithm: Compute $P(a_i)$ independently $O(n^2)$

Basic idea: Use recursively Bézout's identity $P(a) = P(x) \bmod (x-a)$

Divide and conquer: Same idea as for DFT = evaluation by repeated division

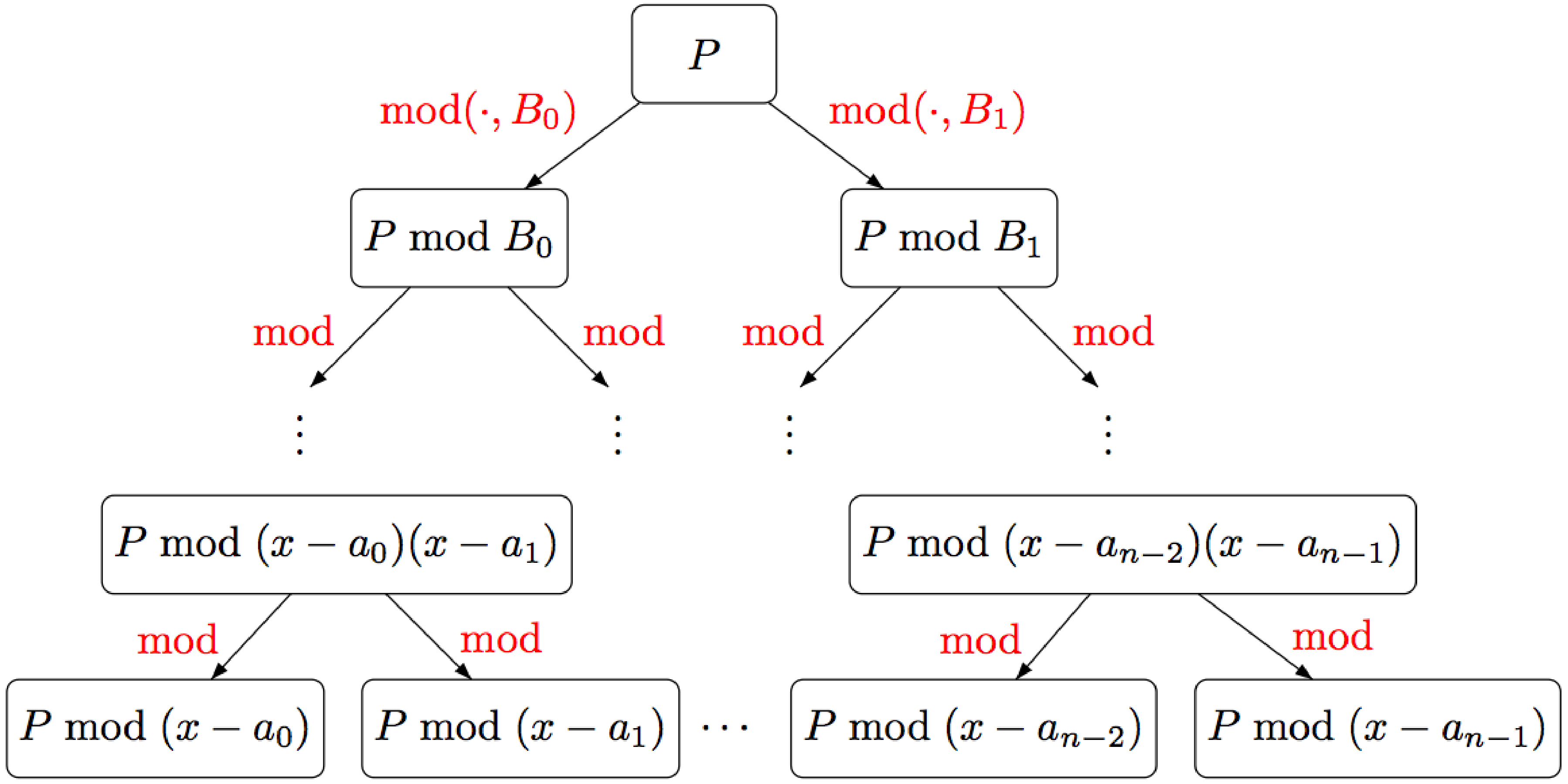
- $P_0 := P \mod \underbrace{(x-a_0)\cdots(x-a_{n/2-1})}_{B_0}$
- $P_1 := P \mod \underbrace{(x-a_{n/2})\cdots(x-a_{n-1})}_{B_1}$

$$\Longrightarrow \begin{cases} P(a_0) = P_0(a_0), \quad \dots, \quad P(a_{n/2-1}) = P_0(a_{n/2-1}) \\ P(a_{n/2}) = P_1(a_{n/2}), \quad \dots, \quad P(a_{n-1}) = P_1(a_{n-1}) \end{cases}$$

Fast multipoint evaluation

[Borodin-Moenck, 1974]

Pb: Given $a_0, \ldots, a_{n-1} \in \mathbb{K}$ and $P \in \mathbb{K}[x]_{<n}$, compute $P(a_0), \ldots, P(a_{n-1})$



DAC Theorem: $\mathsf{E}(n) = 2 \cdot \mathsf{E}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{E}(n) = O(\mathsf{M}(n) \log n)$

Fast interpolation

[Borodin-Moenck, 1974]

Problem: Given $a_0, \dots, a_{n-1} \in \mathbb{K}$ mutually distinct, and $v_0, \dots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(a_0) = v_0, \dots, P(a_{n-1}) = v_{n-1}$

Naive algorithm: Linear algebra, Vandermonde system $O(\mathsf{MM}(n))$

Lagrange's algorithm: Use $P(x) = \sum_{i=0}^{n-1} v_i \frac{\prod_{j\neq i}(x-a_j)}{\prod_{j\neq i}(a_i-a_j)}$ $O(n^2)$

Fast algorithm: Based on the "modified Lagrange formula"

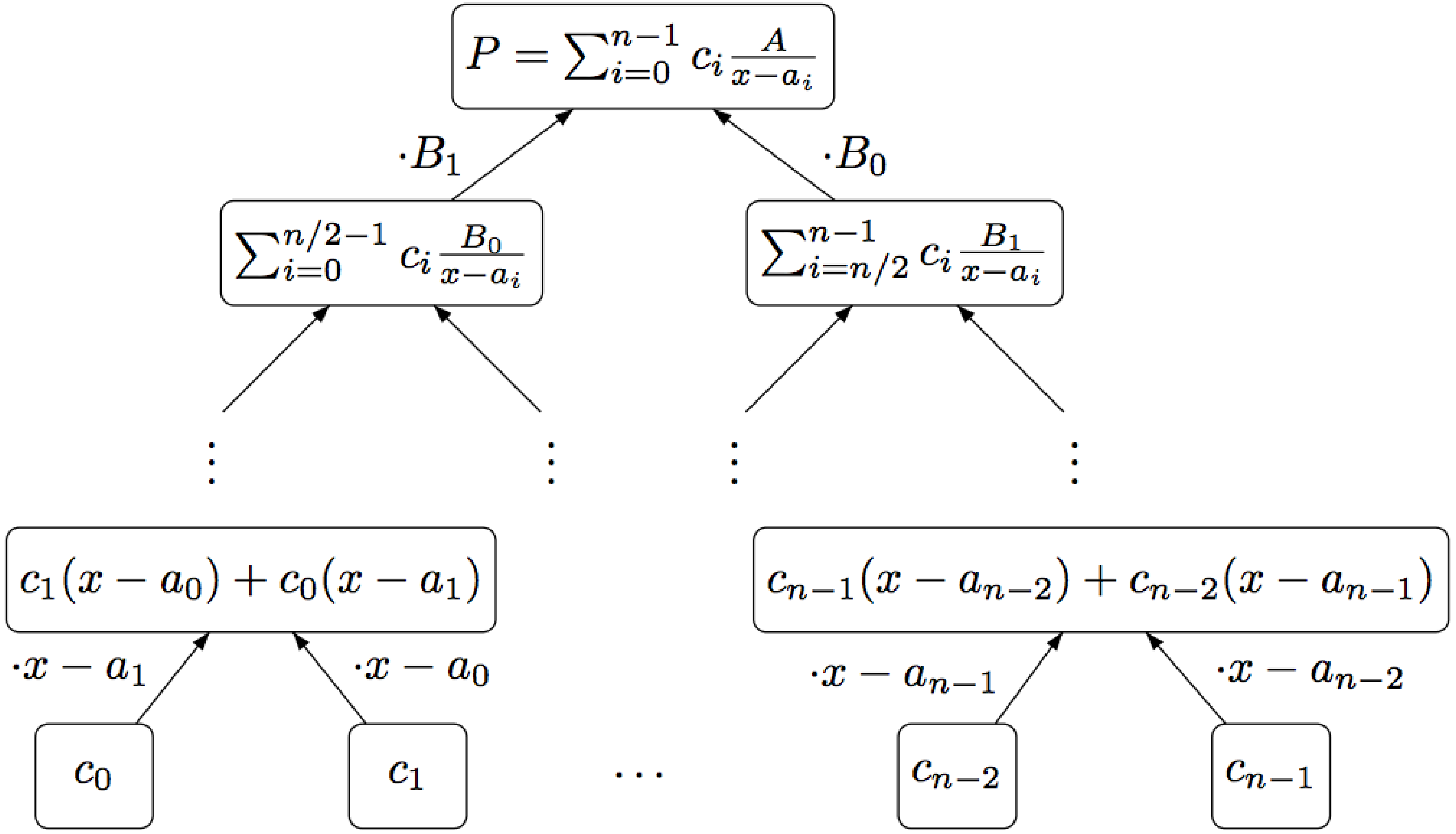
$$P(x) = A(x) \cdot \sum_{i=0}^{n-1} \frac{v_i/A'(a_i)}{x-a_i}$$

- Compute $c_i = v_i/A'(a_i)$ by fast multipoint evaluation $O(\mathsf{M}(n)\log n)$
- Compute $\sum_{i=0}^{n-1} \frac{c_i}{x-a_i}$ by divide and conquer $O(\mathsf{M}(n)\log n)$

Fast interpolation

[Borodin-Moenck, 1974]

Problem: Given $a_0, \dots, a_{n-1} \in \mathbb{K}$ mutually distinct, and $v_0, \dots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(a_0) = v_0, \dots, P(a_{n-1}) = v_{n-1}$



$P = \sum_{i=0}^{n-1} c_i \frac{A}{x-a_i}$
$\cdot B_1$
$\cdot B_0$
$\sum_{i=0}^{n/2-1} c_i \frac{B_0}{x-a_i}$
$\sum_{i=n/2}^{n-1} c_i \frac{B_1}{x-a_i}$
$c_1(x-a_0) + c_0(x-a_1)$
$c_{n-1}(x-a_{n-2}) + c_{n-2}(x-a_{n-1})$
$\cdot x - a_1$
$\cdot x - a_0$
$\cdot x - a_{n-1}$
$\cdot x - a_{n-2}$
c_0
c_1
$\dots$
c_{n-2}
c_{n-1}

DAC Theorem: $\mathsf{I}(n) = 2 \cdot \mathsf{I}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{I}(n) = O(\mathsf{M}(n) \log n)$

Evaluation-interpolation, geometric case

Subproduct tree, geometric case

[B.-Schost, 2005]

Problem: Given $q \in \mathbb{K}$, compute $A = \prod_{i=0}^{n-1}(x - q^i)$

Idea: Compute $B_1 = \prod_{i=n/2}^{n-1}(x - q^i)$ from $B_0 = \prod_{i=0}^{n/2-1}(x - q^i)$, by a homothety

$$B_1(x) = B_0\left(\frac{x}{q^{n/2}}\right) \cdot q^{(n/2)^2}$$

Decrease and conquer:

- Compute $B_0(x)$ by a recursive call
- Deduce $B_1(x)$ from $B_0(x)$ $O(n)$
- Return $A(x) = B_0(x)B_1(x)$ $\mathsf{M}(n/2)$

Master Theorem: $\mathsf{G}(n) = \mathsf{G}(n/2) + O(\mathsf{M}(n)) \implies \mathsf{G}(n) = O(\mathsf{M}(n))$

Fast multipoint evaluation, geometric case

[Bluestein, 1970]

Problem: Given $q \in \mathbb{K}$ and $P \in \mathbb{K}[x]_{<n}$, compute $P(1), P(q), \ldots, P(q^{n-1})$

The needed values are: $$P(q^i) = \sum_{j=0}^{n-1} c_j q^{ij}, \qquad 0 \le i < n$$

Bluestein's trick: $$ij = \frac{(i+j)^2 - i^2 - j^2}{2} \quad \Longrightarrow \quad q^{ij} = q^{(i+j)^2/2} \cdot q^{-i^2/2} \cdot q^{-j^2/2}$$

$$\Longrightarrow \qquad P(q^i) = q^{-i^2/2} \cdot \underbrace{\sum_{j=0}^{n-1} c_j q^{-j^2/2} \cdot q^{(i+j)^2/2}}_{\text{convolution:}}$$

$$[x^{n-1+i}] \left(\sum_{k=0}^{n-1} c_k q^{-k^2/2} x^{n-k-1} \right) \left(\sum_{\ell=0}^{2n-2} q^{\ell^2/2} x^{\ell} \right)$$

Conclusion: Fast evaluation on a geometric sequence in $O(\mathsf{M}(n))$

Fast interpolation, geometric case

[B.-Schost, 2005]

Problem: Given $q \in \mathbb{K}$, and $v_0, \dots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(1) = v_0, \dots, P(q^{n-1}) = v_{n-1}$

Fast algorithm: Modified Lagrange formula

$$P = A(x) \cdot \sum_{i=0}^{n-1} \frac{v_i / A'(q^i)}{x - q^i}, \qquad A = \prod_i (x - q^i)$$

- Compute $A = \prod_{i=0}^{n-1} (x - q^i)$ by decrease and conquer $O(\mathsf{M}(n))$
- Compute $c_i = v_i / A'(q^i)$ by Bluestein's algorithm $O(\mathsf{M}(n))$
- Compute $\sum_{i=0}^{n-1} \frac{c_i}{x - q^i}$ by decrease and conquer $O(\mathsf{M}(n))$

Fast interpolation, geometric case

[B.-Schost, 2005]

Problem: Given $q \in \mathbb{K}$, and $v_0, \dots, v_{n-1} \in \mathbb{K}$, compute $P \in \mathbb{K}[x]_{<n}$ such that $P(1) = v_0, \dots, P(q^{n-1}) = v_{n-1}$

Subproblem: Given $c_0, \dots, c_{n-1} \in \mathbb{K}$, compute $\quad R(x) = \sum_{i=0}^{n-1} \frac{c_i}{x - q^i}$

Idea: change of representation – enough to compute $R \mod x^n$

Second idea: $R \bmod x^n$ = multipoint evaluation at $\{1, q^{-1}, \dots, q^{-(n-1)}\}$:

$$\sum_{i=0}^{n-1} \frac{c_i}{x - q^i} \mod x^n = - \sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1} c_i q^{-i(j+1)} x^j \right) = - \sum_{j=0}^{n-1} C(q^{-j-1}) x^j$$

Conclusion: Algorithm for interpolation at a geometric sequence in $O(\mathsf{M}(n))$ (generalization of the FFT algorithm computing the IDFT)

Product of polynomial matrices

[B.-Schost, 2005]

Problem: Given $A, B \in \mathcal{M}_n(\mathbb{K}[x]_{<d})$, compute $C = AB$

Idea: change of representation – evaluation-interpolation at a geometric sequence $\mathcal{G} = \{1, q, q^2, \dots, q^{2d-2}\}$

- Evaluate A and B at $\mathcal{G}$ $\quad O(n^2\,\mathsf{M}(d))$
- Multiply values $C(v) = A(v)B(v)$ for $v \in \mathcal{G}$ $\quad O(d\,\mathsf{MM}(n))$
- Interpolate C from values $\quad O(n^2\,\mathsf{M}(d))$

Total complexity $\quad O(n^2\,\mathsf{M}(d) + d\,\mathsf{MM}(n))$

First exercise for next Thursday

Let f and g be two polynomials in $\mathbb{K}[x,y]$ of degrees at most d_x in x and at most d_y in y.

(a) Show that it is possible to compute the product $h = fg$ using

$$O(\mathsf{M}(d_x d_y))$$

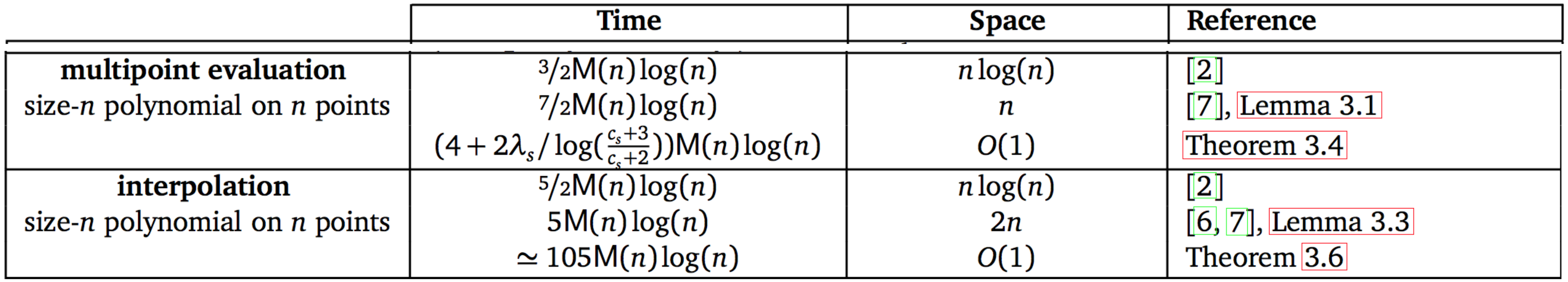
arithmetic operations in $\mathbb{K}$.
Hint: Use the substitution $x \leftarrow y^{2d_y+1}$ to reduce the problem to the product of univariate polynomials.

(b) Improve this result by proposing an evaluation-interpolation scheme which allows the computation of h in

$$O(d_x\,\mathsf{M}(d_y) + d_y\,\mathsf{M}(d_x))$$

arithmetic operations in $\mathbb{K}$.

1. Space-saving versions



	Time	**Space**	**Reference**
multipoint evaluation size-n polynomial on n points	$3/2\mathsf{M}(n)\log(n)$	$n\log(n)$	[2]
	$7/2\mathsf{M}(n)\log(n)$	n	[7], Lemma 3.1
	$(4+2\lambda_s/\log(\frac{c_s+3}{c_s+2}))\mathsf{M}(n)\log(n)$	$O(1)$	Theorem 3.4
interpolation size-n polynomial on n points	$5/2\mathsf{M}(n)\log(n)$	$n\log(n)$	[2]
	$5\mathsf{M}(n)\log(n)$	$2n$	[6, 7], Lemma 3.3
	$\simeq 105\mathsf{M}(n)\log(n)$	$O(1)$	Theorem 3.6

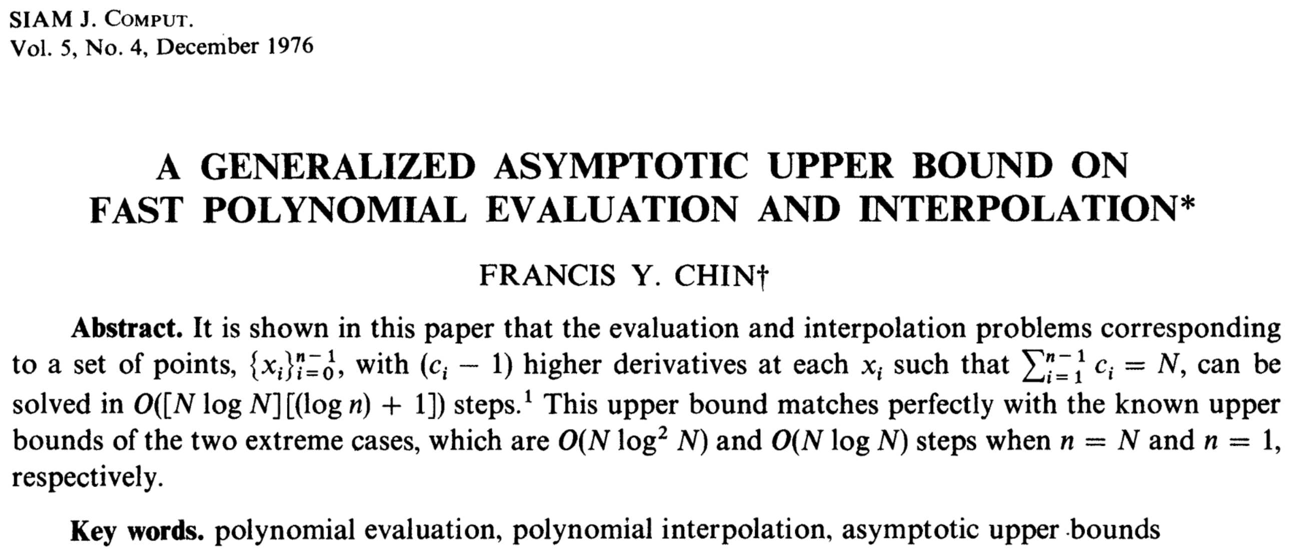
[Giorgi, Grenet & Roche, ISSAC, 2020]

[2] A. Bostan, G. Lecerf, and É. Schost. 2003. Tellegen's Principle into Practice. In ISSAC'03, 37–44.

[6] J. von zur Gathen and V. Shoup. 1992. Computing Frobenius maps and factoring polynomials. Comput. Complex. 2, 3 (1992), 187–224.

[7] P. Giorgi, B. Grenet, and D. S. Roche. 2019. Generic reductions for in-place polynomial multiplication. In ISSAC'19, 187–194.

2. More general evaluation and interpolation



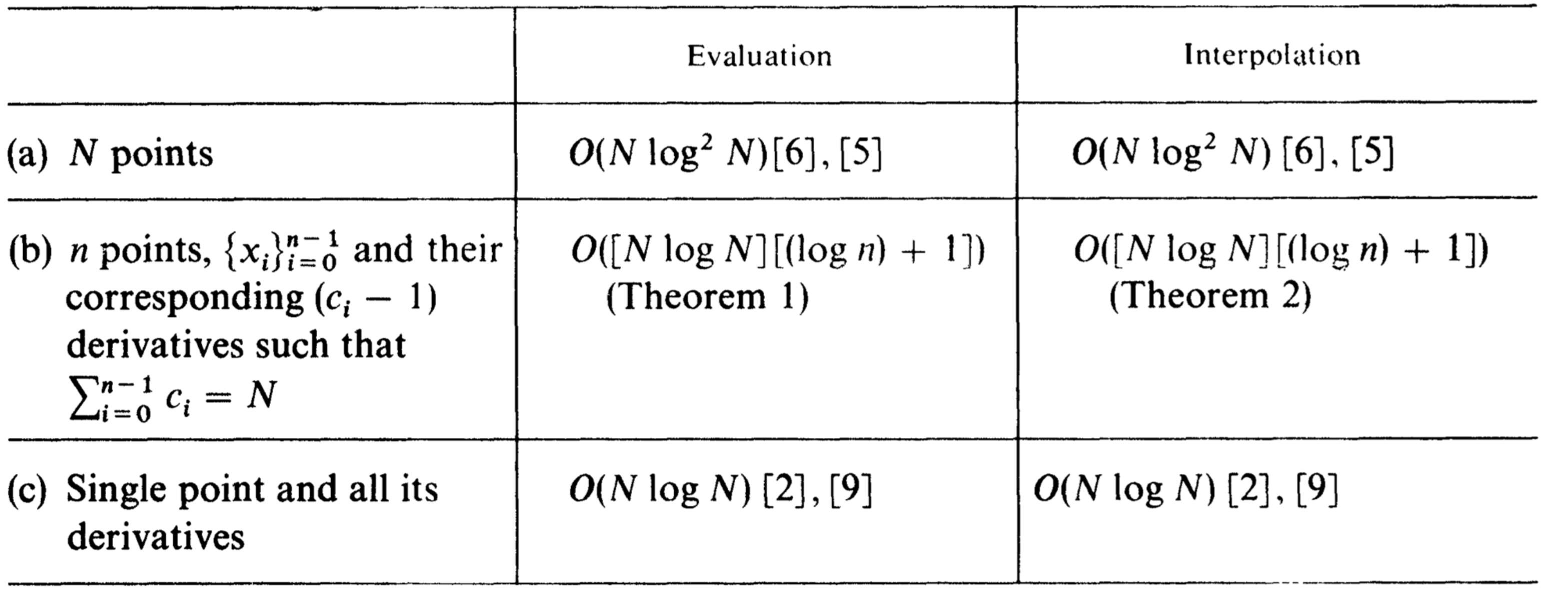
SIAM J. COMPUT.
Vol. 5, No. 4, December 1976

A GENERALIZED ASYMPTOTIC UPPER BOUND ON FAST POLYNOMIAL EVALUATION AND INTERPOLATION*

FRANCIS Y. CHIN†

Abstract. It is shown in this paper that the evaluation and interpolation problems corresponding to a set of points, $\{x_i\}_{i=0}^{n-1}$, with $(c_i - 1)$ higher derivatives at each x_i such that $\sum_{i=1}^{n-1} c_i = N$, can be solved in $O([N \log N][(\log n) + 1])$ steps.[1] This upper bound matches perfectly with the known upper bounds of the two extreme cases, which are $O(N \log^2 N)$ and $O(N \log N)$ steps when $n = N$ and $n = 1$, respectively.

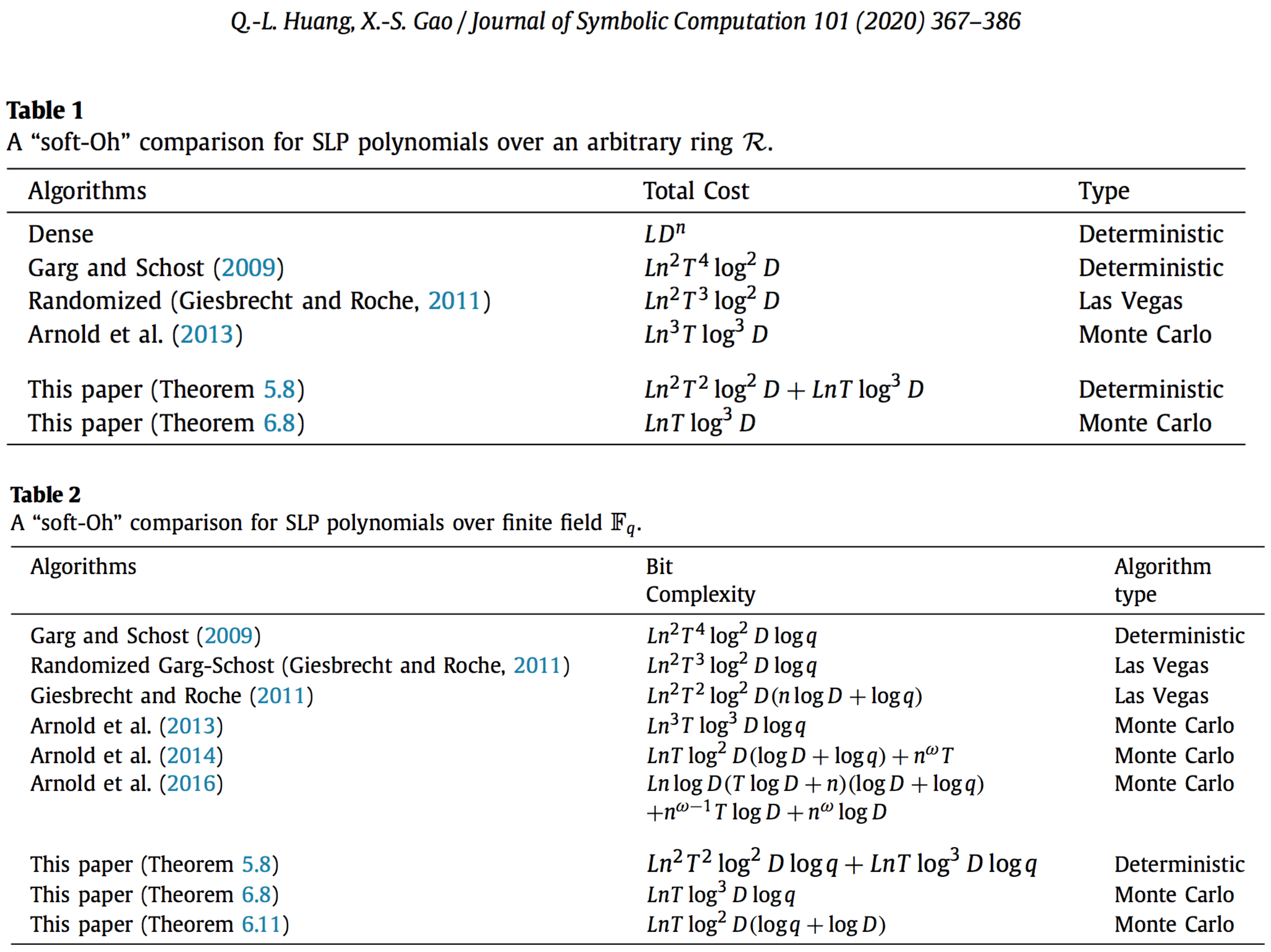
Key words. polynomial evaluation, polynomial interpolation, asymptotic upper bounds



	Evaluation	Interpolation
(a) N points	$O(N \log^2 N)$ [6], [5]	$O(N \log^2 N)$ [6], [5]
(b) n points, $\{x_i\}_{i=0}^{n-1}$ and their corresponding $(c_i - 1)$ derivatives such that $\sum_{i=0}^{n-1} c_i = N$	$O([N \log N][(\log n) + 1])$ (Theorem 1)	$O([N \log N][(\log n) + 1])$ (Theorem 2)
(c) Single point and all its derivatives	$O(N \log N)$ [2], [9]	$O(N \log N)$ [2], [9]

[Chin, SIAM J. Comput., 1976]

3. Multivariate sparse interpolation



Q.-L. Huang, X.-S. Gao / Journal of Symbolic Computation 101 (2020) 367–386

Table 1
A “soft-Oh” comparison for SLP polynomials over an arbitrary ring $\mathcal{R}$.

Algorithms	Total Cost	Type
Dense	LD^n	Deterministic
Garg and Schost (2009)	$Ln^2T^4\log^2 D$	Deterministic
Randomized (Giesbrecht and Roche, 2011)	$Ln^2T^3\log^2 D$	Las Vegas
Arnold et al. (2013)	$Ln^3T\log^3 D$	Monte Carlo
This paper (Theorem 5.8)	$Ln^2T^2\log^2 D + LnT\log^3 D$	Deterministic
This paper (Theorem 6.8)	$LnT\log^3 D$	Monte Carlo

Table 2
A “soft-Oh” comparison for SLP polynomials over finite field $\mathbb{F}_q$.

Algorithms	Bit Complexity	Algorithm type
Garg and Schost (2009)	$Ln^2T^4\log^2 D\log q$	Deterministic
Randomized Garg-Schost (Giesbrecht and Roche, 2011)	$Ln^2T^3\log^2 D\log q$	Las Vegas
Giesbrecht and Roche (2011)	$Ln^2T^2\log^2 D(n\log D+\log q)$	Las Vegas
Arnold et al. (2013)	$Ln^3T\log^3 D\log q$	Monte Carlo
Arnold et al. (2014)	$LnT\log^2 D(\log D+\log q)+n^\omega T$	Monte Carlo
Arnold et al. (2016)	$Ln\log D(T\log D+n)(\log D+\log q)$ $+n^{\omega-1}T\log D+n^\omega\log D$	Monte Carlo
This paper (Theorem 5.8)	$Ln^2T^2\log^2 D\log q+LnT\log^3 D\log q$	Deterministic
This paper (Theorem 6.8)	$LnT\log^3 D\log q$	Monte Carlo
This paper (Theorem 6.11)	$LnT\log^2 D(\log q+\log D)$	Monte Carlo

[Huang & Gao, JSC, 2020]

GCD and Extended GCD

GCD

If $A, B \in \mathbb{K}[x]$, then $G \in \mathbb{K}[x]$ is a gcd of A and B if

- G divides both A and B,
- any common divisor of A and B divides G.

▷ It is a generator of the ideal of $\mathbb{K}[x]$ generated by A and B, i.e.,

$$\left\{U \cdot A + V \cdot B \,\middle|\, U, V \in \mathbb{K}[x]\right\} = \left\{W \cdot G \,\middle|\, W \in \mathbb{K}[x]\right\}$$

▷ It is unique up to a constant: the gcd, after normalization (G monic)

▷ It is useful for:

- normalization (simplification) of rational functions
- squarefree factorization of univariate polynomials

▷ Computation: Euclidean algorithm

Euclidean algorithm

Euclid(A, B)

```
Input  $A$ and $B$ in $\mathbb{K}[x]$.

Output  A gcd $G$ of $A$ and $B$.

  1. $R_0 := A$; $R_1 := B$; $i := 1$.

  2. While $R_i$ is non-zero, do:

         $R_{i+1} := R_{i-1} \bmod R_i$

         $i := i + 1$.

  3. Return $R_{i-1}$.
```

▷ Correctness: $\gcd(F, G) = \gcd(G, F \bmod G)$

▷ Termination: $\deg(B) > \deg(R_2) > \deg(R_1) > \cdots$

▷ Quadratic complexity: $O\big(\deg(A)\deg(B)\big)$ operations in $\mathbb{K}$

Extended GCD

If $A, B \in \mathbb{K}[x]$, then $G = \gcd(A, B)$ satisfies (Bézout relation)

$$G = U \cdot A + V \cdot B, \quad \text{with } U, V \in \mathbb{K}[x]$$

▷ The co-factors U and V are unique if one further asks

$$\deg(U) < \deg(B) - \deg(G) \quad \text{and} \quad \deg(V) < \deg(A) - \deg(G)$$

Then one calls (G, U, V) the extended gcd of A and B.

▷ Example: for $A = a + bx$ with $a \neq 0$ and $B = 1 + x^2$,

$$G = 1 \quad \text{and} \quad \frac{a - bx}{a^2 + b^2} \cdot A + \frac{b^2}{a^2 + b^2} \cdot B = 1$$

Extended GCD

Usefulness of Bézout coefficients:

- modular inversion and division in a quotient ring $Q = \mathbb{K}[x]/(B)$: A is invertible in Q if and only if $\gcd(A, B) = 1$. In this case: the inverse of A in Q is equal to U, where $U \cdot A + V \cdot B = 1$.
- Next lecture (14/10): proof of "Any algebraic function is D-finite"

▷ Example: For $A = a + bx, B = 1 + x^2$, the inverse of A mod B is

$$U = \frac{a - bx}{a^2 + b^2}.$$

▷ Computation: Extended Euclidean algorithm

Extended Euclidean algorithm

ExtendedEuclid(A, B)

```
Input  $A$ and $B$ in $\mathbb{K}[x]$.

Output  A gcd $G$ of $A$ and $B$, and cofactors $U$ and $V$.

  1. $R_0 := A$; $U_0 := 1$; $V_0 := 0$; $R_1 := B$; $U_1 := 0$; $V_1 := 1$; $i := 1$.
  2. While $R_i$ is non-zero, do:
      (a) $(Q_i, R_{i+1}) := \text{QuotRem}(R_{i-1}, R_i)$        $\# R_{i-1} = Q_i R_i + R_{i+1}$
      (b) $U_{i+1} := U_{i-1} - Q_i U_i$; $V_{i+1} := V_{i-1} - Q_i V_i$.
      (c) $i := i + 1$.
  3. Return $\big(R_{i-1}, U_{i-1}, V_{i-1}\big)$.
```

▷ Correctness: $R_i = U_i A + V_i B$ (by induction):

$$R_{i+1} = R_{i-1} - Q_i R_i = U_{i-1} A + V_{i-1} B - Q_i (U_i A + V_i B) = U_{i+1} A + V_{i+1} B$$

▷ Quadratic complexity: $O\big(\deg(A)\deg(B)\big)$ operations in $\mathbb{K}$

Resultants

Definition

The Sylvester matrix of $A = a_m x^m + \cdots + a_0 \in \mathbb{K}[x]$, $(a_m \neq 0)$, and of $B = b_n x^n + \cdots + b_0 \in \mathbb{K}[x]$, $(b_n \neq 0)$, is the square matrix of size $m+n$

$$\mathsf{Syl}(A,B) = \begin{bmatrix} a_m & a_{m-1} & \dots & a_0 & & & \\ & a_m & a_{m-1} & \dots & a_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & a_m & a_{m-1} & \dots & a_0 \\ b_n & b_{n-1} & \dots & b_0 & & & \\ & b_n & b_{n-1} & \dots & b_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & b_n & b_{n-1} & \dots & b_0 \end{bmatrix}$$

The resultant $\mathsf{Res}(A,B)$ of A and B is the determinant of $\mathsf{Syl}(A,B)$.

▷ Definition extends to polynomials over any commutative ring R.

Key observation

If $A = a_m x^m + \cdots + a_0$ and $B = b_n x^n + \cdots + b_0$, then

$$\begin{bmatrix} a_m & a_{m-1} & \dots & a_0 & & & \\ & \ddots & \ddots & & & \ddots & \\ & & a_m & a_{m-1} & & \dots & a_0 \\ b_n & b_{n-1} & \dots & b_0 & & & \\ & \ddots & \ddots & & & \ddots & \\ & & b_n & b_{n-1} & & \dots & b_0 \end{bmatrix} \times \begin{bmatrix} \alpha^{m+n-1} \\ \vdots \\ \alpha \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha^{n-1}A(\alpha) \\ \vdots \\ A(\alpha) \\ \alpha^{m-1}B(\alpha) \\ \vdots \\ B(\alpha) \end{bmatrix}$$

Corollary: If $A(\alpha) = B(\alpha) = 0$, then $\mathsf{Res}\,(A, B) = 0$.

Example: the discriminant

The discriminant of A is the resultant of A and of its derivative A'.

E.g. for $A = ax^2 + bx + c$,

$$\mathsf{Disc}(A) = \mathsf{Res}\,(A, A') = \det \begin{bmatrix} a & b & c \\ 2a & b & \\ & 2a & b \end{bmatrix} = -a(b^2 - 4ac).$$

E.g. for $A = ax^3 + bx + c$,

$$\mathsf{Disc}(A) = \mathsf{Res}\,(A, A') = \det \begin{bmatrix} a & 0 & b & c & \\ & a & 0 & b & c \\ 3a & 0 & b & & \\ & 3a & 0 & b & \\ & & 3a & 0 & b \end{bmatrix} = a^2(4b^3 + 27ac^2).$$

$\triangleright$ The discriminant vanishes when A and A' have a common root, that is when A has a multiple root.

Main properties

- Link with gcd $\mathsf{Res}\,(A,B) = 0$ if and only if $\gcd(A,B)$ is non-constant.
- Elimination property
  There exist $U, V \in \mathbb{K}[x]$ not both zero, with $\deg(U) < n$, $\deg(V) < m$ and such that the following Bézout identity holds in $\mathbb{K} \cap (A,B)$:
  $$\mathsf{Res}\,(A,B) = UA + VB.$$
- Poisson formula
  If $A = a(x-\alpha_1)\cdots(x-\alpha_m)$ and $B = b(x-\beta_1)\cdots(x-\beta_n)$, then
  $$\mathsf{Res}\,(A,B) \;=\; a^n b^m \prod_{i,j}(\alpha_i - \beta_j) \;=\; a^n \prod_{1\le i\le m} B(\alpha_i).$$
- Multiplicativity
  $$\mathsf{Res}\,(A{\cdot}B, C) = \mathsf{Res}\,(A,C){\cdot}\mathsf{Res}\,(B,C), \quad \mathsf{Res}\,(A, B{\cdot}C) = \mathsf{Res}\,(A,B){\cdot}\mathsf{Res}\,(A,C).$$

Proof of Poisson's formula

▷ Direct consequence of the key observation:

If $A = (x-\alpha_1)\cdots(x-\alpha_m)$ and $B = (x-\beta_1)\cdots(x-\beta_n)$ then

$$\mathsf{Syl}(A,B) \times \begin{bmatrix} \beta_1^{m+n-1} & \dots & \beta_n^{m+n-1} & \alpha_1^{m+n-1} & \dots & \alpha_m^{m+n-1} \\ \vdots & & \vdots & \vdots & & \vdots \\ \beta_1 & \dots & \beta_n & \alpha_1 & \dots & \alpha_m \\ 1 & \dots & 1 & 1 & \dots & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \beta_1^{n-1}A(\beta_1) & \dots & \beta_n^{n-1}A(\beta_n) & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ A(\beta_1) & \dots & A(\beta_n) & 0 & \dots & 0 \\ 0 & \dots & 0 & \alpha_1^{m-1}B(\alpha_1) & \dots & \alpha_m^{m-1}B(\alpha_m) \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & B(\alpha_1) & \dots & B(\alpha_m) \end{bmatrix}$$

▷ To conclude, take determinants and use Vandermonde's formula

Application: computation with algebraic numbers

Let $A = \prod_i (x - \alpha_i)$ and $B = \prod_j (x - \beta_j)$ be polynomials of $\mathbb{K}[x]$. Then

$$A \oplus B := \prod_{i,j} (t - (\alpha_i + \beta_j)) = \mathsf{Res}_x(A(x), B(t - x)),$$

$$\prod_{i,j} (t - (\beta_j - \alpha_i)) = \mathsf{Res}_x(A(x), B(t + x)),$$

$$A \otimes B := \prod_{i,j} (t - \alpha_i \beta_j) = \mathsf{Res}_x(A(x), x^{\deg B} B(t/x)),$$

$$\prod_i (t - B(\alpha_i)) = \mathsf{Res}_x(A(x), t - B(x)).$$

In particular, the set $\overline{\mathbb{Q}}$ of algebraic numbers is a field.

Proof: Poisson's formula. E.g., first one: $\prod_i B(t - \alpha_i) = \prod_{i,j} (t - \alpha_i - \beta_j)$.

▷ The same formulas apply mutatis mutandis to algebraic power series.

A beautiful identity of Ramanujan's

$$\frac{\sin\frac{2\pi}{7}}{\sin^2\frac{3\pi}{7}} - \frac{\sin\frac{\pi}{7}}{\sin^2\frac{2\pi}{7}} + \frac{\sin\frac{3\pi}{7}}{\sin^2\frac{\pi}{7}} = 2\sqrt{7}.$$

- ▷ If $p = \pi/7$ then $\sin(kp) = (\alpha^k - \alpha^{-k})/(2\,i)$, where $\alpha = e^{ip}$, with $\alpha^7 = -1$
- ▷ Since $\alpha \in \overline{\mathbb{Q}}$, any rational expression in the $\sin(kp)$ is in $\mathbb{Q}(i)(\alpha)$ thus in $\overline{\mathbb{Q}}$

```
> f:=sin(2*p)/sin(3*p)^2-sin(p)/sin(2*p)^2+sin(3*p)/sin(p)^2:
> expand(convert(f, exp)):
> F:=normal(subs(exp(I*p)=alpha, %));
```

$$\frac{2\,i\,(\alpha^{16} + 5\,\alpha^{14} + 12\,\alpha^{12} + \alpha^{11} + 20\,\alpha^{10} + 3\,\alpha^9 + 23\,\alpha^8 + 3\,\alpha^7 + 20\,\alpha^6 + \alpha^5 + 12\,\alpha^4 + 5\,\alpha^2 + 1)}{\alpha\,(\alpha^2 - 1)\,(\alpha^2 + 1)^2\,(\alpha^4 + \alpha^2 + 1)^2}$$

- ▷ In particular our LHS, $F(\alpha) = \frac{N(\alpha)}{D(\alpha)}$, is an algebraic number
- ▷ Resultant $R(t) := \mathrm{Res}_x(x^7 + 1, t \cdot D(x) - N(x))$ annihilates $F(\alpha)$

```
> R:=factor(resultant(x^7+1, t*denom(F)-numer(F), x));
```

$$-1274\,i\,(t^2 - 28)^3$$

Shanks' 1974 identities

$$\sqrt{11+2\sqrt{29}}+\sqrt{16-2\sqrt{29}+2\sqrt{55-10\sqrt{29}}}=\sqrt{5}+\sqrt{22+2\sqrt{5}}$$

$$\sqrt{\sqrt{m+n}+\sqrt{n}}+\sqrt{\sqrt{m+n}+m-\sqrt{n}+2\sqrt{m\left(\sqrt{m+n}-\sqrt{n}\right)}}$$

$$=\sqrt{m}+\sqrt{2\sqrt{m+n}+2\sqrt{m}}$$

Two more exercises for next Thursday

(1) Let $P, Q \in \mathbb{K}[x]$ be two polynomials, and let $N \in \mathbb{N} \setminus \{0\}$.

(a) Show that the unique monic polynomial in $\mathbb{K}[x]$ whose roots are the N-th powers of the roots of P can be obtained by a resultant computation.

(b) If P is the minimal polynomial of an algebraic number α, show that one can determine an annihilating polynomial of $Q(\alpha)$ using a resultant.

(2) The aim of this exercise is to prove algorithmically the following identity:

$$\sqrt[3]{\sqrt[3]{2} - 1} = \sqrt[3]{\frac{1}{9}} - \sqrt[3]{\frac{2}{9}} + \sqrt[3]{\frac{4}{9}}. \qquad (1)$$

Let $a = \sqrt[3]{2}$ and $b = \sqrt[3]{\frac{1}{9}}$.

(a) Determine $P_c \in \mathbb{Q}[x]$ annihilating $c = 1 - a + a^2$, using a resultant.

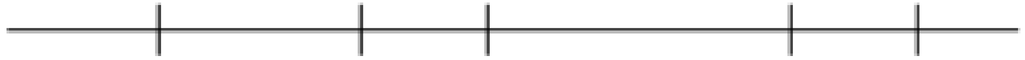
(b) Deduce $P_R \in \mathbb{Q}[x]$ annihilating the RHS of (1), by another resultant.

(c) Show that the polynomial computed in (b) also annihilates the LHS of (1).

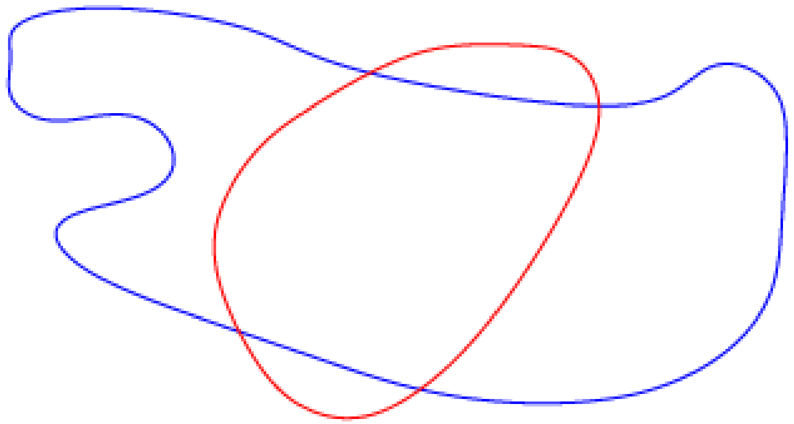
(d) Conclude.

Systems of two equations and two unknowns

Geometrically, roots of a polynomial $f \in \mathbb{Q}[x]$ correspond to points on a line.



Roots of polynomials $A \in \mathbb{Q}[x, y]$ correspond to plane curves $A = 0$.

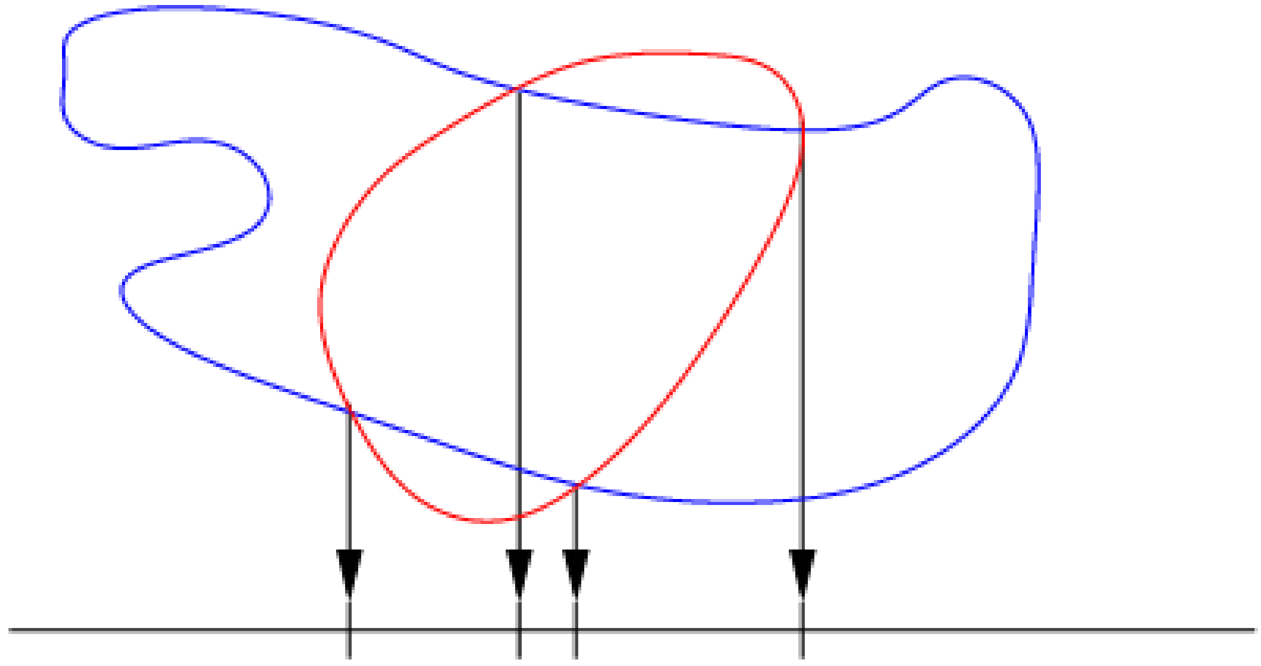


Let now A and B be in $\mathbb{Q}[x, y]$. Then:

- either the curves $A = 0$ and $B = 0$ have a common component,
- or they intersect in a finite number of points.

Application: Resultants compute projections

Theorem. Let $A = a_m y^m + \cdots$ and $B = b_n y^n + \cdots$ be polynomials in $\mathbb{Q}[x][y]$. The roots of $\mathsf{Res}_y(A,B) \in \mathbb{Q}[x]$ are either the abscissas of points in the intersection $A = B = 0$, or common roots of a_m and b_n.



Proof. Elimination property: $\mathsf{Res}\,(A,B) = UA + VB$, for $U, V \in \mathbb{Q}[x,y]$.

Thus $A(\alpha,\beta) = B(\alpha,\beta) = 0$ implies $\mathsf{Res}_y(A,B)(\alpha) = 0$

Application: implicitization of parametric curves

Task: Given a rational parametrization of a curve

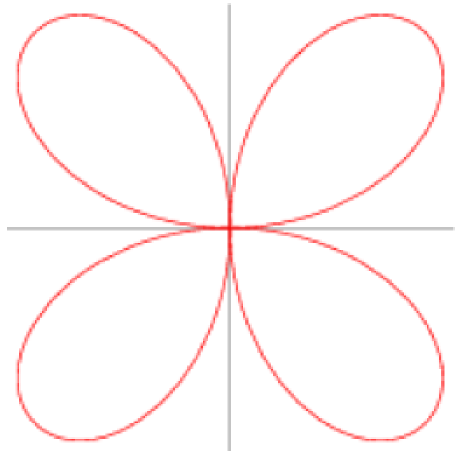
$$x = A(t), \quad y = B(t), \qquad A, B \in \mathbb{K}(t),$$

compute a non-trivial polynomial in x and y vanishing on the curve.

Recipe: take the resultant in t of numerators of $x - A(t)$ and $y - B(t)$.

Example: for the four-leaved clover (a.k.a. quadrifolium) given by

$$x = \frac{4t(1-t^2)^2}{(1+t^2)^3}, \quad y = \frac{8t^2(1-t^2)}{(1+t^2)^3},$$



$$\mathrm{Res}_t((1+t^2)^3x - 4t(1-t^2)^2, (1+t^2)^3y - 8t^2(1-t^2)) = 2^{24}\left((x^2+y^2)^3 - 4x^2y^2\right).$$

Computation of the resultant

An Euclidean-type algorithm for the resultant bases on:

- If $A = QB + R$, and $R \neq 0$, then (by Poisson's formula)

$$\mathsf{Res}\,(A, B) = (-1)^{\deg A \deg B}\, \mathsf{lc}(B)^{\deg A - \deg R}\, \mathsf{Res}\,(B, R).$$

- If B is constant, then $\quad \mathsf{Res}\,(A, B) = B^{\deg A}$.

If $(R_0, \dots, R_{N-1}, R_N = \gcd(A, B), 0)$ is the remainder sequence produced by the Euclidean algorithm for $R_0 = A$ and $R_1 = B$, then

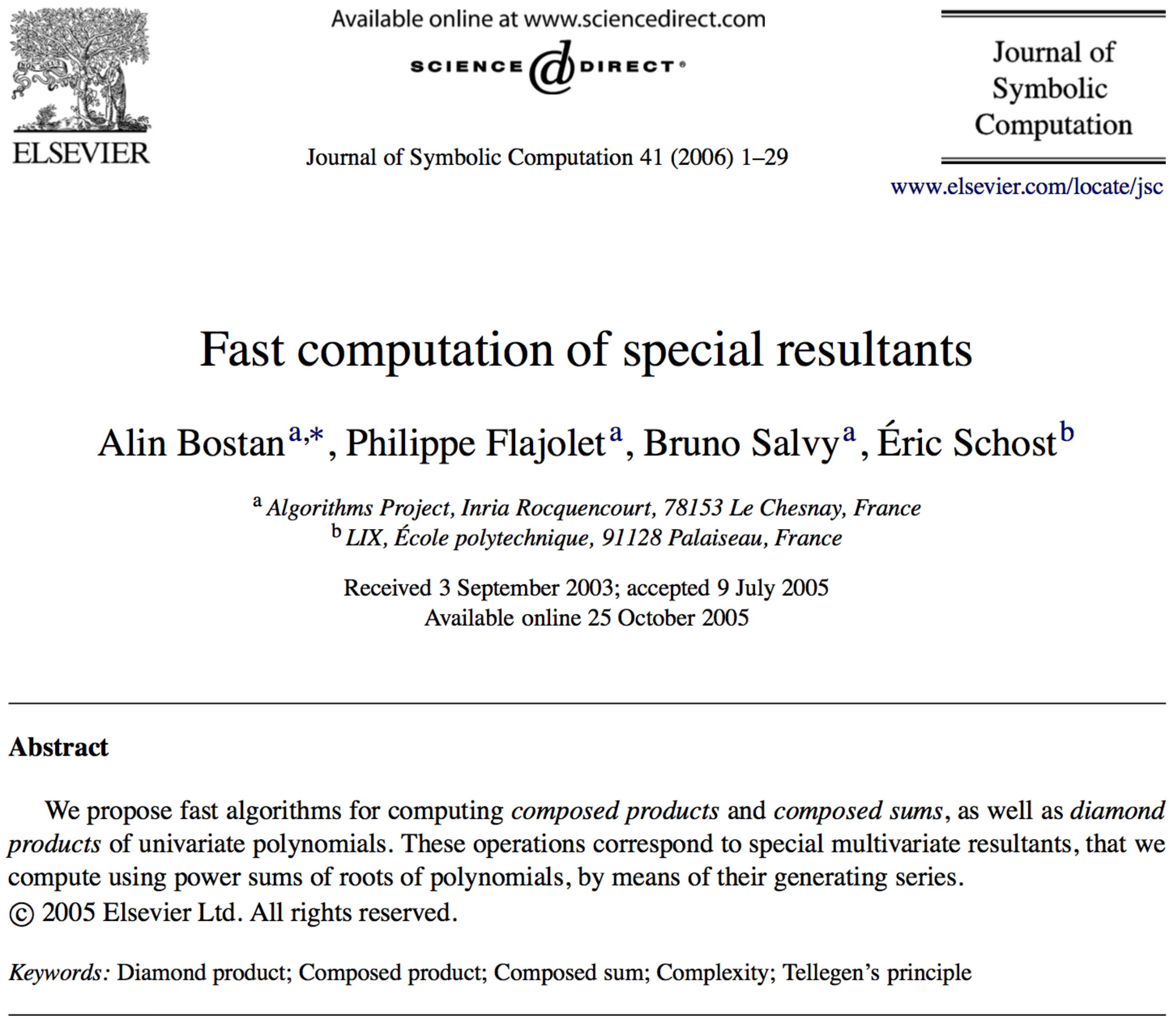
- either $\deg R_N$ is non-constant, and $\mathsf{Res}\,(A, B) = 0$,
- or $\mathsf{Res}\,(A, B) = R_N^{\deg R_{N-1}} \prod_{i=0}^{N-2} (-1)^{\deg R_i \deg R_{i+1}}\, \mathsf{lc}(R_{i+1})^{\deg R_i - \deg R_{i+2}}$.

▷ This leads to a $O(N^2)$ algorithm for $\mathsf{Res}\,(A, B)$, where $\deg(A), \deg(B) \leq N$.

▷ A divide-and-conquer $O(\mathsf{M}(N) \log N)$ algorithm requires extra-work.

Bonus

1. Fast Manipulation of Algebraic Numbers



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Symbolic Computation 41 (2006) 1–29

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Fast computation of special resultants

Alin Bostan[a,*], Philippe Flajolet[a], Bruno Salvy[a], Éric Schost[b]

[a] *Algorithms Project, Inria Rocquencourt, 78153 Le Chesnay, France*
[b] *LIX, École polytechnique, 91128 Palaiseau, France*

Received 3 September 2003; accepted 9 July 2005
Available online 25 October 2005

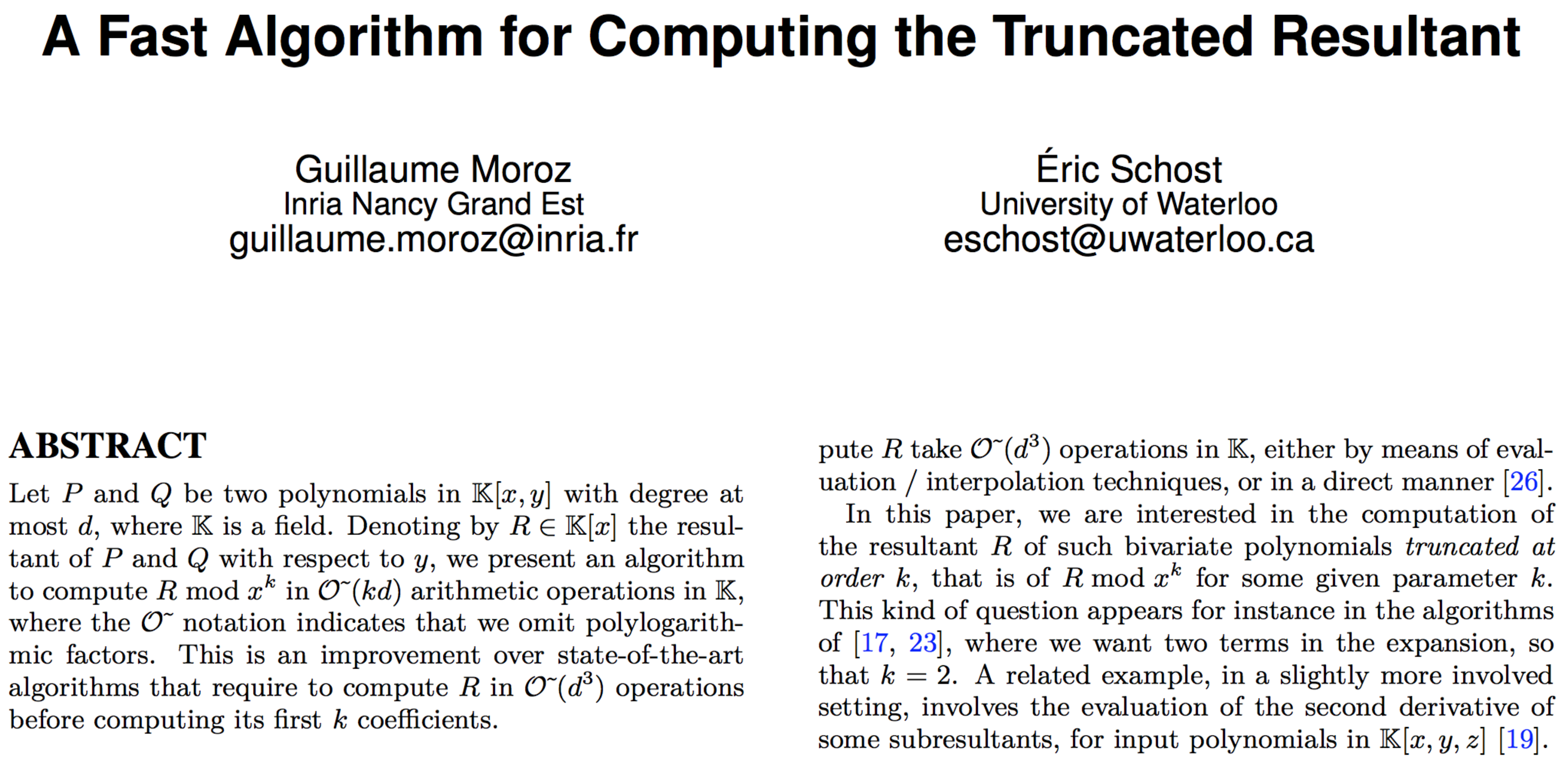
Abstract

We propose fast algorithms for computing *composed products* and *composed sums*, as well as *diamond products* of univariate polynomials. These operations correspond to special multivariate resultants, that we compute using power sums of roots of polynomials, by means of their generating series.

Keywords: Diamond product; Composed product; Composed sum; Complexity; Tellegen's principle

▷ Composed sum $A \oplus B$ and composed product $A \otimes B$ in $\tilde{O}(\deg A \cdot \deg B)$

2. Computing the Truncated Resultant



A Fast Algorithm for Computing the Truncated Resultant

Guillaume Moroz
Inria Nancy Grand Est
guillaume.moroz@inria.fr

Éric Schost
University of Waterloo
eschost@uwaterloo.ca

ABSTRACT

Let P and Q be two polynomials in $\mathbb{K}[x,y]$ with degree at most d, where $\mathbb{K}$ is a field. Denoting by $R \in \mathbb{K}[x]$ the resultant of P and Q with respect to y, we present an algorithm to compute $R \bmod x^k$ in $O\tilde{~}(kd)$ arithmetic operations in $\mathbb{K}$, where the $O\tilde{~}$ notation indicates that we omit polylogarithmic factors. This is an improvement over state-of-the-art algorithms that require to compute R in $O\tilde{~}(d^3)$ operations before computing its first k coefficients.

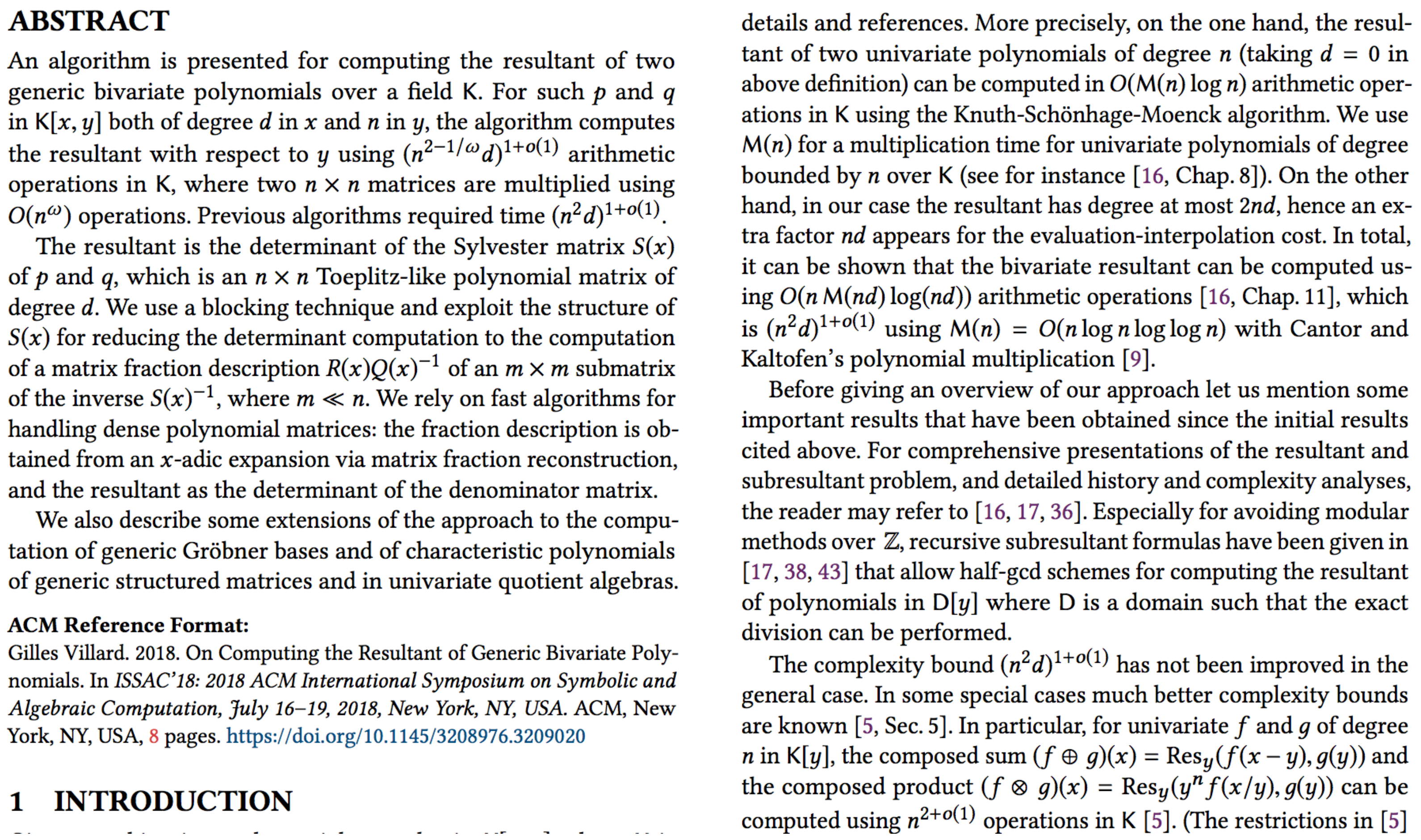
pute R take $O\tilde{~}(d^3)$ operations in $\mathbb{K}$, either by means of evaluation / interpolation techniques, or in a direct manner [26].

In this paper, we are interested in the computation of the resultant R of such bivariate polynomials *truncated at order* k, that is of $R \bmod x^k$ for some given parameter k. This kind of question appears for instance in the algorithms of [17, 23], where we want two terms in the expansion, so that $k = 2$. A related example, in a slightly more involved setting, involves the evaluation of the second derivative of some subresultants, for input polynomials in $\mathbb{K}[x,y,z]$ [19].

[Moroz & Schost, ISSAC 2016]

▷ $\mathrm{Res}_y(P(x,y),Q(x,y)) \bmod x^k$ in $\tilde{O}(kd)$, where $d = \max(\deg P, \deg Q)$

3. Resultant of Generic Bivariate Polynomials



ABSTRACT

An algorithm is presented for computing the resultant of two generic bivariate polynomials over a field K. For such p and q in $K[x, y]$ both of degree d in x and n in y, the algorithm computes the resultant with respect to y using $(n^{2-1/\omega}d)^{1+o(1)}$ arithmetic operations in K, where two $n \times n$ matrices are multiplied using $O(n^\omega)$ operations. Previous algorithms required time $(n^2 d)^{1+o(1)}$.

The resultant is the determinant of the Sylvester matrix $S(x)$ of p and q, which is an $n \times n$ Toeplitz-like polynomial matrix of degree d. We use a blocking technique and exploit the structure of $S(x)$ for reducing the determinant computation to the computation of a matrix fraction description $R(x)Q(x)^{-1}$ of an $m \times m$ submatrix of the inverse $S(x)^{-1}$, where $m \ll n$. We rely on fast algorithms for handling dense polynomial matrices: the fraction description is obtained from an x-adic expansion via matrix fraction reconstruction, and the resultant as the determinant of the denominator matrix.

We also describe some extensions of the approach to the computation of generic Gröbner bases and of characteristic polynomials of generic structured matrices and in univariate quotient algebras.

ACM Reference Format:
Gilles Villard. 2018. On Computing the Resultant of Generic Bivariate Polynomials. In *ISSAC'18: 2018 ACM International Symposium on Symbolic and Algebraic Computation, July 16–19, 2018, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3208976.3209020

1 INTRODUCTION

details and references. More precisely, on the one hand, the resultant of two univariate polynomials of degree n (taking $d = 0$ in above definition) can be computed in $O(\mathsf{M}(n) \log n)$ arithmetic operations in K using the Knuth-Schönhage-Moenck algorithm. We use $\mathsf{M}(n)$ for a multiplication time for univariate polynomials of degree bounded by n over K (see for instance [16, Chap. 8]). On the other hand, in our case the resultant has degree at most $2nd$, hence an extra factor nd appears for the evaluation-interpolation cost. In total, it can be shown that the bivariate resultant can be computed using $O(n\,\mathsf{M}(nd) \log(nd))$ arithmetic operations [16, Chap. 11], which is $(n^2 d)^{1+o(1)}$ using $\mathsf{M}(n) = O(n \log n \log\log n)$ with Cantor and Kaltofen's polynomial multiplication [9].

Before giving an overview of our approach let us mention some important results that have been obtained since the initial results cited above. For comprehensive presentations of the resultant and subresultant problem, and detailed history and complexity analyses, the reader may refer to [16, 17, 36]. Especially for avoiding modular methods over $\mathbb{Z}$, recursive subresultant formulas have been given in [17, 38, 43] that allow half-gcd schemes for computing the resultant of polynomials in $D[y]$ where D is a domain such that the exact division can be performed.

The complexity bound $(n^2 d)^{1+o(1)}$ has not been improved in the general case. In some special cases much better complexity bounds are known [5, Sec. 5]. In particular, for univariate f and g of degree n in $K[y]$, the composed sum $(f \oplus g)(x) = \mathrm{Res}_y(f(x-y), g(y))$ and the composed product $(f \otimes g)(x) = \mathrm{Res}_y(y^n f(x/y), g(y))$ can be computed using $n^{2+o(1)}$ operations in K [5]. (The restrictions in [5]

[Villard, ISSAC 2018]

▷ $\mathrm{Res}_y(P(x,y), Q(x,y))$ of *generic* P, Q of degree d in $\tilde{O}(d^{3-1/\omega})$