Newton iteration for power series
&
Fast Evaluation and Interpolation

Alin Bostan

MPRI COMPALG
October 1st, 2025

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 $Q^2, \ldots, Q^d =: R$ and $R^2, \ldots, R^{d-1}$ 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 $\quad O(d\,\mathsf{M}(n))$

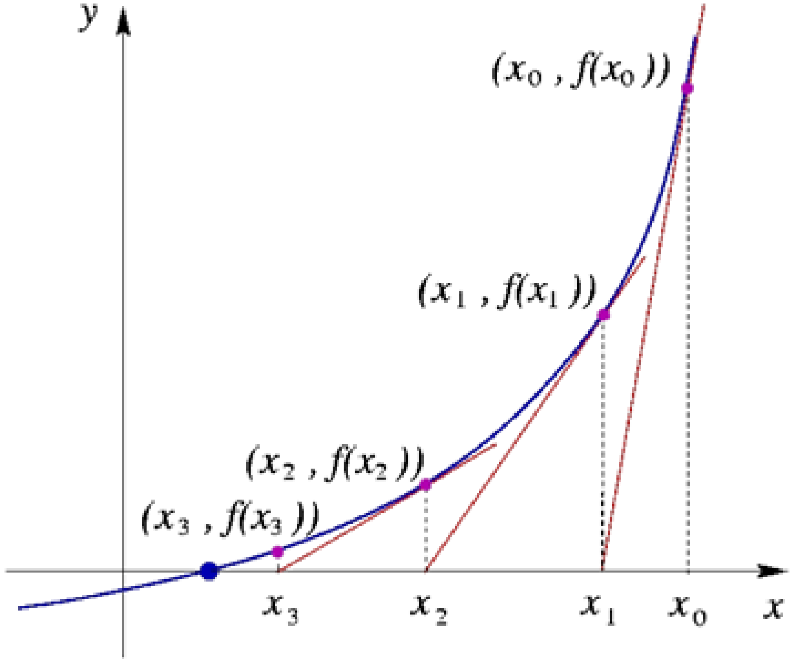
Newton Iteration

Newton's tangent method: real case

[Newton, 1671]

To solve (find a real root of) an equation $f(x) = 0$ for a sufficiently smooth f:

1. Make a rough estimate to define an initial approximation x_0
2. Evaluate the intercept x_1 of the tangent line to $f(x) = 0$ at $(x_0, f(x_0))$
3. Use x_1 as a new (finer) estimate and repeat the procedure



$(x_{\kappa+1}, 0)$ belongs to $y - f(x_\kappa) = f'(x_\kappa) \cdot (x - x_\kappa) \implies \boxed{x_{\kappa+1} = x_\kappa - f(x_\kappa)/f'(x_\kappa)}$

Newton's tangent method: real case

[Newton, 1671]

To compute better and better approximations for $\sqrt{2}$, take $f(x) = x^2 - 2$:

$$x_{\kappa+1} = \mathcal{N}(x_\kappa) = x_\kappa - (x_\kappa^2 - 2)/(2x_\kappa), \quad x_0 = 1$$

```
> x[0]:=1;
> for k from 0 to 4 do
        x[k+1]:=evalf(x[k] - (x[k]^2 - 2)/(2*x[k]), 32); od;
```

$$x_1 = 1.5000000000000000000000000000000$$

$$x_2 = 1.4166666666666666666666666666667$$

$$x_3 = 1.4142156862745098039215686274510$$

$$x_4 = 1.4142135623746899106262955788902$$

$$x_5 = 1.4142135623730950488016896235025$$

Newton's tangent method: real case

[Newton, 1671]

To compute better and better approximations for $\sqrt{2}$, take $f(x) = x^2 - 2$:

$$x_{\kappa+1} = \mathcal{N}(x_\kappa) = x_\kappa - (x_\kappa^2 - 2)/(2x_\kappa), \quad x_0 = 1$$

```
> x[0]:=1;
> for k from 0 to 4 do
        x[k+1]:=evalf(x[k] - (x[k]^2 - 2)/(2*x[k]), 2^(k+1)); od;
```

$$x_1 = 1.5$$

$$x_2 = 1.417$$

$$x_3 = 1.4142163$$

$$x_4 = 1.414213562375745$$

$$x_5 = 1.4142135623730950488016912069469$$

6 *The Method of* FLUXIONS,

Refolution of affected Equations may be compendioufly perform'd in Numbers, and then I fhall apply the fame to Species.

20. Let this Equation $y^3 - 2y - 5 = 0$ be propofed to be refolved, and let 2 be a Number (any how found) which differs from the true Root lefs than by a tenth part of itfelf. Then I make $2 + p = y$, and fubftitute $2 + p$ for y in the given Equation, by which is produced a new Equation $p^3 + 6p^2 + 10p - 1 = 0$, whofe Root is to be fought for, that it may be added to the Quote. Thus rejecting $p^3 + 6p^2$ becaufe of its fmallnefs, the remaining Equation $10p - 1 = 0$, or $p = 0{,}1$, will approach very near to the truth. Therefore I write this in the Quote, and fuppofe $0{,}1 + q = p$, and fubftitute this fictitious Value of p as before, which produces $q^3 + 6{,}3q^2 + 11{,}23q + 0{,}061 = 0$. And fince $11{,}23q + 0{,}061 = 0$ is near the truth, or $q = -0{,}0054$ nearly, (that is, dividing $0{,}061$ by $11{,}23$, till fo many Figures arife as there are places between the firft Figures of this, and of the principal Quote exclufively, as here there are two places between 2 and $0{,}005$) I write $-0{,}0054$ in the lower part of the Quote, as being negative; and fuppofing $-0{,}0054 + r = q$, I fubftitute this as before. And thus I continue the Operation as far as I pleafe, in the manner of the following Diagram:

$y^3 - 2y - 5 = 0$	$+2{,}10000000$ $-0{,}00544852$ $+2{,}09455148$, &c. $= y$
$2 + p = y.$ $\quad +y^3$ $-2y$ -5	$+8 + 12p + 6p^2 + p^3$ $-4 - 2p$ -5
The Sum	$-1 + 10p + 6p^2 + p^3$
$0{,}1 + q = p.$ $\quad +p^3$ $+6p^2$ $+10p$ -1	$+0{,}001 + 0{,}03q + 0{,}3q^2 + q^3$ $+0{,}06 + 1{,}2 + 6,$ $+1, + 10,$ $-1,$
The Sum	$0{,}061 + 11{,}23q + 6{,}3q^2 + q^3$
$-0{,}0054 + r = q.$ $\quad q^3$ $+6{,}3q^2$ $+11{,}23q$ $+0{,}061$	$-0{,}000000157464 + 0{,}00008748r - 0{,}0162r^2 + r^3$ $+0{,}000183708 - 0{,}06804 + 6{,}3$ $-0{,}060642 + 11{,}23$ $+0{,}061$
The Sum	$+0{,}0005416 + 11{,}162r$
$-0{,}00004852 + s = r.$	

[Newton, 1671 – English translation "Method of Fluxions" (1736) by Colson]

Newton's tangent method: power series case

To compute better and better approximations for $\sqrt{1-t}$, iterate:

$$x_{\kappa+1} = \mathcal{N}(x_\kappa) = x_\kappa - (x_\kappa^2 - (1-t))/(2x_\kappa), \quad x_0 = 1$$

```
> x[0]:=1;
> for k from 0 to 2 do
>     x[k+1]:=series(x[k]-(x[k]^2 - (1-t))/(2*x[k]),t,10); od;
```

$$x_0 = 1$$

$$x_1 = 1 - \frac{1}{2}t$$

$$x_2 = 1 - \frac{1}{2}t - \frac{1}{8}t^2 - \frac{1}{16}t^3 - \frac{1}{32}t^4 - \frac{1}{64}t^5 - \frac{1}{128}t^6 - \frac{1}{256}t^7 - \frac{1}{512}t^8 - \frac{1}{1024}t^9 + \cdots$$

$$x_3 = 1 - \frac{1}{2}t - \frac{1}{8}t^2 - \frac{1}{16}t^3 - \frac{5}{128}t^4 - \frac{7}{256}t^5 - \frac{21}{1024}t^6 - \frac{33}{2048}t^7 - \frac{107}{8192}t^8 - \frac{177}{16384}t^9 + \cdots$$

Newton's tangent method: power series case

To compute better and better approximations for $\sqrt{1-t}$, iterate:

$$x_{\kappa+1} = \mathcal{N}(x_\kappa) = x_\kappa - (x_\kappa^2 - (1-t))/(2x_\kappa), \quad x_0 = 1$$

```
> x[0]:=1;
> for k from 0 to 2 do
>     x[k+1]:=convert(series(x[k] - (x[k]^2 - (1-t))/(2*x[k]),
              t, 2^(k+1)), polynom); od;
```

$$x_0 = 1$$

$$x_1 = 1 - \frac{1}{2}t$$

$$x_2 = 1 - \frac{1}{2}t - \frac{1}{8}t^2 - \frac{1}{16}t^3$$

$$x_3 = 1 - \frac{1}{2}t - \frac{1}{8}t^2 - \frac{1}{16}t^3 - \frac{5}{128}t^4 - \frac{7}{256}t^5 - \frac{21}{1024}t^6 - \frac{33}{2048}t^7$$

Formal Newton iteration – principle and main result

To solve $\varphi(g) = 0$ in $\mathbb{K}[[x]]$ ($\varphi \in \mathbb{K}[[x]][[y]]$, $\varphi(0) = 0$ and $\varphi_y(0) \neq 0$), iterate

$$g_{\kappa+1} = g_\kappa - \frac{\varphi(g_\kappa)}{\varphi_y(g_\kappa)} \mod x^{2^{\kappa+1}}$$

▷ "1-line proof":

$$g - g_{\kappa+1} = g - g_\kappa + \frac{\varphi(g) + (g_\kappa - g)\varphi_y(g) + O((g-g_\kappa)^2)}{\varphi_y(g) + O(g - g_\kappa)} = O((g-g_\kappa)^2)$$

▷ The number of correct coefficients doubles after each iteration

▷ Total cost $\leq$ **2** $\times$ $\Big($ the cost of the last iteration $\Big)$

Theorem [Cook 1966, Sieveking 1972 & Kung 1974, Brent 1975]
Division, logarithm and exponential of power series in $\mathbb{K}[[x]]$ can be computed at precision N using $O(\mathsf{M}(N))$ operations in $\mathbb{K}$

Division and logarithm of power series

[Sieveking-Kung, 1972]

To compute the reciprocal of $f \in \mathbb{K}[[x]]$, choose $\varphi(g) = 1/g - f$:

$$g_0 = \frac{1}{f_0} \quad \text{and} \quad g_{\kappa+1} = g_\kappa + g_\kappa(1 - f g_\kappa) \mod x^{2^{\kappa+1}} \quad \text{for } \kappa \geq 0$$

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

Corollary: division of power series at precision N in $O(\mathsf{M}(N))$

Corollary: Logarithm $\log(f) := -\sum_{i \geq 1} \frac{(1-f)^i}{i}$ of $f \in 1 + x\mathbb{K}[[x]]$ in $O(\mathsf{M}(N))$:

- compute the Taylor expansion of $h = f'/f$ modulo x^{N-1} $O(\mathsf{M}(N))$
- take the antiderivative (i.e., primitive with 0 constant term) of h $O(N)$

Details on power series inversion

Lemma Given $F \in \mathbb{K}[[x]]$ with $F(0) \neq 0$, $n \in \mathbb{N}_{>0}$, and $G \in \mathbb{K}[[x]]$ s.t. $G - F^{-1} = O(x^n)$, then $\mathcal{N}(G) := 2G - GFG$ satisfies $\mathcal{N}(G) - F^{-1} = O(x^{2n})$.

Proof: Writing $1 - GF = x^n H$, then inverting $F = G^{-1}(1 - x^n H)$ yields

$$F^{-1} = (1 + x^n H + O(x^{2n}))G = G + (1 - GF)G + O(x^{2n}) = \mathcal{N}(G) + O(x^{2n}).$$

Algorithm (series inversion by Newton iteration)

Input: Truncation T to order $N \in \mathbb{N}_{>0}$ of a series $F \in \mathbb{K}[[x]]$ with $F(0) \neq 0$.

Output: The truncation S to order N of the inverse series F^{-1}.

If $N = 1$, return $T(0)^{-1}$. Otherwise:

1. Recursively compute the truncation G to order $\lceil N/2 \rceil$ of T^{-1}.
2. Return $S := G + \mathrm{rem}((1 - GT)G, x^N)$.

Details on power series inversion

Algorithm (series inversion by Newton iteration)

Input Truncation T to order $N \in \mathbb{N}_{>0}$ of a series $F \in \mathbb{K}[[x]]$ with $F(0) \neq 0$.

Output The truncation S to order N of the inverse series F^{-1}.

If $N = 1$, return $T(0)^{-1}$. Otherwise:

1. Recursively compute the truncation G to order $\lceil N/2 \rceil$ of T^{-1}.
2. Return $S := G + \operatorname{rem}((1 - GT)G, x^N)$.

Correctness proof Assume $T^{-1} = G + O(x^{\lceil N/2 \rceil})$ by induction. By Lemma,

$$\mathcal{N}(G) - T^{-1} = O(x^{2\lceil N/2 \rceil}) = O(x^N).$$

Write $F = T + O(x^N) = T(1 + O(x^N))$, so that $F^{-1} = T^{-1} + O(x^N)$. Then,

$$F^{-1} - S = (F^{-1} - T^{-1}) + (T^{-1} - \mathcal{N}(G)) + (\mathcal{N}(G) - S) = O(x^N).$$

Application: 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

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

Algorithm:

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

Exponentials of power series and 1st order LDE

[Brent, 1975]

To compute the exponential $\exp(f) := \sum_{i\geq 0} \frac{f^i}{i!}$, choose $\varphi(g) = \log(g) - f$:

$$g_0 = 1 \quad \text{and} \quad g_{\kappa+1} = g_\kappa - g_\kappa\left(\log(g_\kappa) - f\right) \mod x^{2^{\kappa+1}} \quad \text{for } \kappa \geq 0.$$

Complexity: $\mathsf{C}(N) = \mathsf{C}(N/2) + O(\mathsf{M}(N)) \implies \mathsf{C}(N) = O(\mathsf{M}(N))$

Corollary: Solve first order linear differential equations $af' + bf = c$ in $O(\mathsf{M}(N))$

- if $c = 0$ then the solution is $f_0 = \exp\left(-\int b/a\right)$ $O(\mathsf{M}(N))$
- else, variation of constants: $f = f_0 g$, where $g' = c/(af_0)$ $O(\mathsf{M}(N))$

▷ Main difficulty for higher orders: for non-commutativity reasons, the matrix exponential $Y(x) = \exp(\int A(x))$ is not a solution of $Y' = A(x)Y$.

▷ [B.-Chyzak-Ollivier-Salvy-Schost-Sedoglavic 2007] $O(\mathsf{M}(N))$ for any order

Application: conversion coefficients $\leftrightarrow$ power sums

[Schönhage, 1982]

Any polynomial $F = x^n + a_1 x^{n-1} + \cdots + a_n$ in $\mathbb{K}[x]$ can be represented by its first n power sums $S_i = \sum_{F(\alpha)=0} \alpha^i$

Conversions coefficients $\leftrightarrow$ power sums can be performed

- either in $O(n^2)$ using Newton identities (naive way):

$$i a_i + S_1 a_{i-1} + \cdots + S_i = 0, \quad 1 \leq i \leq n$$

- or in $O(\mathsf{M}(n))$ using generating series

$$\frac{\mathsf{rev}(F)'}{\mathsf{rev}(F)} = -\sum_{i \geq 0} S_{i+1} x^i \quad \Longleftrightarrow \quad \mathsf{rev}(F) = \exp\left(-\sum_{i \geq 1} \frac{S_i}{i} x^i\right)$$

Application: special bivariate resultants

[B.-Flajolet-S-Schost, 2006]

Composed products and sums: manipulation of algebraic numbers

$$F \otimes G = \prod_{F(\alpha)=0, G(\beta)=0} (x - \alpha\beta), \quad F \oplus G = \prod_{F(\alpha)=0, G(\beta)=0} (x - (\alpha + \beta))$$

Output size: $N = \deg(F)\deg(G)$

Linear algebra: χ_{xy}, χ_{x+y} in $\mathbb{K}[x,y]/(F(x), G(y))$ $O(\mathsf{MM}(N))$

Resultants: $\mathrm{Res}_y \left(F(y), y^{\deg(G)} G(x/y)\right)$, $\mathrm{Res}_y \left(F(y), G(x-y)\right)$ $O(N^{1.5})$

Better: $\otimes$ and $\oplus$ are easy in Newton representation $O(\mathsf{M}(N))$

$$\sum_{\alpha,\beta} (\alpha\beta)^s = \sum_\alpha \alpha^s \cdot \sum_\beta \beta^s \qquad \text{and}$$

$$\sum_{s \geq 0} \frac{\sum_{\alpha,\beta} (\alpha+\beta)^s}{s!} x^s = \left(\sum_{s \geq 0} \frac{\sum_\alpha \alpha^s}{s!} x^s \right) \left(\sum_{s \geq 0} \frac{\sum_\beta \beta^s}{s!} x^s \right)$$

Corollary: Fast polynomial shift $P(x+a) = P(x) \oplus (x+a)$ $O(\mathsf{M}(\deg(P)))$

A first exercise for next Wednesday

(2) Assume that $F \in \mathbb{K}[[x]]$ with $F(0) = 1$.

(a) What is the complexity of computing $\sqrt{F}$, by using $\sqrt{F} = \exp(\frac{1}{2} \log F)$?

(b) Describe a Newton iteration that directly computes $\sqrt{F}$, without appealing to successive logarithm and exponential computations.

(c) Estimate the complexity of the algorithm in (b).

Bonus

Newton iteration – main theorem

1. ("Implicit function theorem") Let $\varphi \in \mathbb{K}[[x,y]]$ s.t. $\varphi(0,0)=0$ and $\varphi_y(0,0) \neq 0$. There exists a unique solution $S \in x\mathbb{K}[[x]]$ to $\varphi(x,S)=0$.
2. ("Newton iteration") Define $Y_\kappa = S \bmod x^{2^\kappa}$. Then,

$$Y_0 = 0 \quad \text{and} \quad Y_{\kappa+1} = Y_\kappa - \frac{\varphi(x,Y_\kappa)}{\varphi_y(x,Y_\kappa)} \mod x^{2^{\kappa+1}} \qquad \text{for } \kappa \geq 0.$$

Proof of (1). Let $\varphi(x,y) = \sum_{j\geq 0} f_j y^j$ with $f_j = \sum_{i \geq 0} f_{j,i} x^i$.
Then $\varphi(x,S)=0$, with $S = \sum_{\ell \geq 1} s_\ell x^\ell$, is equivalent to

$$f_{0,0} = 0,\ f_{1,0}s_1 + f_{0,1} = 0,\ f_{1,0}s_\kappa + \mathrm{Pol}_\kappa(s_1,\dots,s_{\kappa-1}, f_{j,i}, i+j \leq \kappa) = 0$$

Since $f_{0,0} = \varphi(0,0) = 0$ and $f_{1,0} = \varphi_y(0,0) \neq 0$, system has a unique solution.

Newton iteration – main theorem

1. ("Implicit function theorem") Let $\varphi \in \mathbb{K}[[x, y]]$ s.t. $\varphi(0,0) = 0$ and $\varphi_y(0,0) \neq 0$. There exists a unique solution $S \in x\mathbb{K}[[x]]$ to $\varphi(x, S) = 0$.
2. ("Newton iteration") Define $Y_\kappa = S \bmod x^{2^\kappa}$. Then,

$$Y_0 = 0 \quad \text{and} \quad Y_{\kappa+1} = Y_\kappa - \frac{\varphi(x, Y_\kappa)}{\varphi_y(x, Y_\kappa)} \mod x^{2^{\kappa+1}} \qquad \text{for } \kappa \geq 0.$$

Proof of (2). $Y_0 = S \bmod x$, hence $Y_0 = S(0) = 0$. By Taylor's formula,

$$0 = \varphi(x, S) = \varphi(x, Y_\kappa + (S - Y_\kappa)) = \varphi(x, Y_\kappa) + \varphi_y(x, Y_\kappa) \cdot (S - Y_\kappa) + O((S - Y_\kappa)^2).$$

Now, $\varphi_y(x, Y_\kappa) \bmod x = \varphi_y(0,0) \neq 0$, hence $\varphi_y(x, Y_\kappa)$ invertible. Thus,

$$0 = \frac{\varphi(x, Y_\kappa)}{\varphi_y(x, Y_\kappa)} + S - Y_\kappa + O(x^{2^{\kappa+1}}) \Longrightarrow Y_\kappa - \frac{\varphi(x, Y_\kappa)}{\varphi_y(x, Y_\kappa)} \bmod x^{2^{\kappa+1}} = S \bmod x^{2^{\kappa+1}} = Y_{\kappa+1}.$$

Examples: reciprocal and exponential, again

▷ Using $\varphi(x,y) = (F(0)^{-1}+y)^{-1} - F(x)$ to invert $F \in \mathbb{K}[[x]]$, will find

$$S = F(x)^{-1} - F(0)^{-1}$$

after using the Newton operator $\mathcal{N} : G \mapsto 2(G + \frac{1}{F(0)}) - F(G + \frac{1}{F(0)})^2 - \frac{1}{F(0)}$.

$\Longrightarrow$ this is equivalent to $\mathcal{N} : G \mapsto 2G - FG^2$ with initial value $G = F(0)^{-1}$

▷ Using $\varphi(x,y) = F(x) - \log(1+y)$, to compute exp of $F \in x\mathbb{K}[[x]]$, will find

$$S = \exp(F) - 1$$

after using the Newton operator $\mathcal{N} : G \mapsto G + (1+G)(F - \log(1+G))$.

$\Longrightarrow$ this is equivalent to $\mathcal{N} : G \mapsto G + G(F - \log G)$ with initial value $G = 1$

Fast Evaluation and Interpolation

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

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

Interpolation Given $v_0, \dots, v_{n-1} \in \mathbb{A}$, 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

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

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 $a+bx+cx^2+dx^3 \dots x^{n-1}=y$, 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 $a+bx+cx^2+dx^3+ex^4 \dots x^{n-1}=y=$

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

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

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

[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.

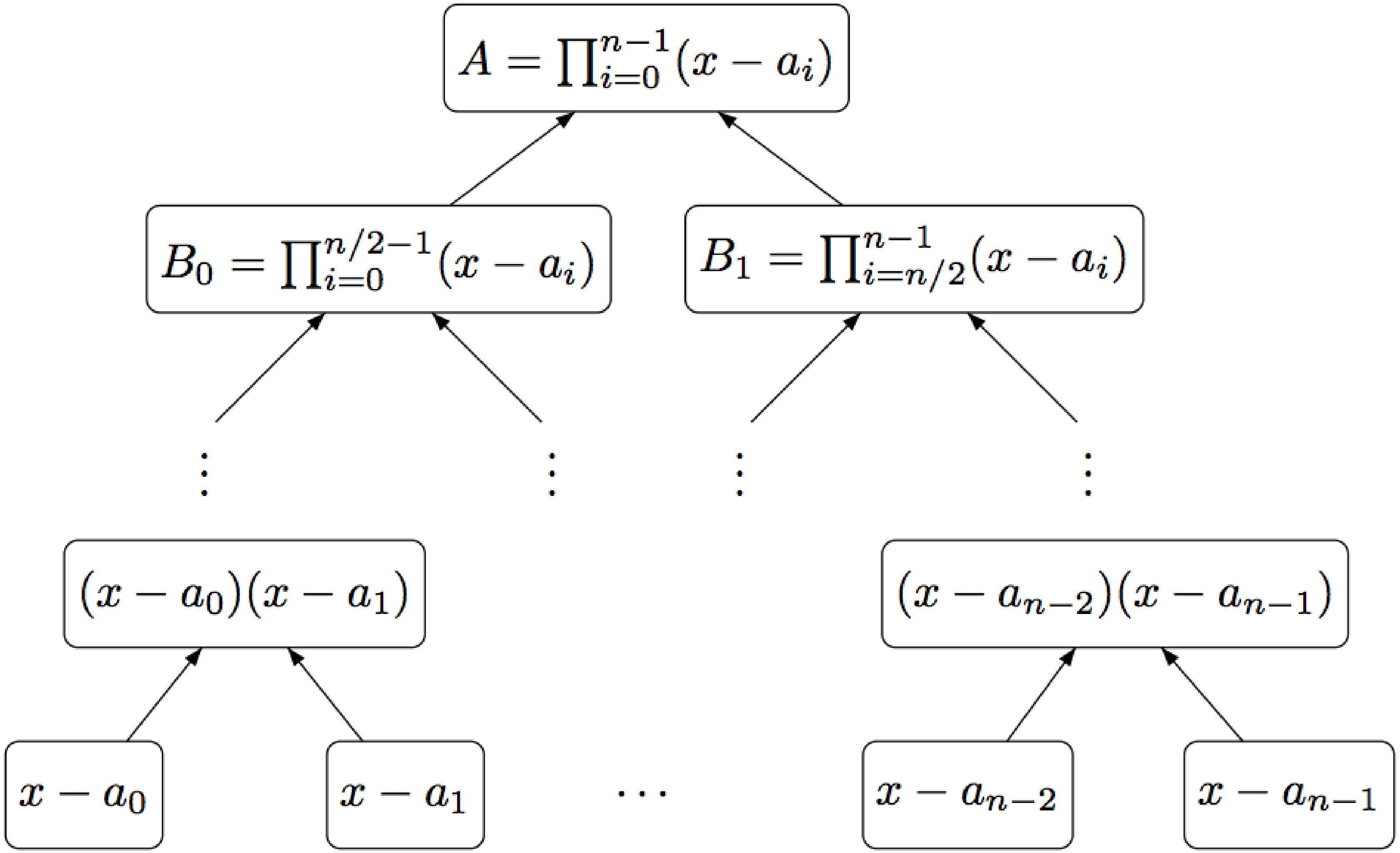
[Lagrange, 1795 – "Sur l'usage des courbes dans la solution des problèmes"]

Evaluation-interpolation, general case

Subproduct tree

[Horowitz, 1972]

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



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

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

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

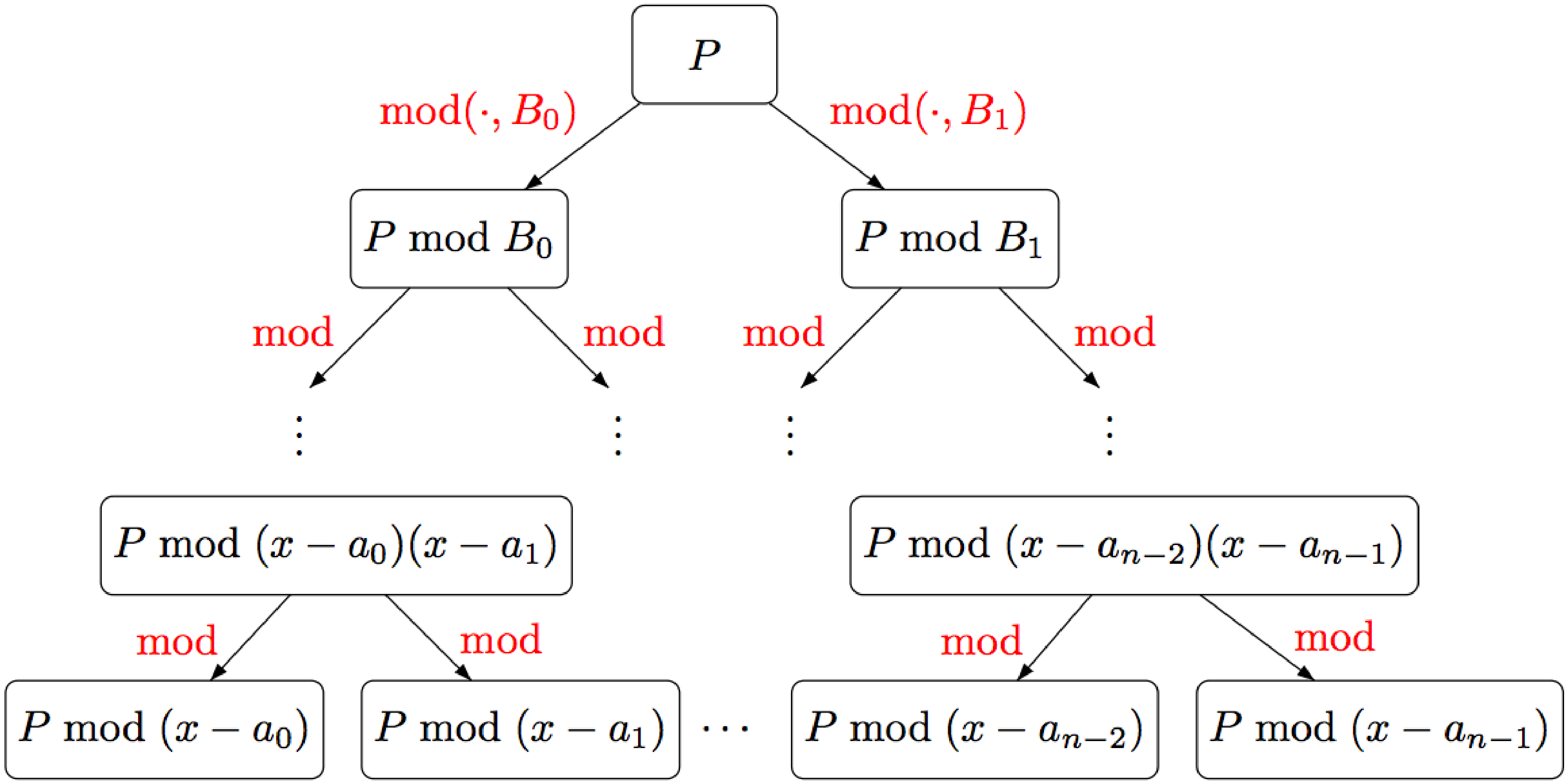
- $P_0 := P \mod \underbrace{(x - a_0) \cdots (x - a_{n/2-1})}_{B_0}$
- $P_1 := P \mod \underbrace{(x - a_{n/2}) \cdots (x - a_{n-1})}_{B_1}$

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

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

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"

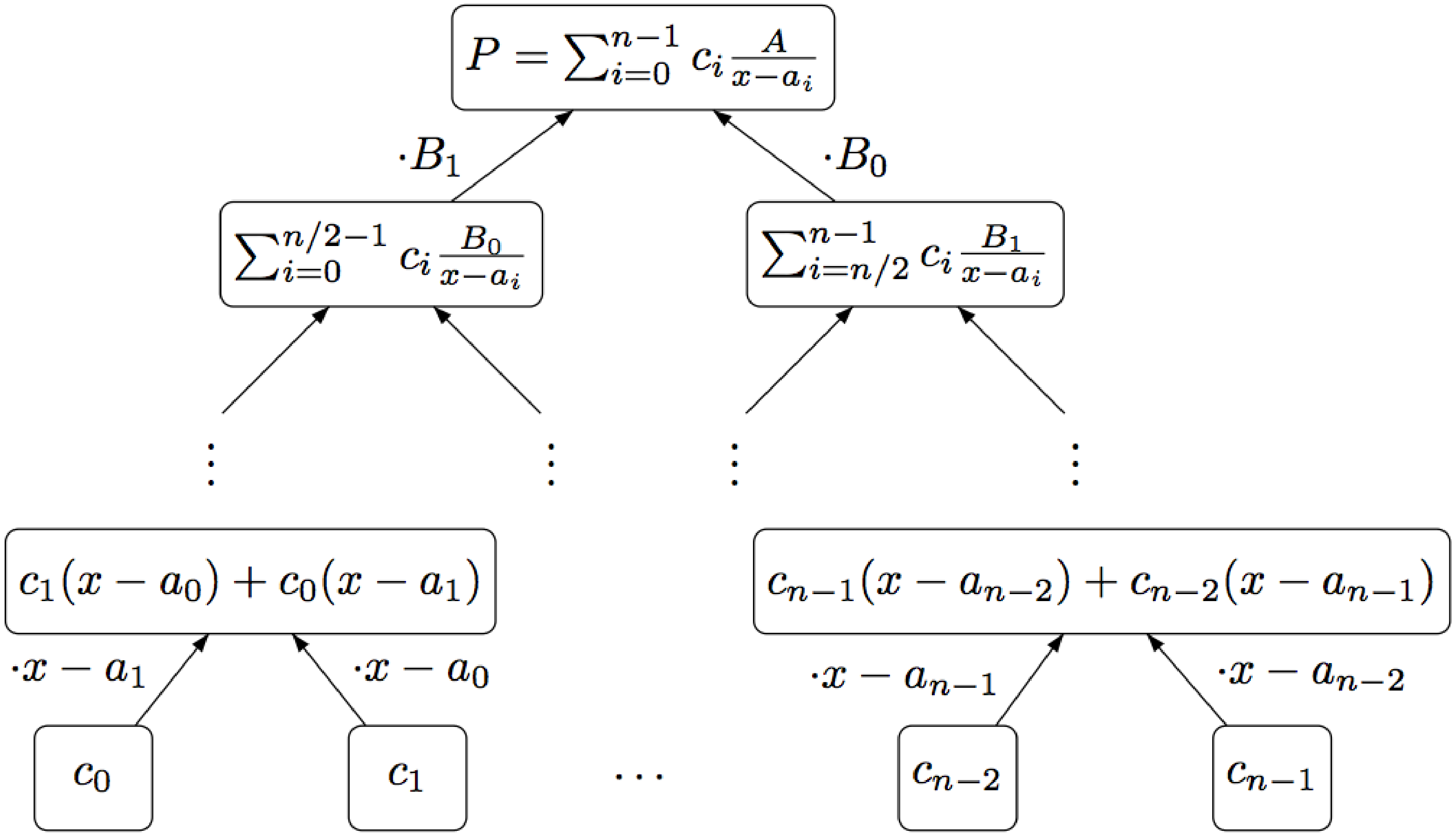
$$P(x) = A(x) \cdot \sum_{i=0}^{n-1} \frac{v_i / A'(a_i)}{x - a_i}$$

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



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 $B_1 = \displaystyle\prod_{i=n/2}^{n-1}(x - q^i)$ from $B_0 = \displaystyle\prod_{i=0}^{n/2-1}(x - q^i)$, 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 $B_0(x)$ $\quad O(n)$
- Return $A(x) = B_0(x)B_1(x)$ $\quad \mathsf{M}(n/2)$

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

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: $$P(q^i) = \sum_{j=0}^{n-1} c_j q^{ij}, \qquad 0 \leq i < n$$

Bluestein's trick: $$ij = \binom{i+j}{2} - \binom{i}{2} - \binom{j}{2} \implies q^{ij} = q^{\binom{i+j}{2}} \cdot q^{-\binom{i}{2}} \cdot q^{-\binom{j}{2}}$$

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

$$\text{coeff. of } x^{n-1+i} \text{ in } \left(\sum_{j=0}^{n-1} c_j q^{-\binom{j}{2}} x^{n-j-1}\right)\left(\sum_{\ell=0}^{2n-2} q^{\binom{\ell}{2}} x^{\ell}\right)$$

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

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

Subproblem: Given $c_0, \ldots, c_{n-1} \in \mathbb{K}$, 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 $\{1, q^{-1}, \ldots, q^{-(n-1)}\}$:

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

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 $\mathcal{G} = \{1, q, q^2, \ldots, q^{2d-2}\}$

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

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

A second exercise for next Wednesday

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]