

# Efficient Algorithms in Computer Algebra (2023–2024)



<https://wikimpri.dptinfo.ens-cachan.fr/doku.php?id=cours:c-2-22>

**Alin Bostan**

alin.bostan@inria.fr

**Pierre Lairez**

pierre.lairez@inria.fr

**Marc Mezzarobba**

marc@mezzarobba.net

**Vincent Neiger**

vincent.neiger@lip6.fr

September 21, 2023

Introduction — Computer Algebra and Complexity — Fast Multiplication

# Part I

## Introduction

## Practical questions

- language: English or French?
- course is “breakable”
  - following the second part only is not reasonable
  - following the first part only is feasible (but not recommended)

## Professors

- Alin Bostan ( $\sim 12h$ ), Inria Saclay  
webpage: <https://mathexp.eu/bostan/>  
email: [alin.bostan@inria.fr](mailto:alin.bostan@inria.fr)
- Pierre Lairez ( $\sim 12h$ ), Inria Saclay  
webpage: <https://pierre.lairez.fr/>  
email: [pierre.lairez@inria.fr](mailto:pierre.lairez@inria.fr)
- Marc Mezzarobba ( $\sim 12h$ ), CNRS  
webpage: <https://marc.mezzarobba.net/>  
email: [marc@mezzarobba.net](mailto:marc@mezzarobba.net)
- Vincent Neiger ( $\sim 12h$ ), Sorbonne Université  
webpage: <https://vincent.neiger.science/>  
email: [vincent.neiger@lip6.fr](mailto:vincent.neiger@lip6.fr)

## Material

- webpage for the course, with info and material (frequent updates):  
<https://wikimpri.dptinfo.ens-cachan.fr/doku.php?id=cours:c-2-22>
- all slides in English
- book in French, printed 2017 version is cheap ( $\approx 15\text{€}$ )
- updated pdf, legally cost-free, is available here:  
<https://hal.archives-ouvertes.fr/AECF/>

## Calendar

- always refer to the webpage! ask us by email in case of doubts
- **time:** Thursdays, 16:15-19:15; **location:** room 1004
- first period: from today to 16/11 **except 05/10**, exam on 23/11
- second period: from 07/12 to 08/02, exam on 29/02  
note: 28/12 & 04/01  $\rightsquigarrow$  holidays

# Organisation

## How to work?

- with pen and paper + with a computer (SageMath, Maple, ...)
- in class: pay close attention, be proactive, ask questions
- at home: weekly regular work  $\gg$  intense sprint 4 days before exam

## To help you towards this:

- basic questions/exercises/examples/demos during each class  
take advantage of them!
- exercises proposed at the end of each session
  - it is in your interest to study them during the week
  - beginning of next session: one of you volunteers to correct it
- 3-hour tutored exercise session on 16/11  
practice on exam-like exercises, with at least one professor available for you

# Master's Internships?

## Interested? Let's discuss soon!

- we provide research internship subjects in our respective teams  
MathExp, Inria Saclay & MAX, Polytechnique & PolSys, LIP6 / Sorbonne U.
- we can also provide advice for other internships/PhD opportunities with colleagues in computer algebra-related domains  
in Bordeaux, Grenoble, Lille, Limoges, Lyon, Montpellier, Nancy, Rennes, Toulouse, ...
- wide range of topics, with common denominators:
  - involve a variable, but nonnegligible, amount of mathematics/algebra
  - questions of effectiveness/efficiency of computations

# Master's Internships?

## Possible research subjects:

- Algorithms for factoring differential equations (Bostan)  
with T. Rivoal, CNRS @ Grenoble <https://specfun.inria.fr/bostan/mpri/grigoriev.pdf>
- Algorithms for solving q-difference equations (Bostan)  
with L. Di Vizio, CNRS @ Versailles (for details: Bostan is teaching next course on 28/09)
- several subjects + PhD funding, MAX team @ LIX/Polytechnique (Mezzarobba)
  - . Reliable numerical integration
  - . Fast evaluation of elementary functions with medium precision
  - . Sparse interpolation of rational functions
  - . Numerical approach to structural parameter identifiability
- several subjects + PhD funding, PolSys team @ LIP6/Sorbonne (Neiger)
  - . Computing contiguity/multiplication matrices for statistical physics
  - . Algebraic cryptanalysis of new NIST multivariate signature schemes
  - . Calcul formel, bases de Gröbner, résolution de systèmes structurés à variables réelles
  - . Méthodes algébriques pour le calcul de topologies d'ensembles semi-algébriques

contact us asap if interested

# Master's Internships?

## ISSAC 2023 conference topics: Algorithmic aspects

- Exact and symbolic linear, polynomial and differential algebra
- Symbolic-numeric, homotopy, perturbation and series methods
- Computational algebraic geometry, group theory and number theory, quantifier elimination and logic
- Computer arithmetic
- Summation, recurrence equations, integration, solution of ODEs & PDEs
- Symbolic methods in other areas of pure and applied mathematics
- Complexity of algebraic algorithms and algebraic complexity

## ISSAC 2023 conference topics: Software aspects

- Design of symbolic computation packages and systems
- Language design and type systems for symbolic computation
- Data representation
- Considerations for modern hardware
- Algorithm implementation and performance tuning
- Mathematical user interfaces
- Use with systems for, e.g., digital libraries, courseware, simulation and optimization, automated theorem-proving, computer-aided design, and automatic differentiation.



# Contents

This year:

- power series, polynomials, matrices
- linear recurrences and linear differential equations
- polynomial matrices and Hermite-Padé approximation
- factorization of polynomials, lattice reduction
- binomial sums
- opening to combinatorics

Related topics / courses:

- techniques in cryptography and cryptanalysis → C-2-12-1
- arithmetic algorithms for cryptology → C-2-12-2
- polynomial systems, computer algebra, and applications → C-2-13-1
- error correcting codes and applications to cryptography → C-2-13-2

# Our viewpoint

Computer Algebra = design of fast algebraic algorithms on exact representations of mathematical objects in the computer.

This is a part of “doing mathematics” on the computer.

# Motivating Examples

A computational proof of

Theorem (Ramanujan)

$$\sqrt[3]{\cos \frac{2\pi}{7}} + \sqrt[3]{\cos \frac{4\pi}{7}} + \sqrt[3]{\cos \frac{8\pi}{7}} = \sqrt[3]{\frac{5 - 3\sqrt[3]{7}}{2}}$$

is by combining resultants for elimination.

# Motivating Examples

A computational proof of:

Theorem (Apéry, 1978)

$$\zeta(3) := 1 + \frac{1}{2^3} + \frac{1}{3^3} + \frac{1}{4^3} + \dots \notin \mathbb{Q}.$$

“ $\zeta(3)$  is irrational.”

Proof: relies crucially on proving that both

$$a_n := \sum_{k=0}^n \binom{n}{k}^2 \binom{n+k}{k}^2 \left( \sum_{m=1}^n \frac{1}{m^3} + \sum_{m=1}^k \frac{(-1)^{m-1}}{2m^3 \binom{n}{m} \binom{n+m}{m}} \right)$$

$$\text{and} \quad b_n := \sum_{k=0}^n \binom{n}{k}^2 \binom{n+k}{k}^2$$

satisfy

$$n^3 a_n - (2n-1)(17n^2 - 17n + 5)a_{n-1} + (n-1)^3 a_{n-2} = 0 \quad (n \geq 2).$$

*Recurrence can be discovered and verified in a few seconds!*

# On the computer

## List of Computer Algebra Systems (subjective selection)

Axiom, CoCoA, Derive, GAP, Macaulay2, Magma, Maple, Mathemagix, Mathematica, Maxima, MuPAD, PARI/GP, Reduce, SageMath, Singular

- more recently: development of optimized open-source libraries
- SageMath gathers many such state-of-the-art libraries

## what you can compute in about 1 second with fflas-ffpack with NTL

> PLUQ	$m = 3800$	1.00s	> PolMul	$d = 7 \times 10^6$	1.03s
> LinSys	$m = 3800$	1.00s	> Division	$d = 4 \times 10^6$	0.96s
> MatMul	$m = 3000$	0.97s	> XGCD	$d = 2 \times 10^5$	0.99s
> Inverse	$m = 2800$	1.01s	> MinPoly	$d = 2 \times 10^5$	1.10s
> CharPoly	$m = 2000$	1.09s	> MPEval	$d = 1 \times 10^4$	1.01s

## Part II

# Computer algebra and computability/effectiveness

# A negative result

## The Richardson–Matiyasevich Theorem

In the class of expressions obtained from a variable  $X$  and the constant 1 by application of the ring operations  $+$ ,  $-$ ,  $\times$  and composition with the function  $\sin(\cdot)$  and the absolute value function  $|\cdot|$ , the test of equivalence to 0 is undecidable.

- equality test is a zero test (as soon as subtraction exists)
- no “good” `simplify()`; it is made of heuristics
- computer algebra: work with algebraic constructs that preserve the decidability of the zero test

# Effective algebraic structures

## Definition

An algebraic structure (group, ring, field, vector space, ...) is *effective* if it is endowed with:

- a data structure to represent its elements;
  - algorithms to perform its inner operations and to test equality and other predicates.
- 
- e.g. an effective ring comes with algorithms for: equality, addition, subtraction, multiplication
  - composition of data structures via lists/arrays



# Integers and variants

## Machine integers / word-size integers

- processor “integers” = integers modulo  $2^w$  (usually  $w = 32$  or  $w = 64$ )
- operations  $=, +, -, \times$  in hardware

## Big integers, a.k.a. *bignums*

- unique writing  $N = (-1)^{\varepsilon} \times (a_0 + a_1B + \dots + a_kB^k)$  for a fixed base  $B$
- Lemma: the ring  $\mathbb{Z}$  of relative integers is effective.

## Modular integers

- core tool: Euclidean division with remainder in  $\mathbb{Z}$
- Lemma: for any integer  $n \geq 2$ , the ring  $\mathbb{Z}/n\mathbb{Z}$  is effective

- ~> avoid intermediate expression swell
- ~> reconstructions by the Chinese Remainder Theorem (CRT)
- ~> allows probabilistic heuristics by calculations modulo  $n$
- ~> algorithms with deterministic outputs by controlling sizes and bad  $n$

# Vectors and matrices

Vector: typically an array of pointers to the coefficients  
(or simply, a memory-contiguous array if a coefficient fits into a machine word)

## Proposition

If  $\mathbb{K}$  is an effective field,

- the vector space  $\mathbb{K}^n$  is effective,
- the ring  $\mathcal{M}_n(\mathbb{K}) = \mathbb{K}^{n \times n}$  is effective.

Depending on the application, dense or sparse representation  
→ different algorithms!

# Polynomials, fractions

Depending on the application, dense or sparse representation  
→ different algorithms!

## Proposition

If  $\mathbb{A}$  is an effective ring, then so is  $\mathbb{A}[X]$ .

- multivariate polynomials by iteration  
(not necessarily done this way in practice)
- strong connection between univariate polynomials and big integers

## Proposition

If  $\mathbb{A}$  is an effective domain, then its fraction field is effective.

- provides  $\mathbb{Q}$  and  $\mathbb{K}(X)$
- representation variants:  $\text{Frac}(\mathbb{Z}[X, Y])$ ,  $\text{Frac}(\text{Frac}(\text{Frac}(\mathbb{Z})[X])[Y])$ , ...

# Truncated formal power series

- truncated series  $a_0 + a_1X + \dots + a_{N-1}X^{N-1} + O(X^N)$   
 $\rightsquigarrow$  represented as a polynomial of degree  $< N$
- optimized algorithm for the *short product*  
[Schönhage: “Never waste a factor of 2!”]

## Proposition

If  $\mathbb{A}$  is an effective ring and if  $N \in \mathbb{N}$ , then  $\mathbb{A}[X]/(X^N)$  is an effective ring.

- computing approximations
- representing (exactly) rational fractions if numerators and denominators are with bounded degrees
- reconstructing linear differential equations with polynomial coefficients (*guessing*); analogue for linear recurrence equations
- reconstructing bivariate polynomials from univariate series solutions: for factorization and for solving polynomial systems

# Equations as the right data structures

Non-explicit or infinite mathematical objects can be represented exactly on the computer when they are solutions to finite equations:

- $\sqrt{2}$  = just a symbol whose square is 2,
- $\ln x$  = just a symbol whose derivative is  $1/x$ .

—→ algorithms on **implicit representations**

# Algebraic numbers = univariate polynomials


## Proposition

If  $\mathbb{K}$  is an effective field, then so is its algebraic closure  $\bar{\mathbb{K}}$ .

Calculations by resultants, series, gcd.

## Consequence: “easy” computational proof of

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

 SageMath example `</>`

# Polynomial systems

## Proposition

Let  $\mathbb{K}$  be an effective field, and  $f_1, \dots, f_m$  be polynomials from the ring  $\mathbb{K}[X_1, \dots, X_n]$ . Then the quotient ring  $\mathbb{K}[X_1, \dots, X_n]/(f_1, \dots, f_m)$  is effective.

- algorithms by resultants or by Gröbner bases ( $\rightsquigarrow$  course 2-13-1)
- very strong connection to geometry

# Linear differential equations, linear recurrence equations

If  $\mathbb{K}$  is an effective field, the set...

$\left\{ \text{formal power series } \sum_{n \in \mathbb{N}} a_n X^n \in \mathbb{K}[[X]] \text{ that are solutions to} \right.$   
linear differential equations with coefficients from  $\mathbb{K}[X]$   
 $\left. \right\}$

is an effective ring.

If  $\mathbb{K}$  is an effective field, the set...

$\left\{ \text{sequences } (a_n)_{n \in \mathbb{N}} \in \mathbb{K}^{\mathbb{N}} \text{ that are solutions to} \right.$   
linear recurrences with coefficients in  $\mathbb{K}[n]$   
 $\left. \right\}$

is an effective ring.

- special functions in mathematical physics; combinatorial sequences
- algorithms by a non-commutative variant of resultants
- equality test reduces to the identification of initial conditions



# Systems of linear differential and recurrence equations

Algorithms (of *creative telescoping*) make possible the automatic proof of identities like:

$$\sum_{k=0}^n \left( \sum_{j=0}^k \binom{n}{j} \right)^3 = n2^{3n-1} + 2^{3n} - 3n2^{n-2} \binom{2n}{n},$$
$$\int_0^{+\infty} x J_1(ax) I_1(ax) Y_0(x) K_0(x) dx = -\frac{\ln(1-a^4)}{2\pi a^2}.$$

(binomial coefficients, Bessel functions, etc)

## Part III

# Computer algebra and complexity/efficiency

# Measures of (time) complexity

## Random access machines (RAM)

- two tapes + arbitrarily many registers, all containing integers
- read, write, addition, subtraction, product, division, jumps

## Arithmetic complexity

- counts operations on some effective algebraic structure  $\mathbb{A}$  (arithmetic operations and tests of predicates)
- does not count copies, operations on loop counters, indirections
- okay if operations in  $\mathbb{A}$  are preponderant and on similar sizes
- caution: memory is neglected, e.g. matrix transposition is free, etc.

## Bit complexity

- counts operations on digits of integers written in  $B$
- better if intermediate calculations are with integers of variable sizes

# Notation $O(\cdot)$ and variations

Various “size” parameters of data structures: number of digits, degrees, matrix dimensions, etc.

## Comparaisons of complexities as functions of “sizes”

- $f(n) = O(g(n))$  as  $n \rightarrow \infty$  if  $\exists K > 0, \exists N > 0, \forall n > N, |f(n)| \leq K |g(n)|$
- be cautious if there are several parameters
- simplification to hide logarithms:  $f(n) = \tilde{O}(g(n))$  as  $n \rightarrow \infty$  if  $\exists k \geq 0, f(n) = O(g(n) \log^k |g(n)|)$
- $f(n) = \Theta(g(n))$  as  $n \rightarrow \infty$  if  $\exists K > 0, \exists K' > 0, \exists N > 0, \forall n > N, K' |g(n)| \leq |f(n)| \leq K |g(n)|$
- remark:  $f(n) = \Theta(g(n))$  iff  $f(n) = O(g(n))$  and  $g(n) = O(f(n))$

Caution: a few chapters in the book write  $f(X) := g(X) + O(X^N)$  to mean “compute and store the polynomial remainder  $f(X) := \text{rem}(g(X), X^N)$ ”. This is a ternary notation “ $? := ? + O(X^?)$ ”, not to be confused with “ $? = ? + O(X^?)$ ”.

# What efficiency?

- generally speaking: worst-case time complexity
- often, this reflects the average-case complexity  
at least for operations concerning fundamental algebraic structures

An algorithm is **quasi-optimal**...

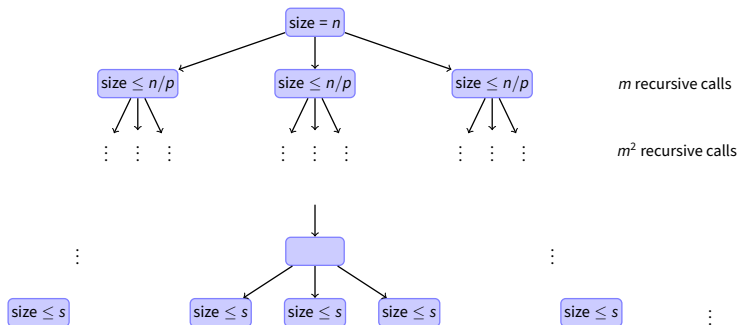
if its complexity is some  $\tilde{O}$  of the sum of the sizes of its input and output

(We will slightly extend this definition for linear algebra.)

**ultimate goal**

design quasi-optimal algorithms

# The programming paradigm “divide and conquer” (DAC)



$m$  recursive calls, on  $p$  times smaller data, down to threshold  $s \geq p$

When is this of interest? Where are the most costly manipulations?

# Complexity analysis of “divide and conquer”

## Theorem

Let  $T$  be an increasing function and let  $C$  be a function ruled by the inequality

$$C(n) \leq \begin{cases} T(n) + mC(\lceil n/p \rceil), & \text{if } n \geq s \geq p \\ \kappa & \text{otherwise,} \end{cases}$$

with  $m > 0$  and  $\kappa > 0$ , and so that there exist  $q$  and  $r$  with  $1 < q \leq r$  satisfying

$$qT(n) \leq T(pn) \leq rT(n), \quad \text{for all sufficiently large } n.$$

Then, when  $n \rightarrow \infty$ ,

*[dominant cost:]*

$$C(n) = \begin{cases} O(T(n)), & \text{if } q > m, & \text{[top of tree]} \\ O(T(n) \log_p n), & \text{if } q = m, & \text{[all levels]} \\ O(n^{\log_p(m/p)} T(n)) & \text{if } q < m. & \text{[bottom of tree]} \end{cases}$$

## Part IV

# Fast polynomial multiplication



# A long history (and a spoiler)

## Theorem (for the dense representation and coefficients in a ring $\mathbb{A}$ )

The multiplication of polynomials of degree at most  $n$  in  $\mathbb{A}[X]$  requires:

- $O(n^2)$  operations in  $\mathbb{A}$  by the naive algorithm (dates back to Antiquity);
- $O(n^{\log_2 3})$  ops in  $\mathbb{A}$  by the algorithm by Karatsuba (and Ofman) (1963);
- $O(n \log n)$  operations in  $\mathbb{A}$  when  $\mathbb{A}$  contains enough “good” roots of unity via Fast Fourier Transform (FFT), known to Gauss (1805), rediscovered by Cooley and Tukey (1965)
- $O(n \log n \log \log n)$  operations in  $\mathbb{A}$  by the algorithm by Schönhage and Strassen (1971) generalizing the FFT applicability by introducing “virtual” roots of unity
- $O(n \log n \log \log n)$  operations in  $\mathbb{A}$  by Cantor and Kaltofen (1991), for general  $\mathbb{A}$  arbitrary (possibly non-commutative) algebra,  $O(n \log n)$  mul. and  $O(n \log n \log \log n)$  add./sub.

recently, after the breakthrough of Fürer’s algorithm (2007) which multiplies integers of size  $n$  in  $O(n \log n K^{\log^* n})$  bit operations (constant  $K$ ):

- *bit complexity*  $O(n \log(p) \log(n \log(p)) 8^{\log^*(n \log(p))})$  by Harvey, van der Hoeven, Lecerf (JACM 2017), for polynomials over  $\mathbb{F}_p$  with  $p$  prime  
 $\log^* n = \text{minimal number } k \text{ such that } \log^{\circ k} n \leq 1$
- under a number-theoretic conjecture: for polynomials over  $\mathbb{F}_q$ , *bit complexity*  $O(n \log(q) \log(n \log(q)))$  (Harvey and van der Hoeven, JACM 2022).

# Naive multiplication algorithm

$$F = f_0 + \dots + f_n X^n, \quad G = g_0 + \dots + g_n X^n \quad \longrightarrow \quad H := FG = h_0 + \dots + h_{2n} X^{2n}$$

$$h_i = \sum_{j=0}^i f_j g_{i-j}, \quad h_{2n-i} = \sum_{j=0}^i f_{n-j} g_{n-i+j}, \quad (0 \leq i \leq n)$$

$$\sum_{i=0}^n \left( \begin{array}{c} i+1 \text{ multiplications} \\ i \text{ additions} \end{array} \right) + \sum_{i=0}^{n-1} \left( \begin{array}{c} i+1 \text{ multiplications} \\ i \text{ additions} \end{array} \right) = \left( \begin{array}{c} (n+1)^2 \text{ multiplications} \\ n^2 \text{ additions} \end{array} \right)$$

# Karatsuba's multiplication: the idea in degree 1

$$F = f_0 + f_1X, G = g_0 + g_1X \longrightarrow H := FG = h_0 + h_1X + h_2X^2$$

Naively, 4 multiplications:  $h_0 = f_0g_0$ ,  $h_1 = f_0g_1 + f_1g_0$ ,  $h_2 = f_1g_1$

## Some easy evaluations

(up to some interpretation at infinity)

$$h_0 = H(0) = F(0)G(0) = f_0g_0$$

$$h_0 + h_1 + h_2 = H(1) = F(1)G(1) = (f_0 + f_1)(g_0 + g_1)$$

$$h_2 = H(\infty) = F(\infty)G(\infty) = f_1g_1$$

## Gain of one multiplication

$$h_1 = (f_0 + f_1)(g_0 + g_1) - f_0g_0 - f_1g_1$$

# Karatsuba's algorithm

**Input**  $F, G$  of degrees at most  $n - 1$ .

**Output**  $H = FG$ .

- 1 If  $n = 1$ , return  $FG$ .
- 2 Set  $k = \lceil n/2 \rceil$  and decompose  $F$  et  $G$  according to the equation

$$F = F^{(0)} + F^{(1)}X^k, \quad G = G^{(0)} + G^{(1)}X^k,$$

- 3 Recursively compute  $H_0 = F^{(0)}G^{(0)}$  and  $H_2 = F^{(1)}G^{(1)}$ .
- 4 Compute  $A = F^{(0)} + F^{(1)}$  et  $B = G^{(0)} + G^{(1)}$ .
- 5 Recursively compute  $C = AB$ .
- 6 Compute  $H_1 = C - H_0 - H_2$ .
- 7 Return  $H_0 + H_1X^k + H_2X^{2k}$ .

Remark:  $A, B$  have degree  $< k$ ;  $H_0, H_1, H_2, C$  have degree  $< n$ .

# Complexity analysis of Karatsuba's algorithm

## Theorem

If  $n$  is a power of 2, Karatsuba's algorithm computes the product of two polynomials of degrees at most  $n - 1$  in at most  $9n^{\log_2 3}$  operations in  $\mathbb{A}$ .

Proof: For  $n = 2^\ell$ , that is to say  $\ell = \log_2 n$ :

$$\begin{aligned}K(n) &\leq 3K(n/2) + 4n \\&\leq 3^2K(n/2^2) + 4n(3/2 + 1) \\&\leq \dots \\&\leq 3^\ell K(n/2^\ell) + 4n((3/2)^{\ell-1} + \dots + 3/2 + 1) \\&\leq 3^\ell K(1) + 4n \frac{(3/2)^\ell - 1}{3/2 - 1} \\&\leq 3^\ell (1 + 4 \cdot 2) = 9n^{\log_2 3}.\end{aligned}$$

# Algorithms by Cook (1963), Toom (1966), Schönhage (1966), Knuth (1969)

## Theorem

For a given  $\epsilon > 0$ , by increasing the number of evaluation points one obtains an algorithm of complexity  $O(n^{1+\epsilon})$ .

# Multiplication by DFT (*Discrete Fourier Transform*): the idea

Relies on a suitable choice of points for evaluation/interpolation

$$\begin{aligned}\text{DFT} : \mathbb{A}[X]_{<n} &\xrightarrow{\sim} \mathbb{A}^n \\ P &\longmapsto (P(\omega^0), \dots, P(\omega^{n-1}))\end{aligned}$$

**Input**  $F$  and  $G$  two polynomials,  $n$  an integer, and  $\omega$  a principal  $n$ th root of unity (*definition to come*).

**Output**  $\text{rem}(FG, X^n - 1)$ .

- ❶ *Precomputation.* Compute the powers  $\omega^2, \dots, \omega^{n-1}$ .
- ❷ *Evaluation.* Compute  $(u_i)_{i=0}^{n-1} := \text{DFT}(F)$  et  $(v_i)_{i=0}^{n-1} := \text{DFT}(G)$ .
- ❸ *Coordinate-wise product.* Compute  $W := (u_i v_i)_{i=0}^{n-1}$ .
- ❹ *Interpolation.* Compute and return  $\text{DFT}^{-1}(W)$ .

There also remains to guarantee a fast computation of DFT and  $\text{DFT}^{-1}$ .

# Roots of unity of a general ring $\mathbb{A}$

## Finer and finer definitions

- $\omega$  is an  $n$ th root of unity if  $\omega^n = 1$
- $\omega$  is a primitive  $n$ th root of unity if  $\omega^n = 1$  and if  $0 < t < n \Rightarrow \omega^t - 1$  non-zero
- $\omega$  is a principal  $n$ th root of unity if  $\omega^n = 1$  and if  $0 < t < n \Rightarrow \omega^t - 1$  non-zero and non-zero-divisor

## Properties

- $\omega$  is a primitive  $n$ th root of 1  $\Rightarrow \omega^{-1}$  is a primitive  $n$ th root of 1
- $n = pq$  and  $\omega$  is a primitive  $n$ th root of 1  $\Rightarrow$   
 $\omega^p$  is a primitive  $q$ th root of 1
- $\omega$  is a primitive  $n$ th root of 1 and  $0 < \ell < n \Rightarrow \sum_{j=0}^{n-1} \omega^{\ell j} = 0$
- three analogous statements for principal roots of unity



# DFT (evaluation) by FFT (*Fast Fourier Transform*)

**Input**  $P = p_0 + \dots + p_{n-1}X^{n-1}$ ; the powers  $1, \omega, \dots, \omega^{n-1}$  of some principal  $n$ th root of unity  $\omega$ ,  $n$  being a power of 2.

**Output**  $(P(\omega^0), \dots, P(\omega^{n-1}))$ .

- ① If  $n = 1$ , return  $p_0$ , otherwise, set  $k = n/2$  and calculate

$$R_0(X) := \sum_{j=0}^{k-1} (p_j + p_{j+k})X^j, \quad \bar{R}_1(X) := R_1(\omega X) = \sum_{j=0}^{k-1} (p_j - p_{j+k})\omega^j X^j.$$

- ② Recursively compute the DFT of  $R_0$  and  $\bar{R}_1$  on the family  $(1, \omega^2, \dots, (\omega^2)^{k-1})$ . (“time decimation”)
- ③ Return  $(R_0(1), \bar{R}_1(1), R_0(\omega^2), \bar{R}_1(\omega^2), \dots, R_0((\omega^2)^{k-1}), \bar{R}_1((\omega^2)^{k-1}))$ .

## Correctness

$n = 2k$  and  $\omega$  is a primitive/principal  $n$ th root of 1  $\Rightarrow \omega^k = -1$

$$P = (X^k - 1)Q_0 + R_0 = (X^k + 1)Q_1 + R_1 \Rightarrow P(\omega^\ell) = \begin{cases} R_0(\omega^\ell) & \text{if } \ell \text{ even,} \\ R_1(\omega^\ell) & \text{if } \ell \text{ odd.} \end{cases}$$

# Complexity analysis of the FFT algorithm

## Theorem

For  $n$  a power of 2, Fast Fourier Transform (FFT) requires  $\simeq \frac{3n}{2} \log n$  operations in  $\mathbb{A}$ . Each multiplication in  $\mathbb{A}$  done by the algorithm is between an element of  $\mathbb{A}$  and some power of  $\omega$ .

Proof: For  $n = 2^\ell$ , that is to say  $\ell = \log_2 n$ :

$$\begin{aligned} F(n) &\leq 2F(n/2) + \frac{3n}{2} \\ &\leq 2^2 F(n/2^2) + \frac{3n}{2} (2/2 + 1) \\ &\leq \dots \\ &\leq 2^\ell F(n/2^\ell) + \frac{3n}{2} (2^{\ell-1}/2^{\ell-1} + \dots + 2/2 + 1) \\ &\leq nF(1) + \frac{3n}{2} \ell = \frac{1 + 3 \log_2 n}{2} n. \end{aligned}$$

# Interpolate is evaluate (!)

Given the Vandermonde matrix  $V_\omega := \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{n-1} \\ \vdots & & & \vdots \\ 1 & \omega^{n-1} & \dots & \omega^{(n-1)^2} \end{pmatrix}$ , we

have:

$$\text{DFT}(P) = (P(\omega^0), \dots, P(\omega^{n-1})) = (p_0, \dots, p_{n-1})V_\omega.$$

## Lemma

If  $\omega \in \mathbb{A}$  is a principal  $n$ th root of unity, then  $V_{\omega^{-1}}V_\omega = nI_n$ .

Proof:

$$\sum_{k=0}^{n-1} \omega^{-(i-1)k} \omega^{k(j-1)} = \sum_{k=0}^{n-1} \omega^{(j-i)k} = n\delta_{ij}.$$

$$\text{Said differently: } (\text{DFT}_\omega)^{-1} = \frac{1}{n} \text{DFT}_{\omega^{-1}}.$$

# Complexity analysis of multiplication by FFT when $\mathbb{A}$ contains roots of unity for all $n = 2^k$

**Input**  $F$  and  $G$  two polynomials,  $n$  an integer, and  $\omega$  a principal  $n$ th root of unity, assumed to exist in  $\mathbb{A}$ .

**Output**  $FG$ , assumed to be of degree  $< n$ , a power of 2.

- 1 *Precomputation.* Calculate the powers  $\omega^2, \dots, \omega^{n-1}$ .
- 2 *Evaluation.* Compute  $(u_i)_{i=0}^{n-1} := \text{DFT}_\omega(F)$  and  $(v_i)_{i=0}^{n-1} := \text{DFT}_\omega(G)$  by FFT.
- 3 *Coordinate-wise product.* Compute  $W := (u_i v_i)_{i=0}^{n-1}$ .
- 4 *Interpolation.* Compute, using FFT, and return  $\frac{1}{n} \text{DFT}_{\omega^{-1}}(W)$ .

## Theorem

If 2 is invertible in  $\mathbb{A}$ , if  $n$  is some power of 2, and if  $\omega$  is a principal  $n$ th root of unity in  $\mathbb{A}$ , the product of two polynomials whose sum of degrees is  $< n$  can be computed in  $\frac{9}{2}n \log n + O(n)$  operations in  $\mathbb{A}$ . Only  $n$  of the products are between two elements of  $\mathbb{A}$  that are general elements (that is, not powers of  $\omega$  or  $1/n$ ).

# “Practical” remark: Fourier primes

## Proposition

The finite field  $\mathbb{F}_q$  with  $q$  elements contains a primitive  $n$ th root of unity if and only if  $n$  divides  $q - 1$

## Good prime numbers

A prime  $p$  is called *FFT prime* if it has the form  $p = 2^e \ell + 1$  for  $e$  “big enough”

$$p := 4179340454199820289 = 29 \times 2^{57} + 1, \mathbb{A} := \mathbb{F}_p, n := 2^{57}, \\ \omega := 21 \text{ is a primitive } n\text{th root of unity}$$

# Sketch of the Schönhage–Strassen algorithm

## “Virtual” roots of unity

If 2 is invertible in  $\mathbb{A}$  and if  $n$  is a power of 2, then  $\omega = X + (X^n + 1)$  is a principal  $(2n)$ th root of 1 in  $\mathbb{A}[X]/(X^n + 1)$  (which is not always a domain, even for a field  $\mathbb{A}$ ).

**Input**  $F$  and  $G$  of degrees  $< n = 2^k$ , for  $k > 2$ .

**Output**  $\text{rem}(FG, X^n + 1)$ .

① Let  $d = 2^{\lfloor k/2 \rfloor}$  and  $\delta = n/d$ . Rewrite  $F$  and  $G$  in the form

$$\bar{F}(X, Y) = F_0(X) + \dots + F_{\delta-1}(X)Y^{\delta-1}, \quad \bar{G}(X, Y) = G_0(X) + \dots + G_{\delta-1}(X)Y^{\delta-1},$$

with  $F_i, G_i$  of degrees  $< d$  and s.t.  $F(X) = \bar{F}(X, X^d)$  and  $G(X) = \bar{G}(X, X^d)$ .

② Compute  $\bar{H} := \text{rem}(\bar{F}\bar{G}, Y^\delta + 1)$  in  $\mathbb{B}[Y]$  by a variation of FFT, where  $\mathbb{B} = \mathbb{A}[X]/(X^{2d} + 1)$  and by recursive calls for products in  $\mathbb{B}$ .

③ Return  $H(X, X^d)$ .

## Theorem

Let  $\mathbb{A}$  be a ring in which 2 is invertible, with known inverse. Then, two polynomials of  $\mathbb{A}[X]$  of degrees  $< n$  can be multiplied in  $O(n \log n \log \log n)$  operations in  $\mathbb{A}$ .

# Multiplication functions

Abstraction  
of cost functions



Expression of complexity independent  
of the multiplication algorithm

## Definition

$M : \mathbb{N}_{>0} \rightarrow \mathbb{R}_{>0}$  is a *multiplication function* for  $\mathbb{A}[X]$  if:

- all  $P, Q$  of degree  $< n$  in  $\mathbb{A}[X]$  can be multiplied in at most  $M(n)$  arithmetic operations in  $\mathbb{A}$ ;
- $n \mapsto M(n)/n$  is an increasing function of  $n \in \mathbb{N}_{>0}$ ;
- for all  $m$  and  $n$  of  $\mathbb{N}_{>0}$ ,  $M(mn) \leq m^2 M(n)$ .

## Properties

- (*superlinearity*)  $n \leq M(n)$ ;  $M(m) + M(n) \leq M(m+n)$ ;  $m M(n) \leq M(mn)$ .
- (*usual special cases*)  $2M(n) \leq M(2n)$ ;  $\sum_i M(n_i) \leq M(\sum_i n_i)$ .
- (*at most quadratic*)  $M(n) \leq n^2$ .