Dense Linear Algebra
—from Gauss to Strassen—

Slides: Alin Bostan

Lecturer: Vincent Neiger

MPRI C-2-22

October 7, 2024

Master Internships

▷ Various topics in computer algebra; design / analyze / apply algorithms for:

- *Numeric-symbolic polynomial system solving*
  `http://dl.vwx.fr/PoK06ZjlaYw1uFst/stage-odelix-polsys.pdf`
  MAX team, Saclay. Contact: Mezzarobba, or emails in the description.
- *Numerical approach of generalized flatness*
  `https://magix.lix.polytechnique.fr/node/stage-node-e.en.html`
  MAX team, Saclay. Contact: Mezzarobba, or emails in the description.
- *Validated Numerical Software For Algebraic Curves With Singularities*
  `https://cfhp.univ-lille.fr/files/students/These-2025-NumPuiseux.pdf`
  CFHP team, Lille. Contact: Neiger, or emails in the description.
- *Algorithms for algebraic approximants and guess-and-prove approaches*
  PolSys team, Paris 5. Contact: Bostan / Neiger
- *Computational real algebraic geometry with a view towards optical system design.* PolSys team, Paris 5. Contact: Neiger
- ...

⇝ PhD funding available, contact us asap if interested

Introduction

Context

▷ Customary philosophy in mathematics:
"a problem is trivialized when it is reduced to a linear algebra question"

▷ From a computational viewpoint, it is important to address *efficiency issues* of the various linear algebra operations

▷ The most fundamental problems in linear algebra:

- linear system solving $Ax = b$,
- computation of the inverse A^{-1} of a matrix A,
- computation of determinant, rank,
- computation of minimal polynomial, characteristic polynomial,
- computation of canonical forms (LU / LDU / LUP decompositions, echelon forms, Frobenius forms = block companion, ...),
- computation of row/column reduced forms.

Warnings

▷ Natural mathematical ideas may lead to highly inefficient algorithms!
E.g., the definition of $\det(A)$, with exponential complexity in the size of A.
Also, Cramer's formulas for system solving are not very useful in practice.

▷ In all what follows, we will work with a (commutative) effective field $\mathbb{K}$, and with the (non-commutative) algebra $\mathcal{M}_n(\mathbb{K})$ of square matrices over $\mathbb{K}$.

▷ NB: most results extend to the case where $\mathbb{K}$ is replaced by a commutative effective ring $\mathbb{A}$, and to rectangular (instead of square) matrices.

Gaussian elimination

Theorem 0

For any matrix $A \in \mathcal{M}_n(\mathbb{K})$, one can compute in $O(n^3)$ operations in $\mathbb{K}$:

1. the rank $\mathrm{rk}(A)$
2. the determinant $\det(A)$
3. the inverse A^{-1}, if A is invertible
4. a (vector/affine) solutions basis of $Ax = b$, for any b in $\mathbb{K}^n$
5. an LUP decomposition (L = unit lower triangular, U = upper triangular, P = permutation matrix)
6. an LDU decomposition (L/U = unit lower/upper triangular, D = diag)
7. a reduced row echelon form (Gauss-Jordan) of A.

▷ based on *elementary row operations*: (1) swapping rows; (2) multiplying rows by scalars; (3) adding a multiple of one row to another row.

Main messages

▷ One can do better than Gaussian elimination!

▷ There exists $2 \leq \omega < 3$, the so-called "matrix multiplication exponent", that controls the complexity of all linear algebra operations.

▷ One can classify linear algebra algorithms in three categories:

- dense, without any structure (today) – their manipulation boils down essentially to *matrix multiplication*: $O(n^3) \to O(n^\omega)$, where $\omega < 2.38$
- sparse (lect. 3, 6, 9) – algos based on *linear recurrences*: $O(n^3) \to \tilde{O}(n^2)$
- structured (Vandermonde, Sylvester, Toeplitz, Hankel,..., lect. 7, 9) – based on *displacement rank* and *polynomial matrices*: $O(n^3) \to \tilde{O}(n)$

Applications

Linear algebra is ubiquitous:

- computations with D-finite power series (lect. 3)
- computations with dense power series (lect. 5)
- computation of terms of a recurrent sequence (lect. 6)
- computations with polynomial matrices (lect. 8–9)
- Hermite-Padé approximants (lect. 9)
- polynomial factorization over finite fields (lect. 10)
- solutions of linear differential equations (lect. 15)

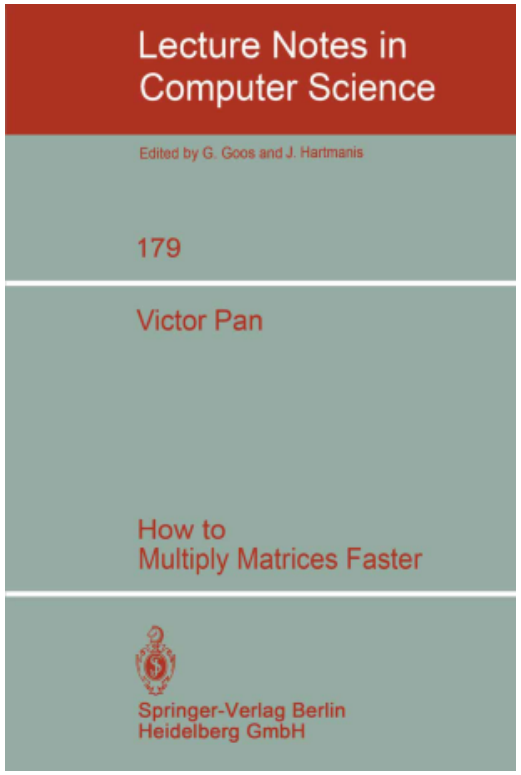
⋮

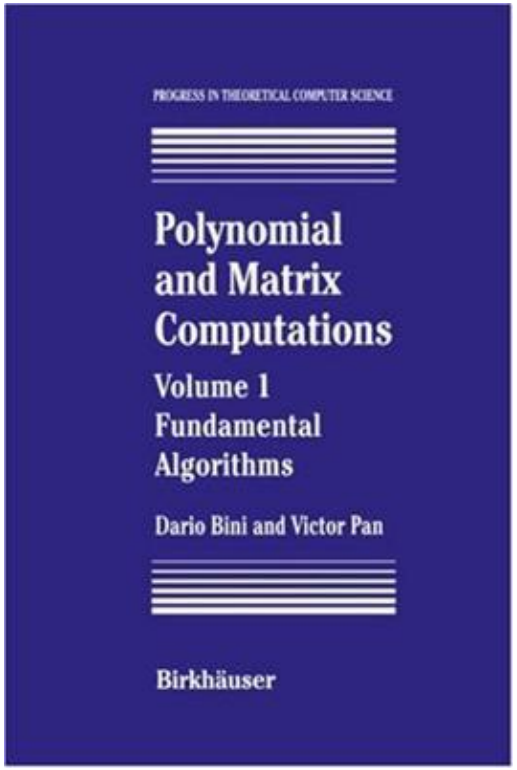
- integer factorization relies on (sparse) linear algebra over $\mathbb{F}_2$
- PageRank webpage ranking system relies on (sparse) linear algebra
- crypto-analysis: discrete logs (sparse)

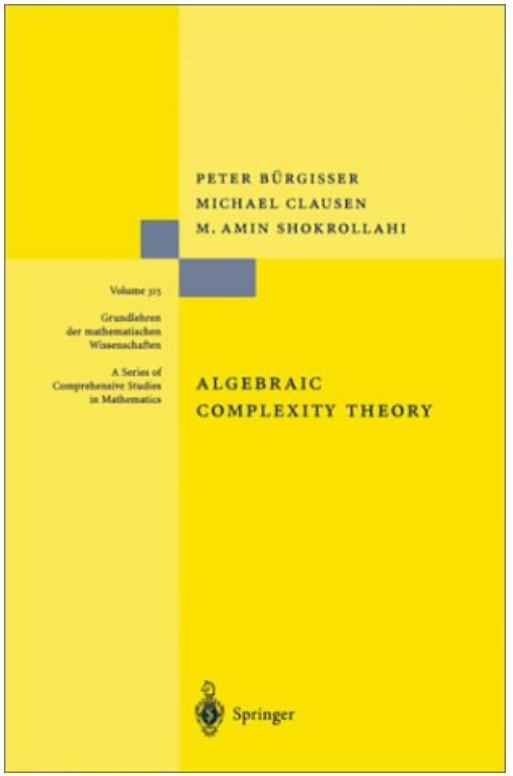
Matrix multiplication

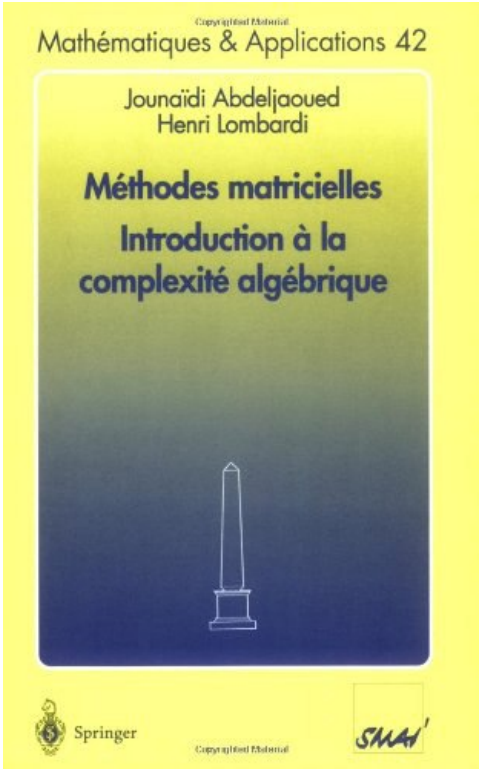
Matrix multiplication

Together with integer and polynomial multiplication, matrix multiplication is one of the most basic and most important operations in computer algebra.


Lecture Notes in
Computer Science
Edited by G. Goos and J. Hartmanis
179
Victor Pan
How to
Multiply Matrices Faster
Springer-Verlag Berlin
Heidelberg GmbH


PROGRESS IN THEORETICAL COMPUTER SCIENCE
Polynomial
and Matrix
Computations
Volume 1
Fundamental
Algorithms
Dario Bini and Victor Pan
Birkhäuser


PETER BÜRGISSER
MICHAEL CLAUSEN
M. AMIN SHOKROLLAHI
Volume 315
Grundlehren
der mathematischen
Wissenschaften
A Series of
Comprehensive Studies
in Mathematics
ALGEBRAIC
COMPLEXITY THEORY
Springer


Mathématiques & Applications 42
Jounaïdi Abdeljaoued
Henri Lombardi
Méthodes matricielles
Introduction à la
complexité algébrique
Springer
SMAI

Matrix-vector product

Theorem [Winograd'67]

The naive algorithm for multiplying a $m \times n$ *generic* matrix by a $n \times 1$ vector (using mn multiplications and $m(n-1)$ additions) *is optimal*.

▷ Natural question: is the naive matrix product in size n (using n^3 $\otimes$ and $n^3 - n^2$ $\oplus$) also optimal?

Complexity of matrix product: main results

Theorem 1 ["naive multiplication is not optimal"]

One can multiply two matrices $A, B \in \mathcal{M}_n(\mathbb{K})$ using:

1. $n^2\lceil\frac{n}{2}\rceil + 2n\lfloor\frac{n}{2}\rfloor \simeq \frac{1}{2}n^3 + n^2$ multiplications in $\mathbb{K}$ [Pan'66-Winograd'68]
2. $n^2\lceil\frac{n}{2}\rceil + (2n-1)\lfloor\frac{n}{2}\rfloor \simeq \frac{1}{2}n^3 + n^2 - \frac{n}{2}$ multiplications in $\mathbb{K}$ [Waksman'69]
3. $O(n^{\log_2 7}) \subset O(n^{2.807355})$ operations in $\mathbb{K}$ [Strassen 1969]
4. $O(n^{2.375477})$ operations in $\mathbb{K}$ [Coppersmith & Winograd, *JSC*, 1990]
5. $O(n^{2.372864})$ operations in $\mathbb{K}$ [Le Gall, *ISSAC*, 2014]
6. $O(n^{2.372860})$ operations in $\mathbb{K}$ [Alman & Vassilevska Williams, *SODA*, 2021]
7. $O(n^{2.371866})$ operations in $\mathbb{K}$ [Duan, Wu & Zhou, *FOCS*, 2023]
8. $O(n^{2.371552})$ operations in $\mathbb{K}$ [Vassilevska Williams, Xu, Xu & Zhou, *arXiv*, 2023]

A Refined Laser Method and Faster Matrix Multiplication

Josh Alman* Virginia Vassilevska Williams†

Abstract

The complexity of matrix multiplication is measured in terms of ω, the smallest real number such that two $n \times n$ matrices can be multiplied using $O(n^{\omega+\epsilon})$ field operations for all $\epsilon > 0$; the best bound until now is $\omega < 2.37287$ [Le Gall'14]. All bounds on ω since 1986 have been obtained using the so-called laser method, a way to lower-bound the 'value' of a tensor in designing matrix multiplication algorithms. The main result of this paper is a refinement of the laser method that improves the resulting value bound for most sufficiently large tensors. Thus, even before computing any specific values, it is clear that we achieve an improved bound on ω, and we indeed obtain the best bound on ω to date:

$$\omega < 2.37286.$$

The improvement is of the same magnitude as the improvement that [Le Gall'14] obtained over the previous bound [Vassilevska W.'12]. Our improvement to the laser method is quite general, and we believe it will have further applications in arithmetic complexity.

has developed a powerful toolbox of techniques, culminating in the best bound to date of $\omega < 2.37287$.

In this paper, we add one more tool to the toolbox and lower the best bound on the matrix multiplication exponent to

$$\omega < 2.37286.$$

The main contribution of this paper is a new refined version of the *laser method* which we then use to obtain the new bound on ω. The laser method (as coined by Strassen [Str86]) is a powerful mathematical technique for analyzing tensors. In our context, it is used to lower bound the "value" of a tensor in designing matrix multiplication algorithms. The laser method also has applications beyond bounding ω itself, including to other problems in arithmetic complexity like computing the "asymptotic subrank" of tensors [Alm19], and to problems in extremal combinatorics like constructing tri-colored sum-free sets [KSS18]. We believe our improved laser method may have other diverse applications.

We will see that our new method achieves better

Faster Matrix Multiplication via Asymmetric Hashing

Ran Duan *
Tsinghua University

Hongxun Wu †
UC Berkeley

Renfei Zhou ‡
Tsinghua University

April 6, 2023

Abstract

Fast matrix multiplication is one of the most fundamental problems in algorithm research. The exponent of the optimal time complexity of matrix multiplication is usually denoted by ω. This paper discusses new ideas for improving the laser method for fast matrix multiplication. We observe that the analysis of higher powers of the Coppersmith-Winograd tensor [Coppersmith & Winograd 1990] incurs a "combination loss", and we partially compensate for it using an asymmetric version of CW's hashing method. By analyzing the eighth power of the CW tensor, we give a new bound of $\omega < 2.371866$, which improves the previous best bound of $\omega < 2.372860$ [Alman & Vassilevska Williams 2020]. Our result breaks the lower bound of 2.3725 in [Ambainis, Filmus & Le Gall 2015] because of the new method for analyzing component (constituent) tensors.

New Bounds for Matrix Multiplication: from Alpha to Omega

Virginia Vassilevska Williams* Yinzhan Xu† Zixuan Xu‡ Renfei Zhou§

Abstract

The main contribution of this paper is a new improved variant of the laser method for designing matrix multiplication algorithms. Building upon the recent techniques of [Duan, Wu, Zhou FOCS'2023], the new method introduces several new ingredients that not only yield an improved bound on the matrix multiplication exponent ω, but also improves the known bounds on rectangular matrix multiplication by [Le Gall and Urrutia SODA'2018].

In particular, the new bound on ω is

$$\omega \leq 2.371552 \text{ (improved from } \omega \leq 2.371866).$$

For the dual matrix multiplication exponent α defined as the largest α for which $\omega(1, \alpha, 1) = 2$, we obtain the improvement

$$\alpha \geq 0.321334 \text{ (improved from } \alpha \geq 0.31389).$$

Similar improvements are obtained for various other exponents for multiplying rectangular matrices.

Exponent of matrix multiplication

Def. $\theta \in [2,3]$ is a *feasible exponent* for matrix multiplication over $\mathbb{K}$ if one can multiply any A and B in $\mathcal{M}_n(\mathbb{K})$ using $O(n^\theta)$ ops. in $\mathbb{K}$.

Def. Exponent of matrix multiplication $\omega = \inf\{\theta \mid \theta \text{ is a feasible exponent}\}$.

Def. $\mathsf{MM} : \mathbb{N} \to \mathbb{N}$ is a *matrix multiplication function* (for a field $\mathbb{K}$) if:

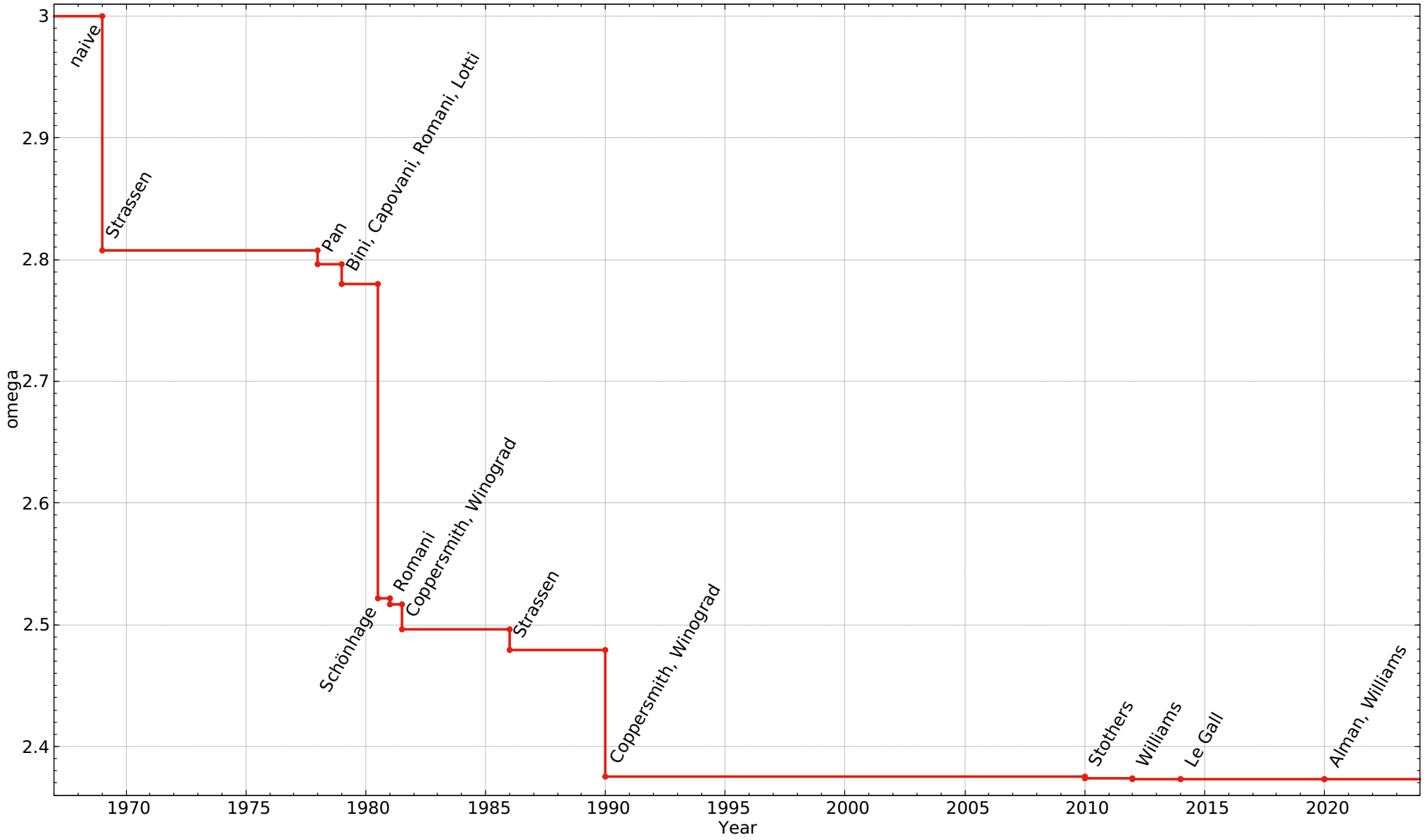
- one can multiply any A, B in $\mathcal{M}_n(\mathbb{K})$ using at most $\mathsf{MM}(n)$ ops. in $\mathbb{K}$
- MM satisfies $\mathsf{MM}(n) \leq \mathsf{MM}(2n)/4$ for all $n \in \mathbb{N}$
- $n \mapsto \mathsf{MM}(n)/n^2$ is increasing

▷ $\omega \in [2, 2.38]$

▷ if $\mathbb{K} \subset \mathbb{L}$ then $\omega_{\mathbb{K}} = \omega_{\mathbb{L}}$ [Schönhage'72], so $\omega_{\mathbb{K}}$ only depends on $\mathrm{char}(\mathbb{K})$

▷ Conjectured: ω does not depend on $\mathbb{K}$

▷ Big open problem: Is $\omega = 2$?


omega
Year
3
2.9
2.8
2.7
2.6
2.5
2.4
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
naive
Strassen
Pan
Bini, Capovani, Romani, Lotti
Schönhage
Romani
Coppersmith, Winograd
Strassen
Coppersmith, Winograd
Stothers
Williams
Le Gall
Alman, Williams

Winograd's algorithm

Naive algorithm for $n = 2$

$$R = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x & y \\ z & t \end{bmatrix} = \begin{bmatrix} ax + bz & ay + bt \\ cx + dz & cy + dt \end{bmatrix}$$

requires $8 \otimes$ and $4 \oplus$

▷ Naive algorithm for arbitrary n requires $n^3 \otimes$ and $(n^3 - n^2) \oplus$

Winograd's idea (1967): Karatsuba-like scheme

$$R = \begin{bmatrix} (a+z)(b+x) - ab - zx & (a+t)(b+y) - ab - ty \\ (c+z)(d+x) - cd - zx & (c+t)(d+y) - cd - ty \end{bmatrix}$$

▷ Drawbacks: uses commutativity (e.g., $zb = bz$); not yet profitable for $n = 2$

Winograd's algorithm

Same idea for $n = 2k$: for $\ell := (a_1, \cdots, a_n)$ and $c := (x_1, \cdots, x_n)^T$

$$\langle \ell | c \rangle = (a_1 + x_2)(a_2 + x_1) + \cdots + (a_{2k-1} + x_{2k})(a_{2k} + x_{2k-1}) - \sigma(\ell) - \sigma(c),$$

where $\sigma(\ell) := a_1 a_2 + \cdots + a_{2k-1} a_{2k}$ and $\sigma(c) := x_1 x_2 + \cdots + x_{2k-1} x_{2k}$

The element $r_{i,j}$ of $R = AX$ is the scalar product $\langle \ell_i | c_j \rangle$, where $\ell_1, \ldots, \ell_n$ are the rows of A and $c_1, \ldots, c_n$ are the columns of X

Winograd's algorithm:

- precompute $\sigma(\ell_i)$ for $1 \leq i \leq n \longrightarrow \quad nk = \frac{n^2}{2} \otimes$ and $n(k-1) = \frac{n^2}{2} - n \oplus$
- precompute $\sigma(c_j)$ for $1 \leq j \leq n \longrightarrow \quad nk = \frac{n^2}{2} \otimes$ and $n(k-1) = \frac{n^2}{2} - n \oplus$
- compute all $r_{i,j} := \langle \ell_i | c_j \rangle \longrightarrow n^2 k = \frac{n^3}{2} \otimes$ and $n^2(n+k+1) = \frac{3n^3}{2} + n^2 \oplus$

▷ Total: $\frac{1}{2}n^3 + n^2 \otimes$ and $\frac{3}{2}n^3 + 2n^2 - 2n \oplus$

Waksman's algorithm

Idea for $n = 2$: write

$$R = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x & y \\ z & t \end{bmatrix} = \begin{bmatrix} ax + bz & ay + bt \\ cx + dz & cy + dt \end{bmatrix}$$

as

$$R = \frac{1}{2} \begin{bmatrix} (a+z)(b+x) - (a-z)(b-x) & (a+t)(b+y) - (a-t)(b-y) \\ (c+z)(d+x) - (c-z)(d-x) & (c+t)(d+y) - (c-t)(d-y) \end{bmatrix},$$

and observe that the sum of the 4 products in red is equal to the sum of the 4 products in blue (and equal to $ab + zx + cd + ty$)

- 2×2 matrix product in 7 commutative $\otimes$, when $\mathrm{char}(\mathbb{K}) \neq 2$
- Idea generalizes to $n \times n$ matrices $\longrightarrow \frac{1}{2}n^3 + n^2 - \frac{n}{2}$ $\otimes$ for even n

Winograd/Waksman: summary

▷ They have cubic complexity, but are nevertheless useful in several contexts, e.g. products of small matrices containing large integers

▷ They already show that naive multiplication is not optimal

▷ Their weakness is the use of commutativity of the base ring, which does not allow a recursive use on blocks

▷ Natural question: can we do 7 non-commutative $\otimes$?

Matrix multiplication

Strassen's algorithm

Matrix multiplication
Strassen's algorithm

Strassen was attempting to prove, by process of elimination, that such an algorithm does not exist when he arrived at it.

> "First I had realized that an estimate tensor rank < 8 for two by two matrix multiplication would give an asymptotically faster algorithm. Then I worked over $\mathbb{Z}/2\mathbb{Z}$ (as far as I remember) to simplify matters."

Strassen's matrix multiplication algorithm

Same idea as for Karatsuba's algorithm: trick in low size + recursion

Additional difficulty: Formulas should be non-commutative

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x & y \\ z & t \end{bmatrix} \iff \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix}$$

Crucial remark: If $\varepsilon \in \{0, 1\}$ and $\alpha \in \mathbb{K}$, then 1 multiplication suffices for $E \cdot v$, where v is a vector, and E is a matrix of one of the following types:

$$\begin{bmatrix} \alpha & \alpha \\ \varepsilon\alpha & \varepsilon\alpha \end{bmatrix}, \begin{bmatrix} \alpha & -\alpha \\ \varepsilon\alpha & -\varepsilon\alpha \end{bmatrix}, \begin{bmatrix} \alpha & \varepsilon\alpha \\ -\alpha & -\varepsilon\alpha \end{bmatrix}$$

Strassen's matrix multiplication algorithm

Problem: Write

$$M = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

as a sum of (strictly) less than 8 elementary matrices.

$$M - \underbrace{\begin{bmatrix} a & a & & \\ a & a & & \\ & & & \\ & & & \end{bmatrix}}_{E_1} - \underbrace{\begin{bmatrix} & & & \\ & & & \\ & & d & d \\ & & d & d \end{bmatrix}}_{E_2} = \begin{bmatrix} & b-a & & \\ c-a & d-a & & \\ & & a-d & b-d \\ & & c-d & \end{bmatrix}$$

Strassen's matrix multiplication algorithm

Problem: Write

$$M = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

as a sum of less than 8 elementary matrices.

$$M - E_1 - E_2 = \underbrace{\begin{bmatrix} & & & \\ & & & \\ & d-a & a-d & \\ & d-a & a-d & \end{bmatrix}}_{E_3} + \begin{bmatrix} & b-a & & \\ c-a & & d-a & \\ & a-d & & b-d \\ & & c-d & \end{bmatrix}$$

Strassen's matrix multiplication algorithm

Problem: Write

$$M = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

as a sum of less than 8 elementary matrices.

$$M - E_1 - E_2 - E_3 = \begin{bmatrix} & b-a & & \\ & & & \\ & a-d & & b-d \\ & & & \end{bmatrix} + \begin{bmatrix} c-a & & d-a & \\ & & & \\ & & & c-d \\ & & & \end{bmatrix}$$

Strassen's matrix multiplication algorithm

Problem: Write

$$M = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

as a sum of less than 8 elementary matrices.

$$M - E_1 - E_2 - E_3 = \underbrace{\begin{bmatrix} & b-a & \\ & & \\ (b-d)-(b-a) & & b-d \end{bmatrix}}_{E_5} + \underbrace{\begin{bmatrix} c-a & & (c-a)-(c-d) \\ & & \\ & c-d & \end{bmatrix}}_{E_6} + E_4 + E_7$$

Strassen's matrix multiplication algorithm

Problem: Write

$$M = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

as a sum of less than 8 elementary matrices.

Conclusion

$$M = E_1 + E_2 + E_3 + E_4 + E_5 + E_6 + E_7$$

$\Longrightarrow$ one can multiply 2×2 matrices using 7 non-comm products instead of 8

DAC Theorem:

$\mathsf{MM}(r) = 7 \cdot \mathsf{MM}(r/2) + O(r^2) \quad \Longrightarrow \quad \mathsf{MM}(r) = O(r^{\log_2(7)}) = O(r^{2.81})$

$$\underbrace{\begin{bmatrix} a & a & \cdot & \cdot \\ a & a & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{E_1} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} a(x+z) \\ a(x+z) \\ \cdot \\ \cdot \end{bmatrix}, \underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & d & d \\ \cdot & \cdot & d & d \end{bmatrix}}_{E_2} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} \cdot \\ \cdot \\ d(y+t) \\ d(y+t) \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & d-a & a-d & \cdot \\ \cdot & d-a & a-d & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{E_3} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} \cdot \\ (d-a)(z-y) \\ (d-a)(z-y) \\ \cdot \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & b-d & \cdot & b-d \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{E_4} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} \cdot \\ \cdot \\ (b-d)(z+t) \\ \cdot \end{bmatrix}, \underbrace{\begin{bmatrix} \cdot & b-a & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & -(b-a) & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{E_5} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} (b-a)z \\ \cdot \\ -(b-a)z \\ \cdot \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ c-a & \cdot & c-a & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{E_6} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} \cdot \\ (c-a)(x+y) \\ \cdot \\ \cdot \end{bmatrix}, \underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & -(c-d) & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & c-d & \cdot \end{bmatrix}}_{E_7} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} \cdot \\ -(c-d)y \\ \cdot \\ (c-d)y \end{bmatrix}$$

▷ In summary, 7 $\otimes$ (non-comm.) and 18 $\oplus$:

$$\begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \times \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix} = \begin{bmatrix} a(x+z)+(b-a)z \\ a(x+z)+(d-a)(z-y)+(c-a)(x+y)-(c-d)y \\ d(y+t)+(d-a)(z-y)+(b-d)(z+t)-(b-a)z \\ d(y+t)+(c-d)y \end{bmatrix}$$

▷ Extension: $n^3 - n(n-1)/2$ non-comm. $\otimes$ for $n \times n$ [Fiduccia'72]

▷ 7 non-comm. $\otimes$ and 15 $\oplus$ [Winograd'71] (instead of 18 $\oplus$ for [Strassen'69])

▷ Optimality: [Winograd'71], [Hopcroft & Kerr'71] (7 $\otimes$); [Probert'73] (15 $\oplus$)

Input Two matrices $A, X \in \mathcal{M}_n(\mathbb{K})$, with $n = 2^k$.

Output The product AX.

1. If $n = 1$, return AX.
2. Write $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, $X = \begin{bmatrix} x & y \\ z & t \end{bmatrix}$, with $a, b, c, d, x, y, z, t \in \mathcal{M}_{n/2}(\mathbb{K})$.
3. Compute recursively the products

$$\begin{aligned} q_1 &= a(x+z), & q_2 &= d(y+t), \\ q_3 &= (d-a)(z-y), & q_4 &= (b-d)(z+t) \\ q_5 &= (b-a)z, & q_6 &= (c-a)(x+y), \quad q_7 = (c-d)y. \end{aligned}$$

4. Compute the sums

$$\begin{aligned} r_{1,1} &= q_1 + q_5, & r_{1,2} &= q_2 + q_3 + q_4 - q_5, \\ r_{2,1} &= q_1 + q_3 + q_6 - q_7, & r_{2,2} &= q_2 + q_7. \end{aligned}$$

5. Return $\begin{bmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{bmatrix}$.

In practice

▷ in a good implementation, Winograd & Waksman algorithms are interesting for small sizes

▷ Strassen's algorithm then becomes the best for $n \approx 64$

▷ Kaporin's algorithm becomes the best for $n \approx 500$

(note: these thresholds depend on the field/ring of coefficients and on the implementation)

▷ best practical algorithm is [Kaporin'04]: it uses $n^3/3 + 4n^2 + 8n$ non-comm. $\otimes$ in size n. Choosing $n = 48$ leads to $O(n^{\log_{48}(46464)}) = O(n^{2.776})$

▷ the vast majority of the other algorithms rely on techniques that are too complex, and that implies very big constants in the $O(\cdot) \longrightarrow$ interesting for sizes over millions or billions ("galactic algorithms")

Other linear algebra problems

Complexity of linear algebra: main results

Theorem 2 ["Gaussian elimination is not optimal"]

Let θ be a feasible exponent for matrix multiplication in $\mathcal{M}_n(\mathbb{K})$. Then, one can compute:

1. the inverse A^{-1} and the determinant $\det(A)$ of $A \in \mathrm{GL}_n(\mathbb{K})$ [Strassen'69]
2. the solution of $Ax = b$ for any $A \in \mathrm{GL}_n(\mathbb{K})$ and $b \in \mathbb{K}^n$ [Strassen'69]
3. the LUP and LDU decompositions of A [Bunch & Hopcroft'74]
4. the rank $\mathrm{rk}(A)$ and an echelon form [Schönhage'72] of any $A \in \mathcal{M}_n(\mathbb{K})$
5. the characteristic polynomial $\chi_A(x)$ and the minimal polynomial $\mu_A(x)$ of any $A \in \mathcal{M}_n(\mathbb{K})$ [Keller-Gehrig'85]

using $\tilde{O}(n^\theta)$ operations in $\mathbb{K}$.

Complexity of linear algebra: main results

Theorem 3 ["equivalence of linear algebra problems"]

The following problems on matrices in $\mathcal{M}_n(\mathbb{K})$

- multiplication
- inversion
- determinant
- characteristic polynomial
- LUP decomposition for matrices of full rank

all have the same asymptotic complexity, up to logarithmic factors.

In other words, the exponent ω controls the complexity of all these problems:

$$\omega = \omega_{\mathsf{inv}} = \omega_{\mathsf{det}} = \omega_{\mathsf{charpoly}} = \omega_{\mathsf{LUP}}$$

▷ Open: are ω_{solve} and ω_{rank} and $\omega_{\mathsf{isinvertible}}$ also equal to ω?

Inversion is not harder than multiplication

▷ [Strassen'69] showed how to reduce matrix inversion (and also linear system solving) to matrix multiplication

▷ His result is: one can invert a (generic) $n \times n$ matrix in $O(n^\theta)$ ops.

$\longrightarrow$ "Gaussian elimination is not optimal"

▷ [Klyuyev & Kokovkin-Shcherbak'65] had previously proven that Gaussian elimination *is optimal* if one restricts to row and column operations.

▷ Strassen's method is a *Gaussian elimination by blocks*, applied recursively

▷ His algo requires 2 inversions, 6 multiplications and 2 additions, in size $\frac{n}{2}$:

$$\mathsf{I}(n) \leq 2\mathsf{I}(n/2) + 6\mathsf{MM}(n/2) + n^2/2 \leq 3\sum_i 2^i \cdot \mathsf{MM}(n/2^i) + O(n^2) = O(\mathsf{MM}(n))$$

Inversion of dense matrices

- Starting point is the (non commutative!) identity ($a, b, c, d \in \mathbb{K}^\star$)

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ ca^{-1} & 1 \end{bmatrix} \times \begin{bmatrix} a & 0 \\ 0 & z \end{bmatrix} \times \begin{bmatrix} 1 & a^{-1}b \\ 0 & 1 \end{bmatrix},$$

where $z = d - ca^{-1}b$ is the *Schur complement* of a in M.

- It is a consequence of Gauss pivoting on M (LDU decomposition)

- The UDL matrix factorization of the inverse of M follows:

$$\begin{aligned} \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} &= \begin{bmatrix} 1 & -a^{-1}b \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} a^{-1} & 0 \\ 0 & z^{-1} \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ -ca^{-1} & 1 \end{bmatrix} \\ &= \begin{bmatrix} a^{-1} + a^{-1}bz^{-1}ca^{-1} & -a^{-1}bz^{-1} \\ -z^{-1}ca^{-1} & z^{-1} \end{bmatrix}. \end{aligned}$$

- This identity being non-commutative, it also holds for *matrices* a, b, c, d

Inversion of dense matrices

[Strassen, 1969]

To invert a dense matrix $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \in \mathcal{M}_n(\mathbb{K})$, with $A, B, C, D \in \mathcal{M}_{\frac{n}{2}}(\mathbb{K})$

0. If $n = 1$, then return M^{-1}.
1. Invert A (recursively): $E := A^{-1}$.
2. Compute the Schur complement: $Z := D - CEB$.
3. Invert Z (recursively): $T := Z^{-1}$.
4. Recover the inverse of M as

$$M^{-1} := \begin{bmatrix} E + EBTCE & -EBT \\ -TCE & T \end{bmatrix}.$$

DAC Theorem: $\mathsf{I}(n) = 2 \cdot \mathsf{I}\left(\frac{n}{2}\right) + O(\mathsf{MM}(n)) \quad \Longrightarrow \quad \mathsf{I}(n) = O(\mathsf{MM}(n))$

Corollary: inversion M^{-1} and system solving $x = M^{-1}b$ in time $O(\mathsf{MM}(n))$

Determinant of dense matrices

[Strassen, 1969]

To compute $\det(M)$ for $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \in \mathcal{M}_n(\mathbb{K})$, with $A, B, C, D \in \mathcal{M}_{\frac{n}{2}}(\mathbb{K})$

0. If $n = 1$, then return M.
1. Compute $E := A^{-1}$ and (recursively) $d_A := \det(A)$.
2. Compute the Schur complement: $Z := D - CEB$.
3. Compute (recursively) $d_Z := \det(Z)$.
4. Recover the determinant $\det(M)$ as $d_A \cdot d_Z$.

DAC Theorem:

$$\mathsf{D}(n) = 2 \cdot \mathsf{D}\left(\tfrac{n}{2}\right) + 2 \cdot \mathsf{I}\left(\tfrac{n}{2}\right) + O(\mathsf{MM}(n)) \quad \Longrightarrow \quad \mathsf{D}(n) = O(\mathsf{MM}(n))$$

Corollary: Determinant $\det(M)$ in time $O(\mathsf{MM}(n))$

Multiplication is not harder than inversion

[Munro, 1973]

Let A and B two $n \times n$ matrices. To compute $C = AB$, set

$$D = \begin{bmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{bmatrix}.$$

Then the following identity holds:

$$D^{-1} = \begin{bmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{bmatrix}$$

Thus $n \times n$ multiplication reduces to inversion in size $3n \times 3n$: $\omega_{\text{mul}} \leq \omega_{\text{inv}}$.

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

Computation of the characteristic polynomial

[Keller-Gehrig, 1985]

▷ Assume $A \in \mathcal{M}_n(\mathbb{K})$ generic, in particular $\chi_A := \det(xI_n - A)$ irred. in $\mathbb{K}[x]$

▷ This implies $\chi_A(x) = \mu_A(x)$ and $\mathcal{B} := \{v, Av, \ldots, A^{n-1}v\}$ is a $\mathbb{K}$-basis of $\mathbb{K}^n$

Lemma. If $v \in \mathbb{K}^n \setminus \{0\}$, then $P := \left[v|Av|\cdots|A^{n-1}v\right]$ is invertible and $C := P^{-1}AP$ is in companion form

Proof. If $\chi_A(x) = x^n - p_{n-1}x^{n-1} - \cdots - p_1x - p_0$, then the matrix C of $f : w \mapsto Aw$ w.r.t. $\mathcal{B}$ is companion, with last column $[p_0, \ldots, p_{n-1}]^T$.

Algorithm.

- Compute the matrix $P := [v|Av|\cdots|A^{n-1}v]$ $O(n^?)$
- Compute the inverse $M := P^{-1}$ $O(n^\theta)$
- Return the last column of MAP $O(n^\theta)$

Computation of the characteristic polynomial

[Keller-Gehrig, 1985]

▷ Remaining task: fast computation of the *Krylov sequence*

$$\{v, Av, \ldots, A^{n-1}v\}$$

▷ Naive algorithm: $v \xrightarrow{A\cdot} Av \xrightarrow{A\cdot} A^2v \xrightarrow{A\cdot} \cdots \xrightarrow{A\cdot} A^{n-1}v$ $O(n^3)$

▷ Keller-Gehrig algorithm: Compute

1. $A_0 := A$ and $A_k := A_{k-1}^2$ for $k = 1, 2, \ldots$ (binary powering) $O(n^\theta \log(n))$
2. $\left[A^{2^k}v | \cdots | A^{2^{k+1}-1}v\right] := A_k \times \left[v | \cdots | A^{2^k-1}v\right]$ for $k = 1, 2, \ldots$ $O(n^\theta \log(n))$

▷ Conclusion: Krylov sequence, and thus $\chi_A(x)$, in $O(n^\theta \log(n))$

The Keller-Gehrig algorithm

Input A matrix $A \in \mathcal{M}_n(\mathbb{K})$, with $n = 2^k$.

Output Its characteristic polynomial $\chi_A(x) = \det(xI_n - A)$.

1. Choose v in $\mathbb{K}^n \setminus \{0\}$.
2. Set $M := A$ and $P := v$.
3. For i from 1 to k, replace P by the horizontal concatenation of P and MP, then M by M^2.
4. Compute $C := P^{-1}AP$ and let $[p_0, \ldots, p_{n-1}]^T$ be its last column.
5. Return $x^n - p_{n-1}x^{n-1} - \cdots - p_0$.

Journal of Complexity 67 (2021) 101572

Contents lists available at ScienceDirect

Journal of Complexity

www.elsevier.com/locate/jco

Deterministic computation of the characteristic polynomial in the time of matrix multiplication ☆

Vincent Neiger [a], Clément Pernet [b,*]

[a] Univ. Limoges, CNRS, XLIM, UMR 7252, F-87000 Limoges, France
[b] Université Grenoble Alpes, Laboratoire Jean Kuntzmann, CNRS, UMR 5224, 700 avenue centrale, IMAG - CS 40700, 38058 Grenoble cedex 9, France

ARTICLE INFO

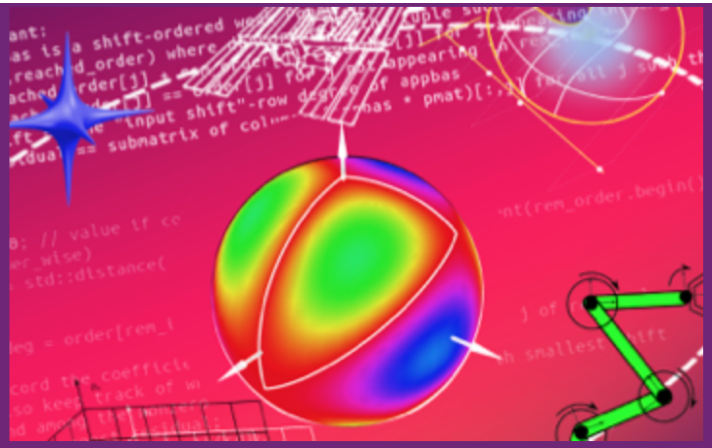
Article history:
Received 9 October 2020
Received in revised form 6 April 2021
Accepted 9 April 2021
Available online 16 April 2021

Keywords:
Characteristic polynomial
Polynomial matrices
Determinant
Fast linear algebra

ABSTRACT

This paper describes an algorithm which computes the characteristic polynomial of a matrix over a field within the same asymptotic complexity, up to constant factors, as the multiplication of two square matrices. Previously, this was only achieved by resorting to genericity assumptions or randomization techniques, while the best known complexity bound with a general deterministic algorithm was obtained by Keller-Gehrig in 1985 and involves logarithmic factors. Our algorithm computes more generally the determinant of a univariate polynomial matrix in reduced form, and relies on new subroutines for transforming shifted reduced matrices into shifted weak Popov matrices, and shifted weak Popov matrices into shifted Popov matrices.

▷ Best Paper Award of the Journal of Complexity 2021



Fundamental Algorithms and Algorithmic Complexity

18–29 sept. 2023
Institut Henri Poincaré
Fuseau horaire Europe/Paris

- Accueil
- Ordre du jour
- Liste des contributions
- Inscription
- Live stream and chat
- RTCA main IHP page
- Dedicated site rtca2023.github.io

Contact

RTCA2023-Paris@ihp.fr

On the complexity of computing characteristic polynomials by Clément Pernet

28 sept. 2023, 14:00
1h
Amphithéâtre Hermite / Darboux (Institut Henri Poincaré)

Description

Abstract. : Among the classical problems in computational linear algebra, the computation of the characteristic polynomial is of great relevance for applications as it reflects most invariants of the input matrix. It is a key component in the solution of many other related problems, such as computing eigenvalues, invariant factors and invariant subspace decomposition, testing matrices for similarity, Krylov methods etc. Computing characteristic polynomials efficiently is surprisingly challenging and has lead to a very diverse algorithmic landscape, as it lies in-between scalar linear algebra and modules of polynomial matrices. For instance, finding a deterministic reduction to dense matrix multiplication was an open-problem until recently. We will introduce some of these algorithmic techniques to present recent complexity improvements for the computation of characteristic polynomials: with dense matrices, first, we will present a recent work achieving the first reduction to matrix multiplication, based on polynomial matrix arithmetic. Then, in the context of matrices with a displacement rank structure, we will present algorithms, leading to the first sub-quadratic time cost.
This talk is based on joint work with P. Karpman, V. Neiger, H. Signargout and G. Villard.

Institut Henri Poincaré

Lemma

A right kernel basis of $\mathbf{A} \in \mathbb{K}[x]^{m \times O(m)}$ *with constant degree can be computed in reduced form in* $O(m^\omega)$ *field operations.*

Corollary

The Krylov matrix $\mathbf{K}_{\mathbf{A},\mathbf{v}} = \begin{bmatrix} \mathbf{v} & \mathbf{A}\mathbf{v} & \dots & \mathbf{A}^{m-1}\mathbf{v} \end{bmatrix}$ with $\mathbf{A} \in \mathbb{K}^{m \times m}$ can be computed in $O(m^\omega)$.

Sketch of proof.

$$\left[\begin{array}{c|c} \mathbf{I}_m - x\mathbf{A} & -\mathbf{v} \end{array}\right] \begin{bmatrix} s \\ t \end{bmatrix} = 0$$

Hence

$$s/t = (\mathbf{I}_m - x\mathbf{A})^{-1}\mathbf{v} = \sum_{i=0}^{\infty} x^k \mathbf{A}^k \mathbf{v}.$$

A truncated series expansion of s/t at order m produces the Krylov iterates. □

Two exercises for next Monday

(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.

Bonus

1. Multiplication in small sizes

Contributed Paper ISSAC '21, July 18–23, 2021, Virtual Event, Russian Federation

The Tensor Rank of 5×5 Matrices Multiplication is Bounded by 98 and Its Border Rank by 89

Alexandre Sedoglavic
UMR CNRS 9189 CRISTAL
Université de Lille
F-59000 Lille, France

Alexey V. Smirnov
Russian Federal Center of Forensic Science
Department of Justice
Moscow, Russia

ABSTRACT

We present a non-commutative algorithm for the product of 3×5 by 5×5 matrices using 58 multiplications. This algorithm allows to construct a non-commutative algorithm for multiplying 5×5 (resp. $10 \times 10, 15 \times 15$) matrices using 98 (resp. 686, 2088) multiplications. Furthermore, we describe an approximate algorithm that requires 89 multiplications and computes this product with an arbitrary small error.

CCS CONCEPTS

• **Computing methodologies** → **Exact arithmetic algorithms**; **Linear algebra algorithms**.

This non-commutative scheme is classically interpreted as a tensor *(see precise encoding in Section 2) and we recall that the number r of its summands is the* rank *of that tensor. In this work, the notations $\langle m \times n \times p : r \rangle$ stands for a tensor of rank r encoding the product $\mathcal{M}_{m,n,p}$. We denote by $\langle m \times n \times p \rangle$ the whole family of such schemes independently of their rank. The* tensor rank *$R\langle m \times n \times p \rangle$ of the considered matrix product is the smallest integer r such that there is a tensor $\langle m \times n \times p : r \rangle$ in $\langle m \times n \times p \rangle$. Similarly, $\{m \times n \times p : r\}$ denotes a computational scheme of rank r involving a parameter ϵ whose limit computes the matrix product $\mathcal{M}_{m,n,p}$ exactly as ϵ tends to zero. The* border rank *of $\mathcal{M}_{m,n,p}$ is the smallest integer r such that there exists an approximate scheme $\{m \times n \times p : r\}$.*

1. Multiplication in small sizes

Algorithm	tensor rank	naive tensor rank	construction	ratio	complexity	stabilizer
⟨2×2×2:7⟩	7	8	Strassen 1969 [DOI]	.8750000000	2.807354922	$(S_3)\rtimes S_3$
⟨3×3×3:23⟩	23	27	Laderman 1976 [DOI]	.8518518518	2.854049830	$(C_2\times C_2)\rtimes S_3$
⟨4×4×4:49⟩	49	64	⟨2×2×2:7⟩ ⊗ ⟨2×2×2:7⟩	.7656250000	2.807354922	$S_3\times S_3$
⟨5×5×5:97⟩	97	125	Kauers and Moosbauer 2022 [arXiv]	.7760000000	2.842427746	C_1
⟨6×6×6:160⟩	160	216	⟨3×3×6:40⟩ ⊗ ⟨2×2×1:4⟩	.7407407407	2.832508438	$(C_2\times C_2)\rtimes S_4\times C_2\times C_2$
⟨7×7×7:250⟩	250	343	⟨4×4×4:49⟩ + 3 ⟨3×3×4:29⟩ + 3 ⟨3×4×4:38⟩	.7288629738	2.837469613	C_1
⟨8×8×8:343⟩	343	512	⟨2×2×2:7⟩ ⊗ ⟨4×4×4:49⟩	.6699218750	2.807354922	$S_3\times S_3\times S_3$
⟨9×9×9:498⟩	498	729	6 ⟨3×3×4:29⟩ + 9 ⟨3×3×5:36⟩	.6831275720	2.826565905	C_1
⟨10×10×10:679⟩	679	1000	⟨2×2×2:7⟩ ⊗ ⟨5×5×5:97⟩	.6790000000	2.831869774	S_3
⟨11×11×11:896⟩	896	1331	⟨2×2×5:18⟩ + 2 ⟨2×3×6:30⟩ + 3 ⟨3×3×5:36⟩ + 6 ⟨2×3×5:25⟩ + 14 ⟨3×3×6:40⟩	.6731780616	2.834961347	C_1
⟨12×12×12:1040⟩	1040	1728	⟨2×4×4:26⟩ ⊗ ⟨6×3×3:40⟩	.6018518518	2.795668800	$C_2\times(C_2\times C_2)\rtimes S_4$
⟨13×13×13:1443⟩	1443	2197	⟨6×7×7:215⟩ + 4 ⟨6×6×7:185⟩ + TA(⟨7×7×7⟩, ⟨7×7×7⟩)	.6568047337	2.836110404	C_1
⟨14×14×14:1720⟩	1720	2744	⟨7×7×7:250⟩ + 3 TA(⟨7×7×7⟩, ⟨7×7×7⟩)	.6268221574	2.823007854	C_1
⟨15×15×15:2088⟩	2088	3375	⟨3×3×5:36⟩ ⊗ ⟨5×5×3:58⟩	.6186666667	2.822681037	C_1
⟨16×16×16:2401⟩	2401	4096	⟨2×2×2:7⟩ ⊗ ⟨8×8×8:343⟩	.5861816406	2.807354922	$S_3\times S_3\times S_3\times S_3$

▷ Sedoglavic: online catalogue of 5426 fast matrix multiplication algorithms

1. Multiplication in small sizes

> Nature. 2022 Oct;610(7930):47-53. doi: 10.1038/s41586-022-05172-4. Epub 2022 Oct 5.

Discovering faster matrix multiplication algorithms with reinforcement learning

Alhussein Fawzi [# 1], Matej Balog [# 2], Aja Huang [# 2], Thomas Hubert [# 2], Bernardino Romera-Paredes [# 2], Mohammadamin Barekatain [2], Alexander Novikov [2], Francisco J R Ruiz [2], Julian Schrittwieser [2], Grzegorz Swirszcz [2], David Silver [2], Demis Hassabis [2], Pushmeet Kohli [2]

Abstract

Improving the efficiency of algorithms for fundamental computations can have a widespread impact, as it can affect the overall speed of a large amount of computations. Matrix multiplication is one such primitive task, occurring in many systems-from neural networks to scientific computing routines. The automatic discovery of algorithms using machine learning offers the prospect of reaching beyond human intuition and outperforming the current best human-designed algorithms. However, automating the algorithm discovery procedure is intricate, as the space of possible algorithms is enormous. Here we report a deep reinforcement learning approach based on AlphaZero[1] for discovering efficient and provably correct algorithms for the multiplication of arbitrary matrices. Our agent, AlphaTensor, is trained to play a single-player game where the objective is finding tensor decompositions within a finite factor space. AlphaTensor discovered algorithms that outperform the state-of-the-art complexity for many matrix sizes. Particularly relevant is the case of 4 × 4 matrices in a finite field, where AlphaTensor's algorithm improves on Strassen's two-level algorithm for the first time, to our knowledge, since its discovery 50 years ago[2]. We further showcase the flexibility of AlphaTensor through different use-cases: algorithms with state-of-the-art complexity for structured matrix multiplication and improved practical efficiency by optimizing matrix multiplication for runtime on specific hardware. Our results highlight AlphaTensor's ability to accelerate the process of algorithmic discovery on a range of problems, and to optimize for different criteria.

1. Multiplication in small sizes

Flip Graphs for Matrix Multiplication

Manuel Kauers*
Jakob Moosbauer†

ABSTRACT

We introduce a new method for discovering matrix multiplication schemes based on random walks in a certain graph, which we call the flip graph. Using this method, we were able to reduce the number of multiplications for the matrix formats $(4, 4, 5)$ and $(5, 5, 5)$, both in characteristic two and for arbitrary ground fields.

ISSAC 2023, July 24–27, 2023, Tromsø, Norway M. Kauers and J. Moosbauer

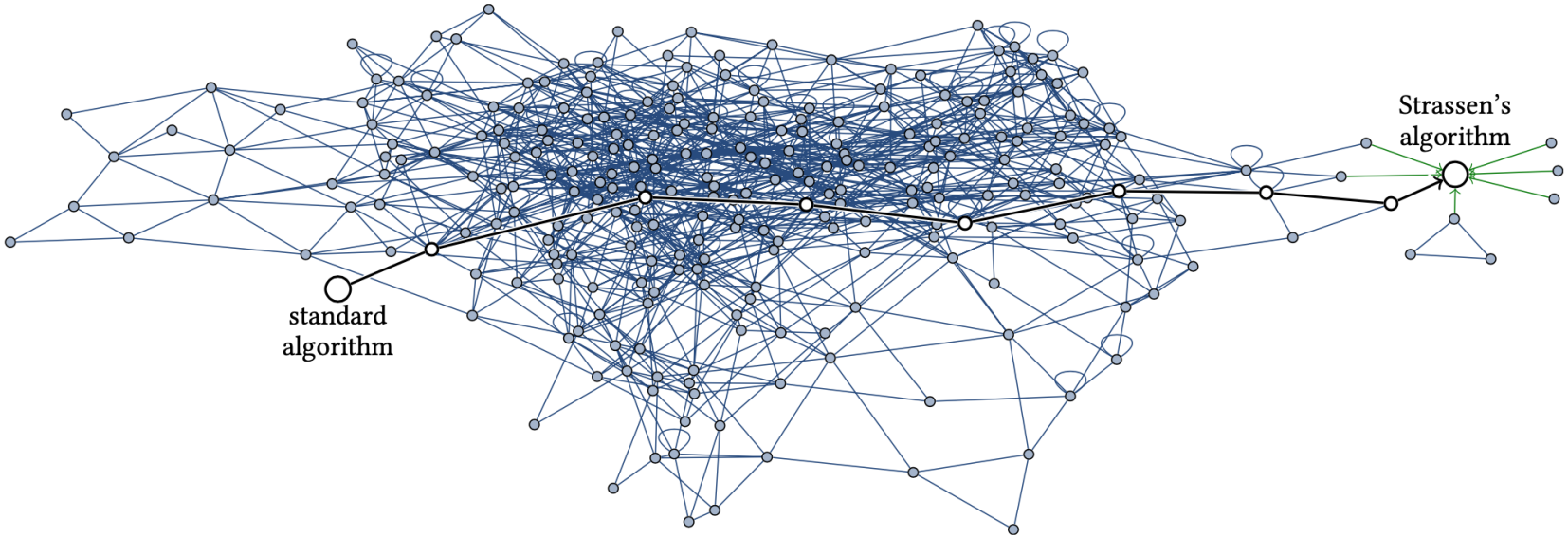


Figure 1: In the $(2, 2, 2)$-flip graph of rank at most 8, this figure shows the component containing the standard algorithm. Flips are depicted by undirected edges and reductions by directed edges.

1. Multiplication in small sizes

(n, m, p)	K	previously best known rank	our rank
(2, 2, 2)	any	7 [18]	7
(2, 2, 3)	any	11 [11]	11
(2, 2, 4)	any	14 [11]	14
(2, 3, 3)	any	15 [11]	15
(2, 2, 5)	any	18 [11]	18
(2, 3, 4)	any	20 [11]	20
(3, 3, 3)	any	23 [14]	23
(2, 3, 5)	any	25 [11]	25
(2, 4, 4)	any	26 [11]	26
(3, 3, 4)	any	29 [17]	29
(2, 4, 5)	any	33 [11]	33
(3, 3, 5)	any	36 [17]	36
(3, 4, 4)	any	38 [17]	38
(2, 5, 5)	any	40 [11]	40
(3, 4, 5)	any	47 [8]	47
(4, 4, 4)	$\mathbb{Z}_2$	47 [8]	47
(4, 4, 4)	any	49 [18]	49
(3, 5, 5)	any	58 [16]	58
(4, 4, 5)	$\mathbb{Z}_2$	63 [8]	**60**
(4, 4, 5)	any	63 [8]	**62**
(4, 5, 5)	any	76 [8]	76
(5, 5, 5)	$\mathbb{Z}_2$	96 [8]	**95**
(5, 5, 5)	any	98 [16]	**97**

▷ [Kauers & Moosbauer, 2023] – best paper award ISSAC'23

2. Is $\omega = 2$?

Matrix Multiplication via Matrix Groups

Abstract

In 2003, Cohn and Umans proposed a group-theoretic approach to bounding the exponent of matrix multiplication. Previous work within this approach ruled out certain families of groups as a route to obtaining $\omega = 2$, while other families of groups remain potentially viable. In this paper we turn our attention to matrix groups, whose usefulness within this framework was relatively unexplored.

We first show that groups of Lie type cannot prove $\omega = 2$ within the group-theoretic approach. This is based on a representation-theoretic argument that identifies the second-smallest dimension of an irreducible representation of a group as a key parameter that determines its viability in this framework. Our proof builds on Gowers' result concerning product-free sets in quasirandom groups. We then give another barrier that rules out certain natural matrix group constructions that make use of subgroups that are far from being self-normalizing.

Our barrier results leave open several natural paths to obtain $\omega = 2$ via matrix groups. To explore these routes we propose working in the continuous setting of Lie groups, in which we develop an analogous theory. Obtaining the analogue of $\omega = 2$ in this potentially easier setting is a key challenge that represents an intermediate goal short of actually proving $\omega = 2$. We give two constructions in the continuous setting, each of which evades one of our two barriers.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Fast matrix multiplication, representation theory, matrix groups

14th Innovations in Theoretical Computer Science Conference (ITCS 2023).
Editor: Yael Tauman Kalai; Article No. 19; pp. 19:1–19:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

3. Better constant for Strassen-Winograd

Matrix Multiplication, a Little Faster

ELAYE KARSTADT and ODED SCHWARTZ, The Hebrew University of Jerusalem

Strassen's algorithm (1969) was the first sub-cubic matrix multiplication algorithm. Winograd (1971) improved the leading coefficient of its complexity from 6 to 7. There have been many subsequent asymptotic improvements. Unfortunately, most of these have the disadvantage of very large, often gigantic, hidden constants. Consequently, Strassen-Winograd's $O(n^{\log_2 7})$ algorithm often outperforms other fast matrix multiplication algorithms for all feasible matrix dimensions. The leading coefficient of Strassen-Winograd's algorithm has been generally believed to be optimal for matrix multiplication algorithms with a 2×2 base case, due to the lower bounds by Probert (1976) and Bshouty (1995).

Surprisingly, we obtain a faster matrix multiplication algorithm, with the same base case size and asymptotic complexity as Strassen-Winograd's algorithm, but with the leading coefficient reduced from 6 to 5. To this end, we extend Bodrato's (2010) method for matrix squaring, and transform matrices to an alternative basis. We also prove a generalization of Probert's and Bshouty's lower bounds that holds under change of basis, showing that for matrix multiplication algorithms with a 2×2 base case, the leading coefficient of our algorithm cannot be further reduced, and is therefore optimal. We apply our method to other fast matrix multiplication algorithms, improving their arithmetic and communication costs by significant constant factors.

CCS Concepts: • **Mathematics of computing → Computations on matrices**; • **Computing methodologies → Linear algebra algorithms**;

Additional Key Words and Phrases: Fast matrix multiplication, bilinear algorithms

ACM Reference format:
Elaye Karstadt and Oded Schwartz. 2020. Matrix Multiplication, a Little Faster. *J. ACM* 67, 1, Article 1 (January 2020), 31 pages.
https://doi.org/10.1145/3364504

3. Better constant for Strassen-Winograd

Table 1. 2×2 Fast Matrix Multiplication Algorithms[2]

Algorithm	Additions	Arithmetic Complexity	IO-Complexity
Strassen [58]	18	$7n^{\log_2 7} - 6n^2$	$12 \cdot M \left(\sqrt{3} \cdot \frac{n}{\sqrt{M}}\right)^{\log_2 7} - 18n^2$
Strassen-Winograd [61]	15	$6n^{\log_2 7} - 5n^2$	$10.5 \cdot M \left(\sqrt{3} \cdot \frac{n}{\sqrt{M}}\right)^{\log_2 7} - 15n^2$
Ours	12	$5n^{\log_2 7} - 4n^2 + 3n^2 \log_2 n$	$9 \cdot M \left(\sqrt{3} \cdot \frac{n}{\sqrt{M}}\right)^{\log_2 7} + 9n^2 \cdot \log_2 \left(\sqrt{2} \cdot \frac{n}{\sqrt{M}}\right)$

- This seems to contradict Probert's optimality result of the constant 15
- Can you see what happens here?