Dense Linear Algebra
—from Gauss to Strassen—

Alin Bostan

MPRI C-2-22
September 27, 2022

M2 Internship Projects

- *Algorithms for the parametrization of plane curves* (Bostan)
- *Algorithms for solving q-difference equations* (Bostan)
- *Multipoint power series expansions* (Lairez)
- *Univariate matrices for faster polynomial system solving* (Neiger)
- *Multi-level algebraic structures and faster guessing* (Neiger)

$\rightsquigarrow$ detailed descriptions available soon; but 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", which 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. 10, 13/12) – algos based on *linear recurrences*: $O(n^3) \to \tilde{O}(n^2)$
- structured (Vandermonde, Sylvester, Toeplitz, Hankel,. . . , lect. 10, 13/12) – algorithms based on the theory of the *displacement rank*: $O(n^3) \to \tilde{O}(n)$

Applications

Linear algebra is ubiquitous:

- computations with dense power series (lecture 4, 11/10)
- computations with D-finite power series (lecture 5, 18/10)
- computation of terms of a recurrent sequence (lecture 6, 25/10)
- Hermite-Padé approximants (lecture 9, 06/12)
- polynomial factorization over finite fields (lect. 11, 03/01)
- symbolic integration & summation (lect. 13, 24/01)

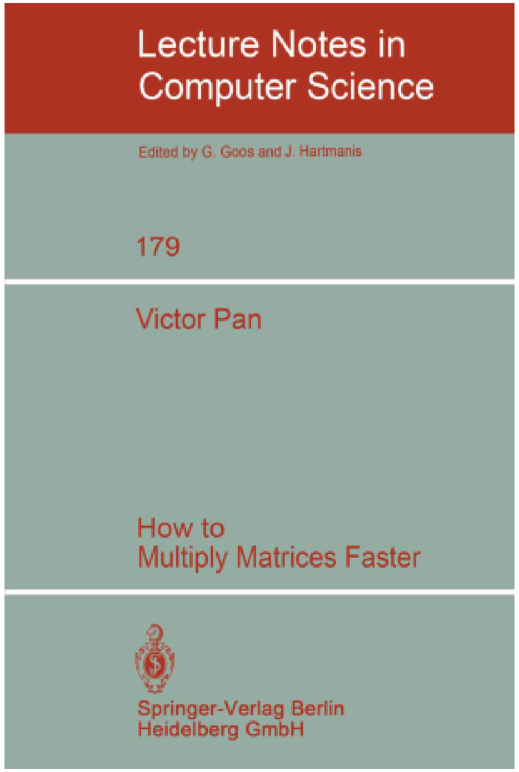
$\vdots$

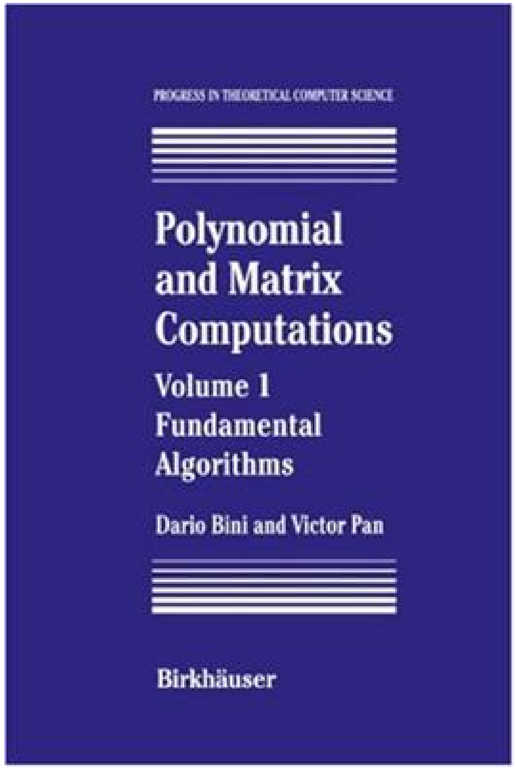
- integer factorization relies on (sparse) linear algebra over $\mathbb{F}_2$
- PageRank webpage ranking system relies on (sparse) linear algebra
- crypto-analysis: discrete logs (sparse)

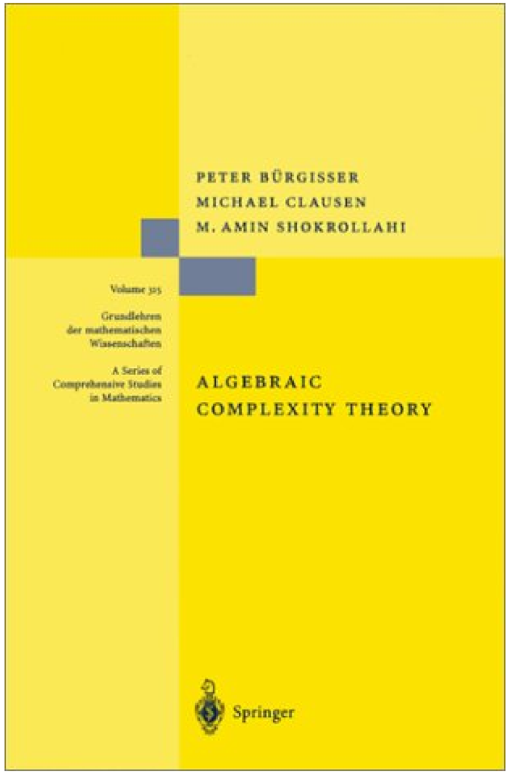
Matrix multiplication

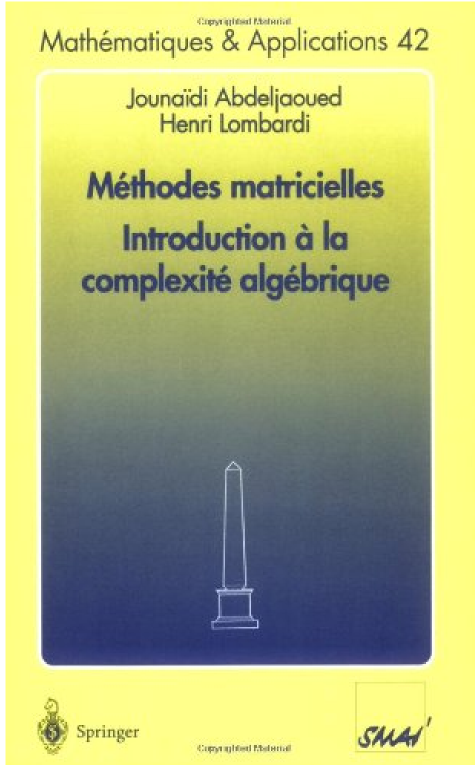
Matrix multiplication

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
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]

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

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:

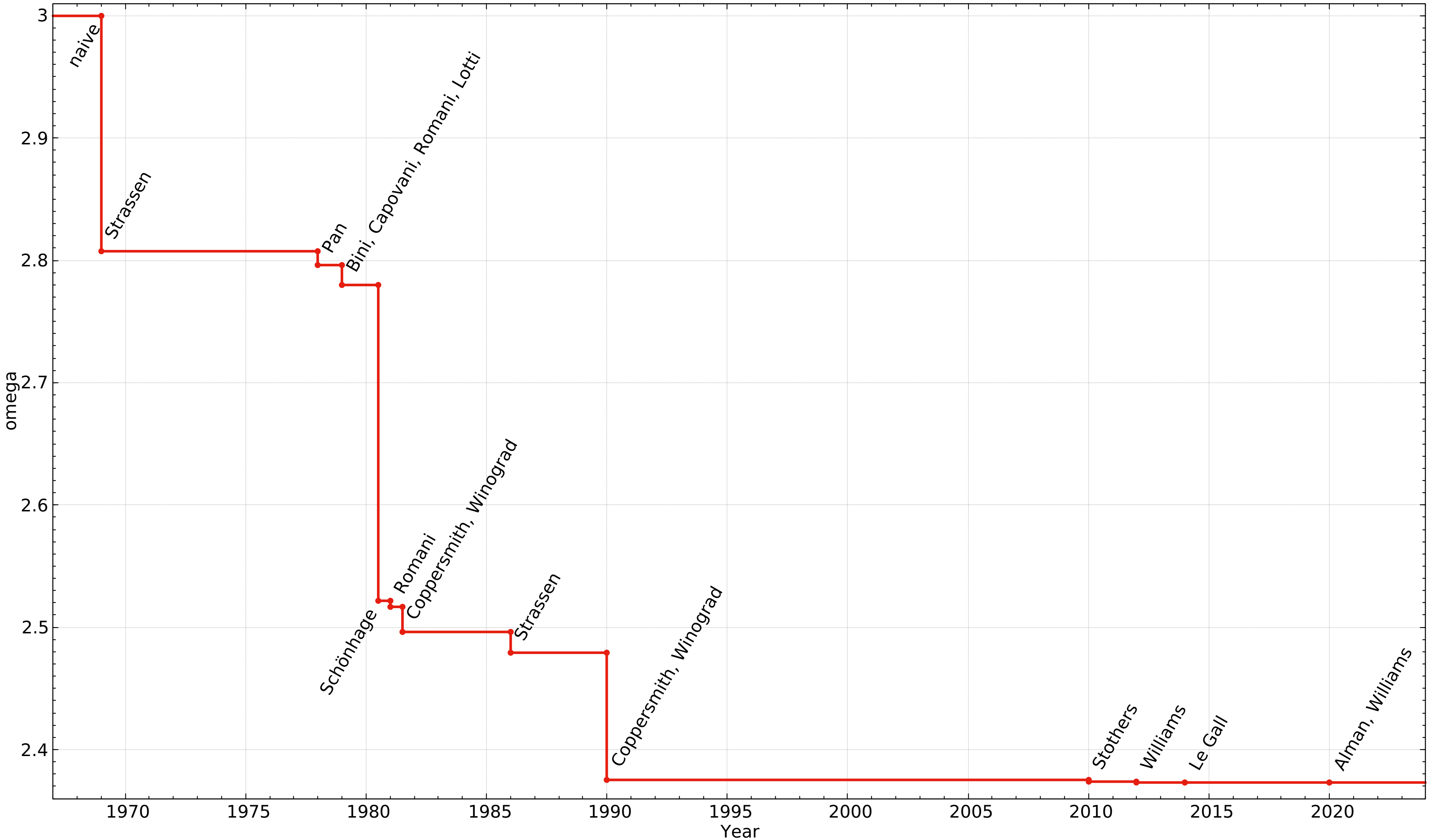
- one can multiply any A, B in $\mathcal{M}_n(\mathbb{K})$ using at most $\mathsf{MM}(n)$ ops. in $\mathbb{K}$
- MM satisfies $\mathsf{MM}(n) \leq \mathsf{MM}(2n)/4$ for all $n \in \mathbb{N}$
- $n \mapsto \mathsf{MM}(n)/n^2$ is increasing

▷ $\omega \in [2, 2.38]$

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

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

▷ Big open problem: Is $\omega = 2$?


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

Mathematics > Algebraic Geometry

[Submitted on 23 Sep 2020]

Bad and good news for Strassen's laser method: Border rank of the 3x3 permanent and strict submultiplicativity

Austin Conner, Hang Huang, J. M. Landsberg

We determine the border ranks of tensors that could potentially advance the known upper bound for the exponent ω of matrix multiplication. The Kronecker square of the small $q=2$ Coppersmith–Winograd tensor equals the 3×3 permanent, and could potentially be used to show $\omega=2$. We prove the negative result for complexity theory that its border rank is 16, resolving a longstanding problem. Regarding its $q=4$ skew cousin in $C^5\otimes C^5\otimes C^5$, which could potentially be used to prove $\omega\leq 2.11$, we show the border rank of its Kronecker square is at most 42, a remarkable sub-multiplicativity result, as the square of its border rank is 64. We also determine moduli spaces $\underline{VSP}$ for the small Coppersmith–Winograd tensors.

Subjects: **Algebraic Geometry (math.AG)**; Computational Complexity (cs.CC)

MSC classes: 68Q15, 15A69, 14L35

Cite as: arXiv:2009.11391 **[math.AG]**

(or arXiv:2009.11391v1 **[math.AG]** for this version)

Bibliographic data

[Enable Bibex (What is Bibex?)]

Submission history

From: J. M. Landsberg [view email]

[v1] Wed, 23 Sep 2020 21:40:15 UTC (41 KB)

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 $1 \le i \le n \longrightarrow$ $nk = \frac{n^2}{2}$ $\otimes$ and $n(k-1) = \frac{n^2}{2} - n$ $\oplus$
- precompute $\sigma(c_j)$ for $1 \le j \le n \longrightarrow$ $nk = \frac{n^2}{2}$ $\otimes$ 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 $n^2(n+k+1) = \frac{3n^3}{2} + n^2$ $\oplus$

$\triangleright$ 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 did 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 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.

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

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:

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

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

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

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

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

▷ 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 $n \approx 500$

▷ 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 two complex, and that implies very big constants in the $O(\cdot)$ $\longrightarrow$ interesting for sizes over millions or billions

▷ magma is one of the few CAS that uses fast matrix multiplication

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.

$\longrightarrow$ "Gauss 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:

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

▷ 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: $\mathsf{I}(n) = 2 \cdot \mathsf{I}\left(\frac{n}{2}\right) + O(\mathsf{MM}(n)) \implies \mathsf{I}(n) = O(\mathsf{MM}(n))$

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 $T := Z^{-1}$ and (recursively) $d_Z := \det(Z)$.
4. Recover the determinant $\det(M)$ as $d_A \cdot d_Z$.

DAC Theorem:

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

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 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 $\mathcal{B} := \{v, Av, \dots, A^{n-1}v\}$ 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 $\chi_A(x) = x^n - p_{n-1}x^{n-1} - \cdots - p_1 x - p_0$, then the matrix C of $f : w \mapsto Aw$ w.r.t. $\mathcal{B}$ is companion, with last column $[p_0, \dots, p_{n-1}]^T$.

Algorithm.

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

Computation of 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.

Two exercises for next Tuesday

(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. 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

1. 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?

2. 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 $\{m \times n \times p : r\}$.*

2. Multiplication in small sizes

Algorithm	tensor rank	naive tensor rank	construction	ratio	complexity	stabilizer
⟨2x2x2:7⟩	7	8	Strassen 1969 [DOI]	.8750000000	2.807354922	$(S_3)\rtimes S_3$
⟨3x3x3:23⟩	23	27	Laderman 1976 [DOI]	.8518518518	2.854049830	$(C_2\times C_2)\rtimes S_3$
⟨4x4x4:49⟩	49	64	⟨2x2x2:7⟩ ⊗ ⟨2x2x2:7⟩	.7656250000	2.807354922	$S_3\times S_3$
⟨5x5x5:98⟩	98	125	⟨5x5x2:40⟩ + ⟨5x5x3:58⟩	.7840000000	2.848800468	C_1
⟨6x6x6:160⟩	160	216	⟨3x3x6:40⟩ ⊗ ⟨2x2x1:4⟩	.7407407407	2.832508438	$(C_2\times C_2)\rtimes S_4\times C_2\times C_2$
⟨7x7x7:250⟩	250	343	⟨4x4x4:49⟩ + 3 ⟨3x3x4:29⟩ + 3 ⟨3x4x4:38⟩	.7288629738	2.837469613	C_1
⟨8x8x8:343⟩	343	512	⟨2x2x2:7⟩ ⊗ ⟨4x4x4:49⟩	.6699218750	2.807354922	$S_3\times S_3\times S_3$
⟨9x9x9:511⟩	511	729	⟨9x9x1:81⟩ + ⟨9x9x8:430⟩	.7009602195	2.838294116	C_1
⟨10x10x10:686⟩	686	1000	⟨2x2x2:7⟩ ⊗ ⟨5x5x5:98⟩	.6860000000	2.836324116	S_3
⟨11x11x11:919⟩	919	1331	⟨6x6x6:160⟩ + 3 ⟨5x5x6:116⟩ + 3 ⟨5x6x6:137⟩	.6904583020	2.845531329	C_1
⟨12x12x12:1040⟩	1040	1728	⟨2x4x4:26⟩ ⊗ ⟨6x3x3:40⟩	.6018518518	2.795668800	$C_2\times(C_2\times C_2)\rtimes S_4$
⟨13x13x13:1443⟩	1443	2197	⟨6x7x7:215⟩ + 4 ⟨6x6x7:185⟩ + TA(⟨7x7x7⟩, ⟨7x7x7⟩)	.6568047337	2.836110404	C_1
⟨14x14x14:1720⟩	1720	2744	⟨7x7x7:250⟩ + 3 TA(⟨7x7x7⟩, ⟨7x7x7⟩)	.6268221574	2.823007854	C_1
⟨15x15x15:2088⟩	2088	3375	⟨3x3x5:36⟩ ⊗ ⟨5x5x3:58⟩	.6186666667	2.822681037	C_1
⟨16x16x16:2401⟩	2401	4096	⟨2x2x2:7⟩ ⊗ ⟨8x8x8:343⟩	.5861816406	2.807354922	$S_3\times S_3\times S_3\times S_3$

▷ Sedoglavic: online catalogue of 5426 fast matrix multiplication algorithms

3. Boolean matrix multiplication

Computer Science > Data Structures and Algorithms

[Submitted on 27 Sep 2021]

Improved algorithms for Boolean matrix multiplication via opportunistic matrix multiplication

David G. Harris

Karppa & Kaski (2019) proposed a novel type of "broken" or "opportunistic" multiplication algorithm, based on a variant of Strassen's alkgorithm, and used this to develop new algorithms for Boolean matrix multiplication, among other tasks. For instance, their algorithm can compute Boolean matrix multiplication in $O(n^{\log_2(6+6/7)} \log n) = O(n^{2.778})$ time. While faster matrix multiplication algorithms exist asymptotically, in practice most such algorithms are infeasible for practical problems. Their opportunistic algorithm is a slight variant of Strassen's algorithm, so hopefully it should yield practical as well as asymptotic improvements to it.
In this note, we describe a more efficient way to use the broken matrix multiplication algorithm to solve Boolean matrix multiplication. In brief, instead of running multiple iterations of the broken algorithm on the original input matrix, we form a new larger matrix by sampling and run a single iteration of the broken algorithm on it. The resulting algorithm has runtime $O(n^{\frac{3\log 6}{\log 7}} (\log n)^{\frac{\log 6}{\log 7}}) \leq O(n^{2.763})$. We also describe an extension to witnessing Boolean matrix multiplication, as well as extensions to non-square matrices.
The new algorithm is simple and has reasonable constants. We hope it may lead to improved practical algorithms

Subjects: **Data Structures and Algorithms (cs.DS)**
Cite as: arXiv:2109.13335 **[cs.DS]**
(or arXiv:2109.13335v1 **[cs.DS]** for this version)

Submission history

From: David Harris [view email]
[v1] Mon, 27 Sep 2021 20:26:20 UTC (17 KB)

4. Multiplying pairs of 2×2 matrices

Mathematics > Algebraic Geometry

[Submitted on 22 Sep 2022]

Tensor rank of the direct sum of two copies of 2×2 matrix multiplication tensor is 14

Filip Rupniewski

The article is concerned with the problem of the additivity of the tensor rank. That is for two independent tensors we study when the rank of their direct sum is equal to the sum of their individual ranks. The statement saying that additivity always holds was previously known as Strassen's conjecture (1969) until Shitov proposed counterexamples (2019). They are not explicit and only known to exist asymptotically for very large tensor spaces. In this article, we show that for some small three-way tensors the additivity holds. For instance, we give a proof that another conjecture stated by Strassen (1969) is true. It is the particular case of the general Strassen's additivity conjecture where tensors are a pair of 2×2 matrix multiplication tensors. In addition, we show that the Alexeev-Forbes-Tsimerman substitution method preserves the structure of a direct sum of tensors.

Comments:	24 pages, 4 figures. arXiv admin note: text overlap with arXiv:1902.06582
Subjects:	**Algebraic Geometry (math.AG)**
MSC classes:	`15A69 (Primary), 14N07, 15A03 (Secondary)`
Cite as:	arXiv:2209.11040 **[math.AG]**
	(or arXiv:2209.11040v1 **[math.AG]** for this version)
	https://doi.org/10.48550/arXiv.2209.11040

Submission history

From: Filip Rupniewski [view email]
[v1] Thu, 22 Sep 2022 14:35:19 UTC (2,099 KB)