



HAL
open science

Strassen's algorithm is not optimally accurate

Jean-Guillaume Dumas, Clément Pernet, Alexandre Sedoglavic

► **To cite this version:**

Jean-Guillaume Dumas, Clément Pernet, Alexandre Sedoglavic. Strassen's algorithm is not optimally accurate. Proceedings of the 49th International Symposium on Symbolic and Algebraic Computation (ISSAC'24), ACM SIGSAM, Jul 2024, Raleigh, NC, United States. 10.1145/3666000.3669697 . hal-04441653v2

HAL Id: hal-04441653

<https://hal.science/hal-04441653v2>

Submitted on 27 Jun 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Strassen's algorithm is not optimally accurate

Jean-Guillaume Dumas
 Université Grenoble Alpes
 UMR CNRS 5224 LJK
 38058 Grenoble, France

Clément Pernet
 Univ. Grenoble Alpes, Grenoble INP
 UMR CNRS 5224 LJK
 38058 Grenoble, France

Alexandre Sedoglavic
 Université de Lille
 UMR CNRS 9189 CRISTAL
 59650 Villeneuve d'Ascq, France

ABSTRACT

We propose a non-commutative algorithm for multiplying 2×2 -matrices using 7 coefficient products. This algorithm reaches simultaneously a better accuracy in practice compared to previously known such fast 2×2 algorithms and a time complexity bound with the best currently known leading term (obtained via alternative basis sparsification). To build this algorithm, we consider matrix and tensor norm bounds governing the stability and accuracy of numerical matrix multiplication. First, we reduce those bounds by minimizing a growth factor along the unique orbit of Strassen's 2×2 -matrix multiplication tensor decomposition. Second, we develop heuristics that minimize the number of operations required to realize a bilinear formula, while further improving its accuracy. Third, we perform an alternative basis sparsification that improves on the time complexity constant and mostly preserves the overall accuracy.

1 INTRODUCTION

The first non-commutative algorithm for multiplying 2×2 -matrices using 7 coefficient products was discovered by Strassen [22]. It was subsequently proven that all such algorithms with 7 multiplications all lie in a single isotropy orbit on Strassen's bilinear tensor decomposition [14]. We here study the numerical accuracy of the recursive application of these 2×2 algorithms over the reals.

We first propose a unified accuracy analysis of such recursive algorithms, generalizing some and improving on other state-of-the-art bounds [1, 2, 4, 7, 8, 10]. Following the approach of [4], we then seek to optimize the growth factor, a parameter governing the accuracy in these bounds, over Strassen's orbit. Since the max-norm, producing the sharpest bounds, precludes smooth optimization, we relax the problem to optimizing a weaker growth factor in the Frobenius norm, which will later demonstrate to better reflect the practical accuracy observed in our experiments.

The most efficient variants are then obtained from these bilinear formulas by minimizing the number of operations required to realize them. Our heuristics for this make use of common sub-expression eliminations with rational coefficients, potential factorization via the kernel of the matrices defining the considered bilinear operators, as well as Tellegen's transposition principle.

While preserving the complexity bound exponent of Strassen's algorithm, $n^{\log_2 7}$, those algorithms require slightly more operations, thus worsening the constant factor of the leading term. We therefore finally propose further variants obtained by an alternative basis sparsification, similar to those introduced in [3, 18]. In fine, again thanks to a minimization of the number of operations required to realize them, we obtain variants having a time complexity bound with the best currently known leading term, that simultaneously

improve on the accuracy (i.e. mostly preserving in practice the numerical accuracy with or without alternative basis sparsification).

Our c++ tools for the minimization of the number of operations are gathered in the `PLINOPT` library [11]. We also forked the Matlab framework of [8] in [12] to experiment our implementations of the resulting fast and accurate 2×2 matrix multiplication algorithms.

Section 2 presents the symmetries of matrix multiplication tensors that we will use. In Section 3 we propose the unified error bounds on bilinear operators and matrix multiplication algorithms, highlighting how the growth factor parameter governs accuracy. On a relaxed growth factor in norm 2, we apply, in Section 4, a descent algorithm to reach some local minima and show in Section 5 that it lies within at most 2.6% of the optimal. Finally, Section 6 presents our minimization heuristics and the obtained matrix multiplication algorithms and their associated accuracy benchmark.

2 MATRIX PRODUCT SEEN AS TENSOR

We recall here the formalism of tensor decomposition allowing to present clearly the symmetries, later used to search for more numerically accurate fast matrix multiplication algorithms in Section 4. We start by briefly recalling tensorial representation of bilinear maps, through the example introduced by Strassen in [22] of fast 2×2 -matrix product, and we refer to [19] for this framework.

The product $C = A \cdot B$ of 2×2 matrices can be computed by Strassen algorithm using the following computations:

$$\begin{aligned} \rho_1 &\leftarrow a_{11}(b_{12} - b_{22}), & \rho_4 &\leftarrow (a_{12} - a_{22})(b_{21} + b_{22}), \\ \rho_2 &\leftarrow (a_{11} + a_{12})b_{22}, & \rho_5 &\leftarrow (a_{11} + a_{22})(b_{11} + b_{22}), \\ \rho_3 &\leftarrow (a_{21} + a_{22})b_{11}, & \rho_7 &\leftarrow (a_{21} - a_{11})(b_{11} + b_{12}), \\ \rho_6 &\leftarrow a_{22}(b_{21} - b_{11}), & \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} &= \begin{bmatrix} \rho_5 + \rho_4 - \rho_2 + \rho_6 & \rho_6 + \rho_3 \\ \rho_2 + \rho_1 & \rho_5 + \rho_7 + \rho_1 - \rho_3 \end{bmatrix}. \end{aligned} \quad (1)$$

This straight-line program (a.k.a. SLP) encodes the following bilinear map over a field \mathbb{K} with m, k, n equal to 2:

$$\begin{aligned} \beta_{\text{MM}}(A, B) : \mathbb{K}^{m \times k} \times \mathbb{K}^{k \times n} &\rightarrow \mathbb{K}^{m \times n}, \\ (A, B) &\mapsto A \cdot B. \end{aligned} \quad (2)$$

Indices m, k, n are kept in this section for the sake of clarity in order to distinguish easily the different spaces involved in the sequel.

Definition 1. The spaces $\mathbb{K}^{\times \times}$ can be endowed with the classical Frobenius inner product $\langle M, N \rangle = \text{Trace}(M^T \cdot N)$ that establishes an isomorphism between $\mathbb{K}^{\times \times}$ and its dual space $(\mathbb{K}^{\times \times})^*$.

Frobenius inner product combines matrix product (2) and the trilinear form $\text{Trace}(C^T \cdot A \cdot B)$ as follows:

$$\begin{aligned} S|_3 : \mathbb{K}^{m \times k} \times \mathbb{K}^{k \times n} \times (\mathbb{K}^{m \times n})^* &\rightarrow \mathbb{K}, \\ (A, B, C^T) &\mapsto \langle C, A \cdot B \rangle. \end{aligned} \quad (3)$$

As the space of trilinear forms is the canonical dual space of order three tensor products, Strassen algorithm (1) is encoded as the

This material is based on work supported in part by the Agence Nationale de la Recherche under grants ANR-15-IDEX-0002, ANR-21-CE39-0006, ANR-22-PECY-0010.

tensor decomposition \mathcal{S} of the matrix multiplication tensor in sum of seven rank-one tensors defined by the following relations:

$$\begin{aligned} \mathcal{S} &= \sum_{i=1}^7 M_i \otimes N_i \otimes O^i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ &+ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \end{aligned} \quad (4)$$

in $(\mathbb{K}^{m \times k})^* \otimes (\mathbb{K}^{k \times n})^* \otimes \mathbb{K}^{m \times n}$ with $m = k = n = 2$. In the above tensor decomposition, each summand is a *rank-one tensor* and its *tensor rank* is the number r of such element (7 there). Given Equation (4), multiplication formula (2) implemented by Eq. (1) is obtained using the third 2-contraction of the tensor $\mathcal{S} \otimes A \otimes B$ as defined in the following map:

$$\left((\mathbb{K}^{m \times k})^* \otimes (\mathbb{K}^{k \times n})^* \otimes \mathbb{K}^{m \times n} \right) \otimes (\mathbb{K}^{m \times k} \otimes \mathbb{K}^{k \times n}) \rightarrow \mathbb{K}^{m \times n}, \quad (5)$$

$$\left(\sum_{i=1}^r M_i \otimes N_i \otimes O^i \right) \otimes (A \otimes B) \mapsto \sum_{i=1}^r \langle M_i, A \rangle \langle N_i, B \rangle O^i.$$

Some formalisms are more adapted to the design of algorithms computing efficiently the matrix product (as shown in Section 6) than direct tensor decompositions. For example, a nice concise representation was introduced in [17]; it encodes the sum of rank-one tensors by three matrices as done for the Strassen tensor decomposition (4) in the following three matrices L_S, R_S and P_S :

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}^T. \quad (6)$$

Notation 2. Given an $m \times k$ -matrix A , we denote by A_i the i th row and by $\text{vec}A$ the row-major vectorization of this matrix, i.e. the vector v in \mathbb{R}^{mk} such that $v_{ik+j} = a_{i,j}$. We also denote by $\text{Mat}_{m,k}(v)$ the reciprocal operation, building an $m \times k$ matrix from an mk -dimensional vector. Thus, the i th line L_{S_i} (resp. R_{S_i}) of matrix L_S (resp. R_S) is the transposition of the row-major vectorization $\text{vec}M_i$ of the first (resp. second $\text{vec}N_i$) component of the i th triad in Equation (4) and the i th column of matrix P_S is the column-major vectorization $\text{vec}O^i$ of its third component.

Definition 3. This encoding of a tensor by three suitable matrices L, R, P is called an *HM representation* and is denoted by $[L; R; P]$.

Equation (11) presented in Section 3 shows that the HM representation allows constructing SLPS for the associated algorithms. We show in Section 6.2 that this could be done efficiently, e.g. using the kernel of L (resp. R) and Tellegen's transposition applied to P . Now we turn to symmetries of matrix product tensor decomposition. Remark that the matrix product is associated to $\text{Trace}(A \cdot B \cdot C)$ by Equation (3) and that, given invertible matrices U, V, W of suitable sizes and the classical trace properties, this trace is equal to:

$$\begin{aligned} \text{Trace}((A \cdot B \cdot C)^T) &= \text{Trace}(C \cdot A \cdot B) = \text{Trace}(B \cdot C \cdot A) \\ \text{and to } \text{Trace}(U^{-1} \cdot A \cdot V \cdot V^{-1} \cdot B \cdot W \cdot W^{-1} \cdot C \cdot U). \end{aligned} \quad (7)$$

These relations illustrate the next theorem and induce the isotropy action on matrix product tensor decomposition presented below:

Theorem 4 ([14, § 2.8]). The isotropy group of the $m \times m$ matrix multiplication tensor is the semidirect product $\text{psl}^\pm(\mathbb{K}^m)^{\times 3} \rtimes \mathfrak{S}_3$,

where psl stands for the group of matrices of determinant ± 1 and \mathfrak{S}_3 for the symmetric group on 3 elements.

Definition 5. Let g denotes $(U \times V \times W)$ in $\text{psl}^\pm(\mathbb{K}^m)^{\times 3}$ and \mathcal{T} a rank-one tensor $A \otimes B \otimes C$; the action $g \diamond \mathcal{T}$ of g on \mathcal{T} is the rank-one tensor $(U^{-T} \cdot A \cdot V^T) \otimes (V^{-T} \cdot B \cdot W^T) \otimes (W^{-T} \cdot C \cdot U^T)$. This action is extended by additivity to higher tensor rank tensors.

Given two isotropies g_1 defined by matrices $(U_1 \times V_1 \times W_1)$ and g_2 defined by matrices $(U_2 \times V_2 \times W_2)$ both in $\text{psl}^\pm(\mathbb{K}^m)^{\times 3}$, the composition $g_1 \circ g_2$ is given by $(U_1 \cdot U_2 \times V_1 \cdot V_2 \times W_1 \cdot W_2)$.

The isotropies action on an HM representation is a direct consequence of the above results and presented in the following lemma.

Lemma 6. Let g be $(U \times V \times W)$ in $\text{psl}^\pm(\mathbb{K}^m)^{\times 3}$ and $[L; R; P]$ be an HM representation of a matrix product tensor decomposition, the action $g \diamond [L; R; P]$ of g on $[L; R; P]$ is another HM representation of a matrix product tensor decomposition defined by:

$$[L \cdot (V^T \otimes U^{-1}); R \cdot (W^T \otimes V^{-1}); (U \otimes W^{-T}) \cdot P]. \quad (8)$$

Dealing with a tensor decomposition or with the associated HM representation is not strictly equivalent; In Definition 5 there is no need to care about the determinants of the matrices (U, V, W) while this fact is no more true for Equation (8) as (say) U acts on two different components.

The following theorem recalls that all 2×2 -matrix product algorithms with 7 coefficient multiplications are obtained by this single orbit of the action of isotropies on Strassen tensor decomposition:

Theorem 7 ([15, § 0.1]). The group $\text{psl}^\pm(\mathbb{K}^m)^{\times 3}$ acts transitively on the variety of fast algorithms multiplying 2×2 -matrices.

Thus, isotropy action on Strassen tensor decomposition may define other matrix product algorithm of same tensor rank but with potentially more interesting characteristics as shown in Section 4. We make explicit these properties in the following section.

3 BILINEAR OPERATOR ACCURACY BOUND

We will consider that any finite-dimensional real vector space \mathbb{U} is equipped with a norm $\|\cdot\|$ and denote by $\|\cdot\|_*$ the related dual norm; for $\phi : \mathbb{U} \rightarrow \mathbb{R}$, its norm $\|\phi\|_*$ is $\sup(|\phi(v)|, \|v\| \leq 1)$. For instance, the max-norm $\|\cdot\|_\infty$ and the one-norm $\|\cdot\|_1$ are dual one with the other, while the two-norm $\|\cdot\|_2$ is self-dual. We will also denote the Hamming weight $\#\{i | x_i \neq 0\}$ of x by $\|x\|_0$. The n -dimensional vector of coefficients x_1, \dots, x_n is denoted by $(x_i)_{i \in \{1..n\}}$ or more succinctly $(x_i)_i$ when the indexing is clear from the context. By extension, we denote $\|x\| \|y\|$ by $\|x; y\|$ and $\|L\| \|R\| \|P\|$ by $\|L; R; P\|$.

Lemma 8. For any matrix A in $\mathbb{R}^{m \times k}$ and any vectors x, y in \mathbb{R}^k the following inequalities hold:

$$\|x \cdot y\| \leq \|x\|_* \|y\|, \quad \|Ax\| \leq \|(\|A_i\|_*)_i\| \|x\|, \quad (9)$$

$$\|Ax\|_\infty \leq \max_{i=1..m} \left(\sum_{j=1}^k |a_{i,j}| \right) \|x\|_\infty \leq k \|A\|_\infty \|x\|_\infty. \quad (10)$$

Given an HM representation $[L; R; P]$ of a matrix multiplication tensor decomposition, one can retrieve the transpose of the multiplication formula (2) implemented by Eq. (1) using the Hadamard

Strassen's algorithm is not optimally accurate

product $A \odot B$ of matrices A and B with the following map:

$$\begin{aligned} \mathbb{K}^{m \times k} \times \mathbb{K}^{k \times n} &\rightarrow \mathbb{K}^{mn \times 1}, \\ (A, B) &\mapsto P^\top \cdot \left((L \cdot \text{vec} A) \odot (R \cdot \text{vec} B) \right). \end{aligned} \quad (11)$$

Hence, we express there a bilinear operator $\beta : \mathbb{R}^e \times \mathbb{R}^f \rightarrow \mathbb{R}^g$ represented by its HM representation $[L; R; P]$ in $\mathbb{R}^{r \times e} \times \mathbb{R}^{r \times f} \times \mathbb{R}^{r \times g}$ as $\beta(u, v) = \sum_{i=1}^r (L_i \cdot u)(R_i \cdot v)(P^\top)_i$. When this operator encodes an $m \times k$ by $k \times n$ matrix multiplication formula, we will thus denote it by β_{MM} and we will have $e = mk, f = kn, g = mn$. We also consider recursive applications of such operators defined as:

$$\begin{aligned} \beta^{(\ell)} : \mathbb{R}^{e_0 e^\ell} \times \mathbb{R}^{f_0 f^\ell} &\rightarrow \mathbb{R}^{g_0 g^\ell}, \\ (u, v) &\mapsto \sum_{i=1}^r \beta^{(\ell-1)}(L_i \cdot u, R_i \cdot v)(P^\top)_i \end{aligned} \quad (12)$$

and $\beta^0 : \mathbb{R}^{e_0} \times \mathbb{R}^{f_0} \rightarrow \mathbb{R}^{g_0}$, a bilinear operator which we will assume to be bounded: $\|\beta^{(0)}(u, v)\| \leq \gamma_0 \|u\| \|v\|$ for all (u, v) in $\mathbb{R}^{e_0} \times \mathbb{R}^{f_0}$. For convenience, we will define the dimensions G as $g_0 g^\ell$ and K as $k_0 k^\ell$. Recall that $(P^\top)_i$ is the i th column of P and remark that the expression $L_i \cdot u$ is an abuse of notation for the operation where each coefficient $l_{i,j}$ of L multiplies a block of $e_0 e^{\ell-1}$ contiguous coefficients of u , namely: $L_i \cdot u = (L_i \text{Mat}_{e, e_0 e^{\ell-1}}(u))^\top$. We will consider the floating point arithmetic in the standard model of [16]: \widehat{x} denotes the computed value for an expression x such that $a \text{ op } b = (a \text{ op } b)(1 + \delta)$ for $\text{op} = +, -, \times, /$ where δ is the unit round off such that $|\delta| \leq \varepsilon$, except when $a \text{ op } b$ is 0 where δ is -1 .

We recall in the following Lemma some classical inequalities:

Lemma 9 (see [8, Eq. (3.5)] and [16, Eq. (4.4)]). For any vectors u and v in \mathbb{R}^n the following inequalities hold:

$$|\widehat{u \cdot v} - u \cdot v| \leq \|u\|_0 \|u\|_* \|v\| \varepsilon + O(\varepsilon^2), \quad (13)$$

$$\left| \sum_{i=1}^n \widehat{u_i} - \sum_{i=1}^n u_i \right| \leq (n-1) \left(\sum_{i=1}^n |u_i| \right) \varepsilon + O(\varepsilon^2). \quad (14)$$

We define now the growth factor used in this work.

Definition 10. The growth factor γ of the formula $[L; R; P]$ computing the bilinear form β is defined by $\max_{j=1 \dots g} \sum_{i=1}^r \|L_i\|_* \|R_i\|_* |p_{i,j}|$.

The growth factor not only bounds the values of bilinear operators, as show in Lemma 11, but is also central in analyzing their forward numerical error, which will be the focus of Theorem 12.

Lemma 11. For any u, v with adequate dimensions, the following relations hold: $\|\beta(u, v)\| \leq \gamma \|u\| \|v\|$, $\|\beta^{(\ell)}(u, v)\| \leq \gamma_0 \gamma^\ell \|u\| \|v\|$ and $\|\beta_{MM}^{(\ell)}(u, v)\|_\infty \leq k_0 k^\ell \|u\|_\infty \|v\|_\infty$.

PROOF. Let D_j denotes $\text{Diag}_{i=1 \dots r}(p_{i,j})$ and c_j be the j th coefficient of $\beta(u, v)$. We have that $|c_j| \leq \|u^\top L^\top D_j R v\| \leq \|u^\top L D_j R\|_* \|v\|$, so that $|c_j| \leq \|L^\top D_j R\|_* \|u\| \|v\| \leq \left\| \sum_{i=1}^r (L_i \otimes R_i) p_{i,j} \right\|_* \|u\| \|v\|$ and $|c_j| \leq \left(\sum_{i=1}^r \|L_i\|_* \|R_i\|_* |p_{i,j}| \right) \|u\| \|v\|$. Finally, the last inequality follows from (9). \square

Theorem 12. Given any choice of norm $\|\cdot\|$, if G denotes $g_0 g^\ell$ and K denotes $k_0 k^\ell$, the error in computing $\beta^{(\ell)}$ is bounded as follows $\|\widehat{\beta^{(\ell)}}(u, v) - \beta^{(\ell)}(u, v)\|_\infty \leq \kappa \|u\| \|v\| \varepsilon + O(\varepsilon^2)$ where either

$$\kappa = \left(\frac{K}{k_0} \right)^{\log_k \gamma} \left(k_0^2 + \frac{Q_0 k_0 \gamma}{\gamma - k} \right) - \frac{Q_0 K \gamma}{\gamma - k} \quad (15)$$

when $\beta^{(\ell)}$ is an $M \times K$ by $K \times N$ matrix multiplication, or

$$\kappa = (G/g_0)^{\log_g \gamma} \gamma_0 \left(1 + (1 + \log_g(G/g_0)) Q_0 \right), \quad (16)$$

otherwise, and $Q_0 = \max_j (\|(P^\top)_j\|_0 + \max_i (\|L_i\|_0 + \|R_i\|_0) \mathbb{1}_{p_{i,j} \neq 0})$ as in [1, Definition 1].

PROOF. By induction, we will prove that the bound is of the form $\|\Delta_{\beta^{(\ell)}}\|_\infty = \|\widehat{\beta^{(\ell)}}(u, v) - \beta^{(\ell)}(u, v)\|_\infty \leq t_\ell \|u\| \|v\| \varepsilon + O(\varepsilon^2)$, clarifying in the process the value for t_ℓ . Consider the block c_j of $G/g = g_0 g^{\ell-1}$ consecutive output coefficients: $c_j = \sum_{i=1}^r H_i p_{i,j}$, where $H_i = \beta^{(\ell-1)}(L_i \cdot u, R_i \cdot v)$. Consider definitions $d_{i,j} = H_i p_{i,j}$ and $\Delta_{d_{i,j}} = \widehat{d_{i,j}} - d_{i,j}$. Then, by Lemma 9:

$$\|\widehat{c_j} - c_j\|_\infty \leq \left\| \sum_{i=1}^r \widehat{d_{i,j}} - \sum_{i=1}^r d_{i,j} \right\|_\infty + \left\| \sum_{i=1}^r \widehat{d_{i,j}} - \sum_{i=1}^r d_{i,j} \right\|_\infty, \quad (17)$$

$$\leq \sum_{i=1}^r \left\| \widehat{H_i p_{i,j}} \right\|_\infty (\|(P^\top)_j\|_0 - 1) \varepsilon + \sum_{i=1}^r \|\Delta_{d_{i,j}}\|_\infty + O(\varepsilon^2). \quad (18)$$

$$\|\Delta_{d_{i,j}}\|_\infty \leq \left\| p_{i,j} \widehat{H_i} - p_{i,j} H_i \right\|_\infty + \left\| p_{i,j} \widehat{H_i} - p_{i,j} H_i \right\|_\infty, \quad (19)$$

$$\leq |p_{i,j}| \|H_i\|_\infty \varepsilon + |p_{i,j}| \|\Delta_{H_i}\|_\infty + O(\varepsilon^2). \quad (20)$$

$\|\Delta_{H_i}\|_\infty$ is equal to $\|\widehat{\beta^{(\ell-1)}}(\widehat{L_i \cdot u}, \widehat{R_i \cdot v}) - \beta^{(\ell-1)}(L_i \cdot u, R_i \cdot v)\|_\infty$ by bilinearity of $\beta^{(\ell-1)}$ and bounded by:

$$\left\| \Delta_{\beta^{(\ell-1)}} \right\|_\infty + \left\| \beta^{(\ell-1)}(\Delta_L, R_i \cdot v) \right\|_\infty + \left\| \beta^{(\ell-1)}(L_i \cdot u, \Delta_R) \right\|_\infty. \quad (21)$$

By Lemma 9 and the induction hypothesis we have

$$\|\Delta_M\|_\infty \leq \|M_i\|_0 \|M_i\|_* \|u\| \varepsilon + O(\varepsilon^2) \text{ with } M \in \{L, R\}, \quad (22)$$

$$\left\| \Delta_{\beta^{(\ell-1)}} \right\|_\infty \leq t_{\ell-1} \|L_i \cdot u + \Delta_L\| \|R_i \cdot v + \Delta_R\| \varepsilon + O(\varepsilon^2), \quad (23)$$

$$\leq c_{\ell-1} \|L_i\|_* \|u\| \|R_i\|_* \|v\| \varepsilon + O(\varepsilon^2). \quad (24)$$

By Lemma 11, the following inequality holds

$$\left\| \beta^{(\ell-1)}(\Delta_L, R_i \cdot v) \right\|_\infty \leq \Theta_0 \Theta^{\ell-1} \|\Delta_L\|_\infty \|R_i\|_* \|v\| \quad (25)$$

for $(\Theta, \Theta_0) = (k, k_0)$ if $\beta = \beta_{MM}$ or (γ, γ_0) otherwise (where $\Theta_0 = \gamma_0$ comes from the current proof with $\ell = 1$ and $g_0 = 1$). Similarly,

$$\left\| \beta^{(\ell-1)}(L_i \cdot u, \Delta_R) \right\|_\infty \leq \Theta_0 \Theta^{\ell-1} \|\Delta_R\|_\infty \|L_i\|_* \|u\|. \quad (26)$$

Gathering Eqs. (18), (20) to (22) and (24) to (26) we deduce that

$$\begin{aligned} \|\widehat{c_j} - c_j\|_\infty &\leq \sum_{i=1}^r \left(\Theta_0 \Theta^{\ell-1} (\|L_i\|_0 + \|R_i\|_0 + \|(P^\top)_j\|_0) + t_{\ell-1} \right) \\ &\quad \times \|L_i\|_* \|R_i\|_* |p_{i,j}| \|u\| \|v\| \varepsilon + O(\varepsilon^2), \end{aligned} \quad (27)$$

and thus that $\|\widehat{c_j} - c_j\|_\infty \leq (\Theta_0 \Theta^{\ell-1} Q_0 + t_{\ell-1}) \gamma \|u\| \|v\| \varepsilon + O(\varepsilon^2)$. As in [16], we deduce that t_ℓ must then satisfy:

$$\begin{cases} t_\ell &= (\Theta_0 \Theta^{\ell-1} Q_0 + t_{\ell-1}) \gamma & \text{for } \ell > 0, \\ t_0 &= k_0^2 & \text{for matrix product,} \\ t_0 &= (1 + Q_0) \gamma_0 & \text{otherwise.} \end{cases} \quad (28)$$

This recurrence relation solves into $t_\ell = \gamma^\ell t_0 + Q_0 \Theta_0 \Theta^\ell \sum_{i=1}^{\ell-1} (\gamma/\Theta)^i$.

In the case of a matrix multiplication operator, t_ℓ is equal to:

$$\gamma^\ell k_0^2 + Q_0 k_0 \gamma \frac{\gamma^\ell - k^\ell}{\gamma - k} = \left(\frac{K}{k_0} \right)^{\log_k \gamma} \left(k_0^2 + \frac{Q_0 \gamma}{\gamma - k} k_0 \right) - \frac{Q_0 \gamma}{\gamma - k} K. \quad (29)$$

	Formula	Applies to	norm	Q		Winograd		Strassen		Equation (38)		Equation (36)	
				Q	γ	Q	γ	Q	γ	Q	γ	Q	γ
Brent [7]	(15)	Strassen only	∞	NA	$\gamma_{1,1,\infty}$			3.67	12				
BL [4] DDHK [10]	(16)	any MM alg.	∞	Q_0^1	$\gamma_{0,1,\infty}^2$	9	18	7	12	9.59	40	9.81	98.54
Higham [16]	(15)	S. & W. only	∞	NA	$\gamma_{1,1,\infty}$	4.94	18	3.83	12				
Ballard et al. [1]	(16)	any MM alg.	∞	Q_0	$\gamma_{1,1,\infty}$	10	18	8	12	12	13	15	17.48
Dai, Lim [8]	[8, Th 3.3]	$\ell = 1$, any alg.	2	$m+n+r$	$\gamma_{2,1}$	15	17.86	15	14.83	15	12.21	15	12.07
			∞	$m+n+r$	$\gamma_{1,\infty,1}$	15	27	15	20	15	22	15	25.14
Here	(15)	any MM alg.	2	Q_0	$\gamma_{2,1,\infty}$	10	8	8	6.83	12	6.05	15	5.97
Here	(16)	any alg.	∞	Q_0	$\gamma_{1,1,\infty}$	10	18	8	12	12	13	15	17.48
Here	(16)	any alg.	∞	Q_0	$\gamma_{1,1,\infty}$	10	18	8	12	12	13	15	17.48

Table 1: Comparing accuracy formulas for recursive bilinear matrix multiplication operators in the form of Theorem 12.

In the general case, the value of t_ℓ becomes: $(1 + (1 + \ell)Q_0)\gamma_0\gamma^\ell$, that is equal to $(G/g_0)^{\log_g \gamma} \gamma_0 (1 + (1 + \log_g(G/g_0))Q_0)$. \square

Theorem 12 generalizes or improves on previous similar results in [1, 7, 8, 10, 16]. In fact, [8] considers a single recursive level without base case; [7, 16] have tight bounds but only for Strassen and Winograd’s algorithms in max-norm; lastly [1, 10] has an additional logarithmic factor likely due to a looser bound on each $\|\beta^{(\ell-1)}\|_\infty$, not exploiting the fact that they are matrix products.

Even though the choice of the max-norm produces the tightest bounds in Theorem 12, as in most previous works, the bounds are stated there for any choice of norm, as in [8]. Alternative norms, such as the 2-norm, may give growth factor expressions more amenable to optimizations, as detailed in Section 4.

Table 1 compares the various existing bounds on numerical accuracy of matrix multiplication algorithms. They depend on the following choices made on the norms to define the growth factor γ :

$$\begin{aligned} \gamma_{0,1,\infty} &= \left\| \left(\left\| \left(\|L_i; R_i\|_0 |p_{i,k}| \right)_{i=1}^r \right\|_k \right)_{k=1}^m \right\|_\infty \|L\|_\infty \|R\|_\infty \|P\|_\infty, \\ &= \left(\max_{k \in \{1 \dots mn\}} \sum_{i=1}^r \|L_i\|_0 \|R_i\|_0 |p_{i,k}| \right) \|L\|_\infty \|R\|_\infty \|P\|_\infty. \end{aligned} \quad (30)$$

$$\gamma_{2,1} = \left\| \left(\|L_i; R_i; P_i\|_2 \right)_{i=1}^r \right\|_1 = \sum_{i=1}^r \|L_i\|_2 \|R_i\|_2 \|P_i\|_2. \quad (31)$$

$$\begin{aligned} \gamma_{q,1,\infty} &= \left\| \left(\left\| \left(\|L_i; R_i\|_q |p_{i,k}| \right)_{i=1}^r \right\|_k \right)_{k=1}^m \right\|_\infty, \\ &= \max_{k \in \{1 \dots mn\}} \sum_{i=1}^r \|L_i\|_q \|R_i\|_q |p_{i,k}| \text{ with } q \in \{1, 2\}. \end{aligned} \quad (32)$$

4 GROWTH FACTOR ALONG ORBITS

In the footsteps of [4], we aim to find an alternative 2×2 matrix product tensor decomposition, in the orbit of Strassen’s one, with improved accuracy, hence minimizing the growth factor. The use of the maxnorm induces an expression Eq. (32) for $\gamma_{1,1,\infty}$ poorly suited for optimizations. We will instead make two relaxations: first, using the 2-norm and second, as in [8], bounding $\gamma_{2,1,\infty}$ by $\gamma_{2,1}$:

$$\max_{k \in \{1 \dots mn\}} \sum_{i=1}^r \|L_i\|_2 \|R_i\|_2 |p_{i,k}| \leq \sum_{i=1}^r \|L_i\|_2 \|R_i\|_2 \|P_i\|_2. \quad (33)$$

¹[4, 10] reach an improved value of Q_0 by assuming all additions are performed following a balanced tree, instead of a worst case estimate as done in all other formulas.

²We applied the same $\gamma_{0,1,\infty}$ for [4], as it seems to be missing a dependency in the magnitude of the coefficients in L, R, P, which was fixed in [10].

Theorem 7 shows that all fast 2×2 matrix product algorithms are in the same orbit under isotropies action introduced in Definition 5. While the tensor rank is invariant under this action, the growth factor is generally not. As its definition is based on Frobenius norm, some isotropies leave it invariant as stated in the following lemma:

Lemma 13. The growth factor $\gamma_{2,1}$ is invariant under the action of the semidirect product $\mathfrak{so}^\pm(\mathbb{K}^n)^{\times 3} \rtimes \mathfrak{S}_3$ induced by the special orthogonal group and the permutation group \mathfrak{S}_3 .

PROOF. By Definition 1, Frobenius norms are invariant under orthogonal transformations and so is $\gamma_{2,1}$ by Eq. (31). Lemma 13 is then derived from Equations (7) and (8). \square

As it is useless to consider isotropies leaving the growth factor invariant, we limit our search to isotropies of the following form:

Lemma 14. The action of $(\mathfrak{h} \times \mathfrak{p})^{\times 3}$ determines the growth factor $\gamma_{2,1}$ for $\mathfrak{h} = \left\{ H_\rho = \begin{bmatrix} \rho & 0 \\ 0 & 1/\rho \end{bmatrix} \mid \rho > 0 \right\}$ and $\mathfrak{p} = \left\{ P_\xi = \begin{bmatrix} 1 & \xi \\ 0 & 1 \end{bmatrix} \mid \xi \in \mathbb{R} \right\}$.

PROOF. Eq. (8) shows that the product of any action, say U, by a non-zero scalar affects the growth factor once in U and once, inverted, in U^{-1} , as norms are absolutely homogeneous. Thus, it is sufficient to consider matrices with determinant 1. Lemma 13 states that orthogonal matrices do not have any effect. From the QR decomposition of any invertible matrices, there remains just the $(\mathfrak{h} \times \mathfrak{p})$ part of $\mathfrak{psl}^\pm(\mathbb{R}^2)$ ’s Iwasawa decomposition in Theorem 7. \square

We should study the action of $(\mathfrak{h} \times \mathfrak{p})^{\times 3}$ on Strassen tensor decomposition in order to find variants with the smaller possible $\gamma_{2,1}$. Unfortunately, a direct definitive result for this question seems to be out of reach, and we present several ersatzes. First, we perform numerical minimization on $\gamma(g \diamond S)$ with a completely generic isotropy g in $\mathfrak{psl}^\pm(\mathbb{R}^2)^{\times 3}$ (involving 6 indeterminates by Lemma 14); this experiment suggests that a suitable isotropy to reach a fast matrix product tensor decomposition with minimal $\gamma_{2,1}$ could be of the form $(U \times U \times U)$ (involving only 2 indeterminates). The following proposition states precisely this possibility (its proof is a simple second partial derivative test presented in Appendix A.1).

Proposition 15. Consider the matrices $U(\rho, \xi) = H_\rho \cdot P_\xi$ and the isotropies $g_{\rho, \xi}$ defined by $U(\rho, \xi)^{\times 3}$. The minimal value on the orbit $g_{\rho, \xi} \diamond S$ of the growth factor $\gamma_{2,1}(g_{\rho, \xi} \diamond S)$ is reached at the point $(\rho, \xi) = (\sqrt[4]{4/3}, -1/2)$ and equal to $4/\sqrt{2} + 16/\sqrt{3} > 12.06603$.

The algorithm corresponding to the point (ρ, ξ) with minimal $\gamma_{2,1}$ on this restricted orbit is given in Eq. (36). We gather in Table 2 values for $\gamma_{2,1}$ of some matrix product tensor decompositions, together with the result obtained in Proposition 15. In Section 6, we compare the implementation of algorithms associated to these tensor decompositions in order to confirm that their numerical accuracy is correlated to their respective $\gamma_{2,1}$ growth factor.

5 UPPER AND LOWER BOUNDS

We explore in this section some bounds on the norm of each component of an HM representation. By the multiplicativity of $L_{p,q}$ norms (even generalized to negative Hölder conjugates), this will always give alternative bounds on the error, a priori less accurate, but potentially easier to apprehend.

Lemma 16. For any HM representation \mathcal{H} , with matrices L, R, P in $\mathbb{K}^{r \times n}$, let $\gamma_{\mathcal{H}}$ be its $\gamma_{2,1}$ growth factor $\gamma_{2,1}(\mathcal{H})$, as in Eq. (31). Then for any strictly positive y and z , we have both:

$$\gamma_{\mathcal{H}} \leq \|\mathcal{H}\|_{2,3} \leq \|\mathcal{H}\|_F \quad \text{and} \quad (34)$$

$$\max \left\{ r^{1+3z} \|\mathcal{H}\|_{2,-\frac{1}{z}} ; \|L\|_{2,-\frac{1}{y}} \cdot \|R\|_{2,-\frac{1}{z}} \cdot \|P\|_{2,\frac{1}{1+y+z}} \right\} \leq \gamma_{\mathcal{H}}. \quad (35)$$

PROOF. Let a_i (resp. b_i, c_i) denotes $\|L_i\|_2$ (resp. $\|R_i\|_2, \|P_i\|_2$). The first right-hand side inequality is the classical Hölder's inequality $\|(a_i \cdot b_i \cdot c_i)_i\|_1 \leq \|(a_i)_i\|_3 \cdot \|(b_i)_i\|_3 \cdot \|(c_i)_i\|_3 = \|\mathcal{H}\|_{2,3}$ on a_i, b_i and c_i with the Hölder conjugates $\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$. The second right-hand side inequality is a direct application of the monotonicity of norms. Then, the left-hand side inequality is obtained by a reverse Hölder's inequality on the vectors a_i, b_i, c_i and 1, with the Hölder conjugates $\frac{1}{-1/z} + \frac{1}{-1/z} + \frac{1}{-1/z} + (1+3z) = 1$. We have indeed that the $(1+3z)$ -norm $\|(1)_i\|_{1/(1+3z)}$ is $(\sum_{i=1}^r 1^{1/(1+3z)})^{1+3z}$. Combined with the relation $\|\mathcal{H}\|_{2,-\frac{1}{z}} = \|(a_i)_i\|_{-\frac{1}{z}} \cdot \|(b_i)_i\|_{-\frac{1}{z}} \cdot \|(c_i)_i\|_{-\frac{1}{z}}$, this shows that the inequality $r^{1+3z} \|\mathcal{H}\|_{2,-\frac{1}{z}} \leq \|(a_i \cdot b_i \cdot c_i)_i\|_1$ holds. Finally, for the other lhs, we use Hölder's inequality on a_i, b_i and c_i , now with Hölder conjugates $\frac{1}{-1/y} + \frac{1}{-1/z} + \frac{1}{1/(1+y+z)} = 1$. \square

Algorithm	$\gamma_{2,1}(\mathcal{H})$	$\ \mathcal{H}\ _{2,3}$	$\ \mathcal{H}\ _F$
Winograd	$7 + \frac{8}{\sqrt{2}} + \frac{9}{\sqrt{3}} \approx 17.853$	$11 + \frac{8}{\sqrt{2}} + \frac{9}{\sqrt{3}}$	$\sqrt{14^3}$
Strassen	$12 + \frac{4}{\sqrt{2}} \approx 14.828$	$2 + \frac{20}{\sqrt{2}}$	$\sqrt{12^3}$
Eq. (38)	$\frac{75}{8} + \frac{4}{\sqrt{2}} \approx 12.203$	$\frac{125}{32} + \frac{4}{\sqrt{2}} + \frac{25}{2\sqrt{5}}$	$\sqrt{\frac{162}{16}^3}$
Eq. (45)	$\frac{75}{8} + \frac{4}{\sqrt{2}} \approx 12.203$	$\frac{125}{32} + \frac{4}{\sqrt{2}} + \frac{25}{2\sqrt{5}}$	$\sqrt{10} \sqrt{\frac{162}{16} \frac{810}{80\sqrt{10}}}$
Eq. (36)	$\frac{16}{\sqrt{3}} + \frac{4}{\sqrt{2}} \approx 12.066$	$\frac{16}{\sqrt{3}} + \frac{4}{\sqrt{2}}$	$\sqrt{10^3}$
Conv.	8.000	8	$\sqrt{8^3}$

Table 2: Illustration of Eq. (34) on several $\mathcal{H} = [L; R; P]$

Table 2 gives the Frobenius and $(2, 3)$ -norms of each of the three matrices defining the HM representation of several matrix product algorithms, as well as their $\gamma_{2,1}$ growth factor.

In the following proposition, we show that—up to orthogonal transformations—the minimum of the Frobenius norm of each of

the three HM representation components defining a fast 2×2 -matrix multiplication algorithms is $\sqrt{10}$.

Proposition 17. The minimal product $\|\mathcal{H}\|_F$ of the three Frobenius norms of the HM representation of any bilinear algorithm for matrix multiplication with 7 multiplications, is $\sqrt{10^3}$.

This proposition's proof is given in Appendix A.1. Remark that this lower bound is reached by the algorithm whose HM representation is given in Equation (36).

$$\begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & \frac{1}{2} & \frac{\sqrt{3}}{6} \\ 0 & 0 & 1 & -\frac{\sqrt{3}}{3} \\ 0 & 1 & 0 & \frac{\sqrt{3}}{3} \\ 0 & 0 & 0 & -\frac{2}{\sqrt{3}} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{\sqrt{3}}{6} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{\sqrt{3}}{6} \end{bmatrix}; \begin{bmatrix} 0 & \frac{2}{\sqrt{3}} & 0 & 0 \\ -1 & \frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{3}}{3} & 0 & -1 \\ \frac{1}{2} & -\frac{\sqrt{3}}{6} & \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{6} & \frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}; \begin{bmatrix} \frac{\sqrt{3}}{6} & \frac{1}{2} & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{3} & 0 & -1 & 0 \\ \frac{\sqrt{3}}{3} & -1 & 0 & 0 \\ \frac{\sqrt{3}}{6} & -\frac{1}{2} & -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{6} & -\frac{1}{2} & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{2}{\sqrt{3}} & 0 & 0 & 0 \end{bmatrix}^T. \quad (36)$$

Remark 18. Similarly, $(\sqrt[4]{3}/\sqrt{2}, \sqrt[4]{3}/\sqrt{6}, \sqrt[4]{3}/\sqrt{2}, -\sqrt[4]{3}/\sqrt{6})$ is a minimum of $\|L \cdot (W \otimes V)\|_{2,3}^3$ as in Proposition 17 for $\|L \cdot (W \otimes V)\|_{2,2}^2$. It turns out that this value is $16/\sqrt{3} + 4/\sqrt{2}$, the same as the $\gamma_{2,1}$ growth factor at this point, proving that our upper bound is reached.

We now turn to potential lower bounds.

Lemma 19. With $W = \begin{bmatrix} r & x \\ 0 & r^{-1} \end{bmatrix}$, $V = \begin{bmatrix} s & y \\ 0 & s^{-1} \end{bmatrix}$, L the first component of Strassen's HM representation given in Equation (6) and any $z \geq 0.5171$, the point $(\sqrt[4]{3}/\sqrt{2}, \sqrt[4]{3}/\sqrt{6}, \sqrt[4]{3}/\sqrt{2}, -\sqrt[4]{3}/\sqrt{6})$ is a local minimum of $\|L \cdot (W \otimes V)\|_{2,-1/z}$ as a function of r, x, s and y .

PROOF. As in the proof of Proposition 17, we give an explicit expression $f_z(r, x, s, y)$ of $\|L \cdot (W \otimes V)\|_{2,-1/z}$ equal to:

$$\left(\begin{aligned} & ((r^2 + x^2)(s^2 + y^2) + (2xy + 1/rs)/rs)^{-1/2z} + (rs)^{1/z} \\ & + (s^2 + (y + 1/s)^2)^{-1/2z} ((r^2 + x^2)^{-1/2z} + r^{1/z}) \\ & + (r^2 + (x - 1/r)^2)^{-1/2z} ((s^2 + y^2)^{-1/2z} + s^{1/z}) \\ & + ((r^2 + x^2)(s^2 + y^2))^{-1/2z} \end{aligned} \right)^{-z}. \quad (37)$$

Then the evaluation of its partial derivatives at the given point is zero, by inspection, for any real z . Now, the roots of the characteristic polynomial of the Hessian of f_z at this point are $\frac{n}{6z}(b_1 \pm \sqrt{\delta_1})$ and $\frac{n}{18z}(b_2 \pm \sqrt{\delta_2})$, with $\tau = \sqrt[2]{3}$ and $\lambda = \sqrt[2]{2}$, for $n = (\lambda + 6\tau)^{-1-z}$, using $b_1 = (32z - 11)\tau^{1+z} + 4z\lambda\sqrt{3}$, and $b_2 = (96z - 63)\tau^{1+z} + 52z\lambda\sqrt{3}$, $\delta_1 = 24(1 - 16z)\tau\lambda + (1344z^2 - 384z + 39)\tau^2 + 48z^2\lambda^2$ and finally $\delta_2 = 72(272z - 237)\tau\lambda + (12096z^2 - 20736z + 8991)\tau^2 + 8112z^2\lambda^2$.

First, δ_1 and δ_2 are positive for positive z , so that the eigenvalues are then always real. Second, both expressions $b_i^2 - \delta_i$ have the same root, strictly less than 0.5171. Third, all four eigenvalues are thus strictly positive for $z \geq 0.5171$. \square

Corollary 20. $11.7554696 < \frac{28}{9} 2^{\frac{11}{14}} 3^{\frac{5}{7}}$ is a lower bound for the $\gamma_{2,1}$ growth factor of an HM formula using 7 products.

PROOF. Following the proof of Lemma 19, we have that equality $f_z \left(\frac{\sqrt[4]{3}}{\sqrt{2}}, \frac{\sqrt[4]{3}}{\sqrt{6}}, \frac{\sqrt[4]{3}}{\sqrt{2}}, -\frac{\sqrt[4]{3}}{\sqrt{6}} \right) = \left(2^{\frac{1}{2z}} + 3^{\frac{1}{2z}} 2^{-\frac{1}{z}} 6 \right)^{-z}$ holds. Denoting this quantity by ζ_z we thus have that $7^{1+3z} \zeta_z^3 \leq 7^{1+3z} \|\mathcal{H}\|_{2,-1/z}$, which

left-hand side limit at $z = \infty$ is $\frac{28}{9} 2^{\frac{11}{14}} 3^{\frac{5}{7}}$. By Eq. (35), this shows that this value is less than or equal to $\gamma_{2,1}$ as announced. It is also the limit of $\zeta_z^{2\zeta} \zeta_{-1-2z} \leq \|L\|_{2,-\frac{1}{z}} \cdot \|R\|_{2,-\frac{1}{z}} \cdot \|P\|_{2,\frac{1}{1+z}}$ at $z = \infty$. \square

Corollary 20 for instance shows that the $\gamma_{2,1}$ growth factor of the conventional algorithm (8) can not be attained by such fast algorithms. Let us see now how this bound behaves in our experiments.

6 ALGORITHMS INTO PRACTICE

In this section, we present several techniques to lower the number of operations used in our algorithms and thus, lower complexity bounds and potentially obtain a better accuracy.

Determining actual complexity bounds requires estimating the number of operations required to implement a given formula. Considering an HM representation, a direct upper bound can be obtained by: first count the number of coefficients different from 0, ± 1 to upper bound the number of multiplications/divisions; second count the number of non-zero coefficients, minus the number of rows, to get an upper bound on the number of additions/subtractions.

To obtain lower operation counts, we use the following techniques: first, we select among equivalently accurate algorithms: this is presented in Section 6.1; second, we factor as much as possible the computations between rows of the HM representations, as in Section 6.2; third, we use dependent rows as more opportunities for factorization, as in Section 6.3. We then present some good candidates (as well as in Appendix A.4) and we eventually look at some potential sparse alternative change of basis in Section 6.4.

6.1 Sparsifying via rotations

We have seen in Lemma 13 that orthogonal transformations leave the Frobenius norm invariant and thus, the $\gamma_{2,1}$ growth factor. Therefore, one can apply 4×4 generic Kronecker products of orthogonal 2×2 (rotation) matrices using Lemma 6 and try to optimize the considered HM representation for several possible goals: (1) a smaller number of non-zero coefficients in HM representation components; (2) a non-zero pattern better suited to factorization (see the technique of Section 6.2); (3) a triangular (sparse) subset of independent rows (see the technique of Section 6.3).

For instance, to obtain Eq. (36), we solve for the minimal values of the Frobenius norms as in Proposition 17 and then for orthogonal transformations that produce as many vectors of the canonical basis as possible. Doing so, we found that with $\gamma_{2,1}$ set to $16/\sqrt{3} + 4/\sqrt{2}$ and HM representation component Frobenius norms set to $\sqrt{10}$, the maximal possible number of canonical vectors was 1. Equation (36) is one of those. Similarly, Equation (45) is an orthogonal optimization of Equation (38), with one canonical vector in each of components of the HM representation. A C++ implementation of these tools is available in the PLINOPT library [11].

6.2 Factoring heuristics

For the implementation of a given linear operator (in this work one of the matrices in the HM representation) one can try to find the shortest straight-line program for its computation. The problem is NP-hard in general (see e.g. [6, § 3.1.1]); but for small matrices, and over the field with 2 elements, [6] and references therein propose several heuristics that potentially reduce the number of operations.

Not all of them are applicable to fields with more elements, but we use a kind of common sub-expression eliminations, the ‘‘cancellation-free’’ search, described in Algorithm 6 and implemented in `plinopt/optimizer -D` [11].

6.3 Kernel computation and Tellegen’s principle

If the rank of the linear operator is lower than its number of rows, then an additional strategy has proven useful: compute first some independent rows, then express the dependent ones by their linear relations. For this, Algorithm 1 computes a left kernel of the linear operator and uses it to compute the dependent rows via linear combinations of the independent ones. This is sometimes faster than directly computing the dependent rows. Of course, if the matrix’s rank is lower than the number of columns, one can apply Algorithm 1 to the transposed matrix and then apply the Tellegen’s transposition principle to recover the transposed linear dependencies (e.g. see [5] and references therein).

Algorithm 1 Kernel decomposition of a linear operator

Input: M in $\mathbb{K}^{m \times n}$ such that $r = \text{Rank } M$.

Output: A straight line program computing $\vec{u} \leftarrow M \cdot \vec{v}$.

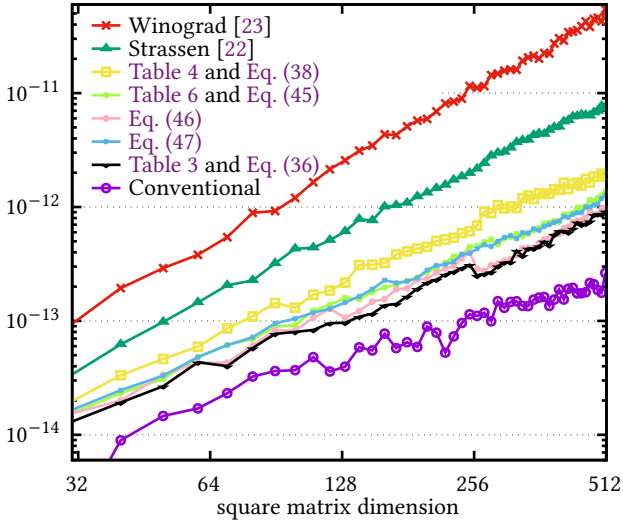
- 1: By Gaussian elimination, compute $M = P \cdot L \cdot U \cdot Q$ with P a permutation matrix, L in $\mathbb{K}^{m \times r}$ be $\begin{bmatrix} L_1 \\ L_2 \end{bmatrix}$ unit upper triangular and L_1 in $\mathbb{K}^{r \times r}$; choosing P so that (1) the first r rows of $P^{-1}M$ are sparsest; (2) L_1 is the sparsest; (3) L_2 is the sparsest;
 - 2: Let σ be the permutation represented by P ;
 - 3: Apply Alg. 6 to $[u_{\sigma(1)} \dots u_{\sigma(r)}]^T \leftarrow [I, 0] \cdot P \cdot M \cdot \vec{v}$;
 $\triangleright [-L_2 \cdot L_1^{-1} \ I_{m-r}]$ is a (sparse) left kernel of M and provides the linear dependencies of the remaining rows
 - 4: Apply Alg. 6 to $[u_{\sigma(r+1)} \dots u_{\sigma(m)}]^T \leftarrow L_2 \cdot L_1^{-1} [u_{\sigma(1)} \dots u_{\sigma(r)}]^T$.
-

Algorithm 1 is implemented in `plinopt/optimizer -K`. The Tellegen’s transposition principle applied to such SLPs is implemented in `plinopt/transposer` [11, 12]. These routines have produced the implementations for our different HM formulas given in the following section (e.g. the implementation Table 3 of Eq. (36) with only 24 additions and 12 multiplications/divisions).

$$\begin{array}{llll}
 t_1 = \frac{\sqrt{3}}{3} a_{22} & t_2 = a_{21} + t_1 & s_1 = \frac{\sqrt{3}}{3} b_{21} & s_2 = s_1 - b_{11} \\
 t_3 = a_{12} + t_2 & l_1 = \frac{\sqrt{3}}{2} a_{11} + \frac{1}{2} t_3 & s_3 = s_2 + b_{22} & r_1 = 2s_1 \\
 l_2 = a_{12} - t_1 & l_3 = t_2 & r_2 = s_2 & r_3 = s_1 - b_{22} \\
 l_4 = 2t_1 & l_5 = l_2 - l_1 & r_4 = \frac{1}{2} s_3 - \frac{\sqrt{3}}{2} b_{12} & r_5 = r_3 + r_4 \\
 l_6 = l_5 + l_4 & l_7 = l_5 + l_3 & r_6 = r_1 - r_5 & r_7 = r_5 - r_2 \\
 \\
 p_1 = l_1 \cdot r_1 & p_2 = l_2 \cdot r_2 & p_3 = l_3 \cdot r_3 & p_4 = l_4 \cdot r_4 \\
 p_5 = l_5 \cdot r_5 & p_6 = l_6 \cdot r_6 & p_7 = l_7 \cdot r_7 & \\
 w_2 = p_5 + p_1 + p_6 & w_1 = p_7 + p_6 & w_3 = w_2 - p_2 & w_5 = \frac{p_4 + w_2}{2} \\
 c_{12} = p_1 - p_3 - w_5 & c_{21} = w_3 - w_5 & c_{22} = \sqrt{3} w_5 & \\
 c_{11} = \frac{\sqrt{3}}{3} (w_3 - c_{12} - 2w_1) & & &
 \end{array}$$

Table 3: SLP of Eq. (36) with 24 add. and 12 mul./div.

Figure 1: Numerical accuracy vs size (normal distribution) error



Remark 21. The accuracy obtained with our different fast variants is given in Figure 1 using the Matlab framework of [8], which we forked in [12] and where we have just added the implementations of the variants presented here. Thus, in Figures 1, 2 and 3 we present the error as the infinity norm of the difference between the result of our implementations and the exact matrix multiplication.

In Figure 1, all our variants, Tables 3, 4 and 6 and Eqs. (46) and (47) with decreasing $\gamma_{2,1}$, are mostly more and more accurate. Our best algorithm presents an order of magnitude advantage over Strassen's and two orders of magnitude advantage over Winograd's. It is then quite close to the conventional algorithm's accuracy. Figure 1 uses normal distribution, the same behavior is obtained, e.g., with a uniform distribution (see corresponding code in [12]).

Remark 22. In [4] the authors consider all bilinear algorithms using 7 multiplications with constants of the form $\pm 2^i$; they showed that Strassen's original method [22] reaches in this class the minimum value 12 of their $\gamma_{0,1\infty}$ factor error bound (while for instance that of Winograd [23] is 18, see also Table 1).

We propose an algorithm in this class that has a worse $\gamma_{0,1\infty}$ of 40, but a $\gamma_{2,1}$ of $4/\sqrt{2} + 75/8 \approx 12.2034$, better than those of Strassen 14.8284 or Winograd 17.8530 (see Table 2). This algorithm is defined by the following HM representation:

$$\begin{bmatrix} 0 & -1 & 1 & 0 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & 1 & -\frac{1}{2} \\ 0 & 1 & 0 & -\frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{2} \\ 1 & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{4} \\ 0 & 1 & 0 & \frac{1}{2} \end{bmatrix}; \begin{bmatrix} 1 & 0 & 0 & -1 \\ 1 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -1 \\ 0 & \frac{1}{4} & -1 & -\frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 1 \\ 1 & -\frac{1}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{1}{4} & 1 & -\frac{1}{2} \end{bmatrix}; \begin{bmatrix} 0 & 1 & 1 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{4} & -\frac{1}{2} & -\frac{1}{2} & 1 \\ -\frac{1}{2} & 0 & 1 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{2} & -1 & 0 & 0 \\ \frac{1}{2} & 0 & 1 & 0 \end{bmatrix}^T. \quad (38)$$

Figure 1 shows that the induced algorithm given in Table 4 is also more accurate in practice than both Strassen's and Winograd's variants.

Remark 23. Eq. (38) was obtained by approximating the minimal point of the $\gamma_{2,1}$ growth factor taken from Proposition 15 with the

$$\begin{aligned} r_1 &= \frac{1}{2}a_{22} & t_2 &= a_{21} - r_1 & u_1 &= \frac{1}{2}b_{12} & s_1 &= b_{11} + u_1 \\ t_3 &= a_{12} + r_1 & t_0 &= t_2 - t_3 & s_2 &= u_1 - b_{22} & u_2 &= s_1 - b_{22} \\ t_4 &= a_{21} + r_1 & r_2 &= t_2 - a_{12} & s_4 &= b_{22} + u_1 & s_0 &= s_1 - s_4 \\ t_5 &= a_{11} + \frac{1}{2}r_2 & t_1 &= t_5 - t_0 & s_3 &= \frac{1}{2}u_2 - b_{21} & s_5 &= s_0 - s_2 \\ p_1 &= t_0 \cdot s_0 & p_2 &= t_1 \cdot s_1 & p_3 &= t_2 \cdot s_2 & p_4 &= t_3 \cdot s_3 \\ p_5 &= t_4 \cdot s_4 & p_6 &= t_5 \cdot s_5 & p_7 &= (t_4 - t_0) \cdot (s_0 - s_3) \\ c_{22} &= p_5 + p_3 & v_1 &= p_1 - p_6 - p_3 & v_2 &= p_7 + p_6 & v_3 &= p_4 + v_1 \\ v_4 &= \frac{c_{22}}{2} & c_{12} &= p_2 + v_1 + v_4 & c_{21} &= v_2 + v_3 + v_4 & c_{11} &= \frac{(c_{12} + v_2 - v_3)}{2} \end{aligned}$$

Table 4: SLP of Eq. (38) with 27 add., 6 div. by 2 and $\gamma_F \approx 12.2034$

smallest powers of 2. Further rational higher-order approximations are obtained in the same vein, giving for instance Eqs. (45) to (47) and Table 6, as shown in Appendix A.2.

6.4 Alternative basis sparsification

The technique of [3, 18] reduces the number of operations by factoring each matrix in the HM decomposition into a sparser one via a 4x4 change of basis (CoB). In a recursive version, the left and right-hand sides (resp. result) of considered CoB can be recursively precomputed (resp. post-computed), for a total cost in $O(n^2 \log n)$. In the meantime the sparser 7x4 matrices are applied, reducing the dominant term of the computation. The optimal decomposition of Winograd's algorithm in [18, § 3.3] reduces the number of intermediate additions from 15 to 12. For a fully recursive version, this reduces the leading term in the cost bound from $6n^{\log_2 7}$ to $5n^{\log_2 7}$.

Applying this approach to the algorithm of Eq. (36), using the CoB of Eq. (39) leads to the sparser HM representation in Eq. (40). The leading term of the cost bound thus is reduced from $13n^{\log_2 7}$ for Table 3 to $5n^{\log_2 7}$ for Algorithms 2 to 5 and Table 5.

$$\begin{bmatrix} 0 & 0 & 0 & \frac{2}{\sqrt{3}} \\ 0 & 1 & 0 & \frac{\sqrt{3}}{3} \\ 0 & 0 & 1 & -\frac{\sqrt{3}}{3} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}; \begin{bmatrix} 0 & \frac{2}{\sqrt{3}} & 0 & 0 \\ 1 & -\frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{3}}{3} & 0 & -1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}; \begin{bmatrix} -\frac{2}{\sqrt{3}} & 0 & 0 & 0 \\ \frac{\sqrt{3}}{3} & -1 & 0 & 0 \\ -\frac{\sqrt{3}}{3} & 0 & -1 & 0 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}^T, \quad (39)$$

$$\begin{bmatrix} 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}; \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \end{bmatrix}; \begin{bmatrix} 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}^T. \quad (40)$$

To obtain this CoB, the generic technique of [3] can be used. In our case, for 4x4 CoB, the following heuristic was sufficient to obtain optimal (12-additions) sparse (0, ±1) intermediate matrices: (1) Find independent columns of each CoB one at a time; (2) For this, alternatively factor-out common coefficients in the resulting columns and find a linear combination minimizing the density of the resulting column, using as coefficients of the combination only in $\{-1, 0, 1\}$ and some of the values of the coefficients of the input; (3) Until this alternation does not sparsify anymore. This heuristic is implemented in `plinopt/sparsifier` [11] and the resulting implementation is shown in Algorithms 2 to 5 and Table 5.

The sparsification process improves the $\gamma_{2,1}$ growth factor: applied to Winograd's original algorithm [23], it goes down from $7 + 8/\sqrt{2} + 9/\sqrt{3} \approx 17.853$ to $4 + 12/\sqrt{2} \approx 12.486$ in [18] and applied on Eq. (36), from $4/\sqrt{2} + 16/\sqrt{3} \approx 12.066$ to $7 + 6/\sqrt{2} \approx 11.243$.

Algorithm 2 LCoB(A, ℓ) left change-of-basis of Eq. (39)

- 1: **if** $\ell \leq 0$ **then return** A . **end if**
- 2: $m_1 = \text{LCoB}(a_{11}, \ell-1)$; $m_2 = \text{LCoB}(a_{21}, \ell-1)$;
- 3: $m_3 = \text{LCoB}(a_{12}, \ell-1)$; $m_4 = \text{LCoB}(a_{22}, \ell-1)$;
- 4: $t_1 = \frac{1}{\sqrt{3}}m_4$; $t_2 = m_3 - m_2$; $t_3 = m_1 + m_4$;
- 5: **return** $\left[\frac{2}{\sqrt{3}}m_4, m_2 + t_1, m_3 - t_1, \frac{1}{2}t_2 - \frac{\sqrt{3}}{2}t_3 \right]$.

Algorithm 3 RCoB(A, ℓ) right change-of-basis of Eq. (39)

- 1: **if** $\ell \leq 0$ **then return** A . **end if**
- 2: $m_1 = \text{RCoB}(a_{11}, \ell-1)$; $m_2 = \text{RCoB}(a_{21}, \ell-1)$;
- 3: $m_3 = \text{RCoB}(a_{12}, \ell-1)$; $m_4 = \text{RCoB}(a_{22}, \ell-1)$;
- 4: $t_1 = \frac{1}{\sqrt{3}}m_2$; $t_2 = m_1 + m_4$; $t_3 = m_2 - m_3$;
- 5: **return** $\left[\frac{2}{\sqrt{3}}m_2, m_1 - t_1, t_1 - m_4, \frac{\sqrt{3}}{2}t_3 - \frac{1}{2}t_2 \right]$.

Algorithm 4 CoBP(A, ℓ) product change-of-basis of Eq. (39)

- 1: **if** $\ell \leq 0$ **then return** A **end if**
- 2: $m_1 = \text{CoBP}(a_{11}, \ell-1)$; $m_2 = \text{CoBP}(a_{21}, \ell-1)$;
- 3: $m_3 = \text{CoBP}(a_{12}, \ell-1)$; $m_4 = \text{CoBP}(a_{22}, \ell-1)$;
- 4: $t_1 = \frac{1}{2}m_4$; $t_2 = m_2 - m_3$; $t_3 = \frac{\sqrt{3}}{2}m_4$;
- 5: **return** $\left[t_3 + \frac{1}{\sqrt{3}}t_2 - m_1 \frac{2}{\sqrt{3}}, -m_2 - t_1, t_1 - m_3, t_3 \right]$.

Algorithm 5 Sparsification applied to Eq. (36) (via Eqs. (39) and (40))

Input: $A, B \in \mathbb{K}^{n_0 2^\ell \times n_0 2^\ell}$.

Output: $C = A \cdot B$.

- 1: $\tilde{A} \leftarrow \text{LCoB}(A, \ell)$; $\tilde{B} \leftarrow \text{RCoB}(B, \ell)$ ▷ Via Algorithms 2 and 3
- 2: $\tilde{C} \leftarrow \tilde{A} \cdot \tilde{B}$; ▷ Via Table 5 with ℓ recursive calls
- 3: **return** $C \leftarrow \text{CoBP}(\tilde{C}, \ell)$. ▷ Via Algorithm 4

However the resulting sparser bilinear operator itself no longer correspond to a matrix multiplication algorithm, and this has two consequences: first, the error of this operator only follows the weaker bound Equation (16) of Theorem 12 (with an additional logarithmic factor). Second, the error bound for the resulting matrix multiplication algorithm must then also include the contribution of the CoB. A tight analysis of this contribution is made in the recent work of [21, Th. I.1]. There, the first error bound for an alternative basis based matrix multiplication algorithm is produced, in the form of Equation (16) where the contribution of the CoB only affects the multiplicative constant Q_0 by a small amount.

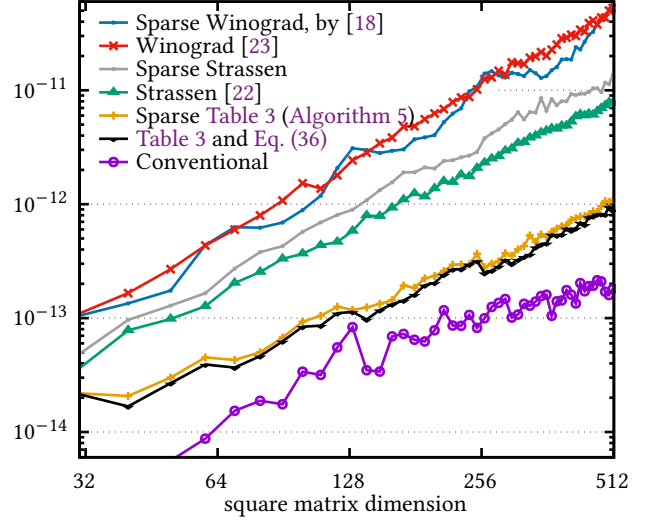
Remark 24. Algorithm 5, enjoys simultaneously the best known leading term in the cost bound, and a close to the best known numerical accuracy for sub-cubic 2×2 algorithms. The former property comes from the fact that Eq. (40) requires only 12 additions. The latter is shown in practice in Figure 2, and the proof of [21, Th. I.1] could be adapted to this algorithm for a theoretical error bound.

Following [20, § 3.2], we can also confirm our algorithms' accuracy on badly conditioned matrices (see Figure 3 in Appendix A.3).

$$\begin{aligned}
s_1 &= a_{11} + a_{12} & s_2 &= a_{11} + a_{22} & s_3 &= a_{11} - a_{21} \\
t_1 &= b_{12} + b_{22} & t_2 &= b_{11} + b_{12} & t_3 &= b_{12} + b_{21} \\
p_1 &= a_{11} \cdot b_{12} & p_2 &= s_1 \cdot b_{21} & p_3 &= a_{21} \cdot t_1 & p_4 &= a_{12} \cdot t_2 \\
p_5 &= s_2 \cdot b_{22} & p_6 &= a_{22} \cdot t_3 & p_7 &= s_3 \cdot b_{11} \\
c_{11} &= p_7 - p_6 & c_{12} &= p_2 + p_3 & c_{21} &= p_4 - p_5 & c_{22} &= p_1 + p_2 + p_5 + p_6
\end{aligned}$$

Table 5: SLP of Eq. (40) with 12 additions

Figure 2: Numerical effect of sparsification (normal distribution) error



Remark 25. To further improve their practical behavior, as done in [9, § 4.3], [2, § 6.1] or [1, § 6], some diagonal scaling adapted to specific input matrices can be added to any algorithms and thus to any of the variants presented here. The idea is to precondition input matrices with well suited matrices thanks to Definition 5.

7 CONCLUSION AND FUTURE WORK

We have presented a technique and a software to find more accurate recursive fast matrix multiplication algorithms. Our analysis shows that our most accurate 2×2 formulas are probably optimal with respect to the tensor nuclear (Frobenius) norm. We still anyway have a potential gap of at most 2.6% to further explore.

We also have shown ways to optimize the time complexity of 2×2 matrix product variants to simultaneously obtain a better accuracy in practice and a time complexity bound with the best currently known leading term (obtained via alternative basis sparsification). There remains to compare the actual timings of these variants on different type of matrices and different range of matrix dimensions.

Also, isotropies play a central role in this matter as shown by the fact that the minimal growth factor reached in this work is exactly the same as that of the algorithm obtained by [13, Eq. (22)], while reconstructing Strassen's algorithm, using only the knowledge of its stabilizer and its representation with minimal Frobenius norms.

REFERENCES

- [1] G. Ballard, A.R. Benson, A. Druinsky, B. Lipshitz, and O. Schwartz. 2016. Improving the Numerical Stability of Fast Matrix Multiplication. *SIAM Journal on Matrix Analysis and Applications* 37, 4 (2016), 1382–1418. [10.1137/15M1032168](https://doi.org/10.1137/15M1032168)
- [2] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. 2012. Communication-Optimal Parallel Algorithm for Strassen's Matrix Multiplication. In *SPAA'12: Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures* (Pittsburgh, PA, USA), 193–204. [10.1145/2312005.2312044](https://doi.org/10.1145/2312005.2312044)
- [3] G. Beniamini and O. Schwartz. 2019. Fast Matrix multiplication via sparse decomposition. In *SPAA'19: Proceedings of the 31st ACM Symposium on Parallel Algorithms and Architectures* (Phoenix, AZ, USA), 11–22. [10.1145/3323165.3323188](https://doi.org/10.1145/3323165.3323188)
- [4] D.A. Bini and G. Lotti. 1980. Stability of fast algorithms for matrix multiplication. *Numer. Math.* 36, 1 (March 1980), 63–72. [10.1007/BF01395989](https://doi.org/10.1007/BF01395989)
- [5] A. Bostan, G. Lecerf, and É. Schost. 2003. Tellegen's principle into practice. In *ISSAC'03: Proceedings of the 2003 ACM International Symposium on Symbolic and Algebraic Computation* (Philadelphia, PA, USA), 37–44. [10.1145/860854.860870](https://doi.org/10.1145/860854.860870)
- [6] J. Boyar, P. Matthews, and R. Peralta. 2013. Logic Minimization Techniques with Applications to Cryptology. *Journal of Cryptology* 26, 2 (April 2013), 280–312. [10.1007/s00145-012-9124-7](https://doi.org/10.1007/s00145-012-9124-7)
- [7] R.P. Brent. 1970. *Algorithms for matrix multiplication*. Technical Report **STAN-CS-70-157**. C.S. Dpt. Stanford University.
- [8] Z. Dai and L.-H. Lim. 2023. Numerical stability and tensor nuclear norm. *Numer. Math.* 155, 3-4 (Nov. 2023), 345–376. [10.1007/s00211-023-01377-5](https://doi.org/10.1007/s00211-023-01377-5)
- [9] J. Demmel, I. Dumitriu, and O. Holtz. 2007. Fast linear algebra is stable. *Numer. Math.* 108, 1 (Oct. 2007), 59–91. [10.1007/S00211-007-0114-X](https://doi.org/10.1007/S00211-007-0114-X)
- [10] J. Demmel, I. Dumitriu, O. Holtz, and R. Kleinberg. 2007. Fast matrix multiplication is stable. *Numer. Math.* 106, 2 (Feb. 2007), 199–224. [10.1007/s00211-007-0061-6](https://doi.org/10.1007/s00211-007-0061-6)
- [11] J.-G. Dumas, B. Grenet, C. Pernet, and A. Sedoglavic. 2024. *PLinOpt, a collection of C++ routines handling linear & bilinear programs*. v2.3, 50ec789. <https://github.com/jgdumas/plinopt> (1.7 kSLOC).
- [12] J.-G. Dumas, C. Pernet, and A. Sedoglavic. 2024. *Matlab accurate fast matrix multiplications via 2 x 2 recursion*. v1.0, 52db87c. <https://github.com/jgdumas/Fast-Matrix-Multiplication> (1.6 kSLOC).
- [13] J.A. Grochow and C. Moore. 2016. *Matrix multiplication algorithms from group orbits*. Technical Report arXiv:arXiv:1612.01527
- [14] de Groote, H.-F. 1978. On varieties of optimal algorithms for the computation of bilinear mappings I. The isotropy group of a bilinear mapping. *Theoretical Computer Science* 7, 2 (1978), 1–24. [10.1016/0304-3975\(78\)90038-5](https://doi.org/10.1016/0304-3975(78)90038-5)
- [15] de Groote, H.-F. 1978. On varieties of optimal algorithms for the computation of bilinear mappings II. Optimal algorithms for 2 x 2-matrix multiplication. *Theoretical Computer Science* 7, 2 (1978), 127–148. [10.1016/0304-3975\(78\)90045-2](https://doi.org/10.1016/0304-3975(78)90045-2)
- [16] N.J. Higham. 2002. *Accuracy and Stability of Numerical Algorithms* (2 ed.). SIAM. [10.1137/1.9780898718027](https://doi.org/10.1137/1.9780898718027)
- [17] J.E. Hopcroft and J. Musinski. 1973. Duality Applied to the Complexity of Matrix Multiplications and other Bilinear Forms. In *5th ACM Symposium on the theory of computing*, (Austin, TX, USA), 73–87. [10.1137/0202013](https://doi.org/10.1137/0202013)
- [18] E. Karstadt and O. Schwartz. 2017. Matrix Multiplication, a Little Faster. In *SPAA'17: Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures* (Washington, DC, USA), 101–110. [10.1145/3087556.3087579](https://doi.org/10.1145/3087556.3087579)
- [19] J.M. Landsberg. 2016. *Geometry and complexity theory*. CSAM, Vol. 169. Cambridge University Press. [10.1017/9781108183192](https://doi.org/10.1017/9781108183192)
- [20] K. Ozaki, T. Ogita, S. Oishi, and S. M. Rump. 2012. Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications. *Numerische Algorithmen* 59, 1 (Jan. 2012), 95–118. [10.1007/S11075-011-9478-1](https://doi.org/10.1007/S11075-011-9478-1)
- [21] O. Schwartz, S. Toled, N. Vaknim, and G. Wiernik. 2024. Alternative Basis Matrix Multiplication is fast and stable. In *IPDPS'24: proceedings of the 38th IEEE International Parallel Distributed Processing Symposium*, (San Francisco, CA, USA).
- [22] V. Strassen. 1969. Gaussian elimination is not optimal. *Numer. Math.* 13, 4 (Aug. 1969), 354–356. [10.1007/BF02165411](https://doi.org/10.1007/BF02165411)
- [23] S. Winograd. 1977. La complexité des calculs numériques. *La Recherche* 8 (1977), 956–963.

A SUPPLEMENTARY MATERIALS

A.1 Computational proofs

We gather here proofs of several propositions that are simple computations on objects presented in our work (verified also in [12]).

PROOF OF PROPOSITION 15. To simplify our computations, we use the following coordinates $(\rho, \xi) = (\sqrt{4/r}, (x-1)/2)$, the matrix $u(\rho, \xi)$ and the associated isotropy $\mathfrak{g}_{\rho, \xi} \diamond \mathcal{S}$. In that case, the explicit expression of the $\gamma_{2,1}$ growth factor $\gamma(\mathfrak{g}_{r,x} \diamond \mathcal{S})$ along this orbit

is $2\sqrt{2} + 3\mathcal{A}$ with $\mathcal{A}(r, x)$ equal to $((1+x)^2 + r)((x-1)^2 + r)/r\sqrt{r}$. To conclude, we prove that the minimum of $\mathcal{A}(r, x)$ in $\mathbb{R}^+ \times \mathbb{R}$ is $16/3\sqrt{3}$. The partial derivatives of $\mathcal{A}(r, x)$ w.r.t. r and x are

$$\frac{\partial \mathcal{A}}{\partial x} = 4 \frac{(x^2 + r - 1)x}{r^{3/2}}, \quad \frac{\partial \mathcal{A}}{\partial r} = \frac{r^2 - 2(x^2 + 1)r - 3(x^2 - 1)^2}{2r^{5/2}}. \quad (41)$$

First, notice that $\frac{\partial \mathcal{A}}{\partial x}(1 - x^2, x)$ is 0 and that $\frac{\partial \mathcal{A}}{\partial r}(1 - x^2, x)$ is equal to $2/((x-1)(1+x))^{3/2}$. The only critical point is x equal to 0 and, as r is positive, it only could be equal to 3. The Hessian matrix is:

$$H(\mathcal{A}(r, x)) = \frac{1}{r^{\frac{3}{2}}} \begin{bmatrix} 4(3x^2+r-1) & -\frac{2}{r}(3x^2+r-3) \\ -\frac{2}{r}(3x^2+r-3) & \frac{r^2-6(x^2+1)r-15(x^2-1)^2}{4r^2} \end{bmatrix}, \quad (42)$$

one can notice that $H(\mathcal{A}(3, 0))$ is equal to $\begin{bmatrix} 2^3 & 0 \\ 0 & 2/3 \end{bmatrix} / 3\sqrt{3}$.

Hence, the second partial derivative test states that $(\sqrt[4]{4/3}, -1/2)$ is a local minimum of $\gamma(\mathfrak{g}_{r,x} \diamond \mathcal{S})$; it is equal to $2\sqrt{2} + 16/\sqrt{3}$ that is ≈ 12.06603143 . To conclude, the $\gamma_{2,1}$ growth factor reaches its global minimal at this point on the considered orbit because it is its only critical point. \square

PROOF OF PROPOSITION 17. Recall that any of the three matrices in the HM representation of such an algorithm is obtained by row and column permutations of one of these matrices, multiplied by the Kronecker product of two invertible 2×2 matrices W and V (see Lemma 6 and Theorem 7). From the analysis of Section 4, we only need to consider the case where W and V are each of the form $\begin{bmatrix} r & x \\ 0 & r^{-1} \end{bmatrix}$, with strictly positive r (matrices W and V are taken in a simpler form for the sake of simplicity). For this, we let W be $\begin{bmatrix} r & x \\ 0 & r^{-1} \end{bmatrix}$, V be $\begin{bmatrix} s & y \\ 0 & s^{-1} \end{bmatrix}$ and choose L the first component of Strassen's HM representation given in Eq. (6). The Frobenius norm of $L \cdot (W \otimes V)$ is given by the following function of r, x, s and y :

$$f(r, x, s, y) = 4r^2s^2 + 3r^2y^2 + 3s^2x^2 + x^2y^2 + \frac{r^2}{s^2} + \frac{s^2}{r^2} + \frac{1}{(rs)^2} + \left(\frac{x}{s} + \frac{1}{rs}\right)^2 + \left(\frac{y}{r} - \frac{1}{rs}\right)^2 + \left(\frac{x}{s} - xy\right)^2 + \left(\frac{y}{r} + xy\right)^2 + \left(xy + \frac{1}{rs}\right)^2 + \left(\frac{s}{r} + xs\right)^2 + \left(\frac{r}{s} - ry\right)^2. \quad (43)$$

Solving the four partial derivatives of the gradient $\left[\frac{\partial f}{\partial r}, \frac{\partial f}{\partial s}, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$, for a simultaneous zero, we obtain that the only real extrema of f are at the four points $r = \pm\sqrt[4]{3}/4, s = \pm r, x = 2/3r^3, y = -2/3s^3$, for which its value is always 10. The additional constraint that r and s are positive, thus gives a single extremum. Now, the Hessian matrix at that point is computed as:

$$H_f \left(\frac{\sqrt[4]{3}}{\sqrt{2}}, \frac{\sqrt[4]{3}}{\sqrt{6}}, \frac{\sqrt[4]{3}}{\sqrt{2}}, -\frac{\sqrt[4]{3}}{\sqrt{6}} \right) = \frac{4}{9} \begin{bmatrix} \frac{195}{\sqrt{3}} & \frac{69}{\sqrt{3}} & -15 & 15 \\ \frac{69}{\sqrt{3}} & \frac{195}{\sqrt{3}} & -15 & 15 \\ -15 & -15 & \frac{45}{\sqrt{3}} & \frac{9}{\sqrt{3}} \\ 15 & 15 & \frac{9}{\sqrt{3}} & \frac{45}{\sqrt{3}} \end{bmatrix}. \quad (44)$$

Leaving out the factor $4/9$, the characteristic polynomial of this matrix is $Z^4 - 160\sqrt{3}Z^3 + 22536Z^2 - 362880\sqrt{3}Z + 5143824$ whose all four roots $\sqrt{7500} \pm \sqrt{5232}, 30 \pm 12\sqrt{3}$ are positive. Therefore the point is a local minimum. From this, we see that $\sqrt{10^3}$ is a lower bound on the product of their Frobenius norms. Furthermore, remark that this lower bound is reached, by the algorithm which HM representation is given in Equation (36). \square

A.2 Rational approximations

As stated in Remark 23, by approximating the minimal point of the $\gamma_{2,1}$ growth factor presented in Proposition 15, we could construct further algorithms presented in this section.

$$\begin{bmatrix} \frac{4}{9} & -\frac{8}{9} & -\frac{8}{9} & -\frac{4}{9} \\ 0 & \frac{5}{9} & 0 & \frac{10}{9} \\ \frac{8}{9} & -\frac{2}{3} & 0 & 0 \\ \frac{4}{9} & \frac{5}{9} & \frac{8}{9} & \frac{4}{9} \\ 0 & -\frac{10}{9} & 0 & 0 \\ \frac{4}{9} & -\frac{1}{3} & -\frac{8}{9} & \frac{2}{3} \\ -\frac{4}{9} & -\frac{2}{9} & \frac{8}{9} & \frac{4}{9} \end{bmatrix}; \begin{bmatrix} -\frac{3}{5} & \frac{4}{5} & -\frac{4}{5} & -\frac{3}{5} \\ 0 & \frac{1}{2} & 0 & -1 \\ -1 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{3}{4} & 0 & 0 \\ 0 & -\frac{3}{10} & \frac{4}{5} & -\frac{4}{5} \\ \frac{3}{5} & -\frac{3}{10} & \frac{4}{5} & -\frac{4}{5} \\ -\frac{3}{5} & -\frac{3}{10} & -\frac{4}{5} & -\frac{4}{5} \\ -\frac{3}{5} & -\frac{3}{10} & -\frac{4}{5} & -\frac{4}{5} \end{bmatrix}; \begin{bmatrix} \frac{9}{20} & \frac{9}{10} & \frac{9}{10} & -\frac{9}{20} \\ 0 & 0 & \frac{27}{40} & -\frac{9}{10} \\ -\frac{9}{8} & 0 & -\frac{9}{16} & 0 \\ \frac{9}{20} & \frac{9}{10} & \frac{9}{40} & \frac{9}{20} \\ -\frac{27}{40} & \frac{9}{10} & \frac{27}{80} & -\frac{9}{20} \\ -\frac{9}{40} & \frac{9}{10} & \frac{27}{80} & -\frac{9}{20} \\ 0 & 0 & -\frac{9}{8} & 0 \\ \frac{9}{20} & \frac{9}{10} & -\frac{9}{40} & -\frac{9}{20} \end{bmatrix}^T. \quad (45)$$

First, Eq. (45) is an orthogonal optimization of Eq. (38), with one canonical vector in each of components of the HM representation. Unfortunately some small non-powers of 2 are then unavoidable, but this gives in Table 6 an algorithm realizing the formula with fewer additions than that of Table 4.

$$\begin{aligned} u_1 &= \frac{1}{2}a_{12} + a_{22} & t_1 &= \frac{10}{9}u_1 & t_2 &= \frac{8}{9}a_{11} - \frac{2}{3}a_{12} & t_4 &= \frac{10}{9}a_{12} \\ t_3 &= \frac{8}{9}a_{21} + \frac{4}{9}(a_{11} + u_1) & t_0 &= t_2 - t_3 & t_5 &= t_1 + t_0 & t_6 &= t_4 + t_0 \\ v_1 &= \frac{1}{2}b_{12} & s_1 &= v_1 - b_{22} & s_2 &= v_1 - b_{11} & s_3 &= \frac{5}{4}b_{12} \\ s_4 &= \frac{2}{5}b_{22} - \frac{4}{5}b_{21} + \frac{3}{5}s_2 & s_0 &= s_1 + s_4 & s_5 &= s_0 - s_2 & s_6 &= s_3 - s_0 \\ p_0 &= t_0 \cdot s_0 & p_1 &= t_1 \cdot s_1 & p_2 &= t_2 \cdot s_2 & p_3 &= t_3 \cdot s_3 \\ p_4 &= t_4 \cdot s_4 & p_5 &= t_5 \cdot s_5 & p_6 &= t_6 \cdot s_6 \\ w_1 &= p_6 + p_0 + p_4 & w_2 &= p_5 + p_6 & w_3 &= p_3 + w_1 & w_4 &= p_2 + p_4 \\ w_5 &= p_1 + w_1 & w_6 &= \frac{9}{20}w_3 & c_{11} &= w_6 - \frac{9}{8}w_4 \\ c_{12} &= \frac{9}{10}w_3 & c_{21} &= \frac{27}{10}w_5 - \frac{9}{8}w_2 + \frac{1}{2}c_{11} & c_{22} &= w_6 - \frac{9}{10}w_5 \end{aligned}$$

Table 6: SLP of Eq. (45), $\gamma_{2,1} \approx 12.2034$, with 24 add. and 19 mul.

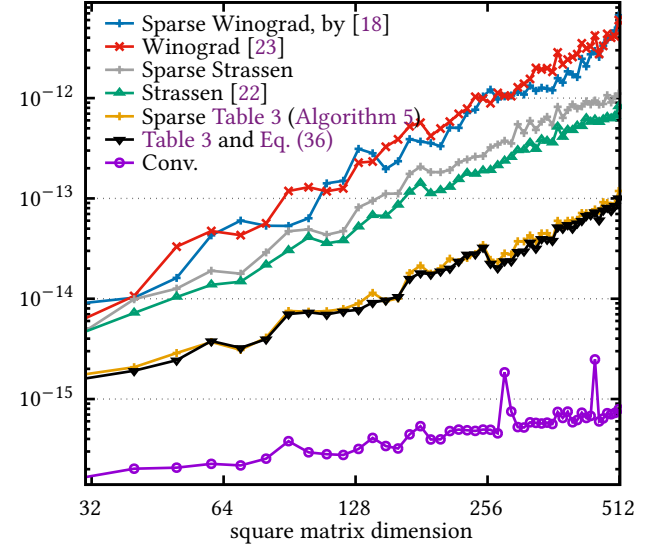
Finally, we present in Eqs. (46) and (47), successive higher-order rational approximations of the point $(\sqrt[4]{4/3}, -1/2)$ reducing the growth factor $\gamma_{2,1}$ to 12.0695 (resp. 12.0661), approaching 12.06603. They then provide rational algorithms whose accuracy is pretty close to our best one, as shown in Figure 1.

$$\begin{bmatrix} -167042 & 295936 & -295936 & -167042 \\ 345665 & 345665 & 345665 & 345665 \\ -178623 & 51622047 & 295936 & 167042 \\ -345665 & -176980480 & 345665 & 345665 \\ 0 & -51622047 & 0 & 334084 \\ -1 & 88490240 & 0 & 345665 \\ 0 & 289 & 0 & 0 \\ 0 & 512 & 0 & 0 \\ -167042 & 289 & 0 & 0 \\ -345665 & 88490240 & 0 & 0 \\ -167042 & 24137569 & 295936 & -167042 \\ -345665 & 88490240 & 345665 & 345665 \\ -167042 & 24137569 & -295936 & 167042 \\ -345665 & 88490240 & 345665 & 345665 \end{bmatrix}; \begin{bmatrix} -256 & -1 & 1 & -256 \\ 289 & 2 & 2 & 289 \\ 0 & 0 & 0 & 0 \\ -345665 & 0 & 345665 & 0 \\ 591872 & 0 & 334084 & 0 \\ -178623 & -1 & 0 & 0 \\ 295936 & 1 & 178623 & 256 \\ 178623 & 2 & 334084 & 289 \\ -289 & 1 & -1 & 256 \\ -1024 & 2 & -2 & 289 \\ -289 & 1 & 1 & -256 \\ -1024 & 2 & 2 & 289 \end{bmatrix}; \begin{bmatrix} 295936 & 295936 & 0 & 0 & 295936 & 295936 & 0 \\ 345665 & 345665 & 345665 & 345665 & 345665 & 345665 & 0 \\ 178623 & -167042 & 1 & 0 & 178623 & 178623 & 0 \\ 345665 & 345665 & 345665 & 345665 & 345665 & 345665 & 0 \\ -178623 & -178623 & 0 & 1 & -178623 & -178623 & 0 \\ -345665 & 345665 & 0 & 1 & 345665 & -345665 & 0 \\ 295936 & 51622047 & 289 & -289 & 51622047 & -31906176129 & -345665 \\ 345665 & 176980480 & 512 & 512 & 176980480 & -102294717440 & -295936 \end{bmatrix}; \begin{bmatrix} -169 & -1 & 1 & -169 \\ -196 & -2 & 2 & -196 \\ 38165 & 0 & 0 & 38165 \\ 33124 & 0 & 0 & 33124 \\ 38165 & 0 & -38165 & 0 \\ 66248 & 1 & 0 & 0 \\ 18957 & 0 & 0 & 0 \\ 33124 & 1 & 0 & 0 \\ 18957 & 1 & 18957 & 169 \\ 66248 & 2 & 38416 & 196 \\ -49 & 1 & -1 & 169 \\ -169 & 2 & -2 & 196 \\ -49 & 1 & 1 & -169 \\ 169 & 2 & 2 & 196 \end{bmatrix}; \begin{bmatrix} -18957 & 19208 & -1 & 0 & -18957 & -18957 & 0 \\ 38165 & 38165 & 38165 & 38165 & 38165 & 38165 & 38165 \\ -33124 & -1857786 & -98 & 98 & -1857786 & 359367849 & 38165 \\ 38165 & 6449885 & 169 & 169 & 6449885 & 1264177460 & 33124 \\ 33124 & 33124 & 0 & 0 & 33124 & 33124 & 0 \\ 38165 & 38165 & 0 & 0 & 38165 & 38165 & 0 \\ -18957 & -18957 & 0 & 1 & 19208 & -18957 & 0 \\ -38165 & -38165 & 0 & 1 & 38165 & -38165 & 0 \end{bmatrix}. \quad (46)$$

A.3 Further numerical experiments

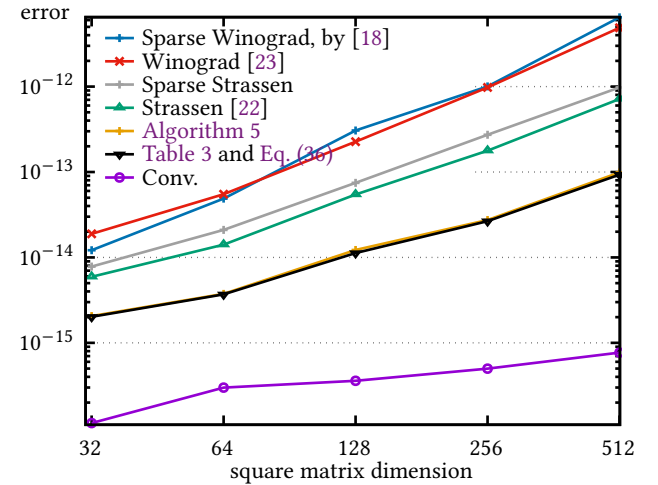
Following [20, § 3.2], we study in Figure 3 the effect of sparsification on random matrix with preassigned singular values and large condition number $\approx 10^{12}$ given by the Matlab function gallery 'randsvd'. The fast variants behavior is unchanged while only the conventional algorithm performs better.

Figure 3: Numerical Effect of Sparsification (large conditioning) error



We here show more evidence on the practical accuracy of the algorithms, with respect to their $\gamma_{2,1}$ growth factor. Figure 4 compares the main possibilities on a uniform $[-1, 1]$ distribution, while Figure 1 was using a normal distribution. The behavior is similar, with again our best variant one or two orders of magnitude more accurate, and being quite close to that of the conventional algorithm.

Figure 4: Numerical accuracy for uniform $[-1, 1]$ distribution



Strassen's algorithm is not optimally accurate

A.4 Cancellation-free search

Algorithm 6 describes a common sub-expression elimination heuristic that reduced the number of operations in our algorithms.

Algorithm 6 Cancellation-free optimization of a linear operator

Input: $M \in \mathbb{K}^{m \times n}$.

Output: A straight-line program computing $x \rightarrow M \cdot x$.

- 1: **repeat** ▷ Precomputing all repeated pairs
 - 2: In each row list all pairs of indices of non-zero coefficients;
 - 3: Among all the rows, find the pair(s) with the maximal number of co-linear representatives;
 - 4: In case of ties, exhaust all the possibilities with maximal pairs (or choose one using a score like that of [6, § 3.2]);
 - 5: Precompute the chosen pair (in a temporary variable);
 - 6: Factor this pair out of all the rows: that is removing the pair from all rows but add a new column to the matrix (representing that pair) with the co-linear multiple of that temporary variable;
 - 7: **until** no pair has more than 1 representative ▷ Multipliers by columns:
 - 8: **for all** equal coefficients in a column (up to sign) **do**
 - 9: Compute the product by the absolute value in a temporary variable;
 - 10: Factor this coefficient out: remove it from the column, add a new column (representing that product) with a ± 1 in the corresponding row(s); ▷ Multipliers by rows:
 - 11: **for all** equal coefficients in a row (up to sign) **do**
 - 12: Compute the sum (or subtraction) of variables with that same coefficients in a temporary variable;
 - 13: Factor the coefficient out: remove it from the row, but add a new column (representing that sum/subtraction) with the coefficient in the same row; ▷ Now the matrix has been simplified
 - 14: Apply the remaining linear operations of the matrix.
-