Some fast algorithms multiplying a matrix by its adjoint

Jean-Guillaume Dumas, Clément Pernet, Alexandre Sedoglavic

Some fast algorithms multiplying a matrix by its adjoint

Jean-Guillaume Dumas[1], Clément Pernet[1], and Alexandre Sedoglavic[2]

[1]Univ. Grenoble Alpes, umr CNRS 5224 LJK
F-38000 Grenoble, France
https://ljk.imag.fr/CAS3C3,
{Jean-Guillaume.Dumas,Clement.Pernet}@univ-grenoble-alpes.fr
[2]Univ. Lille, CNRS, Centrale Lille, umr 9189 CRIStAL
F-59000 Lille, France
http://www.lifl.fr/~sedoglav, Alexandre.Sedoglavic@univ-lille.fr

January 4, 2021

Abstract

We present a non-commutative algorithm for the multiplication of a 2×2 block-matrix by its adjoint, defined by a matrix ring anti-homomorphism. This algorithm uses 5 block products (3 recursive calls and 2 general products)over $\mathbb{C}$ or in positive characteristic. The resulting algorithm for arbitrary dimensions is a reduction of multiplication of a matrix by its adjoint to general matrix product, improving by a constant factor previously known reductions. We prove also that there is no algorithm derived from bilinear forms using only four products and the adjoint of one of them. Second we give novel dedicated algorithms for the complex field and the quaternions to alternatively compute the multiplication taking advantage of the structure of the matrix-polynomial arithmetic involved. We then analyze the respective ranges of predominance of the two strategies. Finally we propose schedules with low memory footprint that support a fast and memory efficient practical implementation over a prime field.

Contents

1 Introduction

Volker Strassen's algorithm [19], with 7 recursive multiplications and 18 additions, was the first sub-cubic time algorithm for matrix product, with a cost of $O\left(n^{2.81}\right)$. Summarizing the many improvements which have happened since then, the cost of multiplying two arbitrary $n \times n$ matrices over a ring $\mathfrak{R}$ will be denoted by $\mathrm{MM}_{\omega}^{\mathfrak{R}}(n) = O(n^{\omega})$ ring operations where $2 < \omega \leq 3$ is any feasible exponent for this operation (see [17] for the best theoretical estimates of ω known to date).

We consider here the computation of the product of a matrix by transpose $A \cdot A^{\intercal}$ or by its conjugate transpose $A \cdot \overline{A}^{\intercal}$, which we handle in a unified way as the product $A \cdot \phi(A)$ where ϕ is a matrix *anti-homomorphism*. In the rest of the paper, $\phi(A)$ will be referred to as the *adjoint* of A. For this computation, the natural divide and conquer algorithm, splitting the matrices in four quadrants, will use 6 block multiplications (as any of the two off-diagonal blocks can be recovered from the other one).We propose instead a new algorithm using only 5 block multiplications, for any antihomomorphism ϕ, provided that the base ring supports the existence of skew unitary matrices.

For this product, the best previously known cost bound was equivalent to $\frac{2}{2^\omega-4}\mathrm{MM}_\omega(n)$ over any field (see [6, § 6.3.1]). With our algorithm, this product can be computed in a cost equivalent to $\frac{2}{2^\omega-3}\mathrm{MM}_\omega(n)$ ring operations when there exists a skew-unitary matrix. Our algorithm is derived from the class of Strassen-like algorithms multiplying 2×2 matrices in 7 multiplications. Yet it is a reduction of multiplying a matrix by its transpose to general matrix multiplication, thus supporting any admissible value for ω. By exploiting the symmetry of the problem, it requires about half of the arithmetic cost of general matrix multiplication when ω is $\log_2 7$.

This paper extends the results of [7] with the following improvements:

1. we generalize the case of the transposition in [7, Algorithm 2] to arbitrary antihomomorphism, including the Hermitian transposition.
2. Our algorithm uses 5 multiplications and the (hermitian) transpose of one these blocks. In [7] a Gröbner basis parameterization is used to search for algorithms, or prove by exhaustive search that there are no better algorithm, *in the Strassen orbit*. We partially address here the more general result that there is no algorithm derived from bilinear forms, with fewer products, by proving the inexistence of an algorithm with four products and the (hermitian) transpose of one of them.
3. In [7] the algorithm is shown to be efficient over $\mathbb{C}$, for a range of matrix multiplication exponents (including all the feasible ones), and for any positive characteristic field, unconditionally. We extend this analysis to the case for the Hermitian transpose: while our five-products algorithm is unusable due to the inexistence of skew unitary matrices over $\mathbb{C}$, we propose a 2M algorithm, adapted from the 3M algorithm for the product of complex matrices.
4. Finally, we propose novel dedicated algorithms for the multiplication of a matrix by its transpose or conjugate transpose over the algebra of quaternions (over $\mathbb{R}$ or any commutative field), improving on the dominant term of the state of the art complexity bounds for these problems.

After a introducing the terminology in Section 2, we will present in Section 3 the main recursive algorithm computing the product of a matrix by its adjoint in 5 block products provided that a skew unitary matrix is given. We survey in Section 4 the most classical instances for the base field to support the existence of skew unitary matrices. We then investigate in Section 5 the minimality of five products for the computing the product of a 2×2 matrix by its hermitian transpose: applying de Groote's technique enables us to state this result partially, for all algorithms using up to one symmetry between a product and its adjoint. Section 6 explores alternative approaches offered by the structure of polynomial arithmetic, when the field is an extension. This includes a new 2M algorithm in Section 6.1 and new algorithms over the algebra of quaternions in Section 6.2. Lastly, we discuss on an implementation of the recursive algorithm for the product of a matrix by its transpose in Section 7.

2 Preliminaries

To unify the notion of transposition and conjugate transposition, we use the formalism of antihomomorphisms and of involutive antihomomorphisms as recalled in the following definitions.

Definition 1. *Let $\mathfrak{R}, \mathfrak{S}$ be two rings, $\phi : \mathfrak{R} \to \mathfrak{S}$ is a* ring antihomomorphism *if and only if, for all (x, y) in $\mathfrak{R} \times \mathfrak{S}$:*

$$\phi(1_{\mathfrak{R}}) = 1_{\mathfrak{S}}, \tag{1a}$$
$$\phi(x + y) = \phi(x) + \phi(y), \tag{1b}$$
$$\phi(xy) = \phi(y)\phi(x). \tag{1c}$$

From this, one can define a matrix antihomomorphism by induction, as shown in Definition 2.

Definition 2. *Over a ring $\mathfrak{R}$, an* involutive matrix antihomomorphism *is a family of applications $\phi_{m,n} : \mathfrak{R}^{m\times n} \to \mathfrak{R}^{n\times m}$ for all (m, n) in $\mathbb{N}^2$ satisfying for additional (ℓ, k) in $\mathbb{N}^2$ and for all A and A' in $\mathfrak{R}^{m\times n}$, for all M in $\mathfrak{R}^{n\times k}$, for all B in $\mathfrak{R}^{m\times k}$, for all C in $\mathfrak{R}^{\ell\times n}$ and for all D in $\mathfrak{R}^{\ell\times k}$ the following relations:*

$$\phi_{m,n} \circ \phi_{n,m} = \mathrm{I}, \tag{2a}$$
$$\phi_{m,n}(A + A') = \phi_{m,n}(A) + \phi_{m,n}(A'), \tag{2b}$$
$$\phi_{m,k}(A \cdot M) = \phi_{n,k}(M) \cdot \phi_{m,n}(A), \tag{2c}$$
$$\phi_{m+\ell,n+k}\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right) = \begin{bmatrix} \phi_{m,n}(A) & \phi_{\ell,n}(C) \\ \phi_{m,k}(B) & \phi_{\ell,k}(D) \end{bmatrix}. \tag{2d}$$

For the convenience, we will denote all applications of this family by ϕ, as the dimensions are clear from the context. This definition implies the following:

Lemma 3. *For all A in $\mathfrak{R}^{m\times n}$ let B, be $\phi(A)$. Then for all suitable (i, j) the coefficient b_{ij} is $\phi(a_{ji})$.*

Proof. By induction, using Equation (2d): if $m = n = 1$, then $\phi(A) = [\phi(a_{11})]$. Then assume the property is true for all $A \in \mathfrak{R}^{m\times n}$ with $m, n \leq N$, and consider a matrix A in $\mathfrak{R}^{(N+1)\times(N+1)}$. Applying Equation (2d) on the block decomposition $A = \begin{bmatrix} A_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ where A_{11} is in $\mathfrak{R}^{N\times N}$ yields the relations:

$$\phi(a) = \begin{bmatrix} \phi(A_{11}) & \phi(a_{21}) \\ \phi(a_{12}) & \phi(a_{22}) \end{bmatrix} = [\phi(a_{ji})]_{ij} \tag{3}$$

by induction hypothesis. The case of matrices in $\mathfrak{R}^{m\times(N+1)}$ and $\mathfrak{R}^{(N+1)\times n}$ is dealt with similarly, using 0-dimensional blocks a_{21} or a_{12} respectively. □

Lemma 4. *For all α in $\mathfrak{R}$ and for all A in $\mathfrak{R}^{m\times n}$, $\phi(\alpha A) = \phi(A)\phi(\alpha)$.*

Proof. By Lemma 3, $\phi(\alpha \mathrm{I}_m) = \phi(\alpha)\mathrm{I}_m = \mathrm{I}_m\phi(\alpha)$. Then by Equation (2c), the relations $\phi(\alpha A) = \phi((\alpha \mathrm{I}_m)A) = \phi(A)\phi(\alpha \mathrm{I}_m) = \phi(A)\phi(\alpha)$ hold. □

The following Lemma 5 shows that Definition 2 is a natural extension of a ring antiendomorphism for matrices.

Lemma 5. *An involutive matrix antihomomorphism is a ring antiendomorphism on its base ring (seen as the ring of* 1×1 *matrices).*

Proof. Equations (2b) and (2c) directly imply Equations (1b) and (1c) respectively when $m = n = k = 1$. Then, we have that $\phi(1) = \phi(1) \cdot 1$. Therefore $\phi(\phi(1)) = \phi(\phi(1) \cdot 1)$ and $1 = \phi(1)\phi(\phi(1)) = \phi(1) \cdot 1$ by Equations (2a) and (2c). This right hand side is equal to that of the first equation, thus proving the equality of the left hand sides and Equation (1a). □

Definition 2 gathers actually all the requirements for our algorithm to work in classical hermitian or non-hermitian cases:

Examples 6. *For matrices over a commutative ring,*

- *the matrix transpose with* $\phi(A) = A^\intercal$ *and*
- *the matrix conjugate transpose,* $\phi(A) = A^H$,

are two examples of matrix anti-homomorphisms. However, for instance, transposition over the quaternions is a counter-example as the non-commutativity implies there that in general $(A \cdot B)^\intercal \neq B^\intercal \cdot A^\intercal$.

Definition 7. *The image* $\phi(A)$ *of a matrix* A *by an antihomomorphism is called the* adjoint *of* A.

Definition 8. *Let* $A \in \mathfrak{R}^{m\times n}$, *we denote respectively by* $Low(A)$ *and* $Up(A)$ *the* $m \times n$ *lower and upper triangular parts of* A, *namely the matrices* L *and* U *verifying*

- $L_{ij} = a_{ij}$ *for* $i \geq j$ *and* $L_{ij} = 0$ *otherwise,*
- $U_{ij} = a_{ij}$ *for* $i \leq j$ *and* $U_{ij} = 0$ *otherwise.*

Lemma 9. *If* $\phi(A) = A$ *in* $\mathfrak{R}^{n\times n}$, *then* $Up(A) = \phi(Low(A))$.

Proof. Applying Lemma 3, the coefficients u_{ij} of $U = \phi(\mathrm{Low}(A))$ for $0 < i \leq j$ satisfy $u_{ij} = \phi(a_{ji})$. Now if $\phi(A) = A$, we have $u_{ij} = a_{ij}$ for $0 < i \leq j$ and $u_{ij} = 0$ otherwise, as $\phi(0) = 0$, by Equation (2b). Hence $U = \mathrm{Up}(A)$. □

Definition 10 (Skew-unitary)**.** *A matrix* Y *in* $\mathfrak{R}^{n\times n}$ *is* skew-unitary *relatively to a matrix antihomomorphism* ϕ *if the following relation holds:*

$$Y \cdot \phi(Y) = -\mathrm{I}_n. \tag{4}$$

For the cost analysis, we will also need the following variant of the Master Theorem, reflecting the constant in the leading term of the computed cost bound.

Lemma 11. *Let $T(n)$ be defined by the recurrence $T(n) = aT(n/2) + b\left(\frac{n}{2}\right)^{\alpha} + o(n^{\alpha})$, where $0 \leq \log_2 a < \alpha$. Then $T(n) = \frac{b}{2^{\alpha}-a} n^{\alpha} + o(n^{\alpha})$.*

Proof.

$$T(n) = a^{\log_2 n} T(1) + \sum_{i=0}^{\log_2(n)-1} a^i b \left(\frac{n}{2^{i+1}}\right)^{\alpha} + o\left(\left(\frac{n}{2^i}\right)^{\alpha}\right)$$

$$= n^{\log_2 a} T(1) + \frac{b}{2^{\alpha}} n^{\alpha} \sum_{i=0}^{\log_2(n)-1} \left(\frac{a}{2^{\alpha}}\right)^i + o(n^{\alpha}) = \frac{b}{2^{\alpha}-a} n^{\alpha} + o(n^{\alpha}).$$

□

3 An algorithm for the product of a matrix by its adjoint with five multiplications

We now show how to compute the product of a matrix by its adjoint with respect to an involutive antihomomorphism in only 5 recursive multiplications and 2 multiplications by any skew-unitary matrix. This is a generalization of [7, Algorithm 2] for any involutive antihomomorphism.

We next give Algorithm 12 for even dimensions. In case of odd dimensions, padding or static/dynamic peeling can always be used [3].

Theorem 13. *Algorithm 12 is correct. Moreover, if any two $n \times n$ matrices over a ring $\mathfrak{R}$ can be multiplied in $MM_{\omega}^{\mathfrak{R}}(n) = O(n^{\omega})$ ring operations for $\omega > 2$, and if there exist a skew-unitary matrix which can be multiplied to any other matrix in $o(n^{\omega})$ ring operations then Algorithm 12 requires fewer than $\frac{2}{2^{\omega}-3} MM_{\omega}^{\mathfrak{R}}(n) + o(n^{\omega})$ ring operations.*

Proof. For the cost analysis, Algorithm 12 is applied recursively to compute three products P_1, P_2 and P_7, while P_4 and P_5 are computed in $\mathrm{MM}_{\omega}^{\mathfrak{R}}(n)$ using the general matrix multiplication algorithm. The second hypothesis is that applying the skew-unitary matrix Y to a $n \times n$ matrix costs $Y(n) = o(n^{\omega})$. Then applying Remark 15 thereafter, the cost $T(n)$ of Algorithm 12 satisfies:

$$T(n) \leq 3T(n/2) + 2\mathrm{MM}_{\omega}^{\mathfrak{R}}(n/2) + 2Y(n) + (7.5)(n/2)^2 + o(n^2) \tag{5}$$

and $T(4)$ is a constant. Thus, by Lemma 11:

$$T(n) \leq \frac{2}{2^{\omega}-3} \mathrm{MM}_{\omega}^{\mathfrak{R}}(n) + o(n^{\omega}). \tag{6}$$

Now for the correction, by Equation (2d), we have to show that the result of Algorithm 12 is indeed:

$$\mathrm{Low}\left(A \cdot \phi\left(A\right)\right) = \mathrm{Low}\left(A \cdot \begin{pmatrix} \phi(A_{11}) & \phi(A_{21}) \\ \phi(A_{12}) & \phi(A_{22}) \end{pmatrix}\right)$$

$$= \begin{pmatrix} \mathrm{Low}(A_{11}\cdot\phi(A_{11}) + A_{12}\cdot\phi(A_{12})) & \times \\ A_{21}\cdot\phi(A_{11}) + A_{22}\cdot\phi(A_{12}) & \mathrm{Low}(A_{21}\cdot\phi(A_{21}) + A_{22}\cdot\phi(A_{22})) \end{pmatrix}$$

Algorithm 12 Product of a matrix by its adjoint

Input: $A \in \mathfrak{R}^{m\times n}$ (with even m and n for the sake of simplicity);
Input: ϕ an involutive matrix antihomomorphism;
Input: $Y \in \mathfrak{R}^{\frac{n}{2}\times\frac{n}{2}}$ skew-unitary for ϕ.
Output: $\mathrm{Low}\,(A\cdot\phi\,(A))$.

Split $A = \left(\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22}\end{smallmatrix}\right)$ where A_{11} is in $\mathfrak{R}^{\frac{m}{2}\times\frac{n}{2}}$

▷ 4 *additions and 2 multiplications by* Y*:*

$S_1 \leftarrow (A_{21} - A_{11})\cdot Y$
$S_2 \leftarrow A_{22} - A_{21}\cdot Y$
$S_3 \leftarrow S_1 - A_{22}$
$S_4 \leftarrow S_3 + A_{12}$

▷ 3 *recursive* (P_1, P_2, P_5) *and* 2 *general products* (P_3, P_4)*:*

$\mathrm{Low}\,(P_1) \leftarrow \mathrm{Low}\,(A_{11}\cdot\phi\,(A_{11}))$
$\mathrm{Low}\,(P_2) \leftarrow \mathrm{Low}\,(A_{12}\cdot\phi\,(A_{12}))$
$P_3 \leftarrow A_{22}\cdot\phi\,(S_4)$
$P_4 \leftarrow S_1\cdot\phi\,(S_2)$
$\mathrm{Low}\,(P_5) \leftarrow \mathrm{Low}\,(S_3\cdot\phi\,(S_3))$

▷ 3 *half additions and* 2 *complete additions:*

$\mathrm{Low}\,(U_1) \leftarrow \mathrm{Low}\,(P_1) + \mathrm{Low}\,(P_5)$
$\mathrm{Low}\,(U_3) \leftarrow \mathrm{Low}\,(P_1) + \mathrm{Low}\,(P_2)$
$\mathrm{Up}(U_1) \leftarrow \phi\,(\mathrm{Low}\,(U_1))$ ▷ *Forms the full matrix* U_1
$U_2 \leftarrow U_1 + P_4$,
$U_4 \leftarrow U_2 + P_3$,
$\mathrm{Low}\,(U_5) \leftarrow \mathrm{Low}\,(U_2) + \mathrm{Low}\,(\phi\,(P_4))$.

return $\left(\begin{smallmatrix} \mathrm{Low}(U_3) & \\ U_4 & \mathrm{Low}(U_5)\end{smallmatrix}\right)$.

First, we have that:

$$\mathrm{Low}\,(U_3) = \mathrm{Low}\,(P_1) + \mathrm{Low}\,(P_2) = \mathrm{Low}\,(A_{11}\cdot\phi\,(A_{11}) + A_{12}\cdot\phi\,(A_{12})). \quad (7)$$

Second, as Y is skew-unitary, then we have that $Y\cdot\phi\,(Y) = -\mathrm{I}_{\frac{n}{2}}$. Also, by Equations (2b) and (2c), $\phi\,(S_2) = \phi\,(A_{22}) - \phi\,(Y)\cdot\phi\,(A_{21})$. Then, denote by R_1 the product:

$$\begin{aligned} R_1 = A_{11}\cdot Y\cdot\phi\,(S_2) &= A_{11}\cdot Y\cdot(\phi\,(A_{22}) - \phi\,(Y)\cdot\phi\,(A_{21})) \\ &= A_{11}\cdot(Y\cdot\phi\,(A_{22}) + \phi\,(A_{21})). \end{aligned} \quad (8)$$

Further, by Equations (2a) and (2c), we have that $P_1 = A_{11}\cdot\phi\,(A_{11})$, $P_2 = A_{12}\cdot\phi\,(A_{12})$, and $P_5 = S_3\cdot\phi\,(S_3)$ are invariant under the action of ϕ. So are therefore, $U_1 = P_1 + P_5$, $U_3 = P_1 + P_2$ and $U_5 = U_1 + (P_4 + \phi\,(P_4))$. By Lemma 9, it suffices to compute $\mathrm{Low}\,(U_1)$ and, if needed, we also have $\mathrm{Up}\,(U_1) = \phi\,(\mathrm{Low}\,(U_1))$.

Then, as $S_3 = S_1 - A_{22} = (A_{21} - A_{11})\cdot Y - A_{22} = -S_2 - A_{11}\cdot Y$, and

$\phi(\phi(S_2)) = S_2$ by Equation (2a), we have that:

$$\begin{aligned} U_1 = P_1 + P_5 &= A_{11} \cdot \phi(A_{11}) + S_3 \cdot \phi(S_3) \\ &= A_{11} \cdot \phi(A_{11}) + (S_2 + A_{11} \cdot Y) \cdot (\phi(S_2) + \phi(Y) \cdot \phi(A_{11})) \\ &= S_2 \cdot \phi(S_2) + \phi(R_1) + R_1. \end{aligned} \tag{9}$$

Also, denote $R_2 = A_{21} \cdot Y \cdot \phi(A_{22})$, so that:

$$\begin{aligned} S_2 \cdot \phi(S_2) &= (A_{22} - A_{21} \cdot Y) \cdot (\phi(A_{22}) - \phi(Y) \cdot \phi(A_{21})) \\ &= A_{22} \cdot \phi(A_{22}) - A_{21} \cdot \phi(A_{21}) - R_2 - \phi(R_2). \end{aligned} \tag{10}$$

Furthermore, from Equation (8):

$$\begin{aligned} R_1 + P_4 &= R_1 + S_1 \cdot \phi(S_2) \\ &= R_1 + (A_{21} - A_{11}) \cdot Y \cdot (\phi(A_{22}) - \phi(Y) \cdot \phi(A_{21})) \\ &= A_{21} \cdot Y \cdot \phi(A_{22}) + A_{21} \cdot \phi(A_{21}) = R_2 + A_{21} \cdot \phi(A_{21}). \end{aligned} \tag{11}$$

This shows, from Equations (9) to (11), that:

$$\begin{aligned} U_5 = U_1 + P_4 + \phi(P_4) &= S_2 \cdot \phi(S_2) + \phi(R_1) + R_1 + P_4 + \phi(P_4) \\ &= A_{22} \cdot \phi(A_{22}) + (-1+2) A_{21} \cdot \phi(A_{21}). \end{aligned} \tag{12}$$

Third, the last coefficient U_4 of the result is obtained from Equations (11) and (12):

$$\begin{aligned} U_4 &= U_2 + P_3 = U_1 + P_4 + P_3 \\ &= A_{22} \cdot \phi(A_{22}) - A_{21} \cdot \phi(A_{21}) - R_2 - \phi(R_2) + R_1 + \phi(R_1) + P_4 + P_3 \\ &= A_{22} \cdot \phi(A_{22}) - \phi(R_2) + \phi(R_1) + P_3 \end{aligned} \tag{13}$$

since by Equation (11), $R_1 + P_4 = R_2 + A_{21} \cdot \phi(A_{21})$. Now

$$\begin{aligned} P_3 = A_{22} \cdot \phi(S_4) &= A_{22} \cdot \phi((A_{21} - A_{11}) \cdot Y + A_{12} - A_{22}) \\ &= \phi(R_2) - \phi(R_1) + A_{21} \cdot \phi(A_{11}), \end{aligned} \tag{14}$$

Hence

$$U_4 = A_{22} \cdot \phi(A_{12}) + A_{21} \cdot \phi(A_{11})$$

□

To our knowledge, the best previously known result was with a $\frac{2}{2^\omega - 4}$ factor instead, see e.g. [6, § 6.3.1]. Table 1 summarizes the arithmetic complexity bound improvements.

Examples 14. *In many cases, applying the skew-unitary matrix Y to a $n \times n$ matrix costs only yn^2 for some constant y depending on the base ring. If the ring is the complex field $\mathbb{C}$ or satisfies the conditions of Proposition 16, there is a square root i of -1. Setting $Y = i\,\mathrm{I}_{n/2}$ yields $Y(n) = n^2$. Otherwise, we*

Problem	Alg.	$O(n^3)$	$O(n^{\log_2(7)})$	$O(n^\omega)$
$A \cdot \phi(A) \in \mathbb{F}^{n\times n}$	[6]	n^3	$\frac{2}{3}\,\mathrm{MM}_{\log_2(7)}(n)$	$\frac{2}{2^\omega-4}\,\mathrm{MM}_\omega(n)$
	Alg. 12	$\mathbf{0.8n^3}$	$\mathbf{\frac{1}{2}\,MM_{\log_2(7)}(n)}$	$\mathbf{\frac{2}{2^\omega-3}\,MM_\omega(n)}$

Table 1: Arithmetic complexity bounds leading terms.

show in Section 4 that in characteristic $p \equiv 3 \bmod 4$, Proposition 17 produces Y equal to $\left(\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right) \otimes \mathrm{I}_{n/2}$ for which $Y(n) = 3n^2$. As a sub-case, the latter can be improved when $p \equiv 3 \bmod 8$: then, Lemma 18 shows that -2 is a square. Therefore, in this case set $a = 1$ and $b \equiv \sqrt{-2} \bmod p$ such that one multiplication is saved. Then the relation $a^2 + b^2 = -1$ there yields $Y = \left(\begin{smallmatrix} 1 & \sqrt{-2} \\ -\sqrt{-2} & 1 \end{smallmatrix}\right) \otimes \mathrm{I}_{n/2}$ for which $Y(n) = 2n^2$.

Remark 15. *Each recursive level of Algorithm 12 is composed of 9 block additions. An exhaustive search on all symmetric algorithms in the orbit of that of Strassen (via a Gröbner basis parameterization [7]) showed that this number is minimal in this class of algorithms. Note also that 3 out of these 9 additions in Algorithm 12 involve symmetric matrices and are therefore only performed on the lower triangular part of the matrix. Overall, the number of scalar additions is $6n^2 + 3/2n(n+1) = 15/2n^2 + 1.5n$, nearly half of the optimal in the non-symmetric case [5, Theorem 1].*

To further reduce the number of additions, a promising approach is that undertaken in [15, 2]. This is however not clear to us how to adapt our strategy to their recursive transformation of basis.

4 Rings with skew unitary matrices

Algorithm 12 requires a skew-unitary matrix. Unfortunately there are no skew-unitary matrices over $\mathbb{R}$, nor $\mathbb{Q}$ for ϕ the transposition, nor over $\mathbb{C}$ for ϕ the Hermitian transposition (there -1 cannot be a sum of real squares for a diagonal element of $Y \cdot \phi(Y)$). Hence, Algorithm 12 provides no improvement in these cases. In other domains, the simplest skew-unitary matrices just use a square root of -1 while others require a sum of squares.

4.1 Over the complex field

Algorithm 12 is thus directly usable over $\mathbb{C}^{n\times n}$ with $\phi(A) = A^\intercal$ and $Y = i\,\mathrm{I}_{\frac{n}{2}}$ in $\mathbb{C}^{\frac{n}{2}\times\frac{n}{2}}$. When complex numbers are represented in Cartesian form, as a pair of real numbers, the multiplications by $Y = i\,\mathrm{I}_{\frac{n}{2}}$ are essentially free since they just exchange the real and imaginary parts, with one sign flip.

As mentioned, for the conjugate transposition, $\phi(A) = A^H$, on the contrary, there are no candidate skew-unitary matrices and we for now report no im-

provement in this case using this approach (but another one does as shown in Section 6.1).

Now, even though over the complex the product of a matrix by its *conjugate* transpose is more widely used, there are some applications for the product of a matrix by its transpose, see for instance [1]. This is reflected in the BLAS API, where both routines `zherk` and `zsyrk` are offered.

4.2 Rings where negative one is a square and $\phi(A) = A^\intercal$

Over some rings , square roots of -1 can also be elements of the base field, denoted i in $\mathfrak{R}$ again. There, Algorithm 12 only requires some pre-multiplications by this square root (with also $Y = i\,\mathrm{I}_{\frac{n}{2}} \in \mathfrak{R}^{\frac{n}{2}\times\frac{n}{2}}$), but *within the ring*.

Further, when the ring is a field in positive characteristic, the existence of a square root of minus one can be characterized, as shown in Proposition 16, thereafter.

Proposition 16. *Fields with characteristic two, p satisfying $p \equiv 1 \bmod 4$, or finite fields that are an even extension, contain a square root of -1.*

Proof. If $p = 2$, then $1 = 1^2 = -1$. If $p \equiv 1 \bmod 4$, then half of the non-zero elements x in the base field of size p satisfy $x^{\frac{p-1}{4}} \neq \pm 1$ and then the square of the latter must be -1. If the finite field $\mathbb{F}$ is of cardinality p^{2k}, then, similarly, there exists elements $x^{\frac{p^k-1}{2}\frac{p^k+1}{2}}$ different from ± 1 and then the square of the latter must be -1. □

4.3 Any field with positive characteristic and $\phi(A) = A^\intercal$

Actually, we show that Algorithm 12 can also be run without any field extension, even when -1 is not a square: form the skew-unitary matrices constructed in Proposition 17, thereafter, and use them directly as long as the dimension of Y is even. Whenever this dimension is odd, it is always possible to pad with zeroes so that $A \cdot A^\intercal = \left(\begin{smallmatrix} A & 0 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} A^\intercal \\ 0 \end{smallmatrix}\right)$.

Proposition 17. *Let $\mathbb{F}_{p^k}$ be a field of characteristic p, there exists (a, b) in $\mathbb{F}_p^2$ such that the matrix:*

$$\left(\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right) \otimes \mathrm{I}_n = \left(\begin{smallmatrix} a\,\mathrm{I}_n & b\,\mathrm{I}_n \\ -b\,\mathrm{I}_n & a\,\mathrm{I}_n \end{smallmatrix}\right) \quad \text{in} \quad \mathbb{F}_p^{2n\times 2n} \tag{15}$$

is skew-unitary for the transposition.

Proof. Using the relation

$$\left(\begin{smallmatrix} a\,\mathrm{I}_n & b\,\mathrm{I}_n \\ -b\,\mathrm{I}_n & a\,\mathrm{I}_n \end{smallmatrix}\right)\left(\begin{smallmatrix} a\,\mathrm{I}_n & b\,\mathrm{I}_n \\ -b\,\mathrm{I}_n & a\,\mathrm{I}_n \end{smallmatrix}\right)^\intercal = (a^2 + b^2)\,\mathrm{I}_{2n}, \tag{16}$$

it suffices to prove that there exist a, b such that $a^2 + b^2 = -1$. In characteristic 2, $a = 1, b = 0$ is a solution as $1^2 + 0^2 = -1$. In odd characteristic, there are $\frac{p+1}{2}$ distinct square elements ${x_i}^2$ in the base prime field. Therefore, there are $\frac{p+1}{2}$ distinct elements $-1 - {x_i}^2$. But there are only p distinct elements in the base field, thus there exists a couple (i, j) such that $-1 - {x_i}^2 = {x_j}^2$ [18, Lemma 6]. □

To further improve the running time of multiplications by a skew-unitary matrix in this case, one could set one of the squares to be 1. This is possible if -2 is a square, for instance when $p \equiv 3 \bmod 8$:

Lemma 18. *If $p \equiv 3 \bmod 8$ then -2 is a square modulo p.*

Proof. Using Legendre symbol, $\left(\frac{-2}{p}\right) = \left(\frac{-1}{p}\right)\left(\frac{2}{p}\right) = (-1)^{\frac{p-1}{2}}(-1)^{\frac{p^2-1}{8}} = (-1)(-1) = 1$ □

Now, Proposition 17 shows that skew-unitary matrices do exist for any field with positive characteristic. For Algorithm 12, we need to build them mostly for $p \equiv 3 \bmod 4$ (otherwise use Proposition 16).

For this, without the extended Riemann hypothesis (ERH), it is possible to use the decomposition of primes into squares:

1. Compute by enumeration a prime $r = 4pk + (3-1)p - 1$, so that both relations $r \equiv 1 \bmod 4$ and $r \equiv -1 \bmod p$ hold;
2. Thus, the methods of [4] allow one to decompose any prime into squares and give a couple (a, b) in $\mathbb{Z}^2$ such that $a^2 + b^2 = r$. Finally, this gives $a^2 + b^2 \equiv -1 \bmod p$.

By the prime number theorem the first step is polynomial in $\log(p)$, as is the second step (square root modulo a prime, denoted `sqrt`, has a cost close to exponentiation and then the rest of Brillhart's algorithm is GCD-like). In practice, though, it is faster to use the following Algorithm 19, even though the latter has a better asymptotic complexity bound only if the ERH is true.

Algorithm 19 `SoS`: Sum of squares decomposition over a finite field

Input: $p \in \mathbb{P}\backslash\{2\}$, $k \in \mathbb{Z}$.
Output: $(a, b) \in \mathbb{Z}^2$, s.t. $a^2 + b^2 \equiv k \bmod p$.

1: **if** $\left(\frac{k}{p}\right) = 1$ **then** ▷ *k is a square mod p*
2: **return** $(\texttt{sqrt}(k), 0)$.
3: **else** ▷ *Find smallest quadratic non-residue*
4: $s \leftarrow 2$; **while** $\left(\frac{s}{p}\right) == 1$ **do** $s \leftarrow s + 1$
5: $c \leftarrow \texttt{sqrt}(s-1)$ ▷ *$s-1$ must be a square*
6: $r \leftarrow ks^{-1} \bmod p$
7: $a \leftarrow \texttt{sqrt}(r)$ ▷ *Now $k \equiv a^2 s \equiv a^2(1+c^2) \bmod p$*
8: **return** $(a, ac \bmod p)$

Proposition 20. *Algorithm 19 is correct and, under the ERH, runs in expected time $\widetilde{O}(\log^3(p))$.*

Proof. If k is square then the square of one of its square roots added to the square of zero is a solution. Otherwise, the lowest quadratic non-residue (LQNR)

modulo p is one plus a square b^2 (1 is always a square so the LQNR is larger than 2). For any generator of $\mathbb{Z}_p$, quadratic non-residues, as well as their inverses (s is invertible as it is non-zero and p is prime), have an odd discrete logarithm. Therefore the multiplication of k and the inverse of the LQNR must be a square a^2. This means that the relation $k = a^2(1+b^2) = a^2 + (ab)^2$ holds.

Now for the running time, under the ERH, [20, Theorem 6.35] shows that the LQNR should be lower than $3\log^2(p)/2 - 44\log(p)/5 + 13$. From this, the expected number of Legendre symbol computations is $O(\log^2(p))$ and this dominates the modular square root computations. □

Remark 21. *Another possibility is to use randomization: instead of using the lowest quadratic non-residue (*LQNR*), randomly select a non-residue s, and then decrement it until $s-1$ is a quadratic residue (1 is a square so this will terminate). In practice, the running time seems very close to that of Algorithm 19 anyway, see, e.g. the implementation in Givaro rev. 7bdefe6,* `https://github.com/linbox-team/givaro`*. Also, when computing t sum of squares modulo the same prime, one can compute the* LQNR *only once to get all the sum of squares with an expected cost bounded by $\widetilde{O}(\log^3(p) + t\log^2(p))$.*

Remark 22. *Except in characteristic 2 or in algebraic closures, where every element is a square anyway, Algorithm 19 is easily extended over any finite field: compute the* LQNR *in the base prime field, then use Tonelli-Shanks or Cipolla-Lehmer algorithm to compute square roots in the extension field.*

Denote by $\mathtt{SoS}(q,k)$ this algorithm decomposing k as a sum of squares within any finite field $\mathbb{F}_q$. This is not always possible over infinite fields, but there Algorithm 19 still works anyway for the special case $k=-1$: just run it in the prime sub-field, since -1 must be in it.

4.4 Finite fields with even extension and $\phi(A) = A^H$

With $\phi(A) = A^H$, we need a matrix Y such that $Y \cdot Y^H = Y \cdot \overline{Y}^{\mathsf{T}} = -\mathrm{I}$. This is not possible anymore over the complex field, but works for any even extension field, thanks to Algorithm 19. To see this, we consider next the finite field $\mathbb{F}_{q^2}$, where q is a power of an arbitrary prime. Given $a \in \mathbb{F}_{q^2}$, we adopt the convention that conjugation is given by the Frobenius automorphism:

$$\overline{a} = a^q. \tag{17}$$

The bar operator is $\mathbb{F}_q$-linear and has order 2 on $\mathbb{F}_{q^2}$.

First, if -1 is a square in $\mathbb{F}_q$, then $Y = \sqrt{-1}\cdot \mathrm{I}_n$ works in $\mathbb{F}_{q^2}$ since then $\overline{\sqrt{-1}} = \sqrt{-1}$: $Y \cdot \overline{Y}^{\mathsf{T}} = \sqrt{-1}\cdot\mathrm{I}_n\sqrt{-1}\cdot\mathrm{I}_n = -\mathrm{I}_n$.

Second, otherwise, $q \equiv 3 \bmod 4$ and then there exists a square root i of -1 in $\mathbb{F}_{q^2}$, from Proposition 16. Further, one can build (a,b), both in the base field $\mathbb{F}_q$, such that $a^2+b^2=-1$, from Algorithm 19. Finally $Y = (a+ib)\cdot\mathrm{I}_n$ in ${\mathbb{F}_{q^2}}^{n\times n}$ is skew-unitary: indeed, since $q \equiv 3 \bmod 4$, we have that $i^q = i^{3+4k} = i^3(-1)^{2k} = -i$ and, therefore, $\overline{a+ib} = (a+ib)^q = a - ib$. Finally $Y\cdot\overline{Y}^{\mathsf{T}} = (a+ib)(a-ib)\cdot\mathrm{I}_n = -\mathrm{I}_n$.

4.5 Any field with positive characteristic and $\phi(A) = A^H$

If -1 is a square in the base field, or within an even extension we have seen in Section 4.4 that there exists diagonal skew-unitary matrices. Otherwise, one can always resort to tridiagonal ones as in Section 4.3. For this, one can always build (a, b) in the base field such that $a^2 + b^2 = -1$ using Proposition 17. Then, $Y = \left(\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right) \otimes \mathrm{I}_n$ is a skew-unitary matrix. Indeed, since a and b live in the base field, they are invariant by the Frobenius automorphism. Therefore, $\overline{Y}^{\mathsf{T}} = \overline{\left(\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right)}^{\mathsf{T}} \otimes \mathrm{I}_n = \left(\begin{smallmatrix} a & -b \\ b & a \end{smallmatrix}\right) \otimes \mathrm{I}_n$ and $Y \cdot \overline{Y}^{\mathsf{T}} = (a^2 + b^2) \cdot \mathrm{I}_n = -\mathrm{I}_n$.

5 Towards a minimality result on the number of multiplications

Our Algorithm 12 computes the product of a matrix over a ring by its (hermitian) transpose using only 5 block multiplications and the (hermitian) transpose of one of these block multiplications. Here, we use consider some vector-spaces and thus, restrict ourselves to consider matrices over a field.

We reformulate in this section the method introduced by de Groote in [10] in order to prove that the tensor rank of the 2×2 matrix product is 7. This method is used to prove the following result:

Theorem 23. *There is no algorithm derived from non-commutative block 2×2 matrix product algorithms that computes the product of a matrix over a field by its (hermitian) transpose using only 4 block multiplications and the (hermitian) transpose of one of these block multiplications.*

This result does not state that it is never possible to multiply by the adjoint using fewer than 5 multiplications as shown by the following remark.

Remark 24. *Over any ring with a square root i of -1, there is a computational scheme requiring 4 multiplications and computing the product of a 2×2-matrix by its transpose:*

$$\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)\left(\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}\right) = \begin{pmatrix} (a+ib)(a-ib) & \times \\ ac+bd & (c+id)(c-id) \end{pmatrix} = \begin{pmatrix} a^2+b^2 & \times \\ ac+bd & c^2+d^2 \end{pmatrix}. \tag{18}$$

This is the case for instance over $\mathbb{F}_2$, where $i = 1$, or over the complex numbers. As this scheme requires for instance that $aib = iba$, at least some commutativity is required, thus in general it does not apply to block matrices and it is therefore not in the scope of Theorem 23.

The following section is devoted to shortly present the framework used in this part of our work.

5.1 The framework of bilinear maps encoded by tensors

We present de Groote's proof using a tensorial presentation of bilinear maps; we recall briefly this standpoint through the following well-known example of

seven multiplications and we refer to [16] for a complete introduction to this framework.

Example 25. *Considered as* 2×2 *matrices, the matrix product* $C = A \cdot B$ *could be computed using Strassen algorithm by performing the following computations (see [19]):*

$$\begin{array}{ll} \rho_1 \leftarrow a_{11}(b_{12} - b_{22}), & \\ \rho_2 \leftarrow (a_{11} + a_{12})b_{22}, & \rho_4 \leftarrow (a_{12} - a_{22})(b_{21} + b_{22}), \\ \rho_3 \leftarrow (a_{21} + a_{22})b_{11}, & \rho_5 \leftarrow (a_{11} + a_{22})(b_{11} + b_{22}), \\ \rho_6 \leftarrow a_{22}(b_{21} - b_{11}), & \rho_7 \leftarrow (a_{21} - a_{11})(b_{11} + b_{12}), \end{array} \tag{19}$$

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} \rho_5 + \rho_4 - \rho_2 + \rho_6 & \rho_6 + \rho_3 \\ \rho_2 + \rho_1 & \rho_5 + \rho_7 + \rho_1 - \rho_3 \end{pmatrix}.$$

With m, n, p *equal to* 2*, this algorithm encodes a bilinear map:*

$$\begin{array}{ccc} \mathbb{F}^{m\times n} \times \mathbb{F}^{n\times p} & \to & \mathbb{F}^{m\times p}, \\ (A, B) & \to & A \cdot B. \end{array} \tag{20}$$

We keep the indices m, n, p *in this section for the sake of clarity in order to distinguish the different spaces involved in the sequel. The spaces* $\mathbb{F}^{\cdot\times\cdot}$ *can be endowed with the Frobenius product* $\langle M, N\rangle = \mathrm{Trace}(M^\intercal \cdot N)$ *that establishes an isomorphism between* $\mathbb{F}^{\cdot\times\cdot}$ *and its dual space* $\left(\mathbb{F}^{\cdot\times\cdot}\right)^\star$*; hence, it allows for example to associate the trilinear form* $\mathrm{Trace}(C^\intercal \cdot A \cdot B)$ *and the matrix multiplication (20):*

$$\begin{array}{rccc} \mathcal{S}|_3 : & \mathbb{F}^{m\times n} \times \mathbb{F}^{n\times p} \times (\mathbb{F}^{m\times p})^\star & \to & \mathbb{F}, \\ & (A, B, C^\intercal) & \to & \langle C, A \cdot B\rangle. \end{array} \tag{21}$$

As by construction, the space of trilinear forms is the canonical dual space of order three tensor products, we could encode the Strassen multiplication algorithm (19) as the tensor $\mathcal{S}$ *defined by:*

$$\begin{aligned} \mathcal{S} = \textstyle\sum_{i=1}^{7} \Sigma_1^i \otimes \Sigma_2^i \otimes S_i^3 = \Sigma_1^i \otimes \Sigma_2^i \otimes S_i^3 = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 & 1 \\ 0 & -1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right) + \\ \left(\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} -1 & 0 \\ 1 & 0 \end{smallmatrix}\right) + \left(\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 & 1 \\ 0 & -1 \end{smallmatrix}\right) + \left(\begin{smallmatrix} 0 & 1 \\ 0 & -1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right) + \\ \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) + \left(\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} -1 & 0 \\ 1 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right) + \left(\begin{smallmatrix} -1 & 0 \\ 1 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right) \otimes \left(\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right) \end{aligned} \tag{22}$$

in $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star \otimes \mathbb{F}^{m\times p}$ *with* $m = n = p = 2$.

Remark that—as introduced in the above Equation (22)—we are going to use in the sequel the *Einstein summation convention* in order to simplify the forthcoming notations (according to this convention, when an index variable appears twice in a term and is not otherwise defined, it represents in fact the sum of that term over all the values of the index).

Starting from the tensor representation $\mathcal{S}$ of our algorithm, we could consider several *contractions* that are the main objects manipulated in the sequel.

5.2 Flattening tensors and isotropies

The *complete* contraction $\mathcal{S}|_3(A \otimes B \otimes C^\intercal)$ is defined as the following map:

$$\begin{array}{c}((\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star \otimes \mathbb{F}^{m\times p}) \otimes (\mathbb{F}^{m\times n} \otimes \mathbb{F}^{n\times p} \otimes (\mathbb{F}^{m\times p})^\star) \to \mathbb{F},\\ \big(\Sigma_1^i \otimes \Sigma_2^i \otimes S_i^3\big) \otimes (A \otimes B \otimes C^\intercal) \to \langle \Sigma_1^i, A\rangle \langle \Sigma_2^i, B\rangle \langle S_i^3, C^\intercal\rangle.\end{array} \tag{23}$$

We already saw informally in the previous section that this complete contraction is $\mathrm{Trace}(A \cdot B \cdot C)$ and we recall in the following remark some of its basic properties.

Remark 26. *Given three invertible matrices:*

$$\alpha \in \mathbb{F}^{m\times m}, \quad \beta \in \mathbb{F}^{p\times p}, \quad \gamma \in \mathbb{F}^{n\times n} \tag{24}$$

that encodes changes of basis, the trace $\mathrm{Trace}(A \cdot B \cdot C)$ *is equal to:*

$$\begin{array}{l}\mathrm{Trace}\big((A \cdot B \cdot C)^\intercal\big) = \mathrm{Trace}(C \cdot A \cdot B) = \mathrm{Trace}(B \cdot C \cdot A),\\ \textit{and } \mathrm{Trace}\big(\alpha^{-1} \cdot A \cdot \beta \cdot \beta^{-1} \cdot B \cdot \gamma \cdot \gamma^{-1} \cdot C \cdot \alpha\big).\end{array} \tag{25}$$

These relations illustrate the following theorem:

Theorem 27 ([12, § 2.8])**.** *The isotropy group of the* $n \times n$ *matrix multiplication tensor is* $\mathrm{PSL}^{\pm}(\mathbb{F}^n)^{\times 3} \rtimes \mathfrak{S}_3$*, where* PSL *stands for the group of matrices of determinant* ± 1 *and* $\mathfrak{S}_3$ *for the symmetric group on* 3 *elements.*

The following classical statement redefines the *sandwiching* isotropy on a matrix multiplication tensor:

Definition 28. *Given* $\mathsf{g} = (\alpha \times \beta \times \gamma)$ *in* $\mathrm{PSL}^{\pm}(\mathbb{F}^n)^{\times 3}$*, its action* $\mathsf{g} \diamond \mathcal{S}$ *on a tensor* $\mathcal{S}$ *is given by* $\mathsf{g} \diamond (\Sigma_1^i \otimes \Sigma_2^i \otimes S_i^3)$ *where each summands is equal to:*

$$\big((\alpha^{-1})^\intercal \cdot \Sigma_1^i \cdot \beta^\intercal\big) \otimes \big((\beta^{-1})^\intercal \cdot \Sigma_2^i \cdot \gamma^\intercal\big) \otimes \big((\gamma^{-1})^\intercal \cdot S_i^3 \cdot \alpha^\intercal\big), \ \forall i \textit{ fixed.} \tag{26}$$

These isotropies will be used later; for the moment, let us now focus our attention on the very specific standpoint on which is based the forthcoming developments: flattenings.

Definition 29. *Given a tensor* $\mathcal{S}$*, the third flattening* $\mathcal{S}|_3^1$ *(a.k.a. third* 1*-contraction) of the tensor* $\mathcal{S}$ *is:*

$$\begin{array}{rccc}\mathcal{S}|_3^1 : & \mathbb{F}^{m\times p} & \to & (\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star,\\ & M & \to & \langle M, S_i^3\rangle\, \Sigma_1^i \otimes \Sigma_2^i.\end{array} \tag{27}$$

Example 30. *To illustrate this definition and some important technicalities, let us consider* $Im\,\mathcal{S}|_3^1$ *the image of the Strassen tensor (22) flattening: this is a subspace of* $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star$*. More precisely, let us first consider only the fifth summand in Equation (22) and the image of its third flattening:*

$$Im\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)\otimes\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)\otimes\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)|_3^1 = \begin{pmatrix} c_{11}+c_{22} & 0\ 0 & c_{11}+c_{22}\\ 0 & 0\ 0 & 0\\ 0 & 0\ 0 & 0\\ c_{11}+c_{22} & 0\ 0 & c_{11}+c_{22}\end{pmatrix} \ \forall (c_{11}, c_{22}) \in \mathbb{F}^2. \tag{28}$$

The indeterminates c_{11} and c_{22} keep track of the domain of the flattening. The 4×4 right-hand side matrix in above expression should not be confused with the Kronecker product $\mathrm{I}_{2\times 2} \otimes \mathrm{I}_{2\times 2}$ involved in the left-hand side. In fact, the result $\mathrm{I}_{4\times 4}$ of this Kronecker product is of classical matrix rank 4 while the rank in $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star$ of the right-hand side matrix (28) is 1 by construction. Hence, even if we present in this section the elements of $\mathbb{F}^{\cdot\times\cdot}$ as matrices, we use a vectorization (e.g. $\mathbb{F}^{m\times n} \simeq \mathbb{F}^{mn}$) of these matrices in order to perform correctly our computations. Taking this standpoint into account we obtain the following description of the whole Strassen third flattening image as:

$$Im\, S|_3^1 = \begin{pmatrix} c_{11} & c_{12} & 0 & 0 \\ 0 & 0 & c_{11} & c_{12} \\ c_{21} & c_{22} & 0 & 0 \\ 0 & 0 & c_{21} & c_{22} \end{pmatrix} \tag{29}$$

that could be guessed almost without computation. In fact, this right-hand side matrix is just the matrix of the bilinear form defining the trilinear encoding of the matrix product:

$$\mathrm{Trace}(A \cdot B \cdot C^\intercal) = \begin{pmatrix} a_{11} & a_{12} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} c_{11} & c_{12} & 0 & 0 \\ 0 & 0 & c_{11} & c_{12} \\ c_{21} & c_{22} & 0 & 0 \\ 0 & 0 & c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{pmatrix}. \tag{30}$$

Hence, the flattening $\mathrm{Im}\, S|_3^1$ is a canonical description of the matrix product independent from the algorithm/tensor used to encode it; in particular, it is an invariant under the action of the isotropies introduced in Definition 28. We are going to use these objects and their properties in the following section.

5.3 Presentation of de Groote's method

We are interested in a situation where, given a bilinear map, a classical representation by a tensor $\mathcal{C}$ is known and we wish to disprove the existence of a tensor representation of a given rank. Inspired by Steinitz exchange theorem, de Groote introduced in [10, § 1.5] the following definition to handle this issue.

Definition 31. *Given a tensor $\mathcal{T}$ encoding a bilinear map whose codomain is $\mathfrak{M}$ and given q rank-one elements $\mathcal{P}^i$, linearly independent in $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star$, let us denote by $\mathcal{L}(Im\, T|_3^1, \mathcal{P}^1, \ldots, \mathcal{P}^q)$ the linear subspace of $\mathfrak{M}$ defined by:*

$$\mathrm{LinearSpan}\left\{ \begin{array}{c} M \in \mathfrak{M} \mid \exists (u_1, \ldots, u_q) \in \mathbb{F}^q, \\ \mathrm{Rank}_{(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star} \left(\mathcal{T}|_3^1(M) + u_j \mathcal{P}^j\right) = 1 \end{array} \right\}. \tag{31}$$

We introduced the notation $\mathfrak{M}$ for the codomain of the considered bilinear map in order to highlight the fact that it is isomorphic—via the Frobenius isomorphism—to the domain of the flattening and to show how it is used in the following.

The following proposition allows to construct an effective test that checks if there exists a tensor of rank $\dim \mathfrak{M} + q$ that defines the considered bilinear map.

Proposition 32. *If there exists a tensor $\mathcal{T}$ of rank $\dim \mathfrak{M} + q$ encoding a bilinear map with codomain $\mathfrak{M}$ then there are q rank-one elements $\mathcal{P}^i$ linearly independent in $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star$ such that $\mathcal{L}(Im\,\mathcal{T}|_3^1, \mathcal{P}^1, \ldots, \mathcal{P}^q)$ is $\mathfrak{M}$.*

Proof. Let us assume that there exists a tensor $\mathcal{C}$ encoding the considered bilinear map, that its tensor rank ρ is greater than $\dim \mathfrak{M} + q$ and that it is defined by the sum $\mathcal{Q}^i \otimes R_i$. Remark that the set $(R_i)_{1\leq i\leq \rho}$ is a generating set of the space $\mathfrak{M}$ by hypothesis.

Suppose now that there exists a tensor $\mathcal{T}$ of rank $r = \dim \mathfrak{M} + q$ encoding with fewer summands the same considered bilinear map:

$$\mathcal{T} = \mathcal{P}^k \otimes S_k, \quad S_k \in \mathfrak{M} \subset \mathbb{F}^{m\times p}, \quad \mathcal{P}^k \in (\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star. \tag{32}$$

The elements $\mathcal{P}^k$ are linearly independent, otherwise $\mathcal{T}$ could be expressed with even fewer terms. Furthermore, there is a subset of $(S_i)_{1\leq i\leq \dim \mathfrak{M}}$ that is a base of $\mathfrak{M}$ (otherwise, $\mathcal{T}$ could not encode the same bilinear map as $\mathcal{C}$).

Suppose that we reorder this set so that the base is $(S_{i+q})_{1\leq i\leq \dim \mathfrak{M}}$. By invariance of the flattening map, the following relations hold:

$$\forall\ M \in \mathfrak{M}, \quad \mathcal{C}|_3^1(M) = \langle M, R_k\rangle \mathcal{Q}^k = \mathcal{T}|_3^1(M) = \langle M, S_k\rangle \mathcal{P}^k. \tag{33}$$

By introducing a base $(B^i)_{1\leq i\leq \dim \mathfrak{M}}$ of $\mathfrak{M}$, we could summarize this situation under a matricial standpoint as follows:

$$\begin{pmatrix} \mathcal{P}^1 \\ \vdots \\ \mathcal{P}^q \\ \langle B^1, R_k\rangle \mathcal{Q}^k \\ \vdots \\ \langle B^{\dim \mathfrak{M}}, R_k\rangle \mathcal{Q}^k \end{pmatrix} = \begin{pmatrix} \mathrm{I}_{q\times q} & 0 \\ C & D \end{pmatrix} \begin{pmatrix} \mathcal{P}^1 \\ \vdots \\ \mathcal{P}^q \\ \mathcal{P}^{q+1} \\ \vdots \\ \mathcal{P}^{q+\dim \mathfrak{M}} \end{pmatrix} \tag{34}$$

where the matrices C and D are such that:

$$\forall\ i \in \{1, \ldots, \dim \mathfrak{M}\}, \begin{array}{ll} C_{ij} = \langle B^i, S_j\rangle, & \forall\ j \in \{1, \ldots, q\}, \\ D_{ij} = \langle B^i, S_{q+j}\rangle, & \forall\ j \in \{1, \ldots, \dim \mathfrak{M}\}. \end{array} \tag{35}$$

As, by hypothesis, $(S_{i+q})_{1\leq i\leq \dim \mathfrak{M}}$ is a basis of $\mathfrak{M}$, the matrix D is invertible and we could rewrite Equation (34) as follows:

$$\begin{array}{l} U = -D^{-1}\cdot C, \\ V = D^{-1}, \end{array} \begin{pmatrix} \mathrm{I}_{q\times q} & 0 \\ U & V \end{pmatrix} \begin{pmatrix} \mathcal{P}^1 \\ \vdots \\ \mathcal{P}^q \\ \langle B^1, R_k\rangle \mathcal{Q}^k \\ \vdots \\ \langle B^{\dim \mathfrak{M}}, R_k\rangle \mathcal{Q}^k \end{pmatrix} = \begin{pmatrix} \mathcal{P}^1 \\ \vdots \\ \mathcal{P}^q \\ \mathcal{P}^{q+1} \\ \vdots \\ \mathcal{P}^{q+\dim \mathfrak{M}} \end{pmatrix}. \tag{36}$$

The $\dim \mathfrak{M}$ lines of the (U, V) matrices in Equation (36) give us $\dim \mathfrak{M}$ vectors $(u_1^j, \ldots, u_q^j, v_1^j, \ldots, v_{\dim \mathfrak{M}}^j)$ such that for all j in $\{1, \ldots, \dim \mathfrak{M}\}$

$$\begin{aligned}\mathrm{Rank}_{(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star} \left(u_i^j \mathcal{P}^i + v_i^j \langle B^i, R_k \rangle \mathcal{Q}^k \right) = \mathrm{Rank}_{(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star} \mathcal{P}^{q+j} \\ = 1. \quad (37)\end{aligned}$$

To conclude, we remark that these last relations show that all the matrices $v_i^j B^i$ are in the subspace $\mathcal{L}(\mathrm{Im}\, \mathcal{T}|_3^1, \mathcal{P}^1, \ldots, \mathcal{P}^q)$. As they are $\dim \mathfrak{M}$ independent linear combinations of basis elements of $\mathfrak{M}$, these matrices form another of its bases and thus the subspace $\mathcal{L}(\mathrm{Im}\, \mathcal{T}|_3^1, \mathcal{P}^1, \ldots, \mathcal{P}^q)$ is equal to $\mathfrak{M}$. □

5.4 Adaptation to the Hermitian case

In order to use Proposition 32 to prove Theorem 23, we have to show that for any element $\mathcal{P} = P \otimes Q$ in $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star$ the subspace $\mathcal{L}(\mathrm{Im}\, \mathcal{T}|_3^1, \mathcal{P}, \mathcal{P}^H)$ is not equal to $\mathfrak{M}$ (with $\mathcal{P}^H = Q^H \otimes P^H$). This vector-space $\mathfrak{M}$ is a 3 dimensional vector-space spanned by all the outputs of our bilinear map. Let us start to make this strategy more precise by the following remark.

Remark 33. *A classical block version of bilinear algorithm (e.g. [6, § 6.3.1]) computing the product of a matrix by its adjoint is:*

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} {A_{11}}^H & {A_{21}}^H \\ {A_{12}}^H & {A_{22}}^H \end{pmatrix} = \begin{pmatrix} A_{11}{A_{11}}^H + A_{12}{A_{12}}^H & \times \\ A_{21}{A_{11}}^H + A_{22}{A_{12}}^H & A_{21}{A_{21}}^H + A_{22}{A_{22}}^H \end{pmatrix}. \quad (38)$$

As the result of this algorithm is self-adjoint, by Lemma 9 there is no need to compute the top-right coefficient and thus, we conclude that the dimension of $\mathfrak{M}$ is at most 3. Hence, there exists a bilinear map encoded by a tensor $\mathcal{H} = \Sigma_1^i \otimes \Sigma_2^i \otimes S_i^3$ of rank 6 that computes the product of a matrix by its hermitian transpose and the image of its third flattening $\mathcal{H}|_3^1$ is

$$Im\, \mathcal{H}|_3^1(\mathbb{F}^{m\times p}) = \begin{pmatrix} c_1 & 0 & 0 & 0 \\ 0 & 0 & c_1 & 0 \\ c_2 & c_3 & 0 & 0 \\ 0 & 0 & c_2 & c_3 \end{pmatrix}. \quad (39)$$

We need a last standard definition in order to classify all possible tensors $\mathcal{P}$ considered in the sequel.

Definition 34. *Given a tensor $\mathcal{P}$ decomposable as sum of rank-one tensors:*

$$\mathcal{P} = \sum_{i=1}^{q} \otimes_{j=1}^{s} P_{ij} \text{ where } P_{ij} \text{ are matrices.} \quad (40)$$

The list $[(\mathrm{Rank}\, P_{ij})_{j=1\ldots s}]_{i=1\ldots q}$ is called the type *of tensor $\mathcal{P}$.*

Remark 35. *In our situation q is one, s is two and the P_{ij} are 2×2 matrices; hence, the tensor $\mathcal{P}$ could only have type $[(1,1)]$, $[(1,2)]$, $[(2,1)]$ or $[(2,2)]$.*

We also use the isotropies presented in Definition 28 in order to simplify as much as possible the tensor $\mathcal{P}$ as illustrated in the proof of the following statement. Furthermore, let us first introduce several notations:

- we denote by $G_{x=0}$ the matrix G in which the indeterminate x is replaced by 0;
- the determinant of the matrix $\left(\begin{smallmatrix} G_{i,j} & G_{i,l} \\ G_{k,j} & G_{k,l} \end{smallmatrix}\right)$ is denoted by $\mathrm{Det}_{[i,j|k,l]}(G)$;
- as the rank of a matrix is invariant under elementary row and columns operations, we also use the notation $G|[i,j|\ell]$ for the matrix resulting from the addition to the ith line of G of its jth line multiplied by ℓ.

Lemma 36. *There is no tensor $\mathcal{P}$ in $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star$ of type $(1,i)$ with i equals to 1 or 2 such that the subspace $\mathcal{L}(\mathcal{H}|_3^1, \mathcal{P}, \mathcal{P}^H)$ is equal to $\mathfrak{M}$.*

Proof. Let us consider a tensor $\mathcal{P} = A \otimes B$ of type $(1,i)$ with $i = 1, 2$. As the first component A is of rank one, there exists two vectors such that:

$$A = \begin{pmatrix} a_1 & a_2 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 b_1 & a_2 b_1 \\ a_1 b_2 & a_2 b_2 \end{pmatrix}. \tag{41}$$

If the coefficient a_2 is zero, we choose a matrix β as the identity matrix. If the coefficient a_1 is zero, we could consider a permutation matrix $\beta = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$; otherwise, if $s = a_1\overline{a_1} + a_2\overline{a_2} \neq 0$, consider:

$$\beta = \begin{pmatrix} \overline{a_1} & a_2 \\ \overline{a_2} & -a_1 \end{pmatrix}. \tag{42}$$

Then we have both $\beta \cdot \beta^H = s \cdot \mathrm{I}_{2\times 2}$ and $A \cdot \beta = \begin{pmatrix} sb_1 & 0 \\ sb_2 & 0 \end{pmatrix}$.

Hence, in any of these cases, there always exists a matrix β, which inverse is a multiple of its hermitian transpose, such that the isotropy g defined by $(\mathrm{I}_{m\times m} \times \beta \times \mathrm{I}_{n\times n})$ satisfies the following properties:

$$\mathsf{g} \diamond \mathcal{P} = \begin{pmatrix} u_1 & 0 \\ u_2 & 0 \end{pmatrix} \otimes \begin{pmatrix} v_1 & v_2 \\ v_3 & v_4 \end{pmatrix}, \quad \mathsf{g} \diamond \mathcal{P}^H = \frac{1}{s}(\mathsf{g} \diamond \mathcal{P})^H \tag{43}$$

With the above notations, conventions and isotropy's action, given any M in $\mathfrak{M}$, the 4×4 matrix $\mathcal{H}|_3^1(M) + y_1\mathcal{P} + y_2\mathcal{P}^H$ is:

$$\begin{pmatrix} y_1u_1v_1 + y_2{v_1}^H{u_1}^H + c_1 & y_1u_1v_2 + y_2{v_1}^H{u_2}^H & y_1u_1v_3 & y_1u_1v_4 \\ y_2{v_3}^H{u_1}^H & y_2{v_3}^H{u_2}^H & c_1 & 0 \\ y_1u_2v_1 + y_2{v_2}^H{u_1}^H + c_2 & y_1u_2v_2 + y_2{v_2}^H{u_2}^H + c_3 & y_1u_2v_3 & y_1u_2v_4 \\ y_2{v_4}^H{u_1}^H & y_2{v_4}^H{u_2}^H & c_2 & c_3 \end{pmatrix}. \tag{44}$$

This matrix is supposed to be of rank one. Thus, all its 2×2 minors are equal to 0. Then, either c_1 or c_3 is equal to 0 and for any such $\mathcal{P}$ the subspace $\mathcal{L}(\mathcal{H}|_3^1, \mathcal{P}, \mathcal{P}^H)$ is thus not $\mathfrak{M}$.

There remains the case $s = 0$. Then let

$$\beta = \begin{pmatrix} {a_1}^{-1} & -a_2 \\ 0 & a_1 \end{pmatrix}. \tag{45}$$

The inverse of β is no longer related to β^H anymore but β still transforms A into a single non-zero column matrix: $A \cdot \beta = \begin{pmatrix} b_1 & 0 \\ b_2 & 0 \end{pmatrix}$. Thus, for this β, the action of the isotropy $\mathfrak{g} = (\mathrm{I}_{m\times m} \times \beta \times \mathrm{I}_{n\times n})$ is:

$$\mathfrak{g} \diamond \mathcal{P} = \begin{pmatrix} b_1 & 0 \\ b_2 & 0 \end{pmatrix} \otimes \begin{pmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{pmatrix} \quad \text{and} \quad \mathfrak{g} \diamond \mathcal{P}^H = U \otimes V. \tag{46}$$

With the above notations, conventions and isotropy's action, given any M in $\mathfrak{M}$, the 4×4 matrix $G = \mathcal{H}|^1_3(M) + y_1\mathcal{P} + y_2\mathcal{P}^H$ is thus:

$$\begin{pmatrix} y_1b_1z_{11}+y_2u_{11}v_{11}+c_1 & y_1b_1z_{12}+y_2u_{11}v_{12} & y_1b_1z_{21}+y_2u_{11}v_{21} & y_1b_1z_{22}+y_2u_{11}v_{22} \\ y_2u_{12}v_{11} & y_2u_{12}v_{12} & y_2u_{12}v_{21}+c_1 & y_2u_{12}v_{22} \\ y_1b_2z_{11}+y_2u_{21}v_{11}+c_2 & y_1b_2z_{12}+y_2u_{21}v_{12}+c_3 & y_1b_2z_{21}+y_2u_{21}v_{21} & y_1b_2z_{22}+y_2u_{21}v_{22} \\ y_2u_{22}v_{11} & y_2u_{22}v_{12} & y_2u_{22}v_{21}+c_2 & y_2u_{22}v_{22}+c_3 \end{pmatrix}. \tag{47}$$

This matrix is supposed to be of rank one in $\mathcal{L}(\mathcal{H}|^1_3, \mathcal{P}, \mathcal{P}^H)$. Thus, all its 2×2 minors are equal to 0.

On the one hand, if y_2 is zero, then the constraints of Equation (47) show that $\mathrm{Det}_{[2,3|4,4]}(G_{y_2=0}) = c_1c_3$ is equal to 0. On the other hand, if y_2 is different from 0 then the minor $\mathrm{Det}_{[2,2|4,4]}(G)$ is equal to $c_3y_2u_{12}v_{12}$ and supposed to be equal to zero by hypothesis. We also have that the minor $\mathrm{Det}_{[2,1|4,4]}(G)$ is equal to $c_3y_2u_{12}v_{11}$ and is also supposed to be equal to zero by hypothesis. We are going to explore all the consequences induced by this constraint.

$$\begin{array}{llll} u_{12} \neq 0, v_{12} \neq 0 & & \to c_3 = 0, \\ u_{12} = 0, v_{12} = 0 & & \to \mathrm{Det}_{[2,1|4,3]}(G_{u_{12}=0,v_{12}=0}) = y_2u_{22}v_{11}c_1 = 0, \\ u_{12} = 0, v_{12} = 0, & u_{22} = 0 & \to \mathrm{Det}_{[2,3|4,4]}(G_{u_{12}=0,v_{12}=0,u_{22}=0}) = c_1c_3 = 0, \\ u_{12} = 0, v_{12} = 0, & v_{11} = 0 & \text{see thereafter} \\ u_{12} = 0, v_{12} \neq 0 & & \to \mathrm{Det}_{[2,2|4,3]}(G_{u_{12}=0}) = -y_2v_{12}u_{22}c_1 = 0, \\ u_{12} = 0, v_{12} \neq 0, & u_{22} \neq 0 & \to c_1 = 0, \\ u_{12} = 0, v_{12} \neq 0, & u_{22} = 0 & \to \mathrm{Det}_{[2,3|4,4]}(G_{u_{12}=0,u_{22}=0}) = c_1c_3 = 0. \end{array} \tag{48}$$

Now, if u_{12} is different from 0, then from the first two minors, either The relations $v_{12} = v_{11} = 0$ hold or $c_3 = 0$.

Further, not both b_1 and b_2 can be zero, otherwise the tensor is of rank 0. W.l.o.g., suppose that $b_2 \neq 0$ and let $G' = G_{v_{12}=0,v_{11}=0}|[1,3|-b_1/b_2]$. Then $\mathrm{Det}_{[1,2|2,4]}(G') = (-b_1/b_2)y_2c_3u_{12}v_{22}$ and either $b_1 = 0$ or $v_{22} = 0$ or $c_3 = 0$. This gives the following distinctions (recall that now $u_{12} \neq 0$ and $y_2 \neq 0$):

$$\begin{array}{llll} b_1 = 0, & & \to \mathrm{Det}_{[1,1|2,4]}({G'}_{b_1=0}) = y_2u_{12}v_{22}c_1 = 0, \\ b_1 = 0, & v_{22} \neq 0 & \to c_1 = 0, \\ b_1 = 0, & v_{22} = 0 & \to \mathrm{Det}_{[1,1|4,4]}(G') = c_1c_3 = 0, \\ b_1 \neq 0, & v_{22} = 0 & \to \mathrm{Det}_{[1,2|4,4]}(G') = (-b_1/b_2){c_3}^2 = 0, \end{array} \tag{49}$$

There remains the case $u_{12} = 0$, $v_{12} = 0$ and $v_{11} = 0$ in Equation (48). Here also, w.l.o.g., suppose that $b_2 \neq 0$ and let $G^* = G_{u_{12}=0,v_{12}=0,v_{11}=0}|[1,3|-b_1/b_2]$.

$$\begin{array}{ll} b_1 = 0 & \rightarrow \mathrm{Det}_{[1,1|2,3]}({G^*}_{b_1=0}) = {c_1}^2 = 0, \\ b_1 \neq 0 & \rightarrow \mathrm{Det}_{[1,2|2,3]}(G^*) = (-b_1/b_2)c_1c_3 = 0. \end{array} \tag{50}$$

Thus, in any cases, for any such $\mathcal{P}$, the set $\mathcal{L}(\mathcal{H}|_3^1, \mathcal{P}, \mathcal{P}^H)$ is not $\mathfrak{M}$.

Note that a computational way to see this, is to perform a Gröbner basis computation, directly from Equation (47): for instance over $\mathbb{C}$ this gives that the relation ${c_1}^2{c_3}^2 = 0$ must hold and that the set is not the full codomain. □

According to Remark 35, the above computations deal with half of the cases to consider. We remark that, mutatis mutandis, similar computations exclude also the existence of an algorithm where $\mathcal{P}$ is of type $(2, 1)$. We could consider now the last case.

Lemma 37. *There is no tensor $\mathcal{P}$ in $(\mathbb{F}^{m\times n})^\star \otimes (\mathbb{F}^{n\times p})^\star$ of type $(2, 2)$ such that the subspace $\mathcal{L}(\mathcal{H}|_3^1, \mathcal{P}, \mathcal{P}^H)$ is equal to $\mathfrak{M}$.*

Proof. First, let us consider a tensor $\mathcal{P}$ of type $(2, 2)$. Thus, there exists β such that the action of the isotropy $\mathsf{g} = (\mathrm{I}_{m\times m} \times \beta \times \mathrm{I}_{n\times n})$ is:

$$\mathsf{g} \diamond \mathcal{P} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{pmatrix} \quad \text{and} \quad \mathsf{g} \diamond \mathcal{P}^H = U \otimes V. \tag{51}$$

With the above notations, conventions and isotropy's action, given any M in $\mathfrak{M}$, the 4×4 matrix $G = \mathcal{H}|_3^1(M) + y_1\mathcal{P} + y_2\mathcal{P}^H$ is:

$$\begin{pmatrix} y_2u_{11}v_{11}+c_1+y_1z_{11} & y_2u_{11}v_{12}+y_1z_{12} & y_2u_{11}v_{21}+y_1z_{21} & y_2u_{11}v_{22}+y_1z_{22} \\ y_2u_{12}v_{11} & y_2u_{12}v_{12} & y_2u_{12}v_{21}+c_1 & y_2u_{12}v_{22} \\ y_2u_{21}v_{11}+c_2 & y_2u_{21}v_{12}+c_3 & y_2u_{21}v_{21} & y_2u_{21}v_{22} \\ y_2u_{22}v_{11}+y_1z_{11} & y_2u_{22}v_{12}+y_1z_{12} & y_2u_{22}v_{21}+c_2+y_1z_{21} & y_2u_{22}v_{22}+c_3+y_1z_{22} \end{pmatrix}. \tag{52}$$

This matrix is supposed to be of rank one in $\mathcal{L}(\mathcal{H}|_3^1, \mathcal{P}, \mathcal{P}^H)$. Thus, all its 2×2 minors are equal to 0.

A Gröbner basis computation over $\mathbb{C}$ shows in that case that the relations ${c_1}^2c_3 = c_1c_2c_3 = c_1{c_3}^2 = 0$ hold and this is sufficient to conclude. Nevertheless, we present a proof that does not require such computations and is valid for any field.

On the one hand, if $y_2 = 0$, then the constraints of Equation (52) show for instance that both $\mathrm{Det}_{[2,1|3,3]}(G_{y_2=0}) = c_1c_2$ and $\mathrm{Det}_{[2,2|3,3]}(G_{y_2=0}) = c_1c_3$ are equal to 0.

On the other hand, if $y_2 \neq 0$ then consider the minor $\mathrm{Det}_{[2,3|3,4]}(G)$, which is equal to $c_1y_2u_{21}v_{22}$ and supposed to be equal to zero by hypothesis. We are going to explore all the consequences induced by this constraint. First, if $y_1 = 0$,

then we have:

$$\begin{array}{llll}
u_{21} \neq 0, v_{22} \neq 0 & & \rightarrow c_1 = 0, \\
u_{21} = 0, v_{22} = 0 & & \rightarrow \mathrm{Det}_{[3,2|4,4]}(G_{y_1=0,u_{21}=0,v_{22}=0}) = {c_3}^2 = 0, \\
u_{21} = 0, v_{22} \neq 0 & & \rightarrow \mathrm{Det}_{[2,2|3,4]}(G_{y_1=0,u_{21}=0}) = -y_2 u_{12} v_{22} c_3 = 0, \\
u_{21} = 0, v_{22} \neq 0, & u_{12} \neq 0 & \rightarrow c_3 = 0, \\
u_{21} = 0, v_{22} \neq 0, & u_{12} = 0 & \rightarrow \mathrm{Det}_{[2,2|3,3]}(G_{y_1=0,u_{21}=0,u_{12}=0}) = -c_1 c_3 = 0, \\
u_{21} \neq 0, v_{22} = 0 & & \rightarrow \mathrm{Det}_{[3,3|4,4]}(G_{y_1=0,v_{22}=0}) = c_3 u_{21} v_{21} y_2 = 0, \\
u_{21} \neq 0, v_{22} = 0, & v_{21} \neq 0 & \rightarrow c_3 = 0, \\
u_{21} \neq 0, v_{22} = 0, & v_{21} = 0 & \rightarrow \mathrm{Det}_{[2,3|4,4]}(G_{y_1=0,v_{22}=0,v_{21}=0}) = c_1 c_3 = 0,
\end{array} \tag{53}$$

If y_1 and y_2 are both non-zero, then, the minor $\mathrm{Det}_{[1,1|2,2]}(G|[1,4|-1])$ is equal to $y_2 c_1 u_{12} v_{12}$ and supposed to be equal to zero by hypothesis. We are going to explore all the consequences induced by this constraint. Let $G' = G|[1,4|-1]$:

$$\begin{array}{llll}
u_{12} \neq 0, v_{12} \neq 0 & & \rightarrow c_1 = 0, \\
u_{12} = 0, v_{12} = 0 & & \rightarrow \mathrm{Det}_{[2,2|3,3]}({G'}_{u_{12}=0,v_{12}=0}) = -c_1 c_3 = 0, \\
u_{12} = 0, v_{12} \neq 0 & & \rightarrow \mathrm{Det}_{[2,3|3,4]}({G'}_{u_{12}=0}) = y_2 c_1 u_{21} v_{22} = 0, \\
u_{12} = 0, v_{12} \neq 0, & u_{21} \neq 0, \quad v_{22} \neq 0 & \rightarrow c_1 = 0, \\
u_{12} = 0, v_{12} \neq 0, & u_{21} = 0 & \rightarrow \mathrm{Det}_{[2,2|3,3]}({G'}_{u_{12}=0,u_{21}=0}) = -c_1 c_3 = 0, \\
u_{12} = 0, v_{12} \neq 0, & v_{22} = 0 & \rightarrow \mathrm{Det}_{[1,3|2,4]}({G'}_{u_{12}=0,u_{21}=0}) = c_1 c_3 = 0, \\
u_{12} \neq 0, v_{12} = 0 & & \rightarrow \mathrm{Det}_{[2,2|3,4]}({G'}_{v_{12}=0}) = -y_2 u_{12} v_{22} c_3 = 0, \\
u_{12} \neq 0, v_{12} = 0, & v_{22} \neq 0 & \rightarrow c_3 = 0, \\
u_{12} \neq 0, v_{12} = 0, & v_{22} = 0 & \rightarrow \mathrm{Det}_{[1,2|3,3]}({G'}_{v_{12}=0,v_{22}=0}) = {c_3}^2 = 0.
\end{array} \tag{54}$$

Thus, in any cases, for any such $\mathcal{P}$, the set $\mathcal{L}(\mathcal{H}\,|_3^1, \mathcal{P}, \mathcal{P}^H)$ is not $\mathfrak{M}$. □

The Proposition 32, together with the computations done in Lemma 37 and in Lemma 36 are sufficient to conclude the proof of Theorem 23.

6 The case of field extensions via matrix polynomial arithmetic

The cost comparison in Table 1 is for matrices over an arbitrary ring with skew unitary matrices. When the ring is an extension, the input of the problem is a polynomial matrix over the base ring. Following the traditional equivalence between polynomial matrices and matrix polynomials, leads to alternative ways to multiply the matrix by its transpose, considering the product of two polynomials with matrix coefficients. More specifically, we will focus on degree two extensions, and compare the costs in terms of number of operations over the base ring.

6.1 The 2M method

Over the field $\mathbb{C}$ of complex numbers, the $3M$ method (Karatsuba) for general matrix multiplication reduces the number of multiplications of real matrices from 4 to 3 [13]: if $\mathrm{MM}_\omega^{\mathbb{R}}(n)$ is the cost of multiplying $n \times n$ matrices over $\mathbb{R}$, then the $3M$ method costs $3\mathrm{MM}_\omega^{\mathbb{R}}(n) + o(n^\omega)$ operations over $\mathbb{R}$. Adapting this

approach for product of matrix by its adjoint yields a $2M$ method using only 2 real products:

Algorithm 38 2M multiplication in an extension

Input: A commutative ring $\mathbb{K}$ and one of its extensions $\mathbb{E}$;
Input: $A \in \mathbb{K}^{m\times n}$ and $B \in \mathbb{K}^{m\times n}$;
Input: ϕ an involutive matrix antihomomorphism of $\mathbb{E}$.
Input: $i \in \mathbb{E}$, commuting with $\mathbb{K}$, and such that $\epsilon = i\phi(i) \in \mathbb{K}$;
Output: $(A + iB) \cdot \phi(A + iB) \in \mathbb{E}^{m\times m}$.
Let $H = A \cdot \phi(B) \in \mathbb{K}^{m\times m}$;
Let $G = (A + B) \cdot \phi(A + \epsilon B) \in \mathbb{K}^{m\times m}$;
return $(G - \epsilon H - \phi(H)) + H\phi(i) + i\phi(H)$.

Lemma 39. *Algorithm 38 is correct. It costs $2MM_\omega^{\mathbb{K}}(n) + o(n^\omega)$ operations over the base ring $\mathbb{K}$.*

Proof. Let $M = (A + iB) \cdot \phi(A + iB)$. By Lemma 4, we have that $\phi(iB) = \phi(B)\phi(i) = \phi(i)\phi(B)$. Thus, $M = A \cdot \phi(A) + A \cdot \phi(B)\phi(i) + iB \cdot \phi(A) + iB \cdot \phi(B)\phi(i)$, by Equations (2b) and (2c). As i commutes with $\mathbb{K}$, we also have that $M = A \cdot \phi(A) + B \cdot \phi(B)\epsilon + A \cdot \phi(B)\phi(i) + iB \cdot \phi(A)$. By Equations (2a) and (2c), we have that $\phi(H) = \phi(\phi(B)) \cdot \phi(A) = B \cdot \phi(A)$. Finally, $G = A \cdot \phi(A) + A \cdot \phi(B)\epsilon + B \cdot \phi(A) + B \cdot \phi(B)\epsilon$ as $\phi(\epsilon) = \phi(\phi(i))\phi(i) = i\phi(i) = \epsilon$. Therefore $G - \epsilon H - \phi(H) = G - H\epsilon - \phi(H) = A \cdot \phi(A) + B \cdot \phi(B)\epsilon$ and $M = G + H\phi(i) + i\phi(H)$. □

Example 40. *For instance, if $\mathbb{K} = \mathbb{R}$, $\mathbb{E} = \mathbb{C}$, then $i = \sqrt{-1}$ satisfies the conditions of Algorithm 38 for both cases when ϕ is the transposition or the conjugate transposition. Therefore, we obtain the multiplications of a matrix by its adjoint, whether it be the transpose or the conjugate transpose, in $2MM_\omega^{\mathbb{R}} + o(n^\omega)$ operations in $\mathbb{R}$. The classical divide and conquer algorithm, see e.g. [6, § 6.3.1], works directly over $\mathbb{C}$ and uses the equivalent of $\frac{2}{2^\omega-4}$ complex floating point $n \times n$ matrix products. Using the $3M$ method for the complex products, this algorithm uses overall $\frac{6}{2^\omega-4}MM_\omega^{\mathbb{R}}(n) + o(n^\omega)$ operations in $\mathbb{R}$. Finally, Algorithm 12 costs $\frac{2}{2^\omega-3}$ complex multiplications for a leading term bounded by $\frac{6}{2^\omega-3}MM_\omega^{\mathbb{R}}(n)$, improving over $2MM_\omega^{\mathbb{R}}$ for $\omega > \log_2(6) \approx 2.585$, but this does not apply to the conjugate transpose case. This is summarized in Table 2, also replacing ω by 3 or $\log_2(7)$ to illustrate the situation for the main feasible exponents.*

6.2 The quaternion algebra

Given a field $\mathbb{K}$ of characteristic not 2, the $\mathbb{K}$-algebra of *quaternions* $\mathbb{H}(\mathbb{K})$ is the $\mathbb{K}$-vector space of all formal linear combinations:

$$x_1 + x_2\mathbf{i} + x_3\mathbf{j} + x_4\mathbf{k}, \quad (x_1, \ldots, x_4) \in \mathbb{K}^4, \tag{55}$$

Problem	Alg.	$\mathrm{MM}_3(n)$	$\mathrm{MM}_{\log_2 7}(n)$	$\mathrm{MM}_\omega(n)$
$A \cdot B$	naive	$8n^3$	$4\,\mathrm{MM}^{\mathbb{R}}_{\log_2(7)}(n)$	$4\,\mathrm{MM}^{\mathbb{R}}_{\omega}(n)$
	3M	$6n^3$	$3\,\mathrm{MM}^{\mathbb{R}}_{\log_2(7)}(n)$	$3\,\mathrm{MM}^{\mathbb{R}}_{\omega}(n)$
$A \cdot A^H$	Alg. 38	$4n^3$	$\mathbf{2\,MM^{\mathbb{R}}_{\log_2(7)}(n)}$	$\mathbf{2\,MM^{\mathbb{R}}_{\omega}(n)}$
	[6]	$\mathbf{3n^3}$	$\mathbf{2\,MM^{\mathbb{R}}_{\log_2(7)}(n)}$	$\frac{6}{2^\omega-4}\,\mathrm{MM}^{\mathbb{R}}_{\omega}(n)$
$A \cdot A^\intercal$	Alg. 38	$4n^3$	$2\,\mathrm{MM}^{\mathbb{R}}_{\log_2(7)}(n)$	$\mathbf{2\,MM^{\mathbb{R}}_{\omega}(n)}$
	[6]	$3n^3$	$2\,\mathrm{MM}^{\mathbb{R}}_{\log_2(7)}(n)$	$\frac{6}{2^\omega-4}\,\mathrm{MM}^{\mathbb{R}}_{\omega}(n)$
	Alg. 12	$\mathbf{2.4n^3}$	$\mathbf{\frac{3}{2}\,MM^{\mathbb{R}}_{\log_2(7)}(n)}$	$\frac{6}{2^\omega-3}\,\mathrm{MM}^{\mathbb{R}}_{\omega}(n)$

Table 2: Multiplication of a matrix by its adjoint or transpose over $\mathbb{C}$: leading term of the cost in number of arithmetic operations over $\mathbb{R}$. Note that $2 < \frac{6}{2^\omega-4}$ only when $\omega < \log_2(7) \approx 2.81$ and $2 < \frac{6}{2^\omega-3}$ only when $\omega < \log_2(6) \approx 2.585$.

the non-commutative multiplication being defined by the bilinear extensions of the relations:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{56}$$

The quaternions can also be seen as a degree 2 extension of a degree 2 extension, but a non-commutative one. Therefore the $3M$ or §$2M$techniques of Section 6.1 only apply directly for the first degree 2 extension, while the second extension would require 4 multiplications. This gives $4 \times 3 = 12$ (resp. $4 \times 2 = 8$) multiplications in the base field for a general matrix multiplication (resp. a multiplication of a matrix by its transpose or conjugate transpose). For the former case, there exist actually algorithms using only 8 multiplications instead of 12. For the latter case, we present algorithms using only 7 multiplications for the transpose case and only 6 multiplications for the conjugate transpose case instead of 8.

6.2.1 Quaternions' multiplication

The multiplication of quaternions is $(x_1+x_2\mathbf{i}+x_3\mathbf{j}+x_4\mathbf{k})(y_1+y_2\mathbf{i}+y_3\mathbf{j}+y_4\mathbf{k}) = (w_1 + w_2\mathbf{i} + w_3\mathbf{j} + w_4\mathbf{k})$, with:

$$w_1 = x_1y_1 - x_2y_2 - x_3y_3 - x_4y_4 \tag{57}$$
$$w_2 = x_1y_2 + x_2y_1 + x_3y_4 - x_4y_3 \tag{58}$$
$$w_3 = x_1y_3 - x_2y_4 + x_3y_1 + x_4y_2 \tag{59}$$
$$w_4 = x_1y_4 + x_2y_3 - x_3y_2 + x_4y_1 \tag{60}$$

Fiduccia showed in [9] how to compute this product with only 10 field multiplications and 25 additions, cleverly using Gauß' trick for the multiplication

of complex numbers in three multiplications. Regarding the minimal number of base field operation required for the multiplication of quaternions, de Groote shows in [11] that 10 multiplications is minimal to compute both $X \cdot Y$ and $Y \cdot X$. In addition, over the reals and the rationals, the minimal number of multiplications is 8 [14] and [10, Proposition 1.7]. The algorithm of [14], requiring also 28 additions, is recalled in Algorithm 41.

Algorithm 41 Howell-Lafon quaternion multiplication

Input: $\vec{x} = x_1 + x_2\mathbf{i} + x_3\mathbf{j} + x_4\mathbf{k} \in \mathbb{H}(\mathbb{K})$, $\vec{y} = y_1 + y_2\mathbf{i} + y_3\mathbf{j} + y_4\mathbf{k} \in \mathbb{H}(\mathbb{K})$
Output: $\vec{x}\vec{y} \in \mathbb{H}(\mathbb{K})$

$$
\begin{array}{llll}
Q_1 & = (x_1 + x_2)(y_1 + y_2), & Q_2 & = (x_4 - x_3)(y_3 - y_4) \\
Q_3 & = (x_2 - x_1)(y_3 + y_4), & Q_4 & = (x_3 + x_4)(y_2 - y_1) \\
Q_5 & = (x_2 + x_4)(y_2 + y_3), & Q_6 & = (x_2 - x_4)(y_2 - y_3) \\
Q_7 & = (x_1 + x_3)(y_1 - y_4), & Q_8 & = (x_1 - x_3)(y_1 + y_4) \\
T_1 & = Q_5 + Q_6, & T_2 & = Q_7 + Q_8 \\
T_3 & = Q_5 - Q_6, & T_4 & = Q_7 - Q_8 \\
T_5 & = T_2 - T_1, & T_6 & = T_1 + T_2 \\
T_7 & = T_3 + T_4, & T_8 & = T_3 - T_4 \\
w_1 & = Q_2 + T_5/2, & w_2 & = Q_1 - T_6/2 \\
w_3 & = T_7/2 - Q_3, & w_4 & = T_8/2 - Q_4
\end{array}
$$

return $w_1 + w_2\mathbf{i} + w_3\mathbf{j} + w_4\mathbf{k}$.

Proposition 42. *Algorithm 41 extends to the case of quaternions with matrix coefficients. If matrix multiplication over the base field costs $MM_\omega^{\mathbb{K}}(n)$ field operations for $n \times n$ matrices and the field matrix addition $O(n^2)$ field additions, then the dominant cost of Algorithm 41 applied to matrices is bounded by $8MM_\omega^{\mathbb{K}}(n)$.*

Proof. Correctness is by inspection since 2 is invertible in $\mathbb{K}$ of characteristic different from 2. The complexity bound is just the fact that the Algorithm performs 8 multiplications of matrices with coefficients in the base field. □

The lowest number of multiplications required to multiply two quaternions being 8, Proposition 42 is the best possible result while keeping the view of the matrices as two quaternions with base field matrix coefficients, X_1, X_2, X_3, X_4 and Y_1, Y_2, Y_3, Y_4. The alternative is to use a matrix with quaternion coefficients and use classical fast matrix algorithms. Next, we see the different alternatives for the multiplication by an adjoint.

6.2.2 Multiplication of a quaternion matrix by its transpose

We now propose several methods to multiply a quaternion matrix by its transpose:

1. First a "7M" method which considers a quaternion with matrix coefficients, and thus reduces everything to seven general matrix multiplications over the base field.

2. Second, one can consider a matrix with quaternion coefficients and just apply any matrix multiplication algorithm where multiplication of coefficients is that of Algorithm 41.

7M method: a quaternion with matrix coefficients. Many simplifications used in computing the square of a quaternion no longer apply when computing the product of a quaternion matrix by its transpose, due to non-commutativity of the matrix product. For $A, B, C, D \in \mathbb{K}^{m\times n}$,

$$(A + B\mathbf{i} + C\mathbf{j} + D\mathbf{k})(A^\intercal + B^\intercal\mathbf{i} + C^\intercal\mathbf{j} + D^\intercal\mathbf{k}) = S_1 + S_2\mathbf{i} + S_3\mathbf{j} + S_4\mathbf{k} \quad (61)$$

where:

$$S_1 = AA^\intercal - BB^\intercal - CC^\intercal - DD^\intercal \quad (62)$$
$$S_2 = (AB^\intercal + BA^\intercal) + (CD^\intercal - DC^\intercal) \quad (63)$$
$$S_3 = (AC^\intercal + CA^\intercal) + (DB^\intercal - BD^\intercal) \quad (64)$$
$$S_4 = (AD^\intercal + DA^\intercal) + (BC^\intercal - CB^\intercal) \quad (65)$$

Using Munro's trick twice, this can be computed with 7 multiplications over the field $\mathbb{K}$ and 17 additions (6 of which are half-additions), as shown in Algorithm 43.

Open question 1. *Multiply a quaternion with matrix coefficients by its transpose in fewer than* 7 *multiplications.*

Using matrices of quaternions and divide and conquer. In the following, for the sake of simplicity, we will consider only square matrices. Here, we consider instead a matrix with quaternion coefficients and perform a matrix-matrix product: since the quaternions are not commutative, then $MM^\intercal$ is not necessarily symmetric, one has to compute both the top right and bottom left corners of the product. Thus no gain is obvious between computing $MM^\intercal$ and MN that way and Algorithm 12 is a priori useless in this case, as remarked in the (counter)-Examples 6. The idea is thus to use a non symmetric algorithm, applied to M and $M^\intercal$. The baseline cost would then again be $\mathrm{MM}_\omega^{\mathbb{H}(\mathbb{K})}(n) = 8\mathrm{MM}_\omega^{\mathbb{K}}(n)$.

Another approach is to use a divide and conquer strategy at the higher level: cut M into $\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$, and compute:

- $M_{11} \cdot {M_{21}}^\intercal$, $M_{12} \cdot {M_{22}}^\intercal$, $M_{21} \cdot {M_{11}}^\intercal$, $M_{22} \cdot {M_{12}}^\intercal$ by the baseline algorithm;

- and $M_{11} \cdot {M_{11}}^\intercal$, $M_{12} \cdot {M_{12}}^\intercal$, $M_{21} \cdot {M_{21}}^\intercal$, $M_{22} \cdot {M_{22}}^\intercal$ by recursive calls.

Algorithm 43 Fast quaternion matrix multiplications by its transpose

Input: $A, B, C, D \in \mathbb{K}^{m\times n}$
Output: $M \cdot M^\intercal \in \mathbb{H}(\mathbb{K})^{m\times m}$, for $M = A + B\mathbf{i} + C\mathbf{j} + D\mathbf{k}$.

$$
\begin{array}{rlrl}
U_1 &= A + B, & U_2 &= A - B \\
U_3 &= C + D & & \\
\hline
P_1 &= CA^\intercal, & P_2 &= DB^\intercal \\
P_3 &= U_3U_2{}^\intercal, & P_4 &= U_1U_3{}^\intercal, \\
P_5 &= AB^\intercal, & P_6 &= CD^\intercal \\
P_7 &= (U_1 + U_3)(U_2{}^\intercal - U_3{}^\intercal) & & \\
\hline
R_1 &= P_5 + P_6, & R_2 &= P_5 - P_6 \\
\mathrm{Low}\,(R_3) &= \mathrm{Low}\,(P_1 + P_1{}^\intercal) & & \\
\mathrm{Low}\,(R_4) &= \mathrm{Low}\,(P_2 - P_2{}^\intercal) & & \\
\mathrm{Low}\,(S_1) &= \mathrm{Low}\,(P_7 - P_3 + P_4 + R_1 - R_2{}^\intercal) & & \\
S_2 &= R_1 + R_2{}^\intercal & & \\
S_3 &= R_3 + R_4 & & \\
S_4 &= P_3 + P_4 - S_3{}^\intercal & &
\end{array}
$$

return $S_1 + S_2\mathbf{i} + S_3\mathbf{j} + S_4\mathbf{k}$

The cost of this divide and conquer strategy is then:

$$C(n) \leq 4C(\frac{n}{2}) + 4\mathrm{MM}_\omega^{\mathbb{H}(\mathbb{K})}(\frac{n}{2}) + o(n^\omega) \leq 4C(\frac{n}{2}) + 32\mathrm{MM}_\omega^{\mathbb{K}}(\frac{n}{2}) + o(n^\omega). \quad (66)$$

By Lemma 11, we have that $C(n) \leq \frac{32}{2^\omega - 4}\mathrm{MM}_\omega^{\mathbb{K}}(n) + o(n^\omega)$, and this is never better that $8\mathrm{MM}_\omega^{\mathbb{K}}(n)$ (but equal when $\omega = 3$ as expected). So this is thus useless too.

But the same strategy can be used with a Strassen-like algorithm instead. Now such algorithms, for instance those of [19, 21], when applied to M and $M^\intercal$, use two recursive calls and five normal multiplications. This is:

$$S(n) \leq 2S(\frac{n}{2}) + 5\mathrm{MM}_\omega^{\mathbb{H}(\mathbb{K})}(\frac{n}{2}) + o(n^\omega) \leq 2S(\frac{n}{2}) + 40\mathrm{MM}_\omega^{\mathbb{K}}(\frac{n}{2}) + o(n^\omega). \quad (67)$$

By Lemma 11, we obtain that this is

$$S(n) \leq \frac{40}{2^\omega - 2}\mathrm{MM}_\omega^{\mathbb{K}}(n) + o(n^\omega). \quad (68)$$

As expected this is again $8\mathrm{MM}_{\log_2(7)}^{\mathbb{K}}(n)$ if a Strassen-like algorithm is also used for the baseline over the field and $\omega = \log_2(7)$. This is worse if $\omega < \log_2(7)$, but better, and only $(6 + \frac{2}{3})\mathrm{MM}_3^{\mathbb{K}}(n)$, if $\omega = 3$.

Positive characteristic quaternions and transposition. In this case, Algorithm 12 is not usable. It nonetheless has an interesting feature: it has 3

symmetric products instead of 2 for the algorithms of [19, 21]. In the quaternion case, as transposition is not an antihomomorphism, one cannot use the symmetries directly to save computations as in general $(M \cdot N)^\intercal \neq N^\intercal \cdot M^\intercal$. But if one is willing to *recompute* $N^\intercal \cdot M^\intercal$ then the algorithm still works. This yields to Algorithm 44 which requires 7 multiplications instead of 5, 15 additions instead of 9, and 4 multiplications by Y or $Y^\intercal$.

Algorithm 44 Product of a matrix by its non antihomomorphic transpose

Input: $A \in \mathfrak{R}^{m \times n}$ (with even m and n for the sake of simplicity);
Input: $Y \in \mathfrak{R}^{\frac{n}{2} \times \frac{n}{2}}$ such that $Y \cdot Y^\intercal = -\mathrm{I}_{\frac{n}{2}}$;
Output: $A \cdot A^\intercal$.

Split $A = \left(\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right)$ where A_{11} is in $\mathfrak{R}^{\frac{m}{2} \times \frac{n}{2}}$

$$
\begin{array}{ll}
S_1 \leftarrow (A_{21} - A_{11}) \cdot Y & ST_1 \leftarrow Y^\intercal \cdot ({A_{21}}^\intercal - {A_{11}}^\intercal) \\
S_2 \leftarrow A_{22} - A_{21} \cdot Y & ST_2 \leftarrow {A_{22}}^\intercal - Y^\intercal \cdot {A_{21}}^\intercal \\
S_3 \leftarrow S_1 - A_{22} & ST_3 \leftarrow ST_1 - {A_{22}}^\intercal \\
S_4 \leftarrow S_3 + A_{12} & ST_4 \leftarrow ST_3 + {A_{12}}^\intercal
\end{array}
$$

$$
\begin{array}{ll}
 & P_1 \leftarrow A_{11} \cdot {A_{11}}^\intercal \\
 & P_2 \leftarrow A_{12} \cdot {A_{12}}^\intercal \\
P_3 \leftarrow A_{22} \cdot ST_4 & PT_3 \leftarrow S_4 \cdot {A_{22}}^\intercal \\
P_4 \leftarrow S_1 \cdot ST_2 & PT_4 \leftarrow S_2 \cdot ST_1 \\
 & P_5 \leftarrow S_3 \cdot ST_3
\end{array}
$$

$$
\begin{array}{ll}
 & U_1 \leftarrow P_1 + P_5 \\
 & U_3 \leftarrow P_1 + P_2 \\
U_2 \leftarrow U_1 + P_4 & UT_2 \leftarrow U_1 + PT_4 \\
U_4 \leftarrow U_2 + P_3 & UT_4 \leftarrow UT_2 + PT_3 \\
 & U_5 \leftarrow U_2 + PT_4
\end{array}
$$

return $\left(\begin{smallmatrix} U_3 & UT_4 \\ U_4 & U_5 \end{smallmatrix}\right)$.

Now, it turns out that transposition is still antihomomorphic if one of the matrices has its coefficients in the base field, as shown by Lemma 45.

Lemma 45. *Let A be in $\mathbb{H}(\mathbb{K})^{m \times k}$ and Y in $\mathbb{K}^{k \times n}$, then $(A \cdot Y)^\intercal = Y^\intercal \cdot A^\intercal$.*

Proof. Since the coefficients of Y are in the base field, they commute with the quaternions. Therefore, $\forall i, j$, $\sum_k a_{jk} y_{ki} = \sum_k y_{ki} a_{jk}$. □

Now, Section 4.3 shows that for any quaternion algebra in positive characteristic, there exist a matrix Y, *in the base field*, such that $Y \cdot Y^\intercal = -\mathrm{I}_{\lfloor \frac{n}{2} \rfloor}$. Therefore, in this case, Lemma 45 shows that in Algorithm 44, $ST_1 = {S_1}^\intercal$, $ST_2 = {S_2}^\intercal$, $ST_3 = {S_3}^\intercal$, $ST_4 = {S_4}^\intercal$. This shows that not only P_1 and P_2 are multiplications of a matrix by its transpose, but also $P_5 = S_3 \cdot {S_3}^\intercal$. Finally, Algorithm 44 thus requires three recursive calls and four general multiplications.

This is:

$$P(n) \leq 3P(\frac{n}{2}) + 4\mathrm{MM}_\omega^{\mathbb{H}(\mathbb{K})}(\frac{n}{2}) + o(n^\omega) \leq 3P(\frac{n}{2}) + 32\mathrm{MM}_\omega^{\mathbb{K}}(\frac{n}{2}) + o(n^\omega) \quad (69)$$

and Equation (68) is modified as:

$$P(n) \leq \frac{32}{2^\omega - 3}\mathrm{MM}_\omega^{\mathbb{K}}(n) + o(n^\omega) \quad (70)$$

As expected this is again $8\mathrm{MM}_{\log_2(7)}^{\mathbb{K}}(n)$ if a Strassen-like algorithm is also used for the baseline over the field and $\omega = \log_2(7)$. This is again worse if $\omega < \log_2(7)$, but better, and only $(6 + \frac{2}{5})\mathrm{MM}_3^{\mathbb{K}}(n)$, if $\omega = 3$.

6.2.3 Multiplication of a quaternion matrix by its adjoint

We now deal with the case of the product of a quaternion matrix with its conjugate transpose. This operator is now an antihomomorphism which allows us to save some computations as in Algorithm 12. Here also we distinguish the matrix of quaternions from the quaternion with matrix coefficients.

Scalar case. The quaternion conjugation satisfies $\overline{XY} = \overline{Y}\,\overline{X}$. Therefore, we have:

$$(a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k})\overline{(a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k})} = a^2 + b^2 + c^2 + d^2 \quad (71)$$

For matrices again simplifications do not occur and the product is then more complex.

Using matrices of quaternions. Now $M \cdot M^H$ is a hermitian matrix and Algorithm 12 works over $\mathbb{H}(\mathbb{K})$. For this, one needs to find a skew-unitary matrix in $\mathbb{H}(\mathbb{K})$. This is impossible in $\mathbb{H}(\mathbb{C})$, but always possible in the quaternions over fields of positive characteristic using sums of squares and Equation (71).

Suppose we use a generic matrix multiplication algorithm over the quaternions with cost bound equivalent to $\mathrm{MM}_\omega^{\mathbb{H}(\mathbb{K})}(n)$ field operations. Then our Algorithm 12 can multiply a matrix of a quaternions by its conjugate transpose with a dominant complexity term bounded by $\left(\frac{2}{2^\omega - 3}\right)\mathrm{MM}_\omega^{\mathbb{H}(\mathbb{K})}(n)$ operations, by Theorem 13.

Now for the quaternions, the best algorithm to multiply any two matrices of quaternions is given by Proposition 42 and uses $8\mathrm{MM}_\omega^{\mathbb{K}}(n)$ field operations if the base field matrix multiplication uses $\mathrm{MM}_\omega^{\mathbb{K}}(n)$. We thus have proven:

Corollary 46. *Algorithm 12 multiplies a quaternion matrix by its conjugate transpose with dominant cost bounded by* $\left(\frac{16}{2^\omega - 3}\right)$ $MM_\omega^{\mathbb{K}}(n)$ *base field operations.*

Directly using quaternions with matrix coefficients. Using a quaternion with matrix coefficients over the field, we have:

$$\begin{aligned} M \cdot M^H &= (A + B\mathbf{i} + C\mathbf{j} + D\mathbf{k}) \cdot (A + B\mathbf{i} + C\mathbf{j} + D\mathbf{k})^H \\ &= (A + B\mathbf{i} + C\mathbf{j} + D\mathbf{k})(A^\intercal - B^\intercal\mathbf{i} - C^\intercal\mathbf{j} - D^\intercal\mathbf{k}) \\ &= H_1 + H_2\mathbf{i} + H_3\mathbf{j} + H_4\mathbf{k} \end{aligned} \tag{72}$$

where:

$$H_1 = AA^\intercal + BB^\intercal + CC^\intercal + DD^\intercal \tag{73}$$
$$H_2 = (BA^\intercal - AB^\intercal) + (DC^\intercal - CD^\intercal) \tag{74}$$
$$H_3 = (CA^\intercal - AC^\intercal) + (BD^\intercal - DB^\intercal) \tag{75}$$
$$H_4 = (DA^\intercal - AD^\intercal) + (CB^\intercal - BC^\intercal) \tag{76}$$

Note that H_1 is symmetric and H_2, H_3, H_4 are skew-symmetric.

The properties of the transpose in the field shows that these can be computed with $4 + 3 * 2 = 10$ multiplications (including four squares).

Now consider $E = A + B\mathbf{i}$ and $F = C + D\mathbf{i}$, so that $M = E + F\mathbf{j}$. This shows that:

$$M \cdot M^H = (E + F\mathbf{j}) \cdot (E^H + \mathbf{j}F^H) = (EE^H + FF^H) + (F\mathbf{j}E^H - E\mathbf{j}F^H) \tag{77}$$

Then, we have:

$$\begin{aligned} F\mathbf{j}E^H &= (C + D\mathbf{i})\mathbf{j}(A^\intercal - B^\intercal\mathbf{i}) \\ &= (CA^\intercal - DB^\intercal)\mathbf{j} + (DA^\intercal + CB^\intercal)\mathbf{k} \\ &= X\mathbf{j} + Y\mathbf{k} \end{aligned} \tag{78}$$

$$\begin{aligned} -E\mathbf{j}F^H &= (A + B\mathbf{i})\mathbf{j}(-C^\intercal + D^\intercal\mathbf{i}) \\ &= (-AC^\intercal + BD^\intercal)\mathbf{j} - (BC^\intercal + AD^\intercal)\mathbf{k} \\ &= -X^\intercal\mathbf{j} - Y^\intercal\mathbf{k} \end{aligned} \tag{79}$$

Using Equations (78) and (79), we thus have Algorithm 47 which uses only 6 multiplications (one of which is a square) and a total of 14 additions, 7 of them being half-additions (On the one hand, 3 multiplications and 5 additions for Equations (78) and (79) overall, then 2 half-additions for $\mathrm{Low}\,(H_3)$ and $\mathrm{Low}\,(H_4)$; on the other hand, 2 multiplications, 1 square, 3 additions and 9 half-additions for $\mathrm{Low}\,(H_1)$ and $\mathrm{Low}\,(H_2)$).

Proposition 48. *Algorithm 47 multiplies a quaternion matrix by its conjugate transpose with cost equivalent to* $\left(5 + \frac{2}{2^\omega - 3}\right) MM_\omega^{\mathbb{K}}(n)$ *base field operations.*

Proof. Algorithm 47 uses 6 multiplications, one of which, Q_6, is the product of a matrix in $\mathbb{K}$ by its transpose. □

Open question 2. *Multiply a quaternion with matrix coefficients by its Hermitian transpose in fewer than* 6 *multiplications (including* 1 *square), or with more squares.*

Algorithm 47 Fast quaternion matrix multiplications by its adjoint

Input: $A, B, C, D \in \mathbb{K}^{m\times n}$
Output: $M \cdot M^H \in \mathbb{H}(\mathbb{K})^{m\times m}$, for $M = A + B\mathbf{i} + C\mathbf{j} + D\mathbf{k}$.

$$
\begin{array}{rlrl}
U_1 & = A + B, & U_2 & = C + D \\
\hline
Q_1 & = CA^{\intercal}, & Q_2 & = DB^{\intercal} \\
Q_3 & = U_2U_1{}^{\intercal} & & \\
Q_4 & = AB^{\intercal}, & Q_5 & = CD^{\intercal} \\
Q_6 & = (U_1 + U_2)(U_1{}^{\intercal} + U_2{}^{\intercal}) & & \\
\hline
T_1 & = Q_1 - Q_2, & T_2 & = Q_4 + Q_5 \\
T_3 & = (Q_1 + Q_2) - Q_3 & & \\
\mathrm{Low}\,(H_1) & = \mathrm{Low}\,(Q_6 - (Q_3{}^{\intercal} + Q_3) - (T_2{}^{\intercal} + T_2)) & & \\
\mathrm{Low}\,(H_2) & = \mathrm{Low}\,(T_2{}^{\intercal} - T_2) & & \\
\mathrm{Low}\,(H_3) & = \mathrm{Low}\,(T_1 - T_1{}^{\intercal}) & & \\
\mathrm{Low}\,(H_4) & = \mathrm{Low}\,(T_3{}^{\intercal} - T_3) & &
\end{array}
$$

return $H_1 + H_2\mathbf{i} + H_3\mathbf{j} + H_4\mathbf{k}$.

Comparison. We summarize the results of this section about quaternion matrices in Table 3.

7 Algorithm into practice

This section reports on an implementation of Algorithm 12 over a prime field, as it is a core ingredient of any such computation in positive characteristic or over $\mathbb{Z}[i]$ or $\mathbb{Q}[i]$. In order to reduce the memory footprint and increase the data locality of the computation, we first need to identify a memory placement and a scheduling of the tasks minimizing the temporary allocations. We thus propose in Table 4 and Figure 1 a memory placement and schedule for the operation $C \leftarrow A \cdot A^{\intercal}$ using no more extra storage than the unused upper triangular part of the result C.

The more general operation $C \leftarrow \alpha A \cdot A^{\intercal} + \beta C$, is referred to as `SYRK` (Symmetric Rank k update) in the BLAS API. Table 5 and Figure 2 propose a schedule requiring only one additional $n/2 \times n/2$ temporary storage.

These algorithms have been implemented as the `fsyrk` routine in the open source `fflas-ffpack` library for dense linear algebra over a finite field [8, from commit 0a91d61e].

Figure 3 compares the computation speed in effective Gfops (a normalization, defined as $n^3/(10^9 \times \text{time})$) of this implementation over $\mathbb{Z}/131071\mathbb{Z}$ with that of the double precision BLAS routines `dsyrk`, the classical cubic-time routine over a finite field (calling `dsyrk` and performing modular reductions on the result), and the classical divide and conquer algorithm [6, § 6.3.1].

The `fflas-ffpack` library is linked with OpenBLAS [22, v0.3.6] and compiled

	Alg.	$\mathrm{MM}_3(n)$	$\mathrm{MM}_{\log_2 7}(n)$	$\mathrm{MM}_\omega(n)$
$A \cdot B$	naive	$32n^3$	$16\,\mathrm{MM}^{\mathbb{K}}_{\log_2(7)}(n)$	$16\,\mathrm{MM}^{\mathbb{K}}_{\omega}(n)$
	[14]	$16n^3$	$8\,\mathrm{MM}^{\mathbb{K}}_{\log_2(7)}(n)$	$8\,\mathrm{MM}^{\mathbb{K}}_{\omega}(n)$
$A \cdot A^H$	Alg. 47	$10.8n^3$	$5.5\,\mathrm{MM}^{\mathbb{K}}_{\log_2(7)}(n)$	$\boldsymbol{\left(\frac{2}{2^\omega-3}+5\right)\mathrm{MM}^{\mathbb{K}}_{\omega}(n)}$
	Alg. 12	$\boldsymbol{6.4n^3}$	$\boldsymbol{4\,\mathrm{MM}^{\mathbb{K}}_{\log_2(7)}(n)}$	$\frac{16}{2^\omega-3}\,\mathrm{MM}^{\mathbb{K}}_{\omega}(n)$
$A \cdot A^\intercal$	Alg. 43	$14n^3$	$\boldsymbol{7\,\mathrm{MM}^{\mathbb{K}}_{\log_2(7)}(n)}$	$\boldsymbol{7\,\mathrm{MM}^{\mathbb{K}}_{\omega}(n)}$
	Eq. (68)	$\boldsymbol{(13+\frac{1}{3})n^3}$	$8\,\mathrm{MM}^{\mathbb{K}}_{\log_2(7)}(n)$	$\frac{40}{2^\omega-2}\mathrm{MM}^{\mathbb{K}}_{\omega}(n)$
> 0 char.	Alg. 44	$\boldsymbol{(12+\frac{4}{5})n^3}$	$8\,\mathrm{MM}^{\mathbb{K}}_{\log_2(7)}(n)$	$\frac{32}{2^\omega-3}\mathrm{MM}^{\mathbb{K}}_{\omega}(n)$

Table 3: Matrix Multiplication over $\mathbb{H}(\mathbb{K})^{n\times n}$: leading term of the cost in number of operations over $\mathbb{K}$. Note that $\left(\frac{2}{2^\omega-3}+5\right) < \frac{16}{2^\omega-3}$ only when $\omega < \log_2(29) - \log_2(5) \approx 2.536$.

#	operation	loc.	#	operation	loc.
1	$S_1 = (A_{21} - A_{11}) \cdot Y$	C_{21}	9	$U_1 = P_1 + P_5$	C_{12}
2	$S_2 = A_{22} - A_{21} \cdot Y$	C_{12}		$\mathrm{Up}(U_1) = \mathrm{Low}(U_1)^\intercal$	C_{12}
3	${P_4}^\intercal = S_2 \cdot {S_1}^\intercal$	C_{22}	10	$U_2 = U_1 + P_4$	C_{12}
4	$S_3 = S_1 - A_{22}$	C_{21}	11	$U_4 = U_2 + P_3$	C_{21}
5	$P_5 = S_3 \cdot {S_3}^\intercal$	C_{12}	12	$U_5 = U_2 + {P_4}^\intercal$	C_{22}
6	$S_4 = S_3 + A_{12}$	C_{11}	13	$P_2 = A_{12} \cdot {A_{12}}^\intercal$	C_{12}
7	$P_3 = A_{22} \cdot {S_4}^\intercal$	C_{21}	14	$U_3 = P_1 + P_2$	C_{11}
8	$P_1 = A_{11} \cdot {A_{11}}^\intercal$	C_{11}			

Table 4: Memory placement and schedule of tasks to compute the lower triangular part of $C \leftarrow A \cdot A^\intercal$ when $k \leq n$. The block C_{12} of the output matrix is the only temporary used.

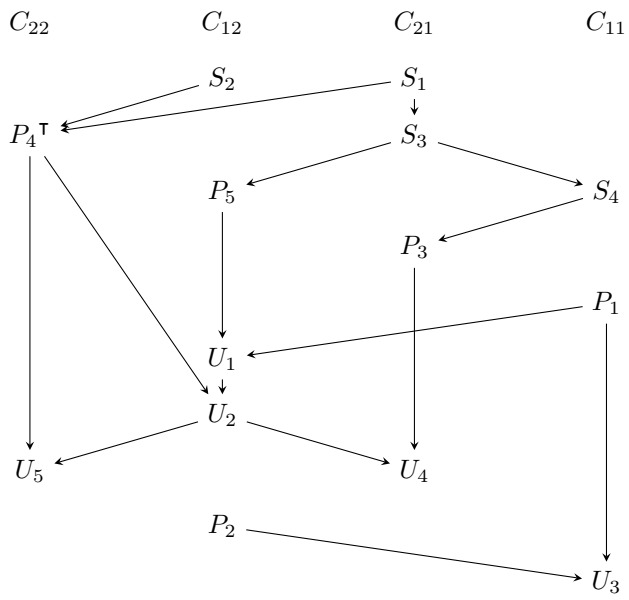

Figure 1: DAG of the tasks and their memory location for the computation of $C \leftarrow A \cdot A^\intercal$ presented in Table 4.

operation	loc.	operation	loc.
$S_1 = (A_{21} - A_{11}) \cdot Y$	tmp	$P_1 = \alpha A_{11} \cdot A_{11}{}^\intercal$	tmp
$S_2 = A_{22} - A_{21} \cdot Y$	C_{12}	$U_1 = P_1 + P_5$	C_{12}
$\mathrm{Up}(C_{11}) = \mathrm{Low}(C_{22})^\intercal$	C_{11}	$\mathrm{Up}(U_1) = \mathrm{Low}(U_1)^\intercal$	C_{12}
$P_4{}^\intercal = \alpha S_2 \cdot S_1{}^\intercal$	C_{22}	$U_2 = U_1 + P_4$	C_{12}
$S_3 = S_1 - A_{22}$	tmp	$U_4 = U_2 + P_3$	C_{21}
$P_5 = \alpha S_3 \cdot S_3{}^\intercal$	C_{12}	$U_5 = U_2 + P_4{}^\intercal + \beta \mathrm{Up}(C_{11})^\intercal$	C_{22}
$S_4 = S_3 + A_{12}$	tmp	$P_2 = \alpha A_{12} \cdot A_{12}{}^\intercal + \beta C_{11}$	C_{11}
$P_3 = \alpha A_{22} \cdot S_4{}^\intercal + \beta C_{21}$	C_{21}	$U_3 = P_1 + P_2$	C_{11}

Table 5: Memory placement and schedule of tasks to compute the lower triangular part of $C \leftarrow \alpha A \cdot A^\intercal + \beta C$ when $k \leq n$. The block C_{12} of the output matrix as well as an $n/2 \times n/2$ block tmp are used as temporary storage.

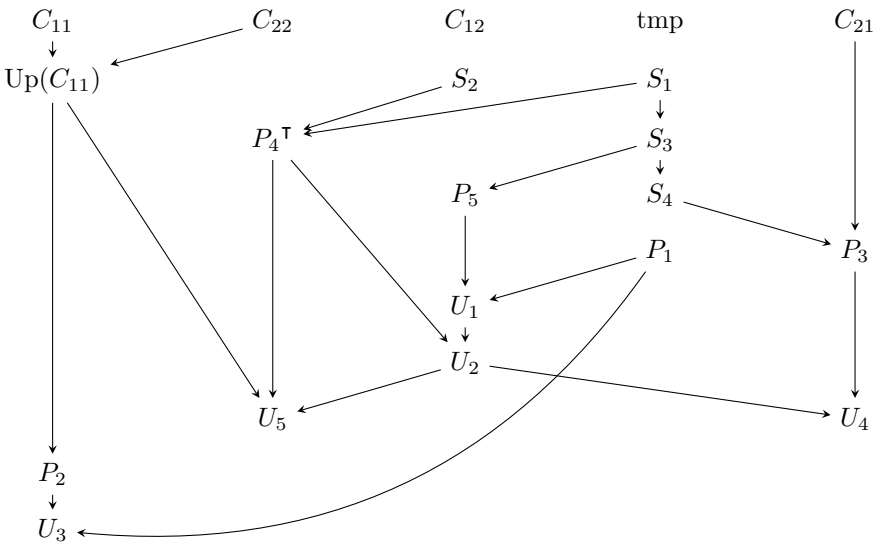

Figure 2: DAG of the tasks and their memory location for the computation of $C \leftarrow \alpha A \cdot A^\intercal + \beta C$ presented in Table 5.

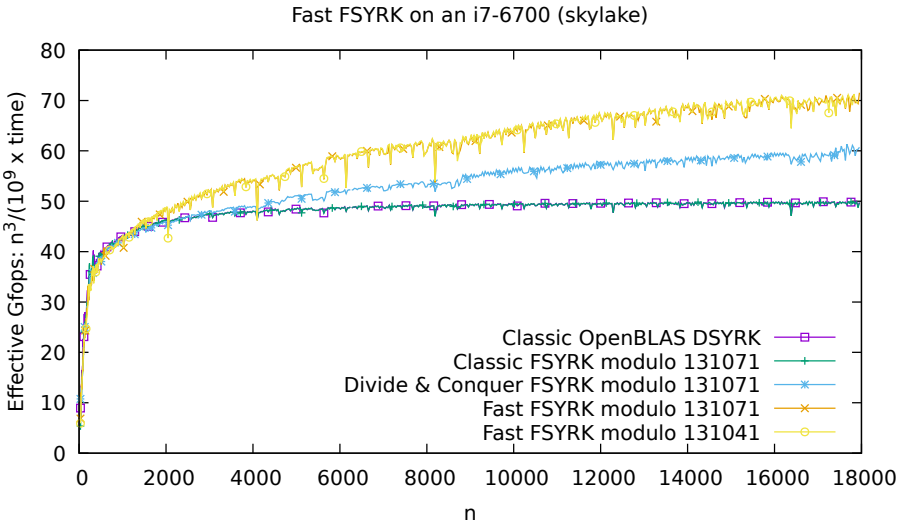

Figure 3: Speed of an implementation of Algorithm 12

with `gcc-9.2` on an Intel skylake i7-6700 running a Debian GNU/Linux system (v5.2.17).

The slight overhead of performing the modular reductions is quickly compensated by the speed-up of the sub-cubic algorithm (the threshold for a first recursive call is near $n = 2000$). The classical divide and conquer approach also speeds up the classical algorithm, but starting from a larger threshold, and hence at a slower pace. Lastly, the speed is merely identical modulo 131041, where square roots of -1 exist, thus showing the limited overhead of the preconditioning by the matrix Y.

8 Perspective

We made progresses in order to prove that five non-commutative products are necessary for the computation of the product of a 2×2 matrix by its adjoint, by applying de Groote's method to this context. However, we only prove that there is no algorithm, derived from a bilinear one, which uses 4 products and the adjoint of one of them. The case where more adjoints of already computed products could be used need to be ruled out in a similar manner. More generally, the possible existence of algorithms not originating from a bilinear algorithm is an even more challenging question.

Over the algebra of quaternions, the natural generalization of Howell and Lafon's algorithm to matrix coefficients yields the 7M and 6M algorithms for the transpose and conjugate transpose respectively. The recursive 5 products algorithm is only usable for the conjugate transpose case in positive characteristic and costs, as expected, half the cost of a general quaternion matrix product for $\omega = \log_2 7$. Yet for $\omega < 2.536$ and for the case of transposition the 6M and 7M algorithms perform best. The minimality of the number 6 of multiplications to multiply a quaternion by its conjugate is an open question, as for the minimality of the number 7 of multiplications to multiply a quaternion by its transpose. For these questions, de Groote's method could provide an answer.

We proposed several algorithms for the product of a matrix by its adjoint, each of which improves by a constant factor the best known costs, depending on the algebraic nature of the field of coefficients and on the underlying matrix exponent to be chosen. When implemented in practice the comparison may become even more complex, as other parameters, such as memory access pattern or vectorization will come into play. Our first experiments show that these constant factor improvements do have a practical impact.

References

[1] M. Baboulin, L. Giraud, and S. Gratton. A parallel distributed solver for large dense symmetric systems: Applications to geodesy and electromagnetism problems. *Int. J. of HPC Applications*, 19(4):353–363, 2005. doi:10.1177/1094342005056134.

[2] G. Beniamini and O. Schwartz. Faster matrix multiplication via sparse decomposition. In *Proc. SPAA'19*, pages 11–22, 2019. doi:10.1145/3323165.3323188.

[3] Brice Boyer, Jean-Guillaume Dumas, Clément Pernet, and Wei Zhou. Memory efficient scheduling of Strassen-Winograd's matrix multiplication algorithm. In *Proc.*, ISSAC'09, pages 135–143. ACM Press, July 2009. doi:10.1145/1576702.1576713.

[4] J. Brillhart. Note on representing a prime as a sum of two squares. *Math. of Computation*, 26(120):1011–1013, 1972. doi:10.1090/S0025-5718-1972-0314745-6.

[5] N. H. Bshouty. On the additive complexity of 2×2 matrix multiplication. *Inf. Processing Letters*, 56(6):329–335, December 1995. doi:10.1016/0020-0190(95)00176-X.

[6] J.-G. Dumas, P. Giorgi, and C. Pernet. Dense linear algebra over prime fields. *ACM TOMS*, 35(3):1–42, November 2008. doi:10.1145/1391989.1391992.

[7] Jean-Guillaume Dumas, Clément Pernet, and Alexandre Sedoglavic. On fast multiplication of a matrix by its transpose. In *Proc.*, ISSAC'20, pages 162–169, New York, July 2020. ACM Press. doi:10.1145/3373207.3404021.

[8] The FFLAS-FFPACK group. *FFLAS-FFPACK: Finite Field Linear Algebra Subroutines / Package*, 2019. v2.4.1. URL: http://github.com/linbox-team/fflas-ffpack.

[9] Charles M. Fiduccia. Fast matrix multiplication. In *Proc.*, STOC '71, pages 45–49, New York, NY, USA, 1971. ACM Press. doi:10.1145/800157.805037.

[10] Hans Friedich Groote, de. On varieties of optimal algorithms for the computation of bilinear mappings II. Optimal algorithms for 2×2-matrix multiplication. *Theoretical Computer Science*, 7(2):127–148, 1978. doi:10.1016/0304-3975(78)90045-2.

[11] Hans Friedrich Groote, de. On the complexity of quaternion multiplication. *Inf. Processing Letters*, 3(6):177 – 179, 1975. doi:10.1016/0020-0190(75)90036-8.

[12] Hans Friedrich Groote, de. On varieties of optimal algorithms for the computation of bilinear mappings I. The isotropy group of a bilinear mapping. *Theoretical Computer Science*, 7(2):1–24, 1978. doi:10.1016/0304-3975(78)90038-5.

[13] N. J. Higham. Stability of a method for multiplying complex matrices with three real matrix multiplications. *SIMAX*, 13(3):681–687, 1992. doi:10.1137/0613043.

[14] Thomas D. Howell and Jean Lafon. The complexity of the quaternion product. Technical report, Cornell University, USA, 1975. URL: https://hdl.handle.net/1813/6458.

[15] E. Karstadt and O. Schwartz. Matrix multiplication, a little faster. In *Proc. SPAA'17*, pages 101–110. ACM, 2017. doi:10.1145/3087556.3087579.

[16] Joseph M. Landsberg. *Geometry and complexity theory*, volume 169 of *Cambridge Studies in Advanced Mathematics*. Cambrigde University Press, December 2016. doi:10.1017/9781108183192.

[17] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proc ISSAC'14*, pages 296–303. ACM, 2014. doi:10.1145/2608628.2608664.

[18] G. Seroussi and A. Lempel. Factorization of symmetric matrices and trace-orthogonal bases in finite fields. *SIAM J. on Computing*, 9(4):758–767, 1980. doi:10.1137/0209059.

[19] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969. doi:10.1007/BF02165411.

[20] S. Wedeniwski. Primality tests on commutator curves. PhD U. Tübingen, 2001. URL: https://d-nb.info/963295438/34.

[21] S. Winograd. La complexité des calculs numériques. *La Recherche*, 8:956–963, 1977.

[22] Zhang Xianyi, Martin Kroeker, et al. *OpenBLAS, an Optimized BLAS library*, 2019. http://www.openblas.net/.