Redesigning Strassen's Algorithm

Undergraduate Summer Research Project 2021

Pádraig Ryan
August 23, 2021

Supervisor: Dr. John Sheekey

Abstract

We study the notions of Bilinear Complexity and tensor rank to understand Strassen's Algorithm for 2×2 matrix multiplication. We then study Grochow & Moore's *Designing Strassen's Algorithm* where a generalisation of Strassen's algorithm for $n \times n$ matrices over $\mathbb{R}$ is derived. Based on this paper, we suggest a method to search for tensor decompositions of the $n \times n$ matrix multiplication tensor, MM_n. We proceed to use the Magma Computational Algebra System to search for such tensor decompositions for certain fields $\mathbb{F}_q^n$ and $\mathbb{Z}^n$, with n and q small.

University College Dublin

Contents

1 Introduction

Tensors are important mathematical objects that have applications across various fields in mathematics and the sciences. The tensor rank, and the equivalent notion of bilinear complexity, are useful tools when facing problems involving tensors.

In this paper, we begin by defining bilinear maps, and consider results on them building up to bilinear complexity. We then switch tracks, introducing tensors and the tensor rank. Using the result that tensor rank and bilinear rank are equivalent, we examine Strassen's algorithm, and matrix multiplication in general, from the perspective of tensors.

We next consider Grochow and Moore's *Designing Strassen's Algorithm*, and consider methods to generalise their construction for the tensor decomposition of the matrix multiplication tensor, MM_n. We apply actions of the General Linear Group to sets of vectors to search for tensor decompositions of MM_n. We derive constraints on the elements of the General Linear group we can use. We then consider the Symmetric Group, and take its representation in the General Linear Group. We implement an algorithm in the Magma Computer Algebra System to search for the tensor decompositions of MM_n under the actions of elements of the Symmetric Group, over finite field vector spaces and over integer vector spaces. The main result of this paper is that our code returned valid tensor decompositions, validating our approach.

2 Notation and Assumptions

For the purposes of this report:

- Vectors, $v \in V$, in are written as column vectors.
- Dual Space covectors, $\nu \in V^*$ are written as row vectors.
- Finite fields are denoted by $\mathbb{F}$ or $\mathbb{F}_q$.
- n-dimensional $\mathbb{F}$-vector spaces are denoted as $\mathbb{F}^n$.
- $(\mathbb{F}^n)^{\otimes m} = \underbrace{\mathbb{F}^n \otimes \mathbb{F}^n \otimes \ldots \otimes \mathbb{F}^n}_{m \text{ times}}$

We take for granted that:

- $(\mathbb{F}^n)^*$ is isomorphic to $\mathbb{F}^n$ when $\mathbb{F}$ is finite dimensional.

3 Bilinear Maps

We first look at the concept of Bilinear maps between vector space. From the concept of a bilinear map, we will introduce the concept of rank. Later, we will see that a bilinear map is equivalent to a tensor, and the ranks are equivalent.

Remark 3.1. *This introduction to Bilinear Maps, Tensors and Ranks is based on Chapter 14 of Algebraic Complexity Theory*[1].

Definition 3.2 (Bilinear Map)**.** *Let U, V and W be k-vector spaces. A map $\phi : U \times V \mapsto W$ is a bilinear map if for all $u, u_i \in U, v, v_j \in V$ and $\lambda \in \mathbb{F}$ the following hold:*

$$\phi(u_1 + u_2, v) = \phi(u_1, v) + \phi(u_2, v)$$

$$\phi(u, v_1 + v_2) = \phi(u, v_1) + \phi(u, v_2)$$

$$\phi(\lambda u, v) = \lambda\phi(u, v) = \phi(u, \lambda v)$$

[1] Peter Bürgisser, Michael Clausen, and Amin Shokrollahi. Algebraic Complexity Theory. Vol. 315. Jan. 1997. ISBN: 978-3-642-08228-3. DOI: 10.1007/978-3-662-03338-8, Chapter 14.

Remark 3.3. *In words: a bilinear map, when fixing any one of its inputs, acts as a linear map on the other input.*

Example 3.4 (Matrix Multiplication forms a bilinear map.)**.** *Let $A, \alpha \in \mathbb{F}^{m\times n}$, $B, \beta \in \mathbb{F}^{n\times p}$ be matrices. Let $\phi : \mathbb{F}^{m\times n} \times \mathbb{F}^{n\times p} \mapsto \mathbb{F}^{m\times p}$ denote matrix multiplication. Let $\lambda \in \mathbb{F}$. Then:*

$$\phi(\lambda A, B) = \lambda\phi(A, B) = \phi(A, \lambda B)$$

$$\phi(A+\alpha, B)_{ij} = \sum_{l=1}^{n}(A_{il}+\alpha_{il})B_{lj} = \sum_{l=1}^{n}(A_{il}B_{lj}+\alpha_{il}B_{lj}) = \phi(A, B)_{ij}\,\phi(\alpha, B)_{ij}$$

and likewise for $\phi(A, B+\beta)$. Hence matrix multiplication is a bilinear map.

Remark 3.5. *The space of all bilinear maps from $U \times V$ to W is denoted Bil$(U, V; W)$*

Definition 3.6 (Bilinear Computation)**.** *Let $\phi : U \times V \mapsto W$ be a bilinear map. For $i \in \underline{r}$ let $f_i \in U^*$, $g_i \in V^*$, $w_i \in W$ be such that:*

$$\phi(u, v) = \sum_{i=1}^{r} f_i(u)g_i(v)w_i$$

for all $u \in U$, $v \in V$, Then $(f_1, g_1, w_1; \ldots; f_r, g_r, w_r)$ is called a bilinear computation of length r for ϕ.

Definition 3.7 (Bilinear Complexity/Rank)**.** *The length of the shortest bilinear computation for ϕ is called the bilinear complexity or bilinear rank of ϕ, and it is denoted $R(\phi/\mathbb{F})$ or just $R(\phi)$.*

Definition 3.8 (Isomorphisms of Bilinear Maps)**.** *$\phi \in$ Bil$(U, V; W)$ is isomorphic to $\phi' \in$ Bil$(U', V'; W')$ if there exist isomorphisms $\alpha : U \mapsto U', \beta : V \mapsto V', \gamma : W \mapsto W'$ such that:*

$$\gamma \circ \phi(u, v) = \phi'(\alpha(u), \beta(v)$$

If $(\alpha, \beta, \gamma) \in (GL(U), GL(V), GL(W))$, then $\phi \simeq \phi'$ and:

$$\phi'(u, v) = \gamma \circ \phi\left(\alpha^{-1}(u), \beta^{-1}(v)\right)$$

4 Tensors

Definition 4.1 (Tensor Product)**.** *Let U, V be $\mathbb{F}$-vector spaces. The tensor product is the $\mathbb{F}$-vector space $U \otimes V$, with the map $\tau \in$ Bil$(U, V; U \otimes V)$ which satisfy the universal property:*

For every $\mathbb{F}$-vector space W and every $\phi \in$ Bil$(U, V; W)$, there exists a unique $\mathbb{F}$-vector space homomorphism $\phi' : U \otimes V \to W$ such that $\phi' \circ \tau = \phi$.

Definition 4.2 (Tensor Product of vectors)**.** *Let $u \in U, v \in V$. Then $u \otimes v = \tau(u, v) \in U \otimes V$.*

Remark 4.3. *Suppose V, W are $\mathbb{F}$-Vector Spaces with bases $\{e_1, \ldots, e_n\}$ and $\{f_1, \ldots, f_m\}$ respectively. Then $V \otimes W$, is an nm-dimensional Vector Space with basis $\{e_i \otimes f_j : i \le n, j \le m\}$.*

Example 4.4. *Consider the Vector Spaces $\mathbb{F}^2$ and $(\mathbb{F}^3)^*$ with the standard basis vectors. The tensor product $W := \mathbb{F}^2 \otimes (\mathbb{F}^3)^*$ is a $2 \cdot 3 = 6$ dimensional vector space with basis vectors:*

$$e_1 \otimes f_1^* = \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, e_1 \otimes f_2^* = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}, e_1 \otimes f_3^* = \begin{bmatrix}0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}$$

$$e_2 \otimes f_1^* = \begin{bmatrix}0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix}, e_2 \otimes f_2^* = \begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}, e_2 \otimes f_3^* = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$$

4.1 Correspondence between Bilinear Maps and Tensors

Proposition 4.5 (Unique Isomorphism Between a Bilinear Map and Tensor)**.** *Let U, V, W be $\mathbb{F}$-Vector Spaces. There exists a unique isomorphism $U^* \otimes V^* \otimes W \to Bil(U, V; W)$ that sends $f \otimes g \otimes w$ to $(u, v) \mapsto f(u)g(v)w$.*

Definition 4.6 (Structural Tensor)**.** *The Structural Tensor of ϕ is the unique tensor $t \in U^* \otimes V^* \otimes W$ associated to the Bilinear Map $\phi \in Bil(U, V; W)$*

Remark 4.7. *At this stage we have defined bilinear maps and tensors, and have shown that we can treat tensors as bilinear maps and visa versa. Hence, anything we have proved for bilinear maps hold for tensors.*

Definition 4.8 (Coordinate Tensor)**.** *If we define bases $(u_i)_{i\leq n}, (v_j)_{j\leq m}, (w_i)_{i\leq q}$ on $\mathbb{F}^n, \mathbb{F}^m, \mathbb{F}^q$, and consider $\phi \in Bil(\mathbb{F}^n, \mathbb{F}^m; \mathbb{F}^q)$, then there exists $t_{ijl} \in \mathbb{F}$ such that:*

$$\phi(u_i, v_j) = \sum_{l=1}^{q} t_{ijl} e_l$$

We call $(t_{ijl})_{ijl} \in \mathbb{F}^{n\times m\times q}$ the coordinate tensor of ϕ.

Remark 4.9. *The coordinate tensor can be considered as a 3-dimensional "cuboid" array of elements of $\mathbb{F}$.*

Definition 4.10 (Tensor Slice)**.** *Let $t := (t_{ijl})_{ijl}$ be a coordinate tensor. Then $\left((t_{ijl})_{j,l} | 1 \leq i \leq n\right)$ is the sequence of 1-slices of t. Similarly, $\left((t_{ijl})_{i,l} | 1 \leq j \leq m\right)$ and $\left((t_{ijl})_{i,j} | 1 \leq l \leq q\right)$ are the sequences of 2-slices and 3-slices respectively.*

Example 4.11. *The 3-slices of MM_2 are:*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.2 Tensor Rank

Definition 4.12. *The Rank, denoted $rk(t)$, of a tensor $t \in (\mathbb{F}^n)^{\otimes n}$ is the minimum number, r, of tuples of n vectors $(u_i^1, \ldots, u_i^n)$ such that;*

$$\sum_{i=1}^{r} u_i^1 \otimes \ldots \otimes u_i^n = t$$

Remark 4.13. *The bilinear complexity, $R(\phi)$, of a bilinear map, ϕ, is equivalent to the rank, $rk(t)$ of its corresponding tensor, t.*

Example 4.14.

$$t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

has tensor rank $rk(t) = 2$, as;

$$t = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

Example 4.15 (Matrix Multiplication Tensor)**.** *The 3-slices of MM_2 are:*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It can be checked that the rank of the Matrix multiplication tensor is no more than 8 (we will see shortly its actually no more than 7).

5 Strassen's Algorithm

Volker Strassen[2] showed in 1969 that multiplying two 2×2 matrices could be done in 7 multiplications.

Example 5.1 (Strassen's Algorithm)**.** *Let A and B be 2×2 matrices.*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Then if we define the following 7 products:

$$I = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$II = (A_{21} + A_{22}) B_{11}$$
$$III = A_{11}(B_{12} - B_{22})$$
$$IV = A_{22}(-B_{11} + B_{21})$$
$$V = (A_{11} + A_{12}) B_{22}$$
$$VI = (-A_{11} + A_{21})(B_{11} + B_{12})$$
$$VII = (A_{12} - A_{22})(B_{21} + B_{22})$$

The product $C := AB$ is given by:

$$AB = \begin{bmatrix} I + IV - V + VII & III + V \\ II + IV & I + III - II + VI \end{bmatrix}$$

We can consider Strassen's Algorithm as a Bilinear Computation of the form:

$$C := \phi(A, B) = \sum_{i=1}^{r} f_i(A) g_i(B) w_i$$

Where $f_i, g_i \in (\mathbb{F}^{n \times n})^*$ and $w_i \in \mathbb{F}^{n \times n}$.

Before we define the dual vectors f_i and g_i, we want to define the dot product of matrices, $A \cdot B : \mathbb{F}^{n \times n} \times \mathbb{F}^{n \times n} \mapsto \mathbb{F}$ as follows:

$$A \cdot B := \sum_{i=1,j=1}^{n,n} A_{ij} B_{ij}$$

Hence, we can use this dot product to define f_i, g_i:

$$f_i(A) := F_i \cdot A \text{ and } g_i(B) := G_i \cdot B$$

From here we notice that Strassen's products, I through VII, are of the form $f_i(A) g_i(B)$. For example:

$$I = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \right) \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \right)$$
$$= f_1(A) g_1(B)$$

Hence,

$$F_i \in \left[\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \right]$$

[2] Volker Strassen. "Gaussian elimination is not optimal". In: Numerische Mathematik 13.4 (Aug. 1969), pp. 354–356. ISSN: 0945-3245. DOI: 10.1007/BF02165411. URL: https://doi.org/10.1007/BF02165411.

$$G_i \in \left[\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}, \begin{bmatrix}0 & 1\\0 & -1\end{bmatrix}, \begin{bmatrix}-1 & 0\\1 & 0\end{bmatrix}, \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}, \begin{bmatrix}1 & 1\\0 & 0\end{bmatrix}, \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}\right]$$

And by contemplating

$$AB = \begin{bmatrix} I + IV - V + VII & III + V \\ II + IV & I + III - II + VI \end{bmatrix},$$

we see:

$$w_i \in \left[\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, \begin{bmatrix}0 & 0\\1 & -1\end{bmatrix}, \begin{bmatrix}0 & 1\\0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0\\1 & 0\end{bmatrix}, \begin{bmatrix}-1 & 1\\0 & 0\end{bmatrix}, \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}\right]$$

And so:

$$C := \phi(A, B) = \sum_{i=1}^{r=7} f_i(A) g_i(B) w_i$$

and the bilinear complexity is no more than 7.

From here, we can consider the correspondence between bilinear maps and tensors (Proposition 4.5), giving us:

$$MM_2 := \sum_{i=1}^{7} F_i \otimes G_i \otimes w_i$$

$$\begin{aligned} MM_2 = &\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \left(\begin{bmatrix}0 & 0\\1 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} \otimes \begin{bmatrix}0 & 0\\1 & -1\end{bmatrix}\right) + \left(\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} \otimes \begin{bmatrix}0 & 1\\0 & -1\end{bmatrix} \otimes \begin{bmatrix}0 & 1\\0 & 1\end{bmatrix}\right) \\ &+ \left(\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}-1 & 0\\1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\1 & 0\end{bmatrix}\right) + \left(\begin{bmatrix}1 & 1\\0 & 0\end{bmatrix} \otimes \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}-1 & 1\\0 & 0\end{bmatrix}\right) + \left(\begin{bmatrix}-1 & 0\\1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 1\\0 & 0\end{bmatrix} \otimes \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}\right) \\ &+ \left(\begin{bmatrix}0 & 1\\0 & -1\end{bmatrix} \otimes \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}\right) \end{aligned}$$

Which clearly has tensor rank, $rk(MM_n) \leq 7$.

We can calculate the first 3-slice by considering each matrix w_i where $(w_i)_{11} \neq 0$

$$((MM_2)_{ij1})_{i,j} = (1)\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + (1)\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}-1 & 0\\1 & 0\end{bmatrix} + (-1)\begin{bmatrix}1 & 1\\0 & 0\end{bmatrix} \otimes \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix} + (1)\begin{bmatrix}0 & 1\\0 & -1\end{bmatrix} \otimes \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}$$

$$= \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & -1 & 0\\0 & 0 & 1 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\0 & -1 & 0 & -1\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\0 & 0 & 1 & 1\\0 & 0 & 0 & 0\\0 & 0 & -1 & -1\end{bmatrix}$$

$$= \begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix}$$

which is exactly the first 3-slice of MM_2 which we saw in Example 4.15. The remaining 3-slices can be calculated similarly.

6 Grochow and Moore's Algorithm

Designing Strassen's Algorithm[3] (Grochow & Moore) describes a way in which Strassen's Algorithm can be generalised to vectors of length n. They started of by defining a *Unitary 2-design*:

Definition 6.1 (Unitary 2-design)**.** *A set, S, of n-dimensional vectors is a unitary 2-design if;*

$$\sum_{v \in S} v = 0 \text{ and } \frac{1}{|S|} \sum_{v \in S} v \otimes v^* = \frac{1}{n}\boldsymbol{I}$$

[3] Joshua A. Grochow and Cristopher Moore. "Designing Strassen's algorithm". In: CoRR abs/1708.09398 (2017). arXiv: 1708.09398. URL: http://arxiv.org/abs/1708.09398.

The following theorem was then proven:

Theorem 6.2. *Let $S = \{w_1, \ldots, w_s\} \subset \mathbb{C}^n$ be a unitary 2-design, and let $s = |S|$. Then the tensor rank of MM_n is at most $s(s-1)(s-2)+1$, and the rank 1 decomposition is given by:*

$$MM_n = \boldsymbol{I}^{\otimes 3} + \frac{n^3}{s^3} \sum_{i,j,k,distinct} (w_i \otimes (w_j - w_i)^*) \otimes (w_j \otimes (w_k - w_j)^*) \otimes (w_k \otimes (w_i - w_k)^*)$$

Example 6.3. *The set, S, of vertices of an equilateral triangle (centred at the origin) form a 2-design.*

$$S := \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1/2 \\ \sqrt{3}/2 \end{bmatrix}, \begin{bmatrix} -1/2 \\ -\sqrt{3}/2 \end{bmatrix} \right\}$$

2-design Condition 1:

$$\sum_{v \in S} v = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/2 \\ \sqrt{3}/2 \end{bmatrix} + \begin{bmatrix} -1/2 \\ -\sqrt{3}/2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 - 1/2 - 1/2 \\ \sqrt{3}/2 - \sqrt{3}/2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

2-design Condition 2:

$$\frac{1}{|S|} \sum_{v \in S} v \otimes v^*$$

$$= \frac{1}{|3|} \cdot \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1/4 & -\sqrt{3}/4 \\ -\sqrt{3}/4 & 3/4 \end{bmatrix} + \begin{bmatrix} 1/4 & \sqrt{3}/4 \\ \sqrt{3}/4 & 3/4 \end{bmatrix} \right)$$

$$= \frac{1}{3} \begin{bmatrix} 3/2 & 0 \\ 0 & 3/2 \end{bmatrix} = \frac{1}{2} \boldsymbol{I}$$

Hence, by Theorem 6.2, the rank of $MM_2 = s(s-1)(s-2)+1 = 7$

7 Search for Finite Field Vectors

Grochow & Moore's construction for MM_2 will only work in fields where $\frac{1}{2}$ and $\sqrt{3}$ exist.
Can we find a construction over finite fields, or over the Integers, similar to Grochow & Moore's construction?
Hence lets consider:

$$MM_n - \mathbf{1}^{\otimes 3} = \lambda \sum_{g \in G} [g_1(w_1) \otimes (g_2(w_2))^*] \otimes [g_3(w_3) \otimes (g_4(w_4))^*] \otimes [g_5(w_5) \otimes (g_6(w_6))^*]$$
$$\in (\mathbb{F}^{n \times n})^{\otimes 3}$$

with subgroup $G \leq GL(n, \mathbb{F})^6$.

Remark 7.1. *As any element of the general linear group is an isomorphism of vector spaces, the rank of the output tensor does not change.*

We would rather work in $(\mathbb{F}^n)^{\otimes 6}$ than $(\mathbb{F}^{n \times n})^{\otimes 3}$, where it is straightforward to iterate through the vectors in the vector space.

Proposition 7.2. *We show that $\pi : (\mathbb{F}^{n \times n})^{\otimes 3} \mapsto (\mathbb{F}^n)^{\otimes 6}$:*

$$\pi\left((e_{i_1} \otimes e_{i_2}^*) \otimes (e_{i_3} \otimes e_{i_4}^*) \otimes (e_{i_5} \otimes e_{i_6}^*)\right) = e_{i_1} \otimes e_{i_2} \otimes e_{i_3} \otimes e_{i_4} \otimes e_{i_5} \otimes e_{i_6}$$

where e_{i_j} are basis vectors of $\mathbb{F}^n$, defines an isomorphism.

Proof. First note that $\mathbb{F}^{n\times n} \simeq \mathbb{F}^n \otimes (\mathbb{F}^n)^*$ Hence:

$$(\mathbb{F}^{n\times n})^{\otimes 3} \simeq (\mathbb{F}^n \otimes (\mathbb{F}^n)^*)^{\otimes 3}$$

As we are working in finite fields: $(\mathbb{F}^n)^* \simeq \mathbb{F}^n$. One such isomorphism is $h : e_i^* \mapsto e_i$. Therefore:

$$\begin{aligned}((\mathbb{F}^n \otimes (\mathbb{F}^n)^*)^{\otimes 3} &\simeq (\mathbb{F}^n \otimes \mathbb{F}^n)^{\otimes 3} \\ &\simeq ((\mathbb{F}^n)^{\otimes 2})^{\otimes 3} \\ &\simeq (\mathbb{F}^n)^{\otimes 6}\end{aligned}$$

This shows $(\mathbb{F}^{n\times n})^{\otimes 3} \simeq (\mathbb{F}^n)^{\otimes 6}$. □

Remark 7.3 (Isomorphism applied to $g \in GL(n,\mathbb{F})^6$)**.** *Let* $g = (g_1, g_2, g_3, g_4, g_5, g_6) \in GL(n,\mathbb{F})^6$

$$\begin{aligned}\pi\left(g_1(e_{i_1})\right. &\otimes (g_2(e_{i_2}))^* \otimes g_3(e_{i_3}) \otimes (g_4(e_{i_4}))^* \otimes g_5(e_{i_5}) \otimes (g_6(e_{i_6}))^*) \\ &= g_1(e_{i_1}) \otimes g_2(e_{i_2}) \otimes g_3(e_{i_3}) \otimes g_4(e_{i_4}) \otimes g_5(e_{i_5}) \otimes g_6(e_{i_6})\end{aligned}$$

We want to pick a subgroup $G \leq GL(n,\mathbb{F})^6$ that fixes MM_n and $\mathbf{1}^{\otimes 3}$.

Theorem 7.4. *Choosing* $g_1^{-T} = g_2$, $g_3^{-T} = g_4$ *and* $g_5^{-T} = g_6$ *fixes* $\pi(\boldsymbol{1}^{\otimes 3})$ *under the action of g. (Where* $H^{-T} := \left(H^{-1}\right)^T$*).*

Proof.

$$\begin{aligned}\pi(\mathbf{1}^{\otimes 3}) = \pi(\mathbf{1}\otimes\mathbf{1}\otimes\mathbf{1}) &= \pi\left(\sum_{i_1=1}^{n}\left(e_{i_1}\otimes e_{i_1}^*\right)\otimes\sum_{i_2=1}^{n}\left(e_{i_2}\otimes e_{i_2}^*\right)\otimes\sum_{i_3=1}^{n}\left(e_{i_3}\otimes e_{i_3}^*\right)\right) \\ &= \pi\left(\sum_{i_1=1,i_2=1,i_3=1}^{n,n,n} e_{i_1}\otimes e_{i_1}^*\otimes e_{i_2}\otimes e_{i_2}^*\otimes e_{i_3}\otimes e_{i_3}^*\right) \\ &\overset{6.2}{=} \sum_{i_1=1,i_2=1,i_3=1}^{n,n,n} e_{i_1}\otimes e_{i_1}\otimes e_{i_2}\otimes e_{i_2}\otimes e_{i_3}\otimes e_{i_3} \\ &= \left(\sum_{i_1=1}^{n} e_{i_1}\otimes e_{i_1}\right)\otimes\left(\sum_{i_2=1}^{n} e_{i_2}\otimes e_{i_2}\right)\otimes\left(\sum_{i_3=1}^{n} e_{i_3}\otimes e_{i_3}\right) \qquad (1)\end{aligned}$$

Similarly by 6.3:

$$g\circ\pi(\mathbf{1}^{\otimes_3}) = \ldots = \left(\sum_{i_1=1}^{n} g_1(e_{i_1})\otimes g_2(e_{i_1})\right)\otimes\left(\sum_{i_2=1}^{n} g_3(e_{i_2})\otimes g_4(e_{i_2})\right)\left(\otimes\sum_{i_3=1}^{n} g_5(e_{i_3})\otimes g_6(e_{i_3})\right)$$

As each $g_j \in GL(n,\mathbb{F})$ is a linear map;

$$g_j(e_{i_k}) = \sum_{s_j=1}^{n} (g_j)_{s_j i_k}\, e_{s_j}$$

Hence:

$$\begin{aligned}g\circ\pi(\mathbf{1}^{\otimes_3}) = \sum_{i_1=1,i_2=1,i_3=1}^{n,n,n} &\left(\sum_{s_1=1}^{n}(g_1)_{s_1 i_1}\, e_{s_1}\right)\otimes\left(\sum_{s_2=1}^{n}(g_2)_{s_2 i_1}\, e_{s_2}\right)\otimes\left(\sum_{s_3=1}^{n}(g_3)_{s_3 i_2}\, e_{s_3}\right)\otimes\left(\sum_{s_4=1}^{n}(g_4)_{s_4 i_2}\, e_{s_4}\right) \\ &\otimes\left(\sum_{s_5=1}^{n}(g_5)_{s_5 i_3}\, e_{s_5}\right)\otimes\left(\sum_{s_6=1}^{n}(g_6)_{s_6 i_3}\, e_{s_6}\right)\end{aligned}$$

$$= \left(\sum_{s_1=1}^{n}\sum_{s_2=1}^{n}\sum_{i_1=1}^{n}(g_1)_{s_1i_1}(g_2)_{s_2i_1}(e_{s_1}\otimes e_{s_2})\right)\otimes\left(\sum_{s_3=1}^{n}\sum_{s_4=1}^{n}\sum_{i_2=1}^{n}(g_3)_{s_3i_2}(g_4)_{s_4i_2}(e_{s_3}\otimes e_{s_4})\right)$$
$$\otimes\left(\sum_{s_5=1}^{n}\sum_{s_6=1}^{n}\sum_{i_3=1}^{n}(g_5)_{s_5i_3}(g_6)_{s_6i_3}(e_{s_5}\otimes e_{s_6})\right)$$
$$\overset{(1)}{=}\left(\sum_{i_1=1}^{n}e_{i_1}\otimes e_{i_1}\right)\otimes\left(\sum_{i_2=1}^{n}e_{i_2}\otimes e_{i_2}\right)\otimes\left(\sum_{i_3=1}^{n}e_{i_3}\otimes e_{i_3}\right)$$

Hence:

$$\lambda_1\left(\sum_{s_1=1}^{n}\sum_{s_2=1}^{n}\sum_{i_1=1}^{n}(g_1)_{s_1i_1}(g_2)_{s_2i_1}(e_{s_1}\otimes e_{s_2})\right)=\left(\sum_{i_1=1}^{n}e_{i_1}\otimes e_{i_1}\right)$$
$$\lambda_2\left(\sum_{s_3=1}^{n}\sum_{s_4=1}^{n}\sum_{i_2=1}^{n}(g_3)_{s_3i_2}(g_4)_{s_4i_2}(e_{s_3}\otimes e_{s_4})\right)=\left(\sum_{i_2=1}^{n}e_{i_2}\otimes e_{i_2}\right)$$
$$\lambda_3\left(\sum_{s_5=1}^{n}\sum_{s_6=1}^{n}\sum_{i_3=1}^{n}(g_5)_{s_5i_3}(g_6)_{s_6i_3}(e_{s_5}\otimes e_{s_6})\right)=\left(\sum_{i_3=1}^{n}e_{i_3}\otimes e_{i_3}\right)$$
$$\lambda_1\lambda_2\lambda_3=1$$

Where $\lambda_k\in\mathbb{F}$.
By fixing $\lambda_k=1$, and matching basis vectors,

$$\sum_{i=1}^{n}(g_k)_{s_ki}(g_l)_{s_li}=\sum_{i=1}^{n}(g_k)_{s_ki}\left(g_l^T\right)_{is_l}=g_k\cdot g_l^T=\mathbf{1}$$

Where $(k,l)\in\{(1,2),(3,4),(5,6)\}$.
And so, since $g_k\in GL(n,\mathbb{F})$ implies g_k is invertible:

$$g_1^{-T}=g_2,\ g_3^{-T}=g_4,\ g_5^{-T}=g_6$$

□

Theorem 7.5. *Choosing $g_6^{-T}=g_1$, $g_2^{-T}=g_3$ and $g_4^{-T}=g_5$ fixes $\pi(MM_n)$ under the action of g.*

Proof. Proof proceeds as in theorem 7.4, with:

$$MM_n=\sum_{i_1=1,i_2=1,i_3=1}^{n,n,n}e_{i_1}\otimes e_{i_2}^*\otimes e_{i_2}\otimes e_{i_3}^*\otimes e_{i_3}\otimes e_{i_1}^*$$

in place of $\mathbf{1}^{\otimes 3}$. □

Proposition 7.6. *Combining Theorems 7.4 and 7.5, (equivalently; choosing g to fix both $\mathbf{1}^{\otimes 3}$ and MM_n) gets us $g=\left(g_1,g_1^{-T},g_1,g_1^{-T},g_1,g_1^{-T}\right)$*

Proof.

$$\begin{aligned}g_1&\overset{7.5}{=}g_6^{-T}\\&\overset{7.4}{=}(g_5^{-T})^{-T}=g_5\\&\overset{7.5}{=}g_4^{-T}\\&\overset{7.4}{=}(g_3^{-T})^{-T}=g_3\\&\overset{7.5}{=}g_2^{-T}\\&\overset{7.4}{=}(g_1^{-T})^{-T}=g_1\end{aligned}$$

□

Theorem 7.7. *To find tensor decomposition of $MM_n - \boldsymbol{1}^{\otimes 3}$, it suffices to find $g_1 \in GL(n, \mathbb{F})$, vectors $w_1, w_2, w_3, w_4, w_5, w_6 \in \mathbb{F}^n$ and scalar $\lambda \in \mathbb{F}$ such that:*

$$\pi(MM_n - \boldsymbol{1}^{\otimes 3}) = \lambda \sum_{g \in G} g_1(w_1) \otimes g_1^{-T}(w_2) \otimes g_1(w_3) \otimes g_1^{-T}(w_4) \otimes g_1(w_5) \otimes g_1^{-T}(w_6)$$

Proof. Applying our isomorphism from Proposition 7.2 and the results of Proposition 7.6, our equation:

$$\pi(MM_n - \mathbf{1}^{\otimes 3}) = \pi \Bigg[\lambda \sum_{g \in G} [g_1(w_1) \otimes (g_2^{-T}(w_2))^*] \otimes [g_3(w_3) \otimes (g_4^{-T}(w_4))^*] \otimes [g_5(w_5) \otimes (g_6^{-T}(w_6))^*] \Bigg]$$

becomes:

$$\pi(MM_n - \mathbf{1}^{\otimes 3}) = \lambda \sum_{g \in G} g_1(w_1) \otimes g_1^{-T}(w_2) \otimes g_1(w_3) \otimes g_1^{-T}(w_4) \otimes g_1(w_5) \otimes g_1^{-T}(w_6) \in (\mathbb{F}^n)^{\otimes 6}$$

□

7.1 Magma Search

We used Magma[4] to search for solutions to the equation from Theorem 7.7;

$$\pi\left(MM_n - \mathbf{1}^{\otimes 3}\right) = \lambda \sum_{g \in G} g_1(w_1) \otimes g_1^{-T}(w_2) \otimes g_1(w_3) \otimes g_1^{-T}(w_4) \otimes g_1(w_5) \otimes g_1^{-T}(w_6)$$

The subgroup, G, was chosen to be the representation of S_{n+1} in $GL(n, \mathbb{F})$. S_{n+1} is the symmetric group on a set of $n+1$ elements, and $|S_{n+1}| = (n+1)!$.

Example 7.8 (Representation of S_3 in $GL(2, \mathbb{F})$)**.** *The subgroup S_3 is generated[5] by*

$$\begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}$$

Example 7.9 (Representation of S_4 in $GL(3, \mathbb{F})$)**.** *The subgroup S_4 is generated by*

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & -1 \\ -1 & 1 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

We wrote two programs in Magma. The first searched for solutions over a finite field $\mathbb{F}_q^n$. The second program searched for solutions over vectors of the form $v \in \{-1, 0, 1\}^n \subset \mathbb{Z}^n$.

7.2 Reducing size of Search Space

Because in the first program, we are iterating through all 6-tuples of vectors in $\mathbb{F}_q^n$, any possible reduction in the number of vectors we need to check will greatly reduce the runtime of the code.

[4] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system. I. The user language". In: J. Symbolic Comput. 24.3-4 (1997). Computational algebra and number theory (London, 1993), pp. 235–265. ISSN: 0747-7171. DOI: 10.1006/jsco.1996.0125. URL: http://dx.doi.org/10.1006/jsco.1996.0125.

[5] George Mackiw. Finite Groups of 2 x 2 Integer Matrices. Dec. 1996. URL: https://www.maa.org/sites/default/files/George_Mackiw20823.pdf.

We can reduce the number of potential solutions we need to check by doing the following:
First note that the tensor product, and elements of $GL(n, \mathbb{F})$ are linear. Hence, for example, if $w_1 = \gamma v$:

$$\begin{aligned}\pi(MM_n - \mathbf{1}^{\otimes 3}) &= \lambda \sum_{g \in G} g_1(w_1) \otimes g_1^{-T}(w_2) \otimes g_1(w_3) \otimes g_1^{-T}(w_4) \otimes g_1(w_5) \otimes g_1^{-T}(w_6) \\ &= \lambda \sum_{g \in G} g_1(\gamma v) \otimes g_1^{-T}(w_2) \otimes g_1(w_3) \otimes g_1^{-T}(w_4) \otimes g_1(w_5) \otimes g_1^{-T}(w_6) \\ &= \lambda\gamma \sum_{g \in G} g_1(v) \otimes g_1^{-T}(w_2) \otimes g_1(w_3) \otimes g_1^{-T}(w_4) \otimes g_1(w_5) \otimes g_1^{-T}(w_6)\end{aligned}$$

Letting any $w_i = \lambda_i v_i$, the scalar λ_i can be extracted from the sum as above.

Let $v \in \mathbb{F}_q^n, v \neq 0$ and consider the set $\{\lambda v : \lambda \in \mathbb{F}_q^n, \lambda \neq 0\} \subset \mathbb{F}_q^n$. As $\mathbb{F}_q^n$ is a field, every λv is unique, and hence the subset has $q-1$ distinct elements. Hence, by only checking one element in each such subset, we reduce the number of checks we need to do in each vector space from $q^n - 1$ to $\frac{q^n-1}{q-1}$. As this reduction applies to each vector space, the total search space is reduced from $(q^n-1)^6$ to $\left(\frac{q^n-1}{q-1}\right)^6$ elements.

For example, in $\mathbb{F}_3^3$, the number of 6-tuples to check is reduced from $\left(3^3-1\right)^6 = 26^6 \approx 3.1(10^8)$ to $\left(\frac{3^3-1}{3-1}\right) = 13^6 \approx 4.8(10^6)$.

A similar reduction can be achieved in the second program, for $\{-1, 0, 1\}^n$, by normalising the vectors checked such that the first non-zero component of each vector is 1.

8 Results

Note for $\mathbb{Z}$, we checked only vectors of the form $v \in \{-1, 0, 1\}^n$

Table 1: Tuples of 6 vectors satisfying Theorem 7.7

n	$\mathbb{Z}^n$	$\mathbb{F}_2^n$	$\mathbb{F}_3^n$	$\mathbb{F}_4^n$	$\mathbb{F}_5^n$	$\mathbb{F}_7^n$	$\mathbb{F}_8^n$
2	6	6	6	6	12	12	6
3	24	24	48				

However, after inspecting the vectors that the magma code produced, it was seen that if $(v_1, \ldots, v_6)$ was a list of vectors in the output, and G was a group representation of $|S_{n+1}|$, for all $g \in G$, $(g, g^{-T}, g, g^{-T}, g, g^{-T}) \circ (v_1, \ldots, v_6)$ is also a vector in the output. Identifying each of these as an equivalence class, we get the results:

Table 2: Equivalence classes of tuples of 6 vectors satisfying Theorem 7.7

n	$\mathbb{Z}^n$	$\mathbb{F}_2^n$	$\mathbb{F}_3^n$	$\mathbb{F}_4^n$	$\mathbb{F}_5^n$	$\mathbb{F}_7^n$	$\mathbb{F}_8^n$
2	1	1	1	1	2	2	1
3	1	1	2				

See Appendix A where an element from each equivalence class is listed.

9 Conclusion

We successfully derived a method for searching for tensor decompositions of the matrix multiplication tensor MM_n over fields $\mathbb{F}_q^n$ and $\mathbb{Z}^n$. This method was implemented in code in the Magma Computer Algebra System, and that code was used to find tensor decompositions of MM_n in the fields $\mathbb{Z}^2$, $\mathbb{F}_2^2$, $\mathbb{F}_3^2$, $\mathbb{F}_4^2$, $\mathbb{F}_5^2$, $\mathbb{F}_7^2$, $\mathbb{F}_8^2$, $\mathbb{Z}^3$, $\mathbb{F}_2^3$ and $\mathbb{F}_3^3$ using the symmetric group S_{n+1}.

Our method, and its implementation in code, can be used on other groups, and on different values of q and n to search for tensor decompositions of the matrix multiplication tensor, over different finite fields.

10 Further Study

- All finite fields where $n = 2, 3$ have tensor decompositions of MM_n corresponding to the decomposition in the integers, restricted to the finite field. Some finite fields, however, have multiple sets of tensor decompositions of MM_n. Is there any reason for this?
- The Symmetric Group S_{n+1} has order $(n+1)!$. Hence it only gives the minimum tensor rank when $n = 2$. For $n = 3$, it gives an upper bound of 25 for the tensor rank. This is better than the naive approach, 27, but falls short of the current best upper bound of 23. For higher values of n, it gets significantly worse. Can picking different subgroups of $GL(n, \mathbb{F})$ get a better bound?
- Is there any way to generate the vectors without searching the tensor product space?

A Appendix: Tensor Decomposition Vectors

A.1 $n = 2$

Remark A.1. *The following sets of vectors generate $\pi\left(MM_n - \mathbf{1}^{\otimes 3}\right)$ when acted on by the matrix representation of S_3 given in Example 7.8.*

$\{-1, 0, 1\}^2 \subset \mathbb{Z}^2$**:**

$$\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right)$$

$\mathbb{F}_2^2$**:**

$$\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right)$$

$\mathbb{F}_3^2$**:**

$$\left(\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right)$$

$\mathbb{F}_4^2$**:**

$$\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right)$$

$\mathbb{F}_5^2$**:**

$$\left(\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\4\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right), \left(\begin{bmatrix}1\\3\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\3\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}1\\4\end{bmatrix}\right)$$

$\mathbb{F}_7^2$**:**

$$\left(\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\6\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right), \left(\begin{bmatrix}1\\5\end{bmatrix}, \begin{bmatrix}1\\6\end{bmatrix}, \begin{bmatrix}1\\3\end{bmatrix}, \begin{bmatrix}1\\4\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}\right)$$

$\mathbb{F}_8^2$**:**

$$\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right)$$

A.2 $n = 3$

Remark A.2. *The following sets of vectors generate $\pi\left(MM_n - \mathbf{1}^{\otimes 3}\right)$ when acted on by the matrix representation of S_4 given in Example 7.9.*

$\{-1, 0, 1\}^3 \subset \mathbb{Z}^3$**:**

$$\left(\begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\-1\end{bmatrix}\right)$$

$\mathbb{F}_2^3$:

$$\left(\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}1\\1\\1\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}1\\1\\0\end{bmatrix},\begin{bmatrix}0\\1\\1\end{bmatrix}\right)$$

$\mathbb{F}_3^3$:

$$\left(\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}1\\1\\1\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}1\\1\\0\end{bmatrix},\begin{bmatrix}0\\1\\2\end{bmatrix}\right),\left(\begin{bmatrix}1\\2\\2\end{bmatrix},\begin{bmatrix}1\\1\\2\end{bmatrix},\begin{bmatrix}0\\1\\2\end{bmatrix},\begin{bmatrix}1\\0\\1\end{bmatrix},\begin{bmatrix}1\\2\\0\end{bmatrix},\begin{bmatrix}0\\1\\1\end{bmatrix}\right)$$

B Appendix: Magma Code

The Magma code is listed below, and can also be found online at:
https://github.com/Padraig-Ryan/Redesigning-Strassens-Algorithm

B.1 Code for $\mathbb{F}_q^n$

Below is the code we used to find tensor decompositions of MM_n in $\mathbb{F}_q^n$ when $q = 2, n = 2$:

```
1  //defining the  vector space to work over F_q^n
2  q := 2;
3  n := 2;
4  Q :=Rationals();
5
6  F := GF(q); // we want to work over F_q^n
7
8  //setting up vector spaces and matrix spaces
9  V := KMatrixSpace(F,n,1);
10 Vec :=VectorSpace(F,n);
11 M := KMatrixSpace(F,n,n);
12
13 //define general linear group
14 G := GL(n,F);
15
16 //S_3 for when n=2
17 if n eq 2 then;
18      gen1 := G![0,1,-1,-1];
19      gen2 := G![1,1,0,-1];
20      Gtest := sub<G|gen1,gen2>;
21 end if;
22
23 // S_4 for when n=3
24 if n eq 3 then;
25      gen1 := G![1,0,0,0,1,0,0,1,-1];
26      gen2 := G![1,0,0,1,-1,1,0,0,1];
27      gen3 := G![-1,1,0,0,1,0,0,0,1];
28      gen4 := G![0,0,-1,-1,1,-1,-1,0,0];
29      Gtest := sub<G|gen1,gen2,gen3,gen4>;
30 end if;
31
32 //SG allows us to test multiple group actions subsequently
33 SG:=[Gtest];
34
35 //---------- vectors to search when working over F_q^n ----------
36 //we generate all possible subspaces of the vector space of dimension 1
37 VR1 := {sub<V|v>: v in V | v ne V!0};
38 VR1 := {Basis(s)[1]: s in VR1}; //select one element from each subspace
39
40 //Matrix Multiplication Tensor
41 MM := function()
42  E := Basis(V);
```

```
 temp1 := TensorProduct(V!0,V!0);
 MM :=  TensorProduct(TensorProduct(temp1,temp1),temp1);
 i :=1;
 j :=1;
 k :=1;
 l :=1;
 s :=1;
 t :=1;

 while i le n do;
  while j le n do;
   while k le n do;
    temp2 :=TensorProduct(E[k],E[i]);
    temp3 :=TensorProduct(E[i],E[j]);
    temp4 :=TensorProduct(E[j],E[k]);
    MM := MM + TensorProduct(TensorProduct(temp2,temp3),temp4);
    k := k+1;

   end while;
   j := j+1;
   k := 1;
  end while;
  i := i+1;
  j:=1;
 end while;
 return MM;
end function;


//id ^{tensor product 3)
calcId := function()
 E := Basis(V);
 i :=1;
 IdIter := [[1,1,1],[1,1,2],[1,2,1],[1,2,2],[2,1,1],[2,1,2],[2,2,1],[2,2,2]];
 list1 := {};
        i:=1;
        while i le n do;
            j:=1;
     while j le n do;
                k:=1;
              while k le n do;
              A:=TensorProduct(E[i], E[i]);
              B:=TensorProduct(E[j], E[j]);
              C:=TensorProduct(E[k], E[k]);
              AB:=TensorProduct(A,B);
              list1 := list1 join {TensorProduct(AB,C)};
              k := k +1;
                 end while;
                 j:=j+1;
            end while;
            i:= i+1;
 end while;
 return &+[a: a in list1];
end function;

//Function to calculate the tensor product of 6 vectors
SixTensor := function(a,b,c,d,e,f)
 s1:=TensorProduct(a,b);
 s2:=TensorProduct(c,d);
 s3:=TensorProduct(e,f);
 return TensorProduct(TensorProduct(s1,s2),s3);

end function;

//applies GL^6 before tensor product
funct0 := function(v1,v2,v3,v4,v5,v6,g1,g2,g3,g4,g5,g6)
```

```
 return SixTensor(g1*v1,g2*v2,g3*v3,g4*v4,g5*v5,g6*v6);
end function;

//applies (g_1,g_1^-t,... in GL^6 to vectors then takes the 6 tensor product
funct3 := function(v1,v2,v3,v4,v5,v6,A)
 B := Transpose(A^-1);
 return funct0(v1,v2,v3,v4,v5,v6,A,B,A,B,A,B);
end function;

Id := calcId();
MMa := MM();
T := Parent(MMa);

//check if we got a set of tensors that match MMa-id
CheckVectorsGrow :=function(v1,v2,v3,v4,v5,v6,H)
 Sum := &+[funct3(v1,v2,v3,v4,v5,v6,A): A in H];
 return Sum in sub<T|MMa-Id> and Sum ne T!0;
        //we check the subspace above because of the lambda multiple term in the equation
end function;


//The main code that iterates through
//the representatives of the subspaces of vectors
GeneralTest := function(SG);
 for H in SG do;
  print "this group:";
  H; //print the group
  count :=0;
  for v in CartesianPower(VR1, 6) do;
   if (CheckVectorsGrow(v[1],v[2],v[3],v[4],v[5],v[6],H)) then;
    print "answer";
    v; //print the solution vectors
    count := count +1; //number of solutions
    print "end answer";
    print "";
   end if;
  end for;
  print "count: " , count;
 end for;
 return 1;
end function;

// start the search
GeneralTest(SG);

print "eof";
```

Output

```
this group:
MatrixGroup(2, GF(2))
Generators:
    [0 1]
    [1 1]

    [1 1]
    [0 1]
answer
<
    [1]
    [1],

    [1]
    [0],

    [1]
    [0],

    [1]
    [1],

    [0]
    [1],

    [0]
    [1]
>
end answer
```

```
answer
<
    [1]
    [1],

    [0]
    [1],

    [0]
    [1],

    [1]
    [1],

    [1]
    [0],

    [1]
    [0]
>
end answer

answer
<
    [1]
    [0],

    [1]
    [1],

    [0]
    [1],

    [0]
    [1],

    [1]
    [1],

    [1]
    [0]
>
end answer

answer
<
    [1]
    [0],

    [1]
    [0],

    [1]
    [1],

    [0]
    [1],

    [0]
    [1],

    [1]
    [1],

    [1]
    [0],

    [1]
    [0],

    [1]
    [1]
>
end answer

answer
<
    [0]
    [1],

    [1]
    [1]
>
end answer

answer
<
    [0]
    [1],

    [1]
    [1],

    [0]
    [1],

    [1]
    [0],

    [1]
    [1],

    [1]
    [0],

    [1]
    [0],

    [1]
    [1]
>
end answer

count:  6
1
eof
```

B.2 Code for $\mathbb{Z}^n$

Below is the code we used to find tensor decompositions of MM_n in $\mathbb{Z}^n$ when $n = 2$:

```
//defining the  vector space to work over F_q^n
n := 2;
Q :=Rationals();

```

```
F :=Rationals(); // if we want to work over Q or Z

//setting up vector spaces and matrix spaces
V := KMatrixSpace(F,n,1);
Vec :=VectorSpace(F,n);
M := KMatrixSpace(F,n,n);

//define general linear group
G := GL(n,F);

//S_3 for when n=2
if n eq 2 then;
    gen1 := G![0,1,-1,-1];
    gen2 := G![1,1,0,-1];
    Gtest := sub<G|gen1,gen2>;
end if;

// S_4 for when n=3
if n eq 3 then;
    gen1 := G![1,0,0,0,1,0,0,1,-1];
    gen2 := G![1,0,0,1,-1,1,0,0,1];
    gen3 := G![-1,1,0,0,1,0,0,0,1];
    gen4 := G![0,0,-1,-1,1,-1,-1,0,0];
    Gtest := sub<G|gen1,gen2,gen3,gen4>;
end if;

//SG allows us to test multiple group actions subsequently
SG:=[Gtest];

//---------- vectors to search when working over Z ----------
//this has similar end results to above, but we have to do it manually,
//because we only want vectors in {-1,0,1}^n
VR1 := [[-1],[0],[1]];
for i in [1..n-1] do;
 VR2 := [];
 for j in [1..#VR1] do;
  for k in [-1..1] do;
   VR2 := Append(VR2,Append(VR1[j],k));
  end for;
 end for;
 VR1 := VR2;
end for;
VR3 := [];
//Convert sequences of sequences into sequences of vectors
for i in VR1 do;
 VR3 := Append(VR3,V!i);
end for;
VR1 := VR3;
VR1 := {v: v in VR1 | v ne V!0};
//select a set of vectors that generate unique subspaces
VR1 := [v:v in VR1 | v[i][1] eq 1 where i is Min({j:j in [1..n] | v[j][1] ne 0}) ];

//Matrix Multiplication Tensor
MM := function()
 E := Basis(V);
 temp1 := TensorProduct(V!0,V!0);
 MM :=  TensorProduct(TensorProduct(temp1,temp1),temp1);
 i :=1;
 j :=1;
 k :=1;
 l :=1;
 s :=1;
 t :=1;

 while i le n do;
  while j le n do;
```

```
71      while k le n do;
72       temp2 :=TensorProduct(E[k],E[i]);
73       temp3 :=TensorProduct(E[i],E[j]);
74       temp4 :=TensorProduct(E[j],E[k]);
75       MM := MM + TensorProduct(TensorProduct(temp2,temp3),temp4);
76       k := k+1;
77
78      end while;
79      j := j+1;
80      k := 1;
81     end while;
82     i := i+1;
83     j:=1;
84   end while;
85   return MM;
86  end function;
87
88
89  //id ^{tensor product 3)
90  calcId := function()
91   E := Basis(V);
92   i :=1;
93   IdIter := [[1,1,1],[1,1,2],[1,2,1],[1,2,2],[2,1,1],[2,1,2],[2,2,1],[2,2,2]];
94   list1 := {};
95          i:=1;
96          while i le n do;
97              j:=1;
98        while j le n do;
99                  k:=1;
100               while k le n do;
101               A:=TensorProduct(E[i], E[i]);
102               B:=TensorProduct(E[j], E[j]);
103               C:=TensorProduct(E[k], E[k]);
104               AB:=TensorProduct(A,B);
105               list1 := list1 join {TensorProduct(AB,C)};
106               k := k +1;
107                  end while;
108                  j:=j+1;
109             end while;
110             i:= i+1;
111   end while;
112   return &+[a: a in list1];
113  end function;
114
115  //Function to calculate the tensor product of 6 vectors
116  SixTensor := function(a,b,c,d,e,f)
117   s1:=TensorProduct(a,b);
118   s2:=TensorProduct(c,d);
119   s3:=TensorProduct(e,f);
120   return TensorProduct(TensorProduct(s1,s2),s3);
121
122  end function;
123
124  //applies GL^6 before tensor product
125  funct0 := function(v1,v2,v3,v4,v5,v6,g1,g2,g3,g4,g5,g6)
126   return SixTensor(g1*v1,g2*v2,g3*v3,g4*v4,g5*v5,g6*v6);
127  end function;
128
129  //applies (g_1,g_1^-t,... in GL^6 to vectors then takes the 6 tensor product
130  funct3 := function(v1,v2,v3,v4,v5,v6,A)
131   B := Transpose(A^-1);
132   return funct0(v1,v2,v3,v4,v5,v6,A,B,A,B,A,B);
133  end function;
134
135  Id := calcId();
136  MMa := MM();
```

```
T := Parent(MMa);

//check if we got a set of tensors that match MMa-id
CheckVectorsGrow :=function(v1,v2,v3,v4,v5,v6,H)
 Sum := &+[funct3(v1,v2,v3,v4,v5,v6,A): A in H];
 return Sum in sub<T|MMa-Id> and Sum ne T!0;
        //we check the subspace above because of the lambda multiple term in the equation
end function;


//The main code that iterates through
//the representatives of the subspaces of vectors
GeneralTest := function(SG);
 for H in SG do;
  print "this group:";
  H; //print the group
  count :=0;
  for v in CartesianPower(VR1, 6) do;
   if (CheckVectorsGrow(v[1],v[2],v[3],v[4],v[5],v[6],H)) then;
    print "answer";
    v; //print the solution vectors
    count := count +1; //number of solutions
    print "end answer";
    print "";
   end if;
  end for;
  print "count: " , count;
 end for;
 return 1;
end function;

// start the search
GeneralTest(SG);

print "eof";
```