Gaussian elimination is not optimal, revisited

Macedo, Hugo Daniel

Published in:
Journal of Logical and Algebraic Methods in Programming

Link to article, DOI:
10.1016/j.jlamp.2016.06.003

Publication date:
2016

Document Version
Peer reviewed version

Gaussian elimination is not optimal, revisited.

Hugo Daniel Macedo[a,b,1,*]

[a] *DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark*
[b] *Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Brazil*

Abstract

We refactor the universal law for the tensor product to express matrix multiplication as the product $M \cdot N$ of two matrices M and N thus making possible to use such matrix product to encode and transform algorithms performing matrix multiplication using techniques from linear algebra. We explore such possibility and show two stepwise refinements transforming the composition $M \cdot N$ into the Naïve and Strassen's matrix multiplication algorithms.

The inspection of the stepwise transformation of the composition of matrices $M \cdot N$ into the Naïve matrix multiplication algorithm evidences that the steps of the transformation correspond to apply Gaussian elimination to the columns of M and to the lines of N therefore providing explicit evidence on why "Gaussian elimination is not optimal", the aphorism serving as the title to the succinct paper introducing Strassen's matrix multiplication algorithm.

Although the end results are equations involving matrix products, our exposition builds upon previous works on the category of matrices (and the related category of finite vector spaces) which we extend by showing: why the direct sum $(\oplus, 0)$ monoid is not closed, a biproduct encoding of Gaussian elimination, and how to further apply it in the derivation of linear algebra algorithms.

Keywords: Tensor product, Matrix multiplication, Category theory

1. Introduction

A universal law is a central concept within the framework of category theory assuring the uniqueness and existence of certain mathematical constructs. In particular, the tensor product $\otimes$ universal law [1, Theorem 22] states that every bilinear map $\beta : V \times W \to X$ is factorizable as the composition $\beta = \lceil \sigma \rceil \cdot \rho$ of a linear map $\lceil \sigma \rceil : V \otimes W \to X$ after the tensor embedding $\rho : V \times W \to V \otimes W$ which combines a pair of inputs $V \times W$, two spaces V and W, into a unique input, the tensor space $V \otimes W$.

*Corresponding author.

[1] The present work was supported by a grant from the CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil

Matrix-matrix multiplication ($\mathbb{MMM}$), the linear algebra operation at the heart of many other operations, which when applied to suitable matrices A and B, that is $\mathbb{MMM}(A, B)$, results in the matrix $C = A \times B$, is a bilinear map and therefore is suitable for the application of the universal law for the tensor product. In other words, it is possible to factorize the operation $\mathbb{MMM} : \mathbb{K}^{m\times q} \times \mathbb{K}^{q\times n} \to \mathbb{K}^{m\times n}$ as the composition of a unique linear map $\lceil\sigma\rceil : \mathbb{K}^{m\times q} \otimes \mathbb{K}^{q\times n} \to \mathbb{K}^{m\times n}$ after $\rho : \mathbb{K}^{m\times q} \times \mathbb{K}^{q\times n} \to \mathbb{K}^{m\times q} \otimes \mathbb{K}^{q\times n}$, the tensor embedding.

As it is folklore, linear maps on finite dimensional vector spaces correspond to matrices [2], thus one could question: What is the matrix involved in matrix multiplication? In this paper we address such question by reformulating it as: What is a concrete matrix representation of the linear map $\lceil\sigma\rceil$? Furthermore, given that in the context of functions between sets one can define a function that composes functions: Could one define a matrix σ that multiplies matrices?

This paper shows how an answer may be given by refactoring the tensor product universal law which is synthetically encoded as the categorical diagram (21). The generic spaces V, W, and X, interpreted as types/objects of the maps/morphisms β, $\lceil\sigma\rceil$, and ρ are instantiated with the $\mathbb{MMM}$ spaces: $\mathbb{K}^{m\times q}$, $\mathbb{K}^{q\times n}$, and $\mathbb{K}^{m\times n}$ respectively. Furthermore, β is instantiated with $\mathbb{MMM}$ in the diagram (21) and we solve the equation encoded in it to calculate a matrix σ that represents the linear map $\lceil\sigma\rceil$.

Furthermore, the quest for such matricial representation of the linear map $\lceil\sigma\rceil$ results in the factorization of $\mathbb{MMM}$ as the product of two matrices $M \cdot N$, where $M = \sigma$ and $N = (\mathbf{vec}\, A \otimes \mathbf{vec}\, B)$. Such representation makes it possible to encode and transform matrix algorithms applying basic linear algebra identities to the matrices σ and $(\mathbf{vec}\, A \otimes \mathbf{vec}\, B)$. In section 5 we explore such possibility and show matricial encodings and stepwise derivations of the algorithms performing Naïve (28) and Strassen's $\mathbb{MMM}$ (30).

The outcome of such derivations is beyond a pen and paper exercise on the derivation of matrix multiplication algorithms because through an inspection of the stepwise derivation of the Naïve algorithm performing $\mathbb{MMM}$ we explicitly find that such algorithm is the result of applying Gaussian Elimination ($\mathbb{GE}$) to the columns of σ and to the lines of $(\mathbf{vec}\, A \otimes \mathbf{vec}\, B)$. Thus the derivation leads us into a fruitful insight into a cornerstone paper [3] on matrix multiplication algorithms.

Such paper was written by Volker Strassen, a famous German mathematician known for his work on the analysis of algorithms and for introducing a $\mathbb{MMM}$ algorithm on square matrices with n lines and columns performing better — Strassen's algorithm runs in $\mathcal{O}(n^{2.807})$ time — than its Naïve version — running in $\mathcal{O}(n^3)$ time. Although the title "Gaussian elimination is not optimal" one does not find why, or an explicit connection between Gaussian elimination and the Naïve $\mathbb{MMM}$ algorithm. The derivation in section 5 explicitly shows that the Naïve algorithm for $\mathbb{MMM}$ is obtained by applying Gaussian elimination, and as the Naïve $\mathbb{MMM}$ is not an optimal algorithm then the conclusion: $\mathbb{GE}$ is not optimal.

Our full exposition expands and relies on previous research effort [4, 5, 6] on the connection between the domain of category theory, linear algebra, and

computer science. Regardless, we encourage the reader to translate the sequence denoted by $\bowtie$ in equation (28) into a product of elementary matrices and to first understand how such matrix product is performing $\mathbb{GE}$ on the $(\mathbf{vec}\, A \otimes \mathbf{vec}\, B)$ lines. Only afterwards and after exercising the matrix product formulation of Strassen's algorithm (30) using plain linear algebra we recommend delving into the full exposition expanding the research connecting category theory, linear algebra, and $\mathbb{MMM}$ algorithm derivation.

The full exposition is structured as follows: In section 2 we set up essential background and identities that are needed to understand and justify in a categorical setting the stepwise justifications. The text is mainly an adaptation of results from [4, 5] that we develop showing why when equipped with the direct sum $(\oplus, 0)$ monoid the category defined lacks closing and thus not Cartesian closed, thus not Turing complete. In section 3 we improve the translation of the traditional specification of the $\mathbb{GE}$ in terms of elementary matrices into the biproduct framework, derive new lemmas (18), and novel notation to specify the column and line application of $\mathbb{GE}$, equations (19) and (20), used in achieving the main results. In section 4 the tensor product universal law diagram is translated into an equation (27) encoding matrix multiplication in terms of a composition of two matrices. In section 5 we present a stepwise derivation of matrix multiplication algorithms, and as a byproduct we show how $\mathbb{GE}$ is related to the Naïve $\mathbb{MMM}$ algorithm. Then in section 6 we finish the exposition with some concluding remarks and possibilities for future work.

2. Background on the category of matrices

When dealing with matrices in a computational context we use a category where matrices are the morphisms and the objects transformed are natural numbers. We build upon the work in [5], but we chose not to depict matrices as arrows with right to left orientation. Although it makes sense to draw arrows backwards, due to the flow of matrix calculations, in a setting where we want to mingle matrices (linear functions) with common (non-linear) functions we opted to make morphism direction evolve.

The $\mathbf{Mat}_{\mathbb{K}}$ *category.* The category of matrices has as objects natural numbers and morphisms the linear maps, we write an arrow $n \xrightarrow{R} m$ to denote a matrix which is traditionally denoted as an object in $\mathbb{K}^{m \times n}$ for a field $\mathbb{K}$.

$$R = \begin{bmatrix} r_{11} & \dots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{m1} & \dots & r_{mn} \end{bmatrix}_{m \times n} \qquad\qquad n \xrightarrow{R} m$$

$\mathbf{Mat}_{\mathbb{K}}$ enables to define the type of a matrix at the correct abstract level when dealing with the computational structure that does not depend on the r_{ij} ele-

ments [2].

Biproducts. The category $\mathbf{Mat}_{\mathbb{K}}$ has biproducts, a *biproduct* diagram for the objects m, n is a diagram of shape

$$\begin{tikzcd} m \arrow[r, shift right, "i_1"'] & r \arrow[l, shift right, "\pi_1"'] \arrow[r, shift left, "\pi_2"] & n \arrow[l, shift left, "i_2"] \end{tikzcd}$$

whose arrows π_1, π_2, i_1, i_2 satisfy the identities which follow:

$$\pi_1 \cdot i_1 = id_m \tag{1}$$
$$\pi_2 \cdot i_2 = id_n \tag{2}$$
$$i_1 \cdot \pi_1 + i_2 \cdot \pi_2 = id_r \tag{3}$$

Morphisms π_e and i_e for $e \in 1, 2$ are termed *projections* and *injections*, respectively. And such morphisms satisfy the orthogonality property:

$$\pi_1 \cdot i_2 = 0 \tag{4}$$
$$\pi_2 \cdot i_1 = 0 \tag{5}$$

As there are as many biproduct diagrams, i.e. projections/injections pairs, as basis of vector spaces of size r, we use coherent Einstein's notation, e.g. π_e^b, i_e^b to distinguish each biproduct family, therefore b is a label indicating to which base the projections/injections belong.

$\mathbf{Mat}_{\mathbb{K}}$ *is self dual.* The dual category of $\mathbf{Mat}_{\mathbb{K}}$, that is, the category with the same objects but where every arrow is reversed, is itself, the category of matrices $\mathbf{Mat}_{\mathbb{K}}$, intuitively the dual arrow is the transposition, which means converting an arrow $n \xrightarrow{R} m$ into an arrow $m \xrightarrow{R^\top} n$, that is, source and target types (dimensions) switch over. By analogy with relation algebra, where a similar operation is termed *converse* and denoted R°, we use such notation instead of $R^\top$ and term "R converse" wherever reading R°. Index-wise, we have, for R :

$$R^\circ = \begin{bmatrix} r_{11} & \dots & r_{m1} \\ \vdots & \ddots & \vdots \\ r_{1n} & \dots & r_{mn} \end{bmatrix} \qquad\qquad m \xrightarrow{R^\circ} n$$

Standard biproducts, the architecture behind blocked linear algebra. Below we draw a diagram in which arrows of shape $n \xrightarrow{M} m$ represent matrices with n (input) columns and m (output) rows. Given matrices R, S, U, V, other universal matrices are brought to light explaining how the construction of a blocked matrix

[2] Assuming the elements are dealt with atomic machine operations, the programmer's task is to organize how to structure such operations.

(either by juxtaposition $\left[R \mid S\right]$ or stacking $\left[\frac{U}{V}\right]$) is made from the original R and S by composition with biproduct (elementary) matrices π_1 and π_2 and adding the results:

$$\begin{aligned} \left[R \mid S\right] &= R \cdot \pi_1 + S \cdot \pi_2 \\ \left[\frac{U}{V}\right] &= i_1 \cdot U + i_2 \cdot V \end{aligned}$$

$$\begin{tikzcd}
 & p & \\
m \arrow[ur, "R"] \arrow[r, shift left, "\pi_1"] & m+n \arrow[u, dashed, "{[R \mid S]}"] \arrow[l, shift left, "i_1"] \arrow[r, shift left, "\pi_2"] & n \arrow[ul, "S"'] \arrow[l, shift left, "i_2"] \\
 & q \arrow[ul, "U"] \arrow[u, dashed, "{\left[\frac{U}{V}\right]}"] \arrow[ur, "V"'] &
\end{tikzcd}$$

The computational content of projection matrices is indistinguishable from no computation. Such matrices only select some values from the input and transport it to the output unchanged, i.e. without any analysis/transformation of the input values. We denote such matrices by π. An example well studied in [5] are the $\pi_1 = [1|0]$ and $\pi_2 = [0|1]$ projections that appear naturally in the case of matrix block operations.

For instance, the following equation captures the universal law of line stacked matrix blocks.

$$X = \left[R \mid S\right] \quad \equiv \quad \begin{cases} X \cdot i_1 = R \\ X \cdot i_2 = S \end{cases} \tag{6}$$

In it, it is expressed that for every matrix X that departs from a sum of objects $m + n$ to another object p, is (up to isomorphism) the juxtaposition/column blocking of the matrices R and S, and also that such juxtaposition always exist.

As evidenced in this example, the computational result of i_1 or i_2 is just a blind positioning of elements. Such operation can be easily discarded by adjusting the operations consuming the juxtaposition.

Other universal laws may be derived as exercise, but we state the one associated to the biproduct equations that is useful in the later.

$$X = \left[\begin{array}{c|c} R_{11} & R_{12} \\ \hline R_{21} & R_{22} \end{array}\right] \quad \equiv \quad \begin{cases} \pi_1 \cdot X \cdot i_1 = R_{11} \\ \pi_1 \cdot X \cdot i_2 = R_{12} \\ \pi_2 \cdot X \cdot i_1 = R_{21} \\ \pi_2 \cdot X \cdot i_2 = R_{22} \end{cases} \tag{7}$$

From (7) we obtain a reflection property, we term it reflection because it is the solution found for the equation $id = \left[\begin{array}{c|c} R_{11} & R_{12} \\ \hline R_{21} & R_{22} \end{array}\right]$.

$$id = \left[\begin{array}{c|c} \pi_1 \cdot i_1 & \pi_1 \cdot i_2 \\ \hline \pi_2 \cdot i_1 & \pi_2 \cdot i_2 \end{array}\right] \tag{8}$$

Linear algebra calculation laws. The following equation is a core property of categories of matrices over a field $\mathbb{K}$. Whereas in categories with products and coproducts without an additive group structure on morphisms the calculation of

a juxtaposition after a stacking is blocked, in $\mathbf{Mat}_{\mathbb{K}}$ we develop the calculation into a sum.

$$\left[R \mid S\right] \cdot \left[\frac{T}{V}\right] \quad = \quad R \cdot T + S \cdot V \tag{9}$$

The following "fusion"-laws which allows a matrix to be distributed into the juxtaposition/stacking operators when in the left/right side respectively.

$$R \cdot \left[T \mid V\right] \quad = \quad \left[R \cdot T \mid R \cdot V\right] \tag{10}$$

$$\left[\frac{R}{S}\right] \cdot T \quad = \quad \left[\frac{R \cdot S}{T \cdot S}\right] \tag{11}$$

And at last the exchange law, which interchanges the possible different ways to construct blocked matrices.

$$\left[\frac{\left[R \mid S\right]}{\left[T \mid V\right]}\right] = \left[\left[\frac{R}{T}\right] \middle| \left[\frac{S}{V}\right]\right] = \left[\frac{R \mid S}{T \mid V}\right] \tag{12}$$

Although in [5] and the previous definition hint on how the dual object would correspond to the traditional notion of transposition, in [5] one also proves the following results for (standard) biproducts projections/injections pairs:

$$\begin{aligned} (\pi_e)^\circ &= i_e \\ i_e^\circ &= \pi_e \end{aligned} \tag{13}$$

which capture that the converse of the projection e of a biproduct is its correspondent injection e, in the previous biproduct $e \in \{0, 1\}$. Likewise for the converse of injections. When exercising such proofs one discovers that the converses of biproducts projections/injections are not plain matrix transposition, but that happens in special cases.

Also needed in our calculations is the following property that relates converse and stacking/juxtaposition operators:

$$\left[R \mid S\right]^\circ = \left[\frac{R^\circ}{S^\circ}\right] \tag{14}$$

$\mathbf{Mat}_{\mathbb{K}}$ *is a Cartesian category.* The biproducts and the 0-object, the natural number 0, entail a Cartesian structure in the category. The usual direct sum operator on vector spaces, $\oplus$, provides a bifunctor transforming objects m and n into the object $m + n$, and matrices R, S into $\left[\frac{R \mid 0}{0 \mid S}\right]$. The direct sum together with projections of the biproduct diagram allow to produce all binary products $(m + n)$. Given that the 0 object exists and "empty" matrices $n \longrightarrow 0$ exist, one forms the 0-ary product. By induction with base cases 0 and unary matrices, and inductive case the binary products we form a category with all binary products, and thus the category is Cartesian.

In addition, a category with the structure $(\oplus, 0)$ is usually termed as a Cartesian monoidal category. As it is well known by choosing the direct sum as monoidal operator one does not obtain a Cartesian Closed Category (CCC), a type of category that is known to model λ-calculus and therefore a Turing-complete model, thus $\mathbf{Mat}_{\mathbb{K}}$ with $(\oplus, 0)$ must be extended to obtain full computation. As an aside, let us explain why the problem relies on the exponential object that must exist to "close" the category.

$\mathbf{Mat}_{\mathbb{K}}$ *equipped with the* $(\oplus, 0)$ *monoid is not closed.* The explanation starts with an exercise on exponential diagrams. The universal law for exponentials in **Set** entails an isomorphism:

$$B \times C \to A \cong B \to A^C$$

A good exercise is to interpret matrices B and C as finite sets with size m and n, and set A to be the field $\mathbb{K}$ of interest, but let us focus on its property that is summarized in a diagram, where it is stated that a function with two inputs can be specialized by passing the first input of type B, obtaining thus $\overline{f}$, the specialized function, which is an universal construct obeying:

$$\begin{tikzcd} B \times C \arrow[r, "\overline{f} \times id"] \arrow[dr, "f"'] & A^C \times C \arrow[d, "ev"] \\ & A \end{tikzcd}$$

What happens when porting such specialized arrow to the case of matrices? One would speculate such diagram carries a rule for partial evaluation of matrix multiplication. $\overline{R}$ would be the multiplication of R with the first part of the input waiting to be fed with the second input to output the totality of the result:

$$\begin{tikzcd} n_1 \oplus n_2 \arrow[r, "\overline{R} \times id"] \arrow[dr, "R"'] & X \oplus n_2 \arrow[d, "ev"] \\ & m \end{tikzcd}$$

In fact the diagram should have the same shape and X would encode the space of matrices $n_2 \longrightarrow m$. By the shape of R and the universal law (6) we have that $R = \left[R_1 \mid R_2\right]$, thus $R = R_1 \cdot \pi_1 + R_2 \cdot \pi_2$. Moreover, the input matrix on the right of R is of the shape $\left[\dfrac{N_1}{N_2}\right]$.

The obvious way to mimic functional specialization in $\mathbf{Mat}_{\mathbb{K}}$ (meaning calculating $\overline{R}$) is to work it out at the **Set** level. Thus, assume in **Set** one has a linear function $\overline{R}$ specializing matrix R, meaning:

$$\overline{R}(N_2) = (R_1 \cdot N_1) + (R_2 \cdot N_2)$$

As the shape of the function $\overline{R}$ is of the form $b+ax$ it is an affine map, not a linear one, which contradicts our assumption. Thus there is no matrix exponential in the general case, at least if R_1 is different from 0. And therefore, due to the fact we have no exponential, the $(\oplus, 0)$ monoidal structure cannot be closed. To obtain exponentials we need the tensor and its related vectorization operation.

Vectorization. The vectorization operation is well studied in [5], and the diagram in (15) summarizes what we need to use in this paper.

$$\begin{tikzcd} \mathbb{K}^{m\times n} \arrow[rr, bend left, "\mathbf{vec}"] \arrow[rr, phantom, "\cong"] & & \mathbb{K}^{m\cdot n} \arrow[ll, bend left, "\mathbf{unvec}"] \end{tikzcd} \tag{15}$$

Such diagram expresses that the **vec** and **unvec** operations establish an isomorphism between the type of matrices $m \times n$ this is the types $n \longrightarrow m$ and the type of vectors of size $m \cdot n$, i.e. $1 \longrightarrow m \cdot n$. In this paper **vec** is performed row by row.

Diagrams and combining categories. When dealing with mathematical concepts that fit into several categories one needs to structure the ontology, and define precisely what the dots and arrows of a diagram mean. For instance, the category $\mathbf{Mat}_{\mathbb{K}}$ just defined is the skeletal category of the **FinVect** category, where morphisms are linear maps and objects are elements of a finite dimension vector space. And if we forget the linearity of such maps we start to reason in terms of **FinSet** the category with finite sets as objects and functions between such sets. In turn **FinSet** is a subcategory of **Set** where the finiteness is not a requirement.

In our work for instance the matrices are both the tabular object, but as well a linear map between finite dimension vector spaces, that are themselves functions between finite sets. To make our diagrams meaningful we always assume the objects and morphisms are in **Set** if not explicit mention is made on the underlying category. Also, because we want to do matrix calculus with some of the arrows we present in diagrams on **Set** we define a functor:

$$\lceil \_ \rceil : \mathbf{Mat}_{\mathbb{K}} \to \mathbf{Set}$$

which enable us to promote matrices to the functions at set level, thus able to lie in a **Set** diagram.

3. Encoding Gaussian elimination using biproducts

The algorithm that is run by a machine to compute the row echelon form matrix corresponding to a given input matrix is termed Gaussian elimination. Alternatively such algorithm can be expressed in terms of stepwise rewriting transformations, that correspond to multiplication of the input with special matrices, termed elementary. In this paper we rely on the second version keeping

in mind that it is possible to state $\mathbb{GE}$ procedurally without depending on matrix multiplication.

Behind the $\mathbb{GE}$ algorithm there is an isomorphism between vector spaces. The traditional algorithm transforms an input matrix into a row echelon form matrix. Such isomorphism is visible in the application of $\mathbb{GE}$ to solve linear equation systems $Rx = b$. Remember that to solve such equation one builds a matrix $\left[R \mid b\right]$, applies $\mathbb{GE}$ and later applies back substitution to obtain x. The isomorphism is visible because the operations of the algorithm are performed both in the matrix R and in the vector b, but its mathematical realization depends on the already mentioned elementary transformations. Let us delve into those.

Elementary matrices as biproducts. In the traditional curricula of linear algebra there is a notion of elementary transformations. We focus on the ones we use in this paper and are:

- Row switching - the transform $E_{i,j}$ permutes the row i with row j.
- Row addition - the transform $E_{i,j}(\alpha)$ multiplies the elements of row i with a non-zero scalar α and adds the result to row j.

Moreover, such transformations have a corresponding matrix representation, and one is able to represent such operations using the biproduct structure. Let us show projections/injections correspond to each transformations:

- Row switching - is a transform where the standard biproduct injections are used, i_1, and i_2 as above, but their positioning is crucial. Analogous to the swap mapping in functional programming, example:

$$E_{2,1} = \left[i_2 \mid i_1\right] = \left[\begin{bmatrix}0\\1\end{bmatrix} \middle| \begin{bmatrix}1\\0\end{bmatrix}\right] = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

- Row addition - is a transform where we use projections/injections which were already dealt with in [5] (for example one can find the following i_1^a and i_2^a in Section 7 where a is replaced by $'$). Such projections/injections correspond to a different biproduct and to distinguish them we write an a. The projections $i_1^a = \begin{bmatrix}1\\\alpha\end{bmatrix}$ and $i_2^a = \begin{bmatrix}0\\1\end{bmatrix}$ perform the intended matricial representation when juxtaposed:

$$E_{1,2}(\alpha) = \left[i_1^a \mid i_2^a\right] = \left[\begin{bmatrix}1\\\alpha\end{bmatrix} \middle| \begin{bmatrix}0\\1\end{bmatrix}\right] = \begin{bmatrix}1 & 0\\\alpha & 1\end{bmatrix}$$

Elementary properties. After encoding elementary matrices as biproducts one is able to use the biproduct calculus in proving the properties of such matrices, for instance we are able to build invertibility proofs for the elementary transformations. Let us exemplify one of such invertibility proofs for a case of row addition:

$$E_{i,j}(\alpha)^{-1} \cdot E_{i,j}(\alpha) = id = E_{i,j}(\alpha) \cdot E_{i,j}(\alpha)^{-1} \tag{16}$$

In the following we give a sketch of the general proof, by exemplifying the reasoning using matrices of type $2 \longrightarrow 2$. The proof evidences that the row addition operation may be leveraged on the third biproduct equation, and the biproduct calculus (note how the proof of (16) is just a rewrite of identities starting on the left-hand side of such statement until the rewrite reaches the right side).

$$
\begin{aligned}
E_{1,2}(\alpha)^{-1} \cdot E_{1,2}(\alpha) &= \left[\frac{\pi_1^a}{\pi_2^a}\right] \cdot \left[i_1^a \mid i_2^a\right] \\
&\quad \{ \text{ (11) and (10) and (12)} \} \\
&= \left[\begin{array}{c|c} \pi_1^a \cdot i_1^a & \pi_1^a \cdot i_2^a \\ \hline \pi_2^a \cdot i_1^a & \pi_2^a \cdot i_2^a \end{array}\right] \\
&\quad \{ \text{ (8) } \} \\
id &= \\
&\quad \{ \text{ (3) } \} \\
&= i_1^a \cdot \pi_1^a + i_2^a \cdot \pi_2^a \\
&\quad \{ \text{ (9)} \} \\
E_{1,2}(\alpha) \cdot E_{1,2}(\alpha)^{-1} &= \left[i_1^a \mid i_2^a\right] \cdot \left[\frac{\pi_1^a}{\pi_2^a}\right]
\end{aligned}
$$

In a similar reasoning let us show how the biproduct calculus brings insight on elementary operations by showing that the inverse of row switching is just a matter of converses:

$$E_{i,j}^{\circ} \cdot E_{i,j} = id = E_{i,j} \cdot E_{i,j}(\alpha)^{\circ} \tag{17}$$

From the proof we conclude in the case of row switching the inverse is just a converse $E_{i,j}^{-1} = E_{i,j}^{\circ}$ and the proof follows, as above, for a particular case in equational reasoning style.

$$
\begin{aligned}
E_{1,2}^{\circ} \cdot E_{1,2} &= \left[i_1 \mid i_2\right]^{\circ} \cdot \left[i_1 \mid i_2\right] \\
&\quad \{ \text{ converse of juxtapos. (14) and injections (13) } \} \\
&= \left[\frac{\pi_1}{\pi_2}\right] \cdot \left[i_1 \mid i_2\right] \\
&\quad \{ \text{ (11) and (10) and (12)} \} \\
&= \left[\begin{array}{c|c} \pi_1 \cdot i_1 & \pi_1 \cdot i_2 \\ \hline \pi_2 \cdot i_1 & \pi_2 \cdot i_2 \end{array}\right] \\
&\quad \{ \text{ (8) } \} \\
id &=
\end{aligned}
$$

$$
\begin{aligned}
& \{ \text{ (3) } \} \\
&= i_1 \cdot \pi_1 + i_2 \cdot \pi_2 \\
& \{ \text{ (9)} \} \\
&= \left[i_1 \mid i_2 \right] \cdot \left[\frac{\pi_1}{\pi_2} \right] \\
& \{ \text{ converse of juxtapos. (14) and projection (13) } \} \\
E_{1,2} \cdot E^{\circ}_{2,1} &= \left[i_1 \mid i_2 \right] \cdot \left[i_1 \mid i_2 \right]^{\circ}
\end{aligned}
$$

It immediately appears as an extension of the proof of (16) with the addition of the converse unfoldings.

The isomorphism in $\mathbb{GE}$. Now that we detailed how elementary matrices act and how the action is reversed, let us unveil the isomorphism associated with $\mathbb{GE}$. The following identity encodes the isomorphism of vector spaces one can build in terms of a sequence (we denote by $\bowtie$) of elementary matrices $E_{i,j}(\alpha)$[3], and the reversed sequence $\bowtie^{\circ}$ (The notation $x \leftarrow S$ means take elements x from sequence S in the respective order).

$$
\left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie^{\circ}} E_{i,j}(\alpha)^{-1} \right) \cdot \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie} E_{i,j}(\alpha) \right) = id \tag{18}
$$

Given a matrix R one is able to choose an adequate sequence ($\bowtie^{\circ}$) of elementary matrices $E_{i,j}(\alpha)$, which when composed in that order on the right of R perform the Gaussian elimination algorithm on the columns of R. We denote the matrix obtained by applying $\mathbb{GE}$ on the columns of a matrix R by $R^{\leftarrow}$, therefore the equality:

$$
R^{\leftarrow}_{\bowtie} = R \cdot \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie^{\circ}} E_{i,j}(\alpha)^{-1} \right) \tag{19}
$$

The reverse of the previous sequence ($\bowtie$) allows one to perform Gaussian elimination in the lines of suitable (correct dimensions and the sequence of elementary transformations achieves the desired row echelon shape) matrix L by post composition (multiplication on the left-hand side):

$$
L^{\uparrow}_{\bowtie} = \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie} E_{i,j}(\alpha) \right) \cdot L \tag{20}
$$

and as in the row case we write $L^{\uparrow}$ to denote the application of $\mathbb{GE}$ to the lines of L. We use the previous definitions to uncover the $\mathbb{GE}$ involved in $\mathbb{MMM}$ in Section 5, but before one needs to get hold of the matricial form for the multiplication.

[3] We abbreviate notation by writing row switching $E_{i,j}$ matrices as $E_{i,j}(0)$ in the sequence, although in the paper text we omit the 0 parameter.

4. Refactoring the tensor product universal law

This section shows how to express matrix multiplication as a composition of two matrices. It follows from reading the theorem related to the universal law of bilinear maps and searching for the matrix related to the unique linear map that such theorem assures to exist.

Bilinear maps universal property. Let V, W, and X be vector spaces. Every bilinear map $\beta : V \times W \to X$ can be factorized into the composition of a unique linear map $\lceil\sigma\rceil$ after the bilinear map ρ that builds the tensor product of V and W. Such property is called the universal property of tensor products and is captured in the following diagram:

$$\begin{tikzcd} V \times W \arrow[r, "\rho"] \arrow[dr, "\beta"'] & V \otimes W \arrow[d, dashed, "\lceil\sigma\rceil"] \\ & X \end{tikzcd} \tag{21}$$

which in short depicts the tensor product $V \otimes W$ as the object associated with the most general bilinear map ρ that can be built from the spaces V and W. As an aside, universal properties involve an object, in this case the tensor space, and morphisms, in this case the morphism ρ.

With a little reflection on the universal property (21) applied in the setting of finite vector spaces one may imagine if there exists a unique linear mapping then there exists a unique matrix involved in computing the product of two matrices. Following that question we instantiate (21) with the spaces of the right dimension obtaining:

$$\begin{tikzcd} \mathbb{K}^{m\times n} \times \mathbb{K}^{n\times p} \arrow[r, "\rho"] \arrow[dr, "\mathbb{MMM}"'] & \mathbb{K}^{m\times n} \otimes \mathbb{K}^{n\times p} \arrow[d, dashed, "\lceil\sigma\rceil"] \\ & \mathbb{K}^{m\times p} \end{tikzcd} \tag{22}$$

where it is not straightforward how to extract σ as a matrix. Notice how such matrix would be multiplied on the left of a matrix with $(m \cdot n)$ lines and $(n \cdot p)$ columns, thus although $\lceil\sigma\rceil$ could transform the $(m \cdot n)$ lines into the number of lines m of the desired resulting matrix, σ cannot control or change the number of columns of the resulting matrix given it is multiplied on the left side. The types explicit that, for suitable R and S the composition of $(n \times p) \xrightarrow{\rho(R,S)} (m \times n)$ and $(m \times n) \xrightarrow{\lceil\sigma\rceil} m$, is an arrow $n \times p \longrightarrow m$, but could never be $p \xrightarrow{R\cdot S} m$ as desired. In summary, a matrix on the left does not interfere with the resulting number of columns.

Before showing how to proceed to obtain $\lceil\sigma\rceil$, let us detour and reflect on the analogy with **Set**, where there is a function $(\cdot)$ composing functions. In the following diagram we observe its type as the downwards diagonal arrow

and it is also depicted the possible universal decomposition of it into the space $S = (C^A)^{(B^A)\times(C^B)}$ of the different ways to compose two functions.

$$\begin{tikzcd}
B^A \times C^B \arrow[r] \arrow[dr, "(\cdot)"'] & S \arrow[d] \\
& C^A
\end{tikzcd}$$

The difference between **Set** and $\mathbf{Mat}_{\mathbb{K}}$ which makes σ difficult to extract is the fact that in **Set** the Cartesian product $\times C^B$ is a left adjoint of $\hat{}^{C^B}$, thus in **Set** we have an exponential object, as we saw in Section 2 that is not the case in $\mathbf{Mat}_{\mathbb{K}}$.

To overcome such problem, one could change the traditional tensor product definition to be already in the expected shape $p \longrightarrow m \cdot n \cdot n$. In that case we obtain the matrix involved in the matrix product after another bilinear map. But in this paper we go further and derive a matrix computing matrix products, after tensoring not the input matrices, but their isomorphic linearized versions. For that we resort to (15), the un/vectorization operations:

$$\begin{tikzcd}
\mathbb{K}^{m\times n} \times \mathbb{K}^{n\times p} \arrow[d, "(\mathbf{vec}\times\mathbf{vec})"] & & \\
\mathbb{K}^{m\cdot n} \times \mathbb{K}^{n\cdot p} \arrow[r, "\rho"] \arrow[dr, "\lceil\mathbf{vec}\,\mathbb{MMM}\rceil"'] & \mathbb{K}^{m\cdot n} \otimes \mathbb{K}^{n\cdot p} \arrow[d, dashed, "\lceil\sigma\rceil"] \\
& \mathbb{K}^{m\cdot p} \arrow[d, "\mathbf{unvec}"] \\
& \mathbb{K}^{m\times p}
\end{tikzcd} \tag{23}$$

where besides the un/vectorizations and ρ mapping every arrow is a matrix. To reach further we can force ρ to be a matrix if we use the Cartesian product $\oplus$ of the $\mathbf{Mat}_{\mathbb{K}}$ category:

$$\begin{tikzcd}
\mathbb{K}^{m\times n} \times \mathbb{K}^{n\times p} \arrow[d, "\oplus\cdot(\mathbf{vec}\times\mathbf{vec})"] & & \\
\mathbb{K}^{m\cdot n} \oplus \mathbb{K}^{n\cdot p} \arrow[r, "\rho"] \arrow[dr, "\lceil\mathbf{vec}\,\mathbb{MMM}\rceil"'] & \mathbb{K}^{m\cdot n} \otimes \mathbb{K}^{n\cdot p} \arrow[d, dashed, "\lceil\sigma\rceil"] \\
& \mathbb{K}^{m\cdot p} \arrow[d, "\mathbf{unvec}"] \\
& \mathbb{K}^{m\times p}
\end{tikzcd} \tag{24}$$

From the diagram we obtain a characterization of matrix multiplication as a composition of a matrix and the bilinear operator generating the tensor product:

$$\lceil \mathbf{vec}\, \mathbb{MMM} \rceil \cdot \lceil (\mathbf{vec}\, R \oplus \mathbf{vec}\, S) \rceil = \lceil \sigma \rceil \cdot \rho \cdot \lceil (\mathbf{vec}\, R \oplus \mathbf{vec}\, S) \rceil \tag{25}$$

where $\lceil \sigma \rceil$ is linear, thus may be represented by a matrix. In summary, because of its type ρ admits a matricial formulation, although not unique.

Equation (25) does not look useful, in fact one defines matrix multiplication using matrix multiplication but not the R and S matrices to be multiplied. Nevertheless, it is easy to note the rightmost composition of matrices is defining the tensor of R and S:

$$\rho \cdot \lceil (\mathbf{vec}\, R \oplus \mathbf{vec}\, S) \rceil = \lceil \mathbf{vec}\, R \otimes \mathbf{vec}\, S \rceil \tag{26}$$

which can be defined using only element-wise multiplication of elements and no matrix multiplication. Thus equation (25) rewrites into:

$$\lceil \mathbf{vec}\, \mathbb{MMM} \rceil \cdot \lceil (\mathbf{vec}\, R \oplus \mathbf{vec}\, S) \rceil = \lceil \sigma \rceil \cdot \lceil (\mathbf{vec}\, R \otimes \mathbf{vec}\, S) \rceil \tag{27}$$

Therefore we factorize matrix multiplication as a matrix σ multiplied by the tensor product matrix of the vectorizations of the input matrices, which eliminates one of the matrix multiplications and when the right side is interpreted as matrices (the $\lceil \_ \rceil$ is removed) we express $\mathbb{MMM}$ as a product of two matrices $\sigma \cdot (\mathbf{vec}\, R \otimes \mathbf{vec}\, S)$.

5. Stepwise derivation of $\mathbb{MMM}$ algorithms

In this section we explore the formulation of $\mathbb{MMM}$ as a product of two matrices obtained in (27) and refine it into the Naïve and Strassen's algorithm, keeping in mind the search for the non-optimality of $\mathbb{GE}$.

5.1. Naïve $\mathbb{MMM}$ algorithm

For the sake of exposition completeness let us remember that to compute the matrix product C of matrix A and B using the Naïve algorithm, that is:

$$C = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \cdot \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right]$$

One relies on arithmetic and the algorithm prescribes the following four sums and eight multiplications:

$$C = \left[\begin{array}{c|c} A_{11} \times B_{11} + A_{12} \times B_{21} & A_{11} \times B_{12} + A_{12} \times B_{22} \\ \hline A_{21} \times B_{11} + A_{22} \times B_{21} & A_{21} \times B_{12} + A_{22} \times B_{22} \end{array}\right]$$

Given there are algorithms requiring less operations the algorithm is not optimal. Moreover, in such shape one cannot explicitly find the $\mathbb{GE}$ in it.

But we can explore the factorization of $\mathbb{MMM}$ as the product $\sigma \cdot (\mathbf{vec}\, A \otimes \mathbf{vec}\, B)$ to encode and derive algorithms. To perform a stepwise derivation of

the Naïve $\mathbb{MMM}$ algorithm we need only to consider its vectorized form and start form the left side of (27). We appeal to the hidden identity which lies in every composition dot to make Gaussian elimination appear in the matrix product algorithm as shown in the following derivation:

$$
\begin{aligned}
&\mathbf{vec}\,\mathbb{MMM} \cdot (\mathbf{vec}\,A \oplus \mathbf{vec}\,B) \\
&\{ \text{ (27) and } (\cdot) \text{ identity } \} \\
&= \sigma \cdot id \cdot (\mathbf{vec}\,A \otimes \mathbf{vec}\,B) \\
&\{ \text{ iso. (18) and set } \bowtie = [E_{7,1}(1), E_{8,2}(1), E_{15,9}(1), E_{16,10}(1), E_{9,3}, E_{10,4}] \} \\
&= \sigma \cdot \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie^{\circ}} E_{i,j}(\alpha)^{-1} \right) \cdot \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie} E_{i,j}(\alpha) \right) \cdot (\mathbf{vec}\,A \otimes \mathbf{vec}\,B) \\
&\{ \text{ def. of column } \mathbb{GE} \text{ (19) and } (\cdot)\text{-associativity } \} \\
&= \sigma^{\leftarrow}_{\bowtie} \cdot \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie} E_{i,j}(\alpha) \cdot (\mathbf{vec}\,A \otimes \mathbf{vec}\,B) \right) \\
&\{ \text{ def. of row } \mathbb{GE} \text{ (20) } \} \\
&= \sigma^{\leftarrow}_{\bowtie} \cdot (\mathbf{vec}\,A \otimes \mathbf{vec}\,B)^{\uparrow}_{\bowtie}
\end{aligned}
$$

A thorough inspection after performing the matrix arithmetic reveals $\sigma^{\leftarrow}_{\bowtie}$ is computationally innocuous, because is corresponds to a projection π_1 for a biproduct where $c = (4 + 12)$ and $(\mathbf{vec}\,A \otimes \mathbf{vec}\,B)^{\uparrow}_{\bowtie}$ is the space containing the Naïve $\mathbb{MMM}$ algorithm in the subspace of size 4 that π_1 projects. Therefore, equation:

$$\mathbf{vec}\,\text{Naïve}\mathbb{MMM}(A, B) = \pi_1 \cdot (\mathbf{vec}\,A \otimes \mathbf{vec}\,B)^{\uparrow}_{\bowtie} \tag{28}$$

encodes a non-optimal algorithm and the derivation shown evidences the algorithm is the result of applying $\mathbb{GE}$. Thus one wonders if that is what Volker Strassen meant with: "Gaussian elimination is not optimal" [3].

5.2. Strassen's $\mathbb{MMM}$ algorithm

Our work allows the study of the space of matrix multiplication algorithms in matricial form, thus future research on what are the different space transforms which are used to achieve different algorithms to multiply matrices is in order, as a first step in that direction let us show how to encode Strassen's algorithm in matrix form.

The algorithm performs better than the Naïve matrix multiplication using only 7 multiplication operations instead of 8. The strategy to multiply:

$$C = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array} \right] \cdot \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array} \right]$$

is to define seven matrices M_j for $j \in \{1, \ldots, 7\}$ as

- $M_1 := (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$
- $M_2 := (A_{21} + A_{22}) \cdot B_{11}$
- $M_3 := A_{11} \cdot (B_{12} - B_{22})$
- $M_4 := A_{22} \cdot (B_{21} - B_{11})$
- $M_5 := (A_{11} + A_{12}) \cdot B_{22}$
- $M_6 := (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$
- $M_7 := (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$

and then obtain the resulting matrix summing such M_j matrices as follows:

$$C = \left[\begin{array}{c|c} M_1 + M_4 - M_5 + M7 & M_3 + M_5 \\ \hline M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{array} \right]$$

Using our approach, we obtain Strassen's algorithm (in this paper we show only the case where $A_{i,j}$ and $B_{i,j}$ are constants) again by performing elementary transformations, but those are not the ones entailing $\mathbb{GE}$, therefore the algorithm improves the non-optimal Naïve multiplication algorithm. The algorithm has again the shape we obtained previously:

$$\sigma \cdot \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie^{\circ}} E_{i,j}(\alpha)^{-1} \right) \cdot \left(\prod_{E_{i,j}(\alpha) \leftarrow \bowtie} E_{i,j}(\alpha) \right) \cdot (\mathbf{vec}\, A \otimes \mathbf{vec}\, B) \qquad (29)$$

but this time $\bowtie$ is a concatenation of the following components corresponding to calculate each M_j matrix.

- $\underset{M_6}{\bowtie} = [E_{9,10}(1), E_{1,10}(-1), E_{2,10}(-1)]$
- $\underset{M_1}{\bowtie} = [E_{13,1}(1), E_{4,1}(1), E_{16,1}(1)]$
- $\underset{M_3}{\bowtie} = [E_{4,2}(-1)]$
- $\underset{M_5}{\bowtie} = [E_{8,4}(1)]$
- $\underset{M_7}{\bowtie} = [E_{7,8}(1), E_{15,8}(-1), E_{16,8}(-1)]$
- $\underset{M_4}{\bowtie} = [E_{13,15}(-1)]$
- $\underset{M_2}{\bowtie} = [E_{9,13}(1)]$

and with the components performing the sums:

- $\underset{C_{22}}{\bowtie} = [E_{1,10}(1), E_{2,10}(1), E_{13,10}(-1)]$

- $\underset{C_{11}}{\bowtie} = [E_{15,1}(1), E_{4,1}(-1), E_{8,1}(1)]$
- $\underset{C_{12}}{\bowtie} = [E_{2,4}(1)]$
- $\underset{C_{21}}{\bowtie} = [E_{13,15}(1)]$

and if one wants to obtain σ as a projection one has to apply as well $\underset{P}{\bowtie} = [E_{4,2}, E_{15,3}, E_{10,4}]$. Therefore, to encode Strassen's algorithm let us define:

$$\bowtie = \underset{M_6}{\bowtie} \mid \underset{M_1}{\bowtie} \mid \underset{M_3}{\bowtie} \mid \underset{M_5}{\bowtie} \mid \underset{M_7}{\bowtie} \mid \underset{M_4}{\bowtie} \mid \underset{M_2}{\bowtie} \mid \underset{C_{22}}{\bowtie} \mid \underset{C_{11}}{\bowtie} \mid \underset{C_{12}}{\bowtie} \mid \underset{C_{21}}{\bowtie} \mid \underset{P}{\bowtie}$$

and factorize equation (29) into the shape of Strassen's algorithm by defining the sequences: $\underset{\mathcal{C}}{\bowtie} = \underset{C_{22}}{\bowtie} \mid \underset{C_{11}}{\bowtie} \mid \underset{C_{12}}{\bowtie} \mid \underset{C_{21}}{\bowtie} \mid \underset{P}{\bowtie}$ and $\underset{\mathcal{D}}{\bowtie} = \underset{M_6}{\bowtie} \mid \underset{M_1}{\bowtie} \mid \underset{M_3}{\bowtie} \mid \underset{M_5}{\bowtie} \mid \underset{M_7}{\bowtie} \mid \underset{M_4}{\bowtie} \mid \underset{M_2}{\bowtie}$ to obtain the encoding of the algorithm in following form:

$$\mathbf{vec}\,\mathrm{Strassen}\mathbb{MMM}(A, B) = \left(\sigma_{\bowtie}^{\leftarrow} \cdot \prod_{\underset{\mathcal{C}}{\bowtie}} E_{i,j}(\alpha) \right) \cdot (\mathbf{vec}\,A \otimes \mathbf{vec}\,B)^{\uparrow}_{\underset{\mathcal{D}}{\bowtie}} \tag{30}$$

where one observes a clear separation between the conquer phase where one sums the matrices M_j and the divide phase where the calculation of such matrices is performed. Such separation mirrors the two phases of the Strassen algorithm, but one should notice several issues pop in our formulation: 1 - the order of calculation of M_j matters, 2 - the rearrangement $\underset{P}{\bowtie}$ is redundant.

6. Conclusion

The main question addressed in this paper is the search for the matrix σ related to the linear map $\lceil \sigma \rceil$ prescribed to uniquely exist for every bilinear map, thus for matrix-matrix multiplication ($\mathbb{MMM}$) algorithms. As a byproduct of such search, we refactor the tensor product universal law and express $\mathbb{MMM}$ as a multiplication of matrices: $\sigma \cdot (\mathbf{vec}\,A \otimes \mathbf{vec}\,B)$. We explore such expression to encode the Naïve and Strassen's algorithms after applying stepwise refinements resulting in equations (28) and (30) respectively. As observed in section 5 the stepwise derivation of the Naïve algorithm corresponds to apply $\mathbb{GE}$ to the columns of σ and to the rows of $(\mathbf{vec}\,A \otimes \mathbf{vec}\,B)$. Such observation reinforces the statement: "Gaussian elimination is not optimal" [3].

Regarding the search for the matrix which multiplies matrices we conclude such matrix is not as generic as the function composing functions $(f \cdot g)(x)$ which lies in a setting enabling the expression of: "apply function g to the input x and the result of such operation $g(x)$ to function f". In the matrix case the matrix composing matrices depends on the formulation of the tensor embedding (26) which one is able to express in matrix form, but it depends on the input matrices A and B and does not work as a matrix.

Our work expands previous research that has been applied to several computer science domains [6, 7, 8, 9], but our exposition steps further some leaps in the achievement of the original goal of combining category theory, linear algebra, and computer science in the derivation of $\mathbb{MMM}$ algorithms [4, 5]. Our work relies on an improved expression of the $\mathbb{GE}$ using biproducts in section 3 and adapts the original formulations to combine general set functions and finite linear maps.

We envisage future research to extend our approach to other algorithms. The approach can be used to study and derive practical algorithms with optimal running times depending on the particular architectures. In addition, given the current belief asserting $\mathbb{MMM}$ is in $\Omega(n^2)$ we envisage a different proof for such and for the optimal theoretical algorithm based on a good setting of projections/injections... After all it is a quest for "the" optimal multiplication.

Acknowledgements.

The author would like to thank Edward Hermann Haeusler and CNPq-Brasil for hosting and allowing this work. Thanks to the reviewers work and for the insightful requests as is the case of the Strassen's algorithm that derived into subsection 5.2 and further conclusions.

[1] G. Birkhoff, S. Mac Lane, A survey of modern algebra (1977).

[2] S. Lang, Linear algebra. Undergraduate texts in mathematics (1987).

[3] V. Strassen, Gaussian elimination is not optimal, Numerische Mathematik 13 (4) (1969) 354–356. doi:10.1007/BF02165411.

[4] H. Macedo, J. Oliveira, Matrices As Arrows! A Biproduct Approach to Typed Linear Algebra, in: Mathematics of Program Construction, Vol. 6120 of Lecture Notes in Computer Science, Springer, 2010, pp. 271–287. doi:10.1007/978-3-642-13321-3_16.

[5] H. Macedo, J. Oliveira, Typing linear algebra: A biproduct-oriented approach, Science of Computer Programming 78 (11) (2013) 2160–2191. doi:10.1016/j.scico.2012.07.012.

[6] H. D. Macedo, J. N. Oliveira, A linear algebra approach to OLAP, Formal Aspects of Computing 27 (2) (2015) 283–307. doi:10.1007/s00165-014-0316-9.

[7] J. N. Oliveira, Towards a linear algebra of programming, Formal Aspects of Computing 24 (4) (2012) 433–458. doi:10.1007/s00165-012-0240-9.

[8] J. Oliveira, V. Miraldo, "Keep definition, change category" - a practical approach to state-based system calculi, Journal of Logical and Algebraic Methods in Programming (2015) .doi:10.1016/j.jlamp.2015.11.007.

[9] D. Murta, J. Oliveira, A study of risk-aware program transformation, Science of Computer Programming 110 (2015) 51 – 77. doi:10.1016/j.scico.2015.04.008.