# L2 MINIMAL ALGORITHMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Abigail Louise Turner

August 2021

L2 MINIMAL ALGORITHMS

Abigail Louise Turner, Ph.D.

Cornell University 2021

We consider putting certain tensors into forms with approximately minimum L2 norm. These tensors describe strategies for computing linear or bilinear maps. Such forms are of interest from a practical perspective because they are particularly numerically stable. They are of interest from a theoretical perspective because they may be unique up to certain orthogonal or unitary transformations.

The main tensors of interest represent (commutative, real) "matrix multiplication algorithms" or "bilinear algorithms." We explain how an algorithm's L2 minimal form might be thought of as optimally stable and as close to attaining the nuclear norm as possible. We demonstrate an algorithm "Strop" that has minimum $L2$ norm among all rank 7 algorithms for $2 \times 2$ matrix multiplication. This leads to better error for typical large matrices than Strassen, at the cost of more operations. Putting Strassen's (or any such) algorithm into this form requires "only" a convex optimization problem on positive definite matrices. In some situations, this optimization enables us to check when algorithms are equivalent in a natural sense.

As a stepping stone we consider L2 minimal forms for analogous "linear algorithms." Such forms have been the subject of extensive study by invariant theorists, but are less known in other circles. We give simple examples of when the existence of these forms, and their use in equivalence testing, is apparent from an optimization perspective.

## BIOGRAPHICAL SKETCH

Abby was born in Northbrook, IL on August 15, 1993. She got her undergraduate degree at the University of Illinois Urbana-Champaign in 2015.

## ACKNOWLEDGEMENTS

To all of the friends, family, and mentors that supported me on my journey (and tolerated my ignoring their messages).

## TABLE OF CONTENTS

## LIST OF FIGURES

CHAPTER 1

# INTRODUCTION

## 1.1 Overview

Consider some bilinear map $\mathcal{M} : V \times V \to V$ for $V$ a (finite dimensional) real vector space. We say $\{(A_i, B_i, C_i)\}_1^r$ is an algorithm for this map assuming $\sum_{i=1}^r \langle A_i, A\rangle\langle B_i, B\rangle C_i^T = \mathcal{M}(A, B)$. Here $\langle\cdot,\cdot\rangle$ represents the usual inner product (also known as the dot product). Such algorithms always exist, but need not be unique.

Thus one might view different algorithms as different strategies for computing $\mathcal{M}(A, B)$. Algorithms with $r$ small may offer faster strategies for computing $\mathcal{M}$. For example, Strassen's algorithm gives a way to multiply $A, B \in \mathbb{R}^{2\times 2}$ using 7 terms [24]: The standard strategy taught in linear algebra is described with $2^3 = 8$ terms. Thus by applying Strassen recursively, larger matrices $A, B \in \mathbb{R}^{n\times n}$ may be multiplied with complexity better than $O(n^3)$.

Algorithms for which $\|A_i\|, \|B_i\|$ and $\|C_i\|$ are small may offer more stable strategies for computing $\mathcal{M}$. To see this intuitively, note for example that an error $E_A$ in our measurement of $A$ will be magnified by larger $A_i$:

$$\begin{aligned}\mathcal{M}(A + E_A, B) - \mathcal{M}(A, B) &= \sum_1^r \langle A_i, A + E_A\rangle\langle B_i, B\rangle C_i^T - \mathcal{M}(A, B)\\ &= \sum_1^r (\langle A_i, A\rangle + \langle A_i, E_A\rangle)\langle B_i, B\rangle C_i^T - \mathcal{M}(A, B)\\ &= \sum_1^r \langle A_i, E_A\rangle\langle B_i, B\rangle C_i^T.\end{aligned}$$

Given an algorithm $\{(A_i, B_i, C_i)\}_1^r$ for $\mathcal{M}$, one can generate other (equivalent)

algorithms for $\mathcal{M}$ by applying certain transformations to $\{(A_i, B_i, C_i)\}_1^r$. The transformations of interest in this thesis are those that yield particularly stable algorithms. The measure of stability we choose is $\sum_1^r \|A_i\|_2\|B_i\|_2\|C_i\|_2$, a relative of the well-known nuclear norm.

The reason for this choice is amenability to optimization: Finding an equivalent algorithm that has (approximately) optimal $\sum_1^r \|A_i\|_2\|B_i\|_2\|C_i\|_2$ is relatively straightforward. Further, putting algorithms into these optimal forms can be helpful for determining when a given $\{(A_i, B_i, C_i)\}_1^r$ and $\{(A_i', B_i', C_i')\}_1^r$ are equivalent.

The fourth chapter of the thesis shows some numerical examples of this that we have considered (chapter three gives some explanation of how to arrive at these numerics). Since most are more familiar with linear maps than bilinear maps, the second chapter does an example in the linear setting. If one thinks of matrices $\{M_i\}_1^r$ in $\mathbb{R}^{n\times n}$ as strategies for computing $M = \sum_1^r M_i$, then $\{PM_iP^{-1}\}_1^r$ for $P \in \mathrm{GL}(n, \mathbb{R})$ are strategies for computing similar linear maps. One can use optimization to construct strategies $\{N_i\}_1^r = \{PM_iP^{-1}\}_1^r$ such that $\sum_1^r \|N_i\|_2^2$ is minimal. This optimization can be helpful for checking whether $\{M_i\}_1^r$ and $\{N_i\}_1^r$ satisfy $\{M_i\}_1^r = \{PN_iP^{-1}\}_1^r$ for some $P \in \mathrm{GL}(n, \mathbb{R})$.

## 1.2 Glossary

This section defines some recurring terms in the thesis. All but the most common should be reintroduced and cited as needed. Thus this section is just intended as a reference guide, in case one gets lost in symbols. Of course, all definitions could be done in considerably more generality. We stick to cases

relevant in this thesis.

**Definition 1** *Trace Operator*

*Given a matrix* $M \in \mathbb{R}^{n \times n}$ *one has that* $tr(M) = \sum_{i=1}^{n} M_{ii}$.

*Fact: This operator is linear and also satisfies* $tr(AB) = tr(BA)$.

**Definition 2** *Vector-Vector Outer Product*

*Given vectors* $v_1, v_2 \in \mathbb{R}^{n \times 1}$, *define the outer product* $v_1 \otimes v_2$ *to be the matrix* $M = v_1 v_2^T$ *with entries denoted by* $M_{ij} = (v_1)_i (v_2)_j$.

*Given vectors* $v_1, v_2, v_3 \in \mathbb{R}^{n \times 1}$, *define the outer product* $(v_1 \otimes v_2) \otimes v_3$ *to be* $M \in \mathbb{R}^{n \times n \times n}$ *with entries denoted by* $M_{ijk} = (v_1)_i (v_2)_j (v_3)_k$.

*We say* $v_1 \otimes v_2$ *and* $v_1 \otimes v_2 \otimes v_3$ *are rank 1.*

**Definition 3** *Matrix-Matrix Outer Product*

*Given matrices* $M_1, M_2 \in \mathbb{R}^{n \times n}$, *define the outer product* $M_1 \otimes M_2$ *to be* $M \in \mathbb{R}^{n \times n \times n \times n}$ *with entries denoted by* $M_{i_1 j_1 i_2 j_2} = (M_1)_{i_1 j_1} (M_2)_{i_2 j_2}$.

*Given matrices* $M_1, M_2, M_3 \in \mathbb{R}^{n \times n}$, *define the outer product* $(M_1 \otimes M_2) \otimes M_3$ *to be* $M \in \mathbb{R}^{n \times n \times n \times n \times n \times n}$ *with entries denoted by* $M_{i_1 j_1 i_2 j_2 i_3 j_3} = (M_1 \otimes M_2)_{i_1 j_1 i_2 j_2} (M_3)_{i_3 j_3}$.

*We say* $M_1 \otimes M_2$ *and* $M_1 \otimes M_2 \otimes M_3$ *are rank 1.*

**Definition 4** *Inner Product*

*Given* $M_1, M_2 \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_k}$, *we use* $\langle M_1, M_2 \rangle$ *to denote the usual inner product:* $\langle M_1, M_2 \rangle = \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=1}^{n_k} (M_1)_{i_1 \ldots i_k} (M_2)_{i_1 \ldots i_k}$.

**Definition 5** *L2 Norm*

*Given* $M \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_k}$, *denote* $\|M\|_2 = \langle M, M \rangle^{\frac{1}{2}}$.

*Fact: For* $v_1, v_2, v_3 \in \mathbb{R}^{n \times 1}$ *then* $\|v_1 \otimes v_2\|_2 = \|v_1\|_2 \|v_2\|_2$ *and* $\|v_1 \otimes v_2 \otimes v_3\|_2 =$

$||v_1||_2||v_2||_2||v_3||_2$. *Similarly, for* $M_1, M_2, M_3 \in \mathbb{R}^{n\times n}$, *one has that* $||M_1 \otimes M_2||_2 = ||M_1||_2||M_2||_2$ *and* $||M_1 \otimes M_2 \otimes M_3||_2 = ||M_1||_2||M_2||_2||M_3||_2$.

*Fact: For* $M \in \mathbb{R}^{n_1\times n_2}$ *and orthogonal* $P \in \mathbb{R}^{n_1\times n_1}, Q \in \mathbb{R}^{n_2\times n_2}$ *one has* $||PMQ||_2 = ||M||_2$. *For such* $M$, $||M||_2$ *is also commonly denoted* $||M||_F = tr(MM^T)$ *(the Frobenius norm).*

**Definition 6** *Linear Tensor*

*Recall that a linear map* $\mathcal{M} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ *can be represented by a matrix* $M \in \mathbb{R}^{n_1\times n_2}$. *We call a linear map equipped with such a representation a linear tensor.*

**Definition 7** *Linear Algorithms*

*Given a linear tensor* $\mathcal{M}$, *define a linear (rank r) algorithm for this tensor to be some* $\{M_i\}_{i=1}^r \subset \mathbb{R}^{n_1\times n_2}$ *such that* $M = \sum_{i=1}^r M_i$ *for rank 1* $M_i$.

**Definition 8** *Rank of a Linear Tensor*

*The minimum r for which there exists a rank r algorithm for a linear tensor is said to be the rank of* $\mathcal{M}$. *Note that this must agree with the rank of the matrix* $M$.

**Definition 9** *Orbit of a Matrix Under the Action of Conjugation*

*We define the orbit of a matrix* $M \in \mathbb{R}^{n\times n}$ *under the action of conjugation to be* $\{PMP^{-1} \mid P \in \mathbb{R}^{n\times n}, det(P) \neq 0\}$.

*Such* $PMP^{-1}$ *represent similar linear maps.*

**Definition 10** *Orbit of a Linear Algorithm Under the Action of Simultaneous Conjugation*

*We define the orbit of a linear algorithm* $\{M_i\}_{i=1}^r \subset \mathbb{R}^{n\times n}$ *to be the collection of* $\{PM_iP^{-1}\}_{i=1}^r$ *for* $P \in \mathbb{R}^{n\times n}$ *with* $det(P) \neq 0$.

**Definition 11** *Bilinear Tensor*

*A bilinear map $\mathcal{M} : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$ can be represented by an element of $M \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. We call a bilinear map equipped with such a representation a bilinear tensor.*

**Definition 12** *Bilinear Algorithms*

*Given a bilinear tensor $\mathcal{M}$, define a bilinear (rank r) algorithm for this tensor to be some $\{M_i\}_{i=1}^r \subset \mathbb{R}^{n_1 \times n_2 \times n_3}$ such that $M = \sum_{i=1}^r M_i$ for rank 1 $M_i$.*

**Definition 13** *Rank of a Bilinear Tensor*

*The minimum r for which there exists a rank r algorithm for a bilinear tensor is said to be the rank of $\mathcal{M}$.*

**Definition 14** *Bilinear Algorithms (Alternative)*

*Given a bilinear tensor $\mathcal{M}$, if $\mathcal{M}$ can also be represented by the sum of outer products of r tuples $(M_1, M_2, M_3) \subset \mathbb{R}^{n \times n}$ then one can also think of $\{M_1 \otimes M_2 \otimes M_3\}_1^r$ as an algorithm for $\mathcal{M}$. Note by Definition 3 that this lives in $\mathbb{R}^{n \times n \times n \times n \times n \times n}$ rather than $\mathbb{R}^{n_1 \times n_2 \times n_3}$ (taking $\mathbb{R}^{n_1}, \mathbb{R}^{n_2}, \mathbb{R}^{n_3}$ to be $\mathbb{R}^{n \times n}$).*

**Definition 15** *Orbit of a Bilinear Algorithm Under the Action of Invertible $(P, Q, R)$*

*Given a bilinear algorithm of the just mentioned form for some $\mathcal{M}$ and $(P, Q, R)$ in $\mathbb{R}^{n \times n}$ such that $det(P)det(Q)det(R) \neq 0$, one can define the orbit of this algorithm by $\{PM_1Q^{-1} \otimes QM_2R^{-1} \otimes RM_3P^{-1}\}_1^r$.*

**Definition 16** *Positive Definite Matrix*

*A symmetric matrix $M \in \mathbb{R}^{n \times n}$ is called positive definite when $x^T Mx > 0$ for all nonzero $x \in \mathbb{R}^n$. We denote the set of such matrices by $\{M > 0\}$.*

**Definition 17** *Gram Matrix*

*Given real vectors $v_1, \ldots, v_n$ the columns of some $X$ then the Gram matrix is given by $X^T X$. When these vectors are linearly independent, this is a positive definite matrix (and every positive definite matrix may be expressed as a Gram matrix).*

**Definition 18** *Positive Definite Square Root*

*Given a positive definite matrix $M$, then $M$ has a unique positive definite root $\sqrt{M}$ such that $\sqrt{M}\sqrt{M} = M$. If one expresses $M = UDU^T$ for $U$ orthogonal, then one can write $\sqrt{M} = U\sqrt{D}U^T$. Interchangeably, we use the notation $M^{\frac{1}{2}}$. The positive definite square root is the only one used in this thesis.*

**Definition 19** *Manifold of Positive Definite Matrices*

*Chapter 1 considers the set $\{M > 0\}$ equipped with the Riemannian metric $\langle A, B\rangle_M = tr(M^{-1}AM^{-1}B)$.*

*Fact: The resulting space is a complete metric space [1].*

**Definition 20** *Product Space of Positive Definite Matrices*

*Chapter 2 considers the set $\{M > 0\}^3$ equipped with the product metric $\langle (P_1, Q_1, R_1), (P_2, Q_2, R_2)\rangle_{(X,Y,Z)} = tr(X^{-1}P_1X^{-1}P_2) + tr(Y^{-1}Q_1Y^{-1}Q_2) + tr(Z^{-1}R_1Z^{-1}R_2)$.*

**Remark 1 (Ekeland [12])** *Given a complete metric space $(X, d)$ and a lower semicontinuous function $f : X \to \mathbb{R} \cup \{\infty\}$ that is bounded below and not identically equal to infinity, there are nearly optimal solutions to the problem $inf_{x\in X} f(x)$.*

## 1.3 Background

Perhaps the most difficult part of this thesis is understanding exactly where the natural equivalence relations on our objects come from. Our best attempt at a discussion of this is encapsulated in a section at the start of each chapter. An entwined difficulty will be separating a tensor from its many interpretations. Recall the situation in linear algebra (the setting of the first chapter): One is given matrices in $\mathbb{R}^{n_1 \times n_2}$, but interprets them as linear maps or bilinear forms. Similarly for multilinear algebra, the same tensors may be thought of as representing various multilinear maps or forms.

Aside from these philosophical queries, the thesis aims to be straightforward. We'll need only basics about bilinear maps and convex optimization, introduced as needed (or see the glossary for a primer).

The main idea is to start with some tensor, represented by a box of numbers $M \in \mathbb{R}^{n_1 \times n_2}$ or $\mathbb{R}^{n_1 \times n_2 \times n_3}$. We are interested in fixed length "algorithms" for this tensor: (finite, ordered) lists of $\{M_i\}_1^r$ such that $\sum_1^r M_i = M$. These algorithms offer strategies for computing $M$. In the linear case, such strategies are straightforward to find. For bilinear maps, this is challenging but can sometimes be done by machine learning or other numerical techniques [13, 23]. Changes of bases (acting in a natural way) that fix $M$ need not fix $\{M_i\}_1^r$. Thus we can use these changes of basis to generate other algorithms that have smaller $\sum_1^r \|M_i\|_2^2$. Minimal algorithms are often more stable and have interesting properties.

Many texts were helpful along the way. These should be referenced where appropriate, but we include our nonexpert memory map of them here for anyone wishing to further explore the material.

For background on working with matrices, see Matrix Analysis [17] and the Matrix Cookbook [22]. For our purposes, Positive Definite Matrices [5] is also helpful.

For background on optimization, An Introduction to Optimization on Manifolds [7] is a good start (the corresponding toolbox Manopt is strongly recommended). For more specifics on our spaces, Introduction to Riemannian Manifolds [20] and Convex Analysis and Optimization in Hadamard Spaces [1]. Lectures on Convex Optimization [21] and Ekeland [12] are also foundational.

For relevant complexity theory and background on Strassen's and other bilinear algorithms, both Algebraic Complexity Theory [8] and Geometry and Complexity Theory [19] are valuable guides.

CHAPTER 2

# LINEAR TENSORS

We start by giving an example in the setting of linear maps. This setting is a bit degenerate (as there are not many ways to compute a linear map). But it is useful for building concepts. In this chapter, elements of $\mathbb{R}^n$ are represented by column vectors. Thus elements of the dual space $(\mathbb{R}^n)^*$ are represented by row vectors. (Loosely, if you want, this amounts to picking between matrices and their transposes).

Consider some linear map from $\mathbb{R}^n$ to $\mathbb{R}^n$, represented by a matrix in $M \in \mathbb{R}^{n\times n}$. This is an example of a tensor.

## 2.1 Linear Algorithms

As alluded to, we will define an algorithm for this tensor to be some set $\{M_k\}_1^r \subset \mathbb{R}^{n\times n}$ such that $M = \sum_k M_k$ and $M_k$ are rank 1. Then, the action of $M$ on $\mathbb{R}^n, (\mathbb{R}^n)^*$, and $\mathbb{R}^n \times (\mathbb{R}^n)^*$ can be understood using $\{M_k\}$. For example, we may compute

$$Mv = M(\textstyle\sum_i \alpha_i e_i) = \alpha_i \sum_i Me_i = \alpha_i \sum_i (\sum_k M_k)\, e_i = \alpha_i \sum_i (\sum_k M_k e_i)\,.$$

Similarly for $v^T M$. To compute $v_1^T M v_2$ note that the properties of the trace operator imply

$$\begin{aligned} v_1^T M v_2 &= \mathrm{tr}(v_1^T M v_2) = \mathrm{tr}(v_2 v_1^T M) = \mathrm{tr}((\textstyle\sum_{i,j} \alpha_{i,j} e_i e_j^T) M) \\ &= \textstyle\sum_{i,j} \alpha_{i,j} \mathrm{tr}(e_i e_j^T M) = \sum_{i,j} \alpha_{i,j} e_j^T M e_i = \sum_{i,j} \alpha_{i,j} (\sum_k e_j^T M_k e_i). \end{aligned}$$

Algorithms with fewer terms and more zeros may offer cheaper ways to compute our tensor. Now recall from linear algebra that matrices $M$ and $PMP^{-1}$

represent the same linear map up to change of basis (for $P$ some invertible $n \times n$ matrix). Thus we might wonder, given an algorithm $\{M_i\}$ for $M$, what do the algorithms $\{PM_iP^{-1}\}$ look like? The answers are of course very different depending on whether we allow $P \in \mathrm{GL}(n, \mathbb{C})$ or just $P \in \mathrm{GL}(n, \mathbb{R})$. We use the latter for this example.

To confound us, notice that this set need not be closed: For example, there may be $\{N_i\}$ such that $\{P_kM_iP_k^{-1}\}$ tends to $\{N_i\}$ (in some matrix norm), but reaching $\{N_i\}$ would require $P_k$ singular. Working in a fixed finite precision, it won't always be possible to distinguish such $\{N_i\}$ from $\{M_i\}$.

### 2.1.1 Algorithm Orbits

Recall that we were studying the $\{PM_iP^{-1}\}$, given $\{M_i\}$ some algorithm for $M$. We'll call the set $\{PM_iP^{-1}\}$ the orbit of the algorithm (under the simultaneous conjugation action by $\mathrm{GL}(n, \mathbb{R})$).

A helpful thing to notice: For each $i$, $\|PM_iP^{-1}\|_2$ is bounded below by Weyl's inequality [17] (assuming $M_i$ nonzero). So whatever algorithms are in our orbit must also have $r$ elements.

Thus, it is enough for us to study cardinality $r$ sets $\{N_i\}$ to study the orbit of the algorithm.

## 2.2 Norm-Based Invariants

Consider the following quantity: $f(\{M_i\}) = \inf_{P \in \mathrm{GL}(n,\mathbb{R})} \sum_1^r \|PM_iP^{-1}\|_2^2$. Clearly, if there is $P \in \mathrm{GL}(n,\mathbb{R})$ such that $\{N_i\} = \{PM_iP^{-1}\}$ then $f(\{N_i\}) = f(\{M_i\})$. In fact, it is enough for the closure of the orbits to intersect. Whether this holds can be checked with reasonable efficiency [11].

We don't have the time or expertise to discuss these results. But we'll think a little about a couple cases from an optimization perspective.

### 2.2.1 Examples of L2 Minimal Forms

One thing $f(\{M_i\}) = \inf_{P \in \mathrm{GL}(n,\mathbb{R})} \sum_1^r \|PM_iP^{-1}\|_2^2$ has going for it, surprisingly, is that it is not too challenging to approximate. Theorem 1 of the appendix gives a proof that this can be formulated as a convex optimization problem over the space of positive definite matrices.

Many other results can then be naturally seen from the optimization perspective. For example, when $M_i$ have no common invariant subspaces it is clear that the inf must be uniquely attained up to orthogonal actions. This is theorems 2 and 3 in the appendix.

Of course much more could be said. But this has been enough to move on to the bilinear case.

CHAPTER 3

# BILINEAR TENSORS

We now leave the "friendly" world of linear maps behind. Consider some bilinear map from $\mathbb{R}^n \times \mathbb{R}^n$ to $\mathbb{R}^n$, represented by a box $M \in \mathbb{R}^{n\times n\times n}$. This is an example of a tensor.

Our $M$ now has rows, columns, and depths. Denote rowwise slices of $M$ by $\{M_i\}$, the columnwise slices by $\{M_j\}$, and the depthwise slices by $\{M_k\}$. Applying $M$ to $(a, b) \in \mathbb{R}^n \times \mathbb{R}^n$ might be done by computing one of $\{a^T M_i b\}, \{a^T M_j b\}, \{a^T M_k b\}$ (or perhaps $\{a^T M_i^T b\}, \{a^T M_j^T b\}, \{a^T M_k^T b\}$). Of course, especially compared to the linear case, there are many choices.

## 3.1 Bilinear Algorithms

As in linear algebra, every $M \in \mathbb{R}^{n\times n\times n}$ has a decomposition into simple outer products $M = \sum_{i=1}^{r} a_i \otimes b_i \otimes c_i$ for $a_i, b_i, c_i \in \mathbb{R}^n$.

An $\{(a_i, b_i, c_i)\}_1^r$ that satisfy this equation are sometimes called an algorithm for $M$ [8]. We also use the term algorithm to describe the outer products $\{a_i \otimes b_i \otimes c_i\}_1^r \subset \mathbb{R}^{n\times n\times n}$, specifying as needed. Every algorithm in the former sense specifies an algorithm in the latter sense. Algorithms in the latter sense may specify many algorithms in the former.

The cardinality and sparsity of algorithms can give some indication of how time consuming they are. The minimum $r$ for which there exists an algorithm is known as the rank of $M$. Thus if $\{(a_i, b_i, c_i)\}_1^r$ has $r$ elements, this is sometimes called a rank $r$ or length $r$ algorithm [15]. Can the actions of bilinear maps be

understood from learning such algorithms, analogous to the matrix case? We next dash these hopes before trying to bandage them.

## 3.2 Matrix Multiplication Algorithms

Consider one of the most common bilinear maps: Matrix multiplication. Let's keep it square for now, so this map $M$ takes $A, B \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$ to $M(A, B) = AB \in \mathbb{R}^{n\times n}$.

It is also helpful to think about this map as defining a trilinear form on $\mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$: One might wish to apply the transpose (dual) of $M(A, B) = AB \in \mathbb{R}^{n\times n}$ to $C \in \mathbb{R}^{n\times n}$. Since these are real matrices, this could be done using the trace inner product $M(A, B, C) = \langle (AB)^T, C\rangle = \langle AB, C^T\rangle = \mathrm{tr}(ABC) = \mathrm{tr}(B^T A^T C^T)$.

Thus real algorithms for this tensor can be represented by $\mathcal{M} = \{(A_i, B_i, C_i)\}_1^r$ in $(\mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n})^r$ such that $AB = \sum_1^r \langle A_i, A\rangle\langle B_i, B\rangle C_i^T$ and $\mathrm{tr}(ABC) = \sum_1^r \langle A_i, A\rangle\langle B_i, B\rangle\langle C_i, C\rangle$. The inner product here is now the usual one on $\vec{M}_i, \vec{M}$, in matrix language $\langle M_i, M\rangle = \mathrm{tr}(M_i^T M)$.

The standard $O(n^3)$ matrix multiplication algorithm taught in linear algebra is described by a rank $n^3$ algorithm for $M$. Algorithms with fewer terms, for example Strassen's, are so-called fast matrix multiplication algorithms with better complexity. These algorithms naturally lead to improved complexity bounds for various linear algebra problems [9]. One drawback of such algorithms is that error guarantees become more difficult [6]. Nevertheless, error encountered in practice is better than expected and algorithms such as Strassen's have been suggested as a useful alternative to standard implementations [3, 2].

At this point in the discussion one might be tempted to think that $\mathcal{M}$ should fit into our earlier framework. It seems this object could be represented in $\mathbb{R}^{n^2} \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^2}$. The catch is the vectorization operator is not canonical: By flattening matrices, information has already been lost. This may be obvious to some, but let's think about it more closely.

These interpretations of $\mathcal{M}$ coexist perfectly naturally. Real algorithms may be stored (approximately) in a computer by matrices $\{(A_i, B_i, C_i)\}_1^r$. Applying them (approximately) to $(A, B)$ or $(A, B, C)$ can be done with "just" scaling and addition (inner products). So in the usual function notation we might be tempted to write something like $\mathcal{M}(A, B, C) = \text{tr}(ABC)$ and $\mathcal{M}(A, B) = AB$. And of course $\mathcal{M}$ can be used to compute both these quantities.

But we could have also applied $\mathcal{M}(A, B)$ to $C$ recursively to compute $CAB$ for example. The relationship between this choice of $\mathcal{M}(A, B, C)$ and $\mathcal{M}(A, B, C) = \text{tr}(ABC)$ is difficult to see (although if $A, B, C$ were elements of $\mathbb{R}$, they would be equivalent). By applying $\mathcal{M}$ recursively repeatedly, we can compute the product of as many matrices as desired. That's a lot of power for $3r$ matrices that describe a bilinear map on a vector space.

To understand these algorithms better, it is helpful to look at the natural equivalence relation on $\{(A_i, B_i, C_i)\}_1^r$.

## 3.3 Matrix Multiplication Algorithm Equivalence

Given some rank $r$ matrix multiplication algorithm $\{(A_i, B_i, C_i)\}_1^r$, the space of equivalent algorithms is generated by the following transformations [15]:

- "scalings" $\lambda$ s.t. $\lambda\{(A_i, B_i, C_i)\}_1^r = \{(\alpha_i A_i, \beta_i B_i, \gamma_i C_i)\}_1^r$ for $\alpha_i\beta_i\gamma_i = 1$.
- "sandwiches" $s$ s.t. $s\{(A_i, B_i, C_i)\}_1^r = \{(PA_iQ^{-1}, QB_iR^{-1}, RC_iP^{-1})\}$ for $P, Q, R \in \mathrm{GL}(n, \mathbb{R})$.

But where do these come from?

### 3.3.1 Scalings

These transformations are evident for any algorithm since the outer product satisfies $\alpha_i A_i \otimes \beta_i B_i \otimes \frac{1}{\alpha_i\beta_i} C_i = A_i \otimes B_i \otimes C_i$. But note that $\lambda$ is parameterized by $\{(\alpha_i, \beta_i)\}_1^r \in (\mathbb{R}\setminus\{0\})^r \times (\mathbb{R}\setminus\{0\})^r$: For each summand one may use a different scaling (otherwise scalings would be a type of sandwich). Without this subtlety, checking matrix multiplication algorithm equivalence would be straightforward.

(The best complexity bounds we are aware of for real matrix multiplication algorithms satisfying certain mild assumptions include an $r!$ factor, though performance in practice is very efficient [4].)

### 3.3.2 Sandwiches

These transformations are more subtle, and come from the defining symmetries of the tensor.

First we just discuss these symmetries. Recall that a matrix $M$ is symmetric when $M^T = M$: This means that as a bilinear form $M(a, b) = a^T M b = a^T M^T b = b^T M a = M(b, a)$. For more general tensors symmetry is an increasingly restric-

tive condition, defined by the corresponding multilinear form being ambivalent to any permutation of the inputs. For bilinear maps, this already requires

$$M(a,b,c) = M(c,a,b) = M(b,c,a) = M(a,c,b) = M(b,a,c) = M(c,b,a).$$

The matrix multiplication tensor is not symmetric (since matrix multiplication does not commute). But it does have insidious symmetries, that come from the invariance of trace under cyclic permutations and transposition:

$$\mathrm{tr}(ABC) = \mathrm{tr}(CAB) = \mathrm{tr}(BCA) = \mathrm{tr}(A^T C^T B^T) = \mathrm{tr}(B^T A^T C^T) = \mathrm{tr}(C^T B^T A^T).$$

But then our algorithms must also satisfy these symmetries:

$$\sum \langle A_i, A\rangle\langle B_i, B\rangle\langle C_i, C\rangle = \sum \langle A_i, C\rangle\langle B_i, A\rangle\langle C_i, B\rangle = \sum \langle A_i, B\rangle\langle B_i, C\rangle\langle C_i, A\rangle$$
$$= \sum \langle A_i, A^T\rangle\langle B_i, C^T\rangle\langle C_i, B^T\rangle = \sum \langle A_i, B^T\rangle\langle B_i, A^T\rangle\langle C_i, C^T\rangle = \sum \langle A_i, C^T\rangle\langle B_i, B^T\rangle\langle C_i, A^T\rangle$$

Rearranging terms, we see that if $\{(A_i, B_i, C_i)\}_1^r$ is a matrix multiplication algorithm, then so must be some $\{(B_i, C_i, A_i)\}_1^r, \{(C_i, A_i, B_i)\}_1^r$ and similarly $\{(A_i^T, C_i^T, B_i^T)\}_1^r, \{(B_i^T, A_i^T, C_i^T)\}_1^r, \{(C_i^T, B_i^T, A_i^T)\}_1^r$. Then why not add these to our list of transformations? In fact these are sandwiches in disguise [4]. This is "obvious" from the algebraic perspective, in the same way that a matrix is similar to its transpose.

A loose way to think about it: We treat the rows of $A, B$ as dual to the columns of $B, C$ respectively. Since we defined our algorithm using the trace inner product, this means the rows of $A_i, B_i, C_i$ should be thought of as dual to the columns of $C_i, A_i, B_i$. Thus we see that if a change of basis $P \in \mathrm{GL}(n, \mathbb{R})$ acts on the rows of $A_i$ then $P^{-1}$ should act on the columns of $C_i$ and so forth.

### 3.3.3 Algorithm Orbit

Given some algorithm $\{(A_i, B_i, C_i)\}_1^r$ we call $\{(\alpha_i P A_i Q^{-1}, \beta_i Q B_i R^{-1}, \frac{1}{\alpha_i \beta_i} R C_i P^{-1})\}_1^r \subset (\mathbb{R}^{n\times n})^3$ the orbit of the algorithm. Weyl's inequality can't save us now. Although the rank tuples $\{(rk(A_i), rk(B_i), rk(C_i))\}_1^r$ must be fixed by this action, the closure of this set contains some odd things. For example, we can send $\beta_i$ to zero by increasing $\alpha_i$.

Numerical analysts might describe this as a preconditioning scheme: If we expect errors in $B$ are large, we could try to compensate by using reduced precision with $B$ and increased precision with $A$. The worst case behavior of such a strategy is bad, but if we were multiplying $(A, B) \in \mathbb{R}^{n\times n} \times \{I + \varepsilon M \mid \|M\|_2^2 = 1\}$ then such an algorithm might be reasonable.

Thus some points in the closure of this set are more naturally visualized in $(\mathbb{R}^{n\times n})^2$. This is just a glimpse at how exotic this orbit may be.

## 3.4 Approximate L2 Minimal Approximate Algorithms

Say we are given some real algorithm $\{(A_k, B_k, C_k)\}_1^r \subset \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$, found numerically up to some precision. We can then measure the quantity $\sum_1^r \|A_k\|_2 \|B_k\|_2 \|C_k\|_2$.

Call this quantity $S(A, B, C)$. Then $S(A, B, C)$ is similar to the tensor nuclear norm [14]. For matrices, this reduces to the usual nuclear norm, familiar in optimization circles. Further, $S(A, B, C)$ can be used to help bound the error of a given fast algorithm [9].

(Aside: The nuclear rank of a tensor is said to be the minimum $r$ for which there is a decomposition attaining the nuclear norm. This is hard to compute in general. For the multiplication of complex numbers, there is a decomposition of the corresponding tensor attaining both the rank and nuclear rank. For the matrix multiplication tensor the nuclear rank is $n^3$, attained by the standard algorithm[14, 10].)

Notice that $S(A, B, C)$ is not impacted by scalings $\{\alpha_i, \beta_i, \frac{1}{\alpha_i\beta_i}\}$, but that $\sum_1^r \|PA_iQ^{-1}\|_2\|QB_iR^{-1}\|_2\|RC_iP^{-1}\|_2$ typically varies for $P, Q, R \in \mathrm{GL}(n, \mathbb{R})$. Plugging in the svds $P = U_P D_P V_P^T$, $Q = U_Q D_Q V_Q^T$, and $R = U_R D_R V_R^T$ (and recalling the orthogonal invariance of the norm), one sees that this variation is due to $D_P, D_Q$ and $D_R$, and $V_P, V_Q$ and $V_R$.

Fortunately, with optimization this may be turned into an invariant of $\{(\alpha_i PA_kQ^{-1}, \beta_i QB_kR^{-1}, \frac{1}{\alpha_i\beta_i} RC_kP^{-1})\}_1^r$.

### 3.4.1 Norm-Based Invariants

Consider the quantity $f(P, Q, R) = \sum_1^r \|PA_iQ^{-1}\|_2\|QB_iR^{-1}\|_2\|RC_iP^{-1}\|_2 = \sum_1^r \|PA_iQ^{-1} \otimes QB_iR^{-1} \otimes RC_iP^{-1}\|_2$ (where $\otimes$ denotes the outer product).

Carrying around subscripts analogous to the linear case is painful, but clearly when algorithms $\mathcal{M} = \{(A_i, B_i, C_i)\}_1^r$ and $\mathcal{M}' = \{(A'_i, B'_i, C'_i)\}_1^r$ are equiva-

lent we have

$$\inf_{(P,Q,R)\in \mathrm{GL}(n,\mathbb{R})^3} \sum_1^r \|PA_iQ^{-1}\|_2 \|QB_iR^{-1}\|_2 \|RC_iP^{-1}\|_2 =$$

$$\inf_{(P,Q,R)\in \mathrm{GL}(n,\mathbb{R})^3} \sum_1^r \|PA'_iQ^{-1}\|_2 \|QB'_iR^{-1}\|_2 \|RC'_iP^{-1}\|_2 =$$

$$\inf_{(P,Q,R)\in \mathrm{GL}(n,\mathbb{R})^3} \sum_1^r \sqrt{\langle P^TPA_i, A_iQ^{-1}Q^{-T}\rangle \langle Q^TQB_i, B_iR^{-1}R^{-T}\rangle \langle R^TRA_i, A_iP^{-1}P^{-T}\rangle} =$$

$$\inf_{(E,F,G)>0} \sum_1^r \sqrt{\langle EA_i, A_iF^{-1}\rangle \langle FB_i, B_iG^{-1}\rangle \langle GA_i, A_iE^{-1}\rangle} =$$

$$\inf_{(E,F,G)>0} \sum_1^r \sqrt{\langle E^{\frac{1}{2}}A_iF^{-\frac{1}{2}}, E^{\frac{1}{2}}A_iF^{-\frac{1}{2}}\rangle \langle F^{\frac{1}{2}}B_iG^{-\frac{1}{2}}, F^{\frac{1}{2}}B_iG^{-\frac{1}{2}}\rangle \langle G^{\frac{1}{2}}A_iE^{-\frac{1}{2}}, G^{\frac{1}{2}}A_iE^{-\frac{1}{2}}\rangle} =$$

$$\inf_{(E,F,G)>0} \sum_1^r \|EA_iF^{-1}\|_2 \|FB_iG^{-1}\|_2 \|GC_iE^{-1}\|_2.$$

This quantity may also be approximated by convex optimization, see theorem 4 of the appendix. Thus, we might try to compute this quantity for two algorithms and compare it. What kind of tolerance to use is a challenging question (particularly given that our algorithms are already approximations).

Similar to the linear case, if the inf is attained and essentially unique then (normalized) optimal points differ only by orthogonal actions. In this situation one has strategies for checking algorithm equivalence when computing the inf is insufficient. For details see theorem 5 of the appendix.

Rank 7 algorithms for 2x2 matrix multiplication are all equivalent to Strassen [16]. Thus the inf is the same for any such algorithms. The optimization reflects this as well as we might hope, with typical computations varying by $\sim 10^{-13}$. For algorithms with different $r, n$ there is more variation. The last chapter of this thesis aims to give an experimental demonstration of this.

CHAPTER 4

# EXAMPLES OF L2 MINIMAL FORMS

These experiments concern (approximate) matrix multiplication algorithms gathered using machine learning techniques [13] implemented in C, via the Matlab parallel computing toolbox.

The infimum were approximated using trust regions methods in the Manopt toolbox.

### 4.0.1 Examples of L2 Minimal Forms: Standard Algorithm

As discussed earlier, the standard algorithm attains $\inf f(P,Q,R)$ and so is already in L2 minimal form. For example, the standard algorithm for $2 \times 2$ matrix multiplication is given by (using $e_{ij}$ to denote the $2 \times 2$ matrix with 1 in the $ij$th entry and 0 else):

$$
\begin{aligned}
(A_i, B_i, C_i)_{i=1} &= (e_{11}, e_{11}, e_{11}) \\
(A_i, B_i, C_i)_{i=2} &= (e_{12}, e_{21}, e_{11}) \\
(A_i, B_i, C_i)_{i=3} &= (e_{11}, e_{12}, e_{21}) \\
(A_i, B_i, C_i)_{i=4} &= (e_{12}, e_{22}, e_{21}) \\
(A_i, B_i, C_i)_{i=5} &= (e_{21}, e_{11}, e_{12}) \\
(A_i, B_i, C_i)_{i=6} &= (e_{22}, e_{21}, e_{12}) \\
(A_i, B_i, C_i)_{i=7} &= (e_{21}, e_{12}, e_{22}) \\
(A_i, B_i, C_i)_{i=8} &= (e_{22}, e_{22}, e_{22}).
\end{aligned}
$$

Clearly $\frac{S(A,B,C)}{n^3} = 1$. The optimization notices this and returns optimizers $\left[\begin{smallmatrix} x_1 & 0 \\ 0 & y_1 \end{smallmatrix}\right], \left[\begin{smallmatrix} x_2 & 0 \\ 0 & y_2 \end{smallmatrix}\right], \left[\begin{smallmatrix} x_3 & 0 \\ 0 & y_3 \end{smallmatrix}\right]$. These transformations just amount to scalings $\{(\alpha_i, \beta_i)\}_1^r$ by the ratios of $x_i, y_i$ (leaving the objective unchanged).

### 4.0.2 Examples of L2 Minimal Forms: Strop

The algorithm due to Strassen for multiplying $2 \times 2$ matrices is given as follows:

$$
\begin{aligned}
(A_i, B_i, C_i)_{i=1} &= \left(\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=2} &= \left(\left[\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 & 1 \\ 0 & -1 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=3} &= \left(\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 & 1 \\ 0 & -1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=4} &= \left(\left[\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} -1 & 0 \\ 1 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=5} &= \left(\left[\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} -1 & 0 \\ 1 & 0 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=6} &= \left(\left[\begin{smallmatrix} -1 & 0 \\ 1 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=7} &= \left(\left[\begin{smallmatrix} 0 & 1 \\ 0 & -1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]\right).
\end{aligned}
$$

Notice that $\frac{S(A,B,C)}{7} = \frac{1}{7}(6\sqrt{2}\sqrt{2} + \sqrt{2}\sqrt{2}\sqrt{2}) = \frac{\sqrt{2}\sqrt{2}}{7}(6 + \sqrt{2}) = \frac{12+2\sqrt{2}}{7}$. It turns out that this is not the best possible. Rounding for aesthetics, we see that optimizers $P, Q, R$ approximately satisfy $P^2 = Q^2 = R^2 = \begin{pmatrix} .632 & .316 \\ .316 & .632 \end{pmatrix}$ or $P^4 = Q^4 = R^4 = \begin{pmatrix} \frac{5}{10} & \frac{4}{10} \\ \frac{4}{10} & \frac{5}{10} \end{pmatrix}$.

Applying these optimizers, we obtain "Strop". Strop is, of course, equivalent to Strassen. However, the form looks quite different. There are many ways to

express this form (for example, the action of any orthogonal $P, Q$ and $R$ will not change the value of $S(A, B, C)$). The expression given here is appealing due to its symmetries and relative ease of representation: Up to ±1, only three numbers are required to yield a form consistent with the numerics. Letting $x = \frac{1}{3+\sqrt{3}}, y_1 = \frac{1}{\sqrt{3}} - \frac{1}{2}$, and $y_2 = \frac{1}{2\sqrt{3}} = \frac{y_1}{2-\sqrt{3}}$ one can write Strop as

$$
\begin{aligned}
(A_i, B_i, C_i)_{i=1} &= \left(\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=2} &= \left(\left[\begin{smallmatrix} x & x \\ 1-x & 1-x \end{smallmatrix}\right], \left[\begin{smallmatrix} 1+y_1 & -y_2 \\ y_2 & -y_1 \end{smallmatrix}\right], \left[\begin{smallmatrix} -x & 1-x \\ x & x-1 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=3} &= \left(\left[\begin{smallmatrix} 1+y_1 & -y_2 \\ y_2 & -y_1 \end{smallmatrix}\right], \left[\begin{smallmatrix} -x & 1-x \\ x & x-1 \end{smallmatrix}\right], \left[\begin{smallmatrix} x & x \\ 1-x & 1-x \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=4} &= \left(\left[\begin{smallmatrix} -y_1 & y_2 \\ -y_2 & 1+y_1 \end{smallmatrix}\right], \left[\begin{smallmatrix} x-1 & x \\ 1-x & -x \end{smallmatrix}\right], \left[\begin{smallmatrix} 1-x & 1-x \\ x & x \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=5} &= \left(\left[\begin{smallmatrix} 1-x & 1-x \\ x & x \end{smallmatrix}\right], \left[\begin{smallmatrix} -y_1 & y_2 \\ -y_2 & 1+y_1 \end{smallmatrix}\right], \left[\begin{smallmatrix} x-1 & x \\ 1-x & -x \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=6} &= \left(\left[\begin{smallmatrix} x-1 & x \\ 1-x & -x \end{smallmatrix}\right], \left[\begin{smallmatrix} 1-x & 1-x \\ x & x \end{smallmatrix}\right], \left[\begin{smallmatrix} -y_1 & y_2 \\ -y_2 & 1+y_1 \end{smallmatrix}\right]\right) \\
(A_i, B_i, C_i)_{i=7} &= \left(\left[\begin{smallmatrix} -x & 1-x \\ x & x-1 \end{smallmatrix}\right], \left[\begin{smallmatrix} x & x \\ 1-x & 1-x \end{smallmatrix}\right], \left[\begin{smallmatrix} 1+y_1 & -y_2 \\ y_2 & -y_1 \end{smallmatrix}\right]\right).
\end{aligned}
$$

Notice that this means Strop might be implemented using just $\{0, 1, 2, 3, 3^{\frac{1}{2}}\}$, along with the usual field operations. For those on a strict diet of rational numbers, one could use the approximations

$$
\begin{aligned}
x &= \frac{p_1 p_2 p_3}{10^{15}} \text{ for primes } p_1 = 3, p_2 = 43, p_3 = 1638177251203 \\
y_1 &= \frac{q_1 q_2 q_3 q_4}{10^{15}} \text{ for primes } q_1 = 2, q_2 = 19, q_3 = 49069, q_4 = 41483083 \\
y_2 &= \frac{r_1 r_2 r_3}{10^{15}} \text{ for primes } r_1 = 2063, r_2 = 11801, r_3 = 11857451.
\end{aligned}
$$

Up to a standard 16 digits of precision, these approximations are indistinguishable from the real $x, y_1$ and $y_2$.

Remark: A naive implementation of Strop using these $p_i, q_i, r_i$ would lead to

disastrous error, due to the limitations of floating point arithmetic. An example of these limitations for the uninitiated: In our first description of Strop, we noted that $y_2 = \frac{y_1}{2-\sqrt{3}}$. This could be seen as a simple consequence of the fact that (for positive numbers, including $x = 2$ and $y = 3$)

$$f(x,y) = \frac{x - y^{\frac{1}{x}}}{xy^{\frac{1}{x}}} - \frac{1}{y^{\frac{1}{x}}} + \frac{1}{x} = 0.$$

For typical values of $x, y$ this is easily verified with standard precision. However, for very large or small values one must cope with overflow or underflow. For example, attempting to plug in $x = \cos(\frac{\pi}{2})$ and $y = \sin(\frac{\pi}{2})$ yields 2 in Matlab. This is just due to Matlab's representation of $x$ (roughly, $6.123233 \cdot 10^{-17}$).

Lastly, for good measure we compute $\frac{S(A,B,C)}{7}$ for Strop. By algebra,

$$\begin{aligned}
\frac{S(A,B,C)}{7} &= \frac{1}{7}\left(\| \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] \|^3 + 6\| \left[\begin{smallmatrix} x & x \\ 1-x & 1-x \end{smallmatrix}\right] \|^2 \| \left[\begin{smallmatrix} 1+y_1 & -y_2 \\ y_2 & -y_1 \end{smallmatrix}\right] \|\right) \\
&= \frac{1}{7}\left(2^{\frac{3}{2}} + 6(2x^2 + 2(1-x)^2)\sqrt{1 + 2y_1 + 2y_1^2 + 2y_2^2}\right) \\
&= \frac{1}{7}\left(2^{\frac{3}{2}} + 2^3\frac{2}{\sqrt{3}}\right).
\end{aligned}$$

### 4.0.3 Error Predictions

This section gives evidence that the quantity $\sum_1^r \|A_i\|_2\|B_i\|_2\|C_i\|_2$ can be helpful for thinking about the error of some "typical algorithms".

Elaborating, we'll consider algorithms with $r = 7, n = 2^k$. The idea is to use Strop $\mathcal{M} = \{(A_i, B_i, C_i)\}_1^r$ as a reference algorithm to compare against. We apply Strop recursively (as naively as possible) to multiply matrices $A, B \in \mathbb{R}^{2^k \times 2^k}$ (with entries uniformly distributed in $(0,1)$) for increasing $k$.

Simultaneously, we consider other algorithms $\mathcal{M}' = \{(A_i', B_i', C_i')\}_1^r$ applied recursively to the same $A, B$. We plot the (log of) error $\|\mathcal{M}(A,B) - AB\|_2$, the (log of) error $\|\mathcal{M}'(A,B) - AB\|_2$, and (log of) estimations of $\|\mathcal{M}'(A,B) - AB\|_2$ given by

$$E(\mathcal{M}', A, B) = \|\mathcal{M}(A,B) - AB\|_2 \left( \frac{\sum_1^r \|A_i'\|\|B_i'\|\|C_i'\|}{\sum_1^r \|A_i\|\|B_i\|\|C_i\|} \right)^k.$$

Of note, we compare Strop against "nearby" algorithms $\mathcal{M}'$ generated by applying a single tuple of $2 \times 2$ matrices $(P, Q, R)$ (each with entries uniformly distributed in $(0, 1)$). This is easier on the eyes compared to using our machine learning generated algorithms, because ratios $\frac{\sum_1^r \|A_i'\|\|B_i'\|\|C_i'\|}{\sum_1^r \|A_i\|\|B_i\|\|C_i\|}$ are usually further from 1: For example, values $\frac{S(A,B,C)}{7}$ for one sample of $\approx$ 800 machine learning generated algorithms varied only from from $\approx 1.7$ to $\approx 3.5$. For one sample of 800 algorithms each obtained by applying a tuple $(P, Q, R)$ (entries uniformly distributed in $(0, 1)$) to Strop, we observe $\frac{S(A,B,C)}{7}$ varying from $\approx 1.9$ to $\approx 3.5 \cdot 10^4$.

In Figure 4.1 (left), we show this process for four $\mathcal{M}'$ that are "good performers." A sequence of $A, B \in \mathbb{R}^{2^k \times 2^k}$ is generated (with $k = 1, \ldots, 8$). Strop's (log) error on each $A, B$ is indicated by the solid line at the bottom. The estimation of an algorithm's (log) error on each $A, B$ is indicated by a dotted line, while its actual (log) error is indicated by a solid line of the same color.

Of course, there are also "bad performers." In Figure 4.1 (right), we compare Strop against ten such $\mathcal{M}'$. For each algorithm $\mathcal{M}'$, a sequence of $A, B \in \mathbb{R}^{2^k \times 2^k}$ is generated (with $k = 1, \ldots, 10$). The error of Strop on this sequence is computed, along with the predicted and true errors of $\mathcal{M}'$. Strop's (log) error on each such sequence is indicated by a red line. Estimations of algorithms' (log) errors are given by blue lines, while yellow lines indicate the actual (log) errors.

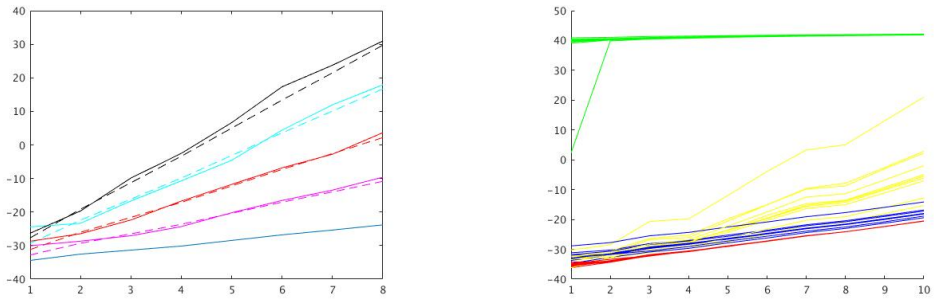

Figure 4.1: **Left**: Comparison of $\log \|\mathcal{M}'(A,B) - AB\|_2$ and $\log E(\mathcal{M}', A, B)$ (vertical) for four $\mathcal{M}'$, on $A, B \in \mathbb{R}^{2^k \times 2^k}$ for k=1 to 8 (horizontal). **Right**: Comparison of $\log \|\mathcal{M}'(A,B) - AB\|_2$ and $\log E(\mathcal{M}', A, B)$ (vertical) for ten $\mathcal{M}'$, on $A, B \in \mathbb{R}^{2^k \times 2^k}$ for k=1 to 10 (horizontal).

Worth noting: Strassen's algorithm would outperform such predictions due to sparsity. Still, Strassen can underperform Strop as $k$ grows (with regards to error, at the cost of time or memory). For example, Figure 4.2 plots the (log) errors of Strop and Strassen when used to multiply sample matrices $A, B \in \mathbb{R}^{2^{11} \times 2^{11}}$ and $A, B \in \mathbb{R}^{2^{12} \times 2^{12}}$.

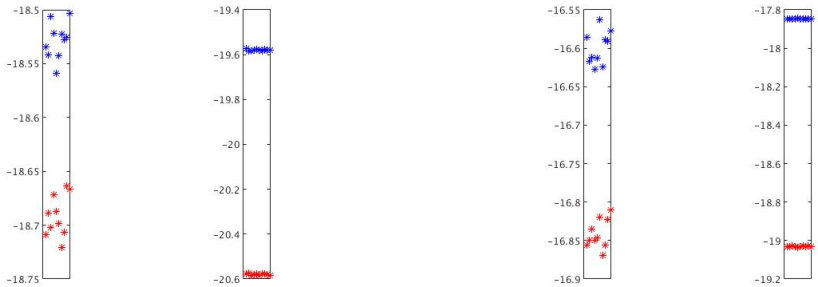

Figure 4.2: $\log \|\mathcal{M}(A,B) - AB\|_2$ for Strassen (blue) and Strop (red) on a sample of $A, B \in \mathbb{R}^{2^{11} \times 2^{11}}$ (left) and $A, B \in \mathbb{R}^{2^{12} \times 2^{12}}$ (right). Matrices $A, B$ were chosen with entries uniformly distributed in $(0, 1)$ (subleft) and $(-1, 1)$ (subright).

### 4.0.4 Examples of Infimal Variations

Just for curiosities sake, we include histograms of optimal $\frac{S(A,B,C)}{r}$ for algorithms with $(r,n) = (8,2)$ and $(r,n) = (23,3)$, generated via machine learning. In the first case we sampled approximately 1600 algorithms, and in the latter case the sample comprised approximately 2000 algorithms. As one can see from the histograms, these are not essentially unique [18]. The distribution of the training examples (which did not vary) should certainly be thought of as contributing to these images.

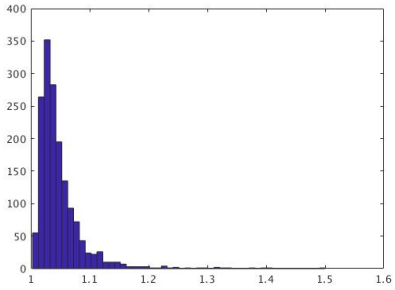


Figure 4.3: Optimal $\frac{S(A,B,C)}{8}$ when r=8,n=2.

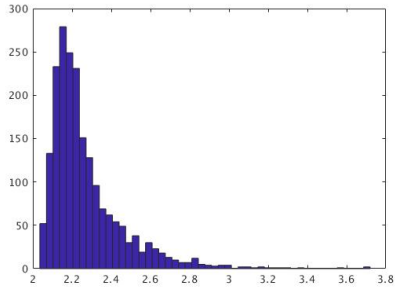


Figure 4.4: Optimal $\frac{S(A,B,C)}{23}$ for algorithms with r=23,n=3.

As one would expect, the histogram for algorithms found with $(r,n) = (7,2)$ looks quite degenerate (since all such algorithms are equivalent). Here we sampled about 800 algorithms.

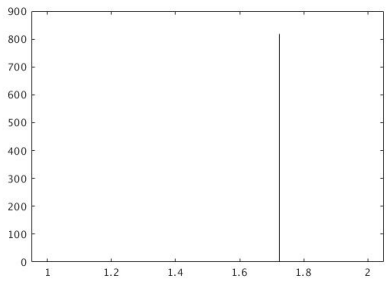


Figure 4.5: Optimal $\frac{S(A,B,C)}{7}$ when r=7,n=2.

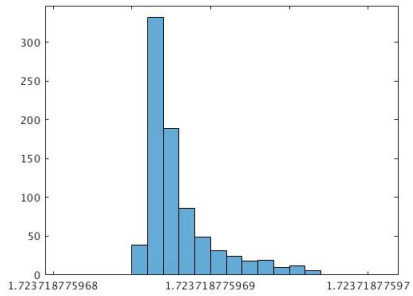


Figure 4.6: Optimal $\frac{S(A,B,C)}{7}$ when r=7, n=2 (zoomed in).

APPENDIX A

# APPENDIX

## A.1 Chapter 1 Proofs

Consider the function $\inf_{P\in \mathrm{GL}(n,\mathbb{R})} f_M(P) = \inf_{P\in \mathrm{GL}(n,\mathbb{R})} \sum_{i=1}^{r} \|PM_iP^{-1}\|_2^2$.

We denote the (possibly empty) set of minimizers as $\mathrm{opt}(f_M) \subseteq \mathrm{GL}(n,\mathbb{R})$. This can be formulated as an optimization problem over the cone of positive definite matrices $\{P \succ 0\}$ after noting that

$$f(P) = \sum_{i=1}^{r} \langle PM_iP^{-1}, PM_iP^{-1}\rangle = \sum_{i=1}^{r} \langle P^TPM_i, M_iP^{-1}P^{-T}\rangle \tag{A.1}$$

$$= \sum_{i=1}^{r} \langle (P^TP)^{\frac{1}{2}} M_i (P^TP)^{-\frac{1}{2}}, (P^TP)^{\frac{1}{2}} M_i (P^TP)^{-\frac{1}{2}}\rangle. \tag{A.2}$$

From this, one sees that for any $P$ there is some $P \succ 0$ that gives the same objective value. Hence it follows that

$$\inf_{P\in \mathrm{GL}(n),\mathbb{R}} \sum_{i=1}^{r} \|PM_iP^{-1}\|_2^2 = \inf_{P>0} \sum_{i=1}^{r} \|PM_iP^{-1}\|_2^2 = \inf_{P>0} \sum_{i=1}^{r} \langle PM_i, M_iP^{-1}\rangle$$

Denote $f_M^+(P) = \sum_{i=1}^{r} \langle PM_i, M_iP^{-1}\rangle$ and $\mathrm{opt}(f_M^+)$ the (possibly empty) minimizing subset of positive definite matrices. Note that $f_M^+(P) = f_M(P^{\frac{1}{2}})$. Further, if $P \in \mathrm{opt}(f_M^+)$, then $P^{\frac{1}{2}} \in \mathrm{opt}(f_M)$. If $P \in \mathrm{opt}(f_M)$, then $P^TP \in \mathrm{opt}(f_M^+)$. The positive definite cone is a much friendlier place than $\mathrm{GL}(n,\mathbb{R})$, certainly from an optimization perspective. A natural Riemannian metric on positive definite matrices is given by $\langle P_1, P_2\rangle_X = \mathrm{tr}(X^{-1}P_1X^{-1}P_2)$ [1].

Thus we obtain the following lemma.

**Theorem 1** *$f_M^+(P)$ is a convex function of $P$.*

Let $P_1, P_2 \succ 0$. The unique geodesic $[P_1, P_2]$ is parameterized for $t \in [0,1]$ by $\gamma(t) = P_1^{\frac{1}{2}}(P_1^{-\frac{1}{2}} P_2 P_1^{-\frac{1}{2}})^t P_1^{\frac{1}{2}}$. Thus we have that

$$\begin{aligned} f_M^+ \circ \gamma(t) &= \sum_{i=1}^r \langle P_1^{\frac{1}{2}}(P_1^{-\frac{1}{2}} P_2 P_1^{-\frac{1}{2}})^t P_1^{\frac{1}{2}} M_i, M_i P_1^{-\frac{1}{2}}(P_1^{\frac{1}{2}} P_2^{-1} P_1^{\frac{1}{2}})^t P_1^{-\frac{1}{2}} \rangle \\ &= \sum_{i=1}^r \langle (P_1^{-\frac{1}{2}} P_2 P_1^{-\frac{1}{2}})^t (P_1^{\frac{1}{2}} M_i P_1^{-\frac{1}{2}}), (P_1^{\frac{1}{2}} M_i P_1^{-\frac{1}{2}})(P_1^{-\frac{1}{2}} P_2 P_1^{-\frac{1}{2}})^t \rangle. \end{aligned}$$

To show the convexity of $f_M^+ \circ \gamma$, we show the summands are convex. Since $P_1^{-\frac{1}{2}} P_2 P_1^{-\frac{1}{2}} \succ 0$, it is enough to show the convexity of $f(t) = \langle X^t M, M X^t \rangle$ when $X \succ 0$ and $M$ arbitrary. In fact, the stronger statement holds that $f(t) = \log \langle X_1^t M, M X_2^t \rangle$ is convex for $X_1, X_2 \succ 0$ and $M$ arbitrary. By continuity, showing midpoint convexity suffices. Thus for $t, t' \in [0,1]$ we must show

$$\begin{aligned} &\log \langle X_1^{\frac{t+t'}{2}} M, M X_2^{\frac{t+t'}{2}} \rangle \leq \frac{1}{2} \log \langle X_1^t M, M X_2^t \rangle + \frac{1}{2} \log \langle X_1^{t'} M, M X_2^{t'} \rangle \\ \Leftrightarrow &\langle X_1^{\frac{t+t'}{2}} M, M X_2^{\frac{t+t'}{2}} \rangle \leq \langle X_1^t M, M X_2^t \rangle^{\frac{1}{2}} \langle X_1^{t'} M, M X_2^{t'} \rangle^{\frac{1}{2}}. \end{aligned}$$

Completing the proof, this is precisely Cauchy Schwarz:

$$\begin{aligned} &\langle X_1^{\frac{t+t'}{2}} M, M X_2^{\frac{t+t'}{2}} \rangle = \langle X_1^{\frac{t'}{2}} X_1^{\frac{t}{2}} M, M X_2^{\frac{t'}{2}} X_2^{\frac{t}{2}} \rangle = \langle X_1^{\frac{t}{2}} M X_2^{\frac{t}{2}}, X_1^{\frac{t'}{2}} M X_2^{\frac{t'}{2}} \rangle \\ \leq &\| X_1^{\frac{t}{2}} M X_2^{\frac{t}{2}} \|_F \, \| X_1^{\frac{t'}{2}} M X_2^{\frac{t'}{2}} \|_F = \langle X_1^{\frac{t}{2}} M X_2^{\frac{t}{2}}, X_1^{\frac{t}{2}} M X_2^{\frac{t}{2}} \rangle^{\frac{1}{2}} \langle X_1^{\frac{t'}{2}} M X_2^{\frac{t'}{2}}, X_1^{\frac{t'}{2}} M X_2^{\frac{t'}{2}} \rangle^{\frac{1}{2}} \\ = &\langle X_1^t M, M X_2^t \rangle^{\frac{1}{2}} \langle X_1^{t'} M, M X_2^{t'} \rangle^{\frac{1}{2}}. \end{aligned}$$

Of course, this tells us nothing about whether the inf is attained. Indeed, there are cases where it is not (though by Ekeland one will still find some approximation since this is a complete metric space [1, 12]). Note that even if the inf is attained, at best it is unique up to scale since $f_M^+(P) = f_M^+(\alpha P)$. Assuming this holds, then optimized forms of $\{M_i\}_1^r$ must differ by only scaling and orthogonal transformations: Consider two optimizers $P_1, P_2 \in \text{opt}(f_M)$.

Then $\alpha P_1^T P_1 = (\sqrt{\alpha} P_1)^T (\sqrt{\alpha} P_1) = P_2^T P_2$. By the uniqueness of the Gram matrix [17], $P_2 = \sqrt{\alpha} Q P_1$ for some $Q$ orthogonal. Thus all optimal forms amount to $\{\alpha Q P_1 M_i P_1^{-1} Q^T\}_1^r$.

Last, we show that this is the situation when $M_i$ have no common invariant subspaces ($V$ such that all $M_i V \subseteq V$).

**Theorem 2** *If $M_i$ have no common invariant subspaces, then* $\inf f_M^+(P)$ *is attained.*

The essential idea is that when $M_i$ have no common invariant subspaces, there can be no zero "depths" of $\{M_i\}$. Thus, it is impossible to send $P$ to a singular matrix while keeping the objective finite.

Formally, recall that $f_M^+(P) = \sum_{k=1}^r \|P^{\frac{1}{2}} M_k P^{-\frac{1}{2}}\|_F^2$. Taking the spectral decomposition gives $P^{\frac{1}{2}} = UDU^T$ for $U$ orthogonal and $D$ diagonal (with positive entries). So we have that $f_M^+(P) = \sum_{k=1}^r \|UDU^T M_k U D^{-1} U^T\|_F^2 = \sum_{k=1}^r \|DU^T M_k U D^{-1}\|_F^2$. Denote the columns of $U$ by $u_1, \ldots, u_n$ and the nonzero entries of $D$ by $\sigma_1, \ldots, \sigma_n$. Then $(DU^T M_k U D^{-1})_{ij} = \frac{\sigma_i}{\sigma_j} u_i^T M_k u_j$.

Let $P_\ell$ be some sequence tending to the inf, with $U_\ell, D_\ell$ decomposing $P_\ell^{\frac{1}{2}}$ as above. Since $f_M^+$ is scale invariant, WLOG assume that each $P_\ell$ has unit norm. Assume that the inf is not attained and so we must have $P_\ell$ (and thus $D_\ell$) tending to a singular matrix. Partition $\{1, \ldots, n\}$ into $S \sqcup C$ as follows: $i \in S$ if $(\sigma_i)_\ell \to 0$ and $i \in C$ if $(\sigma_i)_\ell$ is bounded in some interval. Since $\|P_\ell\| = 1$, we cannot have $(\sigma_i)_\ell \to \infty$ and we must have $|C| \geq 1$. By assumption, $|S| \geq 1$. Let $i \in C$ and $j \in S$. Then $\left(\frac{\sigma_i}{\sigma_j}\right)_\ell \to \infty$ and, since we assumed the inf is not attained, $(u_i^T M_k u_j)_\ell \to 0$ for all $k$. By compactness of $\mathrm{O}(n)$, there exist subspaces $V, V^\perp$ of $\mathbb{R}^n$ so that $\dim V = |C|$, $\dim V^\perp = |S|$ and $u_i \in V$, $u_j \in V^\perp \implies u_i^T M_k u_j = 0$. Thus $V \perp M_k V^\perp$ shows $M_k V^\perp \subseteq V^\perp$ for all $k$ as needed.

Similarly, keeping "depths" nonzero ensure minimizers are unique up to scale.

**Theorem 3** *If $M_i$ have no common invariant subspaces, then minimizers of $f_M^+(P)$ are unique up to scale.*

Let $P_1, P_2$ be minimizers of $f_M^+(P)$. Then by convexity $P_1^{\frac{1}{2}}(P_1^{-\frac{1}{2}}P_2P_1^{-\frac{1}{2}})^tP_1^{\frac{1}{2}}$ is a minimizer for all $t \in [0,1]$. Applying spectral decomposition, there are orthogonal $U$ and diagonal $D$ such that $P_1^{-\frac{1}{2}}P_2P_1^{-\frac{1}{2}} = UD^2U^T$. Thus the following is a constant function of $t$:

$$\begin{aligned}
&\sum_{k=1}^{r}\langle P_1^{\frac{1}{2}}(P_1^{-\frac{1}{2}}P_2P_1^{-\frac{1}{2}})^tP_1^{\frac{1}{2}}M_k, M_kP_1^{-\frac{1}{2}}(P_1^{\frac{1}{2}}P_2^{-1}P_1^{\frac{1}{2}})^tP_1^{-\frac{1}{2}}\rangle \\
&= \sum_{k=1}^{r}\langle (UD^2U^T)^tP_1^{\frac{1}{2}}M_kP_1^{-\frac{1}{2}}, P_1^{\frac{1}{2}}M_kP_1^{-\frac{1}{2}}(UD^2U^T)^{-t}\rangle \\
&= \sum_{k=1}^{r}\langle D^{2t}(U^TP_1^{\frac{1}{2}}M_kP_1^{-\frac{1}{2}}U), (U^TP_1^{\frac{1}{2}}M_kP_1^{-\frac{1}{2}}U)D^{-2t}\rangle
\end{aligned}$$

Call $(U^TP_1^{\frac{1}{2}}M_kP_1^{-\frac{1}{2}}U) = N_k$. Then we have that $\sum_{k=1}^{r}\|D^tN_kD^{-t}\|_F^2$ is a constant function of $t$. Denote the eigenvalues of $D$ by $\sigma_1, \ldots, \sigma_n$, and $n_{ij} = \sum_k (N_k)_{i,j}^2$, then equivalently we have $\sum_{i<j} n_{ij}\left(\frac{\sigma_i}{\sigma_j}\right)^t + n_{ji}\left(\frac{\sigma_j}{\sigma_i}\right)^t$ is a constant function. Taking the second derivative, $\sum_{i<j}\log\left(\frac{\sigma_i}{\sigma_j}\right)^2\left[n_{ij}\left(\frac{\sigma_i}{\sigma_j}\right)^t + n_{ji}\left(\frac{\sigma_j}{\sigma_i}\right)^t\right] = 0$. Hence for any $i \neq j$ either $\sigma_i = \sigma_j$ or $n_{ij} = n_{ji} = 0$.

Since $M_k, M_k'$ have no common invariant subspaces, neither do $N_k, N_k'$. Hence for $i \neq j$, $\sigma_i = \sigma_j$ and so $D = \alpha I$ for some $\alpha \in \mathbb{R}_+$. Hence, $P_1^{-\frac{1}{2}}P_2P_1^{-\frac{1}{2}} = \alpha^2 I$ shows $P_1$ and $P_2$ are the same up to scale.

## A.2 Chapter 2 Proofs

Given some real algorithm $\mathcal{M} = \{(A_i, B_i, C_i)\}$, consider the problem

$$\inf_{(P,Q,R)\in \mathrm{GL}(n,\mathbb{R})^3} \sum_1^r \|PA_iQ^{-1}\|_2 \|QB_iR^{-1}\|_2 \|RC_iP^{-1}\|_2$$
$$= \inf_{(P,Q,R)>0} \sum_1^r \|PA_iQ^{-1}\|_2 \|QB_iR^{-1}\|_2 \|RC_iP^{-1}\|_2.$$

We equip $\{M \succ 0\}^3$ with a Riemannian metric (the product metric [20]) given by

$$\langle (P_1, Q_1, R_1), (P_2, Q_2, R_2)\rangle_{(X,Y,Z)} = \mathrm{tr}(X^{-1}P_1X^{-1}P_2) + \mathrm{tr}(Y^{-1}Q_1Y^{-1}Q_2) + \mathrm{tr}(Z^{-1}R_1Z^{-1}R_2).$$

Geodesics $[(P_1, Q_1, R_1), (P_2, Q_2, R_2)]$ may be parameterized for $t \in [0, 1]$ by

$$\gamma(t) = \left(P_1^{\frac{1}{2}}(P_1^{-\frac{1}{2}}P_2P_1^{-\frac{1}{2}})^tP_1^{\frac{1}{2}}\ ,\ Q_1^{\frac{1}{2}}(Q_1^{-\frac{1}{2}}Q_2Q_1^{-\frac{1}{2}})^tQ_1^{\frac{1}{2}}\ ,\ R_1^{\frac{1}{2}}(R_1^{-\frac{1}{2}}R_2R_1^{-\frac{1}{2}})^tR_1^{\frac{1}{2}}\right).$$

**Theorem 4** *$f_{\mathcal{M}}(P, Q, R)$ is convex.*

Since $(M_1^{-\frac{1}{2}}M_2M_1^{-\frac{1}{2}})^t$ is positive definite when $M_1$ and $M_2$ are, it is sufficient to show that for any $(P_1, Q_1, R_1), (P_2, Q_2, R_2)$ all positive definite,

$$f\left(P_1^{\frac{1}{2}}P_2^tP_1^{\frac{1}{2}}\ ,\ Q_1^{\frac{1}{2}}Q_2^tQ_1^{\frac{1}{2}}\ ,\ R_1^{\frac{1}{2}}R_2^tR_1^{\frac{1}{2}}\right) \text{ is convex in } t.$$

Further, since the sum of convex functions is convex it is enough to show the convexity of

$$f(t) = \sqrt{\langle P_1^{\frac{1}{2}}P_2^tP_1^{\frac{1}{2}}A_i, A_iQ_1^{-\frac{1}{2}}Q_2^{-t}Q_1^{-\frac{1}{2}}\rangle\langle Q_1^{\frac{1}{2}}Q_2^tQ_1^{\frac{1}{2}}B_i, B_iR_1^{-\frac{1}{2}}R_2^{-t}R_1^{-\frac{1}{2}}\rangle\langle R_1^{\frac{1}{2}}R_2^tR_1^{\frac{1}{2}}C_i, C_iP_1^{-\frac{1}{2}}P_2^{-t}P_1^{-\frac{1}{2}}\rangle}.$$

Since $P_i, Q_i, R_i$ symmetric we may rewrite this as

$$\sqrt{\langle P_2^t(P_1^{\frac{1}{2}}A_iQ_1^{-\frac{1}{2}}), (P_1^{\frac{1}{2}}A_iQ_1^{-\frac{1}{2}})Q_2^{-t}\rangle\langle Q_2^t(Q_1^{\frac{1}{2}}B_iR_1^{-\frac{1}{2}}), (Q_1^{\frac{1}{2}}B_iR_1^{-\frac{1}{2}})R_2^{-t}\rangle\langle R_2^t(R_1^{\frac{1}{2}}C_iP_1^{-\frac{1}{2}}), (R_1^{\frac{1}{2}}C_iP_1^{-\frac{1}{2}})P_2^{-t}\rangle}.$$

Thus it would also be enough to show that for arbitrary $M_1, M_2, M_3$ and $P, Q, R > 0$ the function $f(t) = \sqrt{\langle P^t M_1, M_1 Q^{-t}\rangle \langle Q^t M_2, M_2 R^{-t}\rangle \langle R^t M_3, M_3 P^{-t}\rangle}$ is convex. In fact, the stronger statement that $\log \circ f$ is convex holds: This follows by theorem 1 from Chapter 1 since

$$\log(f(t)) = \frac{1}{2}\left(\log\langle P^t M_1, M_1 Q^{-t}\rangle + \log\langle Q^t M_2, M_2 R^{-t}\rangle + \log\langle R^t M_3, M_3 P^{-t}\rangle\right).$$

Again by Ekeland, one can always approximate this quantity. Now we show that when the inf is attained and essentially unique, optimized and normalized algorithms differ only by orthogonal transformations.

**Theorem 5** *Let* $\mathcal{M} = \{(\alpha_i P A_i Q^{-1}, \beta_i Q B_i R^{-1}, \frac{1}{\alpha_i\beta_i} R C_i P^{-1})\}_1^r$. *Assume* $f_{\mathcal{M}} = \inf_{(P,Q,R)\in GL(n,\mathbb{R})} \sum_1^r \|PA_iQ^{-1}\|_2 \|QB_iR^{-1}\|_2 \|RC_iP^{-1}\|_2$ *is attained and that optimizers* $(P_x^*, Q_x^*, R_x^*), (P_y^*, Q_y^*, R_y^*)$ *satisfy* $(P_x^*, Q_x^*, R_x^*) = (\alpha U P_y^*, \beta V Q_y^*, \gamma W R_y^*)$ *for* $\alpha, \beta, \gamma \neq 0$ *and* $U, V, W$ *orthogonal. Then optimal normalized algorithms in* $\mathcal{M}$ *differ only by orthogonal transformations.*

By assumption, there are $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ such that

$$\inf_{(P,Q,R)\in \mathrm{GL}(n,\mathbb{R})^3} \sum_1^r \|PA_iQ^{-1}\|_2 \|QB_iR^{-1}\|_2 \|RC_iP^{-1}\|_2 =$$
$$\sum_1^r \|\mathcal{P}A_i\mathcal{Q}^{-1}\|_2 \|\mathcal{Q}B_i\mathcal{R}^{-1}\|_2 \|\mathcal{R}C_i\mathcal{P}^{-1}\|_2.$$

Consider any algorithms $\mathcal{M}_1 = \{(\alpha_i P_1 A_i Q_1^{-1}, \beta_i Q_1 B_i R_1^{-1}, \frac{1}{\alpha_i\beta_i} R_1 C_i P_1^{-1})\}_1^r$ and $\mathcal{M}_2 = \{(\gamma_i P_2 A_i Q_2^{-1}, \delta_i Q_2 B_i R_2^{-1}, \frac{1}{\gamma_i\delta_i} R_2 C_i P_2^{-1})\}_1^r$.

Let $(P_1^*, Q_1^*, R_1^*)$ attain the inf for $\mathcal{M}_1$. Let $\mathcal{M}_1'$ be the result of applying $(P_1^*, Q_1^*, R_1^*)$ to $\mathcal{M}_1$. Define $(P_0^*, Q_0^*, R_0^*) = (\mathcal{P}P_1^{-1}, \mathcal{Q}Q_1^{-1}, \mathcal{R}R_1^{-1})$. Then $(P_0^*, Q_0^*, R_0^*)$ also

attains the inf, so by assumption

$$\mathcal{M}_1' = \{(\alpha_i \frac{\alpha}{\beta} U(\mathcal{P}A_i\mathcal{Q}^{-1})V^T, \beta_i \frac{\beta}{\gamma} V(\mathcal{Q}B_i\mathcal{R}^{-1})W^T, \frac{1}{\alpha_i\beta_i}\frac{\gamma}{\alpha} W(\mathcal{R}C_i\mathcal{P}^{-1})U^T)\}_1^r$$
$$= \{(\alpha_i' U(\mathcal{P}A_i\mathcal{Q}^{-1})V^T, \beta_i' V(\mathcal{Q}B_i\mathcal{R}^{-1})W^T, \frac{1}{\alpha_i'\beta_i'} W(\mathcal{R}C_i\mathcal{P}^{-1})U^T)\}_1^r.$$

Let $(P_2^*, Q_2^*, R_2^*)$ attain the inf for $\mathcal{M}_2$. Let $\mathcal{M}_2'$ be the result of applying $(P_2^*, Q_2^*, R_2^*)$ to $\mathcal{M}_2$. Since $(P_3^*, Q_3^*, R_3^*) = (\mathcal{P}P_2^{-1}, \mathcal{Q}Q_2^{-1}, \mathcal{R}R_2^{-1})$ also attains the inf, by assumption

$$\mathcal{M}_2' = \{(\gamma_i \frac{\alpha}{\beta} U(\mathcal{P}A_i\mathcal{Q}^{-1})V^T, \delta_i \frac{\beta}{\gamma} V(\mathcal{Q}B_i\mathcal{R}^{-1})W^T, \frac{1}{\gamma_i\delta_i}\frac{\gamma}{\alpha} W(\mathcal{R}C_i\mathcal{P}^{-1})U^T)\}_1^r =$$
$$= \{(\gamma_i' U(\mathcal{P}A_i\mathcal{Q}^{-1})V^T, \delta_i' V(\mathcal{Q}B_i\mathcal{R}^{-1})W^T, \frac{1}{\gamma_i'\delta_i'} W(\mathcal{R}C_i\mathcal{P}^{-1})U^T\}_1^r.$$

Thus we see that by applying some normalization scheme to $\mathcal{M}_1'$ and $\mathcal{M}_2'$ (e.g. setting $\|A_i\|_2 = \|B_i\|_2 = \|C_i\|_2$) we obtain algorithms that differ only by $U, V, W$ orthogonal and scalings $\{(\alpha_i, \beta_i)\}_1^r = \{(\pm 1, \pm 1)\}_1^r$. This completes the proof.

The circumstances of Theorem 5 would be fortuitous: Determining whether algorithms $\mathcal{M}_1', \mathcal{M}_2'$ differ by orthogonal sandwiches and scalings seems simpler than the original problem. For example, one can compare the svds of the relevant $A_i, B_i, C_i$.

## BIBLIOGRAPHY


[1] Miroslav Bacák. *Convex analysis and optimization in Hadamard spaces*. de Gruyter, 2014.

[2] Grey Ballard, Austin R Benson, Alex Druinsky, Benjamin Lipshitz, and Oded Schwartz. Improving the numerical stability of fast matrix multiplication. *SIAM Journal on Matrix Analysis and Applications*, 37(4):1382–1418, 2016.

[3] Austin R Benson and Grey Ballard. A framework for practical parallel fast matrix multiplication. *ACM SIGPLAN Notices*, 50(8):42–53, 2015.

[4] Guillaume O Berger, P-A Absil, Lieven De Lathauwer, Raphaël M Jungers, and Marc Van Barel. Equivalent polyadic decompositions of matrix multiplication tensors. *arXiv preprint arXiv:1902.03950*, 2019.

[5] Rajendra Bhatia. *Positive definite matrices*. Princeton university press, 2009.

[6] Dario Bini and Grazia Lotti. Stability of fast algorithms for matrix multiplication. *Numerische Mathematik*, 36(1):63–72, 1980.

[7] Nicolas Boumal. An introduction to optimization on smooth manifolds. Available online, Nov 2020.

[8] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 2013.

[9] James Demmel, Ioana Dumitriu, Olga Holtz, and Robert Kleinberg. Fast matrix multiplication is stable. *Numerische Mathematik*, 106(2):199–224, 2007.

[10] Harm Derksen. On the nuclear norm and the singular value decomposition of tensors. *Foundations of Computational Mathematics*, 16(3):779–811, 2016.

[11] Harm Derksen and Visu Makam. Algorithms for orbit closure separation for invariants and semi-invariants of matrices. *Algebra & Number Theory*, 14(10):2791–2813, 2020.

[12] Ivar Ekeland. On the variational principle. *Journal of Mathematical Analysis and Applications*, 47(2):324–353, 1974.

[13] Veit Elser. A network that learns strassen multiplication. *The Journal of Machine Learning Research*, 17(1):3964–3976, 2016.

[14] Shmuel Friedland and Lek-Heng Lim. Nuclear norm of higher-order tensors. *Mathematics of Computation*, 87(311):1255–1281, 2018.

[15] H. De Groote. On varieties of optimal algorithms for the computation of bilinear mappings i. the isotropy group of a bilinear mapping. *Theoretical Computer Science*, 7:1–24, 1978.

[16] H. De Groote. On varieties of optimal algorithms for the computation of bilinear mappings. ii. optimal algorithms for 2x2-matrix multiplication. *Theoretical Computer Science*, 7:127–148, 1978.

[17] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[18] Rodney W Johnson and Aileen M McLoughlin. Noncommutative bilinear algorithms for 3*3 matrix multiplication. *SIAM Journal on Computing*, 15(2):595–603, 1986.

[19] Joseph M Landsberg. *Geometry and complexity theory*, volume 169. Cambridge University Press, 2017.

[20] John M. Lee. *Introduction to Riemannian manifolds*, volume 176 of *Graduate Texts in Mathematics*. Springer, 2018.

[21] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[22] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[23] Anh-Huy Phan, Petr Tichavskỳ, and Andrzej Cichocki. Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations. *IEEE Transactions on Signal Processing*, 61(19):4834–4846, 2013.

[24] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.