Fast (and yet correct!) Tensor Processing

Presented by: Ana Sá Oliveira, Edgar Araújo, Gabriel Paiva

Supervised by: José Nuno Oliveira

University of Minho, Informatics Department

March 2026

Overview

Context and Motivation: Tensor Processing

Tensor Processing

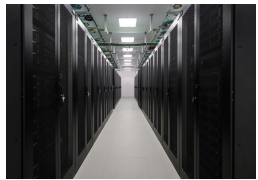
Tensor processing is an advanced form of **Basic Linear Algebra** (**BLAS**).

- Foundation of numeric programming
- Used in computational fluid dynamics and large-scale simulations (tsunamis, epidemics)
- Foundation of **Machine Learning**, in particular, Large Language (Transformer) Models (**LLMs**)
- Important for Probabilistic Programming

Problem: Untyped Linear Algebra

Prioritizing Performance over Safety

- Large-scale tensor processing relies on **HPC** and massively parallel architectures, prioritizing **performance** over safety.
- **Manual coding** predominates to ensure optimal performance.
- **Automated tools aren't trusted** to deliver both performance and correctness simultaneously.



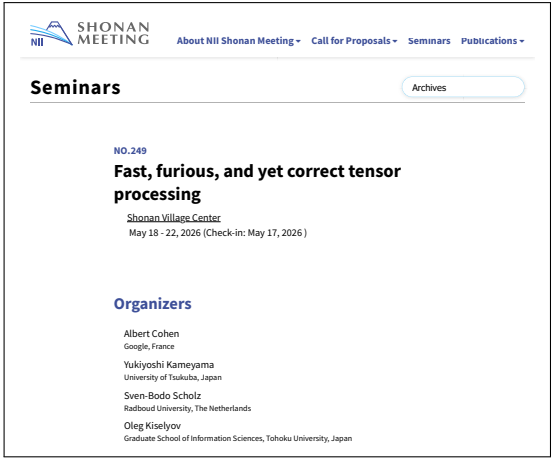
Untyped Linear Algebra

- Frameworks (e.g., PyTorch, TensorFlow) **lack static dimensional type checking**.
- *Shape mismatch* errors happen at runtime → **Wasted time and money**.

TensorFlow

Current Approaches

- Rank-polymorphism
- Tensor comprehensions & transformations
- **Biproducts**
- APL array programming patterns
- Exterior algebra
- Shape calculi / size types



Meeting Goal: Bring isolated communities together to stimulate convergence.

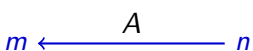
Proposed Solution: Category Theory

The Category of Matrices ($\mathbf{Mat}_K$)

In the category $\mathbf{Mat}_K$:

- **Objects:** Natural numbers ($n, m \in \mathbb{N}$) representing dimensions.
- **Morphisms:** A morphism A from n to m is an $m \times n$ matrix.

The Arrow (Categorical)



The Matrix (Concrete)

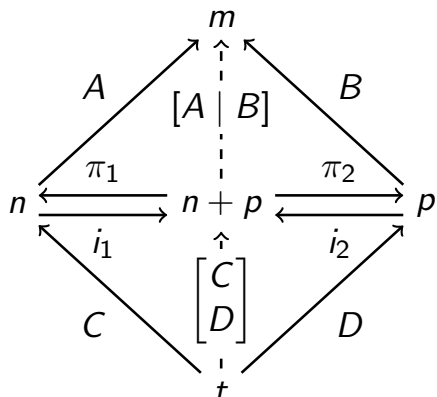
$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

Biproducts: Formalizing Vectorization & Parallelism

We rely on the categorical notion of a **Biproduct** to provide a strict foundation for blocked linear algebra.

This structure allows us to formally specify:

- **Vectorization:** Packing data horizontally for optimized operations.
- **Parallelism:** Splitting computations in blocks to distribute workloads.

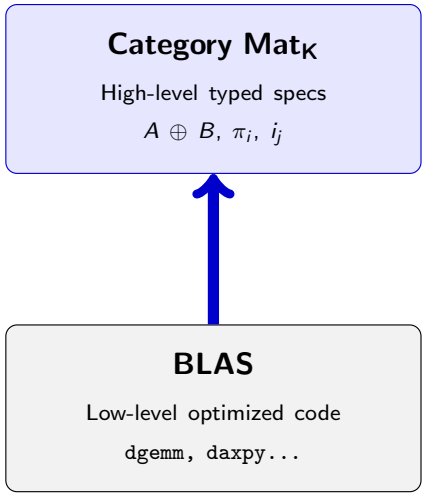


Methodology: Reverse Engineering BLAS

Reverse Engineering

via Categorical & Biproduct Laws

Statically verified type checking



Case Study: AXPY

The AXPY Algorithm

The name **AXPY** stands for the fundamental Linear Algebra operation: $y = a \cdot x + y$.

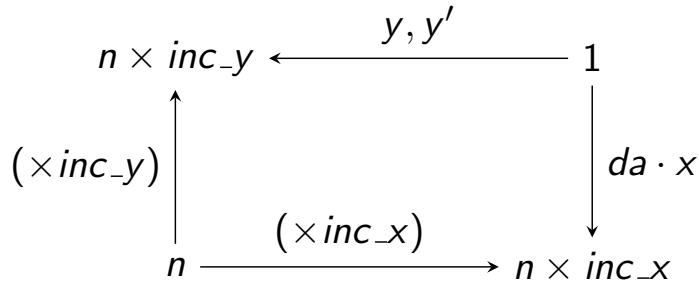
Pseudo-code

$y'[i \times inc\_y] = y[i \times inc\_y] + da \cdot x[i \times inc\_x]$

Visual Example (Vectors):

$$\begin{bmatrix}1\\2\end{bmatrix} + 3 \cdot \begin{bmatrix}4\\5\\6\\3\end{bmatrix}$$ (considering strided access to x)

Type Diagram (Pointwise)



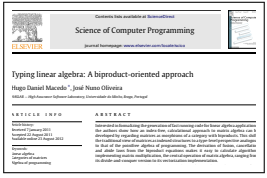
- **Vertical/Horizontal arrows:** Represent index scaling by strides.
- **Top/Right arrows:** Represent the data access and operations (y and x).

Literature Review

Literature Review: Foundational Papers

Our review bridges formal theory and industry practice, drawing from foundational papers on biproduct-oriented linear algebra and state-of-the-art **HPC** technical literature:

Typing linear algebra: A biproduct-oriented approach


Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Typing linear algebra: A biproduct-oriented approach

Hugo Daniel Macedo *, José Nuno Oliveira

HASLAB – High Assurance Software Laboratory, Universidade do Minho, Braga, Portugal

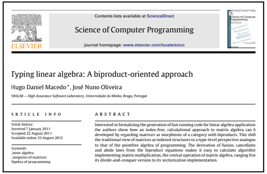
ARTICLE INFO

ABSTRACT

Literature Review: Foundational Papers

Our review bridges formal theory and industry practice, drawing from foundational papers on biproduct-oriented linear algebra and state-of-the-art **HPC** technical literature:

Typing linear algebra: A biproduct-oriented approach


Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Typing linear algebra: A biproduct-oriented approach

Hugo Daniel Macedo*, José Nuno Oliveira

HASLAB — High Assurance Software Laboratory, Universidade do Minho, Braga, Portugal

ARTICLE INFO

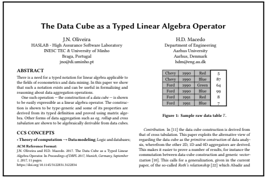
Article history:
Received 7 January 2011
Accepted 22 August 2011
Available online 23 August 2012

Keywords:
Linear algebra
Categories of matrices
Algebra of programming

ABSTRACT

Interested in formalizing the generation of fast running code for linear algebra application the authors show how an index-free, calculational approach to matrix algebra can b developed by regarding matrices as morphisms of a category with biproducts. This shif the traditional view of matrices as indexed structures to a type-level perspective analogou to that of the pointfree algebra of programming. The derivation of fusion, cancellatio and abide laws from the biproduct equations makes it easy to calculate algorithm implementing matrix multiplication, the central operation of matrix algebra, ranging fro its divide-and-conquer version to its vectorization implementation.

The data cube as a typed linear algebra operator


The Data Cube as a Typed Linear Algebra Operator

J.N. Oliveira
HASLAB - High Assurance Software Laboratory
INESC TEC & University of Minho
Braga, Portugal
jno@di.uminho.pt

H.D. Macedo
Department of Engineering
Aarhus University
Aarhus, Denmark
hdm@eng.au.dk

ABSTRACT

There is a need for a typed notation for linear algebra applicable to the fields of econometrics and data mining. In this paper we show that such a notation exists and can be useful in formalizing and reasoning about data aggregation operations.

One such operation — the construction of a *data cube* — is shown to be easily expressible as a linear algebra operator. The construction is shown to be type-generic and some of its properties are derived from its typed definition and proved using matrix algebra. Other forms of data aggregation such as eg. *rollup* and *cross tabulation* are shown to be algebraically derivable from data cubes.

CCS CONCEPTS

• **Theory of computation → Data modeling**; Logic and databases;

ACM Reference Format:
J.N. Oliveira and H.D. Macedo. 2017. The Data Cube as a Typed Linear Algebra Operator. In *Proceedings of DBPL 2017, Munich, Germany, September 1, 2017*, 11 pages.
https://doi.org/10.1145/3122831.3122834

Chevy	1990	Red	5
Chevy	1990	Blue	87
Ford	1990	Green	64
Ford	1990	Blue	99
Ford	1991	Red	8
Ford	1991	Blue	7

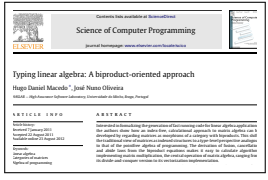
Figure 1: Sample raw data table T**.**

Contribution. In [11] the data cube construction is derived from that of cross tabulation. This paper exploits the alternative view of regarding the data cube as the *primitive construction* of data analysis, wherefrom the other 2D, 1D and 0D aggregators are derived. This makes it easier to prove a number of results, for instance the commutation between data-cube construction and generic *vectorization* [10]. This calls for a generalization, given in the current paper, of the so-called *Roth's relationship* [22] which Abadir and

Literature Review: Foundational Papers

Our review bridges formal theory and industry practice, drawing from foundational papers on biproduct-oriented linear algebra and state-of-the-art **HPC** technical literature:

Typing linear algebra: A biproduct-oriented approach


Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Typing linear algebra: A biproduct-oriented approach

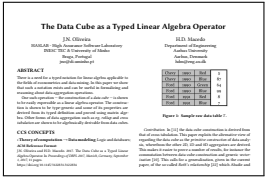
Hugo Daniel Macedo *, José Nuno Oliveira

HASLAB – High Assurance Software Laboratory, Universidade do Minho, Braga, Portugal

ARTICLE INFO

ABSTRACT

The data cube as a typed linear algebra operator


The Data Cube as a Typed Linear Algebra Operator

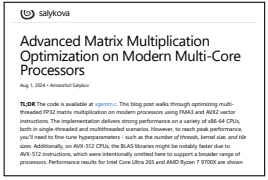
J.N. Oliveira
HASLAB - High Assurance Software Laboratory
INESC TEC & University of Minho
Braga, Portugal
jno@di.uminho.pt

H.D. Macedo
Department of Engineering
Aarhus University
Aarhus, Denmark
hdm@eng.au.dk

ABSTRACT

CCS CONCEPTS

Advanced Matrix Multiplication Optimization


salykova

Advanced Matrix Multiplication Optimization on Modern Multi-Core Processors

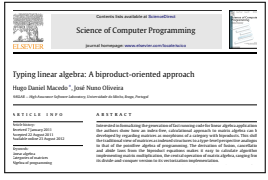
Aug 1, 2024 • Amanzhol Salykov

TL;DR The code is available at sgemm.c. This blog post walks through optimizing multi-threaded FP32 matrix multiplication on modern processors using FMA3 and AVX2 vector instructions. The implementation delivers strong performance on a variety of x86-64 CPUs, both in single-threaded and multithreaded scenarios. However, to reach peak performance, you'll need to fine-tune hyperparameters - such as the *number of threads, kernel size, and tile sizes*. Additionally, on AVX-512 CPUs, the BLAS libraries might be notably faster due to AVX-512 instructions, which were intentionally omitted here to support a broader range of processors. Performance results for Intel Core Ultra 265 and AMD Ryzen 7 9700X are shown

Literature Review: Foundational Papers

Our review bridges formal theory and industry practice, drawing from foundational papers on biproduct-oriented linear algebra and state-of-the-art **HPC** technical literature:

Typing linear algebra: A biproduct-oriented approach


Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Typing linear algebra: A biproduct-oriented approach

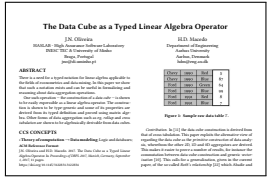
Hugo Daniel Macedo *, José Nuno Oliveira

HASLAB – High Assurance Software Laboratory, Universidade do Minho, Braga, Portugal

ARTICLE INFO

ABSTRACT

The data cube as a typed linear algebra operator


The Data Cube as a Typed Linear Algebra Operator

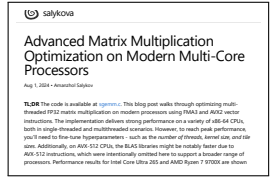
J.N. Oliveira
HASLAB - High Assurance Software Laboratory
INESC TEC & University of Minho
Braga, Portugal
jno@di.uminho.pt

H.D. Macedo
Department of Engineering
Aarhus University
Aarhus, Denmark
hdm@eng.au.dk

ABSTRACT

CCS CONCEPTS

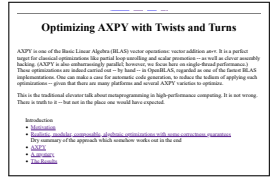
Advanced Matrix Multiplication Optimization


salykova

Advanced Matrix Multiplication Optimization on Modern Multi-Core Processors

Aug 1, 2024 • Amanzhol Salykov

Optimizing AXPY with Twists and Turns


Optimizing AXPY with Twists and Turns

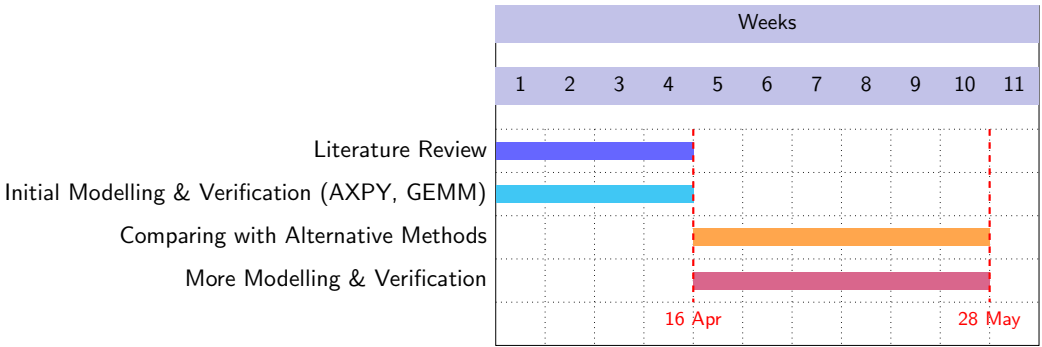
AXPY is one of the Basic Linear Algebra (BLAS) vector operations: vector addition *axpy*. It is a perfect target for classical optimizations like partial loop unrolling and scalar promotion – as well as clever assembly backing. (AXPY is also embarrassingly parallel, however, we focus here on single-thread performance.) These optimizations are indeed carried out – by hand – in OpenBLAS, regarded as one of the fastest BLAS implementations. One can make a case for automatic code generation, to reduce the tedium of applying such optimizations – given that there are many platforms and several AXPY varieties to optimize.

This is the traditional elevator talk about metaprogramming in high-performance computing. It is not wrong. There is truth to it – but not in the place one would have expected.

Introduction

Future Work & Schedule

Future Work & Schedule



Fast (and yet correct!) Tensor Processing

Presented by: Ana Sá Oliveira, Edgar Araújo, Gabriel Paiva

Supervised by: José Nuno Oliveira

University of Minho, Informatics Department

March 2026