On vectors, tensors, matrices, and hypermatrices

Vincent Reverdy

Disclaimer

Warning

The following presentation may contain graphic contents for pure mathematicians. It reflects the standpoint of a computational physicist who has been playing around with differential geometry and general relativity. The author cannot be held responsible for any damage that could be made to spaces of infinite dimensions during this presentation.

Context

Towards linear algebra support in C++

- P0009: mdspan: A Non-Owning Multidimensional Array Reference
- P1417: Historical lessons for C++ linear algebra library standardization
- P1416: SG19 Linear Algebra for Data Science and Machine Learning
- P1166: What do we need from a linear algebra library?
- P1385: A proposal to add linear algebra support to the C++ standard library

Current status

P1385R0: A proposal to add linear algebra support to the C++ standard library

Introduces 3 fundamental classes for linear algebra:

- `row_vector`
- `column_vector`
- `matrix`

On tensors

Tensor algebra is not part of P1385R0

Some of the questions and concerns that were raised in LEWGI

- Is it necessary to have both `row_vector` and `column_vector`?
- What about tensors?
- What should `operator*` be: matrix product, inner product, outer product, Hadamard product. . . ?

Problem statement

Given that:

- Standardizing in layers has been proven to be a working approach for BLAS
- We first need to have the basic building blocks of linear algebra
- Getting them wrong would propagate wrongness in higher layers

Problem

What are the fundamental building blocks we need for the first layer of numerical linear algebra?

Vocabulary types

Vocabulary is key, and we need to get it right. Complexity arises from:

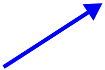
- Overloaded terms
- Different communities may use the same words for different things
- Backward compatibility (`std::vector`)
- Existing practices and libraries

What is this?

$$\begin{pmatrix} 0.5 \\ 1.2 \\ 4.1 \end{pmatrix}$$

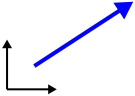
Ceci n'est pas un vecteur

The vector is a lie



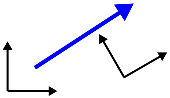
This is probably closer to what a vector look like

The vector is a lie



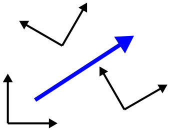
This is probably closer to what a vector look like

The vector is a lie



This is probably closer to what a vector look like

The vector is a lie



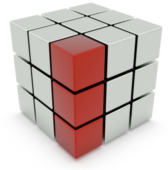
This is probably closer to what a vector look like

Vectors and their representations

On vectors

- A vector is a mathematical object that "exists" independently from its representation in a given base
- A vector has more mathematical "structure" than its representation
- Two vectors belonging to two different vector spaces can have the same coordinates in carefully chosen bases
- Identifying a vector to its coordinates is mathematically wrong, and will lead to inconsistencies

What is this?



(cube of numbers)

Ceci n'est pas un tenseur

Tensors and their representations

On tensors

- A tensor is a mathematical object that “exists” independently from its representation as an array of numbers
- A tensor has more mathematical “structure” than its representation
- It has well-defined rules of transformation (covariance and contravariance properties)

ML tensors

Calling a tensor an array of order/rank N (ML libraries do that) is a terrible mathematical mistake.

Matrices

What a matrix is?

- A matrix is not a rank-2 tensor
- A matrix is not a geometric object
- A matrix is a rank-2 array of numbers

Matrix operations

It has operations (like matrix multiplication) that can be interpreted geometrically when rank-1 arrays are interpreted as the coordinates of vectors, but these operations are independent from their geometric interpretation.

Why should we care?

Risks

If we identify the mathematical object "vector" to its coordinates now, designing a tensor algebra layer on top of a standard linear algebra library will be a nightmare full of inconsistencies.

Historical guess

- So everyone involved in linear algebra libraries has been wrong over the last 50 years?
- In languages with less abstraction capabilities (FORTRAN, C), identifying vectors and their coordinates is convenient
- C++ gives us the opportunity to get things right

What about higher order arrays?

Hypermatrices

Higher order/rank matrices have a name: they are not tensor, they are **hypermatrices**.

What is this?

$$\begin{pmatrix} 0.5 \\ 1.2 \\ 4.1 \end{pmatrix}$$

This is an order/rank-2 hypermatrix of size $(3, 1)$

Hypermatrices

Hypermatrices (correspondance with current names)

- Rank 0: “scalar”
- Rank 1: “vector”
- Rank 2: “matrix”, “row vector”, “column vector”
- Rank 3: “3d matrices”

Layers

Standardization in layers

- Layers of algorithmic complexity
- Layers of mathematical structures

Layers of mathematical structures

- Layer 0: `mdspan`: monodimensional ranges as multidimensional arrays
- Layer 1: multidimensional arrays: storage + `mdspan`
- Layer 2: hypermatrices: mathematical multidimensional arrays
- Layer 3: geometrical objects: vectors, tensors

Remark: layer 1 and 2 may be fused, it's a design decision

Take away

Recommendation

- For the building block of linear algebra, only one abstraction is needed: `hypermatrix`
- It should have rank/order as a template parameter (like `mdspan`)
- Bikeshedding: as the name is not common, this class could be named `matrix` or `basic_matrix`
- The layer of geometric interpretation can be build on top
- If users want non-mathematically accurate names, they can use their typenames
- Making the mistake about vectors, will make tensor algebra a design nightmare