Computing with Tensors: Potential Applications of Physics-Motivated Mathematics to Computer Science

Martine Ceberio  
*University of Texas at El Paso, mceberio@utep.edu*

Vladik Kreinovich  
*University of Texas at El Paso, vladik@utep.edu*

Follow this and additional works at: [http://digitalcommons.utep.edu/cs_techrep](http://digitalcommons.utep.edu/cs_techrep)  
Part of the [Computer Engineering Commons](http://digitalcommons.utep.edu/cs_techrep)

Comments:  
Technical Report: UTEP-CS-08-45  
To appear in *Journal of Uncertain Systems*

**Recommended Citation**  
[http://digitalcommons.utep.edu/cs_techrep/127](http://digitalcommons.utep.edu/cs_techrep/127)
Computing with Tensors: Potential Applications of Physics-Motivated Mathematics to Computer Science

Martine Ceberio and Vladik Kreinovich

Department of Computer Science
University in the Texas at El Paso
500 W. University
El Paso, TX 79968, USA
Emails: mceberio@cs.utep.edu, vladik@utep.edu

Abstract

In this paper, we explain what are tensors and how tensors can help in computing.

1 Why Tensors

One of the main problems of modern computing is that:

• we have to process large amounts of data;
• and therefore, long time required to process this data.

A similar situation occurred in the 19 century physics:

• physicists had to process large amounts of data;
• and, because of the large amount of data, a long time required to process this data.

We will recall that in the 19 century, the problem was solved by using tensors. It is therefore a natural idea to also use tensors to solve the problems with modern computing.
2 Tensors in Physics: A Brief Reminder

Let us recall how tensors helped the 19 century physics; see, e.g., [6]. Physics
starts with measuring and describing the values of different physical quantities.
It goes on to equations which enable us to predict the values of these quantities.

A measuring instrument usually returns a single numerical value. For some
physical quantities (like mass \( m \)), the single measured value is sufficient to
describe the quantity. For other quantities, we need several values. For example,
we need three components \( E_x, E_y, \) and \( E_z \) to describe the electric field at a given
point. To describe the tension inside a solid body, we need even more values:
we need 6 values \( \sigma_{ij} = \sigma_{ji} \) corresponding to different values \( 1 \leq i, j \leq 3: \sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \) and \( \sigma_{13}. \)

The problem was that in the 19 century, physicists used a separate equation
for each component of the field. As a result, equations were cumbersome and
difficult to solve.

The main idea of the tensor approach is to describe all the components of a
physical field as a single mathematical object:

- a vector \( a_i, \)
- or, more generally, a tensor \( a_{ij}, a_{ijk}, \ldots \)

As a result, we got simplified equations – and faster computations.

It is worth mentioning that originally, mostly vectors (rank-1 tensors) were
used. However, the 20 century physics has shown that higher-order matrices
are also useful. For example:

- matrices (rank-2 tensors) are actively used in quantum physics,
- higher-order tensors such as the rank-4 curvature tensor \( R_{ijkl} \) are actively
  used in the General Relativity Theory.

3 From Tensors in Physics to Computing with Tensors

As we have mentioned earlier, 19 century physics encountered a problem of too
much data. To solve this problem, tensors helped.

Modern computing suffers from a similar problem. A natural idea is that
tensors can help. Two examples justify our optimism:

- modern algorithms for fast multiplication of large matrices; and
- quantum computing.
4 Modern Algorithm for Multiplying Large Matrices

In many data processing algorithms, we need to multiply large-size matrices:

\[
\begin{pmatrix}
  a_{11} & \ldots & a_{1n} \\
  \cdots & \cdots & \cdots \\
  a_{n1} & \ldots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
  b_{11} & \ldots & b_{1n} \\
  \cdots & \cdots & \cdots \\
  b_{n1} & \ldots & b_{nn}
\end{pmatrix}
= \begin{pmatrix}
  c_{11} & \ldots & c_{1n} \\
  \cdots & \cdots & \cdots \\
  c_{n1} & \ldots & c_{nn}
\end{pmatrix};
\]

\(c_{ij} = a_{i1} \cdot b_{1j} + \ldots + a_{ik} \cdot b_{kj} + \ldots + a_{in} \cdot b_{nj}.\)  \(\text{(2)}\)

There exist many efficient algorithms for matrix multiplication.

The problem is that for large matrix size \(n\), there is no space for both \(A\) and \(B\) in the fast (cache) memory. As a result, the existing algorithms require lots of time-consuming data transfers (“cache misses”) between different parts of the memory.

An efficient solution to this problem is to represent each matrix as a matrix of blocks; see, e.g., [2, 10]:

\[
A = \begin{pmatrix}
  A_{11} & \ldots & A_{1m} \\
  \cdots & \cdots & \cdots \\
  A_{m1} & \ldots & A_{nm}
\end{pmatrix},
\]

then

\[
C_{\alpha\beta} = A_{\alpha1} \cdot B_{1\beta} + \ldots + A_{\alpha\gamma} \cdot B_{\gamma\beta} + \ldots + A_{\alpha m} \cdot B_{m\beta}.
\]

\(\text{(4)}\)

Comment. For general arguments about the need to use non-trivial representations of 2-D (and multi-dimensional) objects in the computer memory, see, e.g., [21, 22].

In the above idea,

- we start with a large matrix \(A\) of elements \(a_{ij}\);
- we represent it as a matrix consisting of block sub-matrices \(A_{\alpha\beta}\).

This idea has a natural tensor interpretation:

- each element of the original matrix is now represented as
- an \((x, y)\)-th element of a block \(A_{\alpha\beta},\)
- i.e., as an element of a rank-4 tensor \((A_{\alpha\beta})_{xy}.\)

So, in this case, an increase in tensor rank improves efficiency.

Comment. Examples when an increase in tensor rank is beneficial are well known in physics: e.g., a representation of a rank-1 vector as a rank-2 spinor works in relativistic quantum physics [6].
5 Quantum Computing as Computing with Tensors

Classical computation is based on the idea a bit: a system with two states 0 and 1. In quantum physics, due to the superposition principle, we can have states

\[ c_0 \cdot |0 \rangle + c_1 \cdot |1 \rangle \]  

with complex values \( c_0 \) and \( c_1 \); such states are called quantum bits, or qubits, for short.

The meaning of the coefficients \( c_0 \) and \( c_1 \) is that they describe the probabilities to measure 0 and 1 in the given state: \( \text{Prob}(0) = |c_0|^2 \) and \( \text{Prob}(1) = |c_1|^2 \).

Because of this physical interpretation, the values \( c_1 \) and \( c_1 \) must satisfy the constraint \( |c_0|^2 + |c_1|^2 = 1 \).

For an \( n \)-(qu)bit system, a general state has the form

\[ c_{0...00} \cdot |0...00 \rangle + c_{0...01} \cdot |0...01 \rangle + \ldots + c_{1...11} \cdot |1...11 \rangle. \]  

From this description, one can see that each quantum state of an \( n \)-bit system is, in effect, a tensor \( c_{i_1...i_n} \) of rank \( n \).

In these terms, the main advantage of quantum computing is that it can enable us to store the entire tensor in only \( n \) (qu)bits. This advantage explains the known efficiency of quantum computing. For example:

- we can search in an unsorted list of \( n \) elements in time \( \sqrt{n} \) – which is much faster than the time \( n \) which is needed on non-quantum computers [8, 9, 15];

- we can factor a large integer in time which does not exceed a polynomial of the length of this integer – and thus, we can break most existing cryptographic codes like widely used RSA codes which are based on the difficulty of such a factorization on non-quantum computers [15, 18, 19].

6 New Idea: Tensors to Describe Constraints

A general constraint between \( n \) real-valued quantities is a subset \( S \subseteq \mathbb{R}^n \). A natural idea is to represent this subset block-by-block – by enumerating sub-blocks that contain elements of \( S \).

Each block \( b_{i_1...i_n} \) can be described by \( n \) indices \( i_1, \ldots, i_n \). Thus, we can describe a constraint by a boolean-valued tensor \( t_{i_1...i_n} \) for which:

- \( t_{i_1...i_n} = \text{"true"} \) if \( b_{i_1...i_n} \cap S \neq \emptyset \); and

- \( t_{i_1...i_n} = \text{"false"} \) if \( b_{i_1...i_n} \cap S = \emptyset \).

Processing such constraint-related sets can also be naturally described in tensor terms.

This representation speeds up computations; see, e.g., [3, 4].
7 Computing with Tensors Can Also Help Physics

So far, we have shown that tensors can help computing. It is possible that the relation between tensors and computing can also help physics.

As an example, let us consider Kaluza-Klein-type high-dimensional space-time models of modern physics; see, e.g., [7, 11, 12, 13, 16, 20]. Einstein’s original idea [5] was to use “tensors” with integer or circular values to describe these models. From the mathematical viewpoint, such “tensors” are unusual. However, in computer terms, integer or circular data types are very natural: e.g., circular data type means fixed point numbers in which the overflow bits are ignored. Actually, from the computer viewpoint, integers and circular data are even more efficient to process than standard real numbers.

8 Remaining Open Problem

One area where tensors naturally appear is an efficient Taylor series approach to uncertainty propagation; see, e.g., [1, 14, 17]. Specifically, the dependence of the result $y$ on the inputs $x_1, \ldots, x_n$ is approximated by the Taylor series:

$$y = c_0 + \sum_{i=1}^{n} c_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_i \cdot x_j + \ldots$$

(7)

The resulting tensors $c_{i_1 \ldots i_r}$ are symmetric:

$$c_{i_1 \ldots i_r} = c_{\pi(i_1) \ldots \pi(i_r)}$$

(8)

for each permutation $\pi$. As a result, the standard computer representation leads to a $r!$ duplication. An important problem is how to decrease this duplication.

Acknowledgment

This work was supported in part by NSF grant HRD-0734825 and by Grant 1 T36 GM078000-01 from the National Institutes of Health. The authors are thankful to Fred G. Gustavson and Lenore Mullin for her encouragement.

References


