Building blocks: From matrix to tensor operations

Paolo Bientinesi, Lars Karlsson
Umeå Universitet

March 8, 2020
Schloss Dagstuhl

Building blocks

- What are they? Computational bricks, programming modules
- Prominent examples: Linear algebra, Signal processing, Parallel computing

Dense Linear Algebra – 1973

"[..] a fairly small number of basic operations which are generally responsible for a significant percentage of the total execution time" – Hanson, Krogh, Lawson

- `DOT`: $\boldsymbol{w} := \boldsymbol{x}^T \boldsymbol{y}$
- `ELVOP`: $\boldsymbol{y} := \alpha \boldsymbol{x} + \boldsymbol{y}$
- `NRM`: $\eta := (\boldsymbol{x}^T \boldsymbol{x})^{1/2}$

1973 – 1979

- 1973: *"A proposal for standard linear algebra subprograms"* – Hanson, Krogh, Lawson
  Class I: DOT, ELVOP, G2, MG2 – Assembly
  Class II: NRM, XDOT, COPY, SWAP, SCALE, SUM, MAX – Fortran

1973 – 1979

- 1973: *"A proposal for standard linear algebra subprograms"* – Hanson, Krogh, Lawson
  Class I: DOT, ELVOP, G2, MG2 – Assembly
  Class II: NRM, XDOT, COPY, SWAP, SCALE, SUM, MAX – Fortran

- 1974: *"Standardization of FORTRAN callable subprograms for basic linear algebra"* – Lawson

1973 – 1979

- 1973: *"A proposal for standard linear algebra subprograms"* – Hanson, Krogh, Lawson
  Class I: DOT, ELVOP, G2, MG2 – Assembly
  Class II: NRM, XDOT, COPY, SWAP, SCALE, SUM, MAX – Fortran
- 1974: *"Standardization of FORTRAN callable subprograms for basic linear algebra"* – Lawson
- 1975–: LINPACK
- 1977: *"Basic Linear Algebra Subprograms for FORTRAN usage—an extended report"* – Hanson, Krogh, Kinkaid, Lawson
- 1977: *"Fortran BLAS timing"* – Dongarra
  Tests on 24 different computers

1979: BLAS 1

"Basic Linear Algebra Subprograms for FORTRAN usage"
— Hanson, Krogh, Kinkaid, Lawson (ACM TOMS)

"38 subprograms for basic operations of linear algebra"

1979: BLAS 1

"Basic Linear Algebra Subprograms for FORTRAN usage"
— Hanson, Krogh, Kinkaid, Lawson (ACM TOMS)

"38 subprograms for basic operations of linear algebra"

- "aid in **design** and **coding** stages"
- "self-**documenting** quality of code"
- "a reduction of the execution time spent in these operations might be reflected in **cost savings** in the running of programs"
- "the programming of some of these low level operations involves **algorithmic and implementation subtleties** that are likely to be ignored"

- 1988: BLAS 2

  "with some modern machine architectures, the use of the BLAS is not the best way to improve the efficiency of higher level codes. [..] the use of BLAS inhibits this optimization."

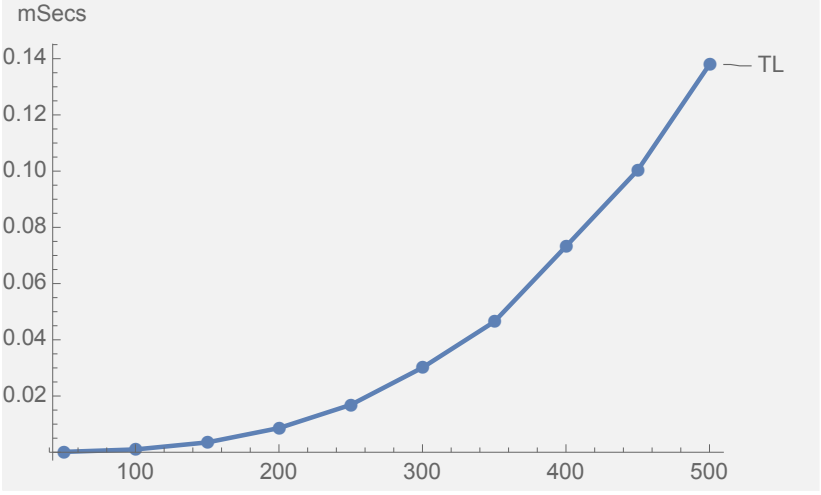
  Matrix-vector operations NOT built on top of BLAS 1

- 1988: BLAS 2
  "with some modern machine architectures, the use of the BLAS is not the best way to improve the efficiency of higher level codes. [..] the use of BLAS inhibits this optimization."
  Matrix-vector operations NOT built on top of BLAS 1

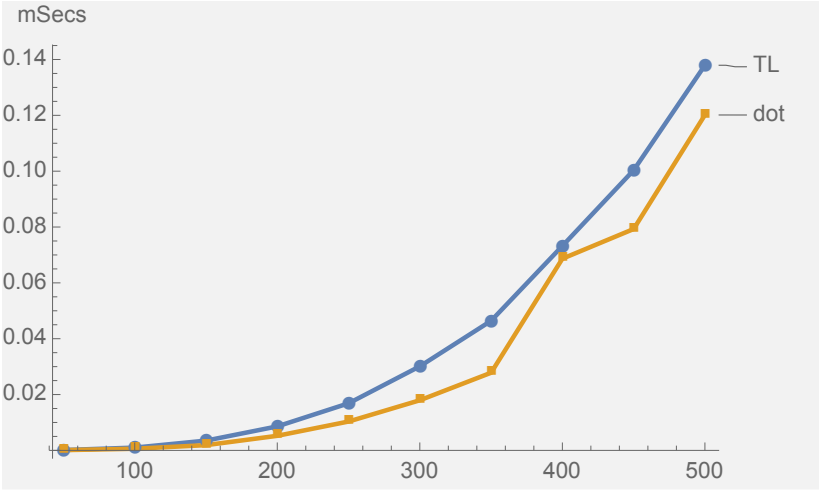
- 1990: BLAS 3
  "Unfortunately, [BLAS 2] is often not well suited to computers with a hierarchy of memory"
  Matrix-matrix operations NOT built on top of BLAS 1 & 2

- Immediate, widespread adoption: LAPACK, ScaLAPACK, PETSc, PLAPACK, ...

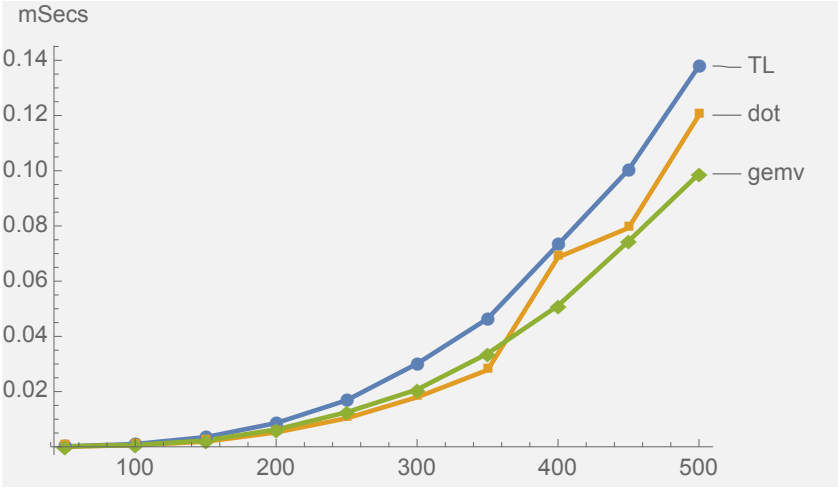
Many ways to skin a cat: Mat-Mat-Mul as a triple loop (no BLAS)



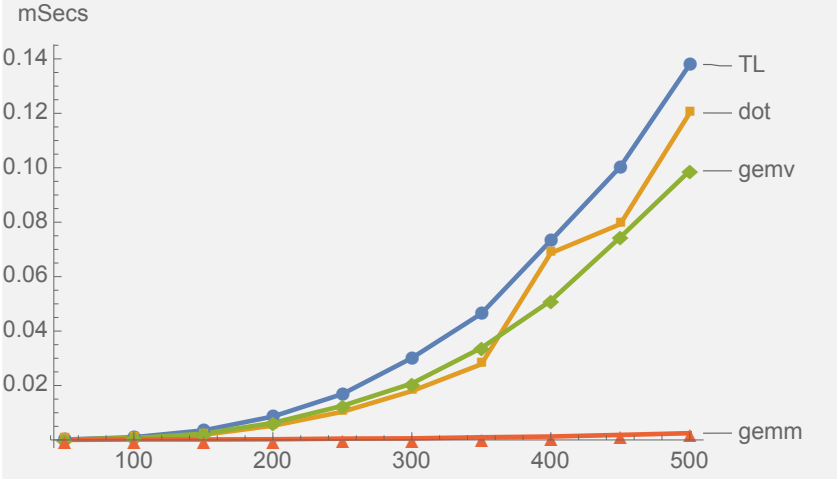
Mat-Mat-Mul via BLAS 1



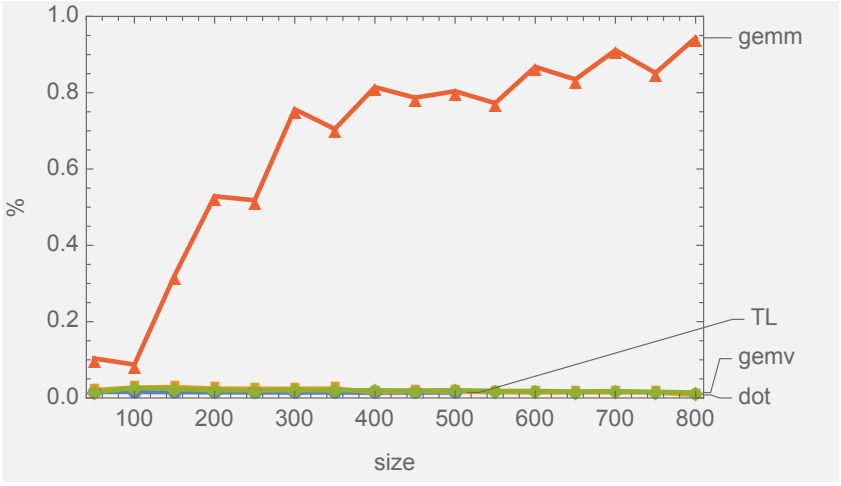
Mat-Mat-Mul via BLAS 2



Mat-Mat-Mul as a GEMM (BLAS 3)



Can we do better? . . . Efficiency



Building blocks for tensor operations?

- Two separate worlds
  - Contractions — Computational physics / chemistry
  - Decompositions — Data science
- Very few software efforts cut across the boundary

Data layout operations

- Reshape
- Permute / transpose
- Sort (sparse)
- Convert data layout
- Partition
- Distribute
- ...

Arithmetic operations

- Add, subtract, multiply, scale
- Inner product
- Norm
- Tensor-times-vector (TTV)
- Tensor-times-matrix (TTM)
- MTTKRP
- Contractions, general
- ...

Decompositions

- CP (CANDECOMP/PARAFAC)
- Tucker
- INDSCAL
- PARAFAC2
- CANDELINC
- DEDICOM
- PARATUCK2
- ...

Tensor BLAS? BLAS 4?

- Contractions, transpositions, ...
- TTV, TTM, Khatri-Rao, MTTKRP, dense/sparse, ...

- Efficiency: wide range of arithmetic intensity
- Layered on BLAS 3 or separate?

Tensor BLAS? BLAS 4?

- Contractions, transpositions, ...
- TTV, TTM, Khatri-Rao, MTTKRP, dense/sparse, ...

- Efficiency: wide range of arithmetic intensity
- Layered on BLAS 3 or separate?

- **Benefits?**
  - Aid in design and coding through a common language
  - Better performance through optimization and auto-tuning
  - Avoid re-inventing the wheel

BLAS/LAPACK – key aspects

- Performance & HW-driven development

BLAS/LAPACK – key aspects

- Performance & HW-driven development
- Community effort: “BLAS Technical Forum” (BLAST), NSF funding, ...
  See also “MPI Forum”, “OpenMP Architecture Review Board”
  → **standardization**

BLAS/LAPACK – key aspects

- Performance & HW-driven development
- Community effort: “BLAS Technical Forum” (BLAST), NSF funding, ...
  See also “MPI Forum”, “OpenMP Architecture Review Board”
  $\rightarrow$ **standardization**
- Preferred outlet: ACM TOMS

BLAS/LAPACK – key aspects

- Performance & HW-driven development
- Community effort: "BLAS Technical Forum" (BLAST), NSF funding, ...
  See also "MPI Forum", "OpenMP Architecture Review Board"
  $\rightarrow$ **standardization**
- Preferred outlet: ACM TOMS
- Specification $\rightarrow$ **automation**
  PhiPAC (1997)/ATLAS (2000), many others (FLAME, CL1CK, LGEN, ...)
  See also FFTW (1998), Sparsity/Oski (2000), Spiral (2000), **TCE** (2001) $\leftarrow$ tensors!
- Abstraction $\rightarrow$ high-level languages
  Matlab, C++, Python, Julia

Summary

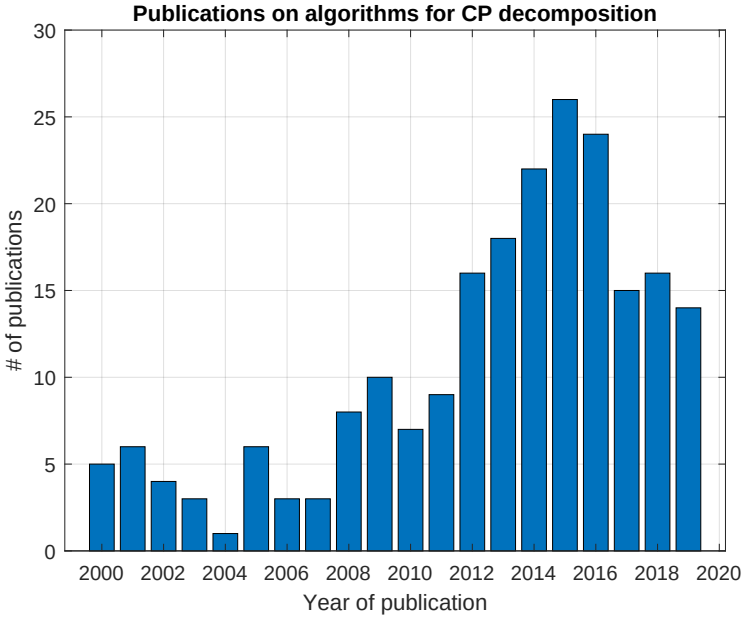
	Matrices	Tensors
Driver	Performance, HW	
Standardization	Interface, ...	
Community effort	BLAST/LAPACK/...	
Preferred outlet	ACM TOMS	
Industry	Wide support	
Language support	plenty	
Automation	plenty	

Building blocks for tensor operations?

- Two separate worlds
  - Contractions — Computational physics / chemistry
  - Decompositions — Data science

Story

- **Background in matrix decompositions and data layout conversion**
- **Started surveying the field of tensor decompositions**
  - From an algorithmic and software perspective
- **Quickly realized there is an abundance of**
  - Decompositions (CP, Tucker, ...)
  - Variants thereof (non-negative, orthogonal, ...)
  - Algorithms (alternating, all-at-once, algebraic, ...)
  - Software packages & languages
  - Papers on software without software


Publications on algorithms for CP decomposition
# of publications
Year of publication
0
5
10
15
20
25
30
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020

Algorithms for CP decomposition

- **Algebraic algorithms**
  - Generalized Rank Annihilation Method
  - Direct TriLinear Decomposition
  - The "algebraic algorithm" by Domanov and De Lathauwer
  - The "simpler algorithm" by Pimentel-Alarcón
  - ...
- **Alternating optimization algorithms**
  - Alternating Least Squares
  - Fast ALS
  - Hierarchical ALS
  - Regularized ALS
  - ...
- **All-at-once optimization algorithms**
  - Gradient descent
  - (Damped) Gauss–Newton
  - Nonlinear CG, GMRES
  - Quasi-Netwon (e.g., L-BFGS)
  - ...
- **Enhancements**
  - Line search
  - Compression
  - Randomization
  - Transient constraints
  - ...

Matlab and R packages with support for CP decomposition (subset)

- **Tensor Toolbox** by Bader, Kolda, & others
  `https://www.tensortoolbox.org/`
- **Tensorlab** by Vervliet, Debals, Sorber, Van Barel, & De Lathauwer
  `https://www.tensorlab.net/index.html`
- **The N-way Toolbox** by Bro & Andersson
  `http://www.models.life.ku.dk/nwaytoolbox`
- **TensorBox** by Phan, Tichavsky, & Cichocki
  `https://github.com/phananhhuy/TensorBox`
- **Tensor Package** by Comon & others
  `http://www.gipsa-lab.fr/~pierre.comon/TensorPackage/tensorPackage.html`

- **multiway** by Helwig
  `https://cran.r-project.org/package=multiway`
- **ThreeWay** by Giordani, Kiers, & Del Ferraro
  `https://cran.r-project.org/package=ThreeWay`
- **rTensor** by Li, Bien, & Wells
  `https://cran.r-project.org/package=rTensor`

C/C++ packages with support for CP decomposition (subset)

- **Genten** by SANDIA (Phipps)
  `https://gitlab.com/tensors/genten`
- **SPLATT** by Smith & Karypis
  `https://github.com/ShadenSmith/splatt`
- **ParTI!** by Li, Ma, & Vuduc
  `https://github.com/hpcgarage/ParTI`
- **Cyclops** by Solomonik & others
  `https://github.com/cyclops-community`

And then there's Python, Fortran, ...

Algorithm versus implementation

- Which **algorithm** is fastest?

Algorithm versus implementation

- Which **algorithm** is fastest?
- Cannot measure the runtime of an algorithm
- **Runtimes** are measured on **implementations**
- **Impl. A** faster than **Impl. B** $\not\Rightarrow$ **Alg. A** faster than **Alg. B**

Algorithm versus implementation

- Which **algorithm** is fastest?
- Cannot measure the runtime of an algorithm
- **Runtimes** are measured on **implementations**
- **Impl. A** faster than **Impl. B** $\not\Rightarrow$ **Alg. A** faster than **Alg. B**
- What if an implementation is flawed or not even available?
- Comparisons involving runtimes have a terrible shelf-life
- **Question**: Need for reproducible benchmarks?

Algorithm versus implementation

- Which **algorithm** is fastest?
- Cannot measure the runtime of an algorithm
- **Runtimes** are measured on **implementations**
- **Impl. A** faster than **Impl. B** $\not\Rightarrow$ **Alg. A** faster than **Alg. B**
- What if an implementation is flawed or not even available?
- Comparisons involving runtimes have a terrible shelf-life
- **Question**: Need for reproducible benchmarks?

> *"A library of test problems supported by the community would facilitate the assessment of new algorithms and implementations."* — ***Charles Van Loan***, *2009*

Algorithm versus implementation, example 1

ASD and SWATLD are the best alternatives. ASD has an advantage over SWATLD, because it converges in reasonable CPU time.
DTLD is fastest, but does not yield a good solution. However, the solution may be good enough for initialisation purposes.

Calculations are performed using Matlab (Mathworks, Natick, MA). The code for DTLD and ALS is taken from the N-way Toolbox [29]. The function for ACOVER is adapted from Jiang et al. [38]. ATLD, ASD, ACOMAR, SWATLD and PALS are performed using in-house functions. The following alterations have been found to be necessary:

Algorithm versus implementation, example 2

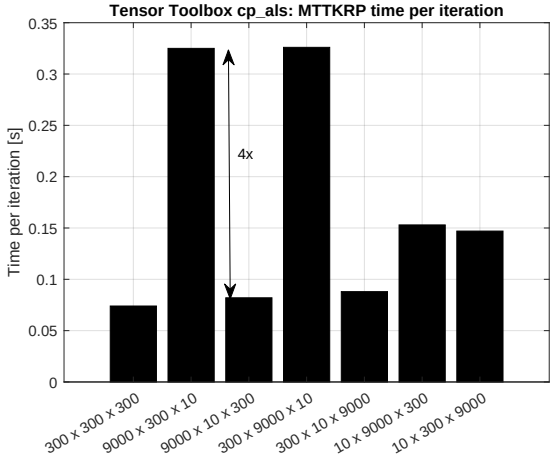
Tensor Toolbox's `cp_als` performance depends on the shape of the tensor.

(An optimal implementation should barely be affected by the shape.)

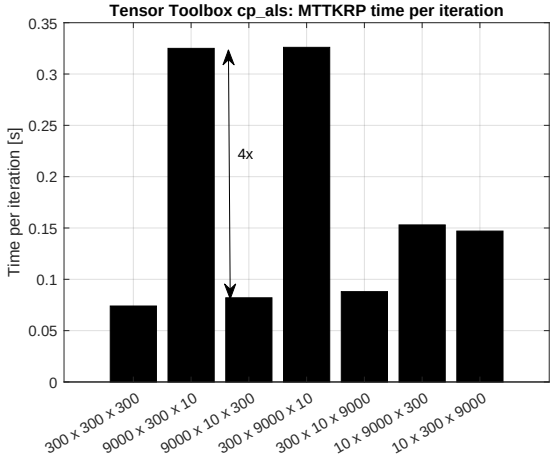
Experiment setup:

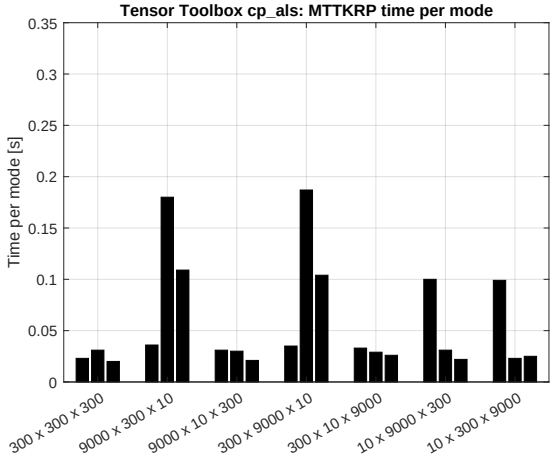
- $\boldsymbol{F = 10}$ terms in the CP model
- Seven three-way arrays with 27M elements each
- Shapes:
  1. $\mathbf{300 \times 300 \times 300}$
  2. $\mathbf{9000 \times 300 \times 10}$
  3. $\mathbf{9000 \times 10 \times 300}$
  4. $\mathbf{300 \times 9000 \times 10}$
  5. $\mathbf{300 \times 10 \times 9000}$
  6. $\mathbf{10 \times 9000 \times 300}$
  7. $\mathbf{10 \times 300 \times 9000}$
- Measure the time per MTTKRP for each mode

Algorithm versus implementation, example 2



Algorithm versus implementation, example 2





Back to building blocks . . .

- **MTTKRP is an obvious candidate**
  - (RE-)Implemented in every package / language
  - Performance critical (and basically a mat-mul)
  - Observing wild differences in performance
    Shape, package

Back to building blocks . . .

- **MTTKRP is an obvious candidate**
  - (RE-)Implemented in every package / language
  - Performance critical (and basically a mat-mul)
  - Observing wild differences in performance
    Shape, package
- **Should be. . .**
  - Implemented once, used many times
  - Highly optimized
  - Optimized for all inputs (as opposed to the common inputs)
  - Compare with BLAS and LAPACK

Back to building blocks . . .

- **MTTKRP is an obvious candidate**
  - (RE-)Implemented in every package / language
  - Performance critical (and basically a mat-mul)
  - Observing wild differences in performance
    Shape, package
- **Should be. . .**
  - Implemented once, used many times
  - Highly optimized
  - Optimized for all inputs (as opposed to the common inputs)
  - Compare with BLAS and LAPACK
- **We have a long way to go (just for MTTKRP)**

Comparing matrix and tensor efforts

	Matrices	**Tensors**
Driver	Performance, HW	Applications
Community effort	BLAST/LAPACK/...	group by group
Industry	Wide support	not much
Standardization	Interface, ...	"pointless"
Preferred outlet	ACM TOMS	—
Language support	plenty	language by language
Automation	plenty	TCE (2001), but then?

Comparing matrix and tensor efforts

	Matrices	**Tensors**
Driver	Performance, HW	Applications
Community effort	BLAST/LAPACK/...	group by group
Industry	Wide support	not much
Standardization	Interface, ...	"pointless"
Preferred outlet	ACM TOMS	—
Language support	plenty	language by language
Automation	plenty	TCE (2001), but then?

"It is pointless to recommend the adoption of a universal tensor notation. However, it would be extremely useful to have a thesaurus that shows how to articulate typical calculations in the different notations and a companion collection of LaTeX macros to facilitate the writing of tensor-related documents." — **Charles Van Loan**, *2009*

Downsides of building blocks

- **Black box nature becomes a limitation**
  - Special properties
  - Successive invocations

Downsides of building blocks

- **Black box nature becomes a limitation**
  - Special properties
  - Successive invocations
- **Fixed functionality becomes a constraint to algorithm development**
  - GEMM-driven development

Questions for the workshop

1. **Is the time ripe for a common open-source software project?**
   - What are the benefits?
   - What are the challenges?

Questions for the workshop

1. **Is the time ripe for a common open-source software project?**
   - What are the benefits?
   - What are the challenges?

2. **What would be a suitable scope and architecture?**
   - Separate domain areas? (e.g., biology, information science)
   - Layers of packages? (similar to BLAS, LAPACK)?

Questions for the workshop

1. **Is the time ripe for a common open-source software project?**
   - What are the benefits?
   - What are the challenges?

2. **What would be a suitable scope and architecture?**
   - Separate domain areas? (e.g., biology, information science)
   - Layers of packages? (similar to BLAS, LAPACK)?

3. **How to organize and fund such an effort?**
   - What can we learn from history?

Advice from the past

> *"We need to make tensor computations as 'easy' as matrix computations. New libraries will be required if the latest and greatest algorithmic ideas are to be accessible to researchers in the sciences and engineering. It is unclear whether there is enough of a 'common denominator' across tensor applications to warrant the development of a tensor LAPACK; packages may have to be specialized to domain areas such as biology and information science."*
>
> *"However, it is very important for the tensor computation community to track research developments associated with high-performance numerical linear algebra. Topics include the exploitation of the multicore architectures, the use of self-tuning linear algebra code-generators, the exploitation of recursive data structures, and the extrapolation of the BLAS philosophy."*

— **Charles Van Loan**, 2009
Workshop on Future Directions in Tensor-Based Computation and Modeling