Tensor computations: A fragmented landscape

Paolo Bientinesi
pauldj@cs.umu.se

January 20, 2021
Huawei (via Zoom)

UMEÅ UNIVERSITY

About me

HPAC High Performance and Automatic Computing

HPC2N

High-Performance Computing Center North

- Taxonomy of contractions: Can you GEMM? — E. Di Napoli, D. Traver-Fabregat

  "Towards an Efficient Use of the BLAS Library for Multilinear Tensor Contractions", AMC 235, 2014

- Performance prediction — E. Peise

  "On the Performance Prediction of BLAS-based Tensor Contractions", PMBS, SC'14

- Density Functional Theory: FLAPW methods — E. Di Napoli, E. Peise

  "High-Performance Generation of the Hamiltonian and Overlap Matrices in FLAPW Methods", CPC 2017

- High-performance kernels — P. Springer

  "TTC: A high-performance Compiler for Tensor Transpositions", ACM TOMS 44(2), 2017
  "Design of a High-Performance GEMM-like Tensor-Tensor Multiplication", ACM TOMS 44(3), 2018
  "Spin Summations: A High-Performance Perspective", ACM TOMS 45(1), 2019

- High-intensity kernels — C. Psarras, L. Karsson

  "Concurrent Alternating Least Squares for multiple simultaneous Canonical Polyadic Decompositions", 2020

Outline

- Part 1: (Dense) Linear Algebra – historical overview
- Part 2: Tensor Operations

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
- 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"

- "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

- 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, ...

- Specialization, optimization, auto-tuning, high-level notation, automation, ...

But ...

But ...

- Rigid interface
- Inflexible black-box nature
- (Often) Sub-optimal at small scale
- . . .

In practice:

Signal Processing	$x := \left(A^{-T}B^TBA^{-1} + R^TLR\right)^{-1} A^{-T}B^TBA^{-1}y$	$R \in \mathbb{R}^{n-1\times n}$, UT; $L \in \mathbb{R}^{n-1\times n-1}$, DI
Kalman Filter	$K_k := P_k^bH^T(HP_k^bH^T + R)^{-1};\ x_k^a := x_k^b + K_k(z_k - Hx_k^b);\ P_k^a := (I - K_KH)\, P_k^b$	
Ensemble Kalman Filter	$X^a := X^b + \left(B^{-1} + H^TR^{-1}H\right)^{-1}\left(Y - HX^b\right)$	$B \in \mathbb{R}^{N\times N}$ SSPD; $R \in \mathbb{R}^{m\times m}$, SSPD
Image Restoration	$x_k := (H^TH + \lambda\sigma^2 I_n)^{-1}(H^Ty + \lambda\sigma^2(v_{k-1} - u_{k-1}))$	
Rand. Matrix Inversion	$\Lambda := S(S^TAWAS)^{-1}S^T;\ \Theta := \Lambda AW;\ M_k := X_kA - I$ $X_{k+1} := X_k - M_k\Theta - (M_k\Theta)^T + \Theta^T(AX_kA - A)\Theta$	
Generalized Least Squares	$b := (X^TM^{-1}X)^{-1}X^TM^{-1}y$	$n > m$; $M \in \mathbb{R}^{n\times n}$, SPD; $X \in \mathbb{R}^{n\times m}$; $y \in \mathbb{R}^n$
Stochastic Newton	$B_k := \frac{k}{k-1}B_{k-1}(I_n - A^TW_k((k-1)I_l + W_k^TAB_{k-1}A^TW_k)^{-1}W_k^TAB_{k-1})$	
Optimization	$x_f := WA^T(AWA^T)^{-1}(b - Ax);\ \ x_o := W(A^T(AWA^T)^{-1}Ax - c)$	
Tikhonov Regularization	$x := (A^TA + \Gamma^T\Gamma)^{-1}A^Tb$	$A \in \mathbb{R}^{n\times m}$; $\Gamma \in \mathbb{R}^{m\times m}$; $b \in \mathbb{R}^{n\times 1}$
Gen. Tikhonov Reg.	$x := (A^TPA + Q)^{-1}(A^TPb + Qx_0)$	$P \in \mathbb{R}^{n\times n}$, SSPD; $Q \in \mathbb{R}^{m\times m}$, SSPD; $x_0 \in \mathbb{R}$
LMMSE estimator	$K_{t+1} := C_tA^T(AC_tA^T + C_z)^{-1};\ x_{t+1} := x_t + K_{t+1}(y - Ax_t);\ C_{t+1} := (I - K_{t+1}A)C_t$	

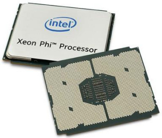
. . .

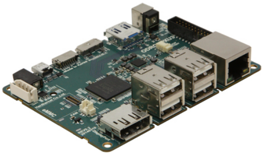
$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H)\, P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k\Theta - (M_k\Theta)^T + \Theta^T (A X_k A - A)\Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \qquad \ldots \qquad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$





MUL ADD MOV

MOVAPD

VFMADDPD ...

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H)\, P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

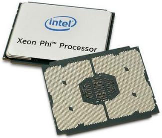
$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

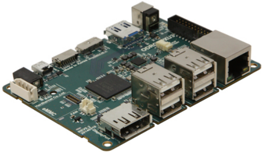
$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \dots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

$y := \alpha x + y$ | $LU = A$ | $\cdots$ | $C := \alpha AB + \beta C$

$X := A^{-1} B$ | $C := AB^T + BA^T + C$ | $X := L^{-1} M L^{-T}$ | $QR = A$

... **BLAS** **LAPACK** ...





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

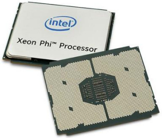
$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

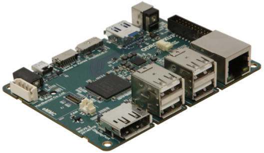
?

$y := \alpha x + y$ | $LU = A$ | $\cdots$ | $C := \alpha AB + \beta C$

$X := A^{-1}B$ | $C := AB^T + BA^T + C$ | $X := L^{-1} M L^{-T}$ | $QR = A$

... BLAS LAPACK ...





MUL ADD MOV

MOVAPD

VFMADDPD ...

$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$

$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$

$x := A(B^T B + A^T R^T \Lambda R A$ … $^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$

LINEAR ALGEBRA MAPPING PROBLEM ("LAMP")

$y := \alpha x + y$ | $LU = A$ | $\cdots$ | $C := \alpha AB + \beta C$

$X := A^{-1} B$ | $C := AB^T + BA^T + C$ | $X := L^{-1} M L^{-T}$ | $QR = A$

... BLAS LAPACK ...

MUL | ADD | MOV

MOVAPD

VFMADDPD ...

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A \ldots$$

$$\ldots^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

LINEAR ALGEBRA MAPPING PROBLEM (“LAMP”)

$y := \alpha x + y$ | $LU = A$ | $\cdots$ | $C := \alpha AB + \beta C$

$X := A^{-1}B$ | $C := AB^T + BA^T + C$ | $X := L^{-1} M L^{-T}$ | $QR = A$

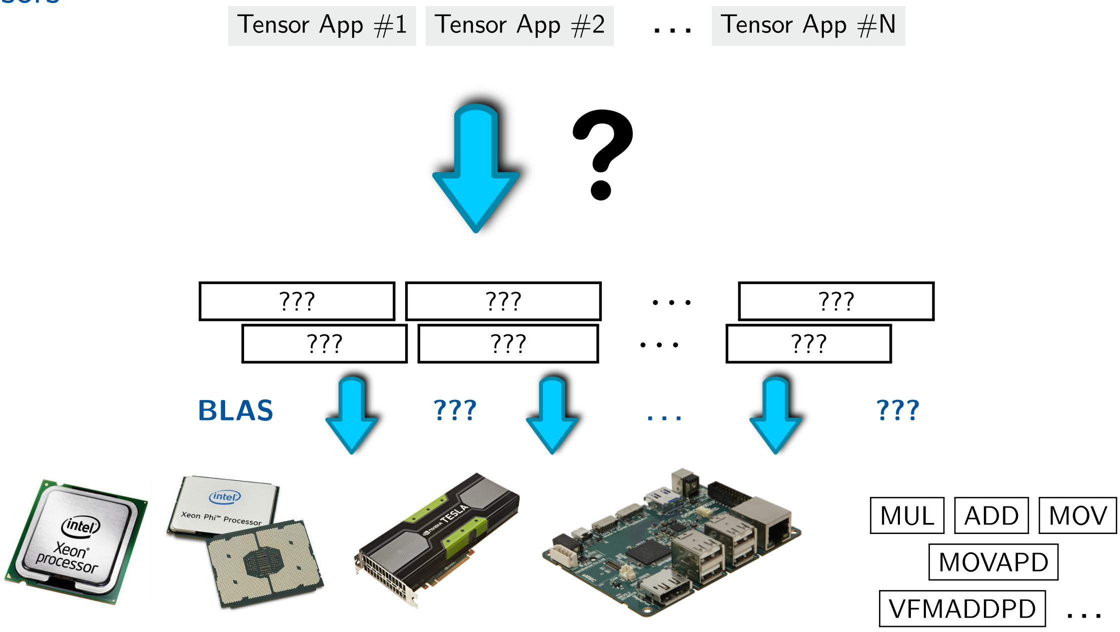
... BLAS LAPACK ...

C. Psarras, H. Barthels, *“The Linear Algebra Mapping Problem. Current state of linear algebra languages and libraries”.* [arXiv:1911.09421]

H. Barthels, C. Psarras, *“Linnea: Automatic Generation of Efficient Linear Algebra Programs”, ACM TOMS, 2021.* [arXiv:1912.12924]

Tensors

Tensors


Tensor App #1
Tensor App #2
...
Tensor App #N
?
???
???
...
???
???
???
...
???
BLAS
???
...
???
intel
Xeon processor
intel
Xeon Phi™ Processor
TESLA
MUL
ADD
MOV
MOVAPD
VFMADDPD
...

Tensors

- No “Tensor BLAS” – collections of building blocks
- No agreement on interface(s)
- Lack of reference implementations
- A jungle of independent libraries and packages, in a variety of languages

Tensor computations

- Two separate worlds
  - Contractions — Computational physics / chemistry
    Tensor = Multi-linear operator
    Generalization of matrix-matrix product

Tensor computations

- Two separate worlds
  - Contractions — Computational physics / chemistry
    Tensor = Multi-linear operator
    Generalization of matrix-matrix product
  - Decompositions — Data science
    Tensor = Collection of data
    Generalization of matrix factorizations

Tensor computations

- Two separate worlds
  - Contractions — Computational physics / chemistry
    Tensor = Multi-linear operator
    Generalization of matrix-matrix product
  - Decompositions — Data science
    Tensor = Collection of data
    Generalization of matrix factorizations
- Terminology and notation vary (and conflict) even within one world
- Very few software efforts cut across the boundary

Representative operations

Data layout operations

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

Arithmetic operations

- Add, subtract, scale
- Inner product
- Norms
- Element-wise operations
- Tensor-times-vector (TTV)
- Tensor-times-matrix (TTM)
- MTTKRP
- Contractions
- ...

Decompositions

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

Representative operations

Data layout operations

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

Arithmetic operations

- Add, subtract, scale
- Inner product
- Norms
- Element-wise operations
- Tensor-times-vector (TTV)
- Tensor-times-matrix (TTM)
- MTTKRP
- Contractions
- ...

Decompositions

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

In setting up a library, where to draw the boundaries?

Contractions

Contractions

- **Tensor Transpositions**

$$\mathcal{B}_{i_1 i_2 \ldots i_N} \leftarrow \alpha \cdot \mathcal{A}_{\pi(i_1 i_2 \ldots i_N)} + \beta \cdot \mathcal{B}_{i_1 i_2 \ldots i_N}$$

- **Summations** — linear summation over tensor transpositions

$$\mathcal{B}_{i_0 i_1 i_2} \leftarrow 2\mathcal{A}_{i_0 i_1 i_2} - \mathcal{A}_{i_2 i_1 i_0} - \mathcal{A}_{i_0 i_2 i_1}$$

$$\mathcal{B}_{i_0 i_1 i_2} \leftarrow 4\mathcal{A}_{i_0 i_1 i_2} - 2\mathcal{A}_{i_1 i_0 i_2} - 2\mathcal{A}_{i_2 i_1 i_0} + \mathcal{A}_{i_1 i_2 i_0} - 2\mathcal{A}_{i_0 i_2 i_1} + \mathcal{A}_{i_2 i_0 i_1}$$

$$\mathcal{B}_{i_0 i_1 i_2 i_3} \leftarrow 2\mathcal{A}_{i_0 i_1 i_2 i_3} - \mathcal{A}_{i_2 i_1 i_0 i_3} - \mathcal{A}_{i_0 i_2 i_1 i_3} - \mathcal{A}_{i_0 i_1 i_3 i_2}$$

- **Tensor Contractions**

$$\mathcal{C}_{\pi_{\mathcal{C}}(I_m \cup I_n)} \leftarrow \alpha \cdot \mathcal{A}_{\pi_{\mathcal{A}}(I_m \cup I_k)} \times \mathcal{B}_{\pi_{\mathcal{B}}(I_n \cup I_k)} + \beta \cdot \mathcal{C}_{\pi_{\mathcal{C}}(I_m \cup I_n)}$$

- **Tensor Transpositions**

  TTC: A high-performance Compiler for Tensor Transpositions. ACM TOMS, 2017

  Compiler: `https://github.com/HPAC/TTC` **Library**: `https://github.com/HPAC/hptt`

- **Summations** — linear summation over tensor transpositions

  Spin Summations: A High-Performance Perspective. ACM TOMS, 2019

  Generator: `https://github.com/springer13/spin-summations`

- **Tensor Contractions**

  Design of a high-performance GEMM-like Tensor-Tensor Multiplication. ACM TOMS, 2018

  Compiler: `https://github.com/HPAC/tccg` **Library**: `https://github.com/springer13/tcl`

But . . .

But . . .

Coupled-Cluster methods

$$
\begin{aligned}
\tau_{ij}^{ab} &= t_{ij}^{ab} + \frac{1}{2} P_b^a P_j^i t_i^a t_j^b, \\
\tilde{F}_e^m &= f_e^m + \sum_{fn} v_{ef}^{mn} t_n^f, \\
\tilde{F}_e^a &= (1-\delta_{ae}) f_e^a - \sum_{m} \tilde{F}_e^m t_m^a - \frac{1}{2} \sum_{mnf} v_{ef}^{mn} t_{mn}^{af} + \sum_{fn} v_{ef}^{an} t_n^f, \\
\tilde{F}_i^m &= (1-\delta_{mi}) f_i^m + \sum_{e} \tilde{F}_e^m t_i^e + \frac{1}{2} \sum_{nef} v_{ef}^{mn} t_{in}^{ef} + \sum_{fn} v_{if}^{mn} t_n^f,
\end{aligned}
$$

$$
\begin{aligned}
\tilde{W}_{ei}^{mn} &= v_{ei}^{mn} + \sum_{f} v_{ef}^{mn} t_i^f, \\
\tilde{W}_{ij}^{mn} &= v_{ij}^{mn} + P_j^i \sum_{e} v_{ie}^{mn} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{mn} \tau_{ij}^{ef}, \\
\tilde{W}_{ie}^{am} &= v_{ie}^{am} - \sum_{n} \tilde{W}_{ei}^{mn} t_n^a + \sum_{f} v_{ef}^{ma} t_i^f + \frac{1}{2} \sum_{nf} v_{ef}^{mn} t_{in}^{af}, \\
\tilde{W}_{ij}^{am} &= v_{ij}^{am} + P_j^i \sum_{e} v_{ie}^{am} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{am} \tau_{ij}^{ef}, \\
z_i^a &= f_i^a - \sum_{m} \tilde{F}_i^m t_m^a + \sum_{e} f_e^a t_i^e + \sum_{em} v_{ei}^{ma} t_m^e + \sum_{em} v_{im}^{ae} \tilde{F}_e^m + \frac{1}{2} \sum_{efm} \\
z_{ij}^{ab} &= v_{ij}^{ab} + P_j^i \sum_{e} v_{ie}^{ab} t_j^e + P_b^a P_j^i \sum_{me} \tilde{W}_{ie}^{am} t_{mj}^{eb} - P_b^a \sum_{m} \tilde{W}_{ij}^{am} t_m^b + P
\end{aligned}
$$

credits to D. Matthews, E. Solomonik, J. Stanton, and J. Gauss

But . . .

Coupled-Cluster methods

$$\tau_{ij}^{ab} = t_{ij}^{ab} + \frac{1}{2} P_b^a P_j^i t_i^a t_j^b,$$

$$\tilde{F}_e^m = f_e^m + \sum_{fn} v_{ef}^{mn} t_n^f,$$

$$\tilde{F}_e^a = (1-\delta_{ae}) f_e^a - \sum_m \tilde{F}_e^m t_m^a - \frac{1}{2} \sum_{mnf} v_{ef}^{mn} t_{mn}^{af} + \sum_{fn} v_{ef}^{an} t_n^f,$$

$$\tilde{F}_i^m = (1-\delta_{mi}) f_i^m + \sum_e \tilde{F}_e^m t_i^e + \frac{1}{2} \sum_{nef} v_{ef}^{mn} t_{in}^{ef} + \sum_{fn} v_{if}^{mn} t_n^f,$$

$$\tilde{W}_{ei}^{mn} = v_{ei}^{mn} + \sum_f v_{ef}^{mn} t_i^f,$$

$$\tilde{W}_{ij}^{mn} = v_{ij}^{mn} + P_j^i \sum_e v_{ie}^{mn} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{mn} \tau_{ij}^{ef},$$

$$\tilde{W}_{ie}^{am} = v_{ie}^{am} - \sum_n \tilde{W}_{ei}^{mn} t_n^a + \sum_f v_{ef}^{ma} t_i^f + \frac{1}{2} \sum_{nf} v_{ef}^{mn} t_{in}^{af},$$

$$\tilde{W}_{ij}^{am} = v_{ij}^{am} + P_j^i \sum_e v_{ie}^{am} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{am} \tau_{ij}^{ef},$$

$$z_i^a = f_i^a - \sum_m \tilde{F}_i^m t_m^a + \sum_e f_e^a t_i^e + \sum_{em} v_{ei}^{ma} t_m^e + \sum_{em} v_{im}^{ae} \tilde{F}_e^m + \frac{1}{2} \sum_{efm}$$

$$z_{ij}^{ab} = v_{ij}^{ab} + P_j^i \sum_e v_{ie}^{ab} t_j^e + P_b^a P_j^i \sum_{me} \tilde{W}_{ie}^{am} t_{mj}^{eb} - P_b^a \sum_m \tilde{W}_{ij}^{am} t_m^b + P$$

credits to D. Matthews, E. Solomonik, J. Stanton, and J. Gauss

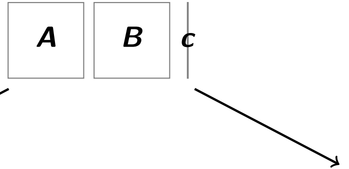
Finite Element 3D diffusion operator

```
TE.BeginMultiKernelLaunch();
TE("T2_e_i1_i2_k3 = B_k3_i3 X_e_i1_i2_i3", T2, B, X);
TE("T1_e_i1_k2_k3 = B_k2_i2 T2_e_i1_i2_k3", T1, B, T2);
TE("U1_e_k1_k2_k3 = G_k1_i1 T1_e_i1_k2_k3", U1, G, T1);
TE("T1_e_i1_k2_k3 = G_k2_i2 T2_e_i1_i2_k3", T1, G, T2);
TE("U2_e_k1_k2_k3 = B_k1_i1 T1_e_i1_k2_k3", U2, B, T1);
TE("T2_e_i1_i2_k3 = G_k3_i3 X_e_i1_i2_i3", T2, G, X);
TE("T1_e_i1_k2_k3 = B_k2_i2 T2_e_i1_i2_k3", T1, B, T2);
TE("U3_e_k1_k2_k3 = B_k1_i1 T1_e_i1_k2_k3", U3, B, T1);
TE("Z_m_e_k1_k2_k3 = U_n_e_k1_k2_k3 D_e_m_n_k1_k2_k3", Z, U,
TE("T1_e_i3_k1_k2 = B_k3_i3 Z1_e_k1_k2_k3", T1, B, Z1);
TE("T2_e_i2_i3_k1 = B_k2_i2 T1_e_i3_k1_k2", T2, B, T1);
TE("Y_e_i1_i2_i3 = G_k1_i1 T2_e_i2_i3_k1", Y, G, T2);
TE("T1_e_i3_k1_k2 = B_k3_i3 Z2_e_k1_k2_k3", T1, B, Z2);
TE("T2_e_i2_i3_k1 = G_k2_i2 T1_e_i3_k1_k2", T2, G, T1);
TE("Y_e_i1_i2_i3 += B_k1_i1 T2_e_i2_i3_k1", Y, B, T2);
TE("T1_e_i3_k1_k2 = G_k3_i3 Z3_e_k1_k2_k3", T1, G, Z3);
TE("T2_e_i2_i3_k1 = B_k2_i2 T1_e_i3_k1_k2", T2, B, T1);
TE("Y_e_i1_i2_i3 += B_k1_i1 T2_e_i2_i3_k1", Y, B, T2);
TE.EndMultiKernelLaunch();
```

credits to A. Fisher – `https://github.com/LLNL/acrotensor`

- “Wrong” level of abstraction for domain scientists
- Mismatch $\rightarrow$ mapping problem

- “Wrong” level of abstraction for domain scientists
- Mismatch $\rightarrow$ mapping problem
- Matrix counterpart: Matrix Chain Problem (aka “parenthesisation”)

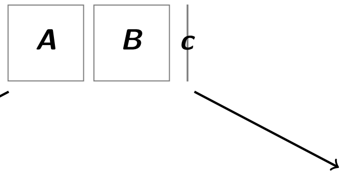


$(\boldsymbol{AB})\boldsymbol{c} \quad \mathcal{O}(\boldsymbol{n}^3)$ $\qquad$ $\boldsymbol{A}(\boldsymbol{Bc}) \quad \mathcal{O}(\boldsymbol{n}^2)$

Product is associative, but its cost is not!

H. Barthels, *“The Generalized Matrix Chain Algorithm”*, CGO’18. [arXiv:1804.04021]

- "Wrong" level of abstraction for domain scientists
- Mismatch $\rightarrow$ mapping problem
- Matrix counterpart: Matrix Chain Problem (aka "parenthesisation")



$\boldsymbol{(AB)c} \quad \mathcal{O}(\boldsymbol{n}^3)$ $\qquad$ $\boldsymbol{A(Bc)} \quad \mathcal{O}(\boldsymbol{n}^2)$

Product is associative, but its cost is not!

H. Barthels, *"The Generalized Matrix Chain Algorithm"*, CGO'18. [arXiv:1804.04021]

- Optimal parenthesisation: Polynomial time (matrices), exponential time (tensors)

Decompositions

With L. Karlsson

Decompositions

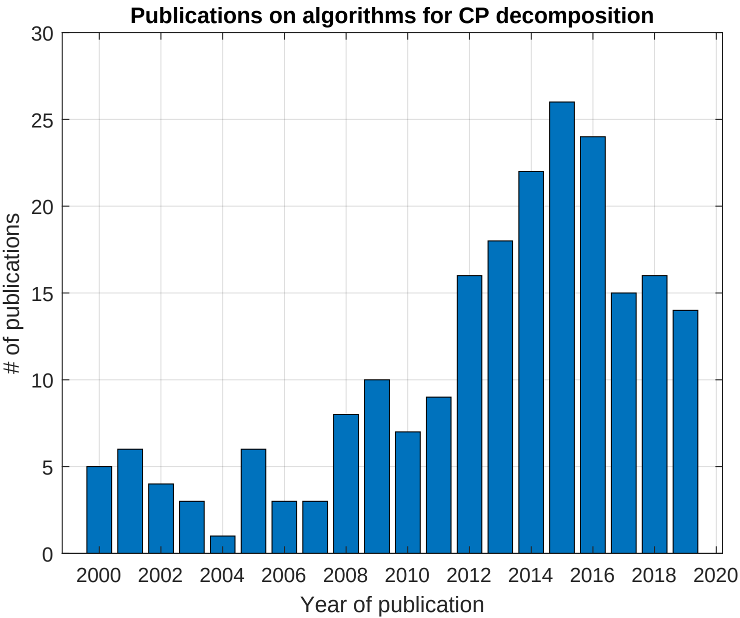
With L. Karlsson

- **Survey of the field**
  - From an algorithmic and software perspective

Decompositions

With L. Karlsson

- **Survey of the field**
  - 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
30
25
20
15
10
5
0
# of publications
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
Year of publication

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, ...

Massive redundancy

- Development (mostly) driven by applications

Massive redundancy

- Development (mostly) driven by applications
- Replication of effort. Wheel reinvented over and over

Massive redundancy

- Development (mostly) driven by applications
- Replication of effort. Wheel reinvented over and over
- Algorithm versus implementation: Which **algorithm** is fastest?

Massive redundancy

- Development (mostly) driven by applications
- Replication of effort. Wheel reinvented over and over
- Algorithm versus implementation: Which **algorithm** is fastest?
  - **Runtimes** are measured on **implementations**
  - Different implementations of the same algorithm (on different languages)
  - **Impl. A** faster than **Impl. B** $\not\Rightarrow$ **Alg. A** faster than **Alg. B**

Is it a good implementation?

Is it a good implementation?

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

An optimal implementation should barely be affected by the shape.

Is it a good implementation?

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:

- CP model of rank 10
- 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. $\mathbf{10 \times 300 \times 9000}$

Is it a good implementation?

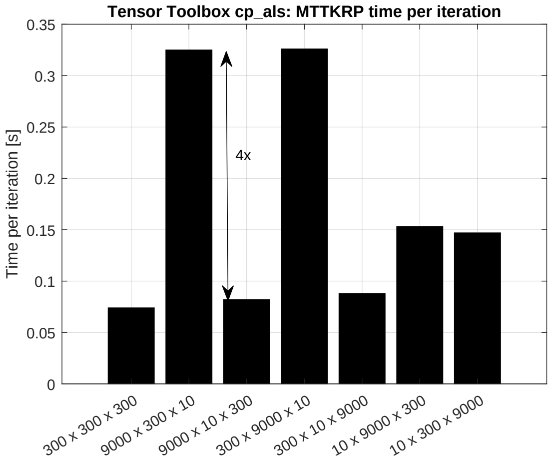
Tensor Toolbox's `cp_als` performance depends on the shape of the tensor.

An optimal implementation should barely be affected by the shape.

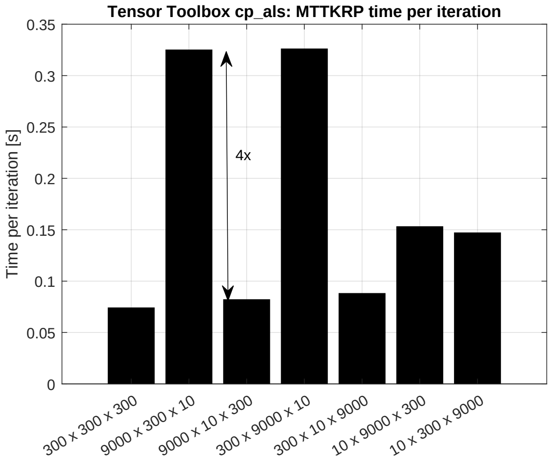
Experiment setup:

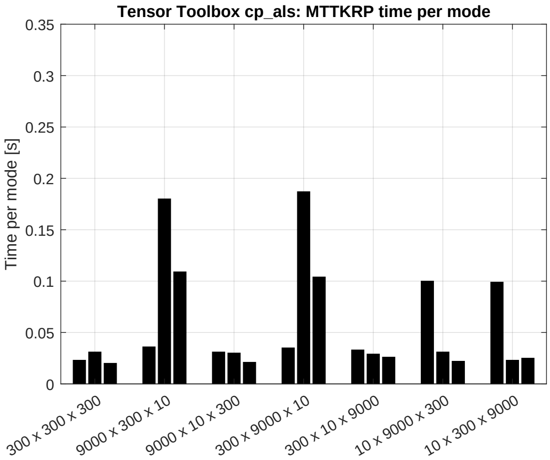
- CP model of rank 10
- Seven three-way arrays with 27M elements each
- Shapes:
  1. $\mathbf{300 \times 300 \times 300}$
  2. $\mathbf{9000 \times 300 \times 10}$
  3. $\mathbf{9000 \times 10 \times 300}$
  4. $\mathbf{300 \times 9000 \times 10}$
  5. $\mathbf{300 \times 10 \times 9000}$
  6. $\mathbf{10 \times 9000 \times 300}$
  7. $\mathbf{10 \times 300 \times 9000}$
- Building block: "Matricized tensor times Khatri-Rao product" (MTTKRP)

Algorithm versus implementation



Algorithm versus implementation





Building blocks for tensor computations

- **MTTKRP is an obvious candidate**
  - (Re-)Implemented in every package / language
  - Performance critical (and basically a matrix multiply)
  - Yielding wild differences in performance (different shapes, different packages)

Building blocks for tensor computations

- **MTTKRP is an obvious candidate**
  - (Re-)Implemented in every package / language
  - Performance critical (and basically a matrix multiply)
  - Yielding wild differences in performance (different shapes, different packages)

- **Should be...**
  - Implemented once, used many times
  - Highly optimized
  - Optimized for all inputs (as opposed to the common inputs)
  - Compared with BLAS in term of efficiency

Building blocks for tensor computations

- **MTTKRP is an obvious candidate**
  - (Re-)Implemented in every package / language
  - Performance critical (and basically a matrix multiply)
  - Yielding wild differences in performance (different shapes, different packages)
- **Should be...**
  - Implemented once, used many times
  - Highly optimized
  - Optimized for all inputs (as opposed to the common inputs)
  - Compared with BLAS in term of efficiency
- **There's a long way to go (just for MTTKRP)**

One specific application: Gas Chromatography

Workflow

One specific application: Gas Chromatography

Workflow

...

4. **Fit model or rank $k \in [1, \ldots, 15]$**, if needed, add non-negativity constraints
   Tensor decompositions: PARAFAC — PARAFAC2 — TUCKER

One specific application: Gas Chromatography

Workflow

. . .

4. **Fit model or rank $k \in [1, \ldots, 15]$**, if needed, add non-negativity constraints
   Tensor decompositions: PARAFAC — PARAFAC2 — TUCKER

5. Determine whether or not one of the models is “right”

One specific application: Gas Chromatography

Workflow

. . .

4. **Fit model or rank $k \in [1, \ldots, 15]$**, if needed, add non-negativity constraints
   Tensor decompositions: PARAFAC — PARAFAC2 — TUCKER

5. Determine whether or not one of the models is "right"
   - : Determine which of the components represent chemical information
   - : Start over; add/change constraints, change model

One specific application: Gas Chromatography

Workflow

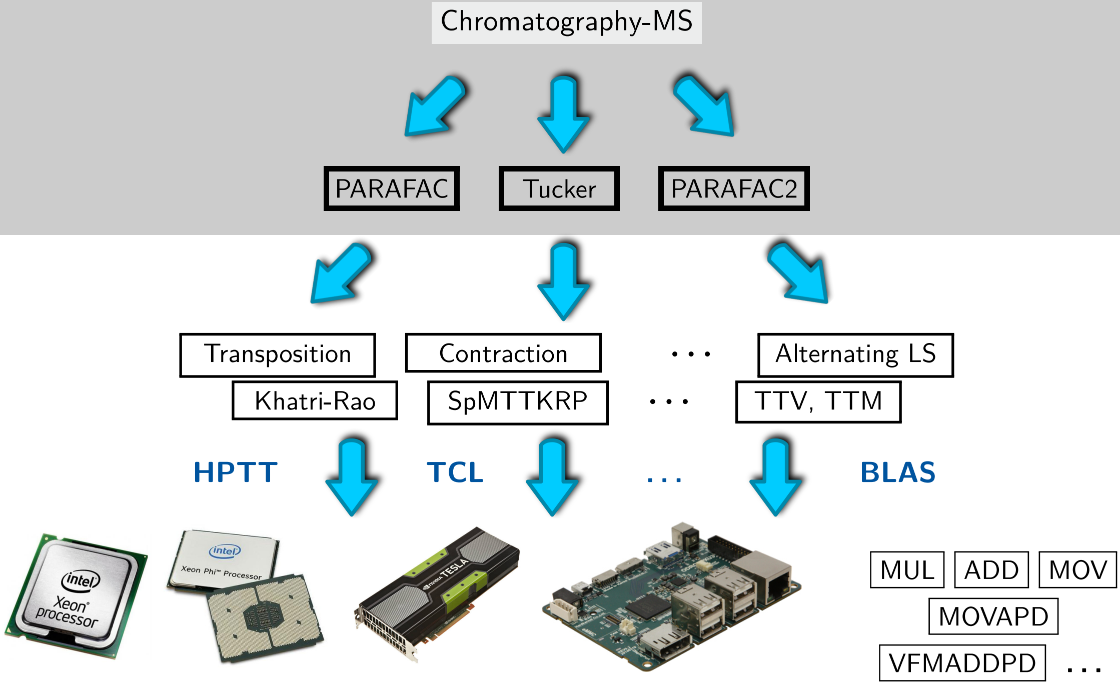
. . .

4. **Fit model or rank $\boldsymbol{k \in [1, \ldots, 15]}$**, if needed, add non-negativity constraints
   Tensor decompositions: PARAFAC — PARAFAC2 — TUCKER

5. Determine whether or not one of the models is "right"
   - : Determine which of the components represent chemical information
   - : Start over; add/change constraints, change model

Computation of each individual model: bandwidth bound!

Hence: *"Concurrent Alternating Least Squares for multiple simultaneous Canonical Polyadic Decompositions", with C. Psarras, L. Larsson. (Submitted).*


Chromatography-MS
PARAFAC
Tucker
PARAFAC2
Transposition
Contraction
. . .
Alternating LS
Khatri-Rao
SpMTTKRP
. . .
TTV, TTM
HPTT
TCL
. . .
BLAS
intel
Xeon processor
intel
Xeon Phi™ Processor
TESLA
MUL
ADD
MOV
MOVAPD
VFMADDPD
. . .

Chromatography-MS

PARAFAC | Tucker | PARAFAC2

Transposition | Contraction | . . . | Alternating LS

Khatri-Rao | SpMTTKRP | . . . | TTV, TTM

HPTT | **TCL** | . . . | **BLAS**

MUL | ADD | MOV

MOVAPD

VFMADDPD . . .

Comparing matrix and tensor efforts

	Matrices	Tensors

Comparing matrix and tensor efforts

	Matrices	Tensors
Driver	performance, HW	applications

Comparing matrix and tensor efforts

	Matrices	Tensors
Driver	performance, HW	applications
Community effort	BLAST/LAPACK/...	group by group

Comparing matrix and tensor efforts

	Matrices	Tensors
Driver	performance, HW	applications
Community effort	BLAST/LAPACK/...	group by group
Industry	wide support	not much

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”

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	—

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

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?

Thank you for the intvitation and for your attention!