The fragmented landscape of tensor computations

Paolo Bientinesi
Umeå Universitet
pauldj@cs.umu.se

October 26, 2022
4th Workshop on Scientific Computing in Sweden (SwedComp22)

About me

About me: Background in linear algebra

- Parallel eigensolvers
- Automatic generation of algorithms
- Stability analysis
- Performance modeling & prediction
- Compilers for matrix operations
- Applications
  - Covariance matrix
  - Genome-Wide Association Studies (GWAS)
- ...

About me: Background in linear algebra

- Parallel eigensolvers
- Automatic generation of algorithms
- Stability analysis
- Performance modeling & prediction
- Compilers for matrix operations
- Applications
  - Covariance matrix
  - Genome-Wide Association Studies (GWAS)
- ...

Everything relies on linear algebra libraries

Hierarchy of linear algebra libraries

PETSc, ...

ScaLAPACK, PLAPACK, ...

LAPACK

BLAS-1, BLAS-2, BLAS-3

More libraries ... (and many more iterative ones)

PETSc, Trilinos, ...

ScaLAPACK, PLAPACK, Elemental, ...

LAPACK, Plasma, SuperMatrix, Magma, ...

BLAS-1, BLAS-2, BLAS-3, ATLAS, BTO-BLAS, BLIS, ...

Applications

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

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

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

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

Applications $\longrightarrow$ libraries

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

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

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

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

$$\boxed{y := \alpha x + y} \quad \boxed{LU = A} \quad \cdots \quad \boxed{C := \alpha AB + \beta C}$$
$$\boxed{X := A^{-1} B} \quad \boxed{C := AB^T + BA^T + C} \quad \boxed{X := L^{-1} M L^{-T}} \quad \boxed{QR = A}$$

Applications $\longrightarrow$ libraries

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

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

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

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

?

$$\boxed{y := \alpha x + y} \quad \boxed{LU = A} \quad \cdots \quad \boxed{C := \alpha AB + \beta C}$$

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

Applications $\longrightarrow$ libraries

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

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

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

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

LINEAR ALGEBRA MAPPING PROBLEM

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

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

Applications $\longrightarrow$ libraries

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

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

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

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

LINEAR ALGEBRA MAPPING PROBLEM

"The Linear Algebra Mapping Problem. Current state of linear algebra languages and libraries", ACM TOMS, 2022
C. Psarras, H. Barthels, P. Bientinesi, [arXiv:1911.09421]

"Benchmarking the Linear Algebra Awareness of TensorFlow and PyTorch", iWAPT-22
A. Sankaran, N.A. Alashti, C. Psarras, P. Bientinesi, [arXiv:2202.09888]

"Linnea: Automatic Generation of Efficient Linear Algebra Programs", ACM TOMS, 2021
H. Barthels, C. Psarras, P. Bientinesi, [arXiv:1912.12924]

Tensors

"**Multi**-linear algebra mapping problem" ?

Tensors applications

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

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

Tensor Application #i ... Tensor Application #j

Tensors applications⟶ lin.alg.libs

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

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

Tensor Application #i ... Tensor Application #j

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

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

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

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

- Performance prediction E. Peise, P.B.

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

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

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

Tensor kernels

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

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

Tensor Application #i **CHEMOMETRICS** ... Tensor Application #j

Contractions | **???** | $A = U\Sigma W^\star$ | $\cdots$ | $C := \alpha AB + \beta C$ | **Tucker** | **???**

??? | **Transposition** | **???** | **CP** | **ALS** | **???** | **MTTKRP**

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

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

- Performance prediction E. Peise, P.B.

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

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

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

- High-performance kernels P. Springer, P.B.
  - *"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*

Survey

"**The landscape of software for tensor computations**"
C. Psarras, L. Karlsson, J. Li, P.B.

`https://arxiv.org/pdf/2103.13756.pdf`

DatM:	Data Manipulation
EWOps:	Element-Wise Operations
Con:	Contractions
SpecCon:	Specific Contractions
Decomp:	Decompositions

ID	Package Name	Functionality					Type	Platform	Language
		DatM	EWOps	SpecCon	Con	Decomp			
0	Acrotensor	–	–	✓	✓	–	D	C, G	C++
1	AdaTM	–	–	✓	–	✓	S	C	C
2	Boost.uBlas.Tensor	✓	✓	✓	✓	–	D	C	C++
3	BTAS	✓	✓	✓	✓	✓	nan	C	C++
4	COGENT	–	–	✓	✓	–	D	G	Python → CUDA
5	COMET	–	–	✓	✓	–	S	C	C++ → C++
6	CoTenGra	–	–	✓	✓	–	D	C, D, G	Python
7	CP-CALS	–	–	✓	–	✓	D	C, G	C++, Mat^{i}
8	CSTF	–	–	–	–	✓	S	D	Scala
9	CuTensor	✓	✓	✓	✓	–	D	G	C, CUDA
10	cuTT	✓	–	–	–	–	D	G	C++, CUDA
11	Cyclops	✓	✓	✓	✓	–	S, D	C, D, G	C++
12	D-Tucker	–	–	–	–	✓	D	C	Matlab
13	DFacTo	–	–	–	–	✓	S	C, D	C++
14	Eigen Tensor	✓	✓	✓	✓	–	D	C, G	C++
15	ExaTN	✓	✓	✓	✓	✓	D	C, D, G	C++, Py^{i}
16	Fastor	✓	✓	✓	✓	–	D	C	C++
17	FTensor	✓	✓	✓	✓	–	D	C	C++

ID	Package Name	Functionality					Type	Platform	Language
		DatM	EWOps	SpecCon	Con	Decomp			
18	Genten	–	–	–	–	✓	D, S	C, G	C++
19	GigaTensor	–	–	–	–	✓	S	C, D	Unknown
20	HPTT	✓	–	–	–	–	D	C	C++, Python[i], C[i]
21	ITensor	–	✓	✓	✓	✓	D, BS	C, G[x]	C++, Julia
22	libtensor	–	–	✓	✓	–	D, BS	C	C++
23	Ltensor	–	–	✓	✓	–	D	C	C++
24	MATLAB	✓	✓	✓	✓	–	D	C	Matlab
25	MultiArray	✓	–	–	–	–	D	C	C++
26	multiway	–	–	–	–	✓	D	C	R
27	N-way toolbox	–	–	–	–	✓	D	C	Matlab
28	NCON	–	–	✓	✓	–	D	C	Matlab
29	netcon	–	–	✓	✓	–	D	C	Matlab
30	NumPy	✓	✓	✓	✓	–	D	C	Python
31	Ocean	✓	✓	–	–	–	D	C, G	C, Py[i]
32	ParCube	–	–	–	–	✓	S	C	Matlab
33	ParTensor	–	–	–	–	✓	D	C, G	C++
34	ParTI!	✓	✓	✓	–	✓	S	C, G	C, CUDA, Mat[x]
35	PLANC	–	–	–	–	✓	D	C, D	C++

ID	Package Name	Functionality					Type	Platform	Language
		DatM	EWOps	SpecCon	Con	Decomp			
36	PLS toolbox	–	–	–	–	✓	D	C	Matlab
37	Pytensor	✓	✓	✓	✓	✓	D, S	C	Python
38	PyTorch	✓	✓	✓	✓	–	D, S	C, G	Python, C++, CUDA
39	quimb	–	–	✓	✓	–	D	C, D, G	Python
40	rTensor	✓	✓	✓	–	✓	D	C	R
41	rTensor (rand.)	–	–	–	–	✓	D	C	Python
42	scikit-tensor	✓	✓	✓	–	✓	D, S	C	Python
43	Scikit-TT	–	–	–	–	✓	D	C	Python
44	SPALS	–	–	–	–	✓	S	C	C++
45	SPARTan	–	–	–	–	✓	S	C	Matlab
46	SPLATT	–	–	✓	–	✓	S	C, D	C, C++, Oct, Mat
47	SuSMoST	–	–	–	–	✓	D	C	Python
48	T3F	✓	✓	–	–	✓	D	C, G	Python
49	TACO	✓	✓	✓	✓	–	D, S	C, G	C++, C++ $\rightarrow$ C++
50	TAL_SH	✓	✓	✓	✓	–	D	C, G	C, C++, Fort
51	TBlis	✓	✓	✓	✓	–	D	C	C++
52	TCCG	–	–	✓	✓	–	D	C	C++
53	TCL	–	–	✓	✓	–	D	C	C++, Python[i]

ID	Package Name	Functionality					Type	Platform	Language
		DatM	EWOps	SpecCon	Con	Decomp			
54	TDALAB	–	–	–	–	✓	D, S	C	Matlab, GUI
55	TeNPy	–	–	–	–	✓	D	C	Python
56	Tensor Fox	–	–	–	–	✓	D, S	C	Python, Matlab
57	Tensor package	–	–	–	–	✓	D	C	Matlab
58	Tensor Toolbox	✓	✓	✓	✓	✓	D, S	C	Matlab
59	tensor_decompositio	–	–	–	–	✓	D	C, D	Python
60	TensorBox	–	–	–	–	✓	D, S	C	Matlab
61	TensorD	–	–	–	–	✓	D	C, G	Python
62	TensorFlow	✓	✓	✓	✓	–	D, S	C, D, G	C++, Python
63	TensorLab	–	–	–	–	✓	D, S	C	Matlab
64	TensorLab+	–	–	–	–	✓	D, S	C	Matlab
65	TensorLy	✓	✓	✓	✓	✓	D	C, G	Python
66	TensorNetwork	–	–	✓	✓	–	D, S	C, G	Python
67	TensorOperations.	✓	✓	✓	✓	–	D	C, G	Julia
68	TensorTrace	–	–	✓	✓	–	D	C	GUI → Py, Jul, Mat
69	Three-Way	–	–	✓	✓	✓	D	C	R
70	TiledArray	✓	✓	✓	✓	–	D, BS	C, D	C++
71	tncontract	–	–	✓	✓	–	D	C	Python

ID	Package Name	Functionality					Type	Platform	Language
		DatM	EWOps	SpecCon	Con	Decomp			
72	TNR	–	–	–	–	✓	D	C	Matlab
73	TorchMPS	–	–	✓	✓	–	D	C	Python
74	TT-Toolbox	✓	✓	–	–	✓	D	C, D^x, G^x	Matlab, Python
75	TTC	✓	–	–	–	–	D	C	Python $\rightarrow$ C++
76	TTV	–	–	✓	–	–	D	C	C++
77	TVM	–	✓	–	–	–	D, S	C, G	Python
78	Uni10	✓	✓	✓	✓	–	D	C, G^x	C++
79	xerus	–	–	✓	✓	✓	D, S	C,	C++

That is A LOT of libraries!!!

That is A LOT of libraries!!!

- The survey is by no means complete.
  We still find libs, and new ones are released regularly.

That is A LOT of libraries!!!

- The survey is by no means complete.
  We still find libs, and new ones are released regularly.

- Even considering different languages, different data types (sparse vs. dense), different tensor orders, different types of parallelism, ...

  this is an enormous duplication of effort and functionality!!
  Also, many (severely) suboptimal implementations!

That is A LOT of libraries!!!

- The survey is by no means complete.
  We still find libs, and new ones are released regularly.

- Even considering different languages, different data types (sparse vs. dense), different tensor orders, different types of parallelism, ...

  this is an enormous duplication of effort and functionality!!
  Also, many (severely) suboptimal implementations!

- 1) Why?!
  2) Is there a way out?

Why?

- (At least) Two separate worlds

Why?

- (At least) Two separate worlds
  - Computational physics / chemistry — "Tensor" = Multi-linear operator

    Contractions = Generalization of matrix-matrix product

Why?

- (At least) Two separate worlds
  - Computational physics / chemistry — "Tensor" = Multi-linear operator

    Contractions = Generalization of matrix-matrix product
  - Data science — "Tensor" = Collection of data

    Decompositions = Generalization of matrix factorizations (with notable differences)

Why?

- (At least) Two separate worlds
  - Computational physics / chemistry — "Tensor" = Multi-linear operator

    Contractions = Generalization of matrix-matrix product
  - Data science — "Tensor" = Collection of data

    Decompositions = Generalization of matrix factorizations (with notable differences)
- Terminology and notation vary (and conflict) even within one world

  "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 [...]" — **Charles Van Loan**, *2009*

Why?

- (At least) Two separate worlds
  - Computational physics / chemistry — "Tensor" = Multi-linear operator

    Contractions = Generalization of matrix-matrix product
  - Data science — "Tensor" = Collection of data

    Decompositions = Generalization of matrix factorizations (with notable differences)
- Terminology and notation vary (and conflict) even within one world

  "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 [...]" — **Charles Van Loan**, *2009*
- Development: application driven — Very few software efforts cut across boundaries

Representative tensor operations / building blocks candidates

Data manipulation

- 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 tensor operations / building blocks candidates

Data manipulation

- 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
- ...

How to set the layers?
E.g., where does the "T-BLAS" end and the "T-LAPACK" begin?

Also . . . one vs. many problem instances

Also . . . one vs. many problem instances

Coupled-Cluster methods

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

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

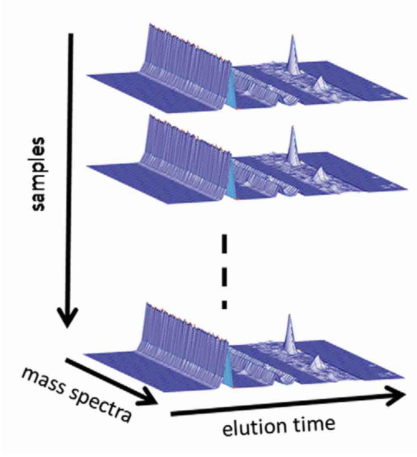
Finite Element 3D diffusion operator

```
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`

One vs. many problem instances (continued)

Chromatography



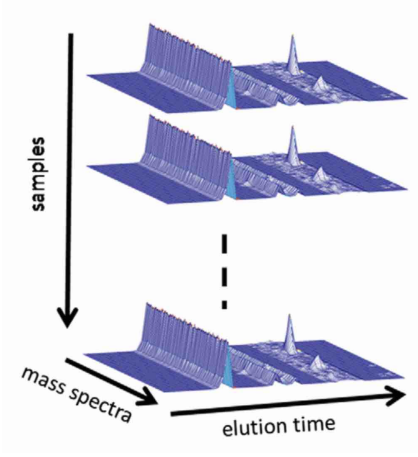
Rasmus Bro

Workflow

- ⋮
- **4. Fit model**: 1–20 components;
  PARAFAC — PARAFAC2 — TUCKER
- **5.** Determine whether or not one of the models is "right"
- **6.** Estimate the sensitivity of the model's parameters

One vs. many problem instances (continued)

Chromatography



Rasmus Bro

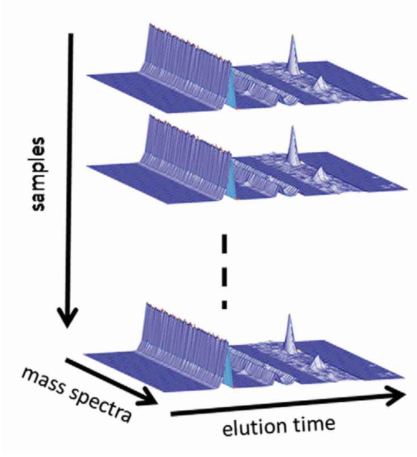
Workflow

- ⋮
- **4. Fit model**: 1–20 components;
  PARAFAC — PARAFAC2 — TUCKER
- **5.** Determine whether or not one of the models is "right"
- **6.** Estimate the sensitivity of the model's parameters

Huge opportunities for multi-instance & multi-problem optimizations

One vs. many problem instances (continued)

Chromatography



Rasmus Bro

Workflow

- ⋮
- **4. Fit model**: 1–20 components;
  PARAFAC — PARAFAC2 — TUCKER
- **5.** Determine whether or not one of the models is "right"
- **6.** Estimate the sensitivity of the model's parameters

Huge opportunities for multi-instance & multi-problem optimizations

Antithesis of decomposition into kernels. Here, **high-performance via grouping**!

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

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

- Performance prediction E. Peise, P.B.

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

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

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

- High-performance kernels P. Springer, P.B.
  - *"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, R. Bro, P.B.
  - *"Algorithm 1026: Concurrent Alternating Least Squares for multiple simultaneous Canonical Polyadic Decompositions", ACM TOMS 48(3), 2022*
  - *"Accelerating jackknife resampling for the Canonical Polyadic Decomposition", Frontiers AMS (8), 2022.*

Conclusions

- The landscape of tensor libraries is not as organized and developed as that of matrix libraries.
- Is there really a need for 80+ different tensor libraries?

Conclusions

- The landscape of tensor libraries is not as organized and developed as that of matrix libraries.
- Is there really a need for 80+ different tensor libraries?
- A well defined set of universal tensor kernels is missing.
- Many applications would benefit from those kernels. Others would not.

Conclusions

- The landscape of tensor libraries is not as organized and developed as that of matrix libraries.
- Is there really a need for 80+ different tensor libraries?
- A well defined set of universal tensor kernels is missing.
- Many applications would benefit from those kernels. Others would not.
- Despite the 80+ libs, the development of the tensor counterparts to BLAS & LAPACK is -still- very much an open problem.
- If you work on tensor computations, I would be happy to hear from you.

Conclusions

Thank you!

- The landscape of tensor libraries is not as organized and developed as that of matrix libraries.
- Is there really a need for 80+ different tensor libraries?
- A well defined set of universal tensor kernels is missing.
- Many applications would benefit from those kernels. Others would not.
- Despite the 80+ libs, the development of the tensor counterparts to BLAS & LAPACK is -still- very much an open problem.
- If you work on tensor computations, I would be happy to hear from you.