When $1 + 1 > 2$
The Power of Interdisciplinary Research

Paolo Bientinesi + Edoardo di Napoli

Aachen Institute for Advanced Study in
Computational Engineering Science
RWTH Aachen University

SLQM Opening workshop, JSC
April 4, 2017

Outline

Good old times

Present

Computational Physics Computational Chemistry Computational Biology . . .

Applied Mathematics / Numerics

Computer science / Algorithms

Implementation / HPC

Pros

- Separation of concerns
- Specialization
- High-performance
- Standardization
- Layering

But...

$$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{u} + \boldsymbol{\epsilon}$$

$$\min_x \|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2$$

LINEAR MIXED MODELS

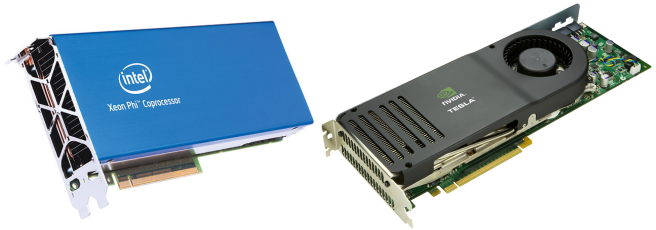
$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

LENNARD-JONES POTENTIAL

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

SCHRÖDINGER EQN.

$\vdots$



But...

$$y = X\beta + Zu + \epsilon$$

$$\min_x \|Ax - b\|^2 + \|\Gamma x\|^2$$

LINEAR MIXED MODELS

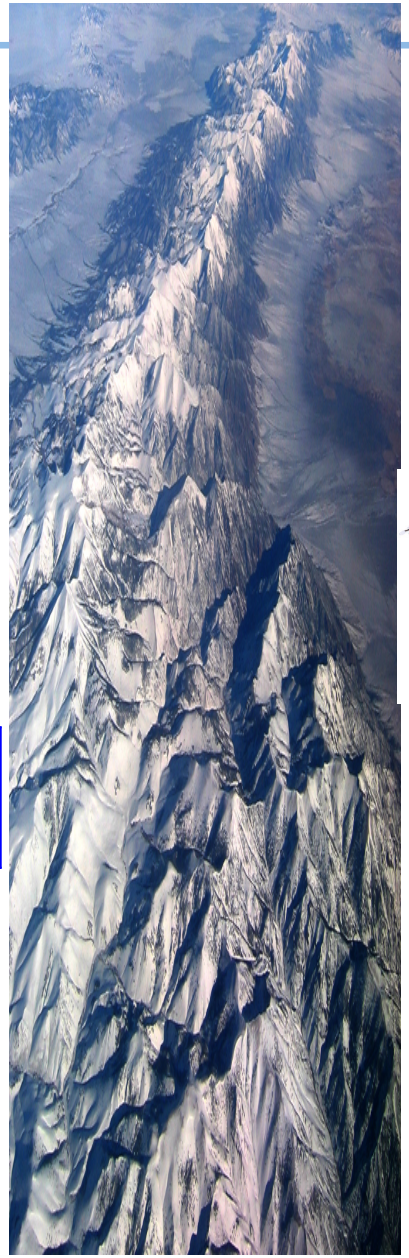
$$V_{LJ} = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$

LENNARD-JONES POTENTIAL

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

SCHRÖDINGER EQN.

$\vdots$



But...

$$y = X\beta + Zu + \epsilon$$

$$\min_x \|Ax - b\|^2 + \|\Gamma x\|^2$$

LINEAR MIXED MODELS

$$V_{LJ} = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$

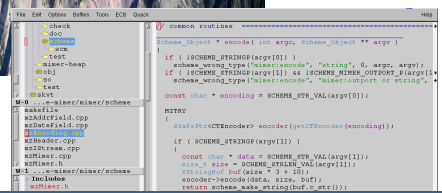
LENNARD-JONES POTENTIAL

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

SCHRÖDINGER EQN.

$\vdots$

HPAC



A successful collaboration

Computer Science

Duke University

Physics

U. of North Carolina

A successful collaboration

Computer Science
Duke University

Physics
U. of North Carolina

- Eigensolvers
- Tensor contractions
- Tensor calculations

Eigensolvers – compartmental approach

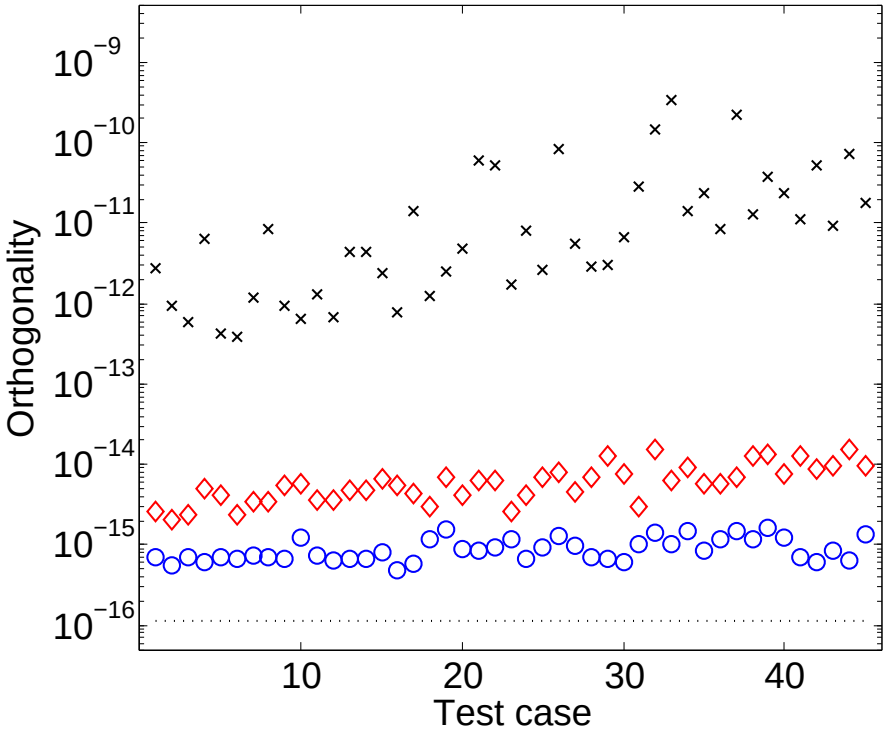
- Development in isolation
- “There is a matrix at the door”
- Expert in numerics, expert in performance
- Disconnect between HPC and users

Accuracy vs. Time

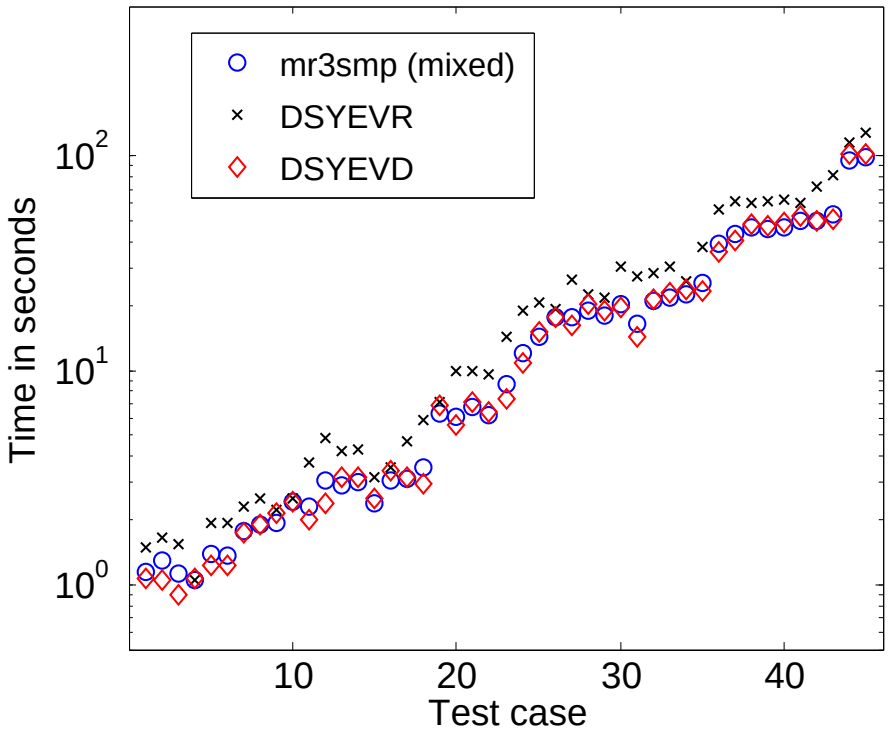
Mixed-precision MR3-based symmetric eigensolver

application matrices, size $\in [1.000, \ldots, 8.000]$

Orthogonality



Execution time: 32 threads



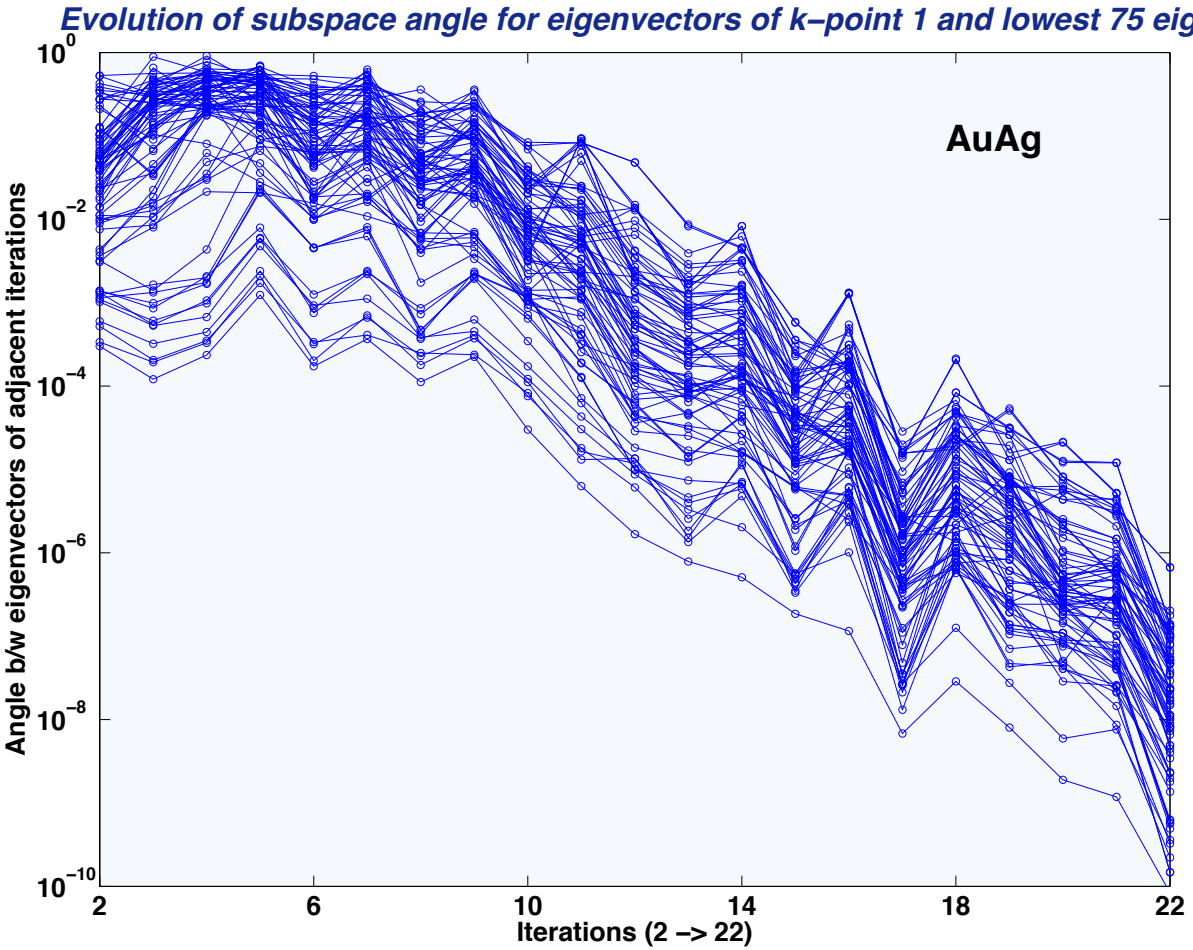
"Improved Accuracy and Parallelism for MRRR-based Eigensolvers",
M. Petschow, P. Bientinesi. SIAM Journal of Scientific Computing, 36(2), 240–263, 2014.

Eigensolvers – collaborative approach

- Eigensolver as part of a **process**
- Where are the matrices coming from?
- What are the known properties?
- What are the eigenvectors used for?

Analysis

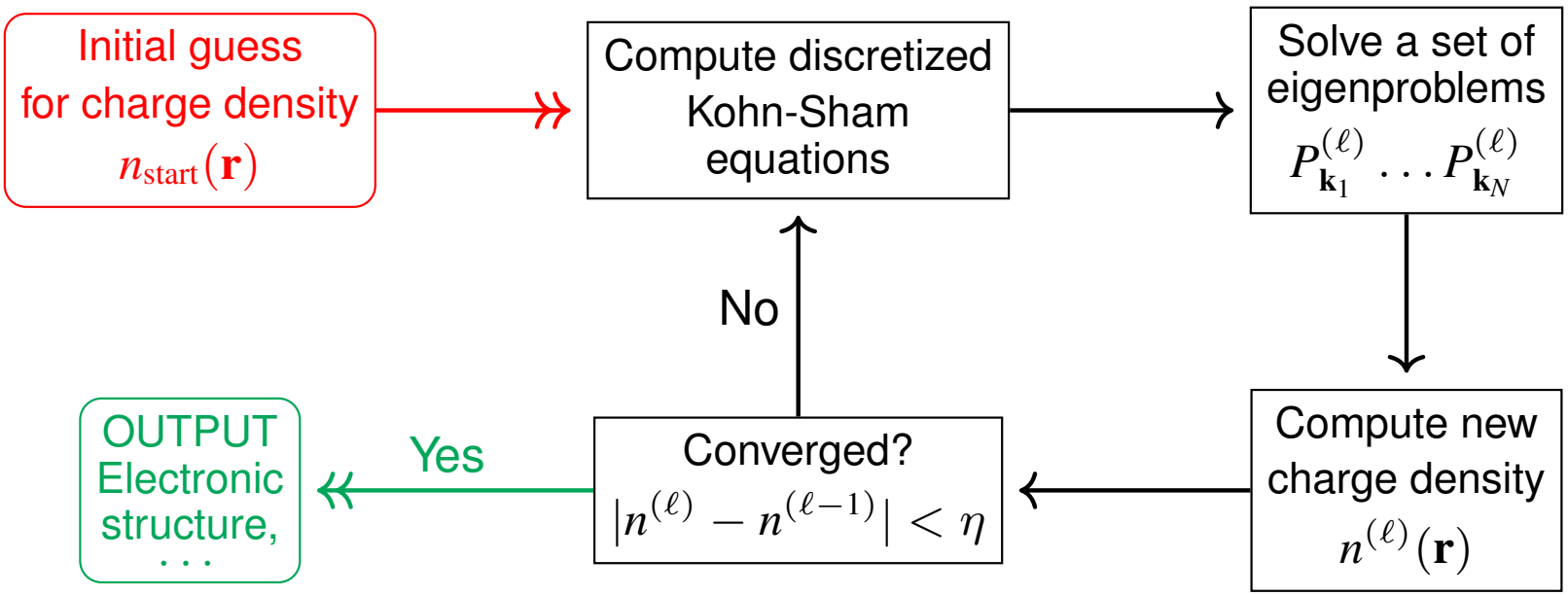
Eigenvectors



"Correlations in Sequences of Generalized Eigenproblems Arising in Density Functional Theory"
E. Di Napoli, S. Bluegel, P. Bientinesi Comp. Phys. Comm., 183(8), 1674–1682, 2012.

A practical Density Functional Theory example

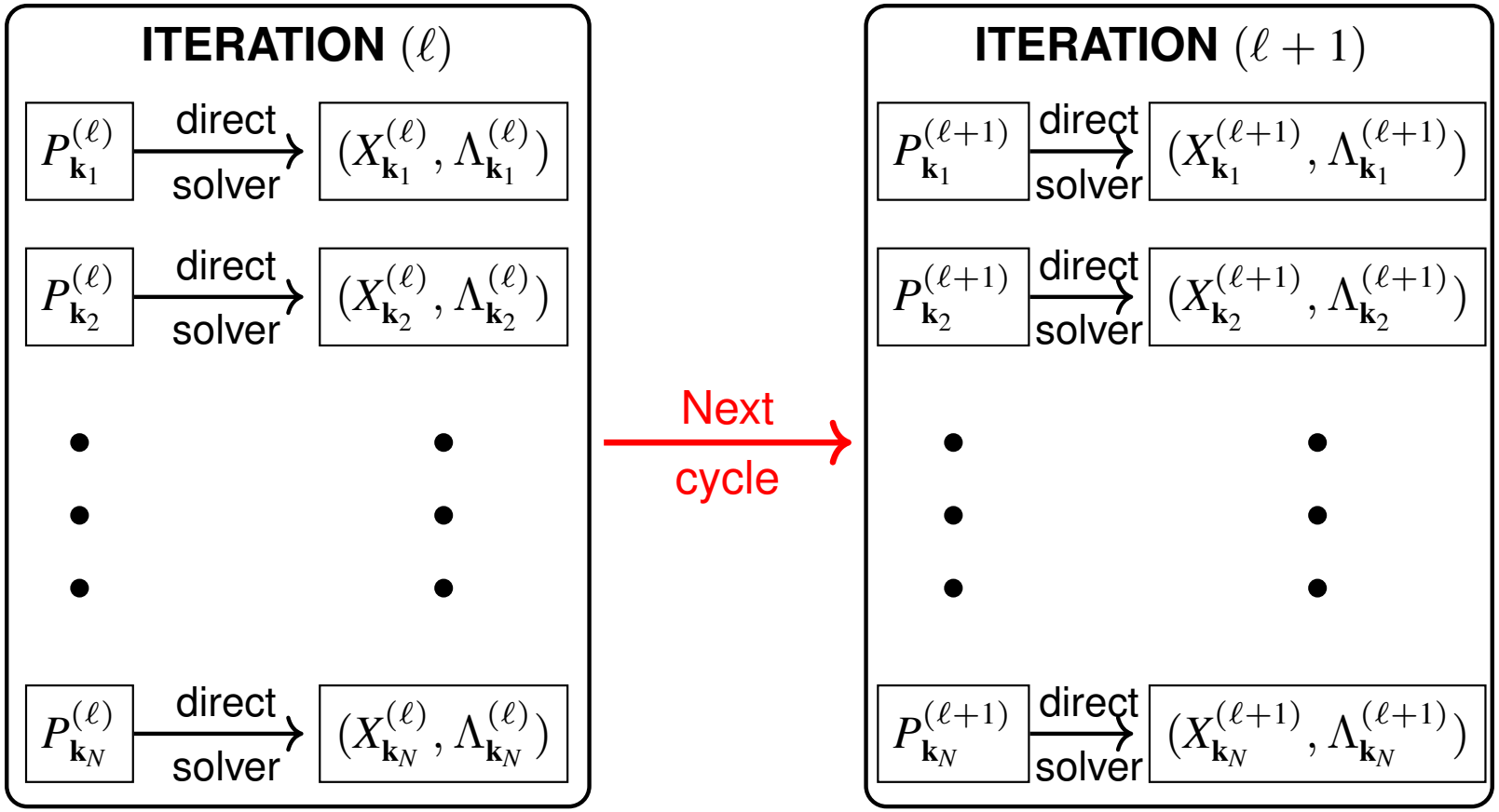
Sequences of eigenvalue problems arising in FLAPW



$$P^{(\ell)}_{\mathbf{k}_i}: \quad H^{(\ell)}_{\mathbf{k}_i} \, x^{(\ell)}_j = \lambda^{(\ell)}_j \, S^{(\ell)}_{\mathbf{k}_i} \, x^{(\ell)}_j \qquad j = 1, \ldots, \text{nev}.$$

Sequences of Eigenproblems

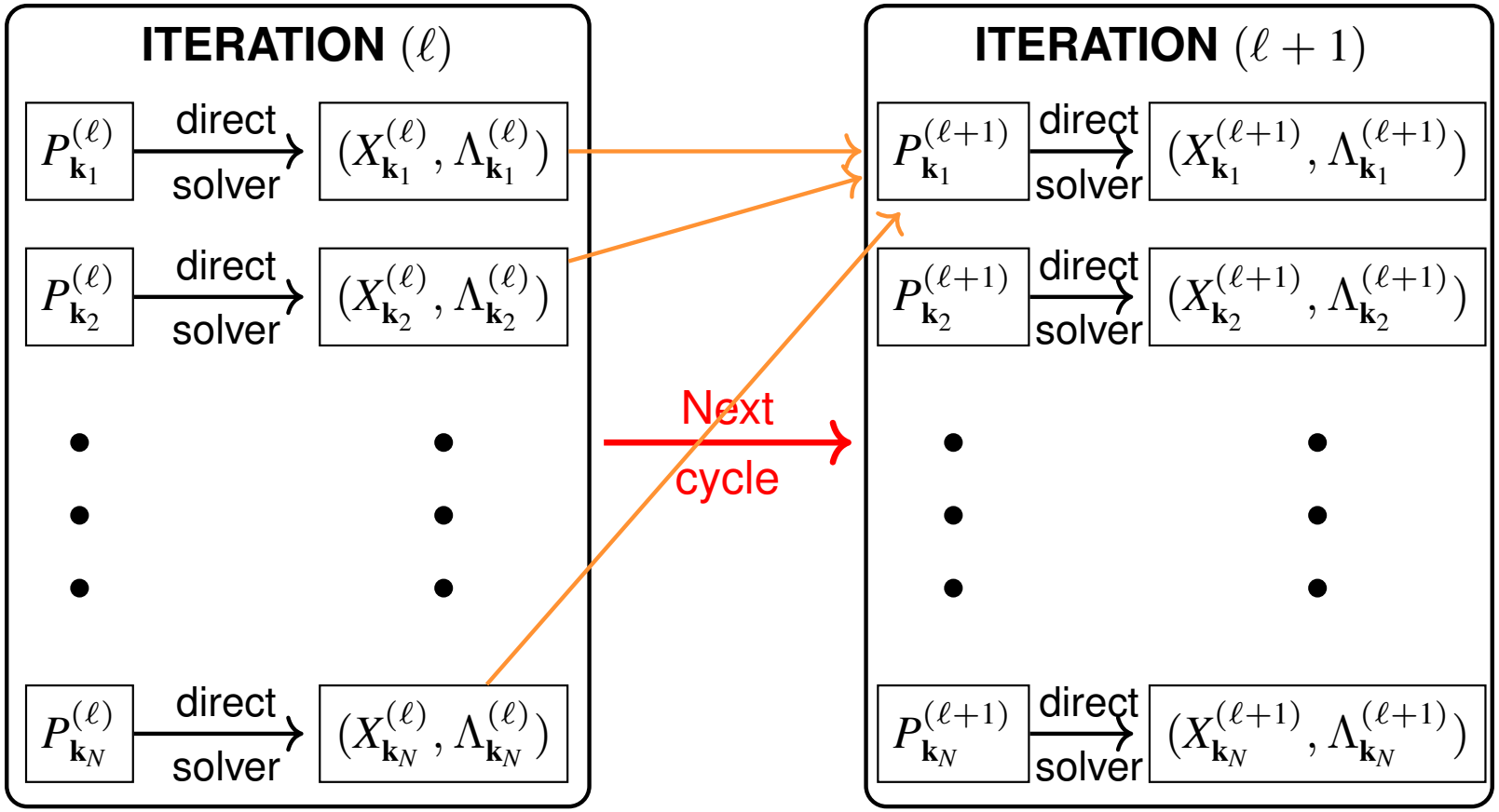
Adjacent iteration cycles



$$X \equiv \{x_1, \ldots, x_{\text{nev}}\} \qquad \Lambda \equiv \text{diag}(\lambda_1, \ldots, \lambda_{\text{nev}})$$

Sequences of Eigenproblems

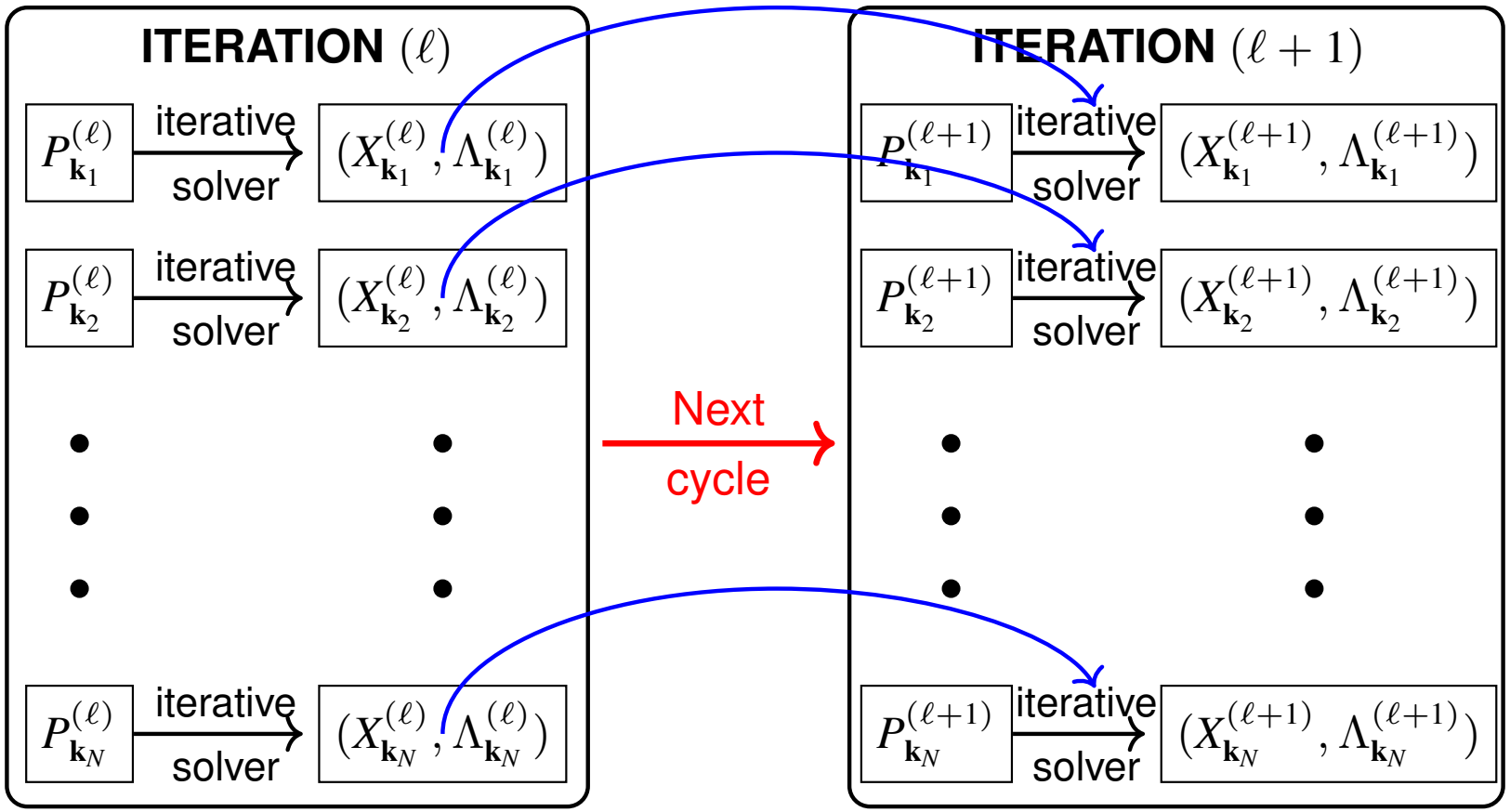
Adjacent iteration cycles



$$X \equiv \{x_1, \ldots, x_{\text{nev}}\} \qquad \Lambda \equiv \text{diag}(\lambda_1, \ldots, \lambda_{\text{nev}})$$

An alternative solving strategy: the ChASE algorithm

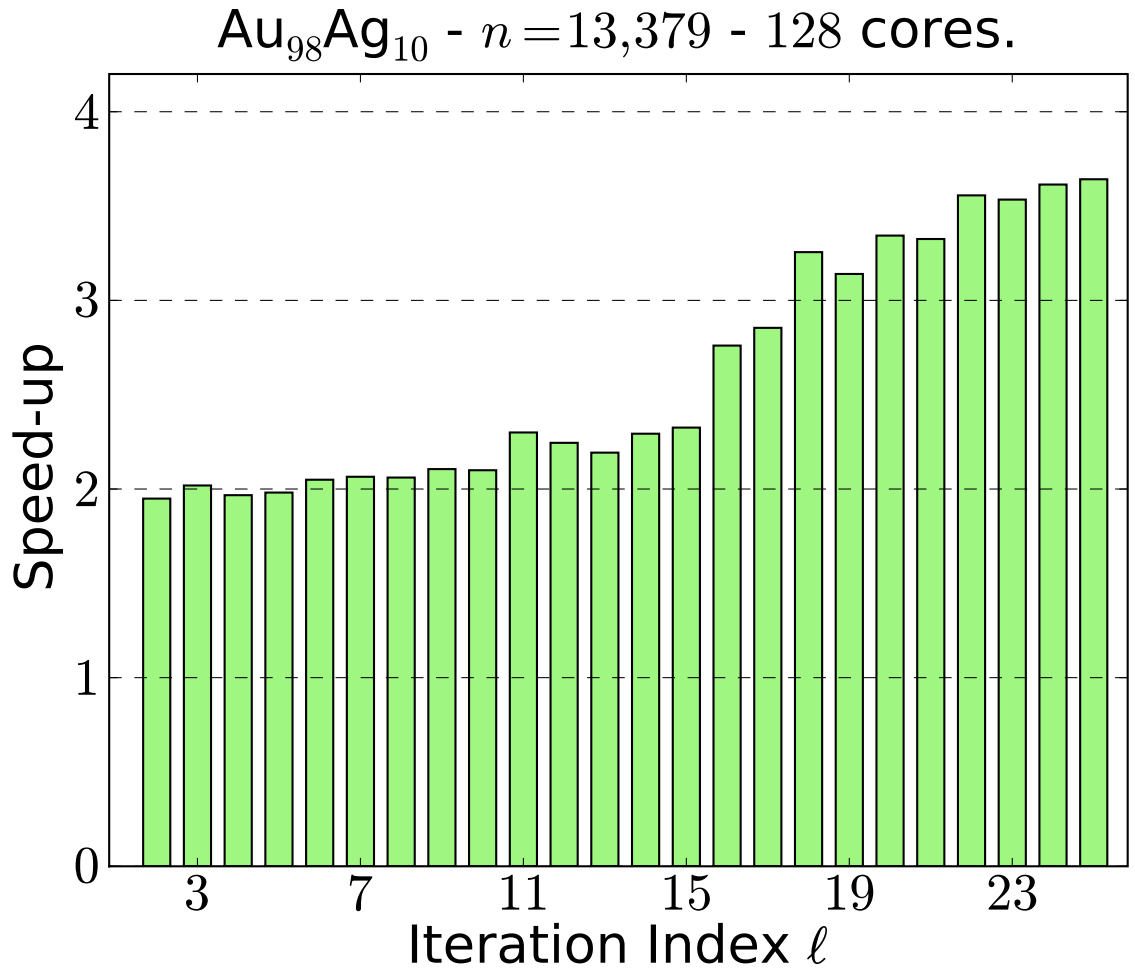
Adjacent cycles



$$X \equiv \{x_1, \ldots, x_{\text{nev}}\} \qquad \Lambda \equiv \text{diag}(\lambda_1, \ldots, \lambda_{\text{nev}})$$

Speed-up

$$\text{Speed-up} = \frac{\text{CPU time (input random vectors)}}{\text{CPU time (input approximate eigenvectors)}}$$



The core of the algorithm: Chebyshev filter

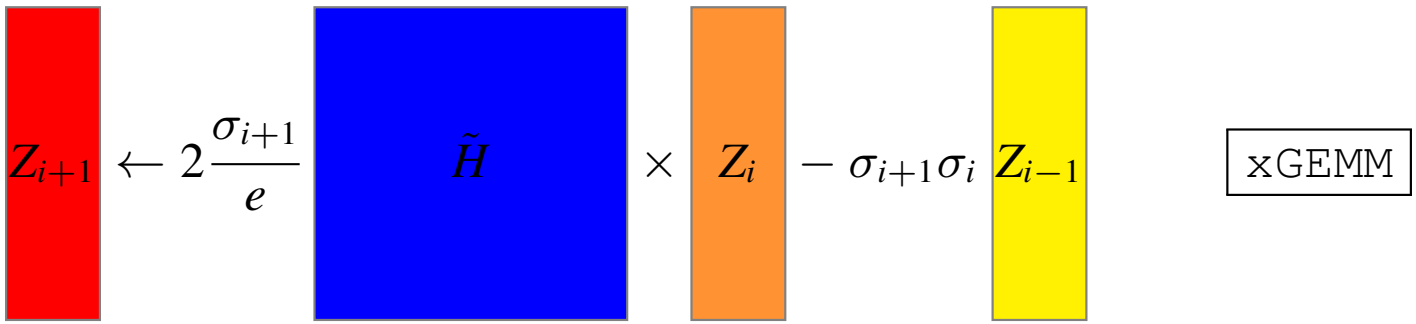
In practice

Three-terms recurrence relation

$$C_{m+1}(t) = 2xC_m(t) - C_{m-1}(t); \qquad m \in \mathbb{N}, \quad C_0(t) = 1, \quad C_1(t) = x$$

$$Z_m \doteq p_m(\tilde{H})\, Z_0 \quad \text{with} \quad \tilde{H} = H - cI_n$$

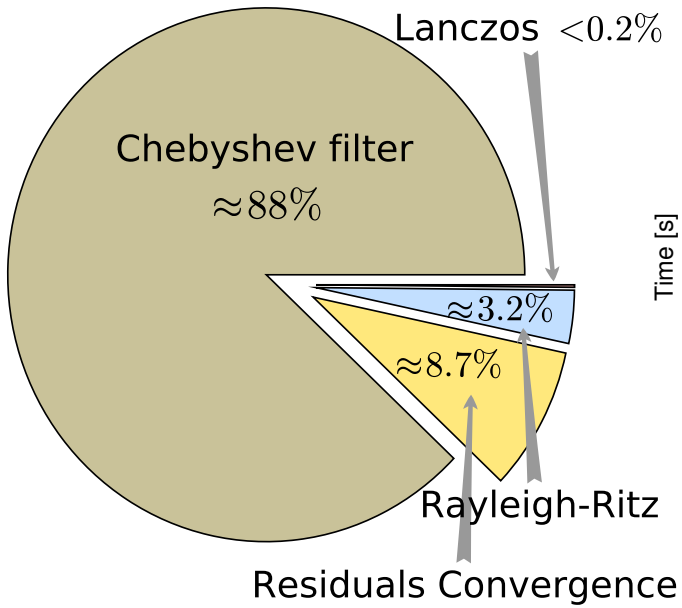
FOR: $i = 1 \rightarrow \text{DEG} - 1$

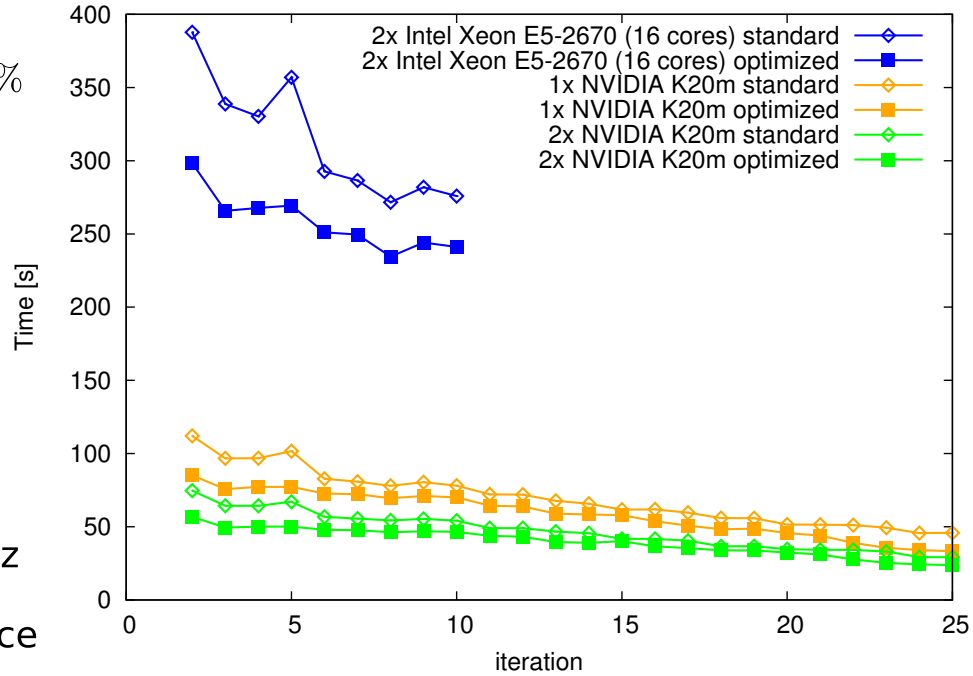


END FOR.

Efficiency and portability

$Au_{98}Ag_{10}$ - $n = 8{,}970$ - 32 cores.





"An optimized and scalable eigensolver for sequences of eigenvalue problems"
M. Berljafa, D. Wortmann and E. Di Napoli. Conc. Comp.: Pract. Exper. 2015; 27:905–922

Tensors

Tensors

- Hot, emerging topic in HPC
- Lost in translation: tensor vs. multi-dimensional array
- Tensor contraction = generalization of matrix product
  $\Rightarrow$ can I use GEMM? Yes/no? When?

Taxonomy

$$V_{h_1 h_2 \ldots} := S_{i_1 i_2 \ldots} T_{j_1 j_2 \ldots}$$

Definition: $\Delta(X) =$ # of free indices of X

Class 1: $\Delta(S) = 0 \ \wedge \ \Delta(T) = 0$

✗ BLAS3 ✗ BLAS2 ✓BLAS1

Class 2: $\Delta(S) \geq 1 \ \wedge \ \Delta(T) = 0$ or $\Delta(S) = 0 \ \wedge \ \Delta(T) \geq 1$

✗ BLAS3 ✓BLAS2 (+ transp) ✓BLAS1

Class 3: $\Delta(S) \geq 1 \ \wedge \ \Delta(T) \geq 1$

✓BLAS3 (+ transp) ✓BLAS2 ✓BLAS1

"Towards an Efficient Use of the BLAS Library for Multilinear Tensor Contractions",
E. Di Napoli, D. Fabregat-Traver, G. Quintana-Orti, P. Bientinesi, Applied Mathematics and Computation, 235, 454–468, 2014.

Tensors (continued)

- Hot, emerging topic in HPC
- Lost in translation: tensor vs. multi-dimensional array
- Tensor contraction = generalization of matrix product
  $\Rightarrow$ can I use GEMM? Yes/no? When?
- Contractions implemented as explicit loops

  Pros: Symmetries, simplifications

  Cons: Efficiency, parallelism, portability!!

Tensors (continued)

- Hot, emerging topic in HPC
- Lost in translation: tensor vs. multi-dimensional array
- Tensor contraction = generalization of matrix product
  $\Rightarrow$ can I use GEMM? Yes/no? When?
- Contractions implemented as explicit loops

  Pros: Symmetries, simplifications

  Cons: Efficiency, parallelism, portability!!
- Our approach:

  - Expose properties

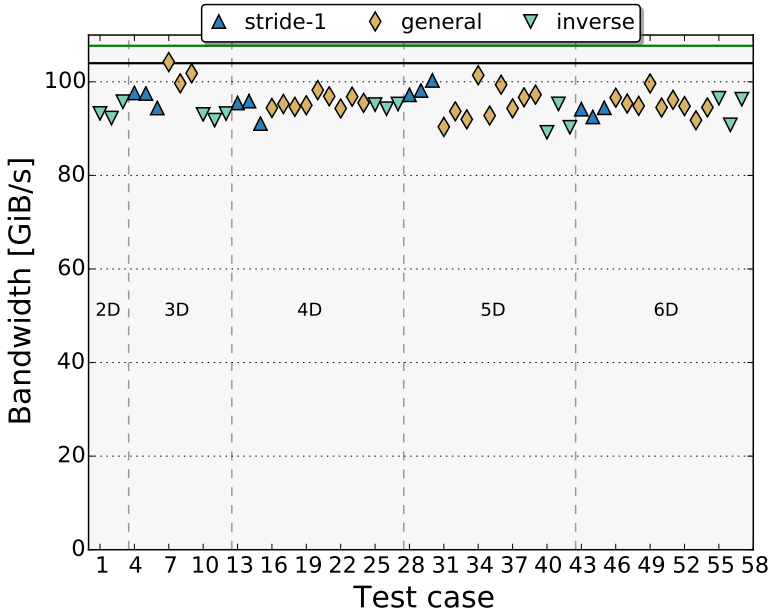
  - Map to linear algebra libraries! (and develop tensor libraries)

  Pros: Inherited efficiency, parallelism, portability

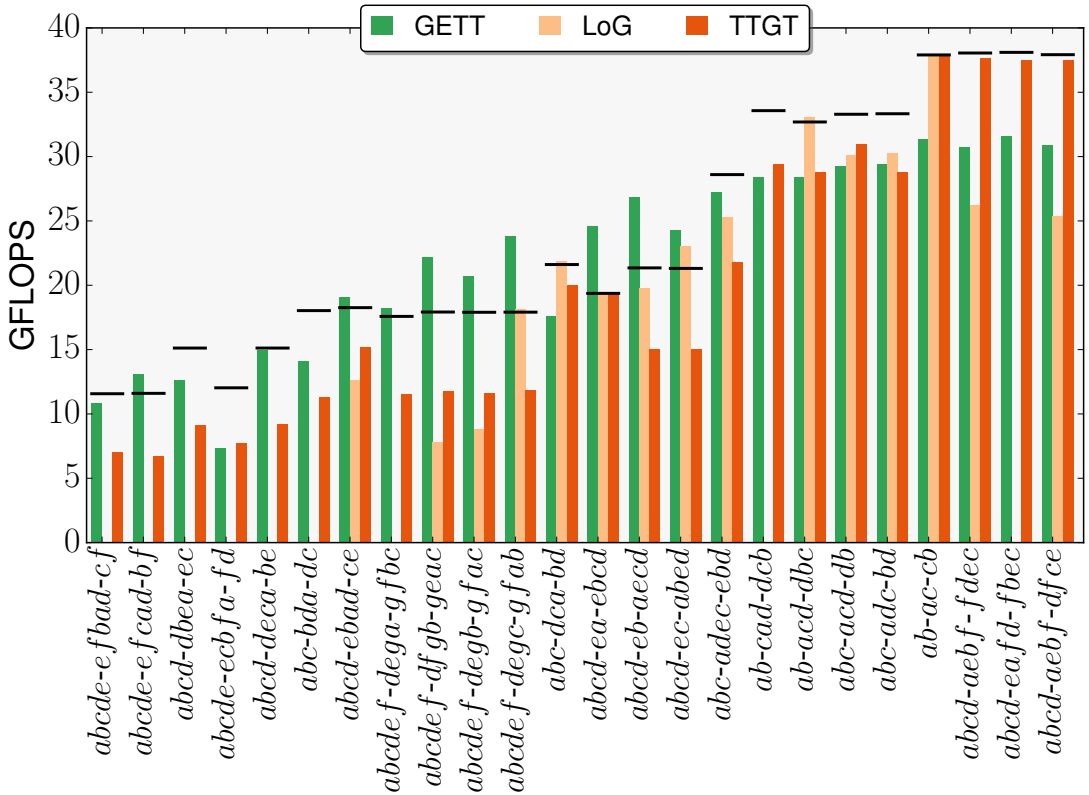
  Cons: Not all symmetries are exploited

Tensor libraries

Tensor transpositions



Tensor contractions



"TTC: A high-performance Compiler for Tensor Transpositions", P. Springer, J.R. Hammond, P. Bientinesi, ACM TOMS, 2017.

"Design of a high-performance GEMM-like Tensor-Tensor Multiplication", P. Springer, P. Bientinesi, ACM TOMS, 2017.

Hamiltonian and Overlap matrices in FLAPW methods

Operatorial form

$$(H)_{G',G} = \sum_a \iint \varphi^*_{G'}(\mathbf{r}) \hat{H}_{\mathrm{KS}} \varphi_G(\mathbf{r}) \mathrm{d}\mathbf{r}, \quad (S)_{G',G} = \sum_a \iint \varphi^*_{G'}(\mathbf{r}) \varphi_G(\mathbf{r}) \mathrm{d}\mathbf{r}.$$

Entrywise form

$$(S)_{G',G} = \sum_a \sum_{L=(l,m)} \left(A_L^{a,G'}\right)^* A_L^{a,G} + \left(B_L^{a,G'}\right)^* B_L^{a,G} \left\|\dot{u}_{l,a}\right\|^2$$

$$\begin{aligned}(H)_{G',G} = \sum_a \sum_{L',L} &\left(\left(A_{L'}^{a,G'}\right)^* T^{[AA]}_{L',L;a} A_L^{a,G}\right) + \left(\left(A_{L'}^{a,G'}\right)^* T^{[AB]}_{L',L;a} B_L^{a,G}\right) \\ &+ \left(\left(B_{L'}^{a,G'}\right)^* T^{[BA]}_{L',L;a} A_L^{a,G}\right) + \left(\left(B_{L'}^{a,G'}\right)^* T^{[BB]}_{L',L;a} B_L^{a,G}\right).\end{aligned}$$

Hamiltonian and Overlap matrices

Tensor form

$$H = \sum_{a=1}^{N_A} \underbrace{A_a^H T_a^{[AA]} A_a}_{H_{AA}} + \underbrace{A_a^H T_a^{[AB]} B_a + B_a^H T_a^{[BA]} A_a + B_a^H T_a^{[BB]} B_a}_{H_{AB+BA+BB}}$$

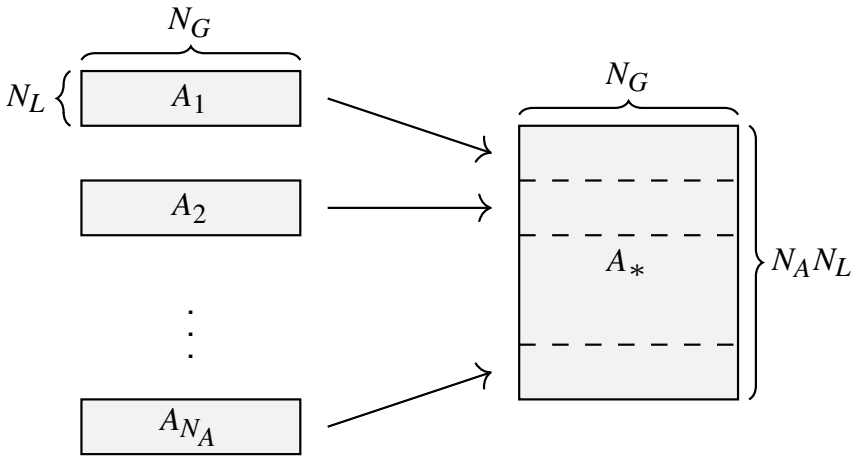
$$S = \underbrace{\sum_{a=1}^{N_A} A_a^H A_a}_{S_{AA}} + \underbrace{\sum_{a=1}^{N_A} B_a^H \dot{U}_a^H \dot{U}_a B_a}_{S_{BB}}$$

Constructing S_{AA}

An example of memory layout re-structuring

$$S_{AA} = \sum_{a=1}^{N_A} A_a^H A_a.$$

1: **for** $a := 1 \rightarrow N_A$ **do**
2: $\quad S_{AA} = A_a^H A_a$ ▷ (`zherk`: $4N_L N_G^2$ Flops)
3: **end for**



1: $S_{AA} = A_\star^H A_\star$ ▷ (`zherk`: $4N_A N_L N_G^2$ Flops)

Constructing $H_{AB+BA+BB}$

An example of algorithm re-structuring

$$
\begin{aligned}
H_{AB+BA+BB} &= \sum_{a=1}^{N_A} B_a^H (T_a^{[BA]} A_a) + (A_a^H T_a^{[AB]}) B_a + \\
&\qquad \frac{1}{2} B_a^H (T_a^{[BB]} B_a) + \frac{1}{2} (B_a^H T_a^{[BB]}) B_a \\
&= \sum_{a=1}^{N_A} B_a^H (T_a^{[BA]} A_a + \frac{1}{2} T_a^{[BB]} B_a) + \\
&\qquad (A_a^H T_a^{[AB]} + \frac{1}{2} B_a^H T_a^{[BB]}) B_a
\end{aligned}
$$

1: **for** $a := 1 \rightarrow N_A$ **do**
2: $\quad Z_a = T_a^{[BA]} A_a$ ▷ (`zgemm`: $8 N_L^2 N_G$ Flops)
3: $\quad Z_a = Z_a + \frac{1}{2} T_a^{[BB]} B_a$ ▷ (`zhemm`: $8 N_L^2 N_G$ Flops)
4: $\quad$ Stack Z_a to $Z_\star$ and B_a to $B_\star$
5: **end for**
6: $H = Z_\star^H B_\star + B_\star^H Z_\star$ ▷ (`zher2k`: $8 N_A N_L N_G^2$ Flops)

Experimental multi-core results

	NaCl ($\mathbf{K}_{max} = 4.0$)					
	IvyBridge			Haswell		
	HSDLA	FLEUR	×	HSDLA	FLEUR	×
1 core	31.53	48.31	**1.53**	19.00	47.41	**2.50**
2 cores	16.10	24.58	**1.53**	9.98	24.95	**2.50**
1 CPU	3.90	6.21	**1.59**	2.25	5.00	**2.22**
2 CPUs	2.61	5.20	**1.99**	1.93	4.03	**2.09**
	TiO_2 ($\mathbf{K}_{max} = 3.6$)					
	IvyBridge			Haswell		
	HSDLA	FLEUR	×	HSDLA	FLEUR	×
1 core	175.53	256.15	**1.46**	106.56	259.91	**2.44**
2 cores	86.68	127.90	**1.48**	53.48	131.21	**2.45**
1 CPU	19.63	29.35	**1.50**	10.63	25.95	**2.44**
2 CPUs	12.25	21.50	**1.76**	7.55	16.76	**2.22**

Table: Scalability of HSDLA and FLEUR: execution times in minutes on Haswell (12 cores / CPU) and IvyBridge (10 cores / CPU); speedups of HSDLA over FLEUR in **bold**.

"High-performance generation of the Hamiltonian and Overlap matrices in FLAPW methods."
E. Di Napoli, E. Peise, M. Hrywniak and P. Bientinesi. Comp. Phys. Comm. 211 (2017) 61–72

Porting to heterogeneous architectures

Back-of-the-envelope analysis

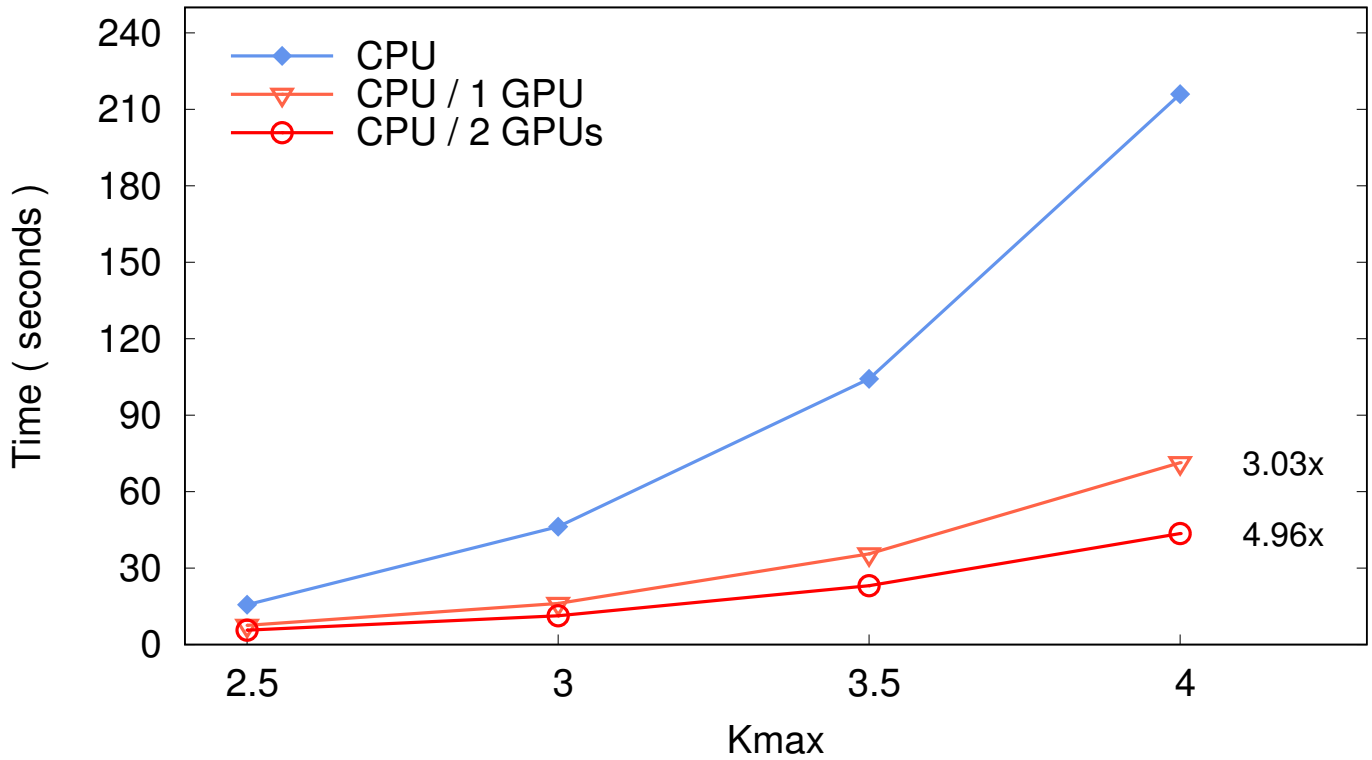
- 5 lines of the algorithm constitute 97% of flops
- Correspond to BLAS-3 operations (`gemm, herk, her2k`)
- High arithmetic intensity and should fit GPUs well

- First step: offload these routine calls
- All BLAS kernels. Which library?
  - cuBLAS
  - cuBLAS-XT
  - MAGMA
  - BLASX
- 3x wrappers (zgemm, zherk, zher2k)
- Init and cleanup of cuda runtime and devices
- Allocate data in page-locked memory

Only around 100 lines of additional code

Experimental multi-GPU results

Test case: AuAg $(N_A = 108, N_L = 121, N_G = [3275 - 13379])$



- CPU: E5-2650, 2 x 8core, 2.0GHz, 64GBs RAM (Sandy Bridge)
- 2 Nvidia Tesla K20X
- Peak performance: 256 GFs/s + 2 x 1.3 TFs/s

"Hybrid CPU-GPU generation of the Hamiltonian and Overlap matrices in FLAPW methods."

D. Fabregat-Traver, D. Davidović, M. Höhnerbach, E. Di Napoli. LNCS, Vol. 10164, pp. 200-211 .

Conclusions

Interdisciplinary research

- Close collaboration & interactions
- Common language!
- New questions, new challenges
- Lots of opportunities

Conclusions

Interdisciplinary research

- Close collaboration & interactions
- Common language!
- New questions, new challenges
- Lots of opportunities
- . . . funding schemes . . .
- Education

Thank you for your attention!