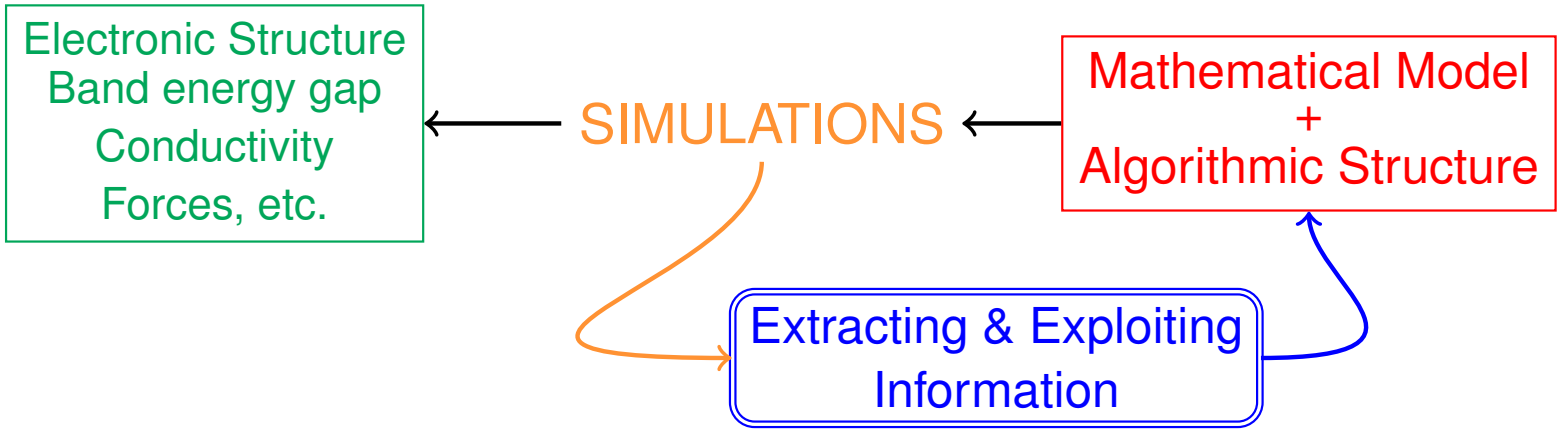
JÜLICH
FORSCHUNGSZENTRUM

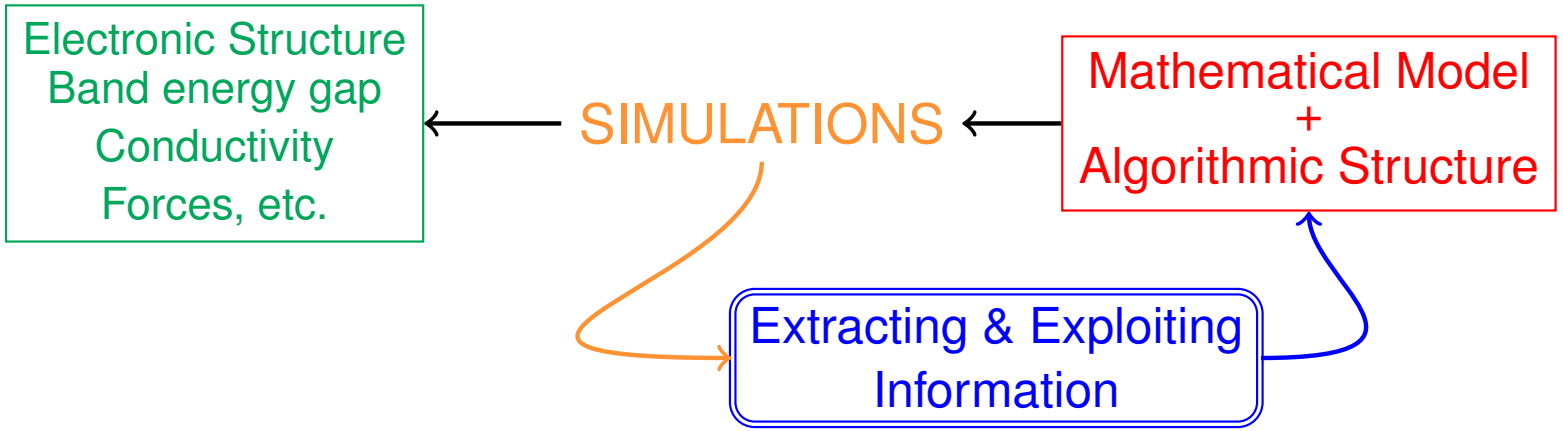
A Parallel and Scalable Iterative Solver for Sequences of Dense Eigenproblems Arising in FLAPW

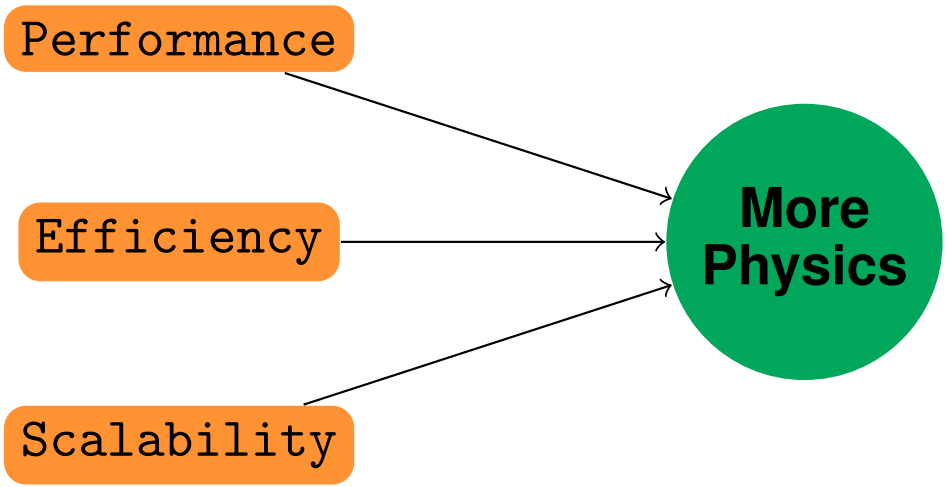
PPAM 2013 Warsaw, Poland, Sept. 10th | M. Berljafa and **E. Di Napoli**

Motivation and Goals



Motivation and Goals





Outline

Sequences of generalized eigenproblems in all-electron computations

The algorithm: Chebyshev Filtered Sub-space Iteration method (**ChFSI**)

ChFSI parallelization and numerical tests

Outline

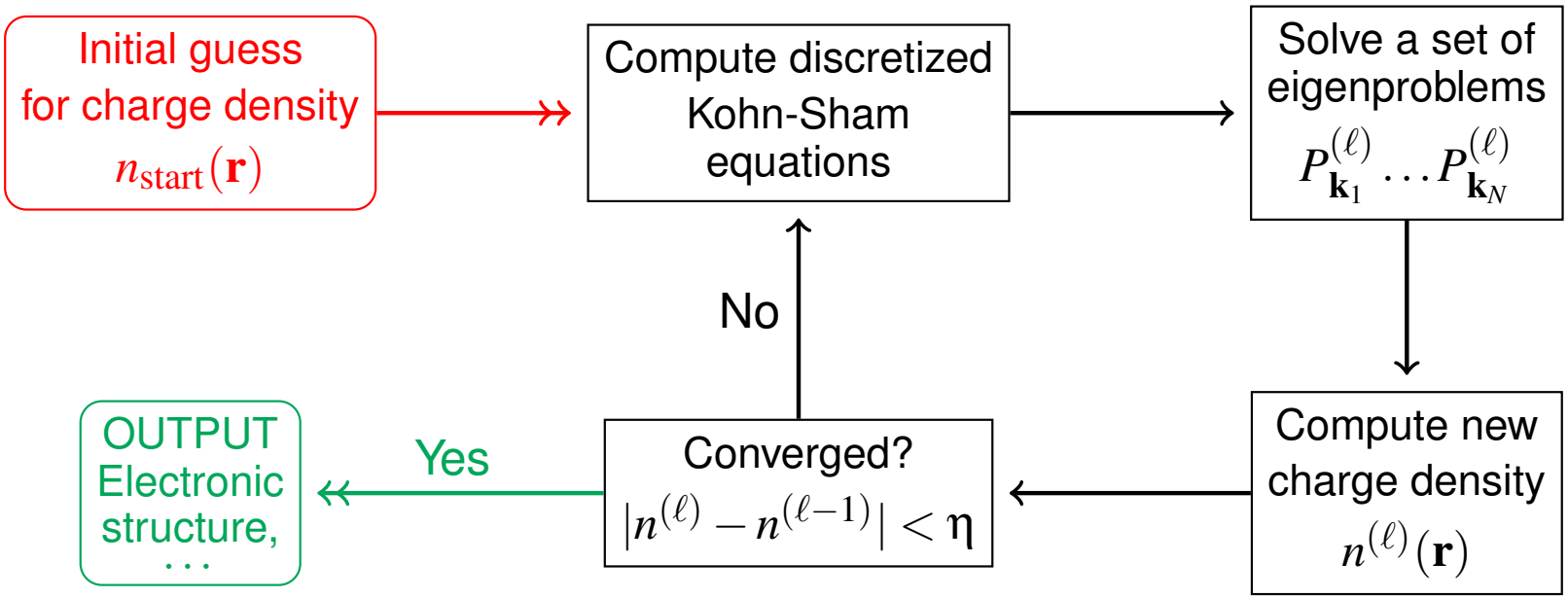
Sequences of generalized eigenproblems in all-electron computations

The algorithm: Chebyshev Filtered Sub-space Iteration method (**ChFSI**)

ChFSI parallelization and numerical tests

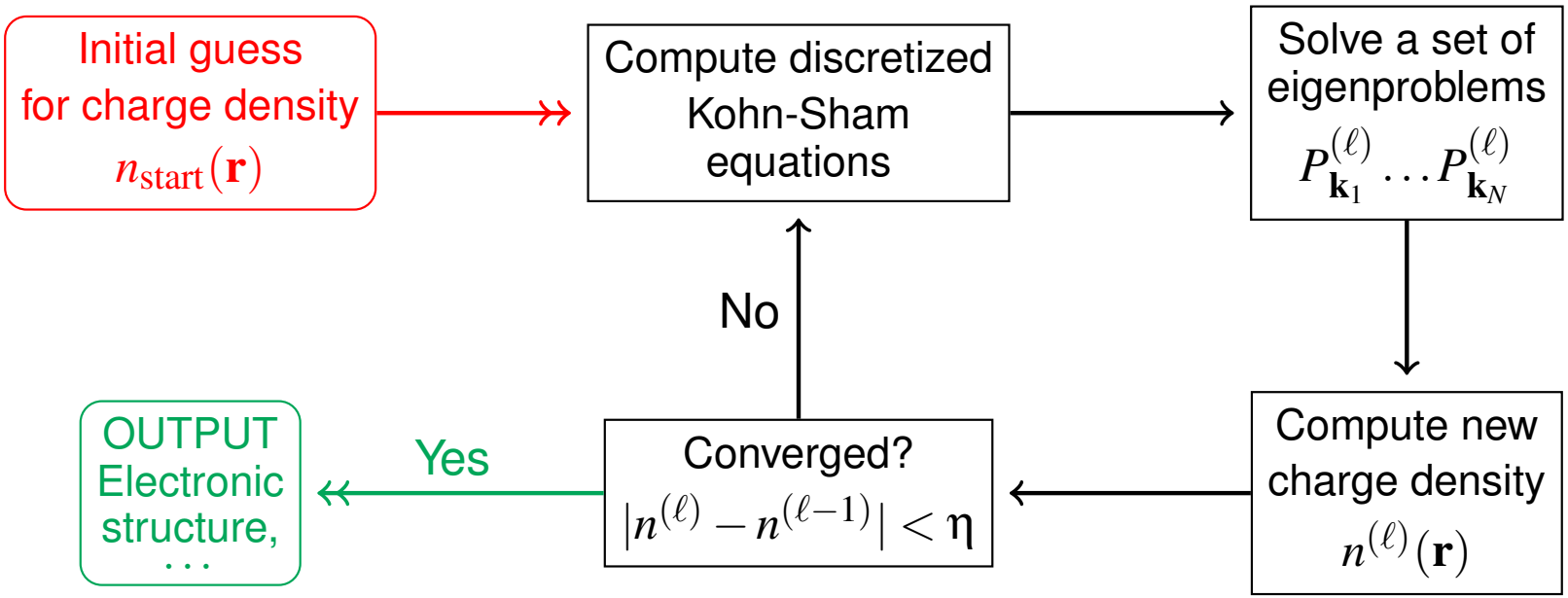
Density Functional Theory scheme

Self-consistent cycle



Density Functional Theory scheme

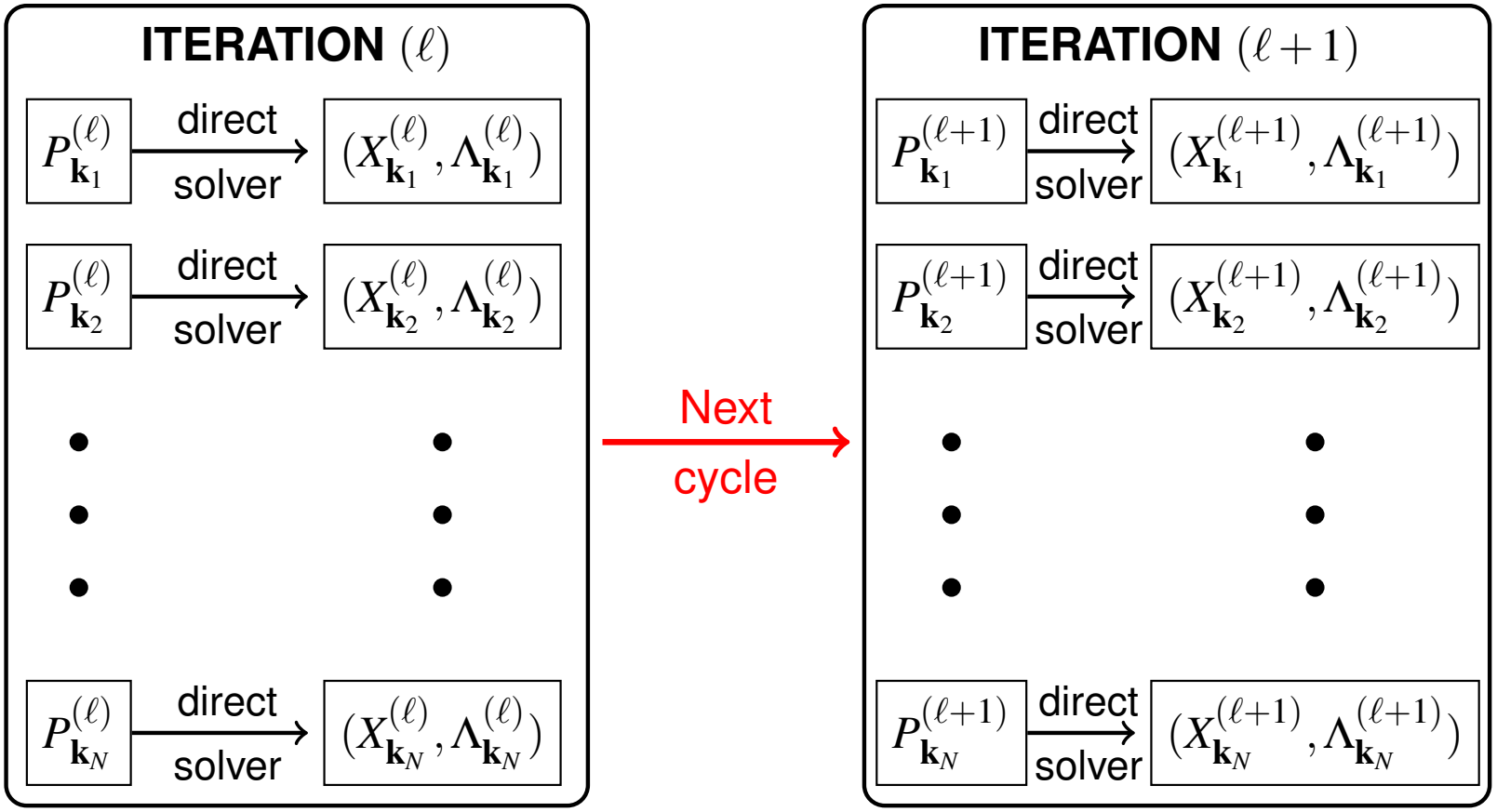
Self-consistent cycle



1. every $P^{(\ell)}_{\mathbf{k}} : A^{(\ell)}_{\mathbf{k}} x = B^{(\ell)}_{\mathbf{k}} \lambda x$ is a generalized eigenvalue problem;
2. A and B are **DENSE** and hermitian (B is also pos. def.);
3. $P_{\mathbf{k}}$s with different **k** index have different size and are independent from each other.
4. $\mathbf{k} = 1 : 10 - 100 \quad ; \quad \ell = 1 : 20 - 50$

Density Functional Theory scheme

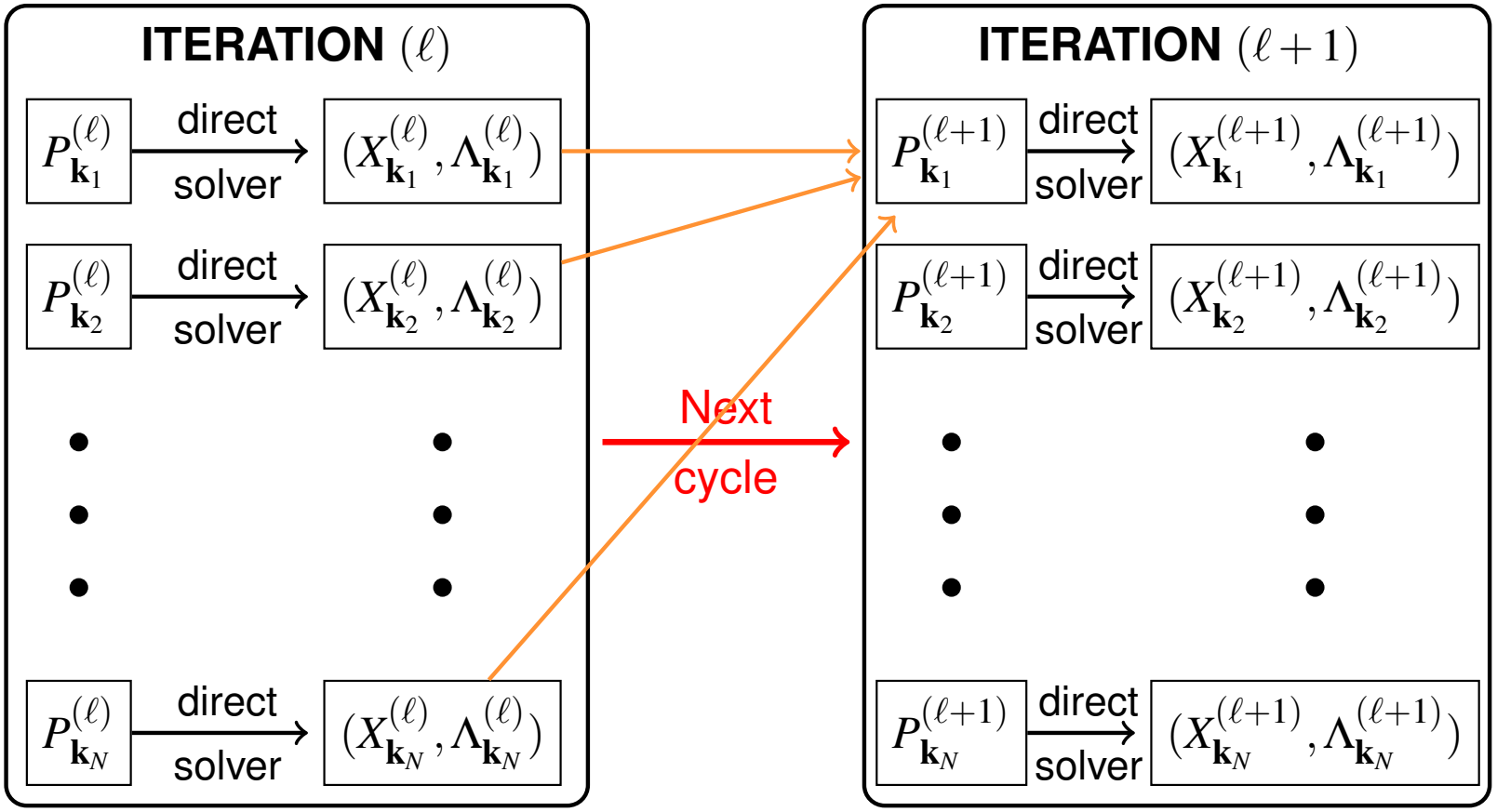
Adjacent cycles



$$X \equiv \{x_1, \ldots, x_n\} \qquad \Lambda \equiv \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

Density Functional Theory scheme

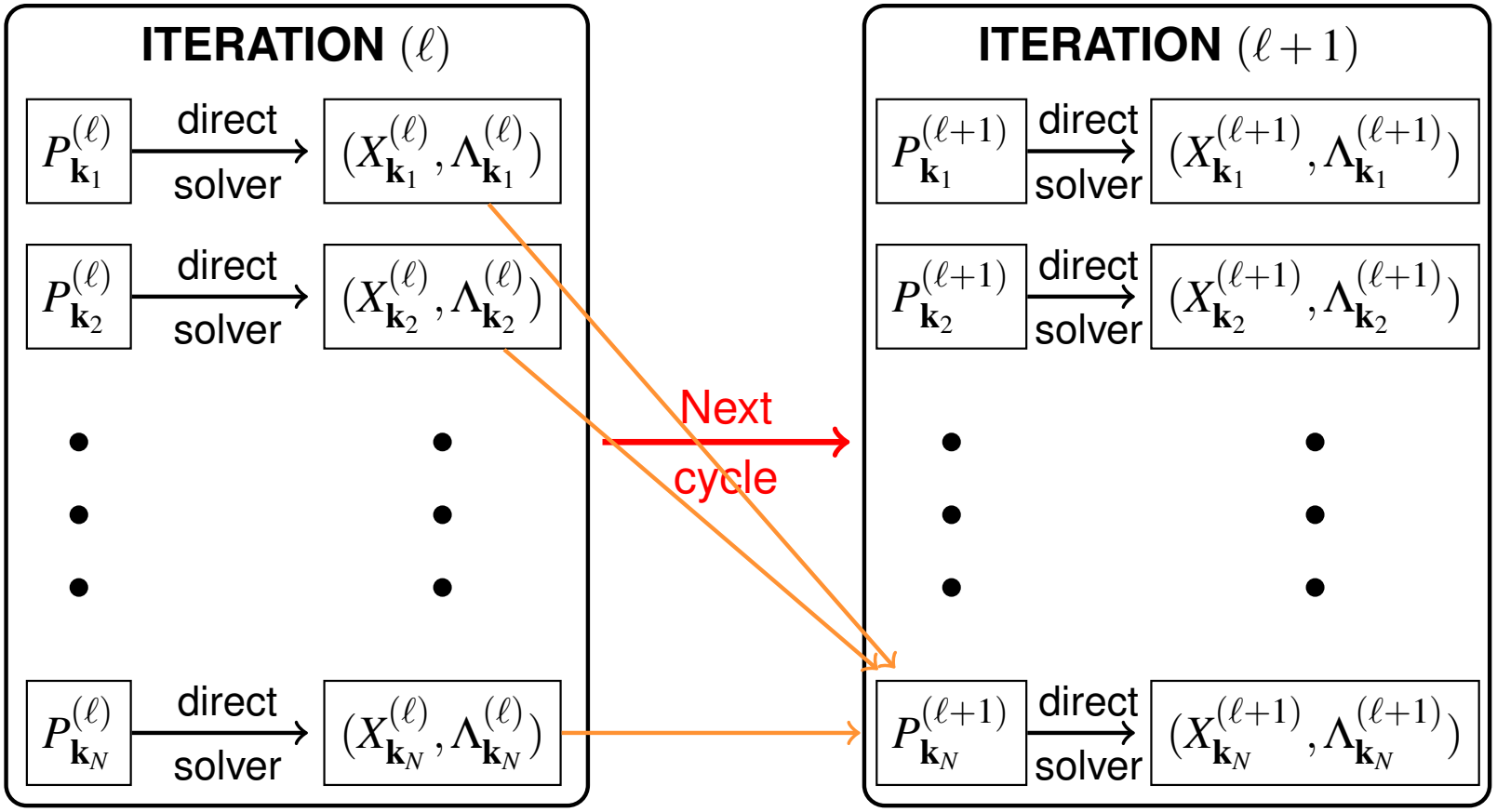
Adjacent cycles



$$X \equiv \{x_1, \ldots, x_n\} \qquad \Lambda \equiv \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

Density Functional Theory scheme

Adjacent cycles



$$X \equiv \{x_1, \ldots, x_n\} \qquad \Lambda \equiv \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

Sequences of eigenproblems

Current solving strategy

- The set of generalized eigenproblems $P^{(1)} \dots P^{(\ell)} P^{(\ell+1)} \dots P^{(N)}$ is handled as a set of disjoint problems $N \times P$;
- Each problem $P^{(\ell)}$ is solved independently using a direct solver as a black-box from a standard library (i.e. ScaLAPACK).

Sequences of eigenproblems

Current solving strategy

- The set of generalized eigenproblems $P^{(1)} \dots P^{(\ell)} P^{(\ell+1)} \dots P^{(N)}$ is handled as a set of disjoint problems $N \times P$;
- Each problem $P^{(\ell)}$ is solved independently using a direct solver as a black-box from a standard library (i.e. ScaLAPACK).

Extracting information $\longrightarrow$ searching for a new strategy

- Treated the set of eigenproblems $N \times P$ as a sequence $\left\{ P^{(\ell)} \right\}$;
- Investigated the presence of a connection between adjacent problems,
  - collected data on **angles** b/w eigenvectors of adjacent eigenproblems;

$$\Theta_{\mathbf{k}_i}^{(\ell)} \equiv \{\theta_1, \dots, \theta_n\} = \mathrm{diag}\left(\mathbb{1} - \langle X_{\mathbf{k}_i}^{(\ell-1)}, \tilde{X}_{\mathbf{k}_i}^{(\ell)} \rangle \right)$$

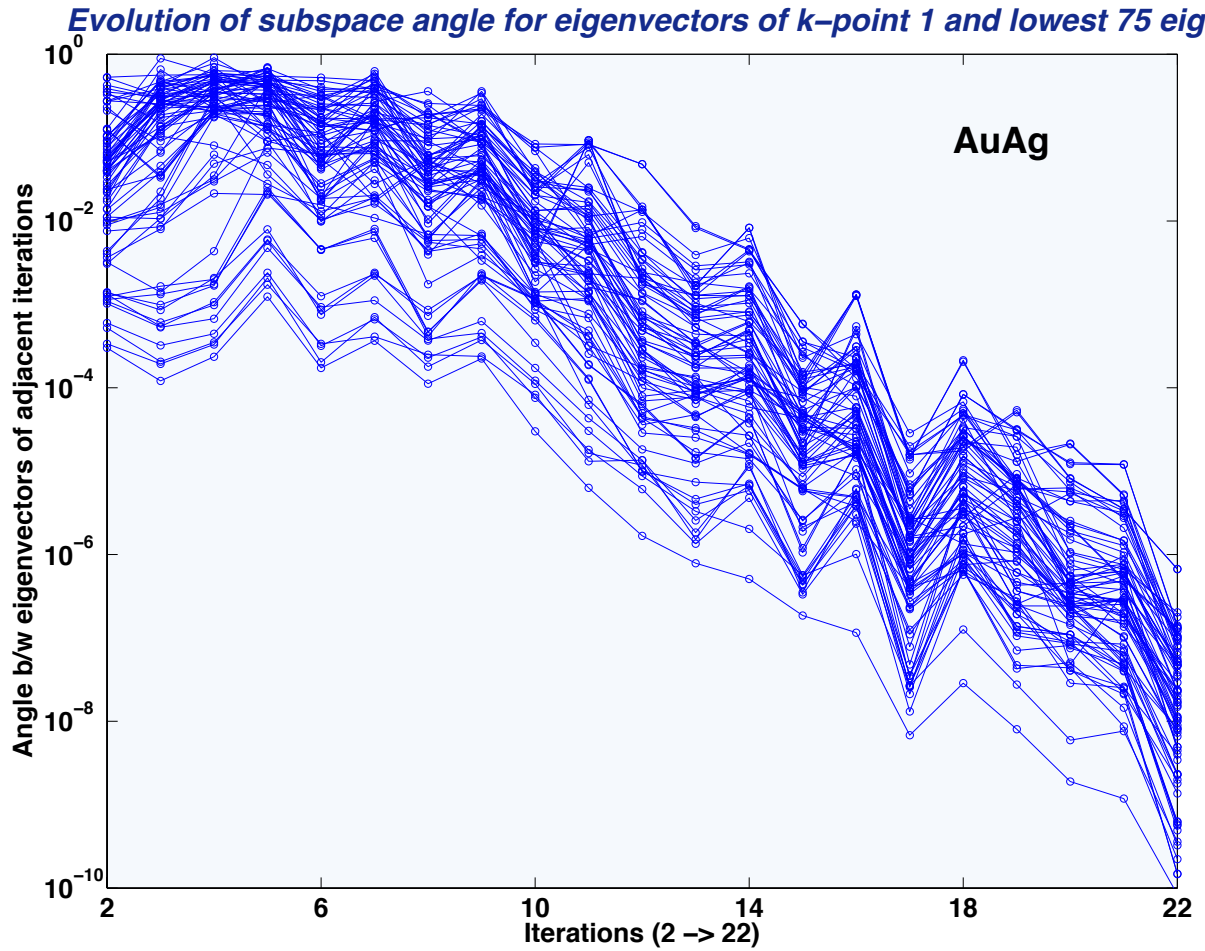
  - uncovered **evolution** of eigenvectors along the sequence

$$\text{for fixed } \mathbf{k}_i \quad \theta_j^{(2)} \geq \theta_j^{(3)} \geq \dots \geq \theta_j^{(N)} : \quad \theta_j^{(2)} \gg \theta_j^{(N)}$$

Angles evolution

fixed $\mathbf{k}$

Example: a metallic compound at fixed $\mathbf{k}$



Correlation and its exploitation

Correlation

- $\exists$ **correlation** between successive eigenvectors $x_j^{(\ell-1)}$ and $x_j^{(\ell)}$;
- angles are **small** after the first few iterations.

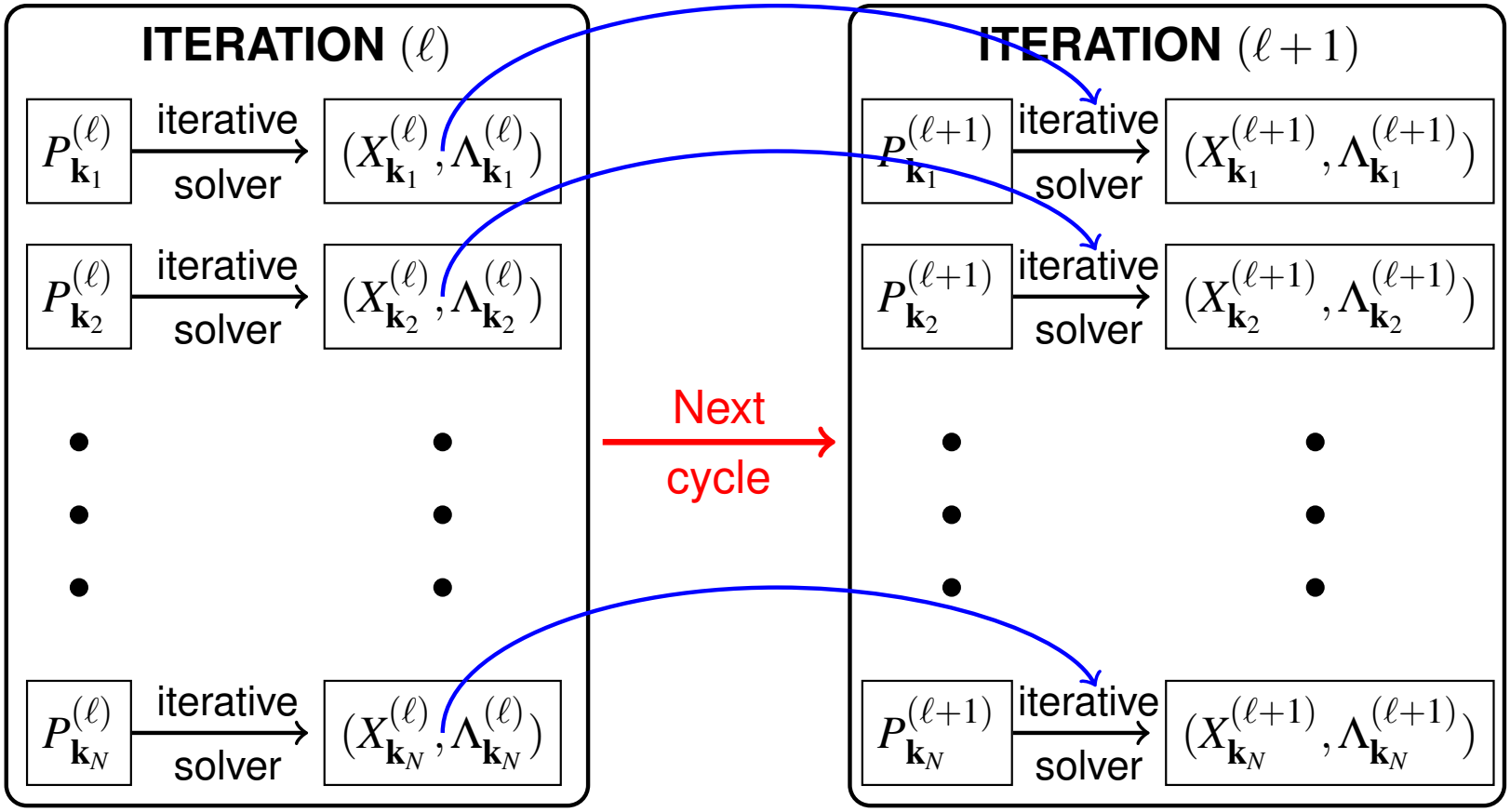
Exploiting information: SIMULATION $\Rightarrow$ ALGORITHM

The stage is favorable to an **iterative eigensolver** where the solution of $P^{(\ell-1)}$ is used to solve $P^{(\ell)}$.

Note: Mathematical model $\not\Rightarrow$ Correlation.
Correlation $\Leftarrow$ **numerical analysis** of the simulation.

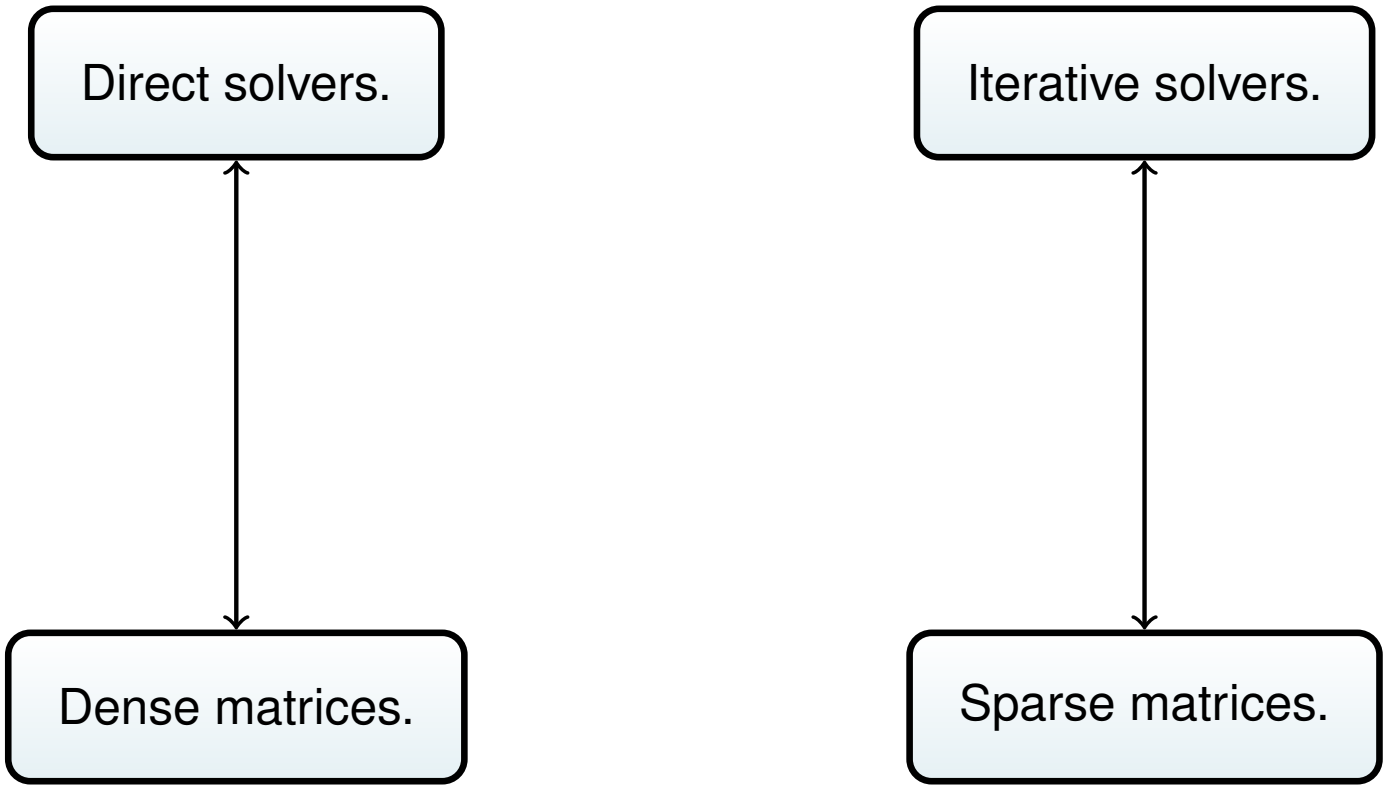
Improved Density Functional Theory scheme

Adjacent cycles

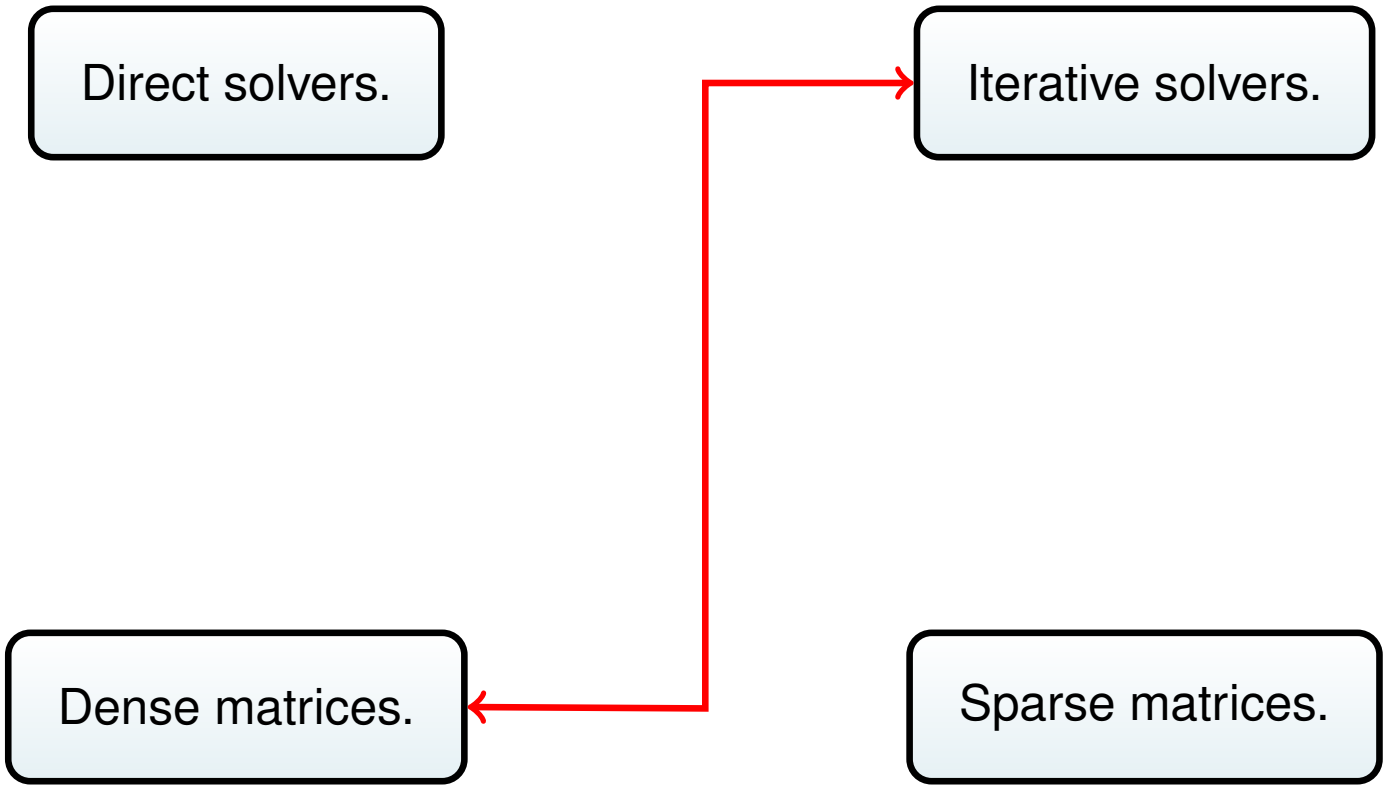


$$X \equiv \{x_1, \ldots, x_n\} \qquad \Lambda \equiv \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

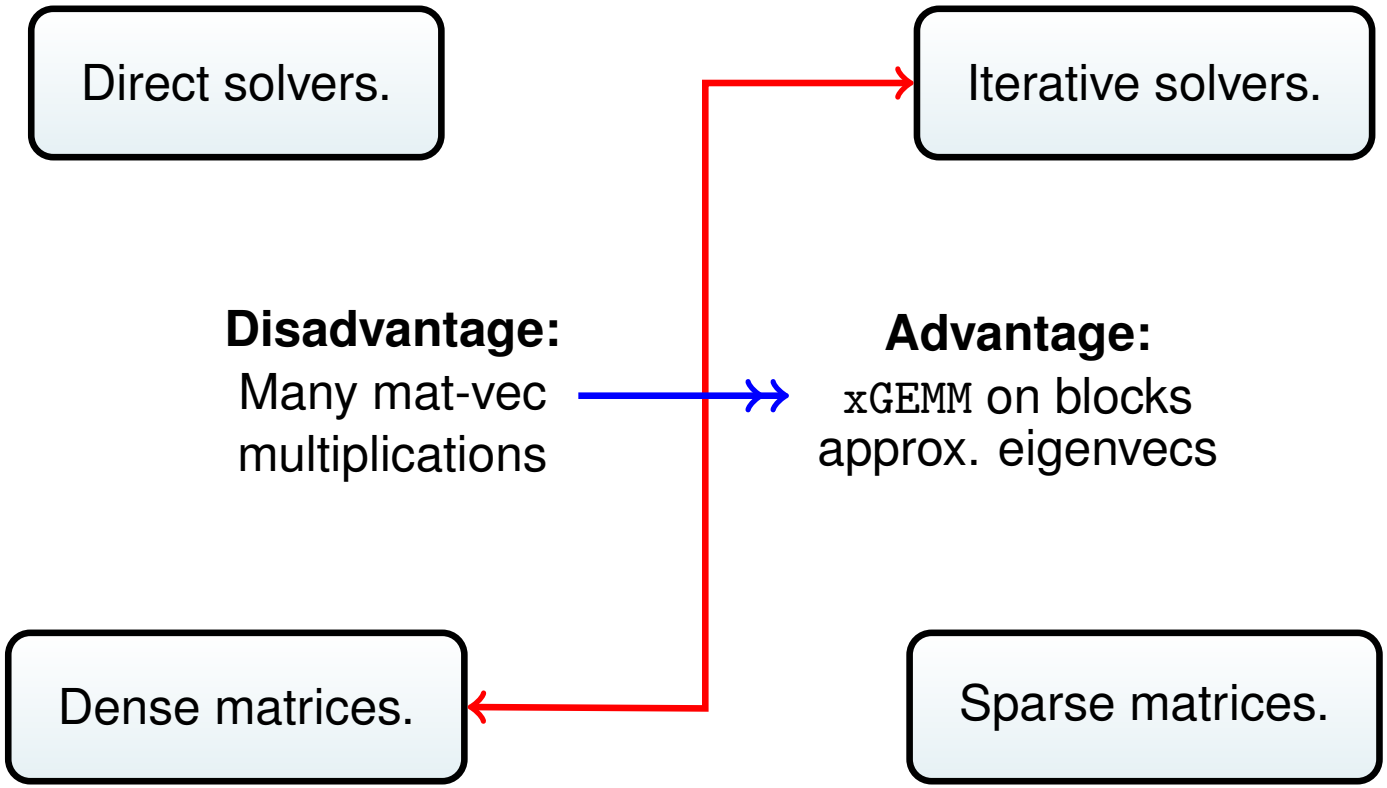
Algorithmic choice



Algorithmic choice



Algorithmic choice



Outline

Sequences of generalized eigenproblems in all-electron computations

The algorithm: Chebyshev Filtered Sub-space Iteration method (**ChFSI**)

ChFSI parallelization and numerical tests

Selecting an iterative eigensolver

Two are the main properties an iterative algorithm has to comply with:

1. The ability to receive as input a sizable set of approximate eigenvectors Z_0 (extracted from $X_{\mathbf{k}_i}^{(\ell-1)}$);
2. The capacity to solve simultaneously for a substantial portion of eigenpairs.

Selecting an iterative eigensolver

Two are the main properties an iterative algorithm has to comply with:

1. The ability to receive as input a sizable set of approximate eigenvectors Z_0 (extracted from $X_{\mathbf{k}_i}^{(\ell-1)}$);
2. The capacity to solve simultaneously for a substantial portion of eigenpairs.

ChFSI constitutes the natural choice:

- it accepts the full set of multiple starting vectors;
- it avoids stalling when facing small clusters of eigenvalues;
- when augmented with polynomial accelerators it has a much faster convergence rate;
- converged eigenvectors can be easily locked;
- the degree of the polynomial can be opportunely optimized.

ChFSI pseudocode

INPUT: Hamiltonian, approximate eigenpairs – (Λ, Z_0), TOL, DEG.
OUTPUT: Wanted eigenpairs W.

1. *Lanczos step.* Identify the bounds for the **eigenspectrum interval**.

REPEAT UNTIL CONVERGENCE:

2. *Chebyshev filter.* **Filter** a block of vectors $W \longleftarrow Z_0$.
3. Re-orthogonalize the vectors outputted by the filter; $W = QR$.
4. Compute the **Rayleigh quotient** $G = Q^{\dagger} H Q$.
5. Compute the primitive Ritz pairs (Λ, Y) by solving for $GY = Y\Lambda$.
6. Compute the approximate Ritz pairs $(\Lambda, W \leftarrow QY)$.
7. *Check* which one among the Ritz vectors *converged*.
8. *Deflate* and *lock* the converged vectors.

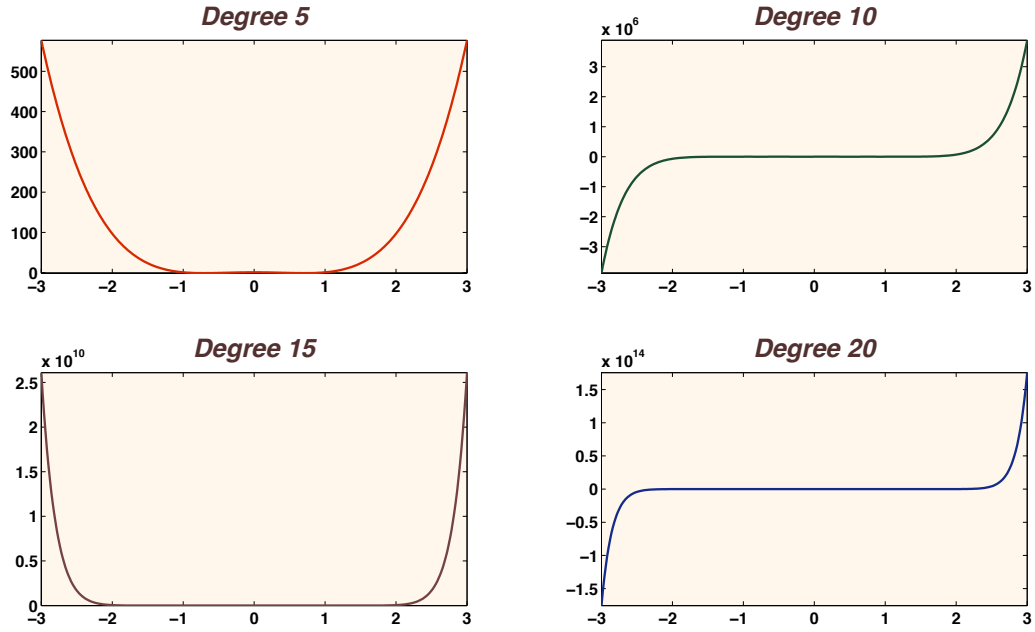
END REPEAT

The core of the algorithm: Chebyshev filter

Chebyshev polynomials

The Chebyshev polynomial C_m of the first kind of order m, is defined as

$$C_m(x) = \begin{cases} \cos\left(m \arccos(x)\right), & x \in [-1,1]; \\ \cosh\left(m \operatorname{arccosh}(x)\right), & |x| > 1. \end{cases}$$



Three-terms recurrence relation

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

The core of the algorithm: Chebyshev filter

The pseudocode

A simple linear transformation maps $[-1,1] \longrightarrow [\alpha,\beta] \subset \mathbb{R}$ defines $c = \frac{\beta+\alpha}{2}$ as the center of the interval and $e = \frac{\beta-\alpha}{2}$ as the width of the interval.

INPUT: Hamiltonian H - approx. eigenvectors Z_0
lowest eigenv. λ_1 - parameters for interval c, e - DEG.
OUTPUT: Filtered vectors W.

1. $\sigma_1 \leftarrow e/(\lambda_1 - c)$
2. $Z_1 \leftarrow \frac{\sigma_1}{e}(H - cI_n)Z_0$ `xGEMM`

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

3. $\sigma_{i+1} \leftarrow \frac{1}{(2/\sigma_1 - \sigma_i)}$
4. $Z_{i+1} \leftarrow 2\frac{\sigma_{i+1}}{e}(H - cI_n)Z_i - \sigma_{i+1}\sigma_i Z_{i-1}$ `xGEMM`

END FOR.

Outline

Sequences of generalized eigenproblems in all-electron computations

The algorithm: Chebyshev Filtered Sub-space Iteration method (**ChFSI**)

ChFSI parallelization and numerical tests

Experimental tests setup

C language implementation of ChFSI

- OMPChFSI – OpenMP parallelization for shared memory
- EleChFSI – Elemental (MPI) parallelization for distributed memory

Matrix sizes: 2,600 $\div$ 13,300.

We used the standard form for the problem

$Ax = \lambda Bx \quad \longrightarrow \quad A'y = \lambda y$ with $A' = L^{-1}AL^{-T}$ and $y = L^T x$

Tests were performed on the JUROPA cluster.

- 2 Intel Xeon 5570 (Nehalem-EP) quad-core processors/node, 2.93GHz;
- 24 GB/node;
- THEORETICAL PEAK PERFORMANCE/CORE=11.71 GFLOPS;
- Minimum absolute tolerance of residuals $r_x = 10^{-10}$;
- All numerical data are MEDIAN values over 12 distinct measurements.

Distributed memory: "Elemental" framework

The core of the library is the two-dimensional cyclic element-wise matrix distribution.

- p MPI processes are logically viewed as a two-dimensional $r \times c$ process grid with $p = r \times c$.
- The matrix $A = [a_{ij}]$ is distributed over the grid in such a way that the process (s,t) owns the matrix.

$$A_{s,t} = \begin{pmatrix} a_{\gamma,\delta} & a_{\gamma,\delta+c} & \cdots \\ a_{\gamma+r,\delta} & a_{\gamma+r,\delta+c} & \cdots \\ \vdots & \vdots & \end{pmatrix},$$

- $\gamma \equiv (s+\sigma_r) \mod r$ and $\delta \equiv (t+\sigma_c) \mod c$,
- σ_r and σ_c are arbitrarily chosen alignment parameters.

Distributed memory: “Elemental” framework

The overall performance is influenced by **grid shape** and **algorithmic block size**.

Grid Shape

For a given number $p > 1$ of processors there are several possible choices for r and c making a grid shape $(r, c) = r \times c$.

It can have a significant impact on the overall performance and is usually influenced by the algorithm.

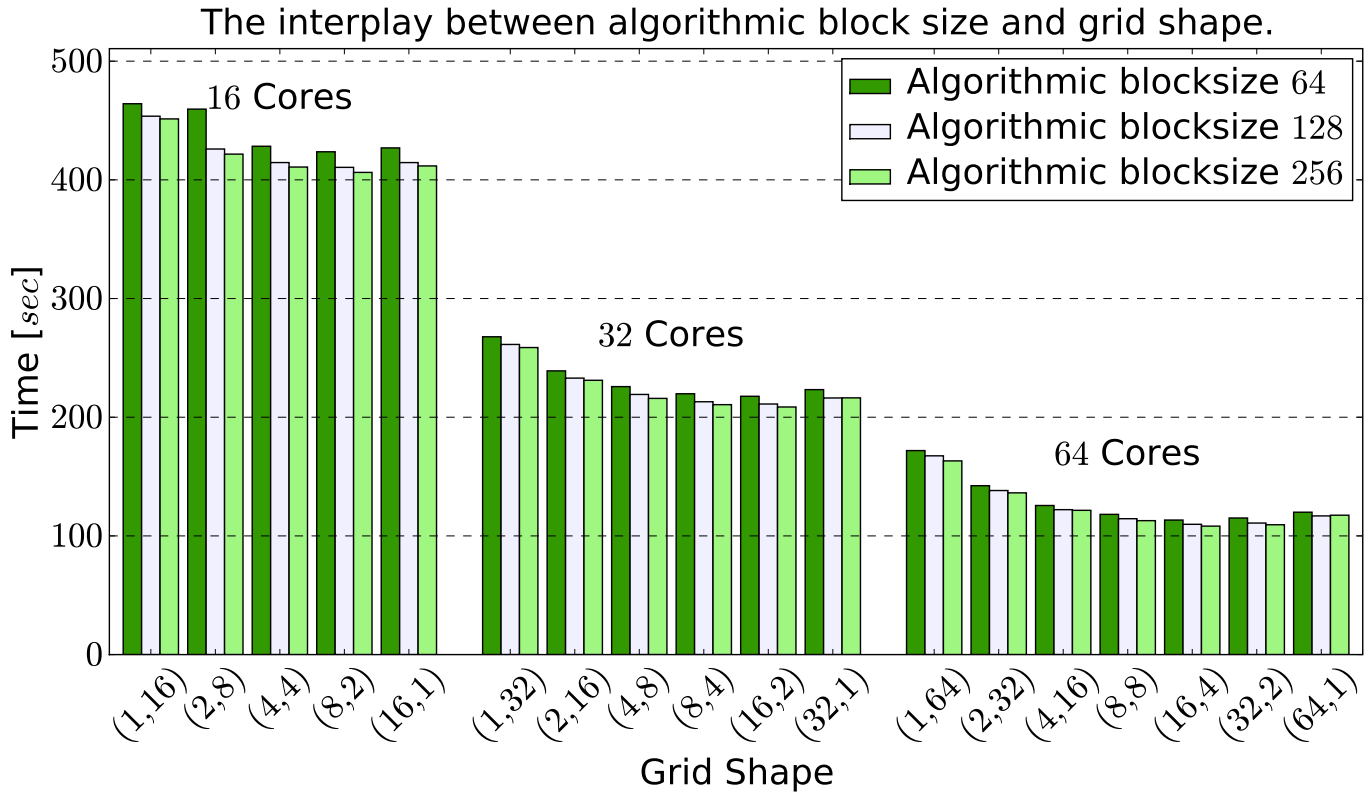
Algorithmic block size

Algorithmic block size: size of blocks of input data, correlated to the square root of the L2 cache.

Not only depends on the algorithm itself, but it is also affected by the architecture.

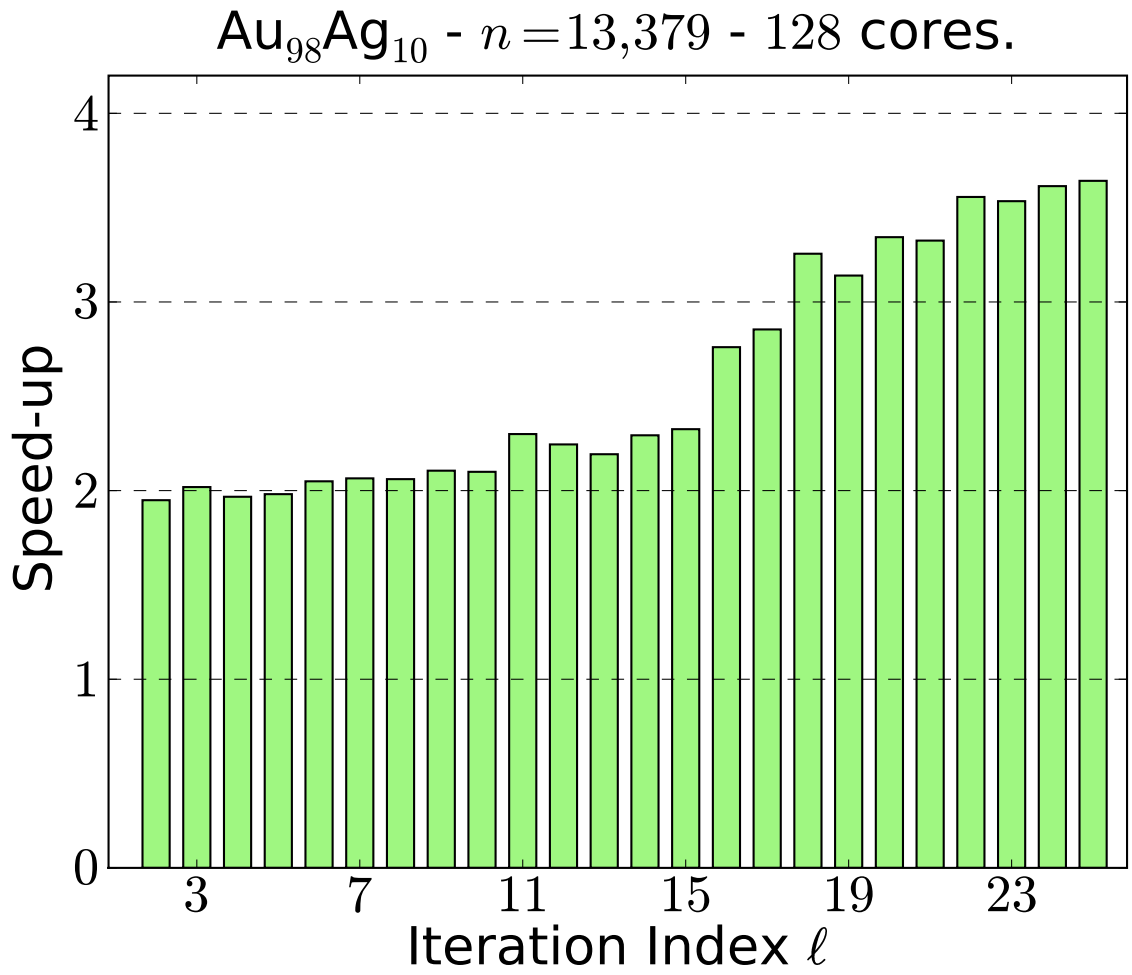
Distributed memory: “Elemental” framework

$\ell = 20$, size $n = 13,379$, and number of sought after eigenpairs $\texttt{nev} = 972$. The problem was solved with EleChFSI using 16, 32, and 64 cores, all possible grid shapes (r, c) and three distinct algorithmic block sizes.



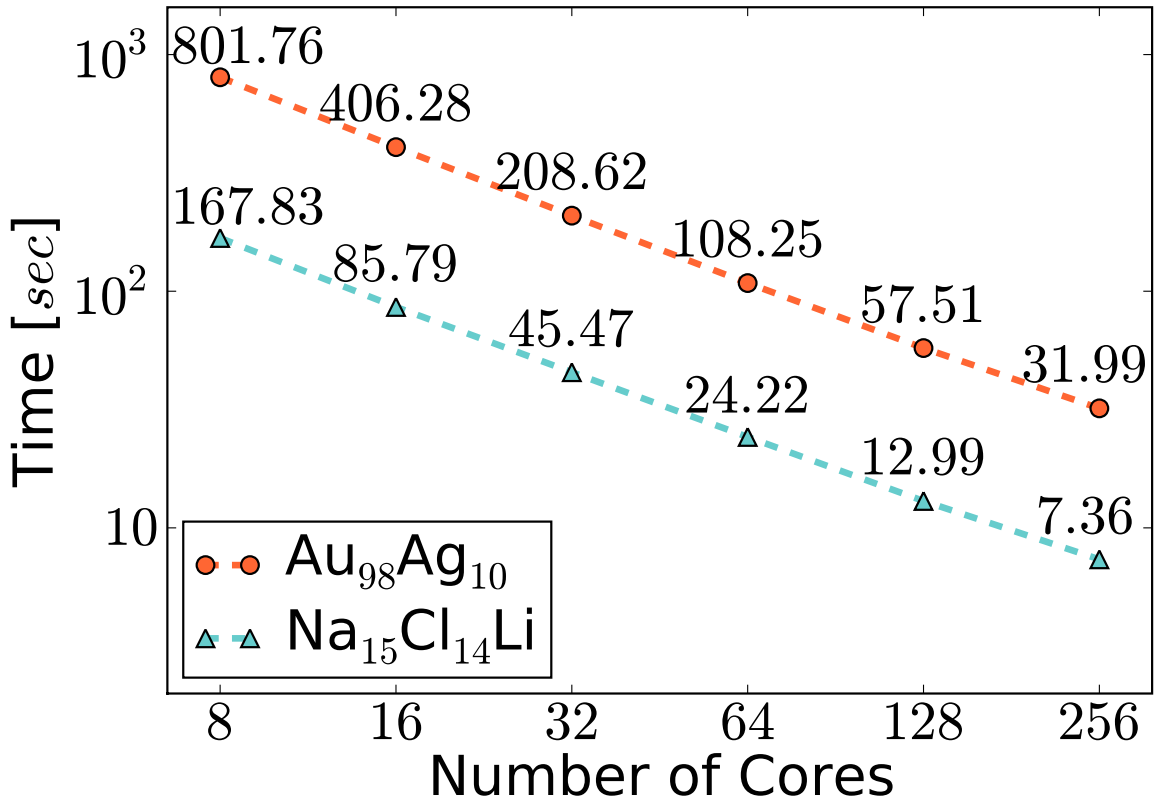
Speed-up

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



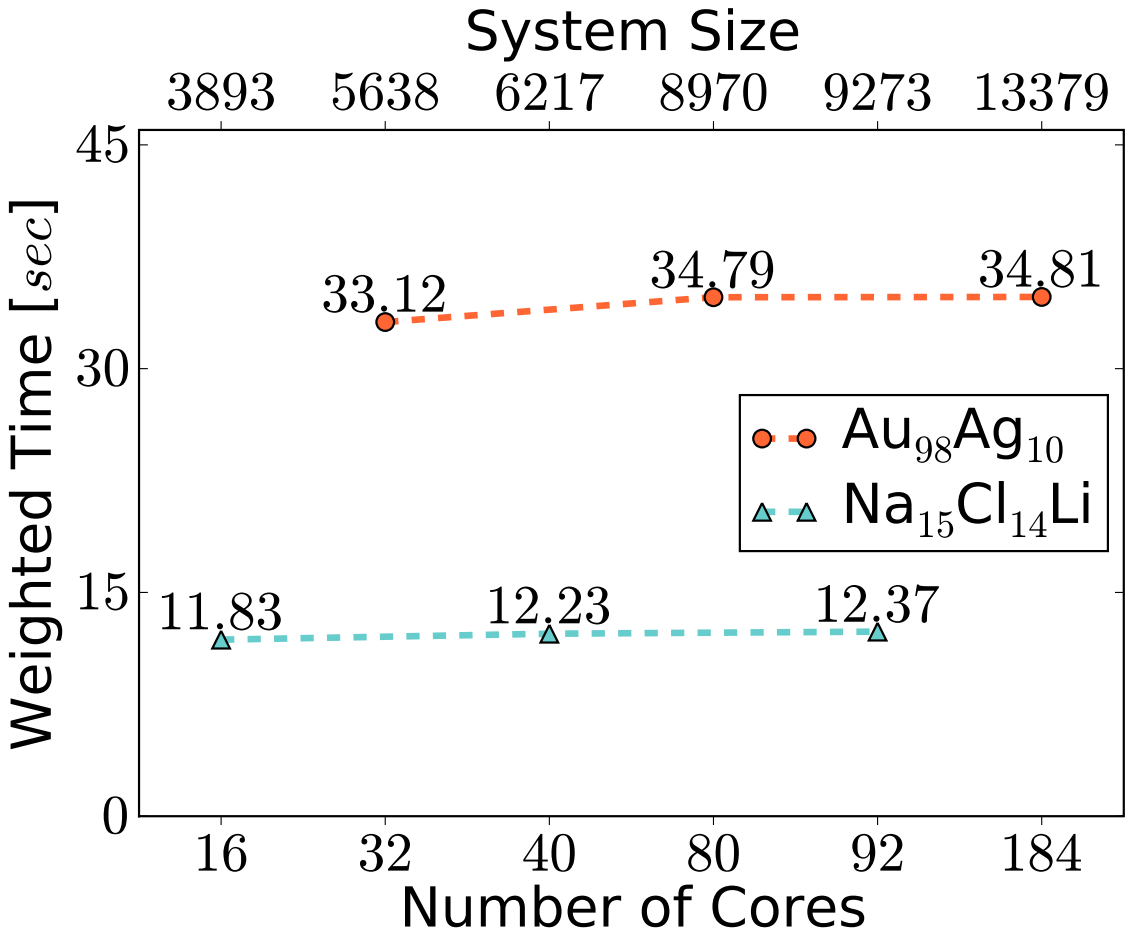
Scalability

Strong Scalability (the size of the eigenproblems are kept fixed while the number of cores is progressively increased) for EleChFSI over two systems of size $n = 13,379$ and $n = 9,273$ respectively.



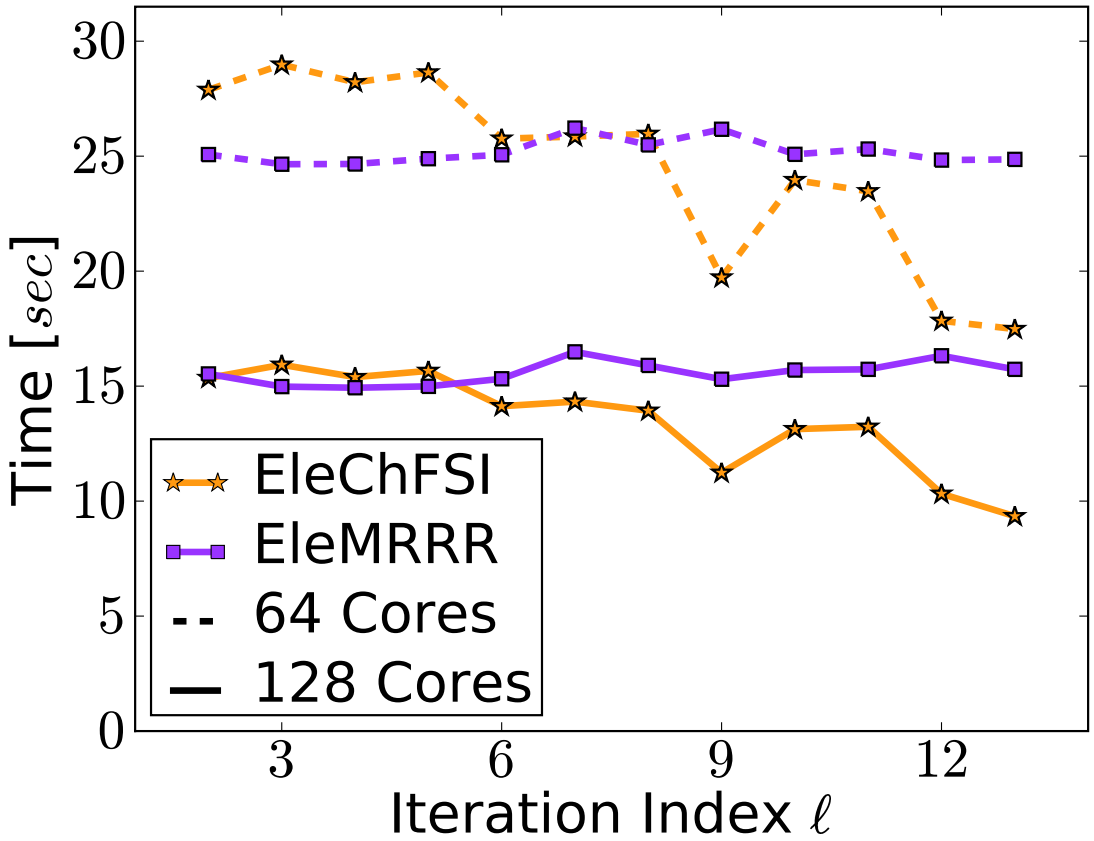
Scalability

Weak Scalability (fixed ratio of data per processor) for EleChFSI. Times are weighted a posteriori keeping into account the ratio of operations per data.



EleChFSI versus direct solvers (parallel MRRR)

For the size of eigenproblems here tested the ScaLAPACK implementation of BXINV is comparable with EleMRRR. For this reason a direct comparison with ScaLAPACK is not included.



Conclusions and future work

Computing **MORE** (with higher efficiency) grants more performance than computing **LESS** (with lower efficiency).

- (Preconditioning) ChFSI with eigenvectors of $P^{(\ell-1)}$ to **speedup** the solution $P^{(\ell)}$ is a successful strategy;
- The parallelization of ChFSI shows great potential for scalability and efficiency;
- The improved use of computing resources together with strong scalability will enable to access **larger physics**;

ONGOING AND FUTURE WORK

1. Finalizing filter optimization by adjusting the degree of the polynomial so to just achieve the required eigenvector residuals.
2. Parallelization of ChFSI for GPUs;

References

1 EDN, and M. Berljafa
A Parallel and Scalable Iterative Solver for Sequences of Dense Eigenproblems Arising in FLAPW
To be published in the proceedings of PPAM 2013. [arXiv:1305.5120]

2 EDN, and M. Berljafa
Block Iterative Eigensolvers for Sequences of Correlated Eigenvalue Problems
Comp. Phys. Comm. 184 (2013), pp. 2478-2488, [arXiv:1206.3768].

3 EDN, P. Bientinesi, and S. Blügel,
Correlation in sequences of generalized eigenproblems arising in Density Functional Theory,
Comp. Phys. Comm. 183 (2012), pp. 1674-1682, [arXiv:1108.2594].