Edoardo Di Napoli

Jülich Supercomputing Center - Institute for Advanced Simulation
Forschungszentrum Jülich GmbH

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

Columbia University, March 5th 2013
Motivation and Goals

Two Facts

- Often algorithmic libraries are used as **black boxes**.
- Very **little information** coming from the physics of the specific application is exploited by scientific computing codes.
Motivation and Goals

Two Facts

- Often algorithmic libraries are used as **black boxes**.
- Very **little information** coming from the physics of the specific application is exploited by scientific computing codes.

One Objective

**Exploiting** physical information extracted from the simulations in order to:

- increase the **performance** of large legacy codes;
- improve the **computational paradigm** on which the codes are based.
Math $\Rightarrow$ Algorithm
Often a “literal” translation of the mathematical model into a combination of algorithmic tasks usually designed in abstraction from other considerations (kernel optimization, tasks harmonization, etc.);

Algorithm $\Rightarrow$ Sim
The simulation is an end product, the successful implementation of an algorithm into a series of machine accessible operations resulting in the computation of meaningful physical quantities;

Math $\Rightarrow$ Sim
The mathematical model and the simulation are considered as practically disjoint.
Math $\Leftrightarrow$ Algorithm
The mathematics model undergoes substantial reformulation in order to cast operations in terms of optimized algorithmic structures. At the same time concerns regarding the numerical stability and accuracy are carefully taken into consideration and included in the reformulation;

Algorithm $\Leftrightarrow$ Sim
Simulations can become a tool to uncover hidden aspects of physical phenomena missed by the mathematical model. Information extracted from a careful analysis of the computations can greatly influence the choice of the optimal algorithm and how it should be used;

Math $\Leftrightarrow$ Sim
Some information extracted from a careful analysis of the simulations can influence so much the algorithmic choice to suggest a complete revolution in the formulation of the mathematical model.
Reverse Simulation
Math, Algorithm $\leftrightarrow$ Sim

total energy
band energy gap
conductivity
forces, etc.

$\leftarrow$ Simulations

$\leftarrow\rightarrow$

Mathematical model
Algorithmic structure

Feedback from analysis of correlated eigenproblems $\{P_i\}$
Sequences of eigenproblems in Density Functional Theory
A glimpse of the results

### Speedup

![Speedup graph](image)

**Speedup vs. Iteration index for Nev=256 and three distinct matrix sizes**

<table>
<thead>
<tr>
<th>Material</th>
<th>Size (n)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaCl</td>
<td>3893</td>
<td></td>
</tr>
<tr>
<td>NaCl</td>
<td>6217</td>
<td></td>
</tr>
<tr>
<td>NaCl</td>
<td>9273</td>
<td></td>
</tr>
</tbody>
</table>

### Scalability

![Scalability graph](image)

**Strong scalability**

<table>
<thead>
<tr>
<th>Material</th>
<th>Size (n)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuAg</td>
<td>8970</td>
<td></td>
</tr>
<tr>
<td>NaCl</td>
<td>6217</td>
<td></td>
</tr>
<tr>
<td>CaFeAs</td>
<td>2612</td>
<td></td>
</tr>
</tbody>
</table>

### Table: Efficiency

<table>
<thead>
<tr>
<th>Self-consistent cycle</th>
<th>Percentage of peak performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>82.3%</td>
</tr>
<tr>
<td>3</td>
<td>81.8%</td>
</tr>
<tr>
<td>4</td>
<td>81.4%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>82.3%</td>
</tr>
<tr>
<td>12</td>
<td>81.8%</td>
</tr>
<tr>
<td>13</td>
<td>81.7%</td>
</tr>
</tbody>
</table>

- **Fast**: a great improvement of the execution time
- **Efficient**: an extensive use of computing resources
- **Scalable**: ideal use of parallel computing architectures
1. Stating the problem: how sequences of generalized eigenproblems arise in all-electron computations

2. Algorithmic choice $\iff$ Eigenvectors angle evolution

3. Tailoring the algorithm: Chebyshev Filtered Sub-space Iteration method (ChFSI)

4. Physics-driven performance boost: numerical results
1. Stating the problem: how sequences of generalized eigenproblems arise in all-electron computations

2. Algorithmic choice ⟷ Eigenvectors angle evolution

3. Tailoring the algorithm: Chebyshev Filtered Sub-space Iteration method (ChFSI)

4. Physics-driven performance boost: numerical results
Quantum Mechanics and its ingredients

- $H = -\frac{\hbar^2}{2m} \sum_{i=1}^{n} \nabla_i^2 - \sum_{i=1}^{n} \sum_{\alpha} \frac{Z_\alpha}{|x_i - a_\alpha|} + \sum_{i<j} \frac{1}{|x_i - x_j|}$

- $\Phi(x_1; s_1, x_2; s_2, \ldots, x_n; s_n)$

$\Phi : \left(\mathbb{R}^3 \times \{-\frac{1}{2}, \frac{1}{2}\}\right)^n \rightarrow \mathbb{R}$ high-dimensional anti-symmetric function – describes the orbitals of atoms and molecules. In the Born-Oppenheimer approximation, it is the solution of the

**Electronic Schrödinger Equation**

$$H \Phi(x_1; s_1, x_2; s_2, \ldots, x_n; s_n) = \mathcal{E} \Phi(x_1; s_1, x_2; s_2, \ldots, x_n; s_n)$$
Density Functional Theory (DFT)

Hohenberg-Kohn theorem (1964)

- ∃ one-to-one correspondence $n(r) \leftrightarrow V_0(r) \implies V_0(r) = V_0(r)[n]$
- ∃! a functional $E[n] : E_0 = \min_n E[n]$

1. $\Phi(x_1; s_1, x_2; s_2, \ldots, x_n; s_n) \implies \Lambda_{i,a} \phi_a(x_i; s_i)$
2. **Charge density** $n(r) = \sum_a f_a |\phi_a(r)|^2$
3. In the Schrödinger equation the exact Coulomb interaction is substituted with an effective potential $V_0(r) = V_1(r) + V_H(r) + V_{xc}(r)$

The high-dimensional Schrödinger equation translates into a set of coupled non-linear low-dimensional self-consistent Kohn-Sham (KS) equations

$$\forall a \quad \text{solve} \quad \hat{H}_{KS} \phi_a(r) = \left(-\frac{\hbar^2}{2m} \nabla^2 + V_0(r)\right) \phi_a(r) = \varepsilon_a \phi_a(r)$$
Typically this set of equations is solved using an iterative self-consistent cycle

\[
\begin{align*}
\text{Initial guess} & \quad n_{\text{init}}(\mathbf{r}) \\
\implies & \quad \text{Compute KS potential} \quad V_0(\mathbf{r})[n] \\
& \quad \text{Solve KS equations} \quad \hat{H}_{\text{KS}} \phi_a(\mathbf{r}) = \varepsilon_a \phi_a(\mathbf{r}) \\
& \quad \uparrow \text{No} \\
\text{OUTPUT} & \quad \text{Converged?} \quad |n^{(\ell+1)} - n^{(\ell)}| < \eta \\
& \quad \text{Compute new density} \quad n(\mathbf{r}) = \sum_a f_a |\phi_a(\mathbf{r})|^2 \\
\iff & \quad \text{Yes} \quad \Rightarrow \text{Energy, forces, ...}
\end{align*}
\]

In practice this iterative cycle is much more computationally challenging and requires some form of broadly defined \textit{discretization}. A common way of discretizing the KS equations is to expand the wave functions \(\phi_a(\mathbf{r})\) on a basis set.
Introduction to FLAPW

Linearized augmented plane wave (LAPW) in a Muffin Tin (MT) geometry

\[
\phi_{k, \nu}(r) = \sum_{|G+k| \leq G_{\text{max}}} c_{k, \nu}^G \psi_G(k, r)
\]

\[
\psi_G(k, r) = \begin{cases} 
\frac{1}{\sqrt{\Omega}} e^{i(k+G)r} & \text{Interstitial (I)} \\
\sum_{\alpha, \ell, m} \left[ a_{\ell m}^\alpha, G(k) u_\ell^\alpha(r) + b_{\ell m}^\alpha, G(k) \dot{u}_\ell^\alpha(r) \right] Y_{\ell m}(\hat{r}_\alpha) & \text{Muffin Tin (MT)}
\end{cases}
\]

Boundary conditions

Continuity of wavefunction and its derivative at MT boundary

\[
d_{\ell m}^{\alpha, G}(k) \quad \text{and} \quad b_{\ell m}^{\alpha, G}(k)
\]
The radial functions $u_\ell^\alpha(r)$ are the solutions of the atomic Schrödinger equations where the potential retains only the spherical part.

\[
\hat{H}_{\text{sph}}^\alpha u_\ell^\alpha(r) = \left[ -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial r^2} + \frac{\hbar^2}{2m} \frac{\ell(\ell + 1)}{r^2} + V_{\text{sph}}^\alpha(r) - r u_\ell^\alpha(r) \right] = E_\ell u_\ell^\alpha(r)
\]

where the $\dot{u}$ can be determined from the energy derivative of the Schrödinger-like equation

\[
\hat{H}_{\text{sph}}^\alpha \dot{u}_\ell^\alpha = E_\ell \dot{u}_\ell^\alpha + u_\ell^\alpha
\]

- $E_\ell$ is a parameter and it is predetermined by optimization
- like the plane wave case there is a cut-off $|G_{\text{max}}|$ (typically $\sim 3.5 - 4.0$)
- unlike the plane waves there is also a cut-off (for open-shell atoms) on $\ell_{\text{max}}$ (typically $\sim 8$)
- unlike the plane waves this set of basis functions is overcomplete $\Rightarrow$ it is NOT an orthogonal set
- a new set of basis functions is computed at each self-consistent cycle
The expansion of is $\phi_{k,\nu}(r)$ in terms of LAPW basis set is then inserted in the KS equations

$$
\psi^*_G(k, r) \sum_{G'} \hat{H}_{KS} c^G_{k,\nu} \psi_{G'}(k, r) = \lambda_{kv} \psi^*_G(k, r) \sum_{G'} c^G_{k,\nu} \psi_{G'}(k, r),
$$

and, by defining the matrix entries for the left and right hand side respectively as Hamiltonian $A_k$ and overlap matrices $B_k$,

$$
[A_k B_k]_{GG'} = \sum_\alpha \int dr \ \psi^*_{G}(k, r) \left[ \hat{H}_{KS} \hat{1} \right] \psi_{G'}(k, r)
$$

one arrives at generalized eigenvalue equations parametrized by $k$

$$
P_k : \sum_{G'} (A_k)_{GG'} c^G_{k,\nu} = \lambda_{kv} \sum_{G'} (B_k)_{GG'} c^G_{k,\nu}
$$

or, in compact form,

$$
A_k x_i = \lambda_i B_k x_i. \quad \text{with} \quad x_i = c^G_{k,\nu}.
$$
Discretized Kohn-Sham scheme

Self-consistent cycle

Initial guess \( n_{\text{start}}(r) \) \( \Rightarrow \) Compute KS potential \( V_0(r)[n] \) \( \Rightarrow \) Solve a set of eigenproblems \( P_{k_1} \cdots P_{k_N} \)

↑ No

↓

Converged?

| \( n^{(\ell+1)} - n^{(\ell)} \) | < \( \eta \)

Compute new density

\( n(r) = \sum_{k,\nu} f_{k,\nu} |\phi_{k,\nu}(r)|^2 \)

OUTPUT

Energy, ...

Yes \( \leftarrow \)

\( n_{\text{start}}(r) \)

\( \Rightarrow \) Compute KS potential

Observations:

1. every \( P_k : Ax = B\lambda x \) is a generalized eigenvalue problem;
2. \( A \) and \( B \) are DENSE and hermitian (\( B \) is also pos. def.);
3. \( P_k \)s with different \( k \) index have different size and are independent from each other.
4. \( k = 1:10-100 ; i = 1:20-50 \)
A (mostly) converging process

The traditional approach

**Initialization:** At every iteration of the cycle all numerical quantities are entirely re-computed

- A new set of basis functions is assembled at each at iteration cycle \((\text{Math } \Leftrightarrow \text{ Sim})\)
- The entries of the matrices \(A\) and \(B\) are re-initialized at each iteration cycle \((\text{Algorithm } \Leftrightarrow \text{ Sim})\)

**Eigenproblems:** Each eigenproblem \(P^{(\ell)}: A^{(\ell)}x = \lambda B^{(\ell)}x\) at iteration \(\ell\) is solved in total independence from the eigenproblem \(P^{(\ell-1)}\) of the previous iteration \((\text{Algorithm } \Leftrightarrow \text{ Sim})\)

**Convergence:** Starting with a electron density close enough to the one minimizing the energy \(E_0\) it is likely to reach convergence within few tens of iterations. Unfortunately there is no general theorem establishing the converging conditions

- # of steps is still uncertain. They depend on the material and the initial guess (area of convergence) \((\text{Math } \Leftrightarrow \text{ Algorithm})\)
- \(|n'(r) - n(r)|\) undergo relatively small decreasing oscillations \((\text{Math } \Leftrightarrow \text{ Algorithm & Sim})\)
A (mostly) converging process

The traditional approach

**Initialization:** At every iteration of the cycle all numerical quantities are entirely re-computed

- A new set of basis functions is assembled at each at iteration cycle \((\text{Math} \not\equiv \text{Sim})\)
- The entries of the matrices \(A\) and \(B\) are re-initialized at each iteration cycle \((\text{Algorithm} \not\equiv \text{Sim})\)

**Eigenproblems:** Each eigenproblem \(P^{(\ell)} : A^{(\ell)} x = \lambda B^{(\ell)} x\) at iteration \(\ell\) is solved in total independence from the eigenproblem \(P^{(\ell-1)}\) of the previous iteration \((\text{Algorithm} \not\equiv \text{Sim})\)

**Convergence:** Starting with a electron density close enough to the one minimizing the energy \(E_0\) it is likely to reach convergence within few tens of iterations. Unfortunately there is no general theorem establishing the converging conditions

- # of steps is still uncertain. They depend on the material and the initial guess (area of convergence) \((\text{Math} \not\equiv \text{Algorithm})\)
- \(|n'(r) - n(r)|\) undergo relatively small decreasing oscillations \((\text{Math} \not\equiv \text{Algorithm & Sim})\)
1. Stating the problem: how sequences of generalized eigenproblems arise in all-electron computations

2. Algorithmic choice $\leftrightarrow$ Eigenvectors angle evolution

3. Tailoring the algorithm: Chebyshev Filtered Sub-space Iteration method (ChFSI)

4. Physics-driven performance boost: numerical results
**FLAPW self-consistent cycle**

**Computational costs**

- Spherical $V_0(r)$ \(\downarrow\) $n_0(r)$ init.
- Calculation of full-potential $V(r)[n]$  
- Determining the Lapw wavefunctions $\psi_G(k, r)$
  - Matrices generation
  - Generalized eigenproblems solution $A_k x = \lambda B_k x$
  - Eigenproblems: distribution and setup
  - Fermi energy calculation
  - Eigenpairs selection
  - Calculation of new charge density $n(r)$  
    - $a^{G}_{lm}$ and $b^{G}_{lm}$ coeff.
    - 3-D stars, 2-D stars lattice harmonics
- Convergence check $n(r)$ mixing  
  - $n(r)$ for next cycle
- Calculation of full-potential $V(r)[n]$  
- Convergence check $n(r)$ mixing  
  - $n(r)$ for next cycle

- ~40% time usage
- ~5–10% time usage
- <1% time usage

---

Edoardo Di Napoli (AICES/JSC)  
An example of reverse simulation  
Columbia University, March 5th 2013 18 / 40
Sequences of eigenproblems

- Consider the set of generalized eigenproblems $P^{(1)} \ldots P^{(\ell)} P^{(\ell+1)} \ldots P^{(N)}$ not as a set of disjoint problems $(P)^N$, but as a sequence;

- Could this sequence $\{P^{(\ell)}\}$ of eigenproblems evolve following a convergence pattern in line with the convergence of $n(r)$?

**Reverse Simulation method**

- numerical simulations analyzed employing a parameter-based “inverse” problem method;
- collected data on deviation angles b/w eigenvectors of adjacent eigenproblems;
- identified “evolutions” of eigenvectors along the sequence.
Example: a metallic compound at fixed $k$

Evolution of subspace angle for eigenvectors of $k$-point 1 and lowest 75 eigs

Iterations ($2 \rightarrow 22$)

Angle b/w eigenvectors of adjacent iterations

AuAg
Correlation and its exploitation

- ∃ correlation between successive eigenvectors $x^{(\ell-1)}$ and $x^{(\ell)}$
- Angles decrease monotonically with some oscillation
- Majority of angles are small after the first few iterations

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

**Algorithm $\Leftrightarrow$ Sim**

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

**Next stages of the investigation:**

1. Development of a **block iterative eigensolver** that can exploit the correlation
2. Investigate if approximate eigenvectors can speed-up the iterative solver of choice
3. Understand if such an iterative method be competitive with direct methods for **dense** problems
Direct solvers.

Iterative solvers.
Algorithmic digression

Direct solvers.

Iterative solvers.

\[
\begin{bmatrix}
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & *
\end{bmatrix}
\]

\[
\begin{bmatrix}
* & * \\
* & * & * \\
* & * \\
* & * & * \\
* & * \\
* & *
\end{bmatrix}
\]

 AX_j = \lambda_j x_j

v = \sum_j \gamma_j x_j

Av = \sum_j \lambda_j \gamma_j x_j \Rightarrow A_k v = \sum_j \lambda_k j \gamma_j x_j = \lambda_1 [x_1 + \sum_{j \geq 2} \lambda_j \lambda_1 x_j]

Rate of convergence \to \text{magnitude of } |\lambda_j| / |\lambda_1|

Dense matrices.
Sparse matrices.
Algorithmic digression

Direct solvers.

\[
\begin{bmatrix}
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
\end{bmatrix}
\]

Iterative solvers.

\[
|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots
\]

\[
Ax_j = \lambda_j x_j
\]

\[
v = \sum_j \gamma_j x_j
\]

\[
Av = \sum_j \lambda_j \gamma_j x_j \Rightarrow A^k v = \sum_j \lambda_j^k \gamma_j x_j = \lambda_1 \left[ x_1 + \sum_{j \geq 2} \frac{\lambda_j}{\lambda_1} x_j \right]
\]

Rate of convergence \(\rightarrow\) magnitude of \(\left| \frac{\lambda_j}{\lambda_1} \right|\)
Algorithmic digression

Direct solvers.

Iterative solvers.

Dense matrices.

Sparse matrices.
Algorithmic digression

- Direct solvers.
- Iterative solvers.
- Dense matrices.
- Sparse matrices.

Matrix $A$:

$$
\begin{bmatrix}
\ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast & \ast \\
\end{bmatrix}
$$

Vector $x$:

$$x = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{bmatrix}$$

Eigenvalue problem:

$$Ax_j = \lambda_j x_j$$

Rate of convergence:

$$\|\|\| \lambda_j - \lambda_1 \|\| \|$$

Dense matrices.

Sparse matrices.

Legend:

- $\mathbf{A}$: Dense matrices.
- $\mathbf{v}$: Sparse matrices.
1. Stating the problem: how sequences of generalized eigenproblems arise in all-electron computations

2. Algorithmic choice ← Eigenvectors angle evolution

3. Tailoring the algorithm: Chebyshev Filtered Sub-space Iteration method (ChFSI)

4. Physics-driven performance boost: numerical results
Two are the essential properties our iterative algorithm has to comply with:
Two are the essential properties our iterative algorithm has to comply with:

1. The ability to receive as input a \textit{sizable set} $Z_0$ of approximate eigenvectors;
2. The capacity to \textit{solve simultaneously} for a substantial portion of eigenpairs.
Two are the essential properties our iterative algorithm has to comply with:

1. The ability to receive as input a **sizable set** $Z_0$ of approximate eigenvectors;
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.
ChFSI pseudocode

**INPUT:** Hamiltonian, approximate eigenpairs \(- (Λ, Z_0)\), TOL, DEG.

**OUTPUT:** Wanted eigenpairs \(W\).

1. **Lanczos step.** Identify the bounds for the interval to be filtered out.

**REPEAT UNTIL CONVERGENCE:**

2. **Chebyshev filter.** Filter a block of vectors \(W \leftarrow Z_0\).
3. **QR decomposition.** Re-orthogonalize the vectors outputted by the filter; \(W = QR\).
4. Compute the Rayleigh quotient \(G = Q^\dagger HQ\).
5. Compute the primitive Ritz pairs \((Λ, Y)\) by solving for \(GY = YΛ\).
6. Compute the approximate Ritz pairs \((Λ, W \leftarrow QY)\).
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

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

Three-terms recurrence relation

$$C_{m+1}(x) = 2xC_m(x) - C_{m-1}(x); \quad m \in \mathbb{N}, \quad C_0(x) = 1, \quad C_1(x) = x$$
The core of the algorithm: Chebyshev filter

The basic principle

**Theorem**

Let $|\gamma| > 1$ and $\mathbb{P}_m$ denote the set of polynomials of degree smaller or equal to $m$. Then the extremum

$$\min_{p \in \mathbb{P}_m, p(\gamma) = 1} \max_{t \in [-1, 1]} |p(t)|$$

is reached by

$$p_m(t) = \frac{C_m(t)}{C_m(\gamma)}.$$ 

A generic vector $\nu$ is very quickly aligned in the direction of the eigenvector corresponding to the extremal eigenvalue $\lambda_1$

$$\nu^m = p_m(A)\nu = \sum_{i=1}^{n} s_i p_m(A)x_i = \sum_{i=1}^{n} s_i p_m(\lambda_i)x_i$$

$$= s_1x_1 + \sum_{i=2}^{n} s_i \frac{C_m(\lambda_i - c)}{C_m(\lambda_1 - c)} x_i$$

$$\sim s_1 x_1$$
A simple linear transformation maps $[-1, 1] \rightarrow [\alpha, \beta] \subset \mathbb{R}$

$$c = \frac{\beta + \alpha}{2} \quad \text{center of the interval} \quad e = \frac{\beta - \alpha}{2} \quad \text{width of the interval}.$$ 

**INPUT:** Hamiltonian $H$ - approx. eigenvectors $Z_0$

lowest eigenv. $\lambda_1$ - parameters for interval $c$, $e$ - DEG.

**OUTPUT:** Filtered vectors $W$.

1. $\sigma_1 \leftarrow e / (\lambda_1 - c)$

2. $Z_1 \leftarrow \frac{\sigma_1}{e} (H - cI_n) Z_0$

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

3. $\sigma_{i+1} \leftarrow \frac{1}{(2/\sigma_1 - \sigma_i)}$

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

**END FOR.**
1. Stating the problem: how sequences of generalized eigenproblems arise in all-electron computations

2. Algorithmic choice $\leftarrow$ Eigenvectors angle evolution

3. Tailoring the algorithm: Chebyshev Filtered Sub-space Iteration method (ChFSI)

4. Physics-driven performance boost: numerical results
Experimental tests setup

Solving with a ChFSI version implemented in C

- Approx. vs Random vectors fed to ChFSI against Iteration Index;
- Parallel ChFSI vs Direct methods against Iteration Index.

Matrix sizes: 2,600 ÷ 13,300.

B is in general almost singular (ill-conditioned).
Examples: \( \text{size}(A) = 50 \rightarrow \kappa(A) \approx 10^4 \); \( \text{size}(A) = 500 \rightarrow \kappa(A) \approx 10^7 \)

We used the standard form for the problem
\[
Ax = \lambda Bx \quad \rightarrow \quad A'y = \lambda y \quad \text{with} \quad A' = L^{-1}AL^{-T} \quad \text{and} \quad y = L^Tx
\]

Tests were performed on JUROPA using 1 node with 8 cores.
- 2 Intel Xeon 5570 (Nehalem-EP) quad-core processors, 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.
Na$_{15}$Cl$_{14}$Li with increasing $|G_{\max}| = 3.0, 3.5, 4.0$
Speed-up for sequential ChFSI

$\text{Au}_{98}\text{Ag}_{10}$ with increasing $|G_{\text{max}}| = 3.0, 3.5$

Graph: Speed-up vs. Iteration index for $\text{Nev}=972$ and 2 distinct matrix sizes

- $\text{AuAg} -- n=5638$
- $\text{AuAg} -- n=8970$

Edoardo Di Napoli (AICES/JSC)
Fraction of computing time for sequential ChFSI extracted from AuAg system (n=8970, nev=972) at iteration 23.

- **Chebyshev filter**: 90%
- **Residuals convergence**: 6%
- **Rayleigh–Ritz**: 4%
- **Lanczos**: <1%
Sequential to Parallel: what and how?

Speed-up of multi-threaded BLAS ChFSI vs sequential ChFSI over 8 cores when fed approx. eigenvectors.

![Speed-up of Multi-threaded ChFSI over Sequential ChFSI w.r.t. Iteration index](image)

- **AuAg** $n=8970$
- **NaCl** $n=6217$

Edoardo Di Napoli (AICES/JSC)  
An example of reverse simulation  
Columbia University, March 5th 2013  
32 / 40
Speed-up of sequential ChFSI and multi-threaded BLAS ChFSI.

Speed-up of approx. vs. random vectors w.r.t. Iteration index

- AuAg --- n=8970 --- 8 cores
- AuAg --- n=8970 --- 1 core

Iteration Index

Speed-up

0 5 10 15 20 25 30
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
### Efficiency

Ratio of performance respect to theoretical peak performance

<table>
<thead>
<tr>
<th>Self-consistent cycle</th>
<th>Full Algorithm</th>
<th>Chebyshev filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>82.3%</td>
<td>91.4%</td>
</tr>
<tr>
<td>3</td>
<td>81.8%</td>
<td>90.9%</td>
</tr>
<tr>
<td>4</td>
<td>81.4%</td>
<td>90.4%</td>
</tr>
<tr>
<td>5</td>
<td>81.7%</td>
<td>90.8%</td>
</tr>
<tr>
<td>6</td>
<td>82.4%</td>
<td>91.5%</td>
</tr>
<tr>
<td>7</td>
<td>82.4%</td>
<td>91.6%</td>
</tr>
<tr>
<td>8</td>
<td>82.4%</td>
<td>91.6%</td>
</tr>
<tr>
<td>9</td>
<td>82.9%</td>
<td>92.1%</td>
</tr>
<tr>
<td>10</td>
<td>81.7%</td>
<td>90.8%</td>
</tr>
<tr>
<td>11</td>
<td>82.3%</td>
<td>91.4%</td>
</tr>
<tr>
<td>12</td>
<td>81.8%</td>
<td>90.9%</td>
</tr>
<tr>
<td>13</td>
<td>81.7%</td>
<td>90.8%</td>
</tr>
</tbody>
</table>
Recapping

1. **Approximate vs. Random:**
   - sequential ChFSI achieves speed-ups in the range $1.5X \div 3.5X$;
   - multi-threaded version of ChFSI achieve speed-ups up to $5X$;

2. **Multi-threaded ChFSI:**
   - by just using the multi-threaded version of BLAS, ChFSI achieve speed-ups above $7X$;
   - the larger the size of the eigenproblem the better ChFSI performs.

3. **Performance:**
   Extensive use of BLAS library in the filter facilitates close to optimal performance.

**Unanswered Questions:**

- Can we do better on shared memory architectures?
- Can we be faster than the direct methods?
Recapping

1. **Approximate vs. Random:**
   - Sequential ChFSI achieves speed-ups in the range $1.5X \div 3.5X$;
   - Multi-threaded version of ChFSI achieve speed-ups up to $5X$;

2. **Multi-threaded ChFSI:**
   - By just using the multi-threaded version of BLAS, ChFSI achieve speed-ups above $7X$;
   - The larger the size of the eigenproblem the better ChFSI performs.

3. **Performance:**
   - Extensive use of BLAS library in the filter facilitates close to optimal performance.

**Unanswered Questions:**

- Can we do better on shared memory architectures?
  - Yes $\rightarrow$ OpenMP.

- Can we be faster than the direct methods?
  - Depends on the percentage of the eigenspectrum sought after and the iteration index.
Parallelizing the filter with OpenMP

Input: Hamiltonian $H$ – vectors to be filtered $Z$ – DEG, c, e.

Output: Filtered vectors – $W$.

1: \textbf{for} $i = 1$ to DEG \textbf{do}
2: \hspace{1em} Compute $\alpha$. Compute $\beta$.
3: \hspace{1em} $W = \alpha HZ + \beta W$.
4: \hspace{1em} SWAP($Z, W$).
5: \hspace{1em} \textbf{end for}
6: SWAP($Z, W$).

\textbf{Figure:} Matrix–matrix multiplication scheme.
Time to completion for AuAg (n=13379) of OpenMP vs. multi-threaded BLAS

- **ChFSI multi-threaded BLAS**
- **ChFSI OpenMP**
OpenMP ChFSI obtains almost ideal strong scalability: the larger the better.

![BChFSI — Strong scalability](image)

- **AuAg** — n=8970
- **NaCl** — n=6217
- **CaFeAs** — n=2612
- **Ideal speed-up**
OpenMP ChFSI vs Direct eigensolvers on shared memory architectures

Comparison with multi-threaded LAPACK MR\(^3\) and MKL multi-threaded LAPACK MR\(^3\).
Comparison with multi-threaded LAPACK MR$^3$ and MKL multi-threaded LAPACK MR$^3$.

<table>
<thead>
<tr>
<th>Iteration index</th>
<th>CPU time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>180</td>
</tr>
<tr>
<td>6</td>
<td>160</td>
</tr>
<tr>
<td>8</td>
<td>140</td>
</tr>
<tr>
<td>10</td>
<td>120</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>80</td>
</tr>
</tbody>
</table>

Time for LAPACK + multi-threaded BLAS and OMP-ChFSI for NaCl (n=9273)

LAPACK + multi-threaded BLAS
MKL LAPACK + multi-threaded BLAS
OpenMP ChFSI

solving for lowest 2.8% of eigenspectrum
CONCLUSIONS

- Feeding eigenvectors of $P^{(\ell-1)}$ to a block iterative solver like ChFSI speed-ups the iterative solver for $P^{(\ell)}$;
- The improved use of computing resources together with strong scalability will enable to access larger physics;
- The algorithmic structure of FLAPW needed to be re-thought: \rightarrow “reverse simulation” can substantially influence the computational paradigm of an application;

ONGOING AND FUTURE WORK

- ChFSI can be extended and improved;
  1. Finalizing filter optimization by adjusting the degree of the polynomial so to just achieve the required eigenvector residuals.
  2. Ongoing work to parallelize ChFSI for distributed memory architectures \rightarrow Elemental;
  3. Parallelization of ChFSI for GPUs by using OpenACC directives;
- Analysis on the structure of the entries of $A$ and $B$ across adjacent iteration seems to suggest an exploitation of low-rank updates for sequences of eigenproblems \rightarrow Hierarchical matrices;
1. M. Berljafa, and EDN
   *A parallel Chebyshev Filtered Subspace Iteration optimized for LAPW-based methods*
   In preparation

2. EDN, and M. Berljafa
   *Block iterative eigensolvers for sequences of correlated eigenvalue problems*

3. EDN, P. Bientinesi, and S. Blügel,
   *Correlation in sequences of generalized eigenproblems arising in Density Functional Theory*,