(Preconditioning) Chebyshev subspace iteration applied to sequences of dense eigenproblems in ab initio simulations

NASCA 2013 Calais, France, June 24th | M. Berljafa and E. Di Napoli
Motivation and Goals

- Electronic Structure
- Band energy gap
- Conductivity
- Forces, etc.

SIMULATIONS

- Mathematical Model
- + Algorithmic Structure

Extracting & Exploiting Information

Performance
- Efficiency
- Scalability
- More Physics
Motivation and Goals

Electronic Structure
Band energy gap
Conductivity
Forces, etc.

SIMULATIONS

Mathematical Model +
Algorithmic Structure

Extracting & Exploiting
Information

SIMULATIONS

Performance

Efficiency

Scalability

More Physics
Outline

Sequences of generalized eigenproblems in all-electron computations

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

Exploiting approximate eigenvectors: numerical results

Optimizations schemes
Outline

Sequences of generalized eigenproblems in all-electron computations

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

Exploiting approximate eigenvectors: numerical results

Optimizations schemes
Density Functional Theory scheme

Self-consistent cycle

Initial guess for charge density $n_{\text{start}}(r)$

Compute discretized Kohn-Sham equations

Solve a set of eigenproblems $P^{(\ell)}_{k_1} \ldots P^{(\ell)}_{k_N}$

Converged?

Yes

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

No

OUTPUT Electronic structure, ...

Compute new charge density $n^{(\ell)}(r)$
Density Functional Theory scheme

Self-consistent cycle

Initial guess for charge density $n_{\text{start}}(r)$

Compute discretized Kohn-Sham equations

Solve a set of eigenproblems $P_{k_1}^{(\ell)} \ldots P_{k_N}^{(\ell)}$

Converged?

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

OUTPUT Electronic structure, ...

Compute new charge density $n^{(\ell)}(r)$

Observations:

1. Every $P_{k}^{(\ell)} : A_{k}^{(\ell)} x = B_{k}^{(\ell)} \lambda x$ 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. $k = 1 : 10 \ldots 100$; $\ell = 1 : 20 \ldots 50$
Sequences of eigenproblems

Current solving strategy

- The set of generalized eigenproblems \( P^{(1)} \ldots P^{(\ell)} P^{(\ell+1)} \ldots P^{(N)} \) is handled as a set of disjoint problems \((P)^N\);

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

Extracting information $\rightarrow$ searching for a new strategy

- Treated the set of eigenproblems $(P)^N$ as a sequence $\{P^{(\ell)}\}$;
- Investigated the presence of a connection between adjacent problems,
  - numerical simulations analyzed employing a parameter-based inverse problem method;
  - collected data on angles b/w eigenvectors of adjacent eigenproblems;
  - discovered evolution of eigenvectors along the sequence.
Angles evolution

fixed $k$

Example: a metallic compound at fixed $k$

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

Angle b/w eigenvectors of adjacent iterations
Correlation and its exploitation

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

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

Exploiting information: **SIMULATION \( \Rightarrow \) ALGORITHM**

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

1. Approximate eigenvectors can **speed-up** iterative solvers (EDN, M. Berljafa [arXiv:1206.3768])
2. Developed of a **block iterative eigensolver (ChFSI)** that can maximally exploit the correlation (EDN, M. Berljafa [arXiv:1305.5120])
3. ChFSI is competitive with direct methods for **dense** problems in ab initio methods (EDN M. Berljafa, and in preparation).
Algorithmic digression

Direct solvers.

Iterative solvers.

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

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

\[
v = \sum_j \gamma_j x_j
\]

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

Rate of convergence → magnitude of $|\lambda_1|/|\lambda_j|$. Dense matrices. Sparse matrices.
Algorithmic digression

Direct solvers.

Iterative solvers.

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

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

\[Ax_j = \lambda_j x_j\]

\[v = \sum_j \gamma_j x_j\]

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

Rate of convergence → magnitude of \[\|\lambda_1 \lambda_j \|\]

Dense matrices.

Sparse matrices.

NASCA 2013 Calais, France, June 24th
M. Berljafa and E. Di Napoli
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 → magnitude of \[\left| \frac{\lambda_1}{\lambda_j} \right|\]
Algorithmic digression

Direct solvers.

Dense matrices.

Iterative solvers.

Sparse matrices.

Rate of convergence → magnitude of $|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots$

Dense matrices.

Sparse matrices.
Algorithmic choice

Direct solvers.

Iterative solvers.

Dense matrices.

Sparse matrices.
Outline

Sequences of generalized eigenproblems in all-electron computations

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

Exploiting approximate eigenvectors: numerical results

Optimizations schemes
Selecting an iterative eigensolver

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

<table>
<thead>
<tr>
<th></th>
<th>The ability to receive as input a sizable set $Z_0$ of approximate eigenvectors;</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>The capacity to <strong>solve simultaneously</strong> for a substantial portion of eigenpairs.</td>
</tr>
</tbody>
</table>
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** $Z_0$ 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**;
- the degree of the polynomial can be opportunoely **optimized**.
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 $v$ is very quickly aligned in the direction of the eigenvector corresponding to the extremal eigenvalue $\lambda_1$

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

$$= s_1x_1 + \sum_{i=2}^{n} s_i \frac{C_m(\lambda_i-c)}{C_m(\lambda_1-c)}x_i \sim s_1x_1$$
The core of the algorithm: Chebyshev filter

The pseudocode

A simple linear transformation maps $[-1, 1] \rightarrow [\alpha, \beta] \subset \mathbb{R}$ 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. $\lambda_1$ - parameters for interval $c$, $e$ - DEG.

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

1. $\sigma_1 \leftarrow \frac{e}{(\lambda_1 - c)}$
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. $Z_{i+1} \leftarrow 2 \frac{\sigma_{i+1}}{e} (H - cI_n) Z_i - \sigma_{i+1} \sigma_i Z_{i-1}$
ChFSI pseudocode

**INPUT:** Hamiltonian, approximate eigenpairs — \((\Lambda, Z_0)\), 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 \((\Lambda, Y)\) 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**
Outline

Sequences of generalized eigenproblems in all-electron computations

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

Exploiting approximate eigenvectors: numerical results

Optimizations schemes
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.

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 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.
Speed-up for sequential ChFSI

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

Speed-up vs. Iteration index for Nev=256 and three distinct matrix sizes

- NaCl -- n=3893
- NaCl -- n=6217
- NaCl -- n=9273
Speed-up for sequential ChFSI

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

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

![Graph showing speed-up vs. iteration index](image)
Sequential to Parallel: “Elemental” framework

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: “Elemental” framework

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.
Sequential to Parallel: “Elemental” framework

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.
Recapping

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

2 **Parallel ChFSI:**
   - the algorithms scales extremely well even for medium size eigenproblems;
   - depending on the number of cores and percentage of eigenspectrum, ChFSI is competitive with direct eigensolvers;
   - having access to more cores enables the investigation of **larger physical systems** (larger eigenproblems).
Recapping

1 Approximate vs. Random:
   - sequential ChFSI achieves speed-ups in the range 1.5X ÷ 3.5X;
   - parallel versions of ChFSI can achieve speed-ups up to 5X.

2 Parallel ChFSI:
   - the algorithms scales extremely well even for medium size eigenproblems;
   - depending on the number of cores and percentage of eigenspectrum, ChFSI is competitive with direct eigensolvers;
   - having access to more cores enables the investigation of larger physical systems (larger eigenproblems).

There is room for further optimizations:
- The degree of the polynomials can be fine-tuned so as to avoid filtering “too much”;
- Profiling the degrees along the sequence provides additional savings.
Outline

Sequences of generalized eigenproblems in all-electron computations

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

Exploiting approximate eigenvectors: numerical results

Optimizations schemes
Convergence ratio and residuals

**Definition**

The **convergence ratio** for the eigenvector $w_i$ corresponding to eigenvalue $\lambda_i \notin [\alpha, \beta]$ is defined as

$$\tau(\lambda_i) = |\rho_i|^{-1} = \min \pm \left| \frac{\lambda_i - c}{e} \pm \sqrt{\left( \frac{\lambda_i - c}{e} \right)^2 - 1} \right|.$$ 

The further away $\lambda_i$ is from the interval $[\alpha, \beta]$ the smaller is $|\rho_i|^{-1}$ and the faster the convergence to $w_i$ is.
Convergence ratio and residuals

Definition

The **convergence ratio** for the eigenvector \( w_i \) corresponding to eigenvalue \( \lambda_i \notin [\alpha, \beta] \) is defined as

\[
\tau(\lambda_i) = |\rho_i|^{-1} = \min \left| \frac{\lambda_i - c}{e} \pm \sqrt{\left( \frac{\lambda_i - c}{e} \right)^2 - 1} \right|
\]

The further away \( \lambda_i \) is from the interval \([\alpha, \beta]\) the smaller is \( |\rho_i|^{-1} \) and the faster the convergence to \( w_i \) is.

For a set of input vectors \( V = \{v_1, v_2, \ldots, v_{\text{nev}}\} \)

Claim

\[
\begin{align*}
\text{Res}(v^m_1) \lesssim & \quad \frac{|\text{const.}|}{|\rho_1|^m} \\
\text{Res}(v^m_i) \lesssim & \quad \frac{|\text{const.}|}{|\rho_i|^m} + |\text{const.}| \frac{|\rho_1|^m}{|\rho_i|^m} \langle \text{eps} \rangle ; \quad k \geq i \geq 2.
\end{align*}
\]
Predicting the minimal polynomial degree

\[
\text{Res}(v_i^m) = \text{Res}(v_i^{m_0}) \left| \frac{1}{\rho_i} \right|^{(m-m_0)}.
\]
Predicting the minimal polynomial degree

$$\text{Res}(v_i^m) = \text{Res}(v_i^{m_0}) \left| \frac{1}{\rho_i} \right|^{(m-m_0)} + \langle \text{eps} \rangle \left| \frac{\rho_1}{\rho_i} \right|^{(m-m_0)} \cdot \rho_i \left| \left( m-m_0 \right) \right|.$$
Optimizations: gains and losses

ChFSI sequential: optimized vs standard \(\rightarrow\) total CPU times.

ChFSI Total

![Graph showing ChFSI sequential: optimized vs standard total CPU times.](image-url)
Optimizations: gains and losses

ChFSI sequential: optimized vs standard → polynomial filter CPU times.
Optimizations: gains and losses

ChFSI sequential: optimized vs standard → Rayleigh-Ritz CPU times.

![Graph showing time in seconds for standard ChFSI vs optimized ChFSI](image-url)

- **Standard ChFSI vs Optimized ChFSI**
  - Time [sec]

- **Iteration Index**

- **Optimized (total time)**
- **Standard (total time)**
- **Optimized (filter time)**
- **Standard (filter time)**
- **Optimized (RR time)**
- **Standard (RR time)**

Rayleigh–Ritz

**NASCA 2013 Calais, France, June 24th**

M. Berljafa and E. Di Napoli

Folie 26
Conclusions and future work

- (Preconditioning) ChFSI with eigenvectors of $P^{(\ell-1)}$ to speed up 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 by using OpenACC directives;
References

1. EDN, and M. Berljafa
   *A Parallel and Scalable Iterative Solver for Sequences of Dense Eigenproblems Arising in FLAPW*
   Submitted to PPAM 2013. [arXiv:1305.5120]

2. EDN, and M. Berljafa
   *Block Iterative Eigensolvers for Sequences of Correlated Eigenvalue Problems*
   Accepted for publication in Comp. Phys. Comm. [arXiv:1206.3768].

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