Iterative eigensolvers in Abinit

Clémentine Barat

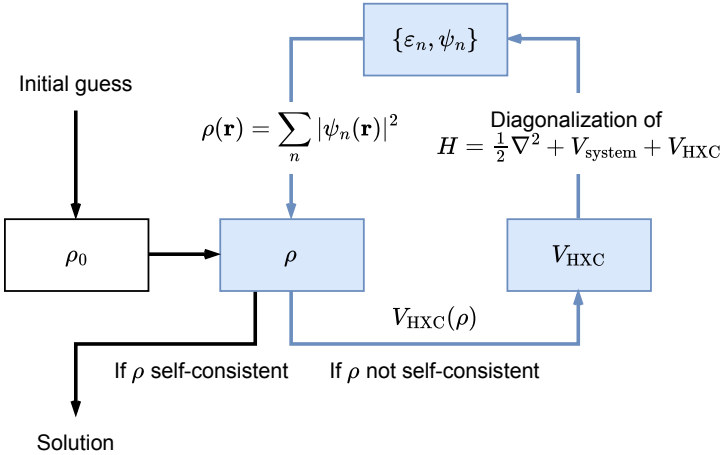
CEA, DAM, DIF, F-91297 Arpajon, France

Université Paris-Saclay, CEA, Laboratoire Matière en Conditions Extrêmes, 91680 Bruyères-le-Châtel, France

Feb. 4, 2026

Contents

Self-consistent field in DFT



Parallelization possibilities in plane-wave DFT

$$\rho(\mathbf{r}) = \sum_{\alpha\in\{\uparrow,\downarrow\}} \sum_{\mathbf{k}\in \mathrm{BZ}_M} \sum_{n=1}^{m} |\psi_n(\mathbf{r})|^2$$

	spins	k-points	bands
Parallelization level			
System-size scaling	2	$\propto 1/S$	$\propto S$
Parallel speedup	linear	linear	?

S: system size

Eigensolver in SCF iterations

The diagonalization is the computational bottleneck of the SCF iterations.

- Iterative, matrix-free eigensolver
- SCF and eigensolver iterations mixed

Algorithm 1 SCF iterations with an exact eigensolver

$\rho \leftarrow$ GUESSDENSITY
while ρ not converged **do**
  $E, \Psi \leftarrow$ EIGENSOLVER($H[\rho]$)
  $\rho \leftarrow$ MAKEDENSITY(Ψ)
end while

Algorithm 2 SCF iteration with an iterative eigensolver

$\Psi \leftarrow$ GUESSVECTORS
$\rho \leftarrow$ MAKEDENSITY(Ψ)
while ρ not converged **do**
  $E, \Psi \leftarrow$ EIGENSOLVERSTEP($H[\rho], \Psi$)
  $\rho \leftarrow$ MAKEDENSITY(Ψ)
end while

The diagonalization problem in DFT

In KSDFT we want to solve

$$H\Psi = \Psi E$$

with $E = \mathrm{Diag}(\varepsilon_1, \ldots, \varepsilon_m)$ diagonal

and $\Psi = (\psi_1|\psi_2|\cdots|\psi_m) \in \mathbb{C}^{N\times m}$ with orthonormal columns

m: Number of bands

N: Number of plane-waves

H: Discretized Hamiltonian

Two main families of iterative eigensolvers are implemented in Abinit to solve this problem:

- Minimization algorithms: An iterative minimization method is used to solve the equivalent minimization problem

$$\min_{X\in\mathbb{C}^{N\times m}} \|X^H H X\|$$
$$\text{s.t.} X^H X = I_m$$

- Filtering algorithms: The eigensubspace $\mathrm{Span}\{\psi_1, \ldots \psi_m\}$ is approximated iteratively with a filter.

The Rayleigh-Ritz method

The Rayleigh-Ritz method gives an approximation of the eigenvalues and eigenvectors of a matrix projected in a given subspace.

Algorithm 3 Rayleigh-Ritz

function RAYLEIGHRITZ(H, X)
  $X \leftarrow$ ORTHO(X)
  $R \leftarrow X^H H X$
  $E, Y \leftarrow$ EIGEN(R)
  $\Psi \leftarrow XY$
  return E, Ψ
end function

- $R = X^H \cdot H \cdot X$ (R: $m \times m$; H: $N \times N$; X: $N \times m$)
- E, Y full diagonalization of R $\longrightarrow$ $\mathcal{O}(m^3)$
- $\Psi = X \cdot Y$

Minimization algorithms for eigenvalue problems

- Eigenvalue problem: Find the m smallest eigenpairs $(\varepsilon_1, \psi_1), \ldots, (\varepsilon_m, \psi_m)$ in $\mathbb{R} \times \mathbb{C}^N$ such that for $1 \leq i \leq m$,

$$H\psi_i = \varepsilon_i \psi_i \tag{1}$$

- Constrained minimization problem:

$$\begin{aligned} &\min_{X \in \mathbb{C}^{N \times m}} \|X^H H X\| \\ &\text{s.t.} X^H X = I_m \end{aligned} \tag{2}$$

The solutions of (2) spans the same subspace as the solutions of (1).

Standard conjugate gradient algorithm for quadratic minimization problems

Quadratic optimization problem

$$\min_{x \in \mathbb{R}^N} q(x) = \frac{1}{2} x^T A x - bx$$

$A \in \mathbb{R}^{N \times N}$ symmetric positive definite
$b \in \mathbb{R}^N$

q can be independently minimized on a set of A-conjugated directions $(d_1, \ldots, d_N$ with $d_i^T A d_j = 0$ for $i \neq j)$.

Algorithm 4 Conjugate gradient

$x \leftarrow$ GUESS
$d \leftarrow -g(x)$ ▷ *line search direction* ▷ *$g(x) = \nabla q(x) = Ax - b$*
while x not converged **do**
  $\alpha \leftarrow$ ARGMIN$(\alpha \mapsto q(x + \alpha d))$ ▷ *minimize q along d*
  $x \leftarrow x + \alpha d$
  $d \leftarrow$ CONJUGATE$(-g(x))$ ▷ *$d \leftarrow -g(x) + \beta d$ such that d is A-conjugate to previous directions*
end while

Conjugate gradient algorithm for eigenvalue problems

Eigenvalue problem

$$\min_{x \in \mathbb{C}^N} \lambda(x) = \frac{x^H H x}{x^H x}$$

$H \in \mathbb{C}^{N \times N}$ hermitian

Locally around a minimizer, λ behaves like a quadratic function with gradient $g(x) = \frac{2}{\|x\|}(H - \lambda(x)I)x$.

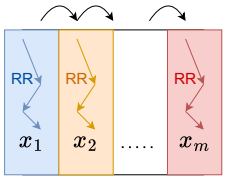
Algorithm 5 Conjugate gradient for eigenvalue problems

$x \leftarrow$ GUESS
$d \leftarrow -g(x)$
▷ *Now* $g(x) = 2/\|x\|(H - \lambda(x)I)x$
while x not converged **do**
$x \leftarrow$ RAYLEIGHRITZMIN$(H, \mathrm{Span}\{x, d\})$
▷ *the minimization of* λ *along* d *is a* 2×2 *eigenvalue problem*
3-terms variant :
$x \leftarrow$ RAYLEIGHRITZMIN$(H, \mathrm{Span}\{x, d, x^{\mathrm{old}}\})$
$d \leftarrow$ CONJUGATE$(-g(x))$
▷ $d \leftarrow -g(x) + \beta d$ *with a* β *that "conjugates"* d *to the previous direction*
end while

Projected conjugate gradient algorithm

Eigenvalue problem in KSDFT

$$\min_{X \in \mathbb{C}^{N \times m}} E(X) = \|X^H H X\|$$
$$\text{s.t.} \quad X^H X = I_m$$



Algorithm 6 Projected conjugate gradient for KSDFT

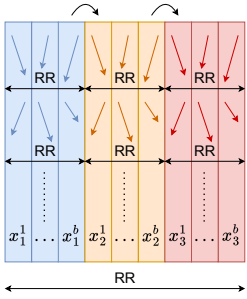
for $i = 1$ to m **do**
  $x_i \leftarrow$ GUESS
  $x_i \leftarrow$ ORTHOGONALPROJ$(x_i, \{x_1, \ldots, x_{i-1}\})$
  $d \leftarrow$ ORTHOGONALPROJ$(-g(x_i), \{x_1, \ldots, x_{i-1}\})$
    ▷ *line search direction d and initial guess orthogonal to previously computed directions*
  while x_i not converged **do**
    $x_i \leftarrow$ RAYLEIGHRITZMIN$(H, \mathrm{Span}\{x_i, d, x_i^{\mathrm{old}}\})$
    $d \leftarrow$ CONJUGATE$(-g(x_i))$
    $d \leftarrow$ ORTHOGONALPROJ$(d, \{x_1, \ldots, x_{i-1}\})$
  end while
end for

The $\{x_i\}$ directly approximate the eigenvectors $\{\psi_i\} \rightarrow$ no final Rayleigh-Ritz needed

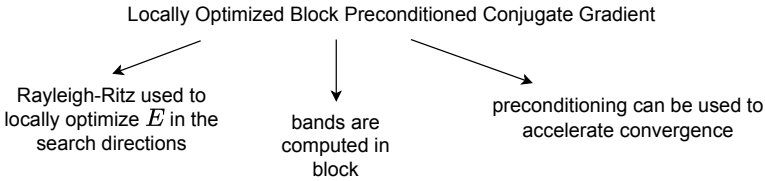
LOBPCG: Locally Optimized Block Preconditioned Conjugate Gradient

Algorithm 7 LOBPCG

for $i = 1$ to n_b **do** ▷*number of blocks = n_b*
  $X_i = (x_i^1 | \cdots | x_i^b) \leftarrow$ GUESS ▷*X_i of size $N \times b$ with b = block size*
  $X_i \leftarrow$ ORTHOGONALPROJ$(X_i, \{X_1, \ldots, X_{i-1}\})$
  for $j = 1$ to b **do**
    $d_j \leftarrow$ ORTHOGONALPROJ$(-g(x_i^j), \{X_1, \ldots, X_{i-1}\})$
  end for ▷*highly parallelizable*
  while X_i not converged **do**
    $X_i \leftarrow$ RAYLEIGHRITZMIN$(H, \mathrm{Span}\{X_i, X_i^{\mathrm{old}}, d_1, \ldots d_b\})$
      ▷*3 term variant → RayleighRitz in dimension $3b$*
    for $j = 1$ to b **do**
      $d_j \leftarrow$ CONJUGATE$(-g(x_i^j))$
      $d_j \leftarrow$ ORTHOGONALPROJ$(d_j, \{X_1, \ldots, X_{i-1}\})$
    end for ▷*highly parallelizable*
  end while
end for
$\psi_1, \ldots, \psi_m \leftarrow$ RAYLEIGHRITZ$(X_1, \ldots, X_{n_b})$



Notes on LOBPCG



- Parallelization possibilities in each block: The b new search directions can be computed in parallel
- One Rayleigh-Ritz in dimension $3b$ per iteration, per block
- **A balance must be found between the parallelization of the Hamiltonian application and the dimension of the Rayleigh-Ritz**
- One final Rayleigh-Ritz in dimension m for stability

Filtering methods

In subspace Filtering methods, we approximate directly the subspace $\mathcal{U} = \mathrm{Span}\{\psi_1, \dots, \psi_m\}$.

- Initialization : Initial (random) subspace of dimension m :

$$\mathcal{X}_0 = \mathrm{Span}\{x_1^0, \dots, x_m^0\}$$

- Iterations : Apply a filter $f : \mathbb{C}^N \to \mathbb{C}^N$ well chosen:

$$\mathcal{X}_{k+1} = f(\mathcal{X}_k) = \mathrm{Span}\{f(x_1^k), \dots, f(x_m^k)\}$$

  - ideally, f is the orthogonal projector on $\mathcal{U}$ such that $\mathcal{X}_1 = f(\mathcal{X}_0) = \mathcal{U}$.
  - in practice, we use polynomial filters $f = Q(H)$ with Q a polynomial:

$$f(x) = Q(H)x = \sum_{j=0}^{d} Q(\varepsilon_i)\langle \psi_i, x\rangle \psi_i$$

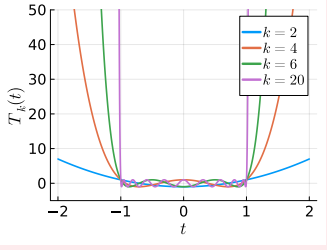
  $\to$ we want Q to be large on $[\varepsilon_1, \varepsilon_m]$ and small on $[\varepsilon_{m+1}, \varepsilon_N]$.

- Final step: Apply the Rayleigh-Ritz method to retrieve the eigenvectors from the subspace estimation

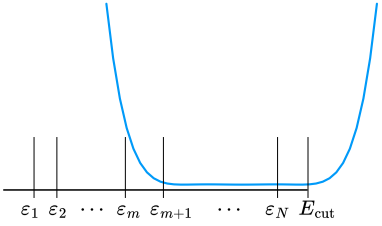
Chebyshev polynomials as filters: ChebFi

Chebyshev polynomials of the first kind T_k

- $T_k(x) \in [-1, 1]$ for $x \in [-1, 1]$
- T_k grows rapidly outside of $[-1, 1]$



Chebyshev polynomials are used as filter, by translating the unwanted part of the spectrum ($[\varepsilon_{m+1}, \varepsilon_N]$) to $[-1, 1]$.



ChebFi in the Self-Consistent Field

Algorithm 8 ChebFi

$\Psi \leftarrow$ GUESSVECTORS
$\rho \leftarrow$ MAKEDENSITY(Ψ)
while ρ not converged **do**
  $H \leftarrow H[\rho]$
  $X \leftarrow F_{\text{cheby}}(H)(\Psi)$
    ▷ *the filter can be applied in parallel to each vector*
    ▷ *degree of the Chebyshev polynomial = number of Hamiltonian application*
  $E, \Psi \leftarrow$ RAYLEIGHRITZ(H, X)
  $\rho \leftarrow$ MAKEDENSITY(Ψ)
end while

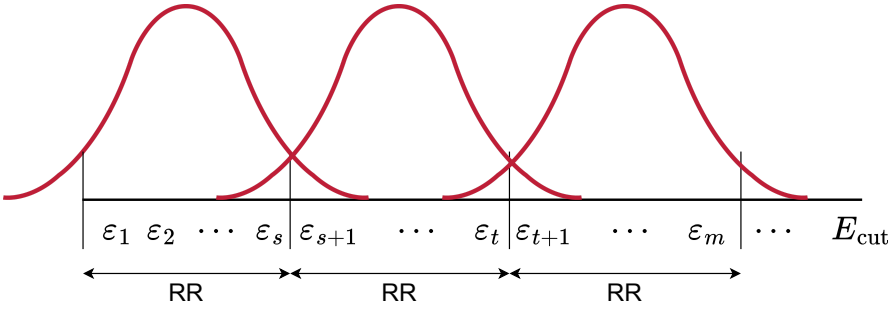
Notes :

- Parallelization possibilities: The filter can be applied in parallel to each vector.
- One Rayleigh-Ritz in dimension m per SCF iteration
- The last bands converge more slowly $\rightarrow$ compute more bands than necessary

To come: Spectrum Slicing

Eigensubspace $\mathrm{Span}\{\psi_1, \ldots, \psi_m\}$ is split into slices that are independently filtered.

One Rayleigh-Ritz in smaller dimension per slice



Compairison of LOBPCG and ChebFi

	LOBPCG	ChebFi
Cost per SCF iteration	$\mathcal{O}(n_{\text{block}} \cdot n_{\text{line}} \cdot (T_H \cdot \frac{m}{n_{\text{block}}} + (3\frac{m}{n_{\text{block}}}))^3 + m^3)$	$\mathcal{O}(n_{\text{deg}} \cdot T_H \cdot m + m^3)$

terms in red can be reduced to 1 with parallelization

T_H *: cost of one Hamiltonian application*

Using MPI parallelism with Abinit

- Job

  Number of MPI processes

  ```
  #MSUB -n <num_procs>
  mpirun -n <num_procs> ...
  ```

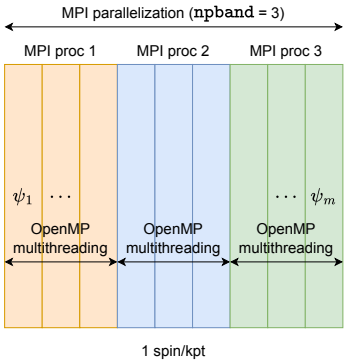
- Abinit input file

  `paral_kgb`
  0 : spin/k-point parallelization only
  1 : spin/k-point & band parallelization

  `np_spkpt`
  number of MPI processes for k-point/spin parallelization

  `npband`
  number of MPI processes for band parallelization

Total number of MPI processes
`<num_procs>` = `np_spkpt` × `npband`



Guidelines: First parallelize over spins and k-points, then over bands

Using OpenMP parallelism with Abinit

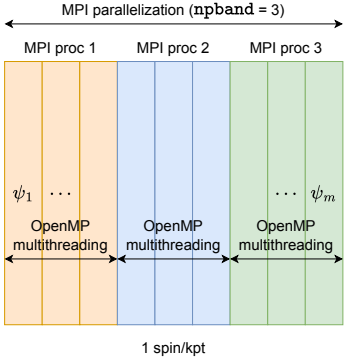
- Job

  Number of threads `#MSUB -c <num_threads>`

- Environment variables

```
export OMP_NUM_THREADS=<num_threads>
```

Guidelines:
Hybrid (MPI+OpenMP) band parallelization is more efficient. `<num_thread>` should be smaller or equal to the number of core per node.



Using GPU parallelism with Abinit

- Abinit input file

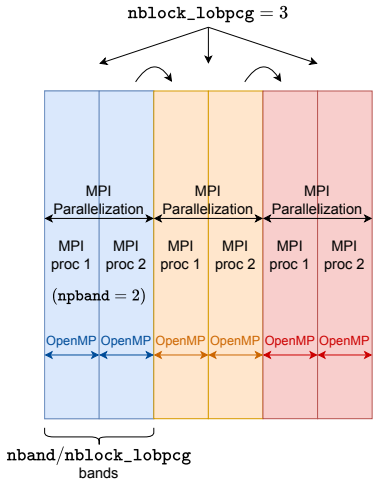
`gpu_option`	0 or `GPU_DISABLED` : no use of GPU 2 or `GPU_OPENMP` : use GPU (OPENMP_OFFLOAD programming model)

Guidelines:
Use one MPI process per GPU & no OpenMP multithreading.

Using LOBPCG in Abinit

- `wfoptalg` : set to 114 to select LOBPCG.
- `nblock_lobpcg` : number of blocks in LOBPCG
- `nline` : number of local searches (Rayleigh-Ritz) per SCF step.

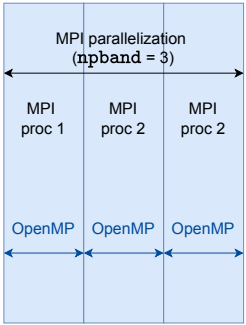
`nband`/`nblock_lobpcg` should be a multiple of `npband` $\times$ <num_threads> for an optimal band parallelization.



Using ChebFi in Abinit

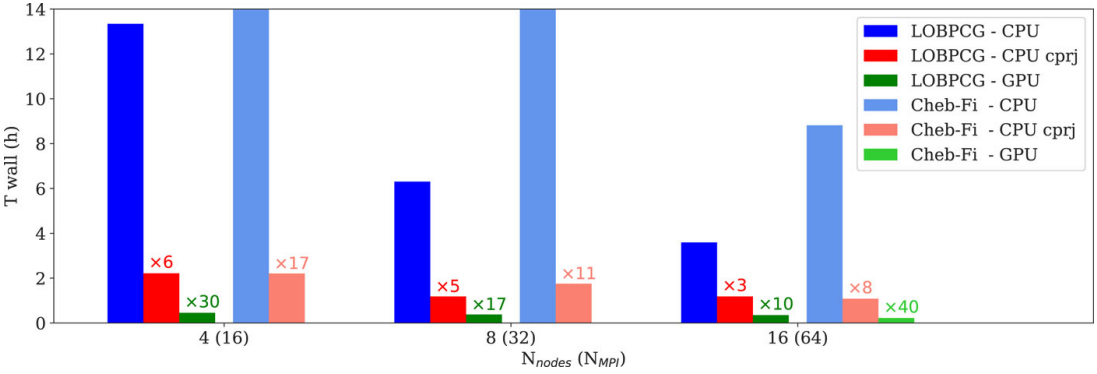
- `wfoptalg` : set to 111 to select ChebFi
- `mdeg_filter` : maximum degree of the Chebyshev polynomial
- `nbdbuf` : number of (highest energy) bands whose convergence won't be checked

`nband` should be a multiple of `npband` $\times$ `<num_threads>` for an optimal band parallelization.

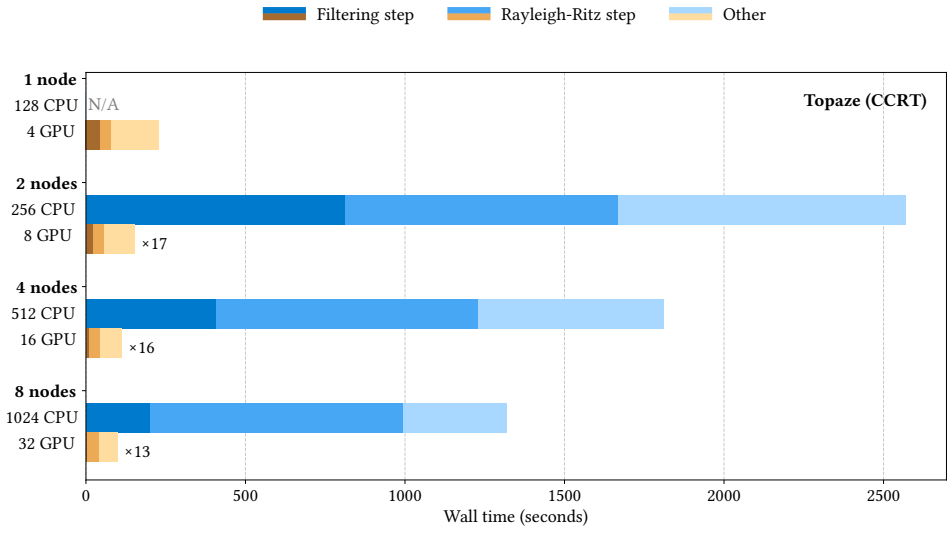


LOBPCG vs ChebFi

1280-atom crystal of Ga_2O_3 with 6144 electronic bands



CPU vs GPU


Filtering step
Rayleigh-Ritz step
Other
Topaze (CCRT)
1 node
128 CPU
4 GPU
N/A
2 nodes
256 CPU
8 GPU
×17
4 nodes
512 CPU
16 GPU
×16
8 nodes
1024 CPU
32 GPU
×13
0
500
1000
1500
2000
2500
Wall time (seconds)

Thank you for your attention !