Towards scalable subspace iteration algorithms: Spectrum slicing

Ioanna-Maria Lygatsika[1,2], Clémentine Barat[1,2,3], Marc Torrent[1,2]

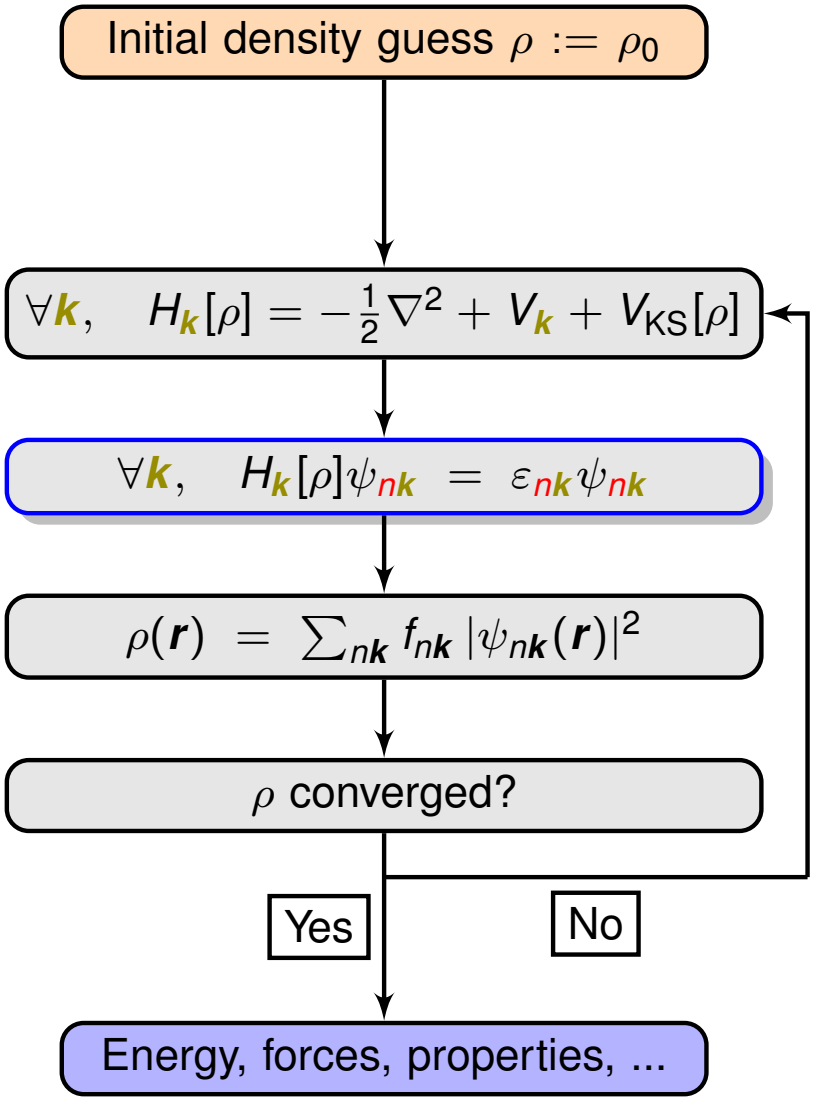
[1]CEA, DAM, DIF, F-91297 Arpajon, France

[2]Laboratoire Matière en conditions extrêmes, Paris-Saclay University, France

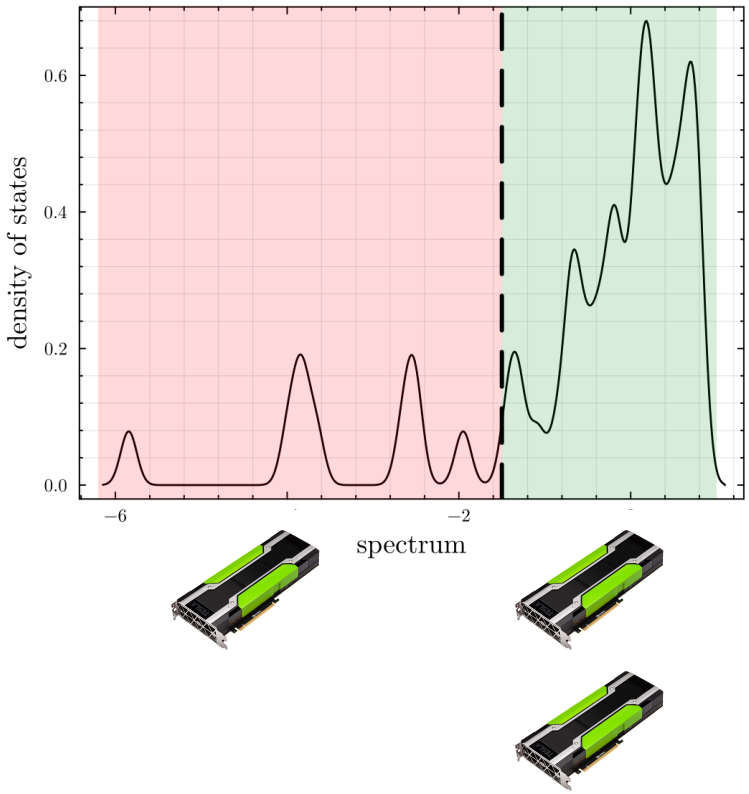
[3] Laboratoire de Mathématiques d'Orsay, Université Paris-Saclay

Abinit Developer Workshop 2024

Exploiting parallelism over bands



- all states are labeled by $1 \leq n \leq$ nband band index and $\boldsymbol{k}$ Bloch vector
- parallelism over $\boldsymbol{k}$-points [OK]
- fixed $\boldsymbol{k}$, parallelism over npw [OK]
- fixed $\boldsymbol{k}$, parallelism over nband



Building blocks of any diagonalisation scheme

$A \in \mathbb{C}^{\text{npw} \times \text{npw}}$ Hermitian matrix, seek the lowest nband $\ll$ npw eigenpairs

What all methods have in common: **Subspace Iteration**

1. Initialise vector space generated by starting vectors $X_0 \in \mathbb{C}^{\text{npw} \times \text{nband}}$, $j = 0$;
2. Update vectors X_{j+1} by diagonalisation in subspace, $j = j + 1$;
3. Iterate until $\|AX_j - \Lambda_j X_j\| < tol$.

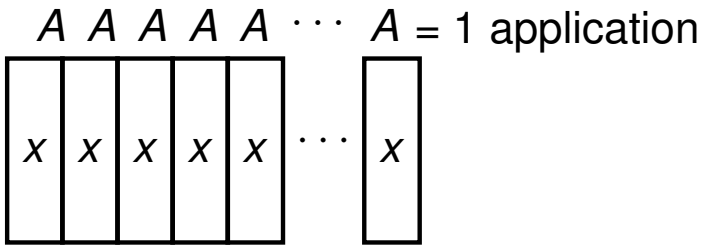
How methods differ:

- Cost of projecting columns of H on subspace.
- Size of diagonalisation problem on subspace.

Hamiltonian application

$$AX$$

batch processing on GPU



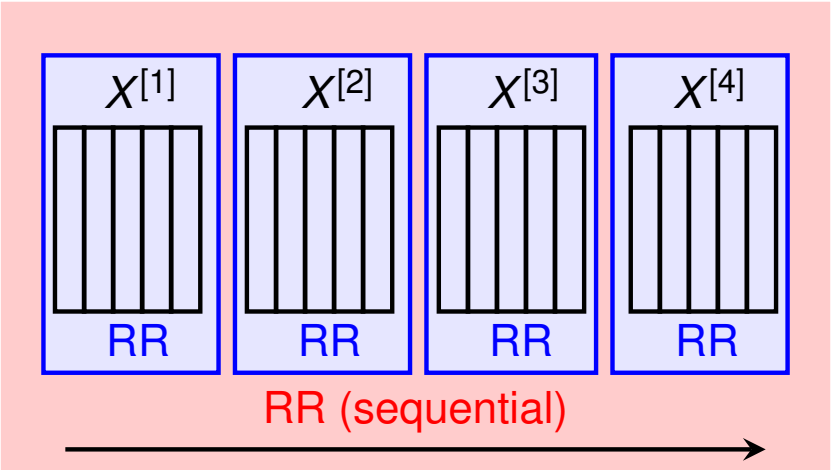
Rayleigh-Ritz

Input: matrix A, initial guess X
Output: approximate eigenpairs of A

1. Orthogonalize X;
2. Project to subspace $B = X^\top AX$;
3. Diagonalize $BY = \Lambda Y$ (**dense**); $\mathcal{O}(n^3)$
4. Return rotated subspace $X' = Y^\top X$.

What we wish to avoid: global Rayleigh-Ritz

Method 1: **Locally Optimal Block Conjugate Gradient** = approximate eigenvectors based on energy minimisation over vectors. Example for nblock= 4:

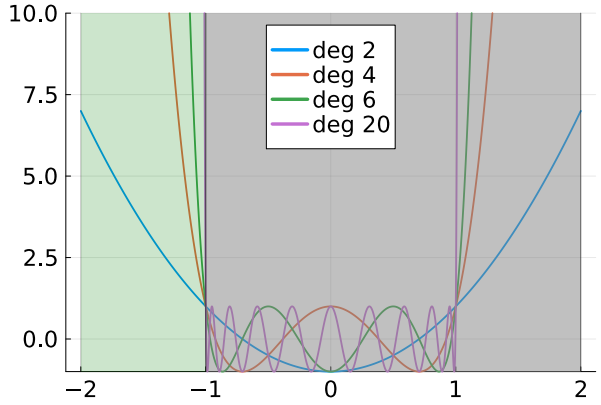


$$X_{j+1} = \mathrm{RR}(X_{j-1}, X_j, AX_j - \Lambda_j X_j)$$

$$\boxed{\text{nblock} \times \left(\frac{3 \times \text{nband}}{P \times \text{nblock}} + \frac{(3 \times \text{nband})^3}{\text{nblock}^3} \right) + \text{nband}^3}$$

Method 2: **Chebyshev filtering** = polynomial filtering based on *scalars* instead of RR per block.

$$\overset{f_n(A)}{\cancel{A^n}} X = \Lambda^n X$$



$$X_{j+1} = f_n(A) X_j$$

$$\boxed{\text{deg} \times \frac{\text{nband}}{P} + \text{nband}^3}$$

What we wish to achieve: divide and conquer

1. Partition the spectrum into subintervals (=*slices*), for evolving matrices (i SCF iter):
   - First few SCF iterations Chebyshev filtering;
   - Compute spectrum bounds: use Ritz values

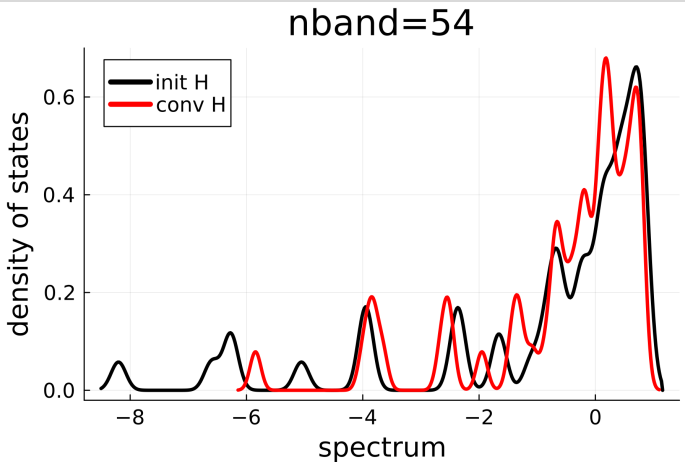
$$\mu_0^{(i)} = \mathrm{diag}(\langle X_{\mathrm{tol}}^{(i-1)}, A^{(i)} X_{\mathrm{tol}}^{(i-1)} \rangle),$$

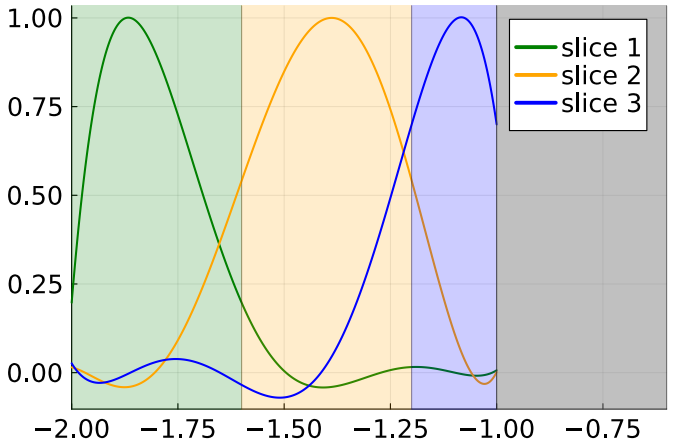
$$L = \min_{\mathrm{nband}} (\mu_0^{(i)} - \| A X_{\mathrm{tol}}^{(i-1)} - \mu_0^{(i)} X_{\mathrm{tol}}^{(i-1)} \|),$$

$$U = \max_{\mathrm{nband}} (\mu_0^{(i)} + \| A X_{\mathrm{tol}}^{(i-1)} - \mu_0^{(i)} X_{\mathrm{tol}}^{(i-1)} \|);$$

   - Cut $[L, U)$ into **nsli** "balanced" slices: bounds $[\ell_k^{(i)}, u_k^{(i)})$ and vectors $m_k^{(i)}$ per slice.
2. Build subspace $X_{\mathrm{tol}}^{(i,k)} \in \mathbb{C}^{\mathrm{npw} \times m_k^{(i)}}$ per slice separately (filter + RR);
3. Combine occupied states

$$\mu_{\mathrm{tol}}^{(i)} = \bigcup_{\substack{1 \le k \le \mathrm{nsli} \\ 1 \le j \le m_k^{(i)}}} \{ \mu_{\mathrm{tol},j}^{(i,k)} \in [\ell_k^{(i)}, u_k^{(i)}) \}$$





$$\boxed{\frac{\mathrm{nslice}}{P} \times \left(\mathrm{deg} \times \frac{\mathrm{nband}}{\mathrm{nslice}} + \frac{\mathrm{nband}^3}{\mathrm{nslice}^3} \right)}$$

Challenge of spectrum slicing

Assume we have nproc processing units.

Problem: how to assign

- vectors (and subspace dimension) to slices,
- processing units to slices,
- interval bounds to slices,

while assuring that each processor has the **same** amount of work.

$$\text{slice } 1 \mapsto X_1, \text{nproc}_1, \ell_1, u_1$$

$$\text{slice } 2 \mapsto X_2, \text{nproc}_2, \ell_2, u_2$$

$$\text{slice } 3 \mapsto X_3, \text{nproc}_3, \ell_3, u_3$$

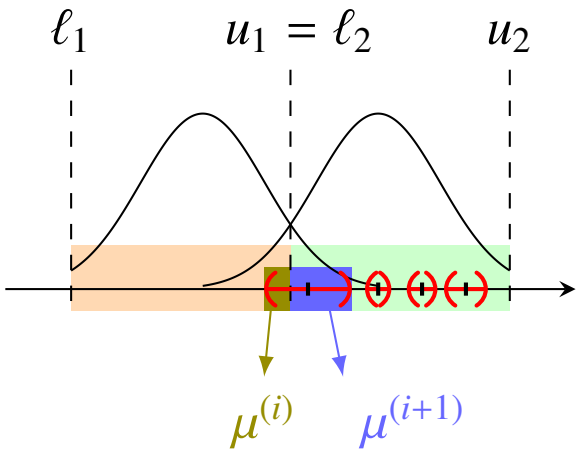
$$\dots$$

Issue 1: Missing eigenpairs

Building the subspace (Saad 2016):

$$\dim\Big(\underbrace{\sum_{j=1}^{m} \psi_j \psi_j^\top}_{\text{projector}} A^n \underbrace{\mathrm{Span}(X_0)}_{\text{initial guess}}\Big) = m$$

for any n ($\Rightarrow$ convergence) if true for $n = 0$, otherwise **missing dimensions**.



Difficulties:

- poorly chosen starting iteration
- number of true vectors m per slice is not known *a priori*
- eigenvalue "jump": a Ritz value outside a slice may be approximating a value inside the slice

$$(\mu - r, \mu + r), \quad r := \|Av - \mu v\|$$

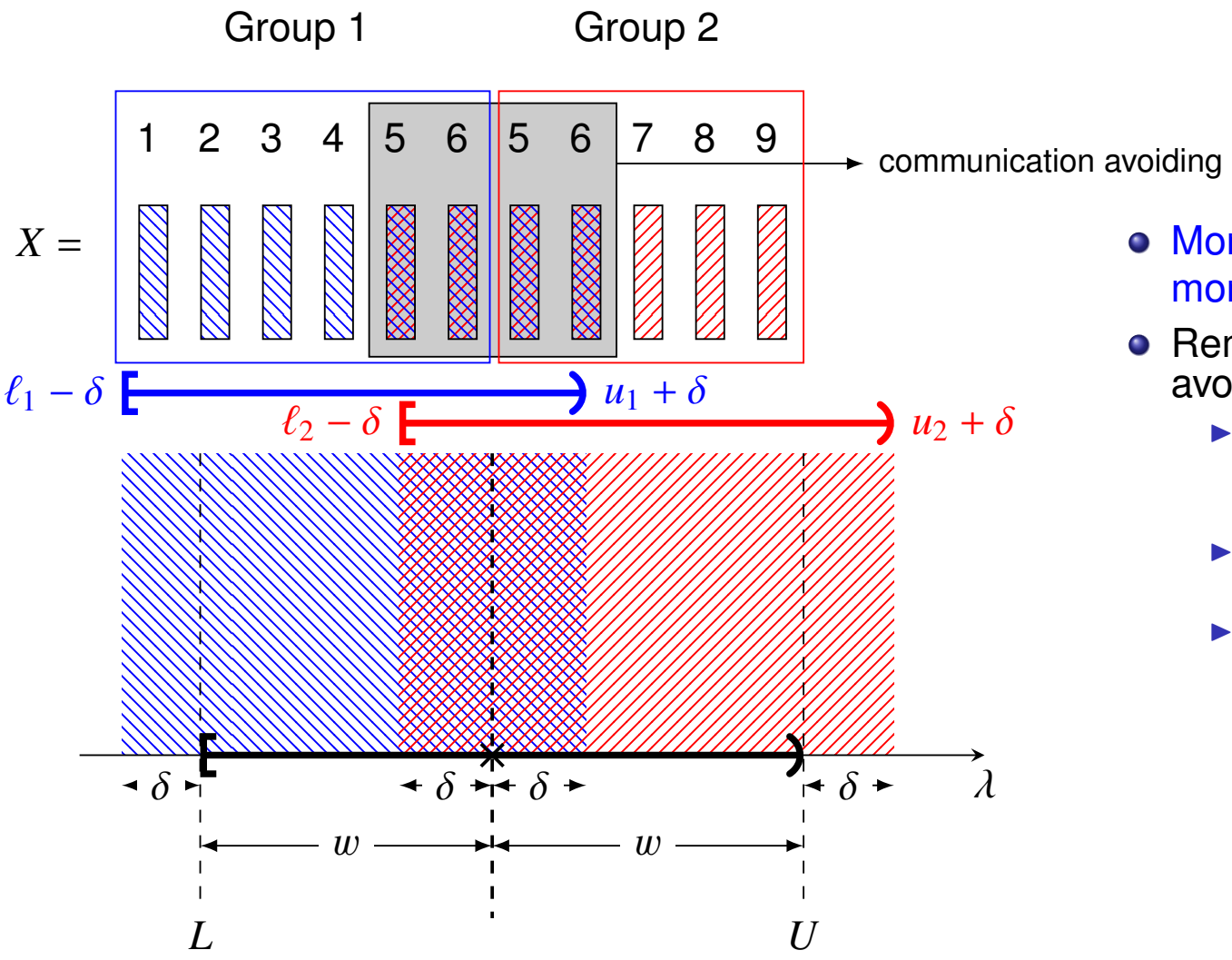
Solution:

- remove duplicates by overlapping slices (Liou et al. 2020)

$$[\ell_i - \delta, u_i + \delta), \quad \delta = \frac{u_i - \ell_i}{10}, \quad 1 \le i \le \text{nsli}$$

- require full slices using condition $\langle v, Av \rangle > u + \delta$ (Schofield et al. 2012)

Issue 2: Processor groups



- More vectors in slice ⇒ more procs in slice.
- Remove duplicates while avoiding:
  - communication of converged vectors near interval boundaries
  - re-orthonormalisation near interval boundary
  - redefinition of MPI communicators?

Issue 3: Load balancing

- Optimize slices (ℓ_i, u_i, m_i): minimise

$$\max(m_i \times \deg_i, m_i^3)$$

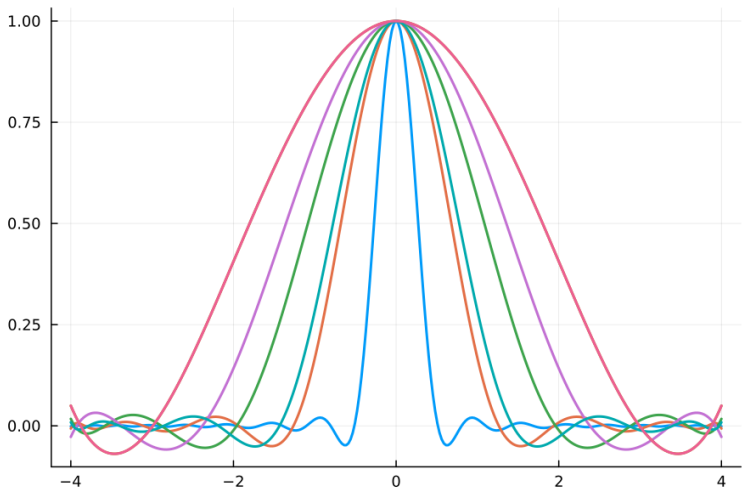
- Convergence rate:
  - **Optimize** m: choose dimension p of the subspace larger than m

$$\frac{|\lambda_m|}{|\lambda_{p+1}|} \ll \frac{|\lambda_m|}{|\lambda_{m+1}|}$$

  - **Optimize deg**: $\alpha > 0$ amplification factor, choose n (Schofield et al. 2012)

$$\min\left(\frac{f_n(\ell_i)}{f_n(\ell_i - \delta)}, \frac{f_n(u_i)}{f_n(u_i + \delta)}\right) \geq \alpha$$

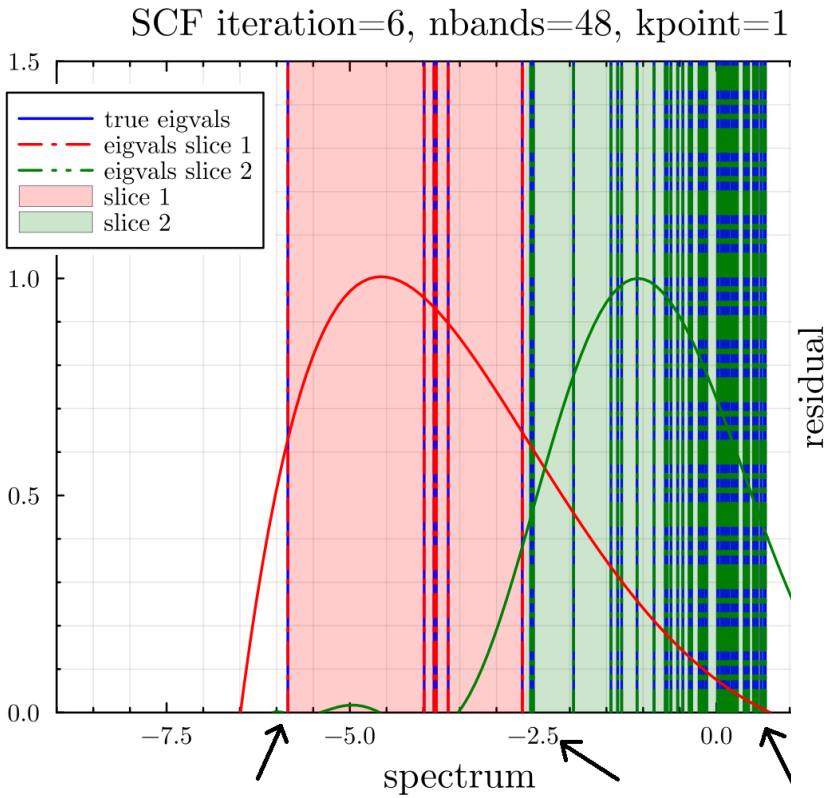
  - Work imbalance: lock converged vectors as lower eigenvalues converge faster

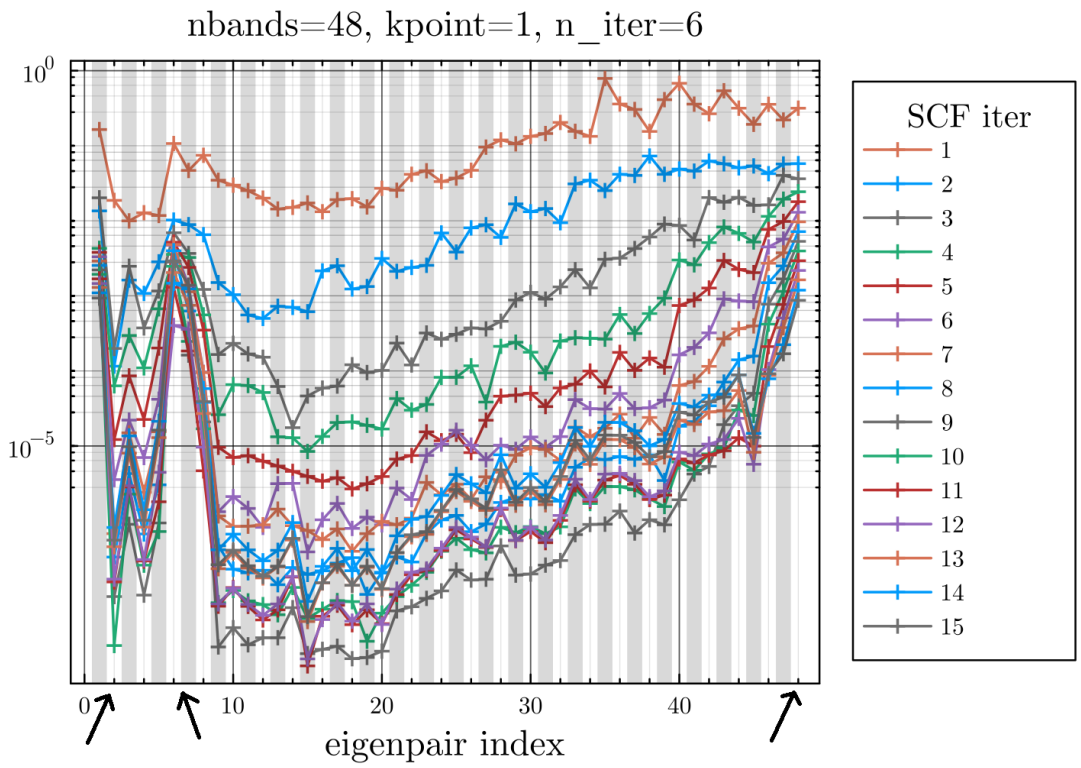


w	0.6	1.6	2.6	3.6	4.6
deg	39	15	9	7	5

Numerical illustration on a toy system

FeNiF6, 48 occupied states, 76 electrons, 1 *k*-point.





	LOBPCG	Cheby	Slicing
SCF iter	20	19	85
Degree	-	10	8,20
Vectors	48	48	6,42

iter	Degree		Vectors	
	s1	s2	s1	s2
1	7	17	8	40
2	8	19	8	40
3-6	8	20	5	43
7-85	8	20	6	42

Sum up: key points of spectrum slicing algorithm

Input : matrix A at current SCF iteration, slice bounds and number of vectors per slice, starting guess X
Output: approximation of N lowest eigenpairs

```
for k = 1, ..., nslice ;                                   // parallel nslice/P
 do
    Initialize J_k ∈ ℕ^{m_k} and Y := X[:, J_k] ;
    Center filter on slice [ℓ_k, u_k) ;
    for m = 2, ..., maxit do
        Apply filter Ynew = f_n(A) Y ;
        Orthonormalise Ynew ;
        Apply Rayleigh-Ritz to Ynew for m_k eigenpairs ;   // Rayleight-Ritz size m_k
        Y = Ynew ;
    X[:, J_k] = Y ;
return X ;
```

Difficulty: subspace dimension is less obvious than scalars

Spectrum slicing =

"slice $\mapsto$ interval"		"slice $\mapsto$ dimension" careful ortho		
parallelism over scalars	+	parallelism over subspace dimension	+	evolving matrix

To be implemented in ABINIT.