JARA | HPC

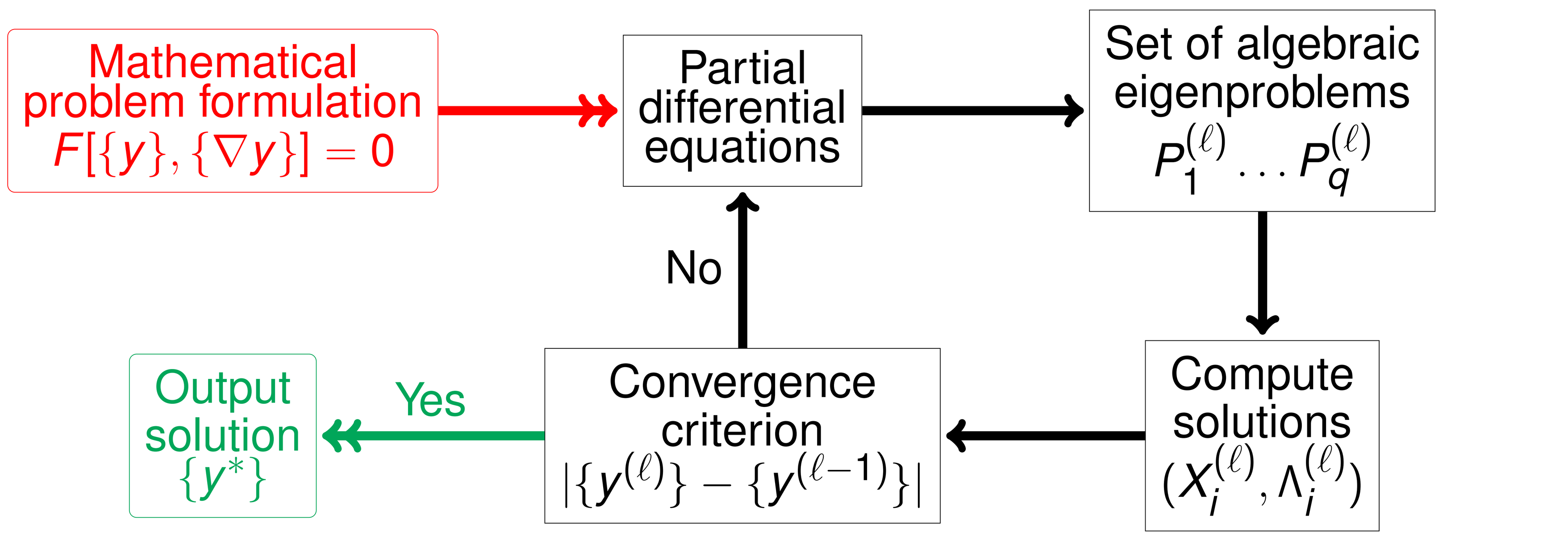
AN INITIATIVE OF RWTH AACHEN UNIVERSITY, JÜLICH FORSCHUNGSZENTRUM

ChASE — CHEBYSHEV ACCELERATED SUBSPACE EIGENSOLVER

Chebyshev Accelerated Subspace Eigensolver

Jan Winkelmann, Paul Springer, Mario Berljafa, and ***Edoardo Di Napoli***

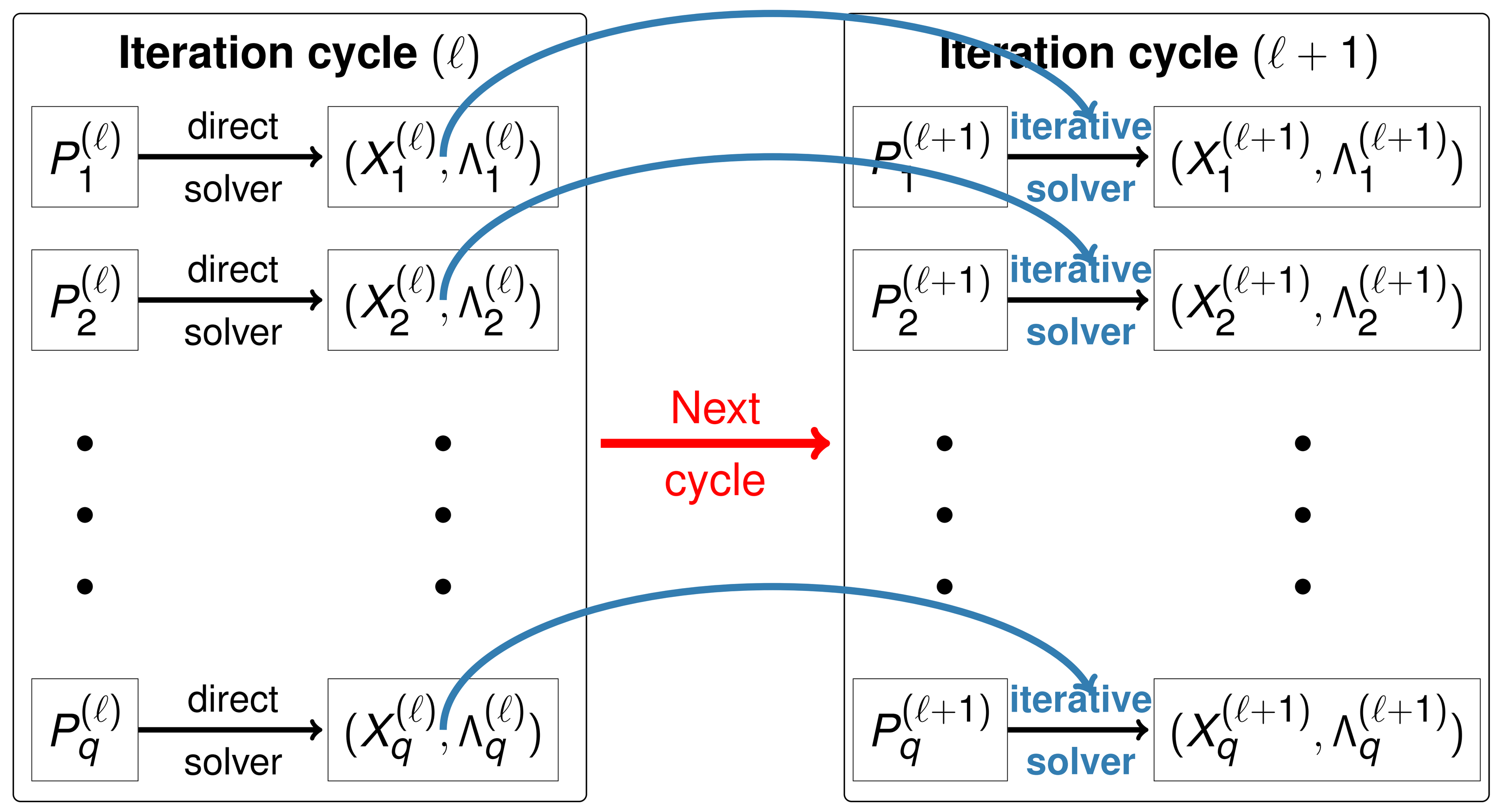
Sequences of Eigenvalue problems



Traditional solving strategy

- Every set $P^{(1)} \ldots P^{(\ell)} P^{(\ell+1)} \ldots P^{(M)}$ is handled as a set of M disjoint eigenproblems with overall complexity $\sim M \times N^3$;
- Each problem $P^{(\ell)}$ is solved independently using a direct solver as a black-box from a standard library (e.g. ScaLAPACK).

Knowledge oriented solving strategy



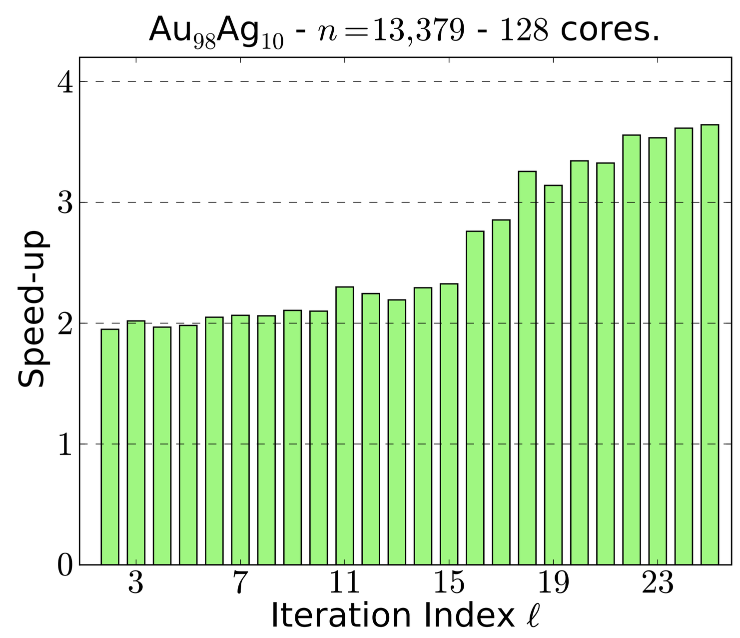
$X \equiv \{x_1,\ldots,x_N\}$ $\Lambda \equiv \mathrm{diag}(\lambda_1,\ldots,\lambda_N)$

Algorithm 1 Chebyshev Accelerated Subspace Eigensolver

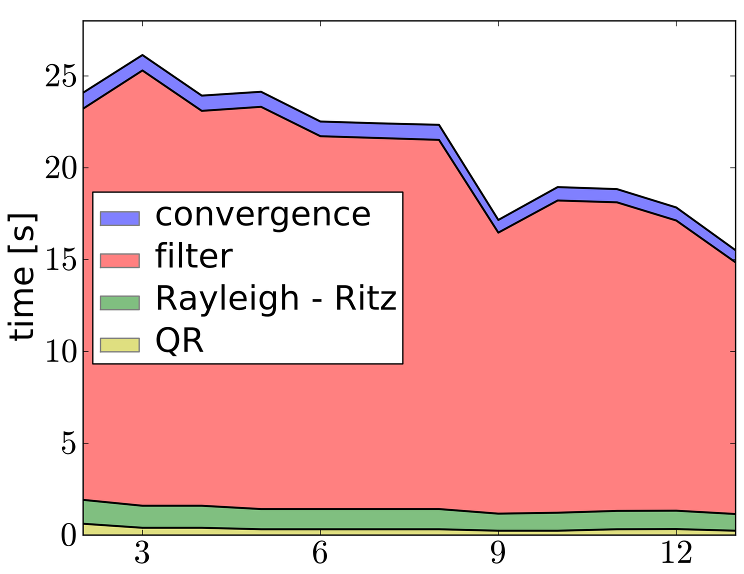
Input: Matrix $A^{(\ell+1)}$, vector matrix $X^{(\ell)}$, values $\Lambda^{(\ell)}$, eigenpairs nev, tolerance tol, poly degree deg, search space nex, parameter opt.

Output: nev extremal eigenpairs $(\Lambda^{(\ell+1)}, X^{(\ell+1)})$.

```
Initialize V = X^(ℓ), μ1 = λ1^(ℓ), μnev+nex = λnev+nex^(ℓ)     ▷ WORKING VARIABLES
(b_sup > λ_N) ← LANCZOS(A^(ℓ+1))                                ▷ SPECTRAL BOUND
Set the degrees m =: (m1, ..., mnev) = (deg, ..., deg).
while nconv =: size(X^(ℓ+1)) < nev do
    V ← FILTER(A^(ℓ+1), b_sup, μ1, μnev+nex, V, m)              ▷ USES ARRAY OF DEGREES
    Reorthonormalize [X^(ℓ+1) V] = QR                           ▷ QR ALGORITHM
    Q ← [Q:,nconv+1 ... Q:,nev+nex]                              ▷ REDUCE TO ACTIVE SUBSPACE
    Compute Rayleigh quotient G = Q† A^(ℓ+1) Q                   ▷ RAYLEIGH-RITZ (Start)
    Solve the reduced problem GW = WΛ̃                            ▷ DIRECT SOLVER
    Compute V = QW                                               ▷ RAYLEIGH-RITZ (End)
    Lock converged eigenpairs into (Λ^(ℓ+1), X^(ℓ+1))            ▷ LOCKING
    Reducing (search space) vector matrix V                      ▷ DEFLATION
    Update variabless μ1 = λ̃1 , μnev+nex = λ̃nev+nex
    Compute the degrees (m1, ..., ma) = m                        ▷ OPTIMIZATION
end while
```



Above: Speed-up after pre-conditioning
Below: ChASE time profile.



Algorithm 2 Optimized Chebyshev Filter

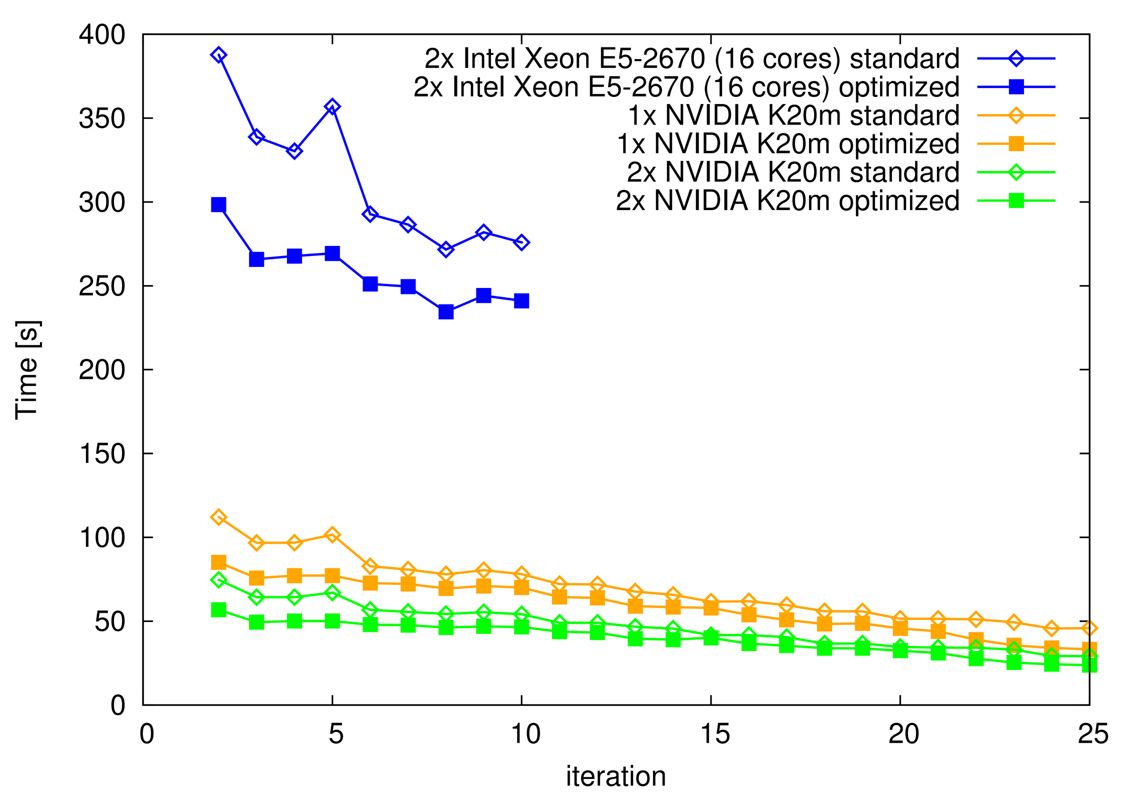
```
procedure FILTER(A^(ℓ+1), b_sup, μ1, μnev+nex, V, m = {m1, ..., mk})
    c = (b_sup + μnev+nex)/2;  e = (b_sup − μnev+nex)/2;
    σ1 = e/(μ1 − c);
    V ← (σ1/e)(A − c I_n) V;
    s ← argmin_{a=1,...,k} m_a ≠ 1
    for i = 1, ..., m_k − 1 do
        σ_{i+1} ← 1/(2/σ1 − σ_i)
        V_{s:n} ← Z_{s:n}
        Z_{s:n} ← 2(σ_{i+1}/e)(A − c I_n) V_{s:n} − σ_{i+1} σ_i V_{s:n}
        s ← argmin_{a=s,...,k} m_a ≠ i + 1
    end for
    return V ← Z
end procedure
```

$$Z \leftarrow 2\frac{\sigma_{i+1}}{e}\,(A - cI) \times V - \sigma_{i+1}\sigma_i\, V$$

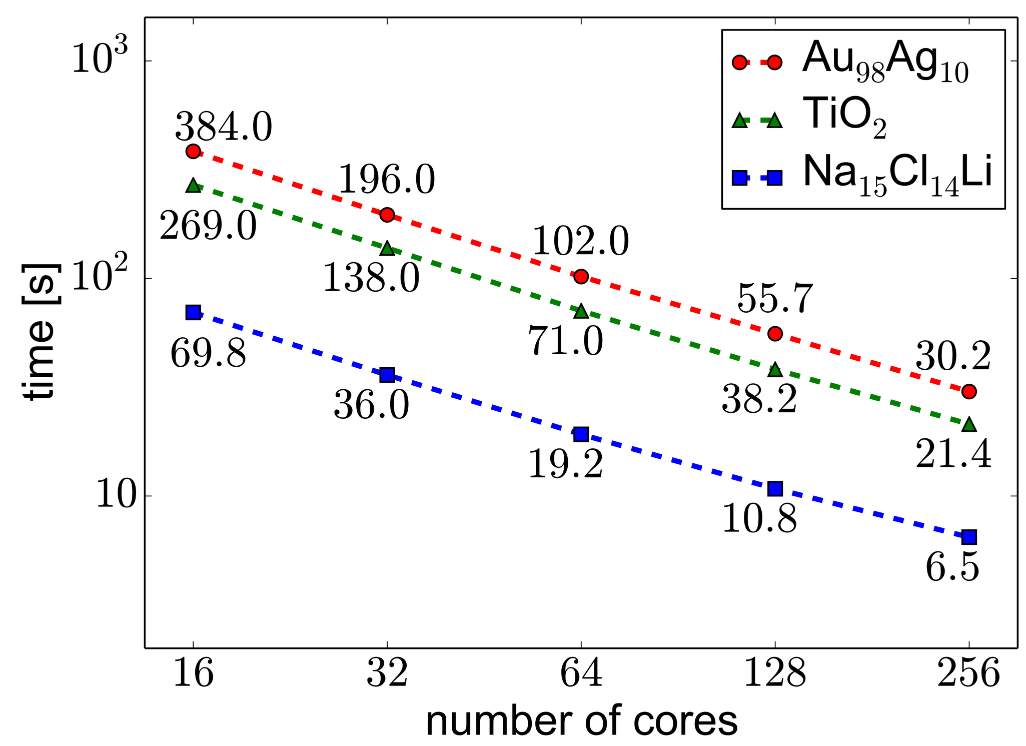
xGEMM

C++ implementations of ChASE

- Multi-core: Multi-threaded BLAS and OpenMP;
- Distributed: MPI based on the `Elemental` library;
- Many-core: CUDA for one or multiple GPUs;
- Distributed CPU/GPU — work in progress.



Time vs sequence index ℓ on multi- and many-cores.



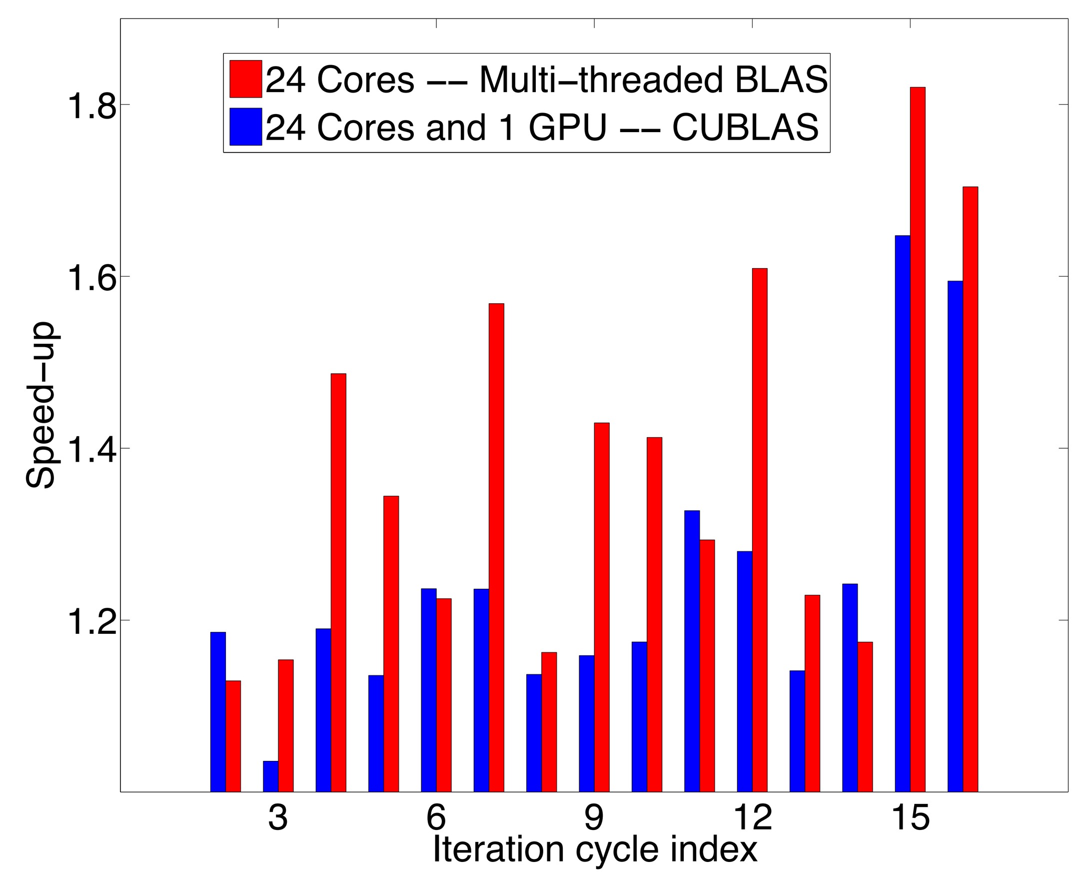
Strong scalability of the MPI implementation
System sizes: $N = 13{,}379 - 12{,}455 - 9{,}273$.

The ChASE library

- Release date: 30th of June 2016;
- Documention: generated through Sphinx;
- License: To Be Determined.

An array of optimal polynomial degrees

Computed using convergence ratio estimates, it minimizes `mat-vec` multiplications.



Speed-up after degree optimization.

[1] J. Winkelmann, M. Berljafa, P. Springer, and E. Di Napoli. *A Chebyshev accelerated subspace iteration eigensolver for sequences of eigenvalue problems.* In preparation.

[2] E. Di Napoli. *Chebyshev acceleration of subspace iteration revisited: convergence and optimal polynomial degree.* In preparation.

[3] M. Berljafa, D. Wortmann, and E. Di Napoli. Concurrency Computat.: Pract. Exper. 27 (2015), pp. 905-922

[4] E. Di Napoli, and M. Berljafa. Comp. Phys. Comm. 184 (2013), pp. 2478-2488