A Jacobi Davidson Method with a Multigrid Solver for the Hermitian Wilson-Dirac Operator

Artur Strebel

Bergische Universität Wuppertal

August 3, 2016

Joint Work

This project is joint work with:

Gunnar Bali[*], Sarah Collins[*], Andreas Frommer[†], Karsten Kahl[†], Matthias Rottmann[†], Benjamin Müller[+], Jakob Simeth[*]

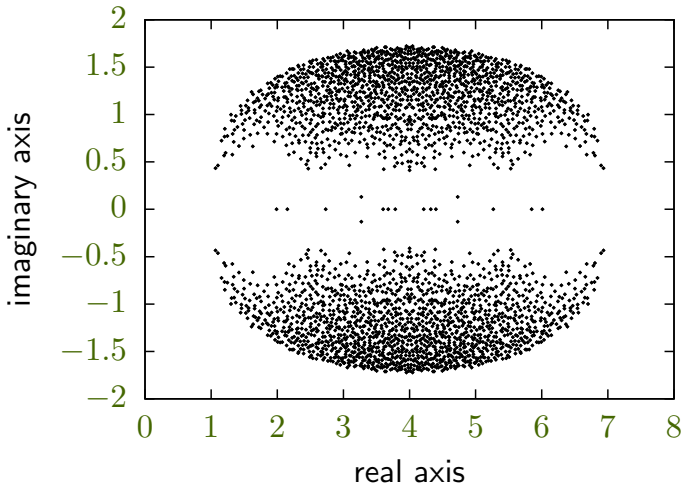
[*]: Universität Regensburg
[†]: Bergische Universität Wuppertal
[+]: Johannes Gutenberg-Universität Mainz

Outline

Lattice QCD: The Wilson Discretization

FD Discretization:

- $N_s^3 \times N_t$ regular lattice $\mathcal{L}$
  (parallelized by distributing sub-blocks)
- nearest neighbor coupling
- 12 variables per site and four 12×12 coupling matrices
- periodic boundary conditions
- γ_5-symmetry: $\gamma_5 D = (\gamma_5 D)^\dagger$
  ($\gamma_5 := \gamma_1\gamma_2\gamma_3\gamma_4,\ \gamma_5^2 = I$)



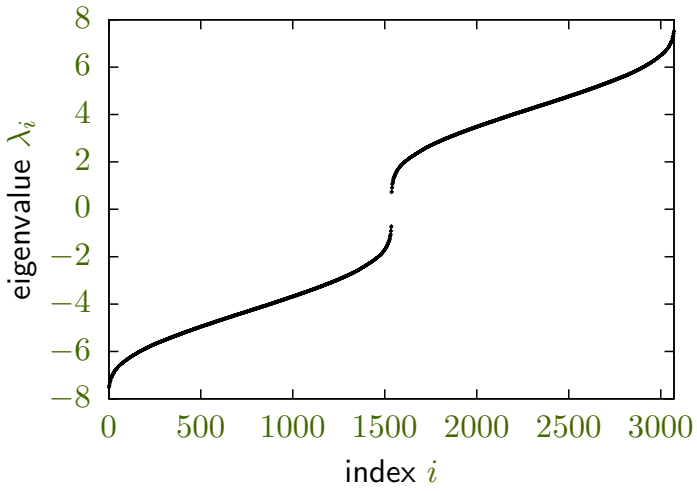
$\mathrm{spec}(D)$

$\mathrm{spec}(D)$ is in the right half plane and symmetric to the real axis

Lattice QCD: The Wilson Discretization

FD Discretization:

- $N_s^3 \times N_t$ regular lattice $\mathcal{L}$
  (parallelized by distributing sub-blocks)
- nearest neighbor coupling
- 12 variables per site and four 12×12 coupling matrices
- periodic boundary conditions
- γ_5-symmetry: $\gamma_5 D = (\gamma_5 D)^\dagger$
  ($\gamma_5 := \gamma_1\gamma_2\gamma_3\gamma_4,\ \gamma_5^2 = I$)



$\mathrm{spec}(\gamma_5 D)$

$\mathrm{spec}(\gamma_5 D)$ is real and maximally indefinite, we denote

$$Q := \gamma_5 D$$

Lattice QCD: Tasks

Application:

- calculating physical observables
  - oftentimes requires $\mathrm{tr}(Q^{-1})$
  - $\mathrm{tr}(Q^{-1}) = \sum \lambda_i^{-1}$
  - compute small eigenvectors exactly
  - treat remainder of spectrum with stochastic algorithm

Challenges:

- todays matrix sizes: $10^7 \times 10^7$ and larger
- (shifted) linear systems with either D or Q to be solved
- small EVs of Q ($=$ singular vectors of D) highly desired

Approaches:

- multigrid approaches for inverting D and Q
- Jacobi-Davidson method $+$ multigrid approach for Q
  $\rightarrow$ cheap eigensolver

Adaptive Algebraic Multigrid Approach for $D^{(\text{non-Hermitian})}$

Two-grid error propagator for ν steps of post-smoothing

$$E_{2g}^{(\nu)} = \underbrace{(I - MD)^{\nu}}_{\text{smoother}} \underbrace{(I - PD_c^{-1}P^{\dagger}D)}_{\text{coarse grid correction}}, \underbrace{D_c := P^{\dagger}DP}_{\text{coarse operator}}$$

- low accuracy for D_c^{-1} and M is sufficient
- introduce recursive construction for $D_c \rightarrow$ multigrid

To Do: Define interpolation P and smoother M

DD-αAMG[ArXiv:1303.1377,1307.6101]

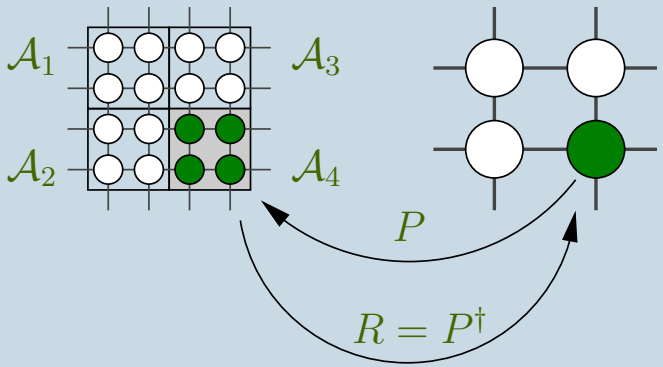
M: Schwarz Alternating Procedure (SAP)
[Lüscher 2003]

P: Aggregation based Interpolation
[Brannick, Clark et al. 2010]

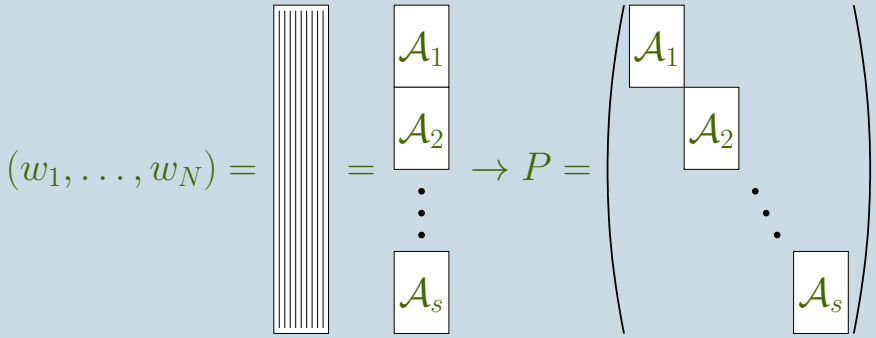
Aggregation Based Interpolation

Construction

- define aggregates: domain decomposition $\mathcal{A}_1, ..., \mathcal{A}_s$



- calculate test vectors $w_1, ..., w_N$
- decompose test vectors over aggregates $\mathcal{A}_1, ..., \mathcal{A}_s$



Adaptive Algebraic Multigrid Approach for $Q = \gamma_5 D$

Construct P such that $P\gamma_5 = \gamma_5 P$
Replacing $D \to Q = \gamma_5 D$ we obtain the two-grid error propagator

$$\tilde{E}_{2g}^{(\nu)} = \underbrace{(I - MQ)^\nu}_{\text{requires new smoother}} \underbrace{(I - PQ_c^{-1}P^\dagger Q)}_{=I-PD_c^{-1}P^\dagger D}, \quad \underbrace{Q_c := P^\dagger Q P}_{\text{new coarse operator}}$$

- P is valid for D and Q
- P preserves $Q_c^\dagger = Q_c \to$ recursive application possible
- SAP smoother does not work anymore

Choice: Replace M by GMRES

γ_5-Preconditioning

- AMG for Q in practice a factor of 2 slower than DD-αAMG
  $\rightarrow$ can be remedied by "γ_5-preconditioning":

$$\begin{aligned} & (Q - \sigma I)\varphi &= \psi \\ \Leftrightarrow \quad & \gamma_5 (Q - \sigma I)\varphi &= \gamma_5 \psi \\ \Leftrightarrow \quad & (D - \sigma\gamma_5)\varphi &= \gamma_5 \psi \end{aligned}$$

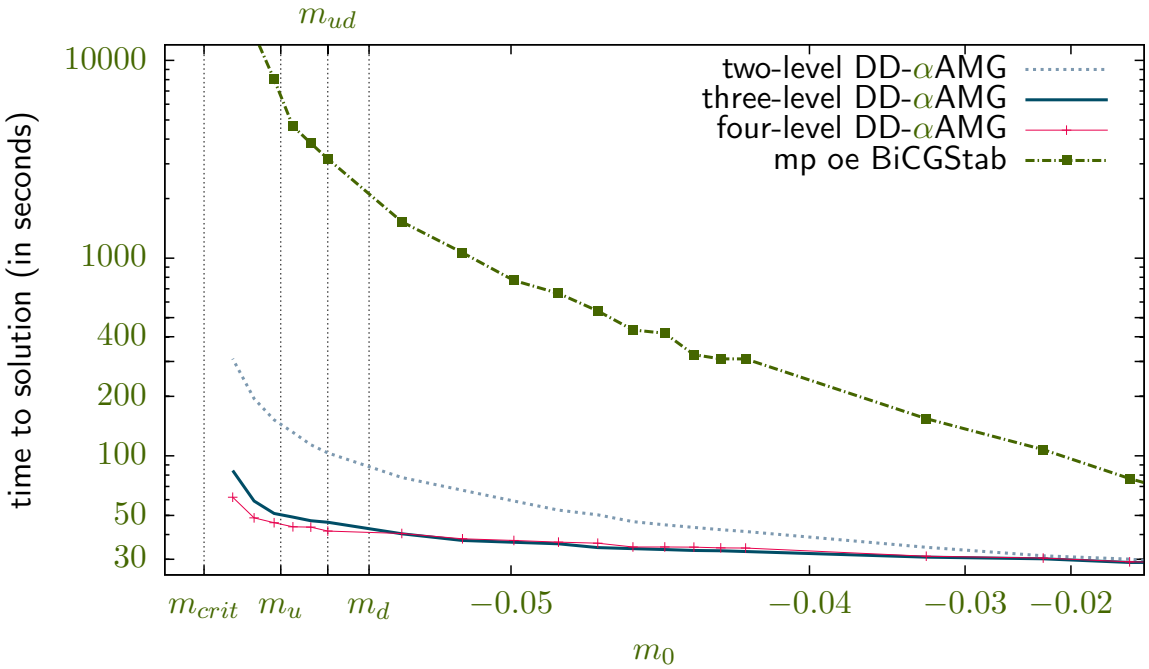
- error propagator of smoother now reads:

$$\tilde{E}_S^{(\nu)} = (I - M(D - \sigma\gamma_5))^\nu$$

- enables to compute $(Q - \sigma)^{-1}$ with AMG efficiently and thus to compute small eigenvectors of Q

AMG for D in Practice: Scaling with the Bare Quark Mass

Configuration: 64×64^3, 128 cores



- lighter quark mass $m_0 \rightarrow$ more ill-conditioned system
- AMG less sensitive to condition number than Krylov subspace methods

An Introduction to Eigensolver Methods

- Applying a matrix A to an arbitrary vector v multiple times converges to *largest* eigenvector
  $\rightarrow$ Power method
- If λ is an eigenvalue of A, then $(\lambda - \sigma)^{-1}$ is an eigenvalue of $(A - \sigma I)^{-1}$
  $\rightarrow$ Inverse iteration
  $\rightarrow$ Now we can compute any eigenpair we want
- Given a normalized vector v the *Rayleigh Quotient* is defined as

$$r(x) = v^{\dagger} A v$$

  $\rightarrow$ $r(x)$ is the best-approximation to an eigenvalue of v
  $\rightarrow$ Rayleigh quotient iteration (cubic convergence for hermitian matrices)

An Introduction to the Jacobi-Davidson Method

The Jacobi-Davidson method combines two ideas

- *Davidson*: Impose Galerkin Condition $AVs - \theta s \perp V$ on eigenvalue problem to some subspace $V = \{v_1, v_2, \ldots, v_n\}$, which leads to

  $$V^\dagger AVs - \theta s = 0$$

  If (θ_i, s_i) is a solution of this equation then $(\theta_i, u_i = Vs_i)$ is called a *Ritz pair* w.r.t. the subspace V
- (θ_i, u_i) approximates eigenpairs $\rightarrow$ How to construct V?
- *Jacobi*: Given an eigenvector approximation u_i, choose an orthogonal correction $t \perp u_i$ such that

  $$A(u_i + t) = \lambda(u_i + t)$$

  We find that t fulfills the equation

  $$(I - u_i u_i^\dagger)(A - \lambda I)(I - u_i u_i^\dagger)t = -(A - \theta_i I)u_i := -r$$

  λ unknown $\rightarrow$ replace by θ_i

An Introduction to the Jacobi-Davidson Method

The Jacobi-Davidson method combines two ideas

- *Davidson*: Impose Galerkin Condition $AVs - \theta s \perp V$ on eigenvalue problem to some subspace $V = \{v_1, v_2, \ldots, v_n\}$, which leads to

$$V^\dagger AVs - \theta s = 0$$

  If (θ_i, s_i) is a solution of this equation then $(\theta_i, u_i = Vs_i)$ is called a *Ritz pair* w.r.t. the subspace V
- (θ_i, u_i) approximates eigenpairs $\rightarrow$ How to construct V?
- *Jacobi*: Given an eigenvector approximation u_i, choose an orthogonal correction $t \perp u_i$ such that

$$A(u_i + t) = \lambda(u_i + t)$$

  We find that t fulfills the equation

$$(I - u_i u_i^\dagger)(A - \theta_i I)(I - u_i u_i^\dagger)t = -(A - \theta_i I)u_i =: -r$$

  Solve inexactly for t and add result to V

The Jacobi-Davidson Method

Algorithm 1: Jacobi-Davidson (basic)

input: initial guess t, desired accuracy ε
output: eigenpair (λ, x)

```
for all m = 1, 2, ... do
    t = t − V V^† t,  v_m = t / ||t||_2
    M = V^† A V
    compute smallest eigenpair (θ, s) of M
    u = V s
    r = A u − θ u
    if ||r||_2 ≤ ε then
        λ = θ,  x = u
    solve (approximately) for t ⊥ u in
        (I − u u^†)(A − θ I)(I − u u^†) t = −r
```

The Jacobi-Davidson Method with AMG

Algorithm 2: Jacobi-Davidson + AMG

input: no. of eigenvalues n, subspace size ℓ, initial guess $[v_0, v_1, \ldots, v_k, t]$, desired accuracy ε

output: set of n eigenpairs (Λ, X)

```
for all m = k+1, k+2, ... do
    t = t - VV^†t, v_m = t/||t||_2
    compute all (θ_i, s_i) with (AV)^†(AV)s_i = θ_i(AV)^†Vs_i
    find smallest θ_i ∉ Λ
    u = Vs_i, r = Au - θ_i u
    if ||r||_2 ≤ ε then
        Λ = [Λ, θ_i], X = [X, u]
        if m ≥ ℓ + n_conv then
            jd_restart
        find smallest θ_i ∉ Λ
        rebuild interpolation
    solve (approximately) (A - θ_i I)t = r
```

Restarting

Goal: Limit the amount of work (and memory) per iteration
$\rightarrow$ control dimension of search space V

Algorithm 3: jd_restart

input: look ahead m, number of converged eigenpairs n_{conv}, search space V
output: new search space V and Ritz pairs (θ_i, s_i)

1 $[\sim, \pi] = \texttt{sort}(|\Theta|, \text{ascending})$
2 **for all** $i = 1, \ldots m + n_{conv}$ **do**
3 $\quad V_i = V s_{\pi(i)}$
4 $V = V(1 : m + n_{conv})$
5 orthogonalize V
6 compute all (θ_i, s_i) with $(AV)^\dagger (AV) s_i = (AV)^\dagger V s_i$

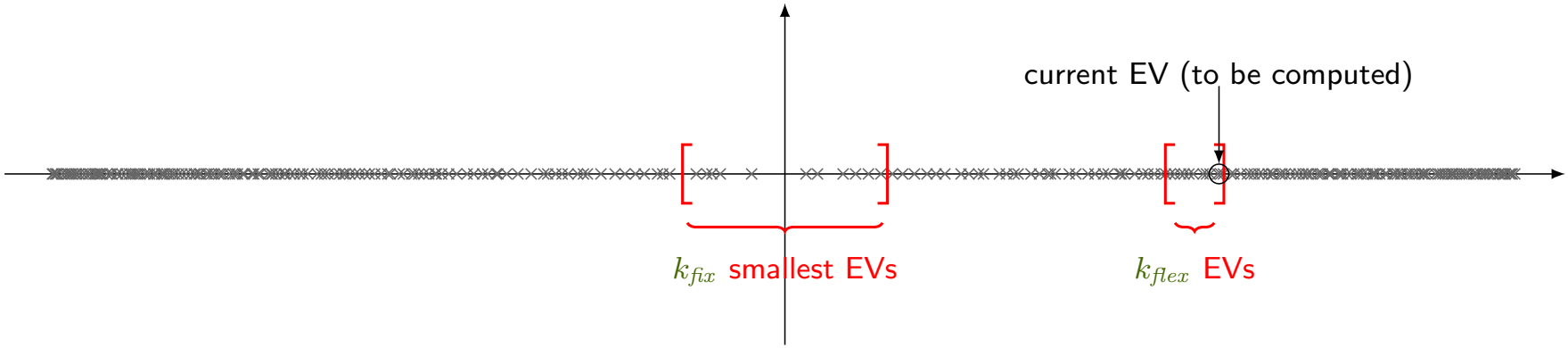
- look ahead m preserves search space information in spirit of thick restarting
- subspace size is bound by $n + \ell$

Interaction between Jacobi-Davidson and AMG

Rebuilding the interpolation

- JD computed all eigenvectors up to some point σ
- AMG needs small eigenvectors of $(A - \sigma I)$ for interpolation P

Idea: Use converged vectors to improve P dynamically:



Build interpolation P with:

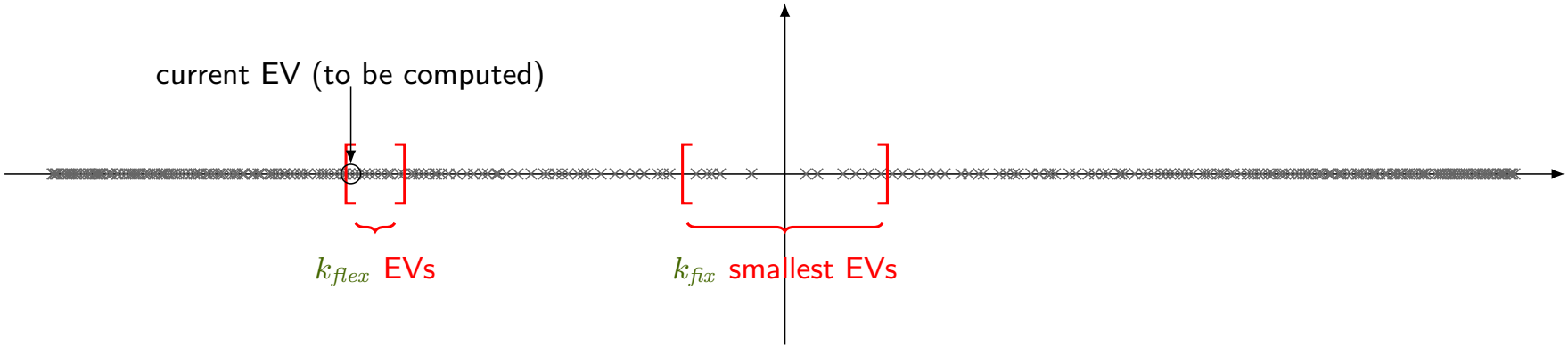
- k_{fix} smallest eigenvectors
- current eigenvector approximation u
- k_{flex} converged eigenvectors closest to u

Interaction between Jacobi-Davidson and AMG

Rebuilding the interpolation

- JD computed all eigenvectors up to some point σ
- AMG needs small eigenvectors of $(A - \sigma I)$ for interpolation P

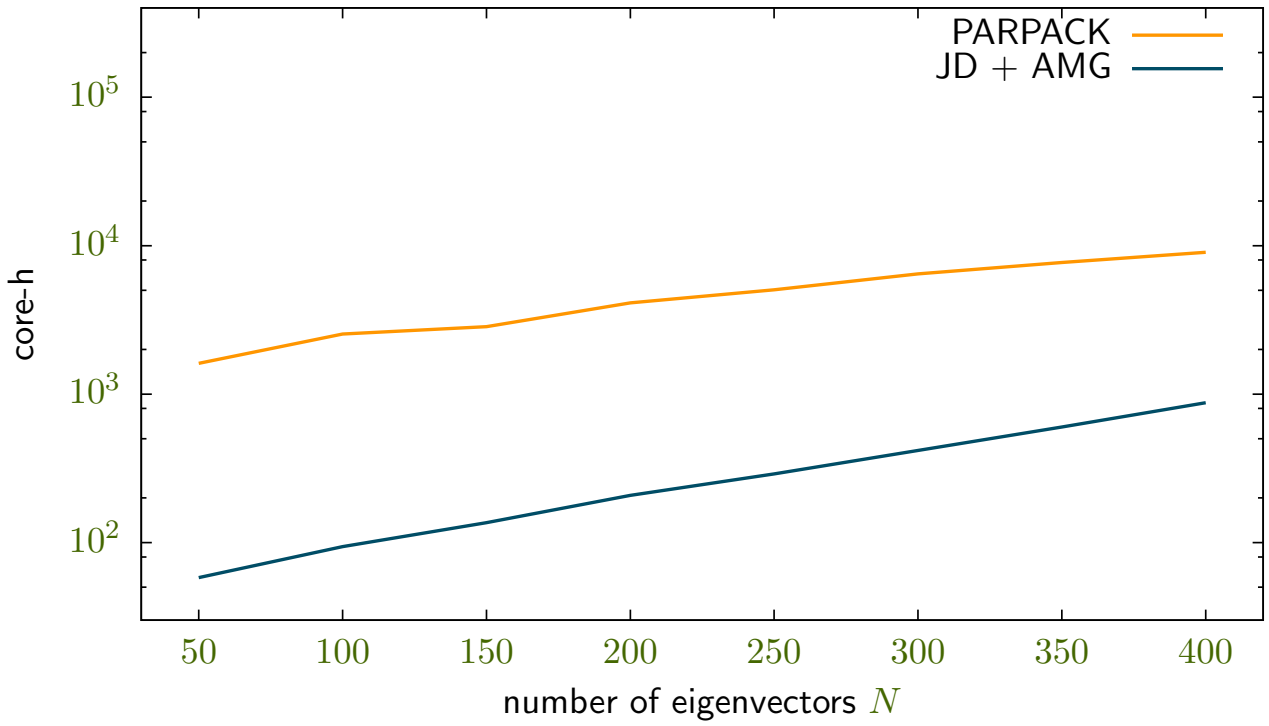
Idea: Use converged vectors to improve P dynamically:



Build interpolation P with:

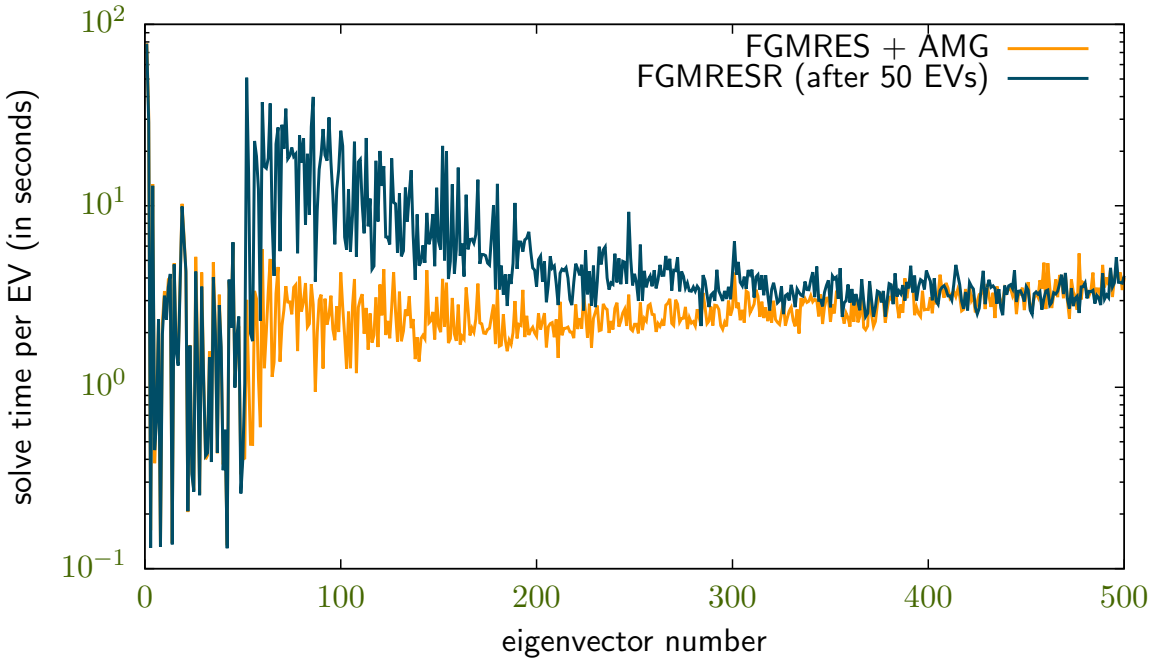
- k_{fix} smallest eigenvectors
- current eigenvector approximation u
- k_{flex} converged eigenvectors closest to u

Comparison with PARPACK: Scaling with the Number of Eigenvectors N



- 48×24^3 lattice ($\approx 8 \times 10^6$ unknowns)
- 648 cores on JURECA
- PARPACK: Krylov subspace dimension $4N$
- JD: $k_{fix} = 8$, $k_{flex} = 16$, search space dimension 150 orthogonalization cost $n_{conv} + 150$

Effectiveness of AMG Solves



- coarse grid correction switched off after 50 converged eigenvectors
- clear loss in solve time for first 400 eigenvectors
- afterwards coarse grid effectiveness deteriorates
  - better interpolation strategy?
  - hybrid approach?
  - different preconditioner at some point?

Summary & Outlook

Summary

- Jacobi-Davidson type algorithm with multigrid support
- good scaling with number of eigenvalues
- scalability mostly depends on solver performance
- speed up can be orders of magnitudes

Improve Eigensolver

- explore possible benefits of AMG with more levels
- study orthogonal projections for solves
- find optimal restarting strategy
- improve interaction between JD and interpolation
- explore scaling behavior with lattice size
  (former approach scaled better than PARPACK)
- compare with other eigensolver approaches (e.g. FEAST)

Acknowledgments

All results computed on JUROPA at
Jülich Supercomputing Centre (JSC)

Funded by Deutsche Forschungsgemeinschaft (DFG),
Transregional Collaborative Research Centre 55
(SFB TR 55)

All configurations provided by BMW-c, QCDSF & CLS