Improving coarsest level solves in multigrid for lattice QCD

Andreas Frommer

DFG Deutsche Forschungsgemeinschaft German Research Foundation
FOR 5269
STIMULATE European Joint Doctorates
BERGISCHE UNIVERSITÄT WUPPERTAL

Joint work with

- *Gustavo Ramirez Hidalgo*,
  Postdoc in DFG Research Unit FOR 5269, formerly PhD student in EU-MCS "Stimulate"
- *Henning Leemhuis*,
  PhD student, University of Wuppertal

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 765048

FOR 5269

Contents

Problem statement

The problem

Given: square matrix D,

- D Wilson, clover improved Wilson or twisted mass discretization of Dirac operator
- periodic, anti-periodic or open boundary conditions
- D represents nearest neighbor coupling on $4d$-lattice ($n_t \times n_x \times n_y \times n_z$ lattice sites)
- $D \in \mathbb{C}^{n \times n}$, with $n = 12 n_t n_x n_y n_z$

Wanted: solution ψ of $D\psi = \varphi$.

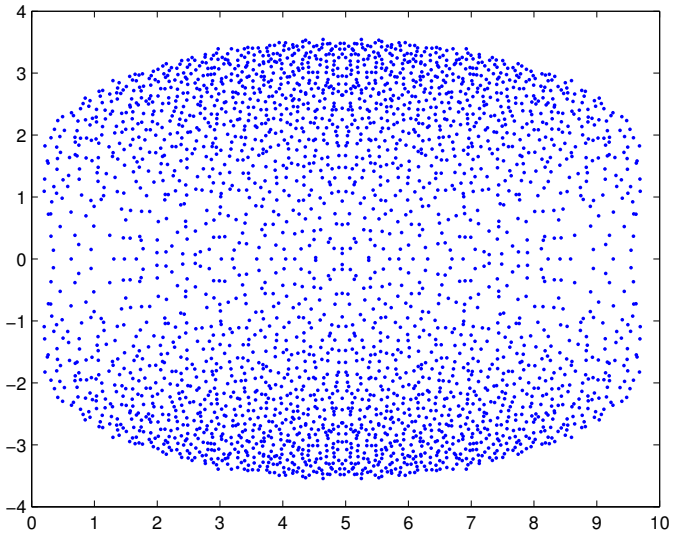
Important: D is ill-conditioned:

cond$(D) = \|D\| \cdot \|D^{-1}\|$ *increases as* n *increases*

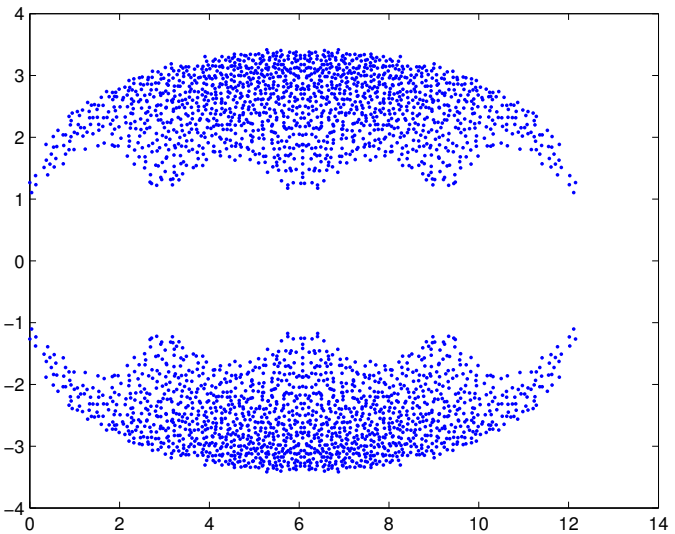
Example: The Wilson-Dirac operator

$$(D_W\psi)(x) = \frac{m_0+4}{a}\psi(x) - \frac{1}{2a}\sum_{\mu=0}^{3}\left((I_4-\gamma_\mu)\otimes U_\mu(x)\right)\psi(x+\hat{\mu}) - \frac{1}{2a}\sum_{\mu=0}^{3}\left((I_4+\gamma_\mu)\otimes U_\mu^H(x-\hat{\mu})\right)\psi(x-\hat{\mu})$$

$\beta = 5.0$



$\beta = 6.0$



Standard iterations for $Dx = b$

Krylov subspace

The k-th Krylov subspace for D and $r \in \mathbb{C}^n$ is

$$\mathcal{K}_k(D, r) = \text{span}\{b, Dr, \ldots, D^{k-1}r\}$$

An iterative Krylov subspace method takes

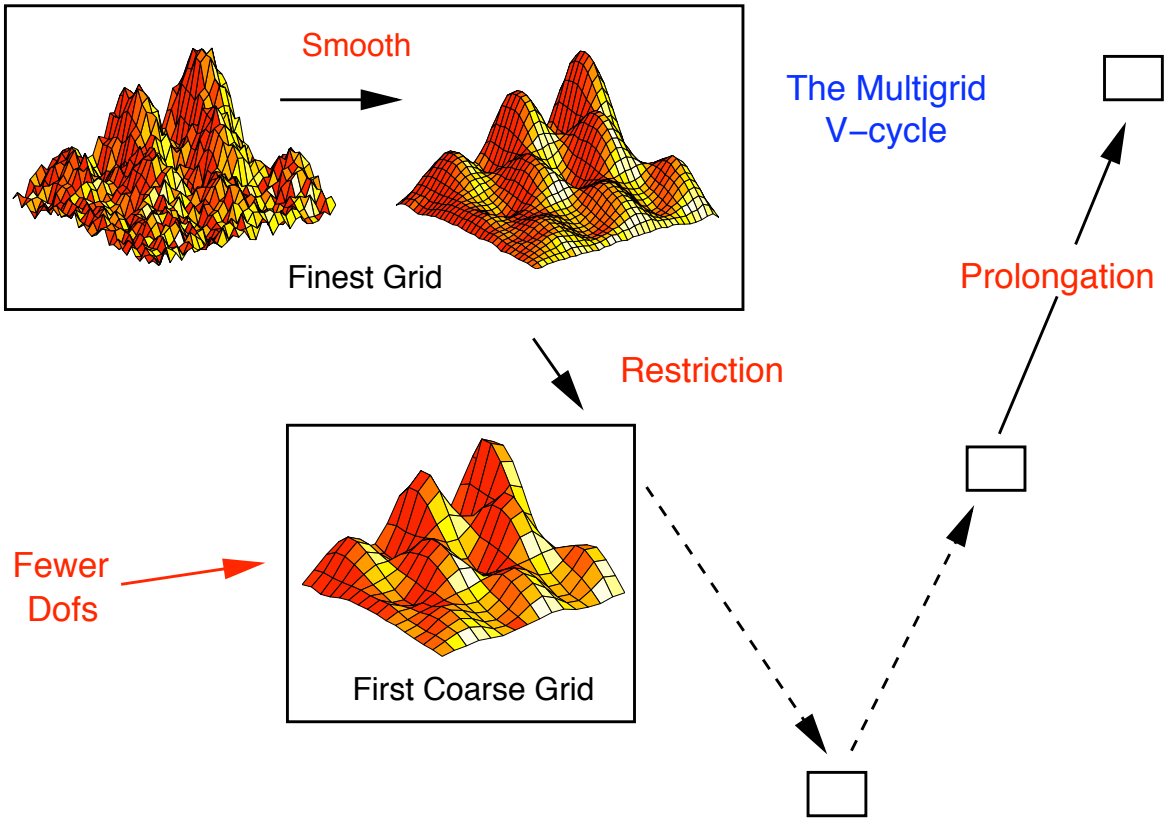
$$x^k \in x^0 + \mathcal{K}_k(D, r^0), \quad r^0 = b - Dx^0 \text{ initial residual}$$

Examples: CG, GMRES

- variational characterization of iterates
- (pseudo-) spectrum matters, convergence speed $\propto$ cond(D)
- lots of theory
- Faber-Manteuffel theorem

The multigrid idea

- hierarchy of operators
- smoothing
- coarse grid correction
- . . . recursively



Multigrid in lattice QCD

Smoother:

- GMRES
- SAP

Coarse grid operator:

- aggregation

Multigrid in lattice QCD

Smoother:

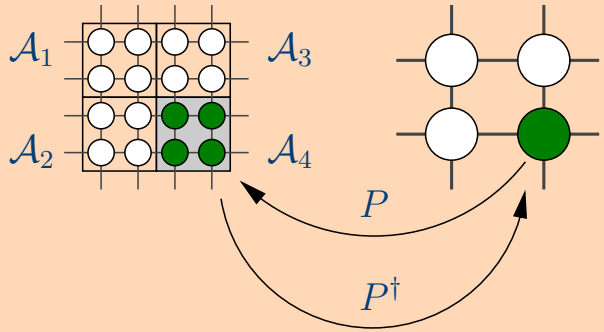
- GMRES
- SAP

Coarse grid operator:

- aggregation

Aggregation:

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



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

$$(v^{(1)}, \ldots, v^{(k)}) = \quad = \begin{matrix}\mathcal{A}_1\\ \mathcal{A}_2\\ \vdots\\ \mathcal{A}_s\end{matrix} \rightarrow P = \begin{pmatrix}\mathcal{A}_1 & & & \\ & \mathcal{A}_2 & & \\ & & \ddots & \\ & & & \mathcal{A}_s\end{pmatrix}$$

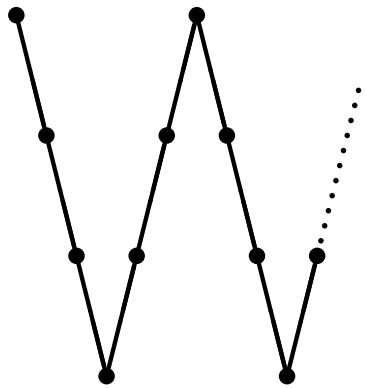
V-cycle multigrid

Algorithm $x \leftarrow$ **v_cycle**(level ℓ, x, r)

1: $x, r \leftarrow$ smoother(D_ℓ, x, r) {pre-smoothing}
2: $r_c \leftarrow R_\ell r$ {restriction}
3: **if** on coarsest level L **then**
4: solve $D_L e_c = r_c$ using favorite solver {low accuracy is ok}
5: **else**
6: $e_c \leftarrow$ **v_cycle**(level $\ell + 1, 0, r_c$)
7: **end if**
8: $x \leftarrow x + P_\ell e_c$ {prolongation + coarse grid correction}
9: $x, r \leftarrow$ smoother(D_ℓ, x, r) {post-smoothing}

Algorithm $x \leftarrow$ **v_cycle_mg**(x, r)

1: **repeat**
2: $x \leftarrow x +$ **v_cycle**(level $1, 0, r$), $r = b - Dx$
3: **until** residual r is small enough



K-cycles

Algorithm $x \leftarrow$ **k_cycle**(level ℓ, x, r)

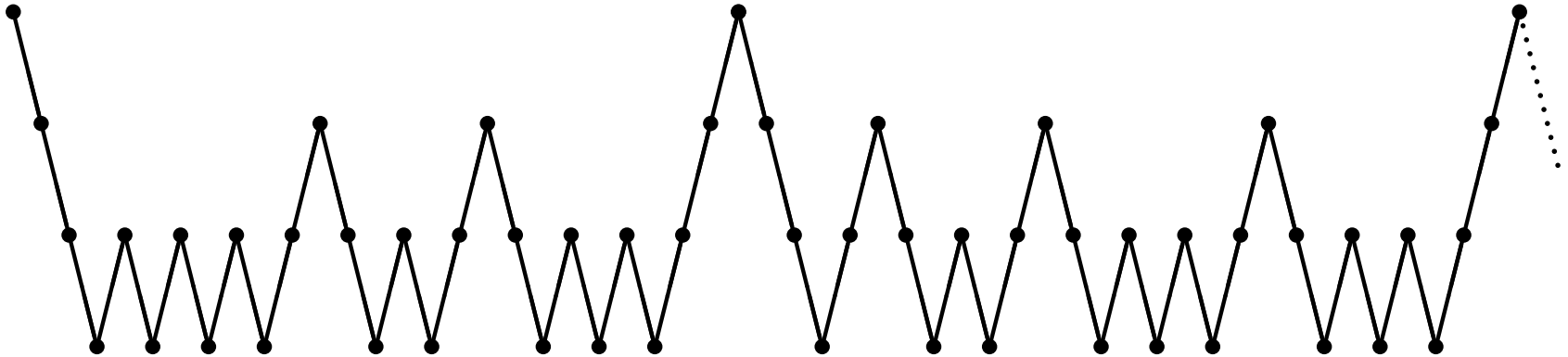
1: $x, r \leftarrow$ smoother(D_ℓ, x, r) {pre-smoothing}
2: $r_c \leftarrow R_\ell r$ {restriction}
3: **if** on coarsest level L **then**
4: solve $D_L e_c = r_c$ using favorite solver {low accuracy is ok}
5: **else**
6: solve $D_{\ell+1} e_c = r_c$ with GMRES prec'd with **k_cycle**(level $\ell + 1, \cdot, \cdot$)
7: $e_c \leftarrow$ **k_cycle**(level $\ell + 1, 0, r_c$) {low accuracy}
8: **end if**
9: $x \leftarrow x + P_\ell e_c$ {prolongation + coarse grid correction}
10: $x, r \leftarrow$ smoother(D_ℓ, x, r) {post-smoothing}

Algorithm $x \leftarrow$ **k_cycle_prec_gmres**(x, r)

1: **repeat**
2: do next step of GMRES prec'd with **k_cycle**(level $1, \cdot, \cdot$)
3: **until** residual r is small enough

K-cycles

- are non-stationary
- visit the coarsest level often



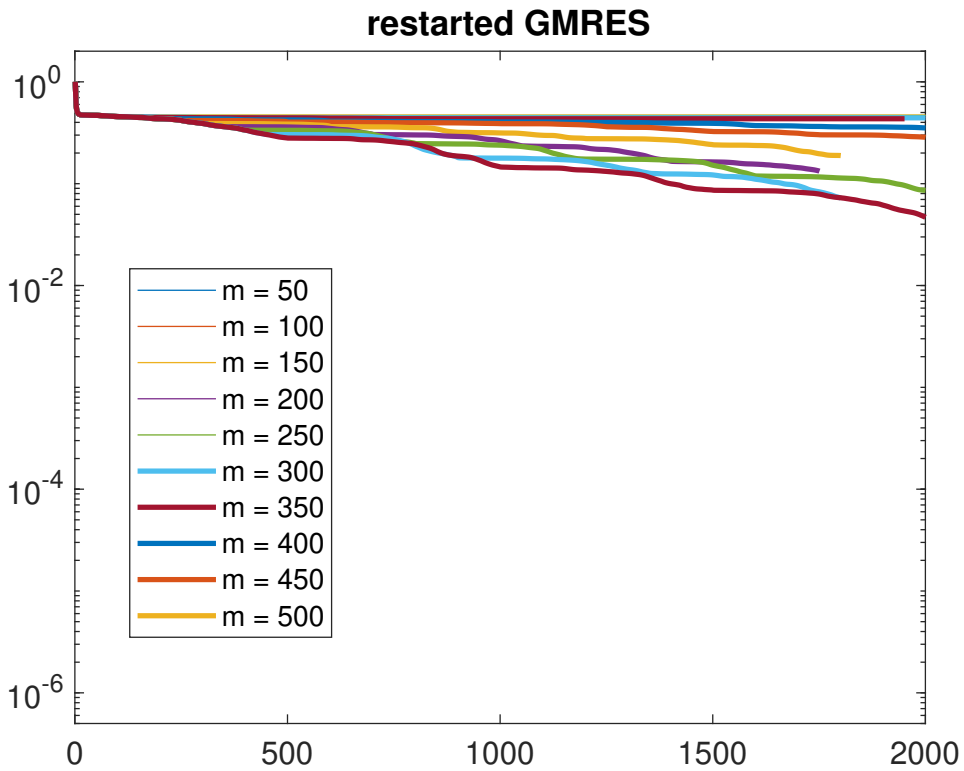
The coarsest system

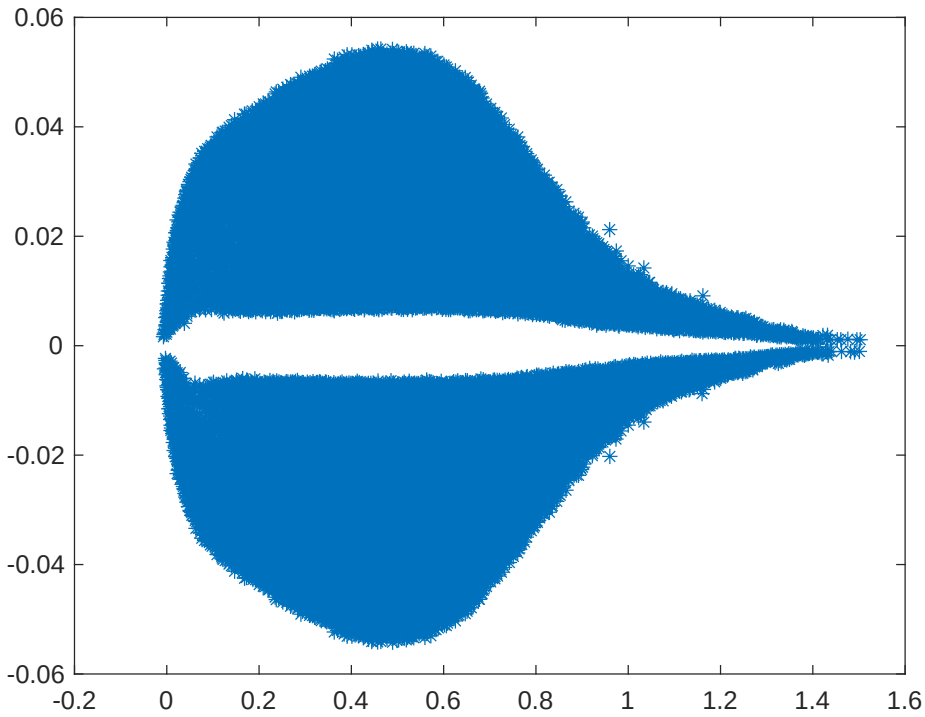
- is the smallest
- is as ill-conditioned as the finest
- is typically visited more often the more levels we have

Twisted mass example

level	lattice dimensions	dofs	n_{tv}	n
1	96 x 48 x 48 x 48	12		128M
/	3 3 3 3		24	
2	32 x 16 x 16 x 16	48		6.5M
/	2 2 2 2		28	
3	16 x 8 x 8 x 8	56		460k
/	2 2 2 2		32	
4	8 x 4 x 4 x 4	64		32k

GMRES on coarsest level





Polynomial preconditioning I

$$Dx = b \quad \rightarrow \quad D\underbrace{q(D)}_{=x} y = b, \quad \text{cond}(Dq(D) \ll \text{cond}(D)$$

GMRES with $Dq(D)$ instead of D

- ◦ $\deg(q) + 1$ mvms with D per iteration
- ◦ Krylov subspace is $\mathcal{K}_k(Dq(D), b) \subseteq \mathcal{K}_{k(\deg(q)+1)}(D, b)$
- − worse iterate for the same effort
- \+ might avoid restarts
- \+ q obtainable via "bootstrap" approach
- \+ no inner products

GMRES refresher

- build ONB $v_1, \ldots, v_k$ for $\mathcal{K}_k(D, r^0)$ via the Arnoldi process (orthogonalize Dv_j against $v_1, \ldots, v_j$)
- Arnoldi relation

$$DV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T, \quad V_k = [v_1 | \cdots | v_k]$$

- $H_k = V_k^* D V_k$ is upper Hessenberg
- $x = x^0 + V_k \zeta_k$ minimizes $\|b - Dx^k\|$ iff ζ_k minimizes $\|\|r^0\| e_1 - \overline{H}_k \zeta\|$, $\overline{H}_k = \begin{bmatrix} H_k \\ h_{k+1,k} e_k^T \end{bmatrix}$

Polynomial preconditioning II

How to get q:

- GMRES (implicitly) builds polynomial q_{k-1} with for which $\|b - D(x^0 + q_{k-1}(D)r^0)\|_2 = \|(I - Dq_{k-1}(D))r^0\|$ is minimal $\rightarrow Dq_{k-1}(D) \approx I$
- q_{k-1} can be retrieved from the harmonic Ritz values θ_i of the Hessenberg matrix of the Arnoldi process

$$q_{k-1}(D) = \sum_{i=1}^{k} \frac{1}{\theta_i} \prod_{j=1}^{i-1} \left(I - \frac{1}{\theta_j} D \right)$$

- Caveat: Use Lejà ordering for the θ_i:

$$\prod_{j=1}^{i-1} |\theta_i - \theta_j| = \max_{\ell=i}^{k} \prod_{j=1}^{i-1} |\theta_\ell - \theta_j|$$

Deflation

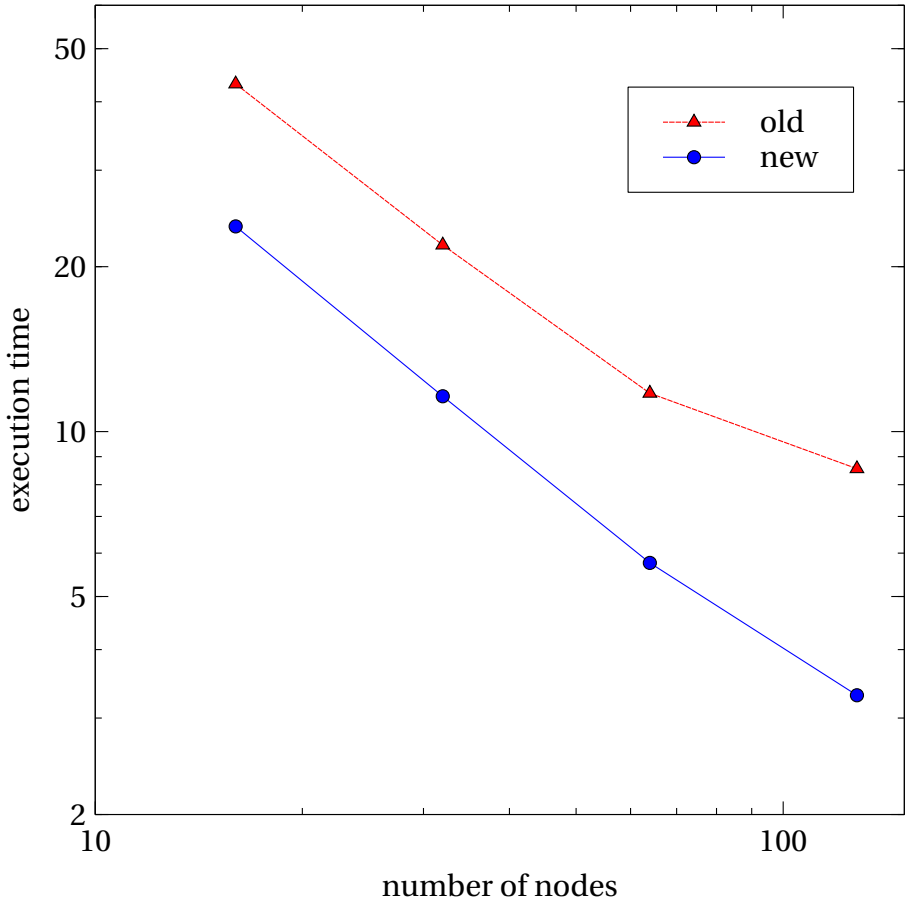
- undeflated GMRES: $x^k \in x^0 + \mathcal{K}_k(D, r^0)$
- deflated GMRES: $x^k \in x^0 + \mathcal{K}_k(D, r^0) + \mathcal{U}$, $\mathcal{U}$ deflation subspace

such that $\|b - Dx^k\|$ is minimal

Deflation à la GCRO-DR

- do a first GMRES cycle, then extract basis U for the m smallest harmonic Ritz vectors
- in all subsequent cycles:
  - extract minimal residual iterate from $\mathcal{K}_k(D, r) + \mathcal{U}$ ("extended" Arnoldi with orth. projection on $D\mathcal{U}$)
  - update $\mathcal{U}$ using harmonic Ritz vectors from current cycle
- stop updating when no further gain
- works for sequences of systems

Strong scaling study



96×48^3 lattice, $\mu = 7.2 \cdot 10^{-4}$, JUWELS, $\deg(q) = 20$, $\dim(\mathcal{U}) = 400$,
32 MPI procs/node, 1 thread/process.

Multigrid and K-cycles

Polynomial preconditioning and GCRO-DR

Direct factorization

Preliminary results!

Why not solve the coarsest system using **Gaussian elimination**?

$$D = PLU, \;\; Dx = b \Leftrightarrow Ly = P^T b, Ux = y$$

- factorization should be cheap $\rightarrow$ exploit sparsity
- factorization should be fast
  - group into matrix-matrix operations ("blocking")
  - make blocks data-sparse
- forward and backward substitution should be fast
  - make blocks data-sparse

BLR-factorization ("block low rank")

- $P_r D P_c \approx LU$
- non-zero block $B \in \mathbb{C}^{m \times \ell}$ in L, U is represented as $B = B_1 B_2, \;\; B_1 \in \mathbb{C}^{m \times r}, B_2 \in \mathbb{C}^{r \times \ell}, \;\; r \ll m, \ell$

MUMPS

MUMPS: multifrontal massively parallel sparse direct solver
https://mumps-solver.org/

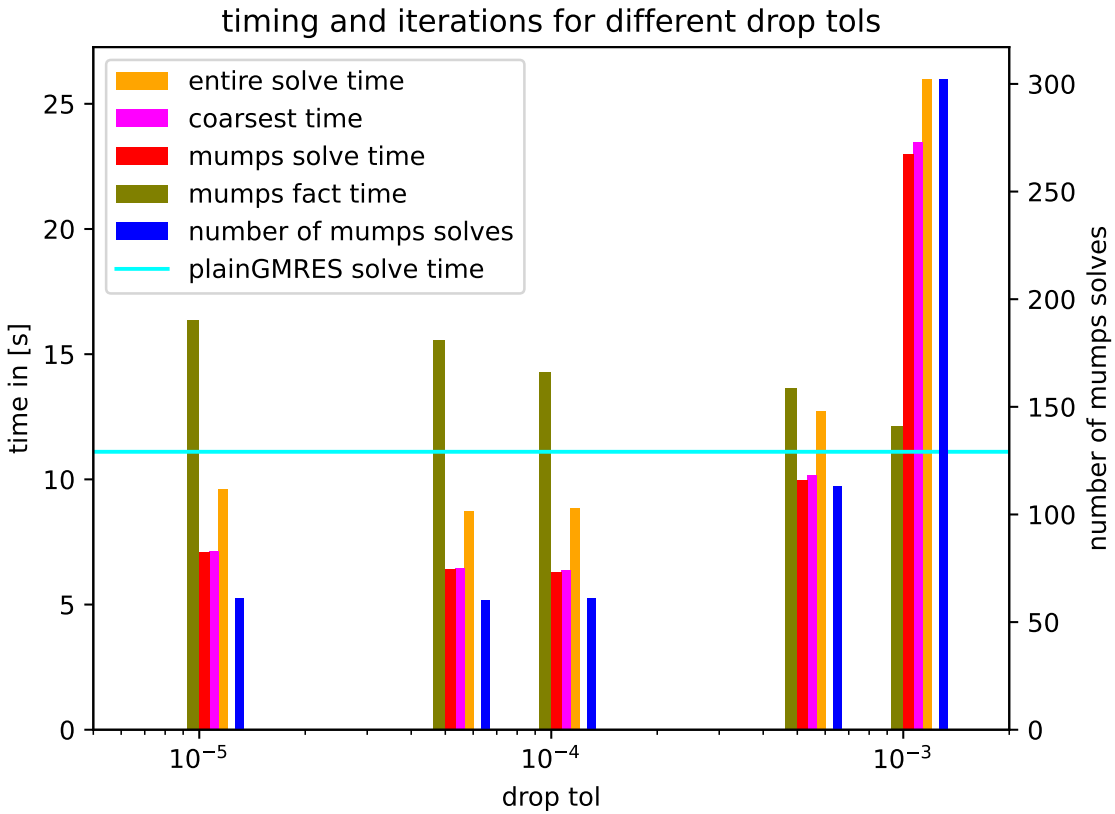
- state-of-the-art
- parallel
- developed since 1991
- scottish and french
- since 2022: BLR – user can specify approximation accuracy

For coarsest grid operator:

+ blocks come naturally
− preserving sparsity is hard due to 4d coupling

We **interface** MUMPS with DDαAMG and use it as a **black box**.

BLR accuracy



usual 96×48^3 configuration, no SSE, no multithreading for MUMPS solve

Agglomeration

Faster on coarser levels using fewer nodes? $\rightarrow$ **agglomeration**

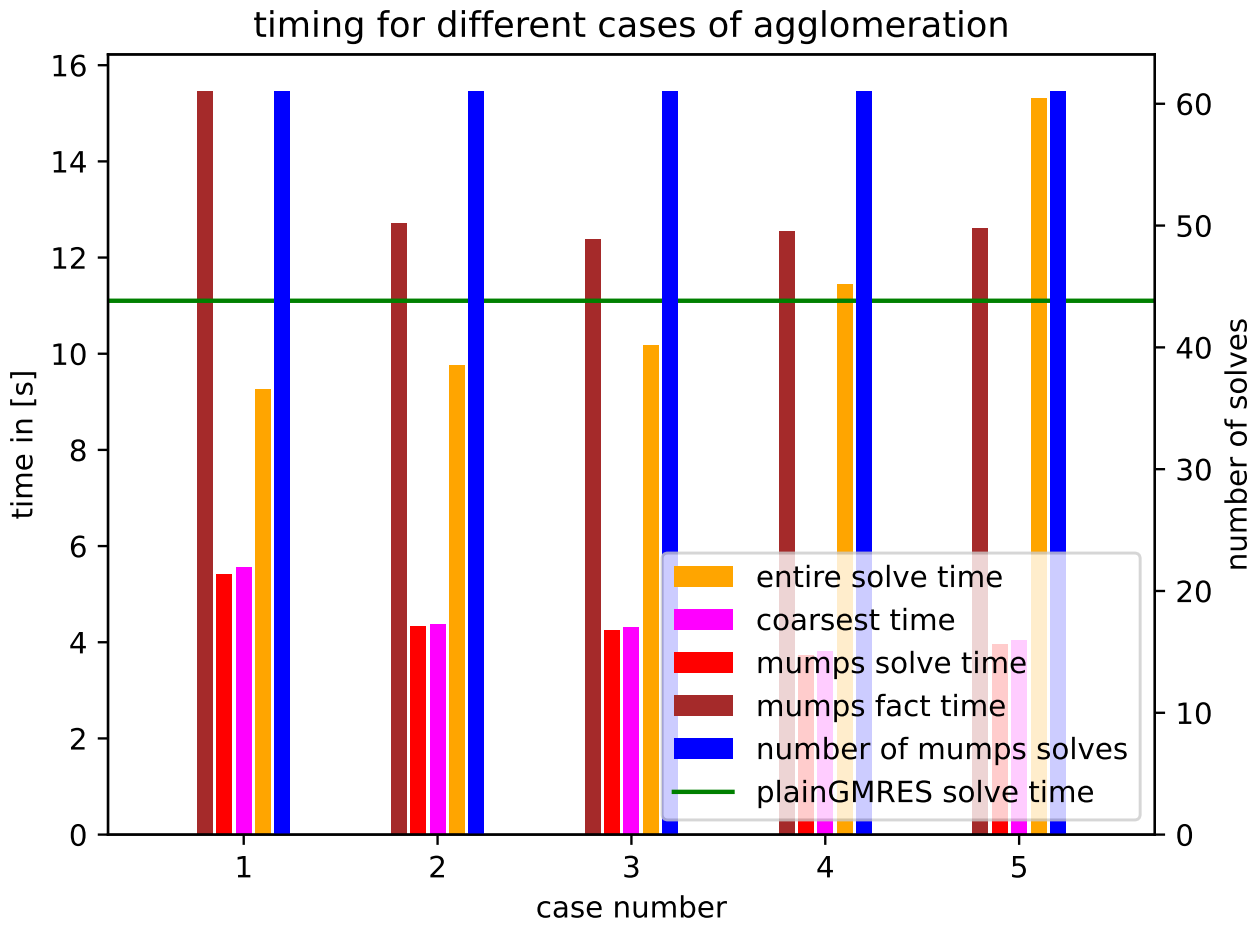
finest level: 96 48 48 48
- local lattice:
  - case1: 16 16 16 16
  - case2,3: 16 16 16 24
  - case4,5: 16 16 24 24

depth 1: 24 12 12 12
- local lattice:
  - case1: 4 4 4 4
  - case2: 4 4 4 6
  - case3: 4 4 4 12
  - case4: 4 4 6 6
  - case5: 4 4 6 12

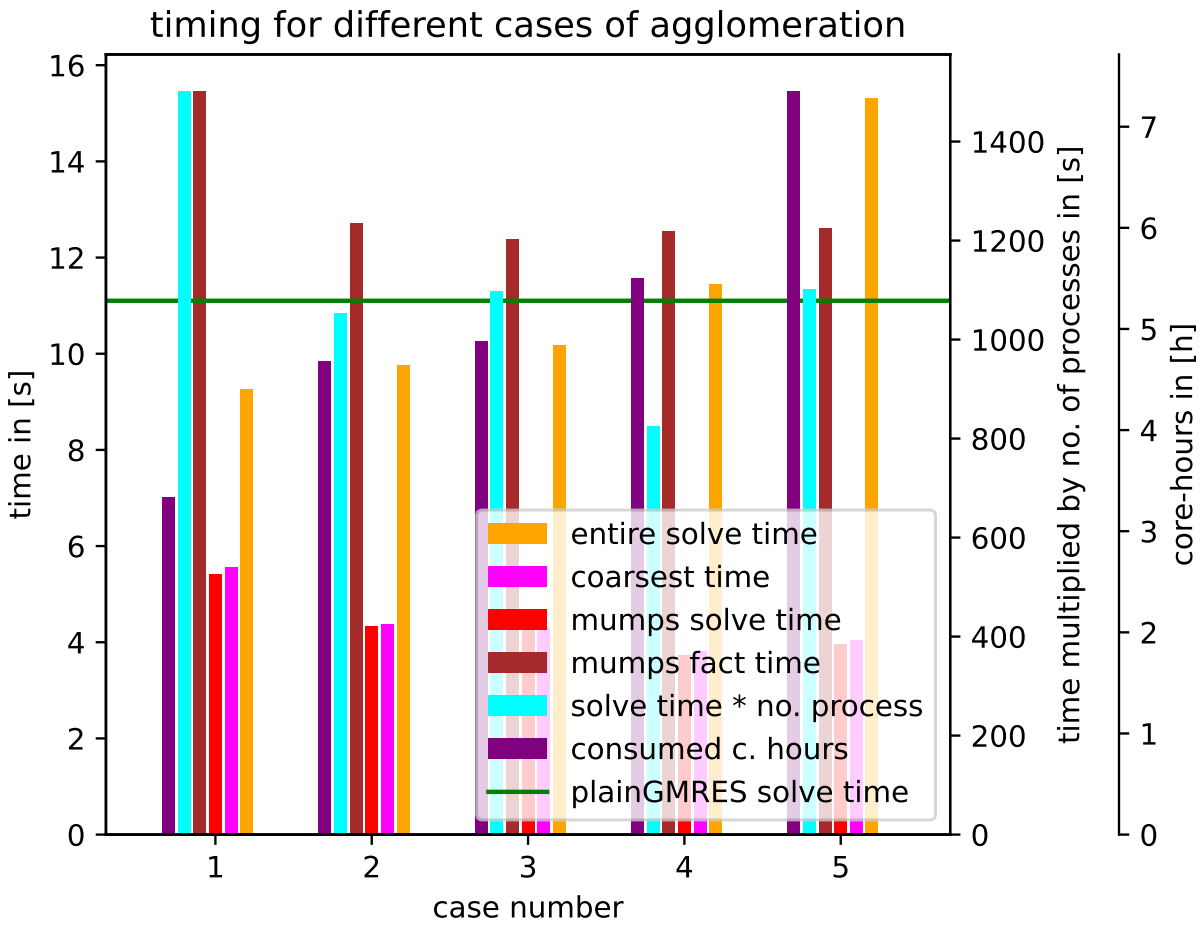
depth 2: 12 6 6 6
- local lattice:
  - case1: 2 2 2 2
  - case2,3: 2 2 2 6
  - case4,5: 2 2 6 6

depth 3: 6 3 3 3
- local lattice:
  - case1: 1 1 1 1
  - case2,3: 1 1 1 3
  - case4,5: 1 1 3 3

Results with agglomeration



Results with agglomeration



Conclusions

- Coarse grid solves tend to become the bottleneck
- polynomial preconditioning + deflation help
- both can be done adaptively
- direct approximate factorizations become increasingly interesting