Multiple Right-hand-side Implementation for DDαAMG

Shuhei Yamamoto
s.yamamoto@cyi.ac.cy

Simone Bacchio, Jacob Finkenrath

The Cyprus Institute

September 22, 2020

Outline

QFT

In path-integral formulation, all physical predictions in QFT is the generating functional:

$$Z = \int \mathcal{D}\phi e^{iS}.$$

In particular, for QCD,

$$S_{\mathrm{QCD}} = \int d^4x \mathcal{L}_{\mathrm{QCD}} = S_\psi + S_G$$

where

$$S_\psi = \int d^4x \sum_f \bar{\psi}_f \left(i\gamma_\mu D_\mu - m_f\right)\psi_f \text{ and } S_G = S_G(A^a_\mu).$$

The expectation value of an observable can be computed using

$$\langle O \rangle = \frac{1}{Z}\int \mathcal{D}\phi O(\phi) e^{iS}.$$

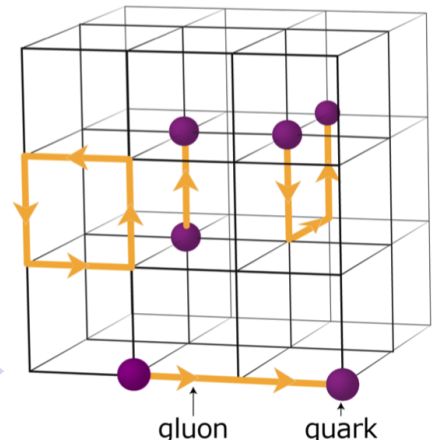
Lattice QCD

Now the task is to evaluate $\langle O \rangle$. We do this non-perturbatively. To be more specific,

- We perform Wick rotation from Minkovski to Eulcidean space-time to obtain
$$\langle O \rangle = \frac{1}{Z} \int \mathcal{D}\phi O(\phi) e^{-S_{\mathrm{E}}}$$
where S_{E} is a real-valued Euclidean action.
- Then, we use $e^{-S_{\mathrm{E}}}$ as a Monte-Carlo weight and have
$$\langle O \rangle = \lim_{N \to \infty} \frac{1}{N} \sum_{i}^{N} O_i \approx \frac{1}{N} \sum_{i}^{N} O_i.$$

To make the computation doable on a computer,

- We discretize the space-time into the lattice with its spacing a and dimension $L^3 \times T$.
- Fermion fields, ψ, live on a lattice site, and gauge fields are replaced by a link, U_μ, connecting the adjacent points.



The Problem

As the fermion fields in the action are Grassmann numbers, using its properties, we can take integration over fermion fields explicitly to obtain

$$\langle O \rangle = \frac{1}{Z} \int \mathcal{D}U O(D^{-1}, U) \mathrm{Det} D(U) e^{-S_G}$$

where D is a Dirac matrix in $S_\psi = \bar{\psi} D \psi = \sum_f \bar{\psi}_f \left(\not{D}_E + m_f \right) \psi_f$.

Evaluation of the expectation value requires

- computation of the inverse of the Dirac matrix repeatedly many times
- generation of gauge configurations according to the weight $e^{-S_G} \mathrm{Det} D(U)$ via importance sampling, e.g., HMC methods

In lattice QCD calculations, the main computational task is to solve the system

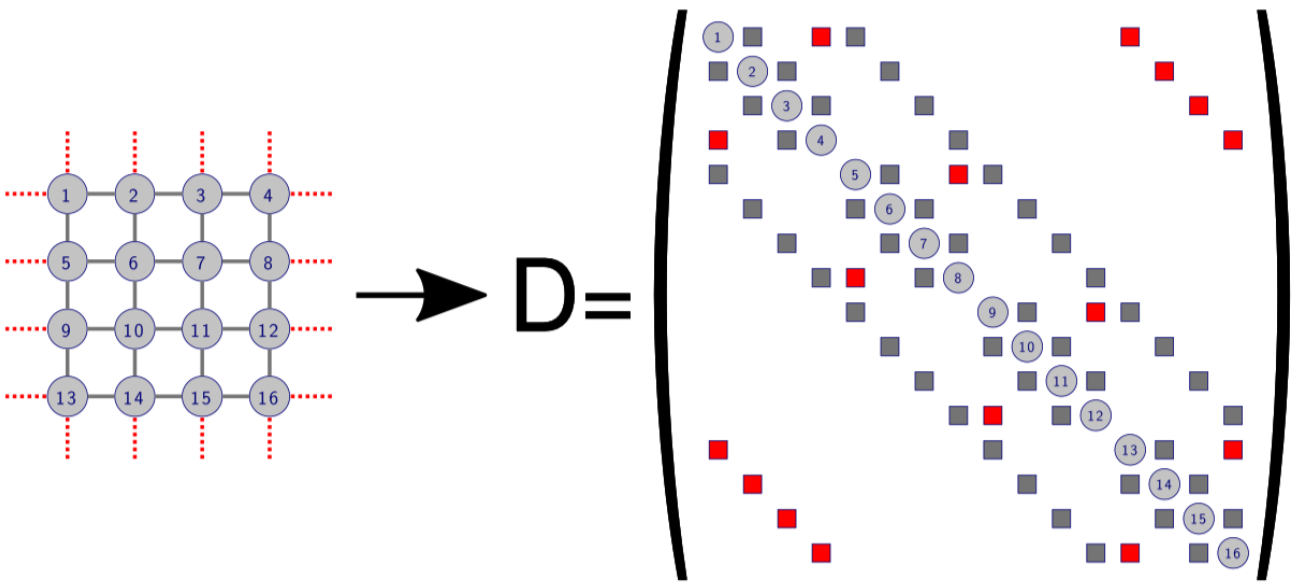
$$Dx = b.$$

The Matrix

- Dirac matrix: $D = \not{\partial}_E + m$
- Wilson Dirac operator:
  $D_{\mathrm{W}} = \frac{1}{2}\sum_{\mu=0}^{3}\left(\gamma_\mu(\Delta_\mu + \Delta^\mu) - a\Delta_\mu\Delta^\mu\right) + \frac{m}{a}$ where

$$\Delta^\mu\psi = \frac{1}{a}\left(U_\mu(x)\psi(x+\hat{\mu}) - \psi(x)\right)$$

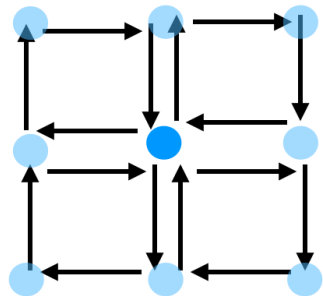
$$\Delta_\mu\psi = \frac{1}{a}\left(\psi(x) - U_\mu^\dagger(x-\hat{\mu})\psi(x-\hat{\mu}) - \psi(x)\right).$$



The Matrix for Twisted Mass Fermions

- The clover-improved Wilson Dirac operator:

$$D_{\mathrm{cW}} = D_{\mathrm{W}} - \frac{c_{sw}}{32} \sum_{\mu,\nu=0}^{3} (\gamma_\mu \gamma_\nu) \otimes (Q_{\mu\nu}(x) - Q_{\nu\mu}(x))$$



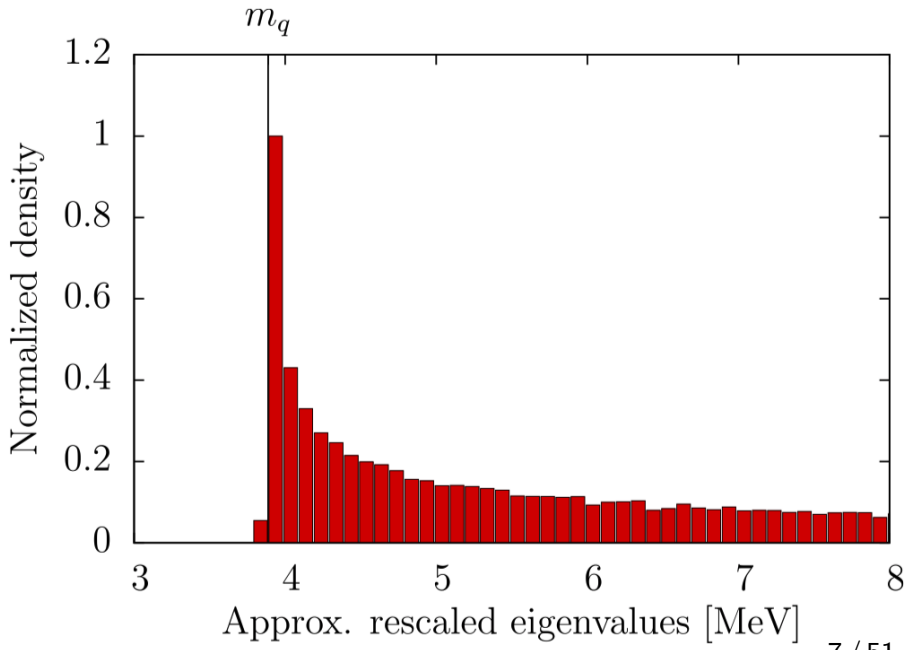
where $Q_{\mu\nu}(x) = \sum_{n=0}^{3} U_{\mu\nu}^{\circlearrowleft n}(x)$ and

$$U_{\mu\nu}(x) = U_\mu(x) U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x)$$
$$U_{\mu\nu}^{\circlearrowleft}(x) = U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x) U_\mu(x)$$

- The degenerated twisted mass operator:

$$D_D(\mu) = (D_{\mathrm{cW}} \otimes I_2) + i\mu(\Gamma_5 \otimes \tau_3) = \begin{pmatrix} D_{\mathrm{TM}}(\mu) & 0 \\ 0 & D_{\mathrm{TM}}(-\mu) \end{pmatrix}$$

where $D_{\mathrm{TM}}(\mu) = D_{\mathrm{cW}} + i\mu\Gamma_5$



Why do we need a linear solver?

- Typically, for modern simulations, the size of the lattice is $V = 128 \times 64^3 \sim 3 \times 10^7$.
- Then, a typical size of the matrix is $n \times n$ with $n = V \times \underbrace{4}_{\text{spins}} \times \underbrace{3}_{\text{colors}} \times \underbrace{2}_{\text{complex}} \sim 8 \times 10^8 \simeq 6.4\text{GB}$
- For general matrix inversion, the computational cost is $\mathcal{O}(n^{2.376})$ (Optimized CW-like algorithms)
- If we were to store all entries of the inverse, we need the memory space of $n \times n \simeq 6.5 \times 10^{17} \simeq 5 \times 10^9\text{GB} = 5\text{EB}$
- It is impractical in lattice computations to invert D explicitly and store all its entries.

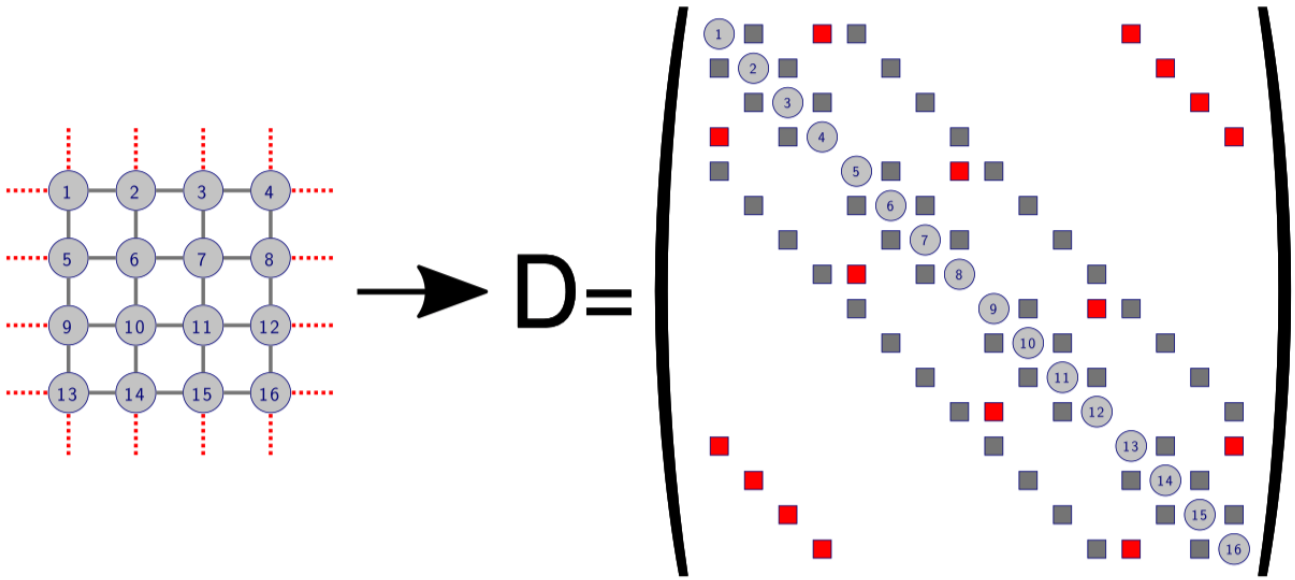
Why do we need a good linear solver?

- We switch to solving $Dx = b$.
- b is usually a random vector or a point source, i.e., $b = \delta_{xa,yb}$ where x, y are lattice sites and a, b are internal quantum numbers.
- In practice, computation of point-to-all propagators, i.e., the solutions of $Dx = b$ with a point source, is often sufficient, e.g., for two-point correlation functions.
- The inversion needs to be done a lot of times.
- For example, computation of nucleon structure requires $\underbrace{4}_{\text{spins}} \times \underbrace{3}_{\text{colors}} \times \underbrace{2}_{\text{flavors}} \times \underbrace{5}_{\text{proj}} \times \underbrace{200}_{\text{src pos}} \times \underbrace{400}_{\text{configs}} \sim 10^7$ inversions.

We need a good solver!

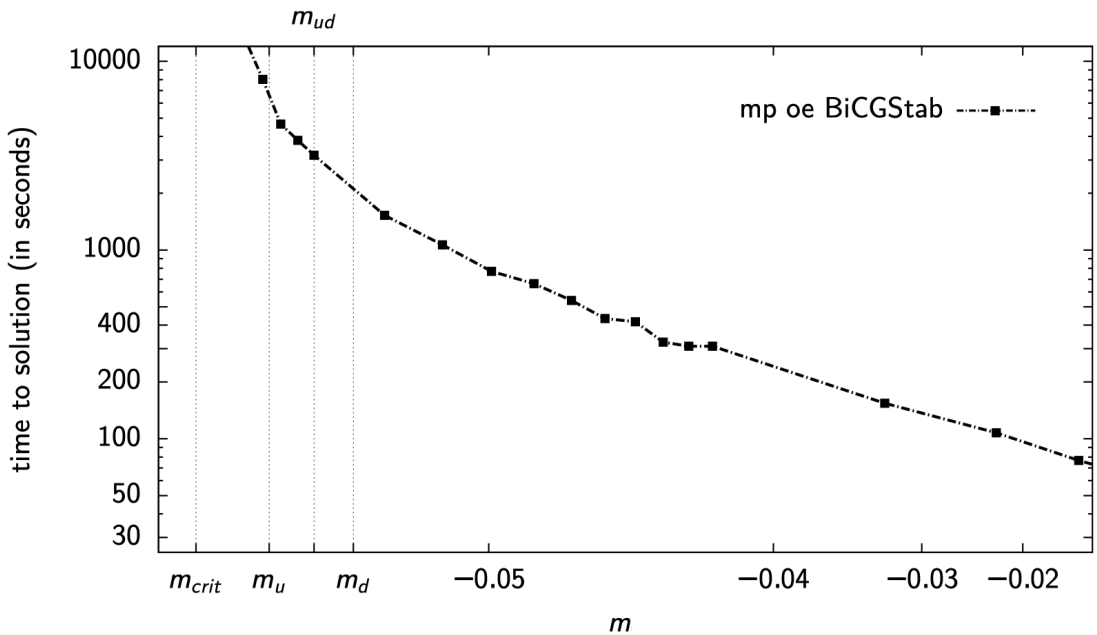
Krylov Solver

- Due to sparse nature of the matrix, the computational cost for matrix-vector multiplication is $\mathcal{O}(n)$
- We use a Krylov method, which consists of constructing a Krylov subspace $\mathcal{K}_k = \text{span}\{b, Db, D^2b, D^3b, \cdots, D^{k-1}b\}$, projecting the system onto $\mathcal{K}$, and solving the projected system to obtain an approximate solution x_k.
- As it is a projection-based method, it does not require a large memory space.



Krylov Solver - Critical Slowing Down

- The condition number for D is roughly proportional to the inverse of the mass.
- The matrix D becomes singular as the mass approaches its critical value m_{critc}.
- Convergence to the solution slows down.
- Inversion on and generation of gauge configuration at physical light quark masses leads to a larger condition number and thus increasing solver time.



Outline

Multigrid Solvers

Among various solvers, a class of solvers based on multigrid approaches in preconditioning Krylov subspace solvers has turned out to be very successful (Luscher 2007b; Luscher 2007a; Osborn et al. 2010; R. Babich et al. 2010; Frommer et al. 2013).

Several of the implementations for clover Wilson fermions that are openly available includes

- A two-level multigrid approach based on Lüschers inexact deflation (Luscher 2007b) available as OpenQCD at http://luscher.web.cern.ch/luscher/openQCD/
- Multigrid with generalized conjugate residual (MG-GCR) (J. Brannick et al. 2008; Clark et al. 2008; Ronald Babich et al. 2009; R. Babich et al. 2010) available as part of the USQCD package QOPQDP at https://github.com/usqcd-software/qopqdp and QUDA at http://lattice.github.io/quda/
- An aggregation-based domain decomposition multigrid (DDαAMG) approach (Frommer et al. 2013) available at https://github.com/DDalphaAMG/DDalphaAMG.

Extending Multigrid Solvers

Extension of DDαAMG to twisted mass fermions:

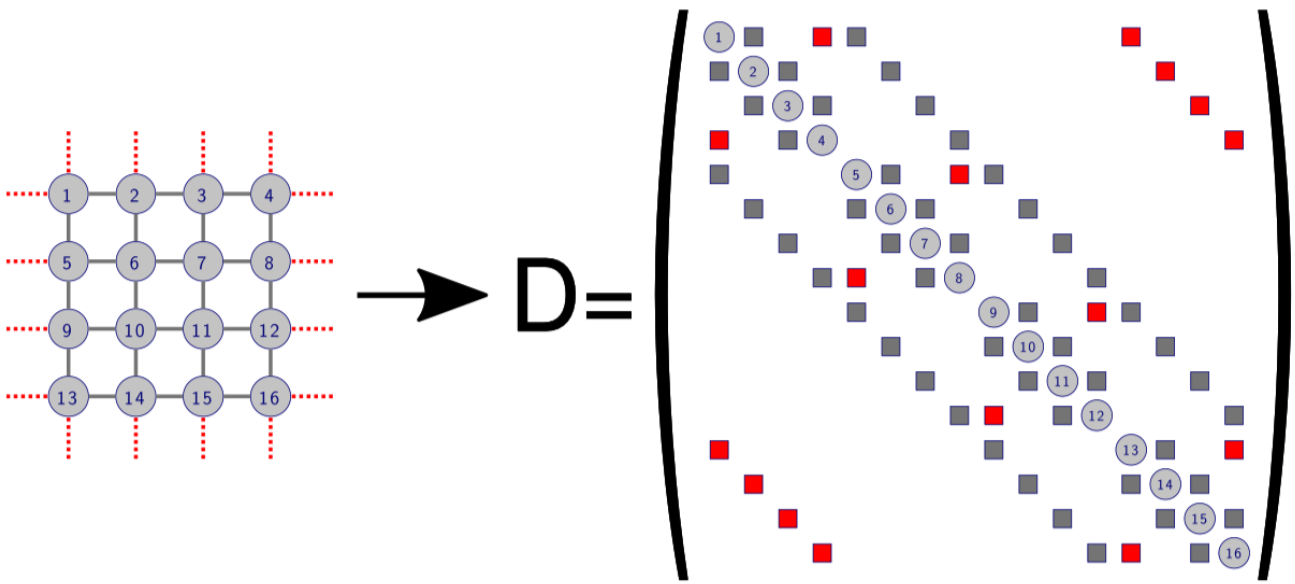
- C. Alexandrou, S. Bacchio, J. Finkenrath, A. Frommer, F. Kahl, and M. Rottmann, *Adaptive Aggregation-based Domain Decomposition Multigrid for Twisted Mass Fermions*, Phys. Rev. D **94**, 11 (2016)
- The generalization to non-degenerate twisted mass fermions is discussed in (Alexandrou, Bacchio, and Finkenrath 2019)
- Publicly available at https://github.com/sbacchio/DDalphaAMG
- A version with new features at https://github.com/sy3394/DDalphaAMG

Extension to other fermions:

- MG-GCR to domain-wall fermions in (Cohen et al. 2011)
- DDαAMG to overlap fermions in (James Brannick et al. 2016)

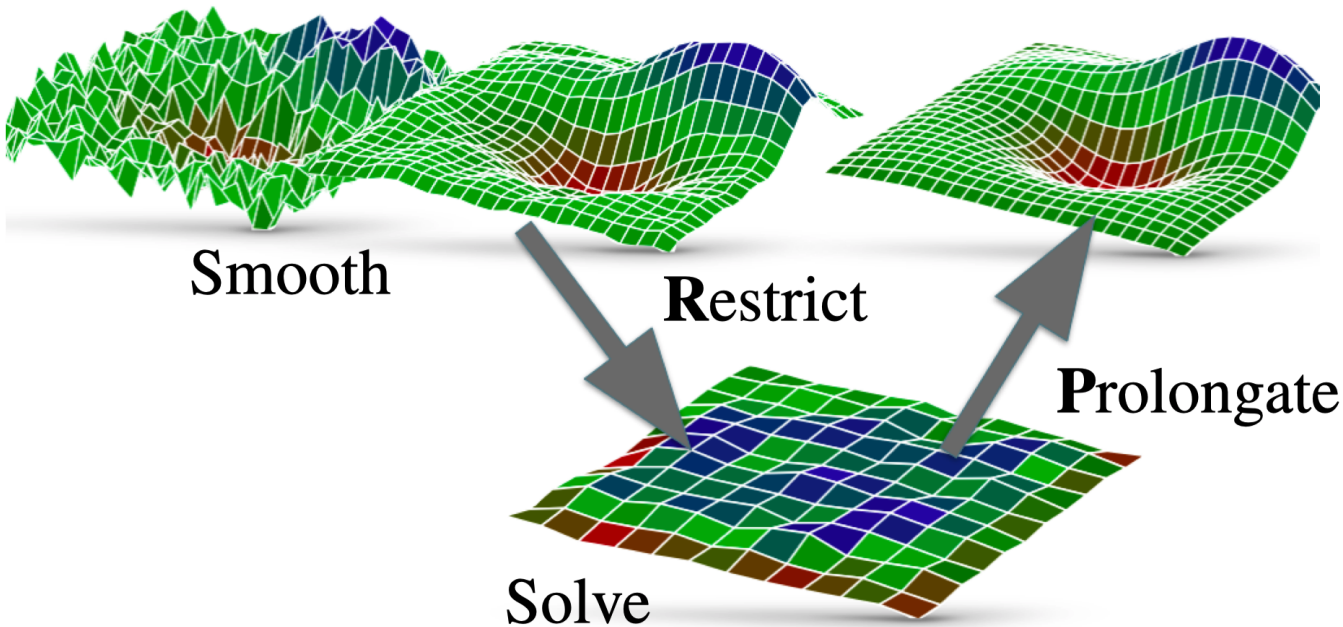
DDαAMG - Basics

- Adaptive Aggregation-based Domain Decomposition Multigrid method
- The algorithm is designed to invert a large sparse matrix effectively
- In particular, the algorithm is applied to Dirac matrices for Wilson or twisted mass fermions
- It takes advantage of the sparse structure of the Dirac matrix.



DDαAMG - Preconditioners

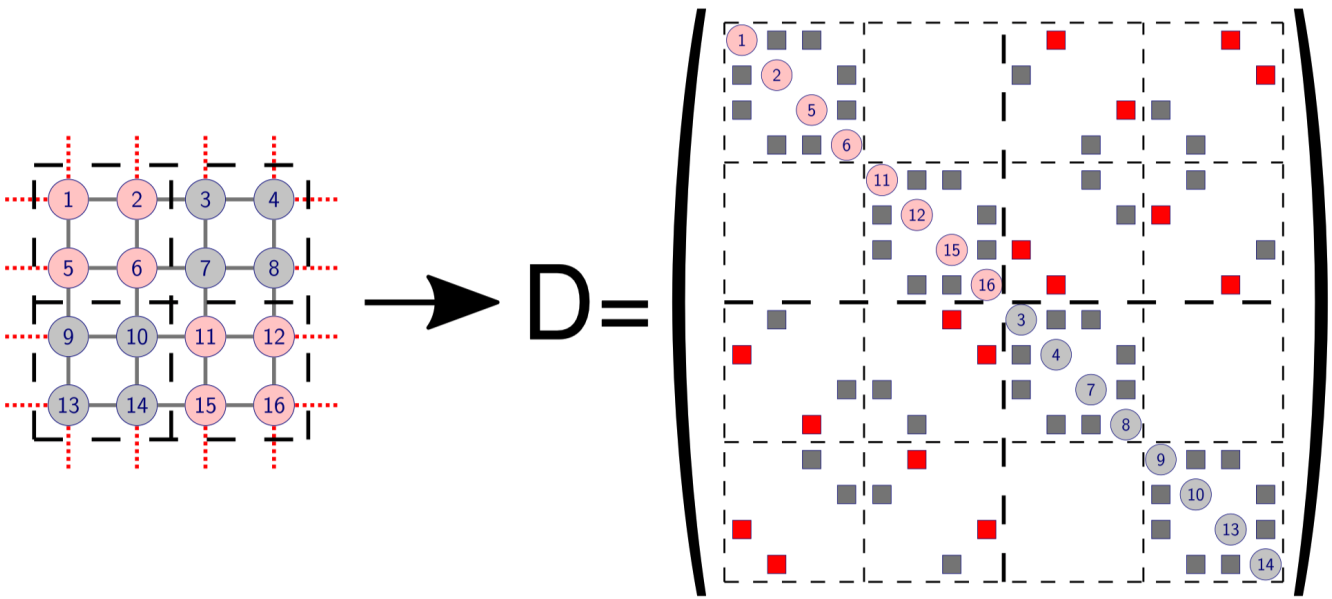
- To circumvent the issue of critical slowing down and effectively invert the large sparse matrix, DDαAMG uses two preconditioners: a smoother and coarse grid correction
- For a smoother, we use red-black Schwarz Alternating Procedure (SAP) (Luscher 2007a).
- For coarse grid correction, we use Algebraic MultiGrid (AMG) (Wesseling 1995).



Picture Courtesy: Luke Olson, http://lukeo.cs.illinois.edu/cs556

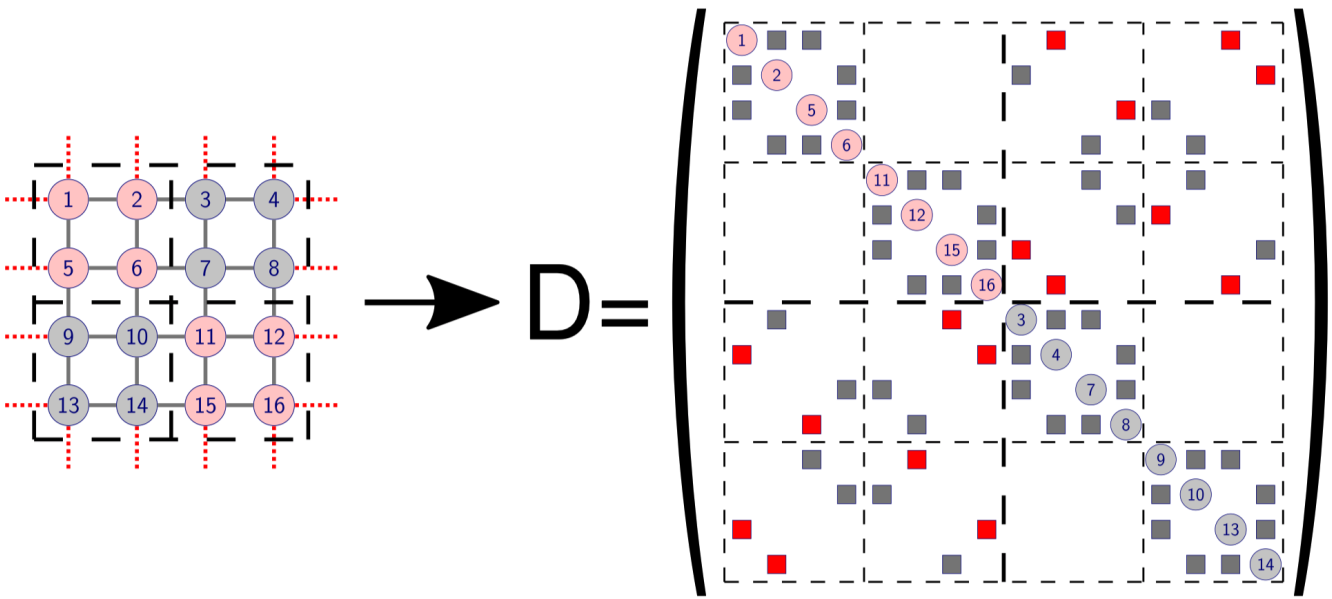
SAP

- The lattice is divided into red blocks and black blocks in a chessboard manner.
- Due to nearest neighbor interactions, block of a color couples to a block of the other color
- After reordering, $D = \begin{pmatrix} D_{rr} & D_{rb} \\ D_{br} & D_{bb} \end{pmatrix}$



SAP

- In SAP, we neglect off-diagonal parts, D_{rb}, D_{br}, i.e., inter-block interactions.
- Then, within each group of blocks, we visit blocks sequentially and invert the block matrix locally.



SAP

- Depending on when we perform the global residual update, error propagates differently.
- In additive SAP, residual update, $r \leftarrow b - Dx$, is done once before local inversion over blocks.

$$\varepsilon \leftarrow \left(I - \sum_i D_i^{-1} D \right) \varepsilon$$

- In multiplicative SAP, residual update is done every time before local inversion on each block.

$$\varepsilon \leftarrow \left[\prod_i (I - D_i^{-1} D) \right] \varepsilon$$

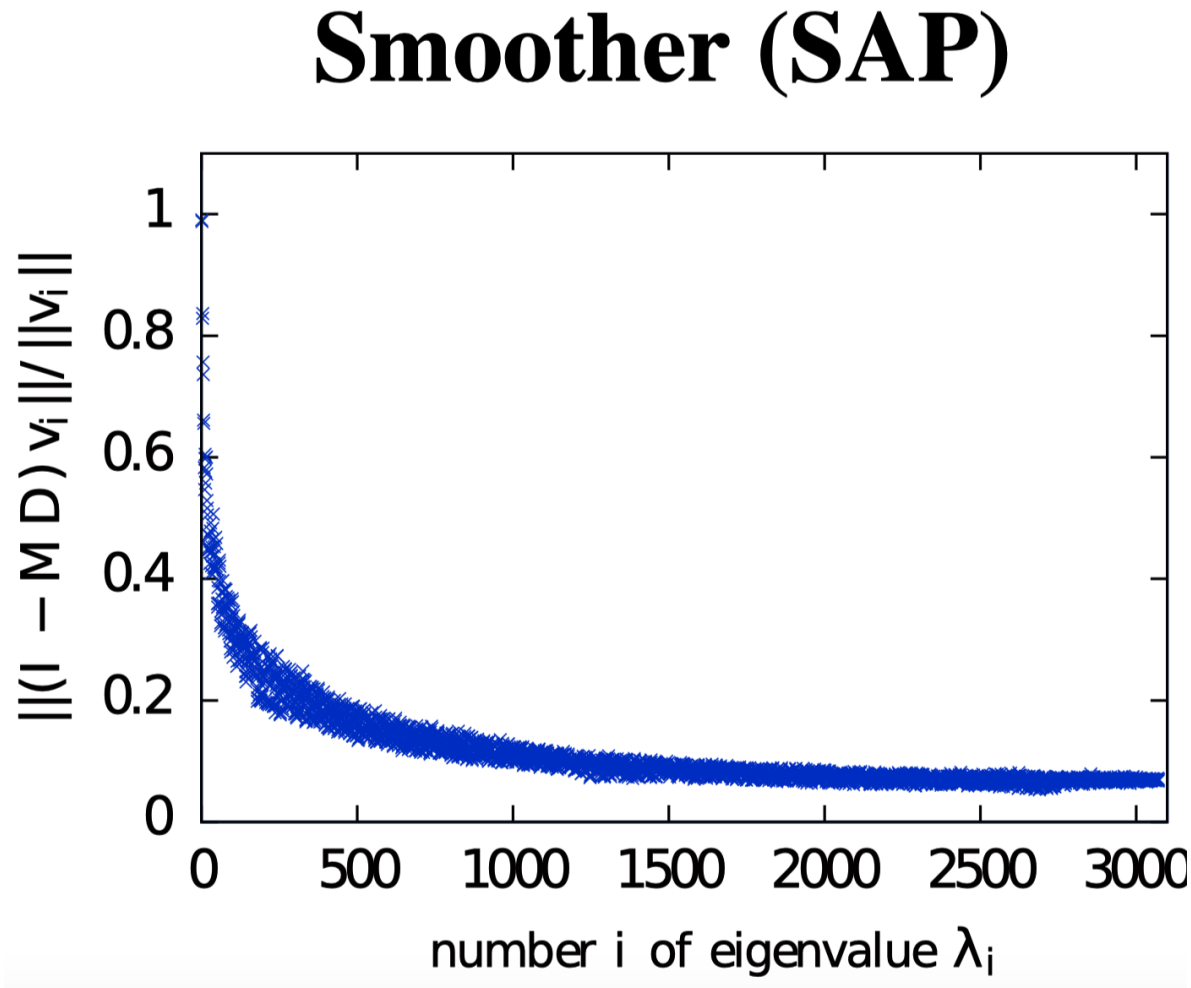
- Information spread faster with multiplicative SAP.

SAP

- In red-black SAP, within a group of blocks of each color, additive SAP is adopted, and over groups of blocks, multiplicative SAP is used.
- Define
$$B_{rr} = \begin{pmatrix} D_{rr}^{-1} & 0 \\ 0 & 0 \end{pmatrix} \text{ and } B_{bb} = \begin{pmatrix} 0 & 0 \\ 0 & D_{bb}^{-1} \end{pmatrix}.$$
- Then, error propagation is given by
$$\varepsilon \leftarrow (I - MD)\varepsilon = (I - B_{rr}D)(I - B_{bb}D)\varepsilon.$$
- Since local block matrices live on a single MPI rank, their inversion does not call for global reduction so that SAP is suitable for parallelization.

SAP

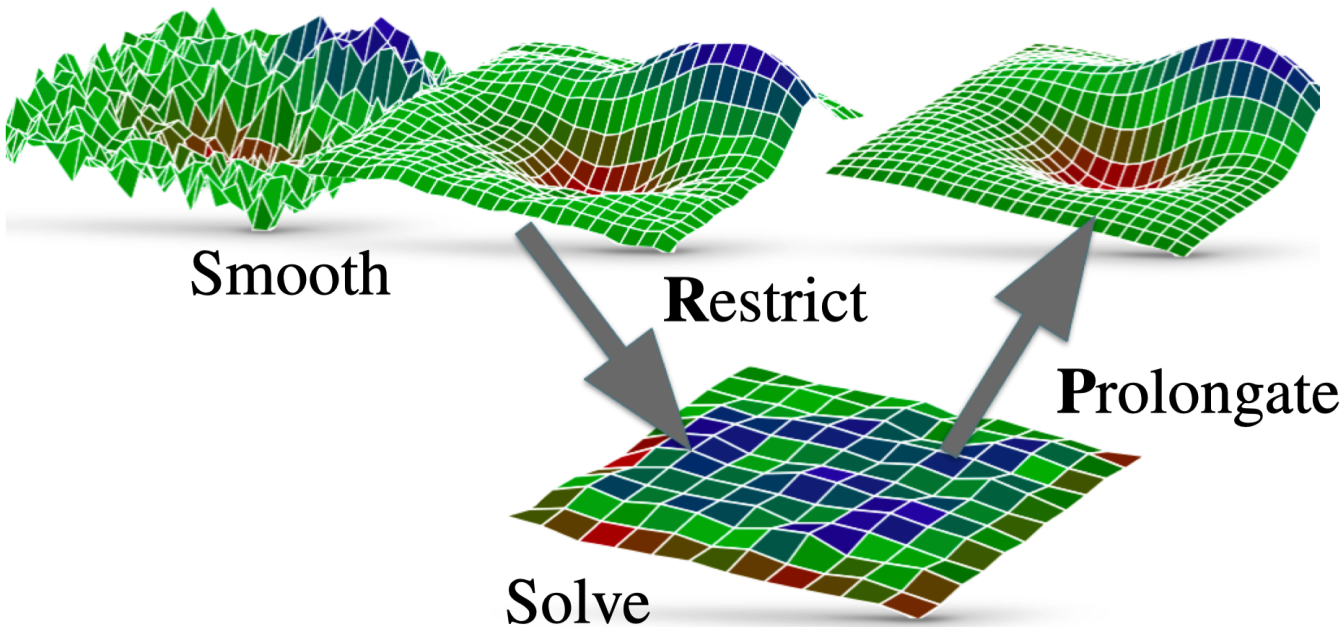
- As it involves inversion of local block matrices, it removes UV-modes from the residual



Coarse Grid Correction

Basic Ideas:

- Start with $Dx = b$
- Restrict the problem onto a coarse grid
- Solve $D_c x_c = r_c$
- Prolong the solution back to the original grid



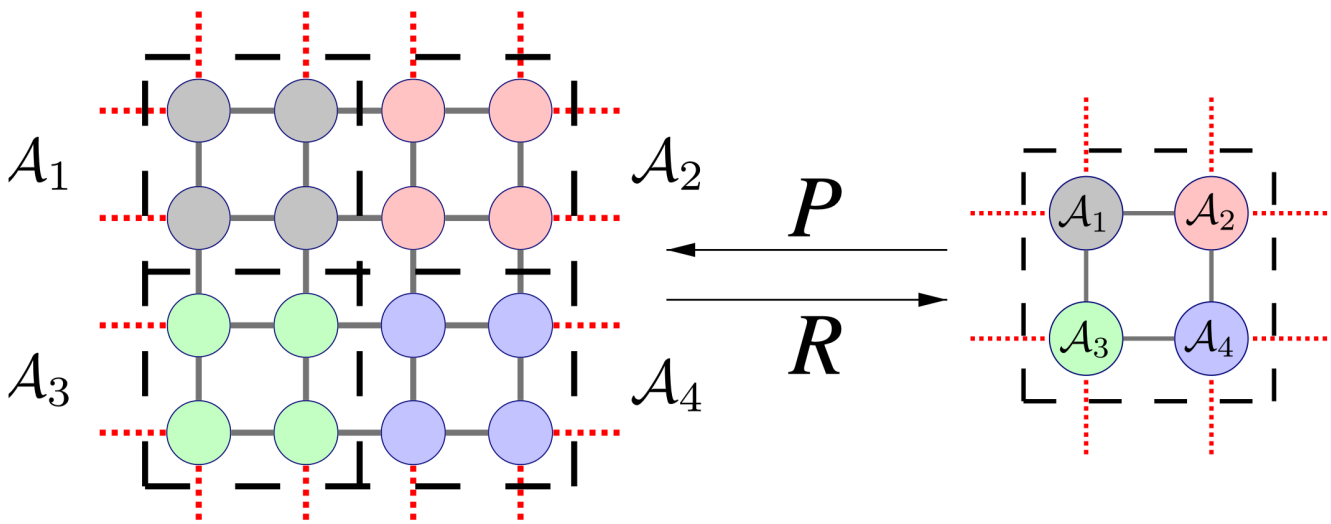
Picture Courtesy: Luke Olson, http://lukeo.cs.illinois.edu/cs550

MG

To perform coarse-grid correction, we use a multigrid method. For that, we need

- the restriction operator, R, which maps the problem from a finer lattice to a coarser lattice
- the prolongation operator, P, which maps the problem from a coarser lattice to a finer lattice
- $RIP = I_c$

The coarsened problem is significantly cheaper to solve.



MG

Using those two operators, we construct a coarse-grid Dirac matrix, $D_c = RDP$, and coarse-grid vectors, $x_c = Rx$, $r_c = R(b - Dx)$. Then,

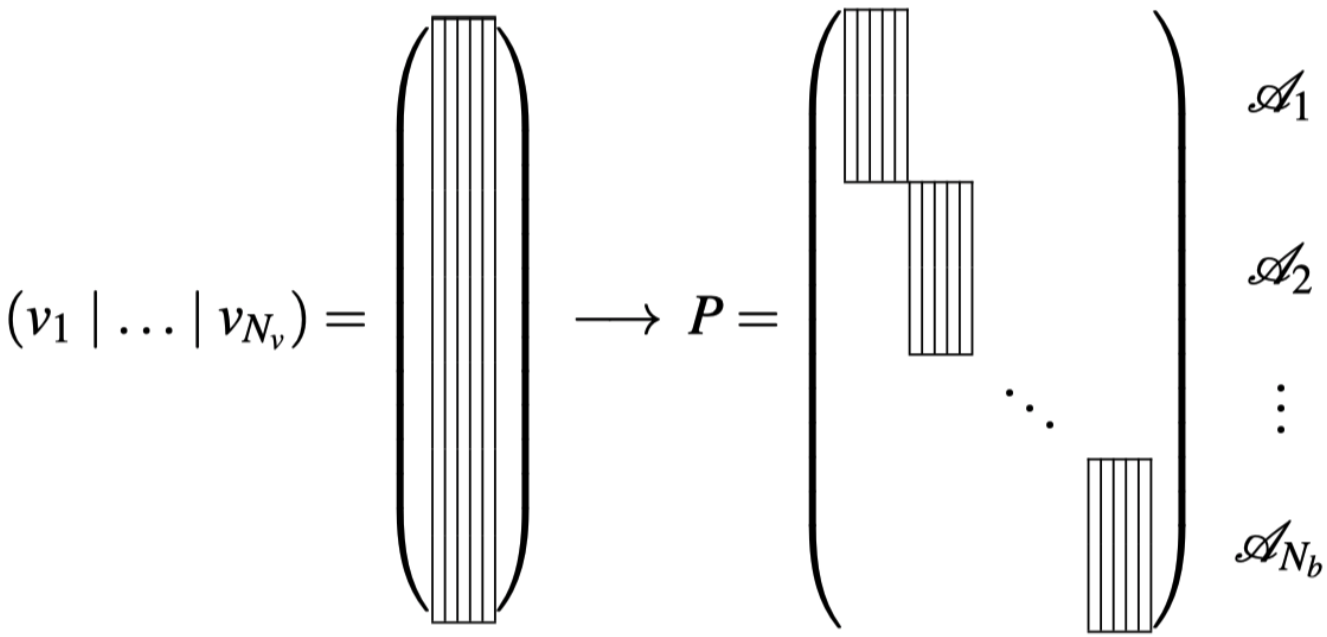
- Coarse-grid inversion: $D_c x_c = r_c$
- Coarse-grid correction: $x \leftarrow x + PD_c^{-1}R(b - Dx)$
- Error propagation: $\varepsilon \leftarrow (I - PD_c^{-1}RD)\varepsilon$

In multi-level approaches, we generalize the coarse-grid correction to

- $R_l I_l P_l = I_{l+1}$
- $D_{l+1} = R_l D_l P_l$ with $D_1 = D$
- $D_{l+1} x_{l+1} = r_{l+1} = R_l(b_l - D_l x_l)$
- $x_l \leftarrow x_l + P_l D_{l+1}^{-1} R_l(b - D_l x_l)$
- $\varepsilon_l \leftarrow (I_l - P_l D_{l+1}^{-1} R_l D_l)\varepsilon_l$

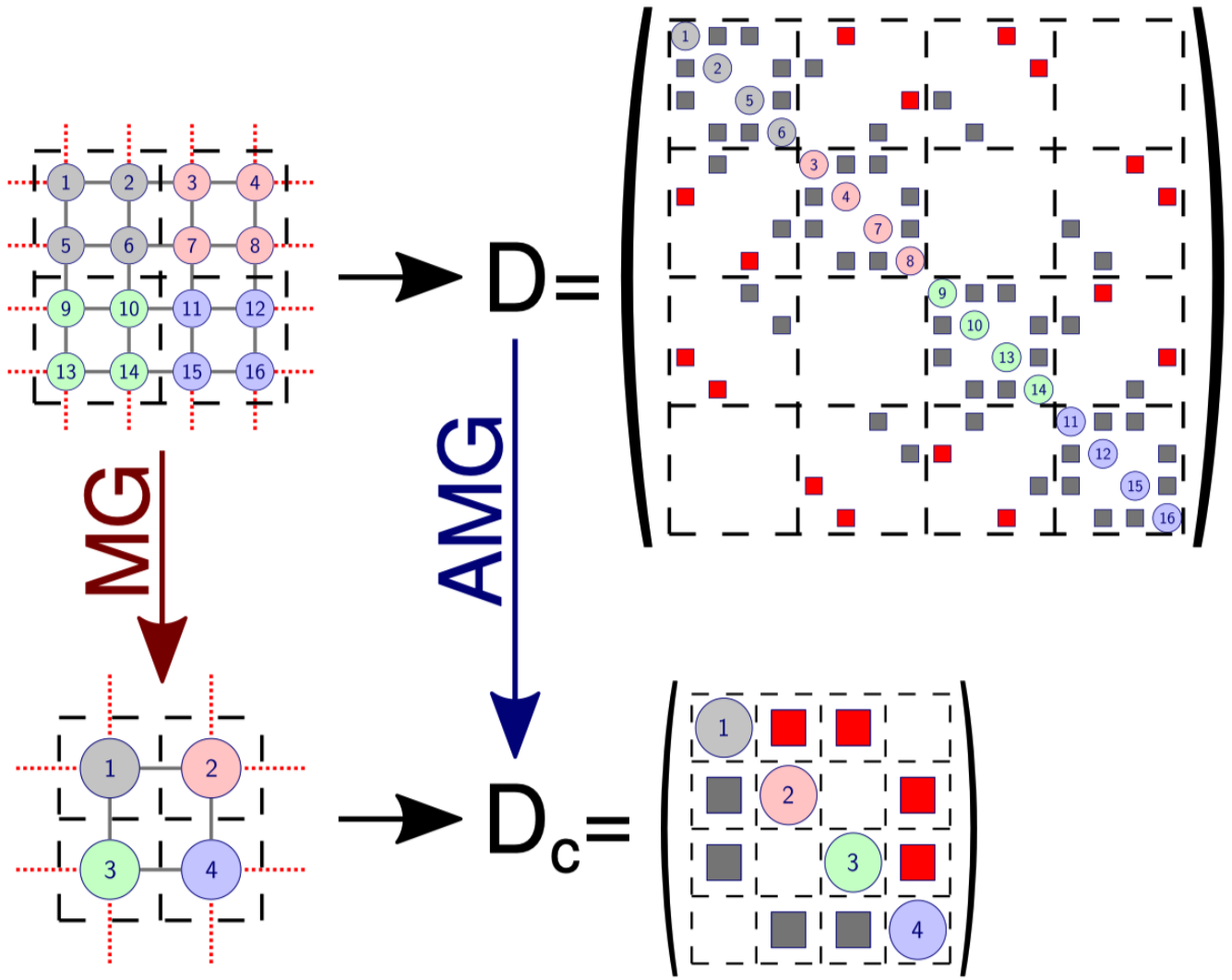
AMG

- For a standard MG methods, restriction and prolongation are defined geometrically, for example using block aggregation.
- In algebraic multigrid approaches, we do not base the definition of P and R on geometry of the lattice.
- Instead, we use eigenvectors of the Dirac operator with small eigenvalues to define P and R based on partition of the lattice into aggregates.
- This constructed P effectively captures the low-modes of D due to local coherence (Luscher 2007b).



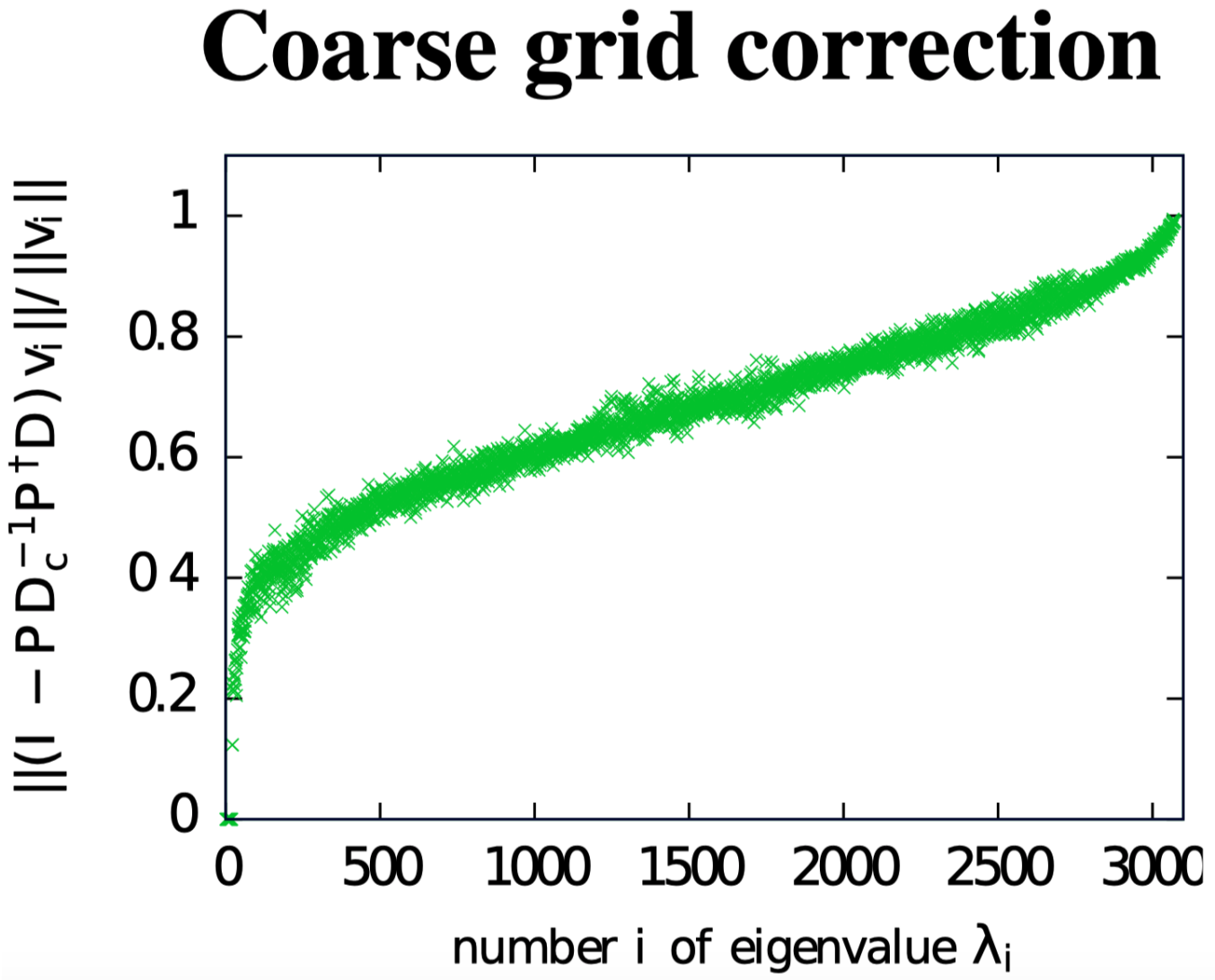
AMG

So in AMG, we define a coarse-grid Dirac matrix directly from the Dirac matrix on the fine lattice.



AMG

Coarse-grid correction removes low-modes from the residual.

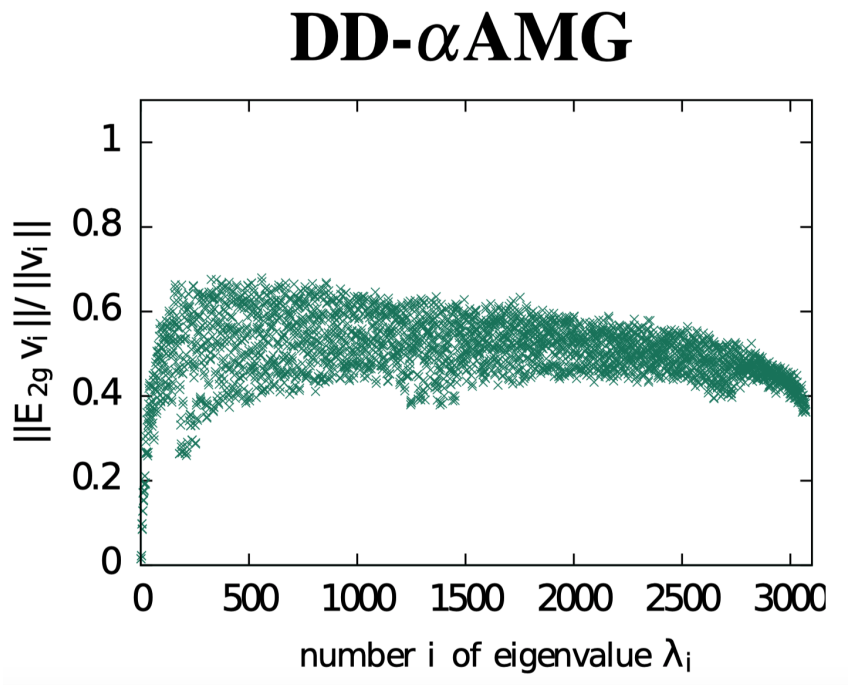


SAP+AMG

So when we combine the two preconditioners as follows,

$$\varepsilon_l \leftarrow (I - M_l D_l)^k (I_l - P_l D_{l+1}^{-1} R_l D_l)(I_l - M_l D_l)^j \varepsilon_l,$$

the low-modes and high-modes of the error are both suppressed.

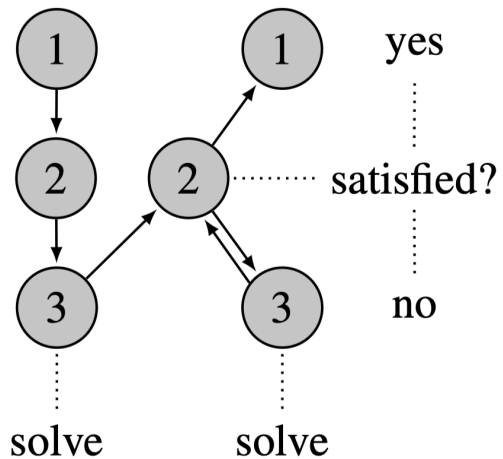


DDαAMG with SAP and AMG

- Iterative adaptive setup phase:
  - We construct P and so the coarse Dirac operator at each level.
  - Instead of applying eigensolver to D to get a few approximate eigenvectors with small eigenvalues, we construct them in an adaptive manner using two preconditioners.
  - First, start with a set of random vectors, v_i's, solve $Dv_i = 0$ using only smoother to reduce the high modes from v_i, and construct initial multigrid hierarchy.
  - Then, compute $D_l^{-1} v_i$ with post-smoothing at each level and update v_i as well as P and D_c.

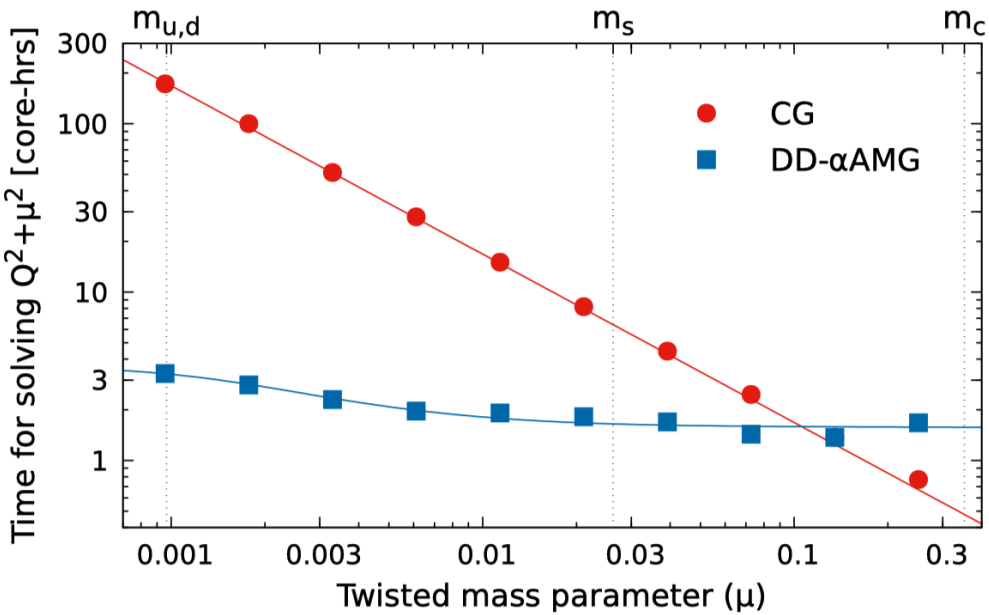
DDαAMG with SAP and AMG

- Solver phase:
  - We combine coarse-grid correction and Krylov method with a cycling strategy called "K-cycle" to obtain a recursive Krylov solver.
  - Residuals, max Krylov space size, and a number of restarts can be set to different values at different levels.
  - In particular, the residual of 10^{-1} is sufficient at the bottom even when the goal residual at the top is 10^{-10}.



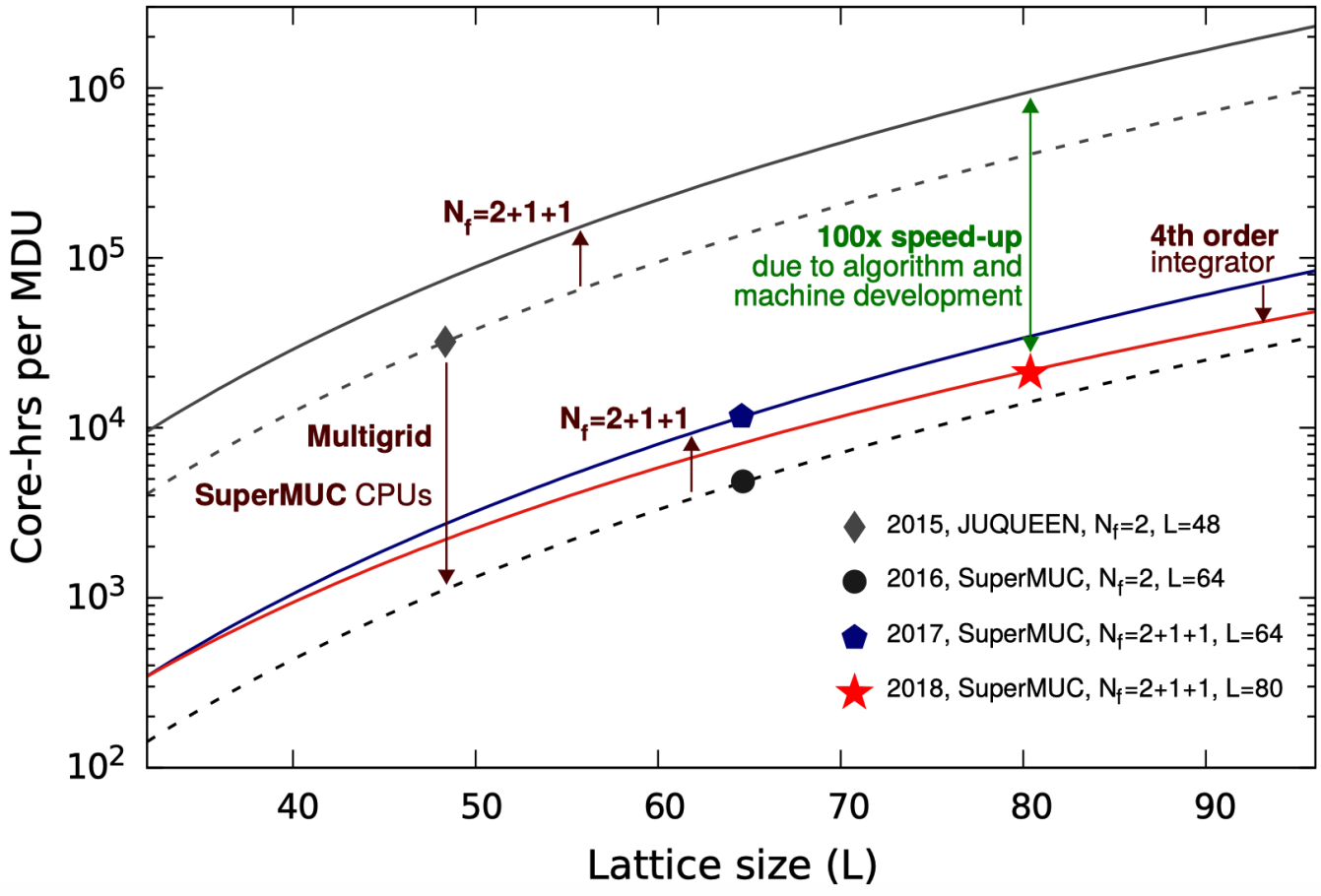
DDαAMG - Performance

- MG solvers outperforms traditional Krylov subspace solvers like the conjugate gradient solver at small quark masses
- DDalphaAMG for twisted mass fermions is two orders of magnitude faster than CG



DDαAMG - Performance

- HMC simulation:



DDαAMG - Scaling

- Bottlenneck of Multigrid methods is the scalability
- Ideal scaling breaks down, and performance stagnates for parallelization above 125 Skylake nodes in case of a 3-level MG approach
- With the current hardware trends higher core counts per node the scalability window will even shrink further

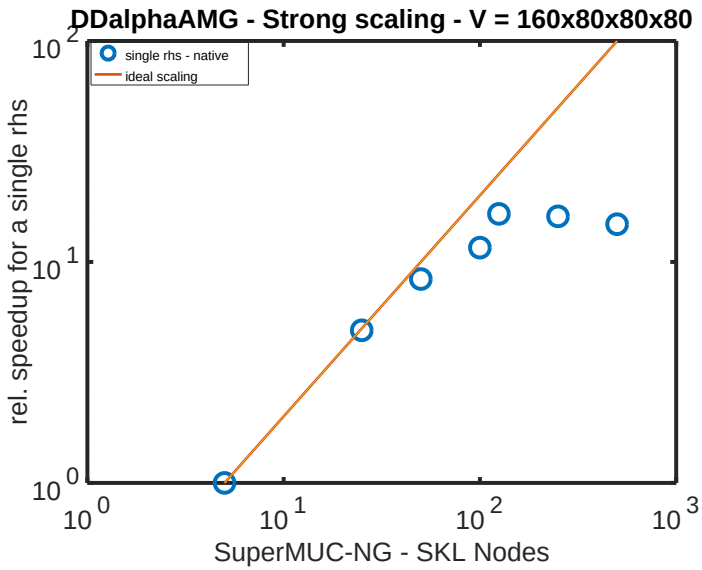


Figure 2: A scaling plot on the ensemble of $N_f = 2 + 1 + 1$ twisted mass clover with $a \sim 0.07$fm and $V = 80^3 x160$ at physical point simulated on SuperMUC-NG (Intel Xeon ("Skylake")) at LRZ

Outline

Multiple R.H.S. - Objectives

Originally,

- the code inverted each rhs one by one
- also, loops are vectorized manually using instruction sets for a specific SIMD extension

However,

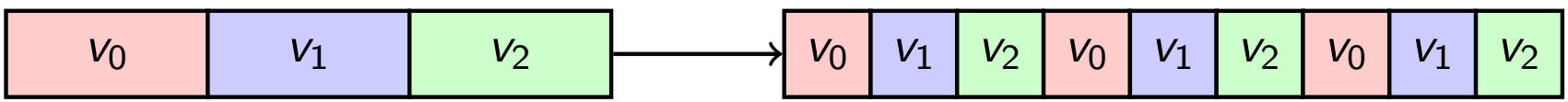
- We can perform multiple inversions more efficiently.
- We also want to improve portability of our code letting compilers perform optimization analysis and vectorization.

Thus,

- We solve the system of equations with multiple right-hand sides (rhs) simultaneously ($b \rightarrow \mathbf{b}$).

Multiple R.H.S. - Implementation

- We define a new data structure for a bundle of vectors.
- Vectors in the bundle are ordered in such a way that the index on vectors runs the fastest.
- All low-level routines are rewrited to respect the new structure.
- We process a bundle of right-hand vectors simultaneously using SIMD vectorization of loops.



Multiple R.H.S. - Implementation

- Instead of manually vectorizing the loops using instruction sets, we auto-vectorize the loops using pragmas: `_Pragma("unroll")`, `_Pragma("vector aligned")`, and `_Pragma("ivdep")`.
- These pragmas are applied to a for-loop of a pre-determined iteration length: `for( jj=0; jj<num_loop; jj++)`.
- The number of rhs are assumed to be multiple of `num_loop`.
- This shifts vectorization from 128 bit to 256 bit

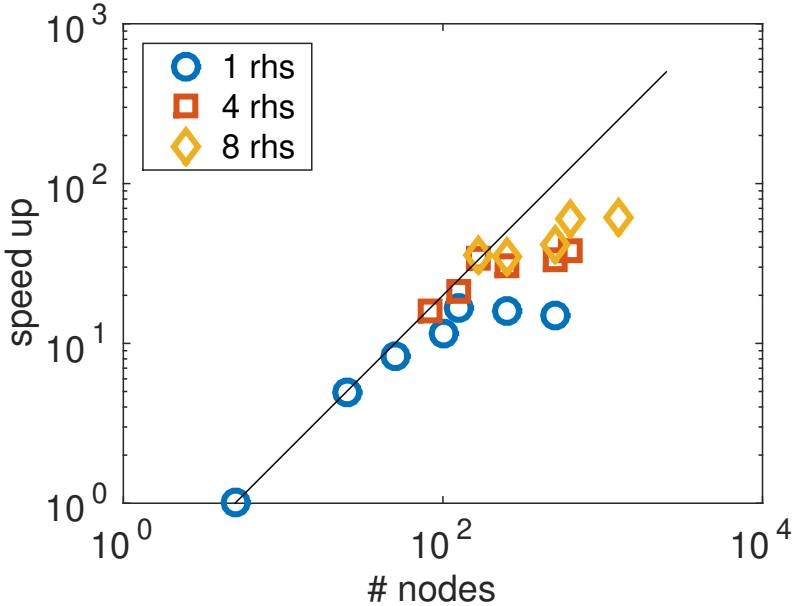
Num. R.H.S.	1 rhs		4 rhs		8 rhs	
Instruction Mix	SP Flops	DP Flops	SP Flops	DP Flops	SP Flops	DP Flops
128-bit	95.26%	86.59%	23.41%	4.99%	24.92%	3.60%
256-bit	2.58%	1.26%	60.68%	78.13%	74.02%	94.76%
Total	97.26%		84.03%		98.81%	

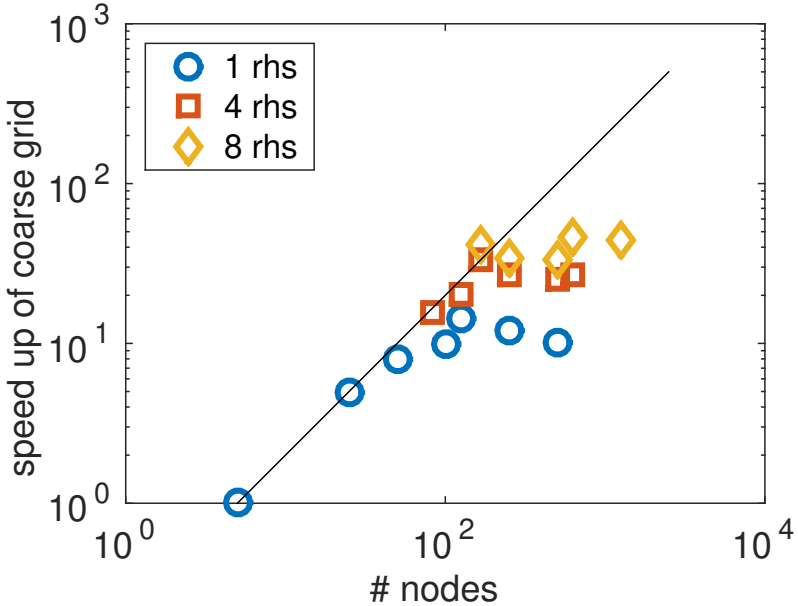
Table 1: Vectorization Reports

Scaling

Conclusion:

- Breakdown of strong scaling can be pushed to higher parallelization, mutiple rhs shows scalability up to 512 nodes





Outline

Incorporating Block Krylov Solver

- Fast Accurate Block Linear krylOv Solver (Fabulous) is an external library implementing block Krylov solvers such as GMRES and GCR (Robbé and Sadkane 2006; Morgan 2005; Agullo, Giraud, and Jing 2014)
- It combines BGMRES with detection of inexact breakdown, deflation, and incremental QR factorization.
- It provides several different orthogonalization schemes.
- We linked the DDαAMG code to Fabulous and make it available non-block GMRES or one of the solvers provided via fabulous library at each level as a solver.
- Our implementation of multiple r.h.s. stultifies inexact breakdown.

Parameters

For Fabulous,

- Solvers: BGMRES, BGCR, BGMRES with deflated restarting (DR), BGMRES with incremental QR factorization (QR), BGMRES with DR and QR (DRQR)
- The number of deflating eigenvectors (for DR methods).
- Orhogonalization schemes: Classical Gram-Schmidt (CGS), Modified Gram-Schmidt (MGS), Iterative CGS (ICGS), Iterative MGS, each possibly with blocking

For DDαAMG,

- 0^{th}-level (bottom) and 1^{st}-level (intermediate) residuals
- 0^{th}-level (bottom) and 1^{st}-level (intermediate) max Krylov space size and number of restarts

Tuning Constants

The systems used for tuning:

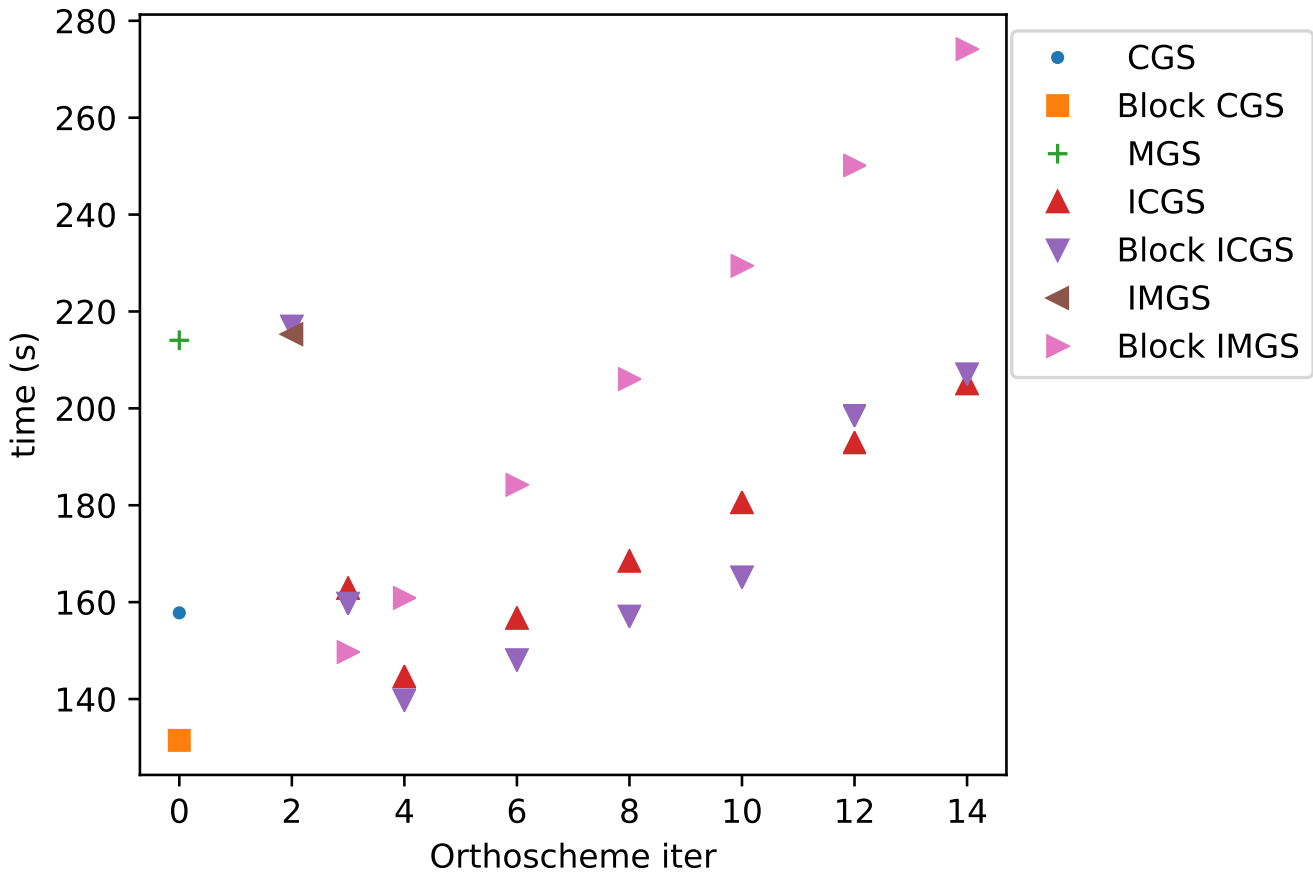
- Lattice: $48^3 \times 98$ at physical point
- System: Cyclone (Intel Xeon Gold 6248) at The Cyprus Institute
- Used 6 nodes with 32 cores each

Fixed solver parameters:

- (The number of levels) $= 3$.
- The solver at the top: FGMRES
- The solver at the level 1: FGMRES

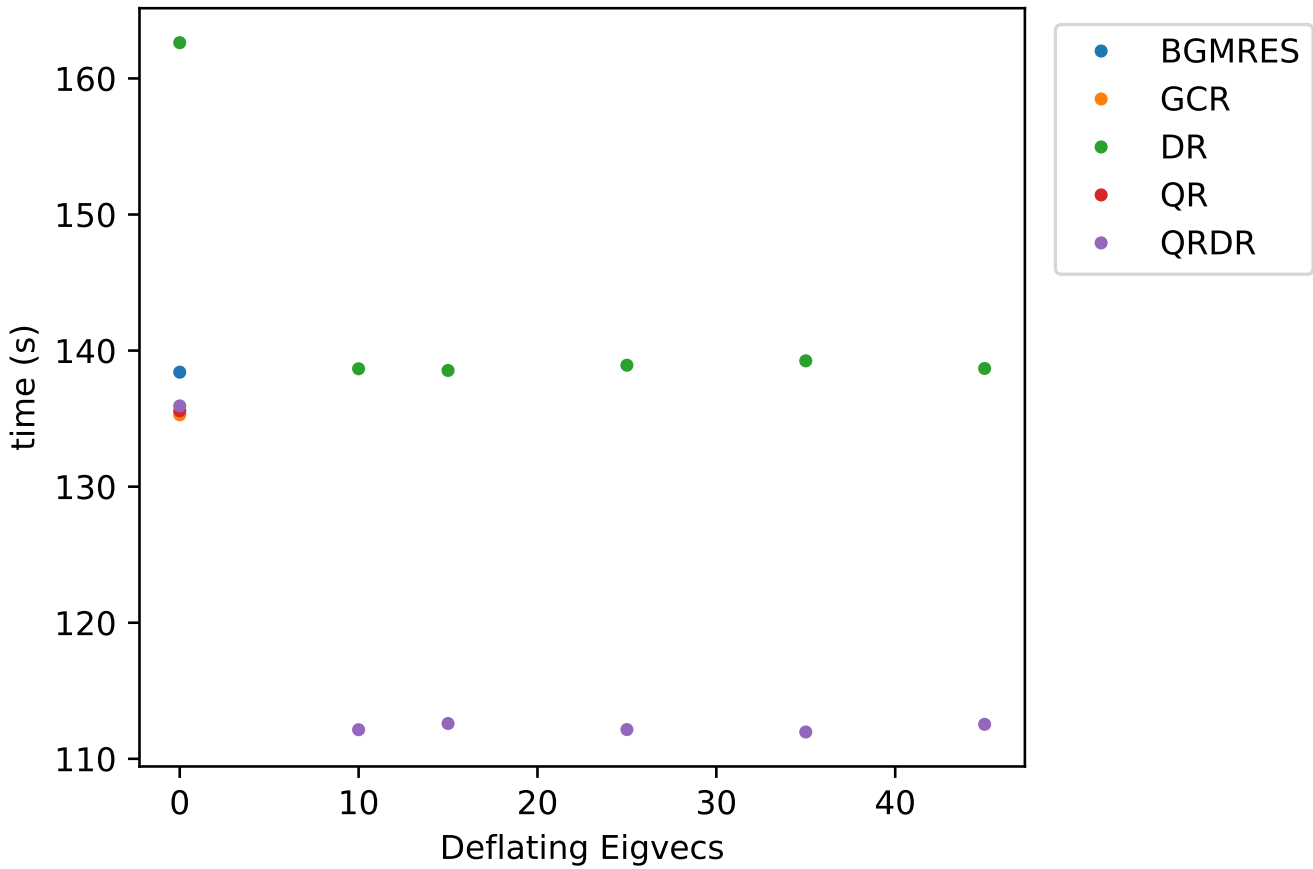
Tuning Orthogonalization Schemes

Solver: BGMRES, Best Orthogonalization Scheme: Block CGS



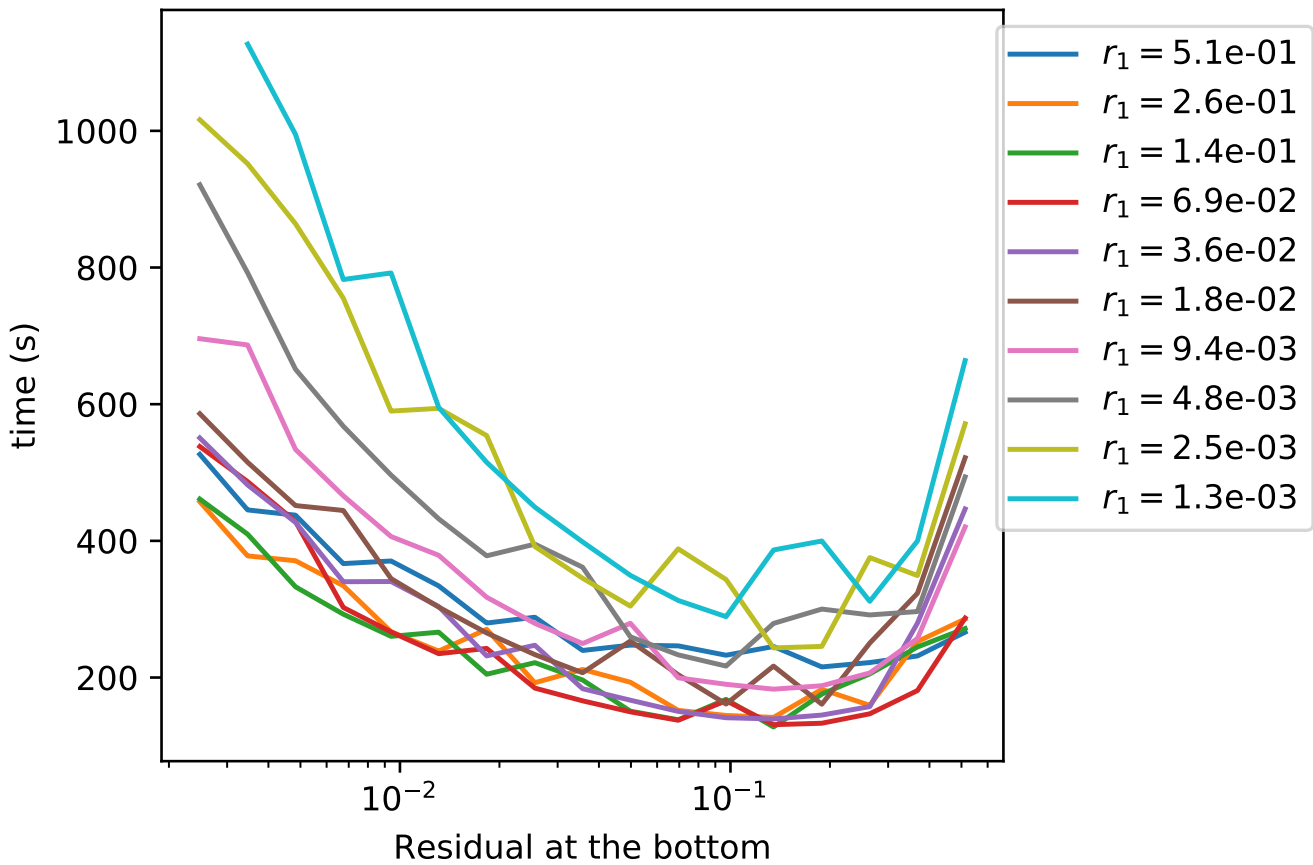
Tuning Solvers

Best Solver: BGMRES with QR and DR



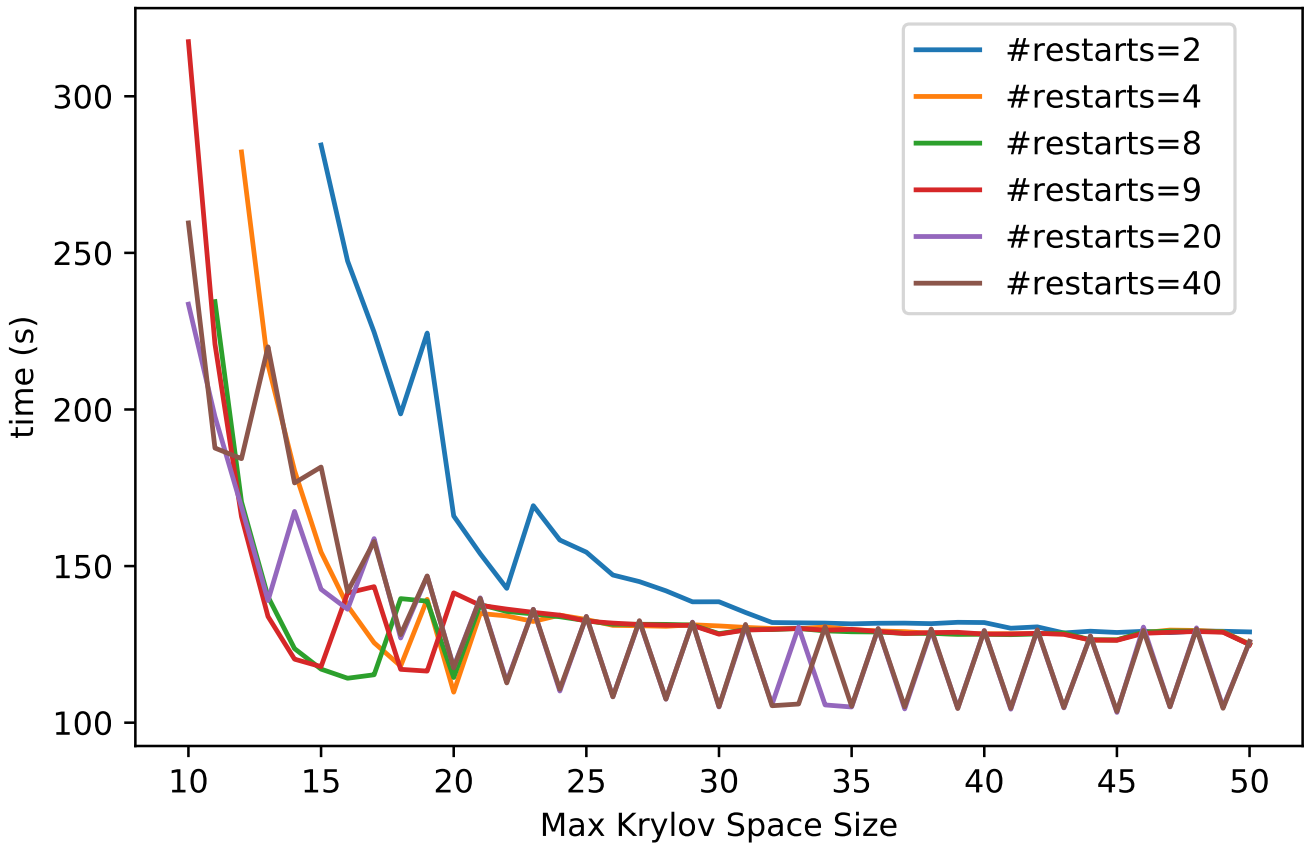
Tuning Residuals

Solver: BGMRES



Tuning Max Krylov Space Size

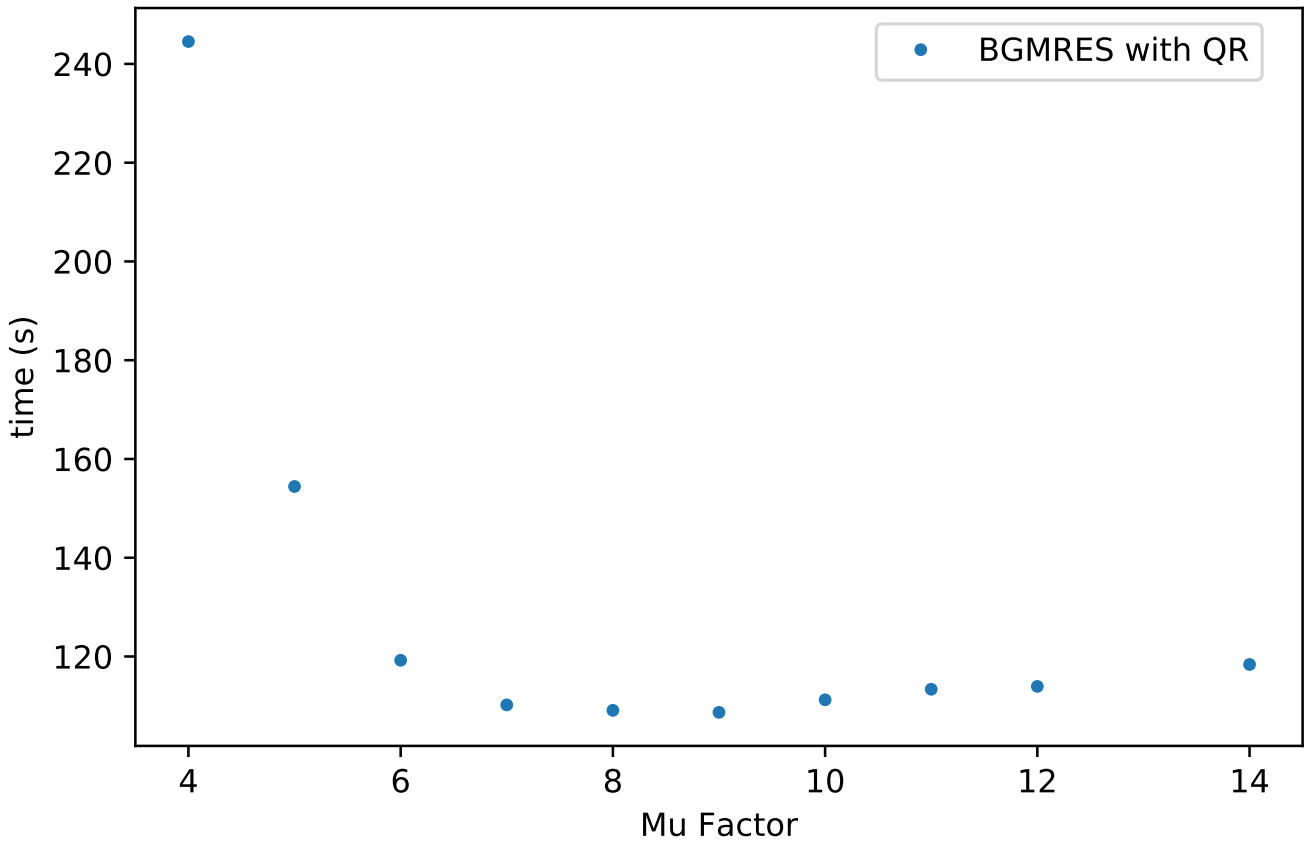
Solver: BGMRES with QR



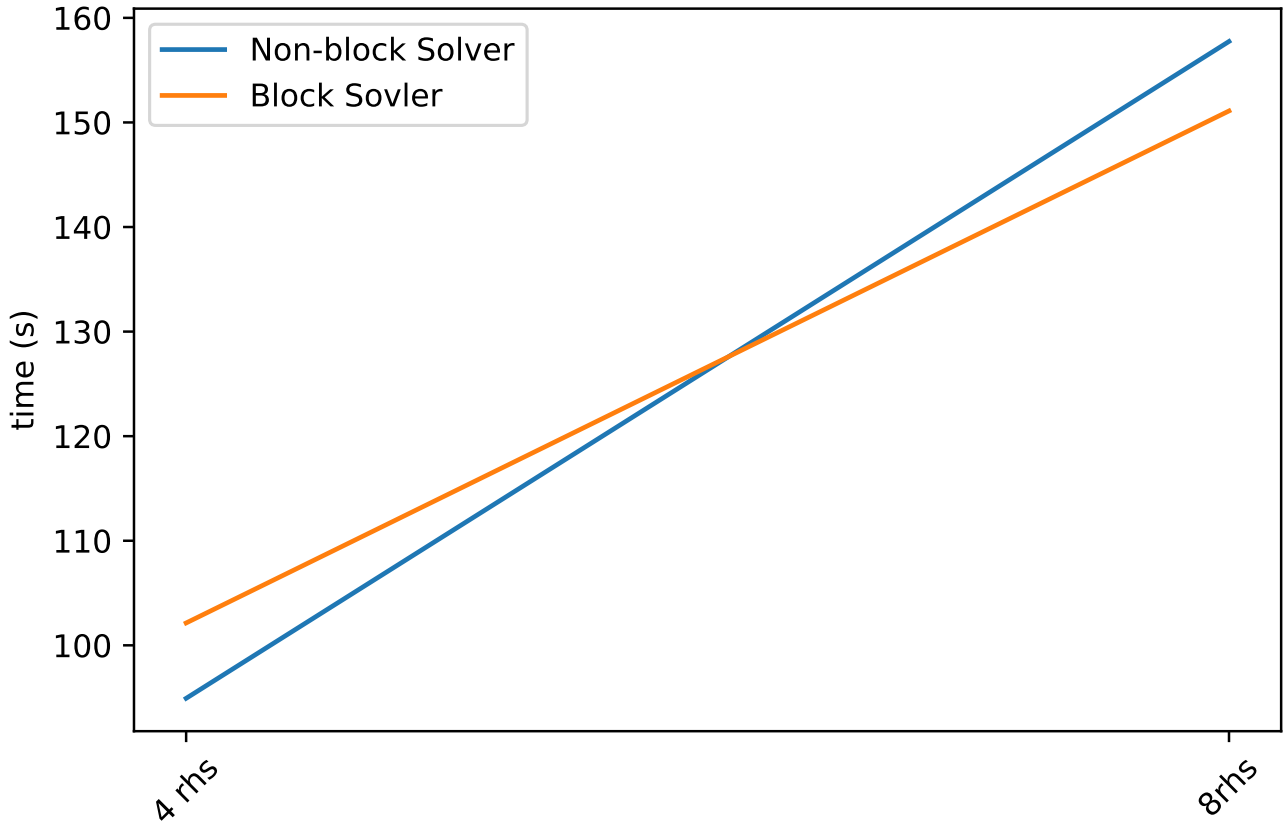
Tuning Mu Factor

- The mu factor is the factor multiplying μ in D_{TM} at the bottom ($\mu \to \delta_\mu \mu$)

Solver: BGMRES with QR



Preliminary Tuning Results



Outline

Outlook

- Scalability is extended by around a factor 5.
- We also want to test out code on different architectures like AMD Epyc's and ARM chips, e.g., Fujitsu A64FX.
- We will test reusable deflation space
- We consider a pipeline version of the solver to speed up the coarse grid correction for extending the scaling region
- DDαAMG using a fabulous solver at the bottom showed a promising result.
- We need to tune with more right-hand sides to confirm the trend.

Thank you!

E. Agullo, L. Giraud, and Y.-F. Jing. "Block GMRES Method with Inexact Breakdowns and Deflated Restarting". In: *SIAM Journal on Matrix Analysis and Applications* 35.4 (2014), pp. 1625–1651. DOI: 10.1137/140961912. eprint: https://doi.org/10.1137/140961912. URL: https://doi.org/10.1137/140961912.

Constantia Alexandrou, Simone Bacchio, and Jacob Finkenrath. "Multigrid approach in shifted linear systems for the non-degenerated twisted mass operator". In: *Comput. Phys. Commun.* 236 (2019), pp. 51–64. DOI: 10.1016/j.cpc.2018.10.013. arXiv: 1805.09584 [hep-lat].

R. Babich et al. "Adaptive multigrid algorithm for the lattice Wilson-Dirac operator". In: *Phys. Rev. Lett.* 105 (2010), p. 201602. DOI: 10.1103/PhysRevLett.105.201602. arXiv: 1005.3043 [hep-lat].

Ronald Babich et al. "The Role of multigrid algorithms for LQCD". In: *PoS* LAT2009 (2009). Ed. by Chuan Liu and Yu Zhu, p. 031. DOI: 10.22323/1.091.0031. arXiv: 0912.2186 [hep-lat].

J. Brannick et al. "Adaptive Multigrid Algorithm for Lattice QCD". In: *Phys. Rev. Lett.* 100 (2008), p. 041601. DOI: 10.1103/PhysRevLett.100.041601. arXiv: 0707.4018 [hep-lat].

James Brannick et al. "Multigrid Preconditioning for the Overlap Operator in Lattice QCD". In: *Numer. Math.* 132.3 (2016), pp. 463–490. DOI: 10.1007/s00211-015-0725-6. arXiv: 1410.7170 [hep-lat].

M.A. Clark et al. "The Removal of critical slowing down". In: *PoS* LATTICE2008 (2008). Ed. by Christopher Aubin et al., p. 035. DOI: 10.22323/1.066.0035. arXiv: 0811.4331 [hep-lat].

Saul D. Cohen et al. "Multigrid Algorithms for Domain-Wall Fermions". In: *PoS* LATTICE2011 (2011). Ed. by Pavlos Vranas, p. 030. DOI: 10.22323/1.139.0030. arXiv: 1205.2933 [hep-lat].

A. Frommer et al. "An adaptive aggregation based domain decomposition multilevel method for the lattice wilson dirac operator: multilevel results". In: (July 2013). arXiv: 1307.6101 [hep-lat].

Martin Luscher. "Deflation acceleration of lattice QCD simulations". In: *JHEP* 12 (2007), p. 011. DOI: 10.1088/1126-6708/2007/12/011. arXiv: 0710.5417 [hep-lat].

Martin Luscher. "Local coherence and deflation of the low quark modes in lattice QCD". In: *JHEP* 07 (2007), p. 081. DOI: 10.1088/1126-6708/2007/07/081. arXiv: 0706.2298 [hep-lat].

Ronald B. Morgan. "Restarted block-GMRES with deflation of eigenvalues". In: *Applied Numerical Mathematics* 54.2 (2005),