Implementation of Simultaneous Inversion of a Multi-shifted Dirac Matrix for Twisted-Mass Fermions within DDαAMG

Shuhei Yamamoto

Simone Bacchio, Jacob Finkenrath

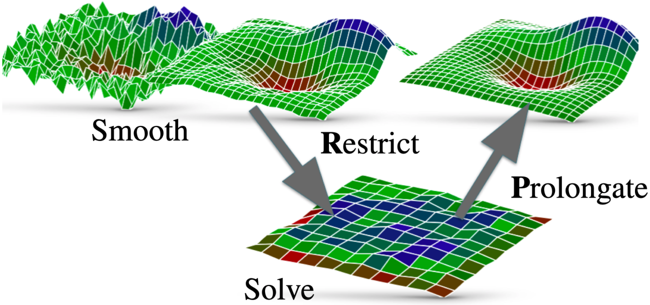
The Cyprus Institute

July 30, 2021

Outline

DDαAMG - Preconditioners

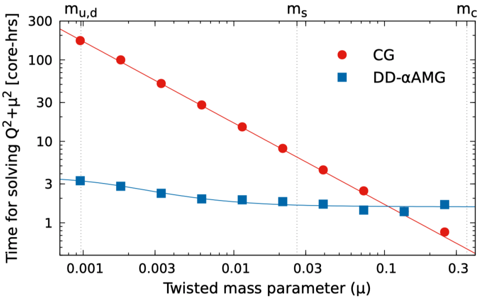
- To circumvent the issue of critical slowing down and effectively invert the large sparse matrix, DDαAMG uses two preconditioners: a smoother and coarse grid correction
- For a smoother, we use red-black Schwarz Alternating Procedure (SAP) (Luscher 2007a).
- For coarse grid correction, we use Algebraic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



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

DDαAMG - Performance

- MG correction accelerates convergence
- 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 - 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

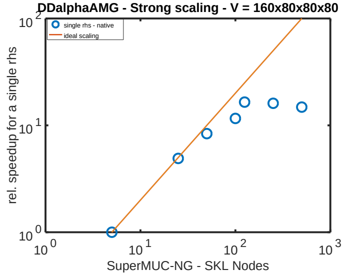


Figure: A scaling plot on the ensemble of $N_f = 2+1+1$ twisted mass clover with $a \sim 0.07$fm and $V = 80^3 x160$ at physical point simulated on SuperMUC-NG (Intel Xeon ("Skylake")) at LRZ

Multiple R.H.S. - Objectives

Originally,

- the code was written for a single rhs
- with multiple rhs, each rhs was inverted one by one
- vectorization of loops was done by chopping a single vector into chunks
- this was done 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.
- Multiple inversion is perfect for rational approximation

Thus,

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

Multiple R.H.S. - Objectives

This allows us to invert

- Dirac matrices for twisted-mass fermions with different μ shifts,

$$D_{\mathrm{TM}}(\mu_i) = D_W + i(\mu + \delta\mu_i)\gamma_5$$

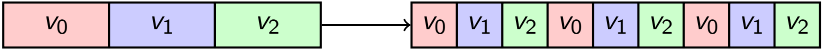
  for different rhs simultaneously.

- degenerate Dirac matrices for twisted-mass fermions for both flavors together

$$\begin{aligned} D_D(\mu) &= (D_{\mathrm{cW}} \otimes I_2) + i\mu(\gamma_5 \otimes \tau_3) \\ &= \begin{pmatrix} D_{\mathrm{TM}}(\mu) & 0 \\ 0 & D_{\mathrm{TM}}(-\mu) \end{pmatrix} \end{aligned}$$

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.
- This reduces data loading time for the matrix.



Multiple R.H.S. - Implementation details

- 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

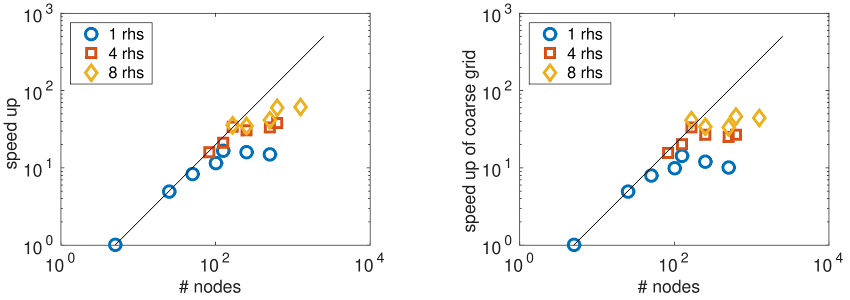
Num. R.H.S.	1 rhs		4 rhs		8 rhs	
Instruction Mix	SP Flops	DP Flops	SP Flops	DP Flops	SP Flops	DP Flops
128-bit	95.26%	86.59%	23.41%	4.99%	24.92%	3.60%
256-bit	2.58%	1.26%	60.68%	78.13%	74.02%	94.76%
Total	97.26%		84.03%		98.81%	

Table: Vectorization Reports

Scaling

Conclusion:

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



Block Solvers

Fast Accurate Block Linear krylOv Solver (Fabulous):

- 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, deflated restarting, and incremental QR factorization.
- It provides several different orthogonalization schemes.

Its usage in DDαAMG:

- 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
- Our implementation of multiple r.h.s. stultifies inexact breakdown.

Setup

Fixed parameters:

- Three-level DDαAMG
- The target residual at the top level: 1×10^{-10}
- Top level solver: FGMRES

Tuning parameters:

- Solvers at the middle and bottom levels (BGMRES, BGCR, BGMRES with deflated restarting (DR), BGMRES with incremental QR factorization (QR), BGMRES with DR and QR (DRQR)
- Residuals at the middle and bottom levels
- Orthogonalization scheme (Classical Gram-Schmidt (CGS), Modified Gram-Schmidt (MGS), Iterative CGS (ICGS), Iterative MGS, each possibly with blocking)
- Size of deflation space at the bottom

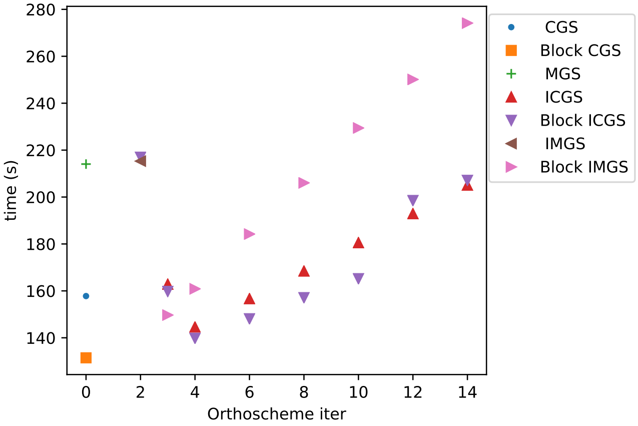
Environment

The systems used for tuning:

- Lattice: $48^3 \times 98$ at physical point
- System: Cyclone (Intel Xeon Gold 6248) at The Cyprus Institute

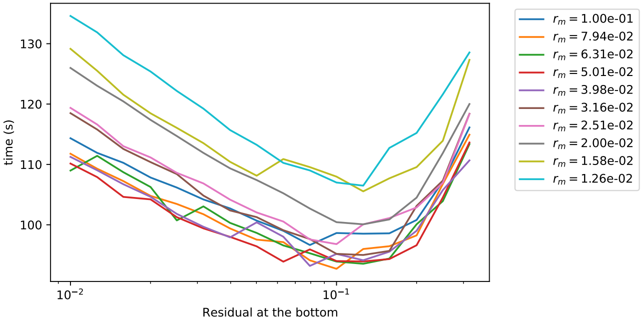
Tuning Orthogonalization Schemes

Solver: BGMRES, Best Orthogonalization Scheme: Block CGS



Tuning Residuals

Middle Solver: FGMRES; Bottom Solver: FGMRES



Tuning Residuals

Middle Solver: BGCR; Bottom Solver: FGMRES

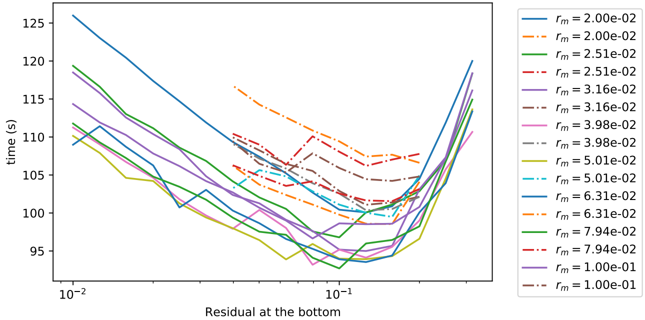


Figure: Comparison of convergence time between AMG with only non-block solvers (solid line) and AMG with mixed solvers (dashed line)

Tuning Residuals

Middle Solver: BGCR; Bottom Solver: BGCR

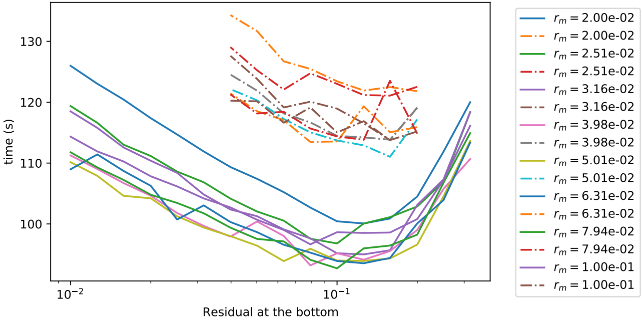


Figure: Comparison of convergence time between AMG with only non-block solvers (solid line) and AMG with mixed solvers (dashed line)

Tuning Residuals

Middle Solver: FGMRES; Bottom Solver: BGCR

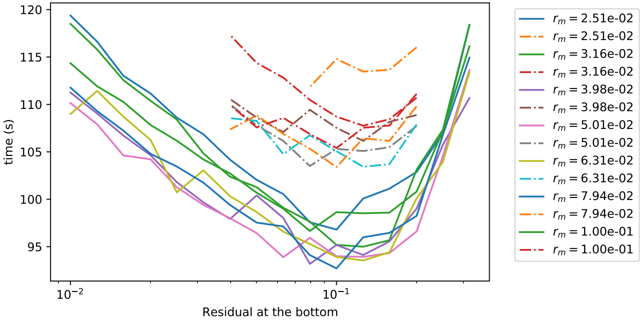


Figure: Comparison of convergence time between AMG with only non-block solvers (solid line) and AMG with mixed solvers (dashed line)

Tuning Residuals

Middle Solver: BGCR; Bottom Solver: FGMRES

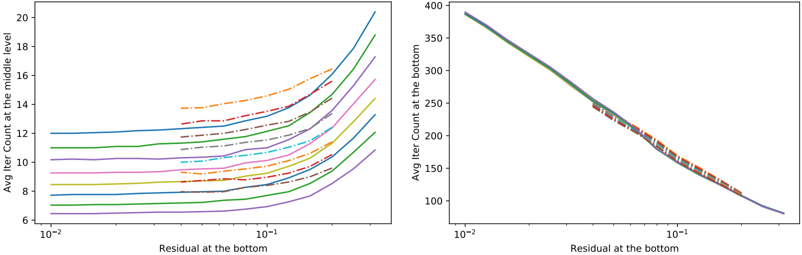
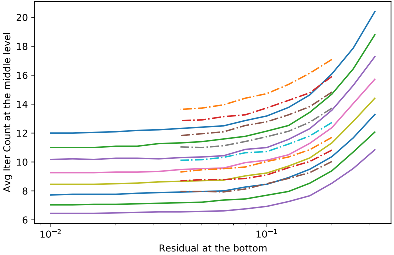


Figure: Comparison of average iteration count between AMG with only non-block solvers (solid line) and AMG with mixed solvers (dashed line)

Tuning Residuals

Middle Solver: BGCR; Bottom Solver: BGCR



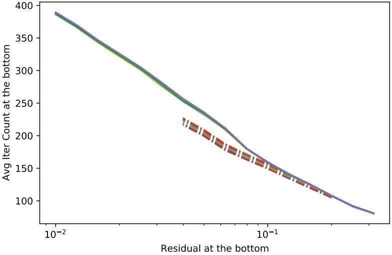


Figure: Comparison of average iteration count between AMG with only non-block solvers (solid line) and AMG with mixed solvers (dashed line)

Tuning Deflation

Middle Solver: BGCR; Bottom Solver: BGCR or BGCRO with deflation

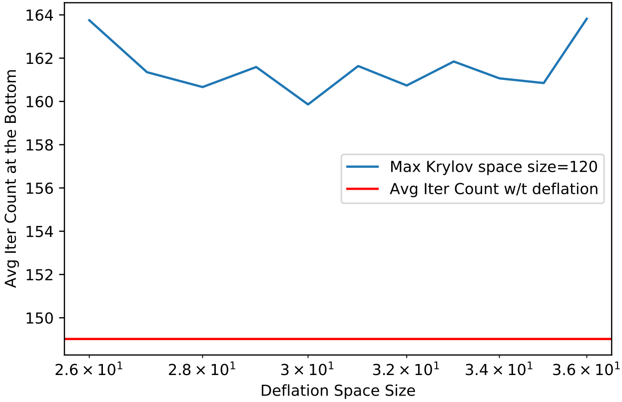


Figure: Comparison of average iteration count at the bottom level with the middle residual 6.31×10^{-2} and bottom residual 0.1 between BGCR with and without deflation

Tuning Deflation

$\delta : 7 \rightarrow 4, 1$ in $D_{TM,bottom} = D_c + i\delta\mu\gamma_5$

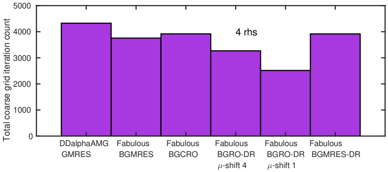


Figure: 4 rhs

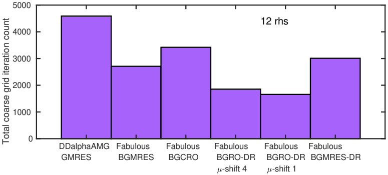


Figure: 12 rhs

Figure: Comparison of total iteration count at the bottom with non-block FGMRES as the middle solver at middle residual, 0.1, bottom residual, 0.1, on the lattice of size $32^3 \times 64$.

Tuning Results

- Inversion by fabulous solvers takes more time to converge
- This is due to overhead of reordering of vectors and MPI_Allreduce calls in the inner product during inversion by fabulous solvers
- As the solver converges quickly at the middle level, block solvers are not effective when used at this level to reduce iteration count
- Block solvers reduce iteration count when used at the bottom
- Deflation in combination with block solvers is helpful in some cases

Outlook

- Scalability is extended by around a factor 5.
- Usage of fabulous in AMG did not reduce overall convergence time due to its overhead
- When used at the bottom, a fabulous solver was effective in reducing iteration count when the bottom residual is smaller than 0.1
- Deflation needs more investigation to find a parameter region where it is effective

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].