Improving HISQ Propagator solves using multigrid

Leon Hostetler[1], Kate Clark[2], Carleton DeTar[3], Steven Gottlieb[1], Evan Weinberg[3]

[1]Indiana University, [2]NVIDIA, [3]The University of Utah

The 42nd International Symposium on Lattice Field Theory
(Lattice 2025)
Nov. 2-8, 2025

Tata Institute of Fundamental Research (TIFR), Mumbai, India

Outline

1. Critical Slowing Down
2. Eigenvector Deflation
3. Staggered Multigrid

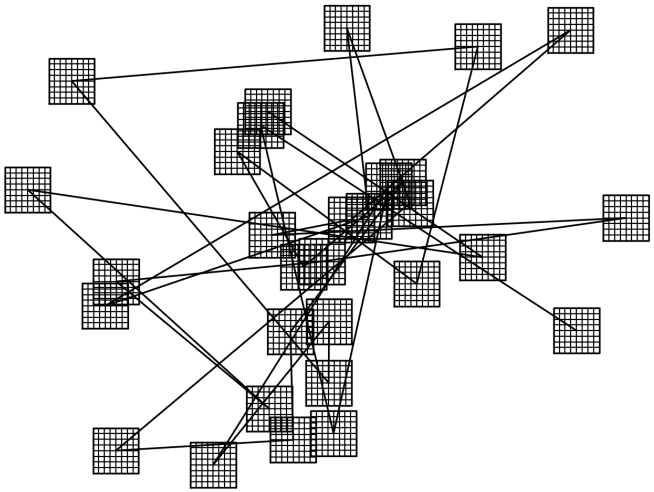
Outline

1. Critical Slowing Down
2. Eigenvector Deflation
3. Staggered Multigrid

Lattice QCD

A typical calculation in lattice QCD involves two steps:

1. Ensemble Generation:



2. Measurement:

$$\overline{O} \approx \frac{1}{N} \sum_{\square} \hat{O} \left(\quad \square \quad \right)$$

Lattice QCD: Computational Resources Used

- A typical calculation (ensemble generation or measurement) requires thousands of runs over a multi-year period
- A typical run may require hundreds of nodes using terabytes of memory and disk space
- Lattice QCD uses $\sim 20\%$ of public U.S. supercomputer time
- MILC's codebase is already highly optimized and used in benchmarks by NERSC and others
- Large speedups will only come from trying new algorithms
- Speedups translate to more science per awarded node-hour!

Our Computational Challenge

1. For both ensemble generation and measurement, the main computational bottleneck is the need to repeatedly invert large sparse "Dirac" matrices
2. **Goal:** Compute $\vec{x} = A^{-1}\vec{b}$ (i.e., a "quark propagator")
3. **Plan:** Solve the linear system $A\vec{x} = \vec{b}$ using conjugate gradient (CG)

$$\begin{array}{c} \\ \sim 10^9 \end{array} \overrightarrow{\qquad} \sim 10^9 \downarrow \begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & c_2 & a_3 & b_3 & \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

Conjugate Gradient

1. Iterative algorithm for solving positive-semidefinite linear systems $A\vec{x} = \vec{b}$
2. Each iteration chooses a new direction based on the residual
3. Convergence rate

$$\frac{||\vec{x}_k - \vec{x}_*||_A}{||\vec{x}_0 - \vec{x}_*||_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k$$

4. The number of iterations needed to reach a given tolerance is

$$\propto \sqrt{\kappa} = \sqrt{\frac{\lambda_{max}}{\lambda_{\min}}}$$

$\kappa =$ condition number, $\lambda =$ eigenvalue

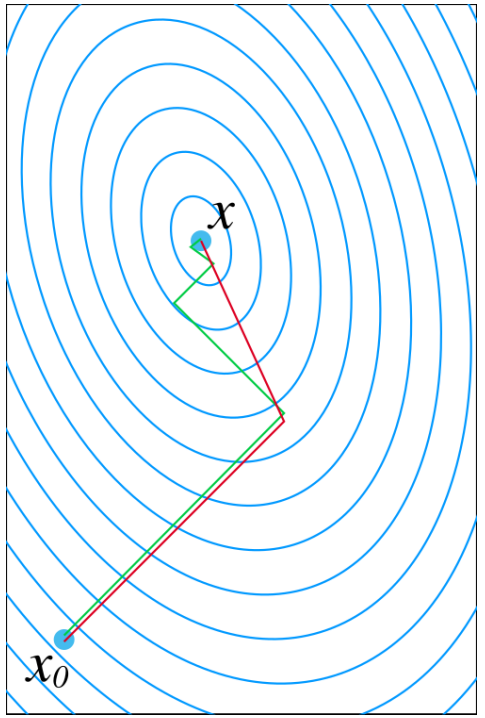
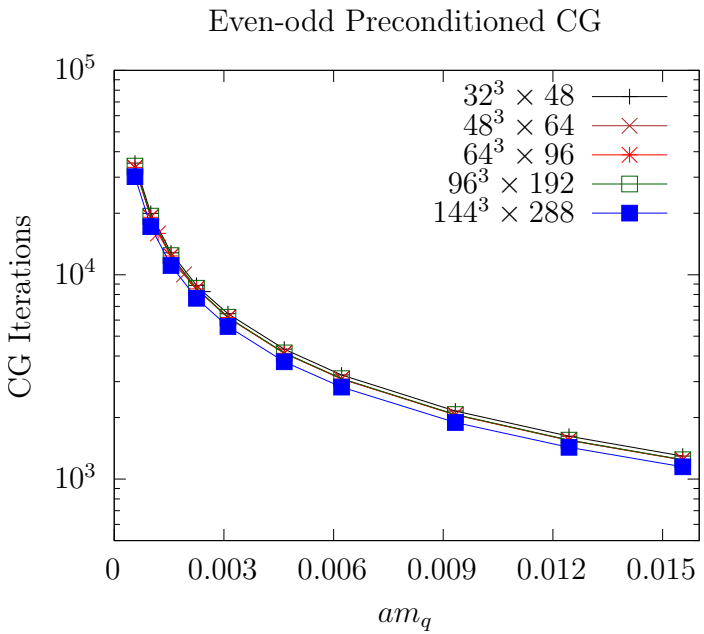
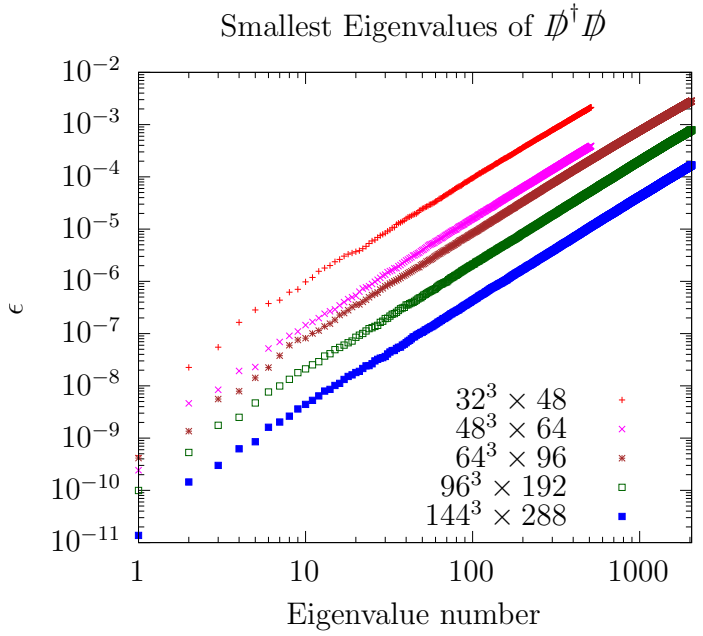


Figure: Public domain, Wikipedia

Problem: Critical slowing down

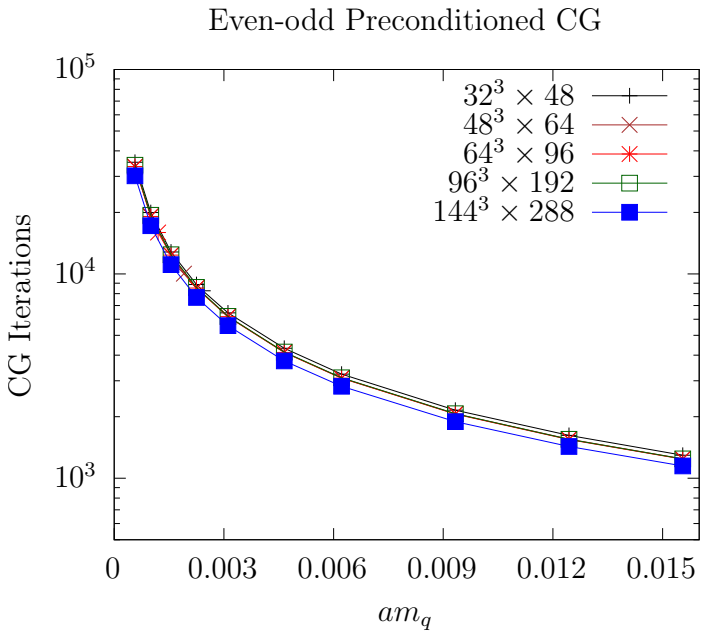


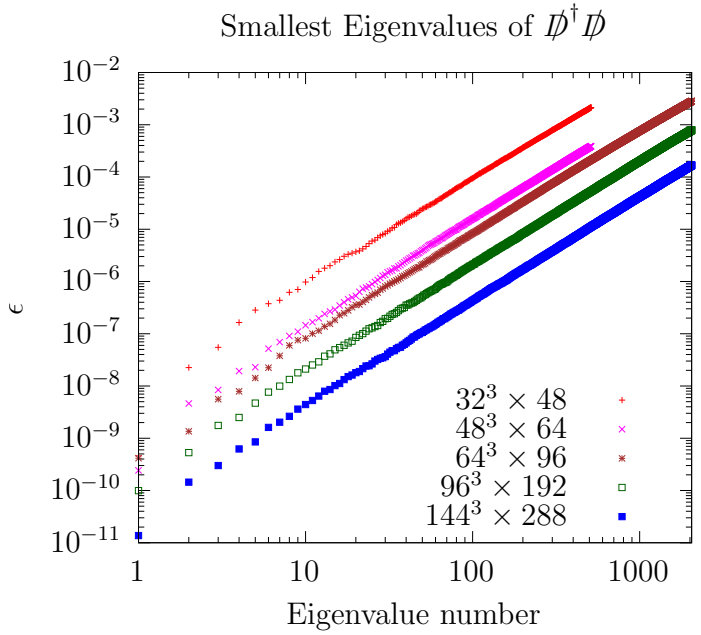


1. The number of CG iterations needed to compute propagators blows up as quark mass is decreased to the physically interesting values
2. CG iterations depends on the condition number of the Dirac matrix

$$\kappa = \frac{\lambda_{max}}{\lambda_{\min}} \approx \frac{23}{\epsilon_{min} + 4m^2} = 10^4 \text{ to } 10^7$$

Problem: Critical slowing down





How can we flatten this curve?

1. **Eigenvector deflation:** Explicitly remove the low modes, solving them exactly and leaving a better-conditioned system with a compressed eigenvalue spectrum
2. **Multigrid:** Solve the low modes on coarser grids (where they are no longer "small")

Outline

1. Critical Slowing Down
2. Eigenvector Deflation
3. Staggered Multigrid

Eigenvector Deflation

1. Deflation:
   - Eigenvectors $|v_i\rangle$ of Dirac operator are used to get an initial guess that has the correct low mode components

$$x = \sum_i |v_i\rangle \frac{1}{\lambda_i} \langle v_i|b\rangle$$

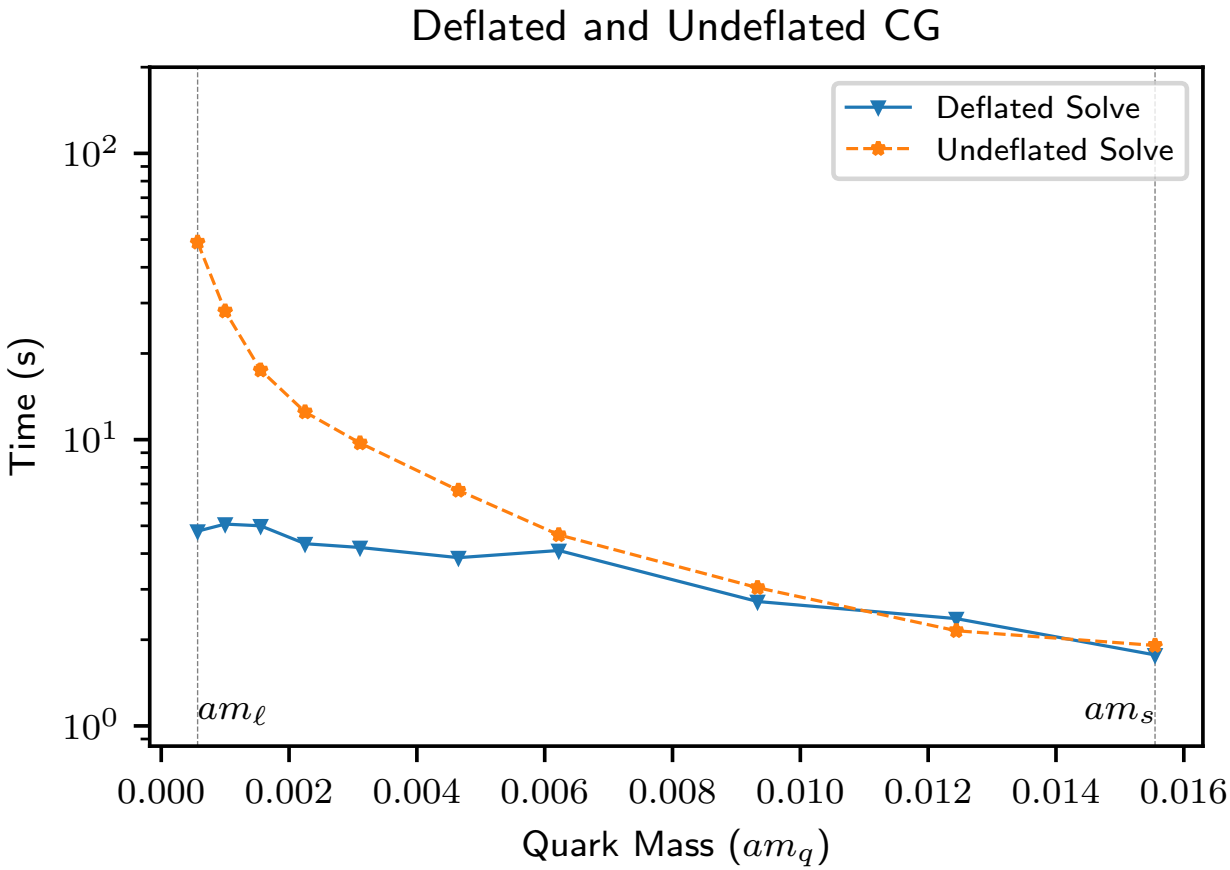
   - Use x as initial guess to CG solver
   - Then CG only has to deal with the high modes which converge more quickly
2. Critical slowing down is shifted from the CG solve to the eigensolve
   - Useful if we can amortize the eigensolve cost over multiple propagator solves
3. As V is increased, deflation becomes relatively more costly

Deflated CG

- We experimented with deflation using the MILC Collaboration's lattice ensembles on the Frontier and Perlmutter supercomputers
- ...using the MILC[1] and QUDA[2] software libraries
- Notable results:
  - Periodically restarting the CG allows us to use single precision eigensolve in certain cases[3]–reducing the memory footprint by a factor of 2
  - Block Lanczos speeds up eigensolve by $\sim$2x
  - We see 10x solve-time speedups (over undeflated CG) at the light quark mass

[1] `https://github.com/milc-qcd/milc_qcd`
[2] `https://github.com/lattice/quda`
[3] Hostetler, DeTar, Clark, Gottlieb, and Weinberg, arXiv:2502.00152

Deflated CG: $144^3 \times 288$ on 384 Perlmutter Nodes



- Deflated CG using 2048 eigenvectors and undeflated CG solve times vs. quark mass
- 10× solve time speedup at the light quark mass over undeflated CG
- Setup time (i.e., eigensolve) takes $\sim$ 3 hours

Deflated CG vs. Ordinary CG

Pros:

- $10\times$ solve time speedup

Cons:

- Need thousands of eigenvectors
- Memory usage is tens of terabytes
- Setup time (i.e., eigensolve) requires hours running on hundreds of nodes

Conclusion:

- Useful if hundreds or thousands of solves are needed per Dirac matrix
  - Useful for some measurements
  - Not useful for ensemble generation
- Useful if we want the eigenvectors anyway

Outline

1. Critical Slowing Down
2. Eigenvector Deflation
3. Staggered Multigrid

Staggered Multigrid

- Multigrid solvers for Wilson fermions have been around for a while
- Staggered fermions proved more challenging
- The key turned out to be a spectral transformation of the fine lattice (Kähler-Dirac preconditioning) that makes it more amenable to multigrid[4] [5]

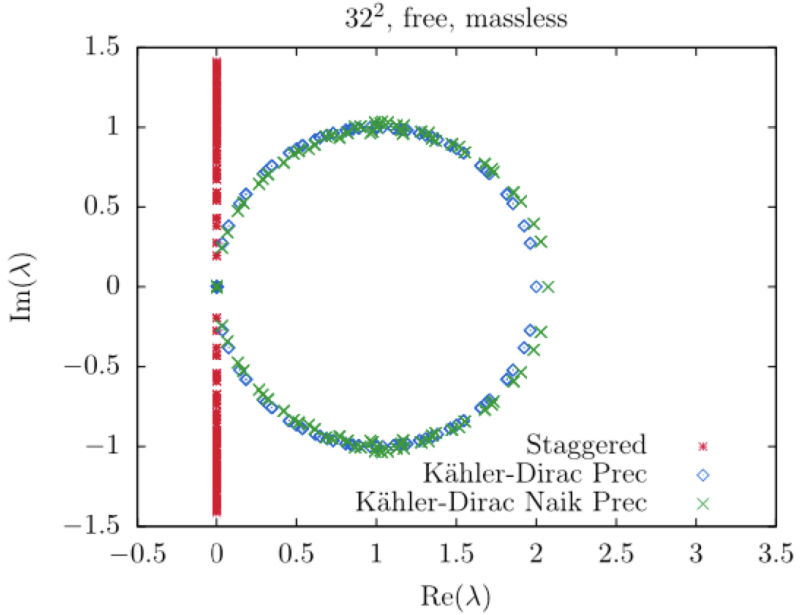


Figure: PRD **97**, 114513 (2018)

- A similar spectral transformation may be useful for using multigrid with 5D domain wall fermions[6]

[4] Brower, Weinberg, Clark, and Strelchenko, Phys. Rev. D. **97**, 114513 (2018)
[5] Ayyar, Brower, Clark, Wagner, and Weinberg, arXiv:2212.12559
[6] Brower, Clark, Weinberg, and Howarth, Phys. Rev. D. **102**, 094517 (2020)

Staggered Multigrid

Setup phase:

1. Level 0: Kähler-Dirac preconditioned operator
2. Level 1 & 2: Compute near-null vectors and coarse Dirac and transfer operators R, P
3. Level 3: Compute eigenvectors

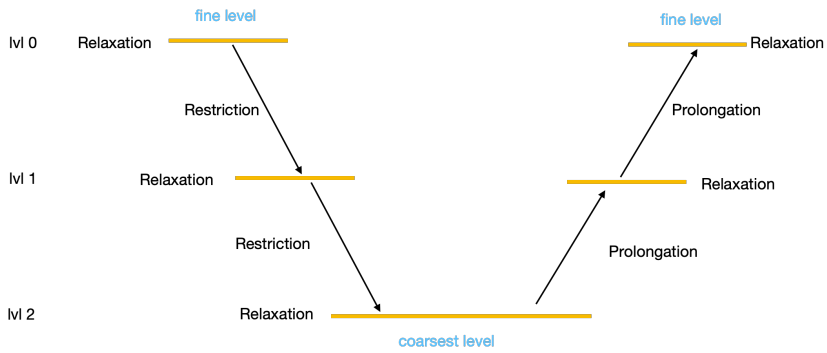
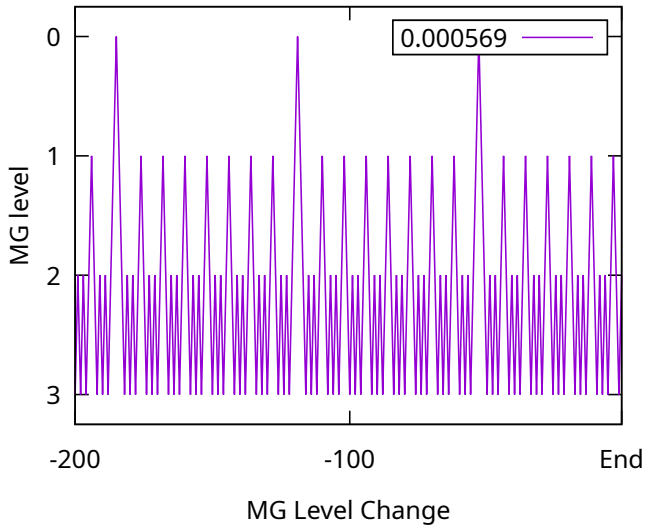


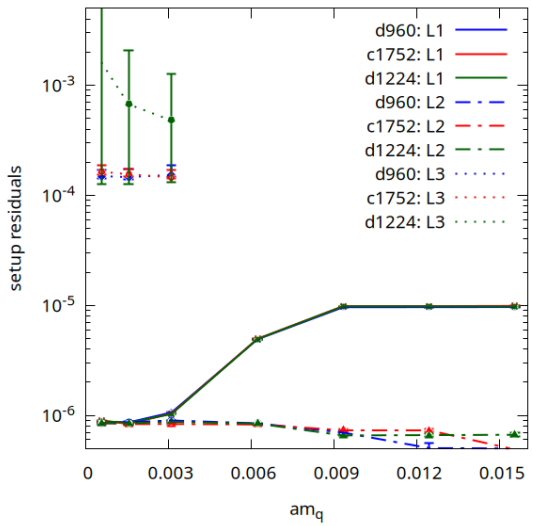
Image from 2212.12559

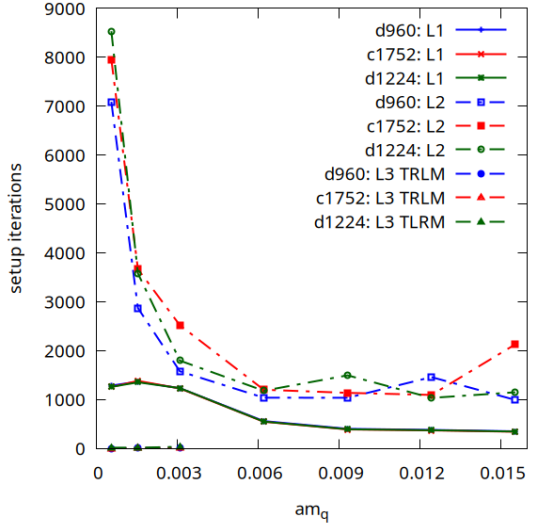
Solve phase:

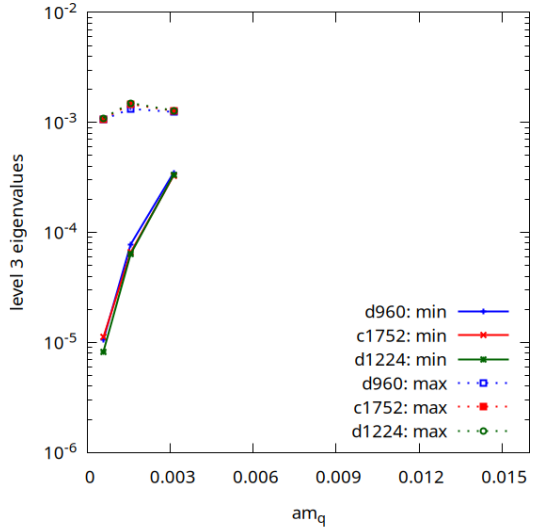
1. For each solver iteration:
   1. Restrict residual to the coarser level
   2. Call the coarser level solver (recurse)
   3. Prolongate error correction and correct solution
   4. Post-smooth the new residual

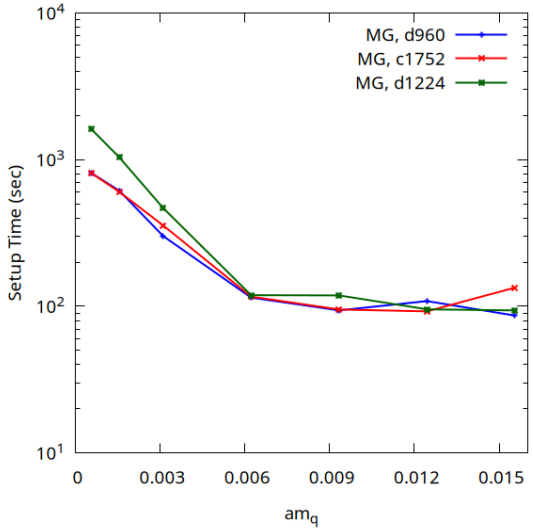


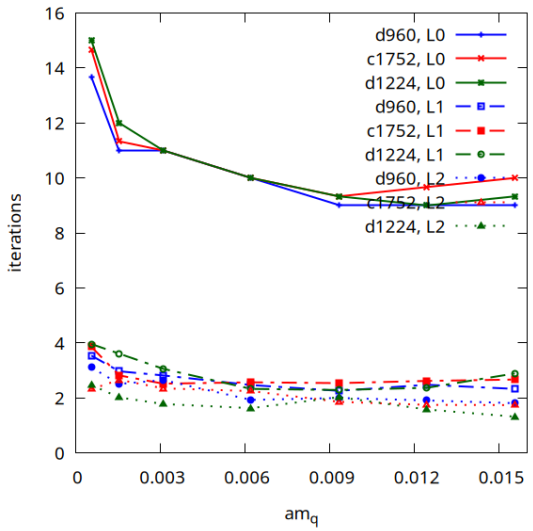
Staggered Multigrid Tuning

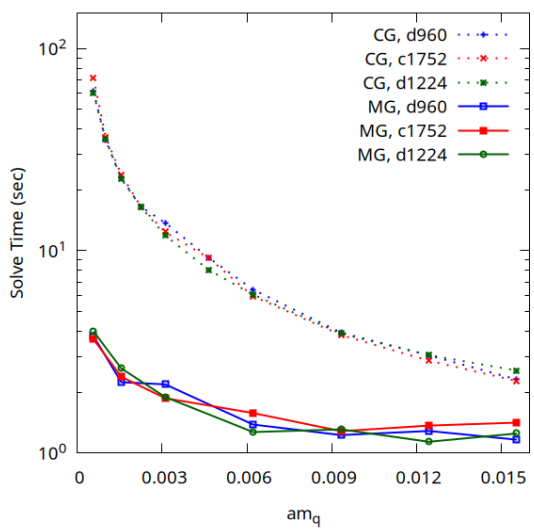










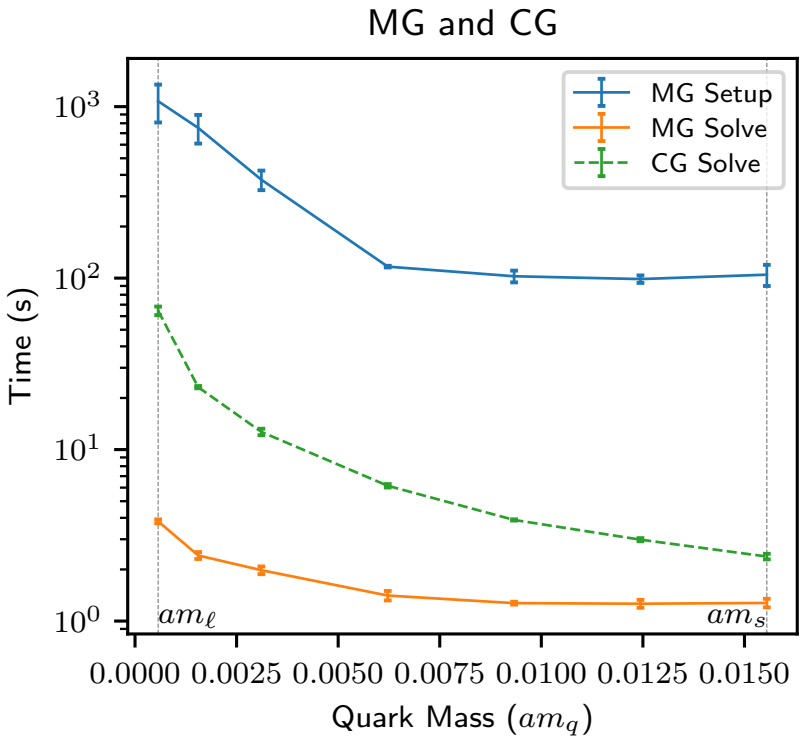


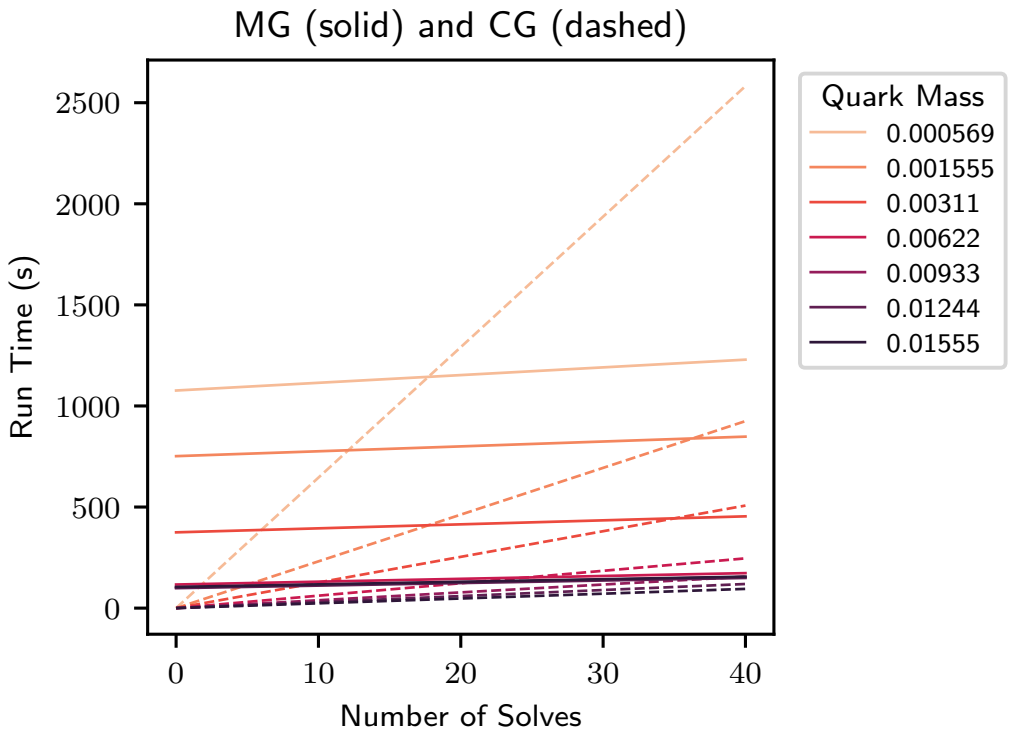
Staggered Multigrid

Figure: NERSC

- We use QUDA to compare multigrid (MG) with conjugate gradient (CG) on MILC's HISQ ensembles
- Tests were performed on Perlmutter with 4 NVIDIA A100s per node
- Results are averaged over 3 configurations

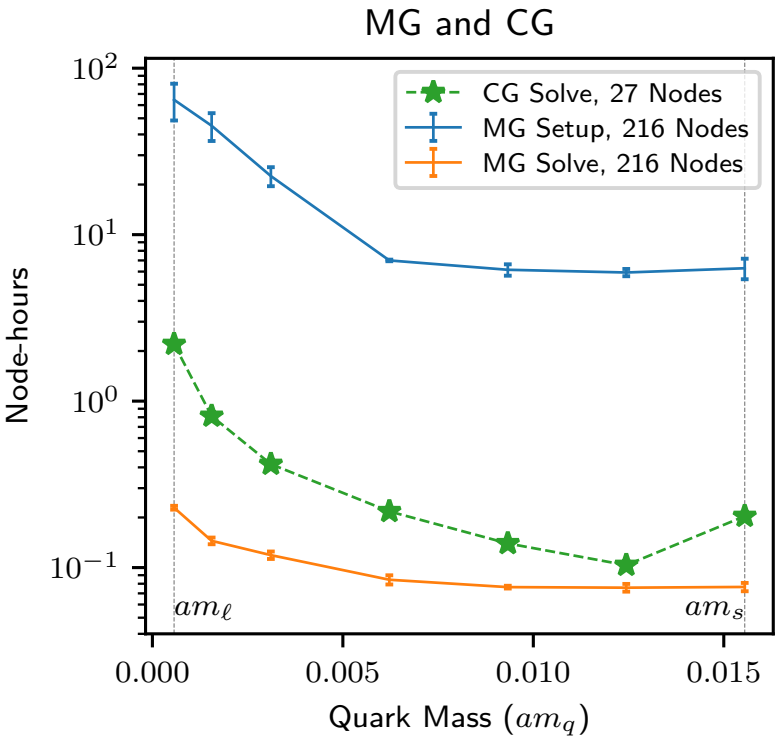
Staggered Multigrid: $144^3 \times 288$ on 216 Perlmutter Nodes

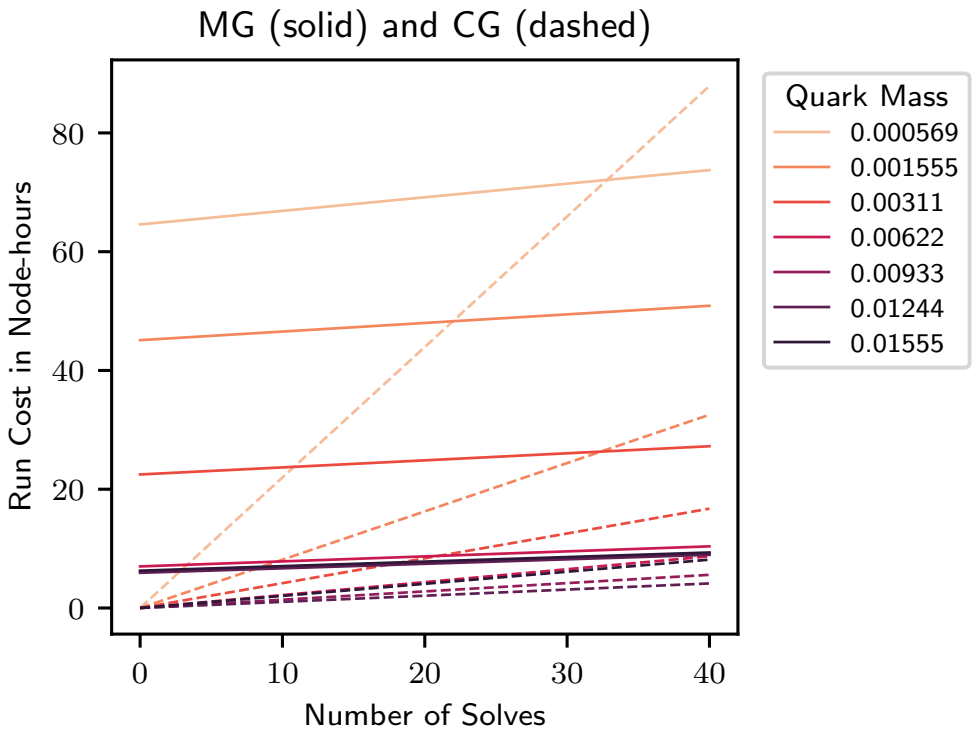




- Left: Multigrid setup and solve times vs. quark mass
- 17× solve time speedup at the light quark mass over CG (green dashed)
- Right: The same data presented in a different way–intersection of solid and dashed lines for a given color (quark mass) shows break-even point

Staggered Multigrid: $144^3 \times 288$ on Perlmutter





- Here, instead of comparing MG and CG on *identical* resources, we do a more fair comparison via node-hours using *minimal* resources
- Here, MG is run on 216 nodes as before
- But, CG, which is relatively lightweight, is run on 27 nodes

Staggered Multigrid vs. Deflation

- Compared to deflated CG:
  - Multigrid has almost 2× faster solve time
  - Multigrid runs on fewer nodes
  - Multigrid has faster setup time
  - Multigrid setup vectors take up less disk space

Staggered Multigrid vs. Deflation

- Compared to deflated CG:
  - Multigrid has almost $2\times$ faster solve time
  - Multigrid runs on fewer nodes
  - Multigrid has faster setup time
  - Multigrid setup vectors take up less disk space
- But...
  - Multigrid requires more parameter tuning (there are far more knobs)
  - In contrast to deflation, multigrid setup vectors are quark mass-dependent, which can add additional tuning steps
  - Eigenvector deflation has variance-reduction applications (low-mode averaging) e.g., disconnected-HVP, reducing the number of required solves for a fixed error budget for some workflows

Summary

- Dirac (large sparse matrix) inversions are the main computational bottleneck in lattice QCD
- Conventional conjugate gradient (CG) suffers critical slowing down due to the condition number blowing up as we approach the physical point
- We need new approaches!
- Our recent testing, primarily done on Perlmutter, shows:
  - Eigenvector deflation $\rightarrow 10\times$ solve time speedups over CG
  - Multigrid $\rightarrow 17\times$ solve time speedups over CG
- These speedups translate directly to more science per node-hour!

Thank you for your attention!

Additional Slides:

Eigenvector Deflation

1. **Eigensolve:** Generate eigenvectors of

$$\not{D}^\dagger \not{D}$$

using thick restarted Lanczos method (TRLM)

2. **Propagator solve:**

$$M^\dagger M \psi = M^\dagger \eta$$

using deflated conjugate gradient for the normal equations (CGNE):

   1. Project eigenvectors $|v_i\rangle$ onto source vector

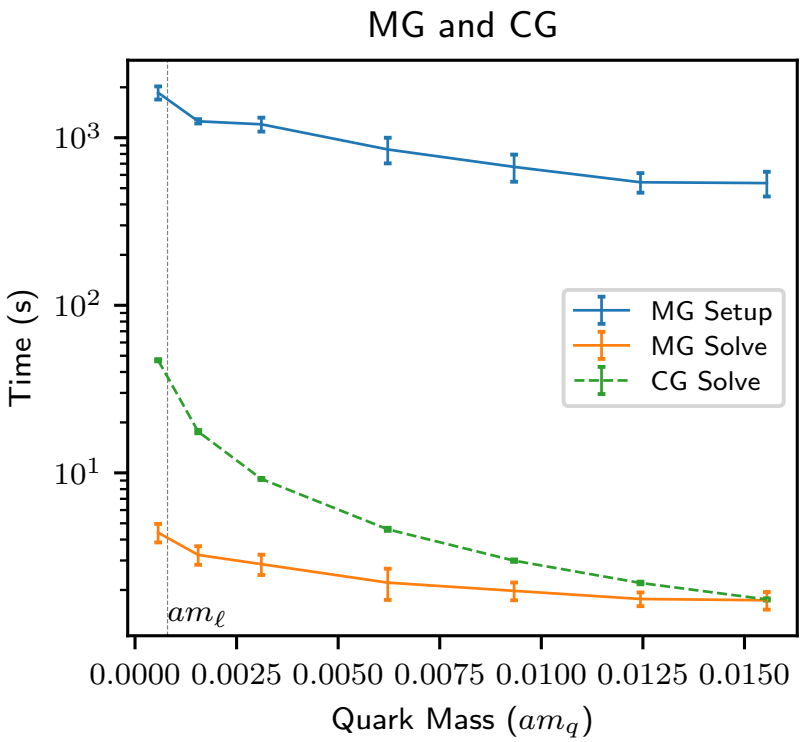
$$x = \sum_i |v_i\rangle \frac{1}{\lambda_i} \langle v_i | \eta \rangle$$

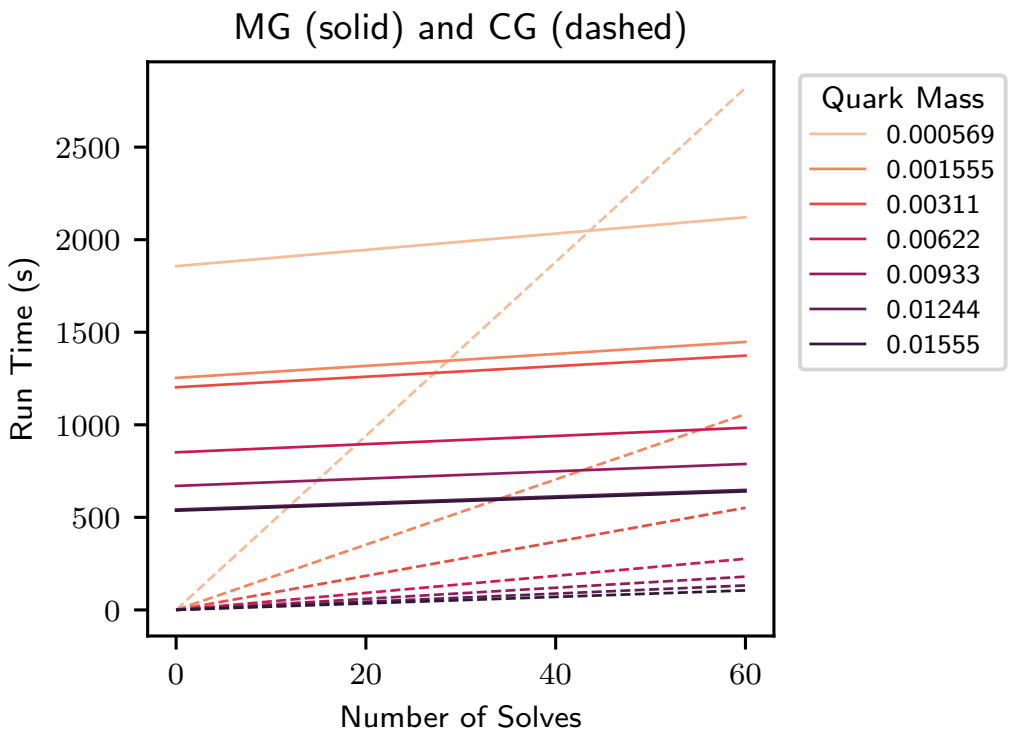
   2. Use x as initial guess to CG solver

Staggered Multigrid Solver Configuration

Level	Solver	Solver Tolerance	Postsmoother
0	MG-precond. GCR	10^{-8}	CA-CGR(8)
1	MG-precond. GCR	0.25 or 8 iters	CA-CGR(8)
2	MG-precond. GCR	0.25 or 8 iters	CA-GCR(8)
3	SVD-deflated CA-GCR	0.25 or 16 iters	N/A

Staggered Multigrid: $96^3 \times 192$ on 48 Perlmutter Nodes





- Left: Multigrid setup and solve times vs. quark mass
- 11× solve time speedup at the lightest mass over CG (green dashed)
- Right: The same data presented in a different way–intersection of solid and dashed lines for a given color (quark mass) shows break-even point