Multigrid Algorithms for Accelerating Fermion Calculations in Lattice QCD

Saul D. Cohen

UNIVERSITY *of* WASHINGTON

Nuclear Theory/INT Seminar
2011 June 20

Outline

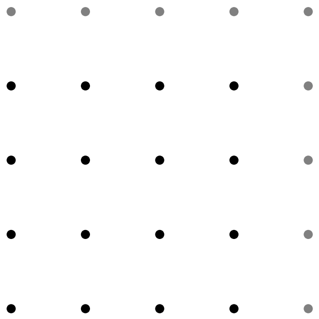
The Path Integral

$$\langle \mathcal{O}[A,\psi,\bar\psi] \rangle = \frac{1}{Z} \int DA\, D\psi\, D\bar\psi\; e^{iS[A,\psi,\bar\psi]} \mathcal{O}[A,\psi,\bar\psi]$$

The Lattice Path Integral

$$\langle \mathcal{O}[A,\psi,\bar\psi]\rangle = \frac{1}{Z}\int DA\,D\psi\,D\bar\psi \left(\prod_i dA(x_i)d\psi(x_i)d\bar\psi(x_i)\right) e^{-S[A,\psi,\bar\psi]}\mathcal{O}[A,\psi,\bar\psi]$$

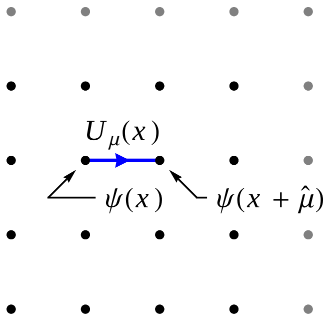
$$x_0 \to -ix_4^{\mathrm{E}} \qquad S \to iS^{\mathrm{E}}$$



The Lattice Path Integral

$$\langle \mathcal{O}[U,\psi,\bar\psi]\rangle = \frac{1}{Z}\int DU\,D\psi\,D\bar\psi\left(\prod_{i,\mu} dU_\mu(x_i)d\psi(x_i)d\bar\psi(x_i)\right) e^{-S[U,\psi,\bar\psi]}\mathcal{O}[U,\psi,\bar\psi]$$

$$U_\mu(x) = e^{iagA^c_\mu(x)\lambda_c}$$

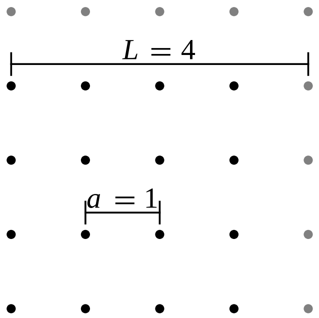


The Lattice Path Integral

$$\langle \mathcal{O}[U,\psi,\bar\psi]\rangle =$$

$$\frac{1}{Z}\int DU\,D\psi\,D\bar\psi\left(\prod_{i,\mu} dU_\mu(x_i)d\psi(x_i)d\bar\psi(x_i)\right) e^{-S[U,\psi,\bar\psi]}\mathcal{O}[U,\psi,\bar\psi]$$

$$1/L \ll m_\pi < \Lambda_{\mathrm{QCD}} \ll 1/a$$

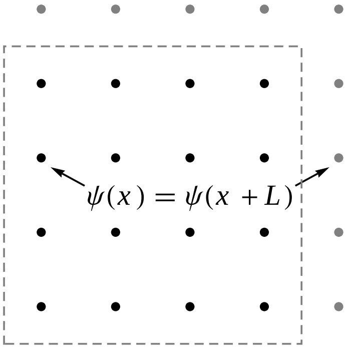


The Lattice Path Integral

$$\langle \mathcal{O}[U,\psi,\bar{\psi}]\rangle =$$

$$\frac{1}{Z}\int DU\,D\psi\,D\bar{\psi}\left(\prod_{i,\mu} dU_\mu(x_i)d\psi(x_i)d\bar{\psi}(x_i)\right) e^{-S[U,\psi,\bar{\psi}]}\mathcal{O}[U,\psi,\bar{\psi}]$$

$$1/L \ll m_\pi < \Lambda_{\mathrm{QCD}} \ll 1/a$$



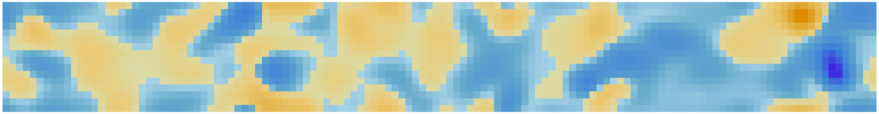
Observables on the Lattice

$$\mathcal{C}(0,t) \xrightarrow{t\to\infty} Ae^{-Mt}$$

$$\xrightarrow{t\to\infty} \langle \overline{u}\gamma_5 d(0)|\pi\rangle\, e^{-M_\pi t}\, \langle \pi|\overline{d}\gamma_5 u(t)\rangle$$

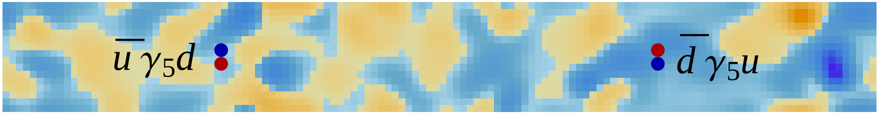
$$= \left\langle \overline{u}\gamma_5 d(0)\ \overline{d}\gamma_5 u(t)\right\rangle$$

$$= \left\langle D^{-1}(0,t)(D^{-1}(0,t))^\dagger \right\rangle$$



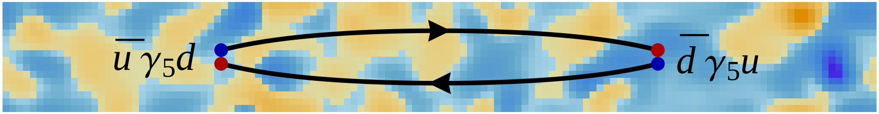
Observables on the Lattice

$$\begin{aligned}
\mathcal{C}_\pi(0,t) &\xrightarrow{t\to\infty} Z_{\pi,\mathrm{src}} Z^*_{\pi,\mathrm{snk}} e^{-M_\pi t} \\
&\xrightarrow{t\to\infty} \langle \overline{u}\gamma_5 d(0)|\pi\rangle\, e^{-M_\pi t}\, \left\langle \pi|\overline{d}\gamma_5 u(t)\right\rangle \\
&= \left\langle \overline{u}\gamma_5 d(0)\ \overline{d}\gamma_5 u(t)\right\rangle \\
&= \left\langle D^{-1}(0,t)(D^{-1}(0,t))^\dagger \right\rangle
\end{aligned}$$



Observables on the Lattice

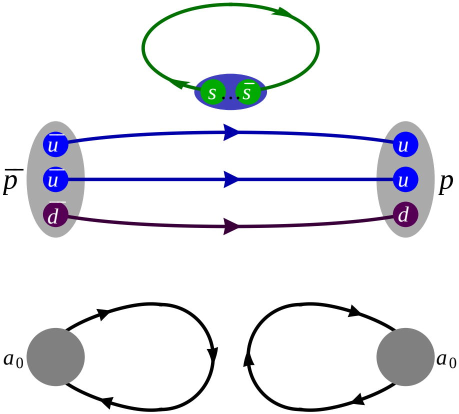
$$\begin{aligned}
\mathcal{C}_\pi(0,t) &\xrightarrow{t\to\infty} Z_{\pi,\mathrm{src}} Z^*_{\pi,\mathrm{snk}} e^{-M_\pi t} \\
&\xrightarrow{t\to\infty} \langle \overline{u}\gamma_5 d(0)|\pi\rangle\, e^{-M_\pi t}\, \langle \pi|\overline{d}\gamma_5 u(t)\rangle \\
&= \left\langle \overline{u}\gamma_5 d(0)\ \overline{d}\gamma_5 u(t)\right\rangle \\
&= \left\langle D^{-1}(0,t)(D^{-1}(0,t))^\dagger \right\rangle
\end{aligned}$$



Observables on the Lattice

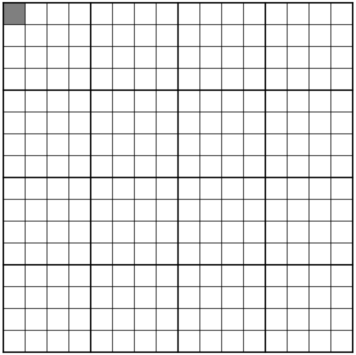
...That Are Exceedingly Difficult

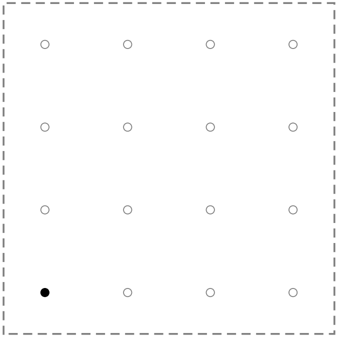
- Disconnected diagrams (strangeness of the proton, dark matter couplings)
- Zero-flavor physics (η', a_0 properties)
- Precision physics (nuclear binding energies, NPLQCD use 100s of sources)



Constructing the Dirac Operator

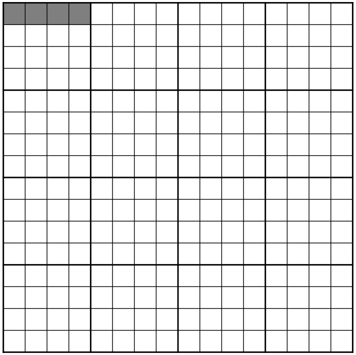
$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu=x,y} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

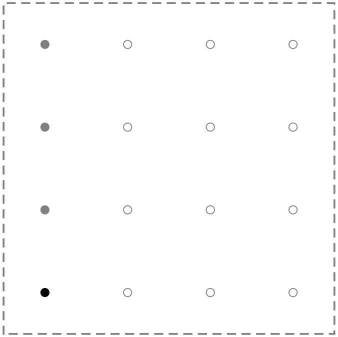




Constructing the Dirac Operator

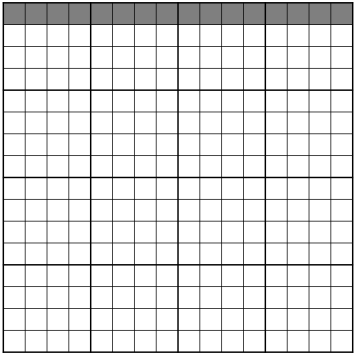
$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu=x,y} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

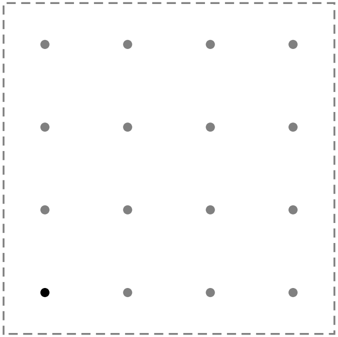




Constructing the Dirac Operator

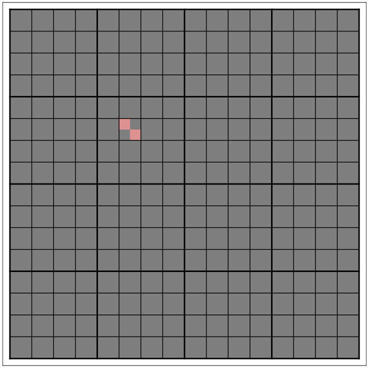
$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu=x,y} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

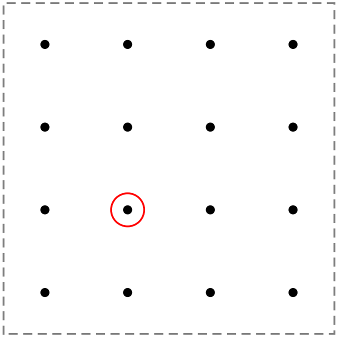




Constructing the Dirac Operator

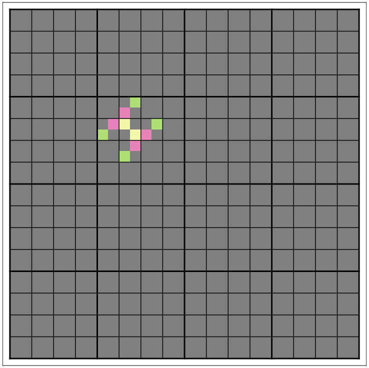
$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu=x,y} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

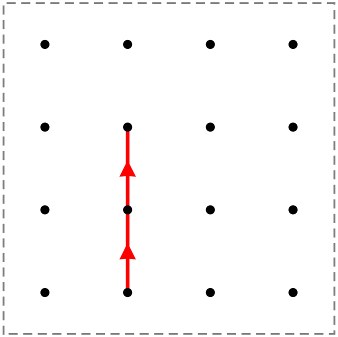




Constructing the Dirac Operator

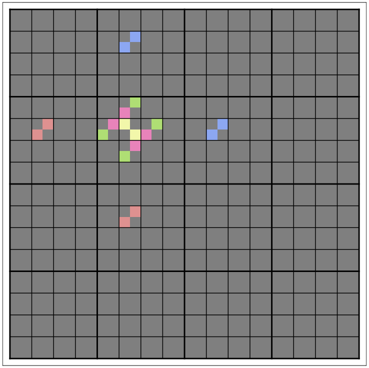
$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu=x,y} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

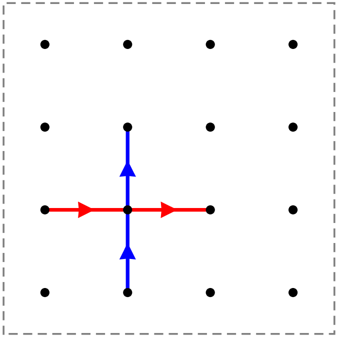




Constructing the Dirac Operator

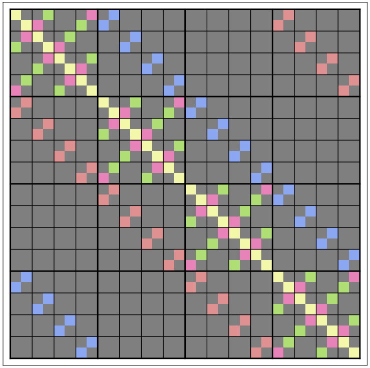
$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu=x,y} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

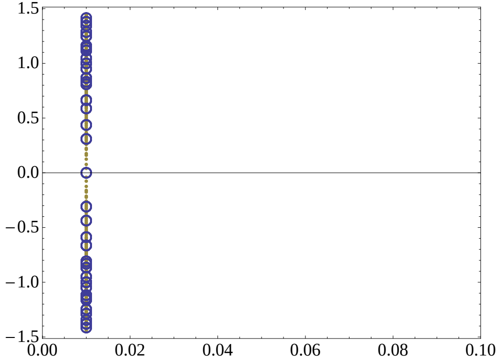




Naive Operator

$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

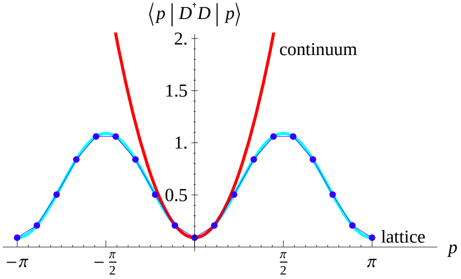




Naive Operator

$$D^{\text{naive}}_{x,x'} = m\,\delta_{x,x'} - \sum_{\mu} \frac{1}{2}\left[\gamma_\mu U_{x,\mu}\delta_{x+\hat{\mu},x'} - \gamma_\mu U^\dagger_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

- Straightforward discretization of continuum operator
- Results in 2^d doubler modes



Wilson Operator

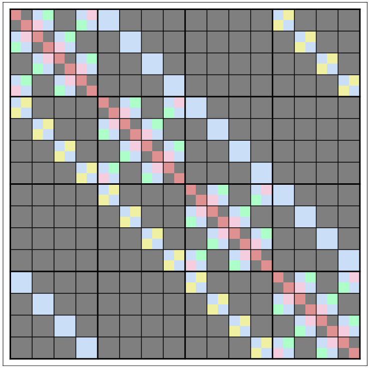
$$D^{\text{Wilson}}_{x,x'} = (m+d)\,\delta_{x,x'} - \sum_{\mu} \frac{1}{2}\left[(1+\gamma_\mu)U_{x,\mu}\delta_{x+\hat{\mu},x'} + (1-\gamma_\mu)U^{\dagger}_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

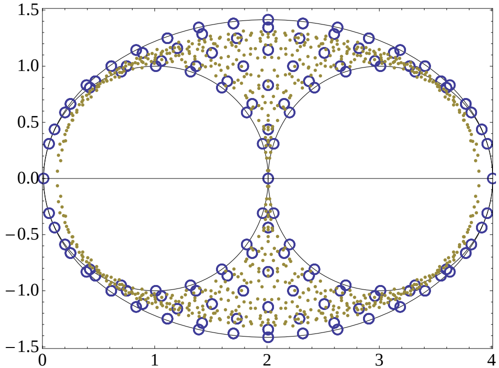
- Suppresses doublers with Wilson term;
  gives them a mass like $1/a$
- Explicitly breaks chiral symmetry;
  large additive renormalizations (e.g. to quark mass)

Wilson Operator

$$D^{\text{Wilson}}_{x,x'} =$$

$$(m+d)\,\delta_{x,x'} - \sum_{\mu} \frac{1}{2}\left[(1+\gamma_\mu)U_{x,\mu}\delta_{x+\hat{\mu},x'} + (1-\gamma_\mu)U^{\dagger}_{x,\mu}\delta_{x,x'+\hat{\mu}}\right]$$

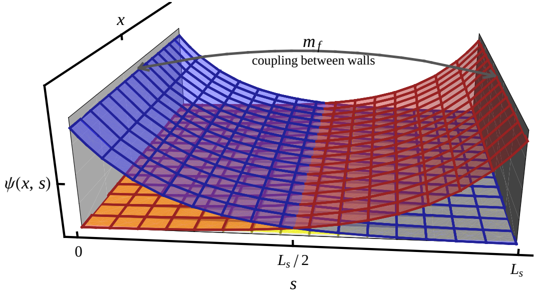




A Chiral Fermion

Domain-Wall Fermions

- Low-lying eigenstates: chiral modes bound to the edges of 5$^{\text{th}}$ dimension
- Very helpful for weak matrix elements (B_K), NPR, operator mixing



Domain-Wall Operator

$$D^{\text{dwf}}_{x,s;x',s'} = \delta_{s,s'} D^{\parallel}_{x,x'} + \delta_{x,x'} D^{\perp}_{s,s'}$$

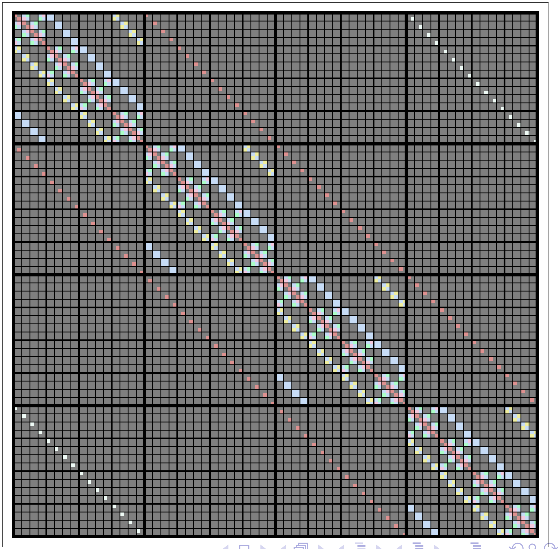
$$D^{\parallel}_{x,x'} = \\ (M_5 - d)\delta_{x,x'} + \frac{1}{2}\sum_{\mu}\left[(1-\gamma_\mu)U_{x,\mu}\delta_{x+\hat{\mu},x'} + (1+\gamma_\mu)U^{\dagger}_{x',\mu}\delta_{x-\hat{\mu},x'}\right]$$

$$D^{\perp}_{s,s'} = \frac{1}{2}\left[(1-\gamma_5)\delta_{s+1,s'} + (1+\gamma_5)\delta_{s-1,s'} - 2\delta_{s,s'}\right] \\ - \frac{m}{2}\left[(1-\gamma_5)\delta_{s,L_s-1}\delta_{0,s'} + (1+\gamma_5)\delta_{s,0}\delta_{L_s-1,s'}\right]$$

Domain-Wall Operator

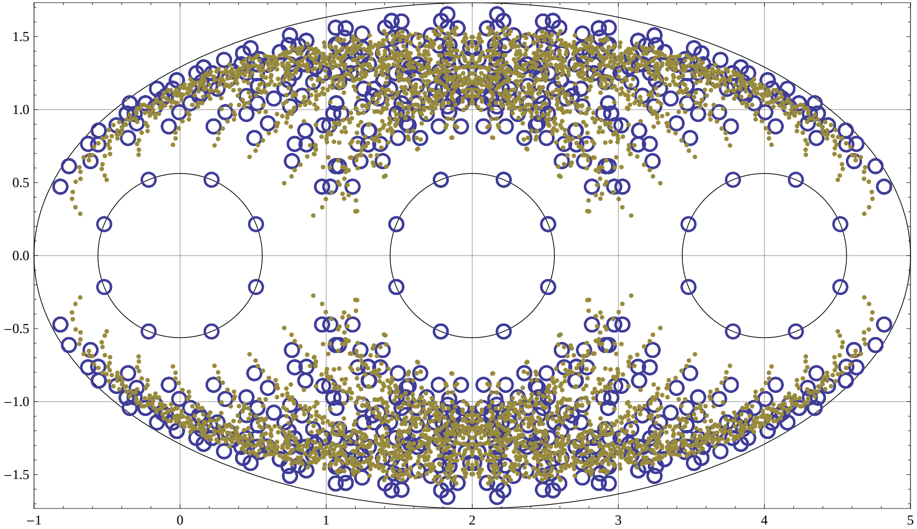
Structure

$$D^{\mathrm{dwf}}_{x,s;x',s'} = \delta_{s,s'} D^{\parallel}_{x,x'} + \delta_{x,x'} D^{\perp}_{s,s'}$$



Domain-Wall Operator

Spectrum



Krylov Subspace Solvers

Krylov solvers spend 50–90% of a lattice calculation's computer power

- Includes the very common conjugate-gradients (CG) algorithm
- Constructs the Krylov subspace $\mathcal{K}_n = \{r, Ar, A^2 r, \ldots, A^n r\}$
- Number of iterations scales like the square-root of the matrix's condition-number: $\sqrt{\kappa}$
- For the Dirac operator, $\kappa \propto 1/m =$ "critical slowing"
- Very efficient at eliminating modes with large eigenvalues

Krylov Subspace Solvers

Krylov solvers spend 50–90% of a lattice calculation's computer power

- Conjugate-Gradients (CG)
- BiConjugate-Gradients Stabilized (BiCGStab)

- Generalized Conjugate-Residuals (GCR)
- Generalized Minimal Residual (GMRes)

- Quasi-Minimal Residual (QMR)
- Transpose-Free Quasi-Minimal Residual (TFQMR)

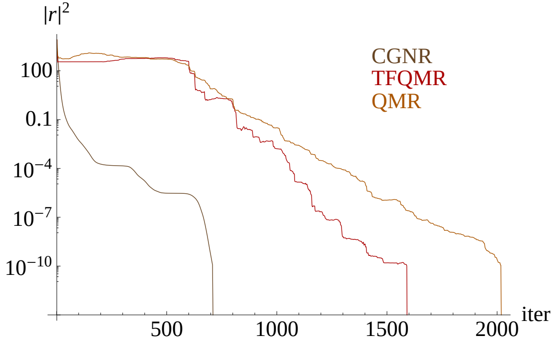
Algorithm Study

Due to the indefinite spectrum of the DWF operator, none of the standard Krylov solvers is guaranteed to work.

Most solvers, including BiCGStab, fail to converge.

TFQMR and QMR work, but very slowly.

CGNR is by far the fastest.

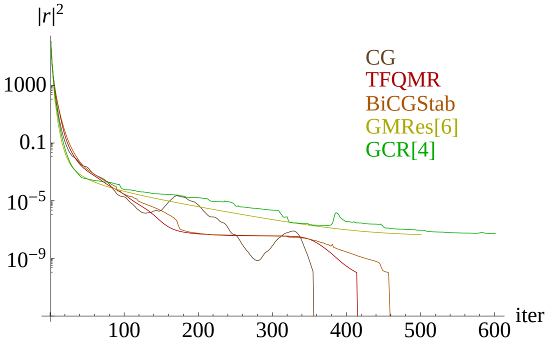


Normal-Equation Solvers

If we accept that we must use the normal equation, most Krylov solvers work.

CG remains fastest.

However, it cannot be preconditioned in a flexible way.

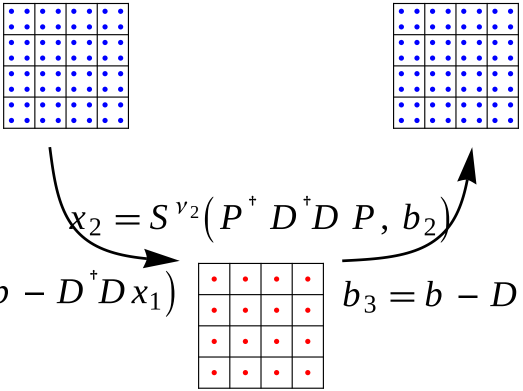


Sketch of Multigrid Algorithm

V-Cycling

$$D\,x = b$$

$$x_1 = S^{\nu_1}\left(D^\dagger D,\ D^\dagger b\right) \qquad x = x_1 + P\,x_2 + S^{\nu_3}\left(D^\dagger D,\ D^\dagger b_3\right.$$



$$x_2 = S^{\nu_2}\left(P^\dagger\ D^\dagger D\ P,\ b_2\right)$$

$$b_2 = P^\dagger\left(D^\dagger b - D^\dagger D\,x_1\right) \qquad b_3 = b - D\,(P\,x_2 + x_1)$$

Sketch of Multigrid Algorithm

Constructing the Coarse Space

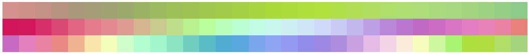
$$V = \left(\begin{array}{cccc|cccc|c} v_1^{(1)} & v_2^{(1)} & v_3^{(1)} & v_4^{(1)} & v_5^{(1)} & v_6^{(1)} & v_7^{(1)} & v_8^{(1)} & \cdots \\ v_1^{(2)} & v_2^{(2)} & v_3^{(2)} & v_4^{(2)} & v_5^{(2)} & v_6^{(2)} & v_7^{(2)} & v_8^{(2)} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right)$$

$$P = \left(\begin{array}{cccc|cccc|c} v_1^{(1)} & v_2^{(1)} & v_3^{(1)} & v_4^{(1)} & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & v_5^{(1)} & v_6^{(1)} & v_7^{(1)} & v_8^{(1)} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ v_1^{(2)} & v_2^{(2)} & v_3^{(2)} & v_4^{(2)} & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & v_5^{(2)} & v_6^{(2)} & v_7^{(2)} & v_8^{(2)} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right)$$

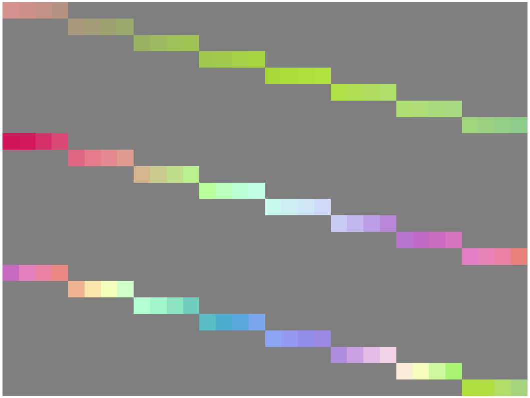
Sketch of Multigrid Algorithm

Constructing the Coarse Space

V:



P:

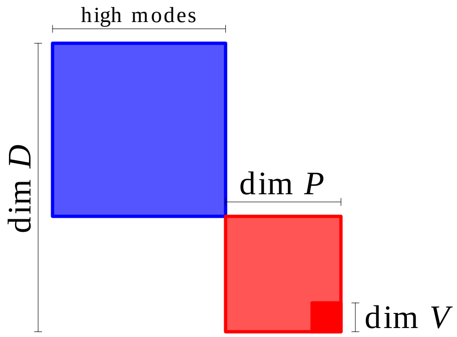


Sketch of Multigrid Algorithm

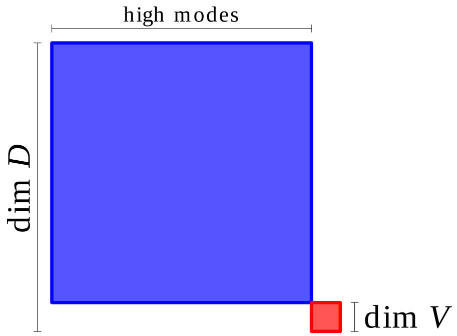
Visualizing the Subspaces

$$r' = \left[1 - AP\frac{1}{P^\dagger AP}P^\dagger\right] r$$

Multigrid

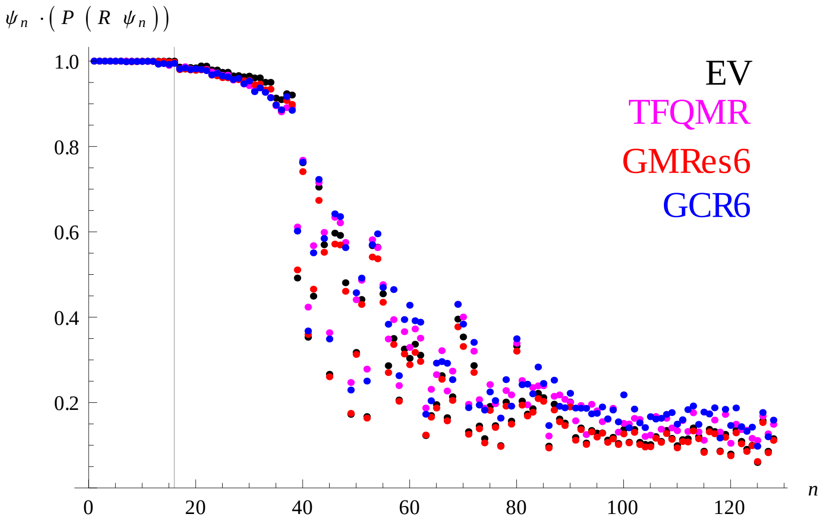


Deflation



Sketch of Multigrid Algorithm

Visualizing the Subspaces



Adaptive Smoothed Algebraic Multigrid

1. Find a set of near-null vectors V for which $\psi^\dagger D^\dagger D\psi \approx 0$ for all $\psi \in V$.
2. Block the vectors to form the prolongator P. Let the unprolongator be $P^\dagger$.
3. Construct the coarse operator $P^\dagger D^\dagger D P$. Use a V-cycle with Krylov smoother and Krylov iteration on coarse operator as a preconditioner to an outer Krylov solver.

The method by which V is constructed does not matter much. Using a solver that will be reused later is convenient, but not necessary.

Adaptive Smoothed Algebraic Multigrid

1. Find a set of near-null vectors V for which $\psi^\dagger D^\dagger D\psi \approx 0$ for all $\psi \in V$.
2. Block the vectors to form the prolongator P. Let the unprolongator be $P^\dagger$.
3. Construct the coarse operator $P^\dagger D^\dagger DP$. Use a V-cycle with Krylov smoother and Krylov iteration on coarse operator as a preconditioner to an outer Krylov solver.

Since $D^\dagger D$ is normal, its left and right eigenvectors are the same. P and R can be constructed from the same set of near-null vectors.

Adaptive Smoothed Algebraic Multigrid

1. Find a set of near-null vectors V for which $\psi^\dagger D^\dagger D\psi \approx 0$ for all $\psi \in V$.
2. Block the vectors to form the prolongator P. Let the unprolongator be $P^\dagger$.
3. Construct the coarse operator $P^\dagger D^\dagger DP$. Use a V-cycle with Krylov smoother and Krylov iteration on coarse operator as a preconditioner to an outer Krylov solver.

The outer solver must be flexibly preconditioned. The selection of the smoother has some effect, but the coarse-level solver can be anything.

Adaptive Smoothed Algebraic Multigrid

1. Find a set of near-null vectors V using the previously described algorithm for which $\psi^\dagger D^\dagger D\psi \approx 0$ for all $\psi \in V$.
2. Block the vectors to form the prolongator P. Let the unprolongator be $P^\dagger$.
3. Construct the coarse operator $P^\dagger D^\dagger DP$. Use a V-cycle with Krylov smoother and Krylov iteration on coarse operator as a preconditioner to an outer Krylov solver.

How is this "adaptive"? We can select new near-null vectors from those that are poorly converged by our current algorithm. Repeat as necessary.

Applied ASAM on DWF Normal Equation

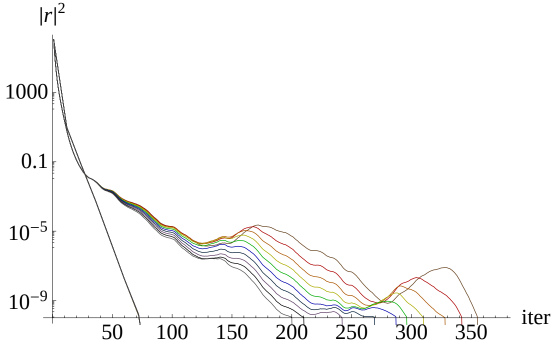
$20^2 \times 8$, U(1), quenched

This algorithm seems to work well on the 2d U(1) DWF normal equation, removing the slowing at small masses.

$20^2 \times 8$ U(1) lattice

$4^2 \times 8$ blocks
$N_v = 16$

Outer solver: CG
Smoother: GMRes(6)
Coarse solver: CG
Coarse r^2: 10^{-3}



Applied ASAM on DWF Normal Equation

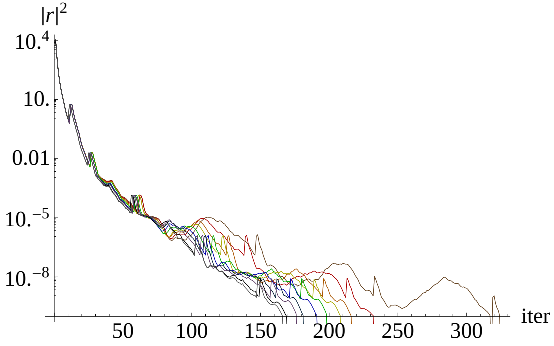
$20^2 \times 8$, U(1), quenched

This algorithm seems to work well on the 2d U(1) DWF normal equation, removing the slowing at small masses.

$20^2 \times 8$ U(1) lattice

$4^2 \times 8$ blocks
$N_v = 16$

Outer solver: CG
Smoother: GMRes(6)
Coarse solver: CG
Coarse r^2: 10^{-3}



Applied ASAM on DWF Normal Equation

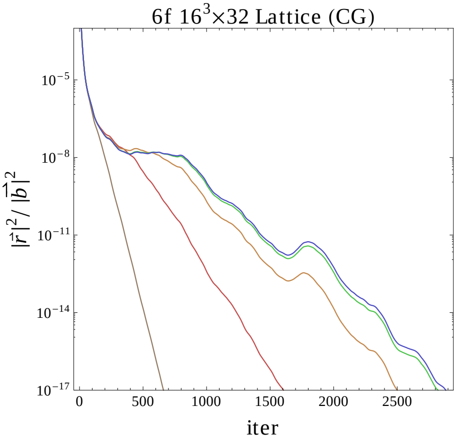
$16^3 \times 32 \times 16$, SU(3), 6-flavor

The effect is equally impressive on a moderately-sized production lattice.

$16^3 \times 32 \times 16$ Technicolor lattice

$4^4 \times 16$ blocks
$N_v = 24$

Outer solver: GCR(12)
Smoother: $12 \times$ GCR(12)
Coarse solver: CG
Coarse r^2: to 10^{-4} then 10^{-2}/cycle



Applied ASAM on DWF Normal Equation

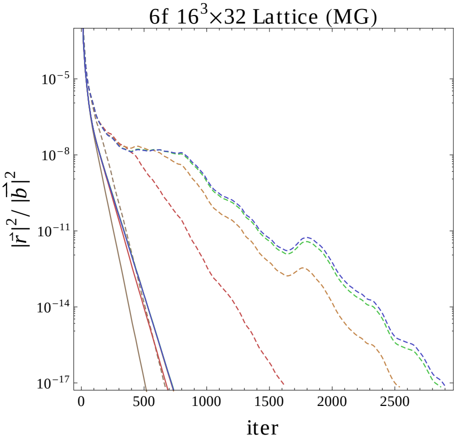
$16^3 \times 32 \times 16$, SU(3), 6-flavor

The effect is equally impressive on a moderately-sized production lattice.

$16^3 \times 32 \times 16$ Technicolor lattice

$4^4 \times 16$ blocks
$N_v = 24$

Outer solver: GCR(12)
Smoother: $12 \times$ GCR(12)
Coarse solver: CG
Coarse r^2: to 10^{-4} then 10^{-2}/cycle



Applied ASAM on DWF Normal Equation

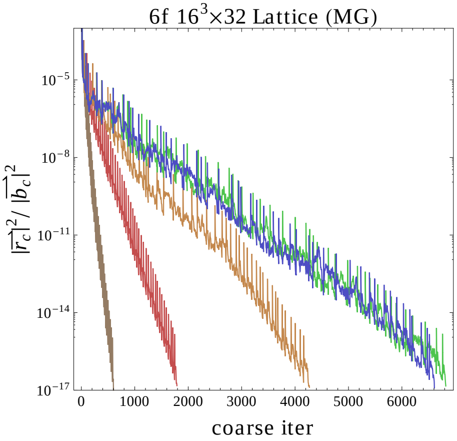
$16^3 \times 32 \times 16$, SU(3), 6-flavor

The effect is equally impressive on a moderately-sized production lattice.

$16^3 \times 32 \times 16$ Technicolor lattice

$4^4 \times 16$ blocks
$N_v = 24$

Outer solver: GCR(12)
Smoother: $12 \times$ GCR(12)
Coarse solver: CG
Coarse r^2: to 10^{-4} then 10^{-2}/cycle



Applied ASAM on DWF Normal Equation

$16^3 \times 32 \times 16$, SU(3), 6-flavor

The effect is equally impressive on a moderately-sized production lattice.

$16^3 \times 32 \times 16$ Technicolor lattice

$4^4 \times 16$ blocks

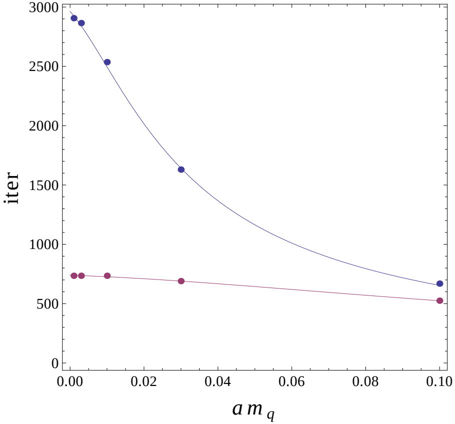
$N_v = 24$

Outer solver: GCR(12)

Smoother: $12 \times$ GCR(12)

Coarse solver: CG

Coarse r^2: to 10^{-4} then 10^{-2}/cycle



Is This Algorithm Good Enough?

- Advantages
  - Blocks away the entire 5th dimension.
  - Factor of 4–8 decrease in fine-operator applications.
  - Removes the critical slowing at small quark masses.
  - Should have excellent scaling with the lattice volume.
- Disadvantages
  - Not a nearest-neighbor operator; poor parallelization.
  - Are there more vectors in P (and R) than we need?

Constructing a 1-Hop Coarse Operator

Can we use $(P^\dagger D^\dagger P)(P^\dagger DP)$ as a coarse operator?

- No, the DWF operator is not normal:
  left-vectors are not right-vectors (conjugated).

Can we use $(P^\dagger \Gamma_5 DP)^2$ as a coarse operator?

- $\Gamma_5 = \gamma_5 \mathcal{R}$, where $\mathcal{R}$ reverses the 5th dimension
- Sounds promising, but empirical tests show no convergence

We know $(P^\dagger D^\dagger 1)(1 DP)$ works; how do we get closer to that?

Constructing a 1-Hop Coarse Operator

Can we use $(P^\dagger D^\dagger P)(P^\dagger DP)$ as a coarse operator?

- No, the DWF operator is not normal:
  left-vectors are not right-vectors (conjugated).

Can we use $(P^\dagger \Gamma_5 DP)^2$ as a coarse operator?

- $\Gamma_5 = \gamma_5 \mathcal{R}$, where $\mathcal{R}$ reverses the 5th dimension
- Sounds promising, but empirical tests show no convergence

We know $(P^\dagger D^\dagger 1)(1DP)$ works; how do we get closer to that?

Constructing a 1-Hop Coarse Operator

Can we use $(P^\dagger D^\dagger P)(P^\dagger DP)$ as a coarse operator?

- No, the DWF operator is not normal:
  left-vectors are not right-vectors (conjugated).

Can we use $(P^\dagger \Gamma_5 DP)^2$ as a coarse operator?

- $\Gamma_5 = \gamma_5 \mathcal{R}$, where $\mathcal{R}$ reverses the 5th dimension
- Sounds promising, but empirical tests show no convergence

We know $(P^\dagger D^\dagger 1)(1DP)$ works; how do we get closer to that?

Constructing a 1-Hop Coarse Operator

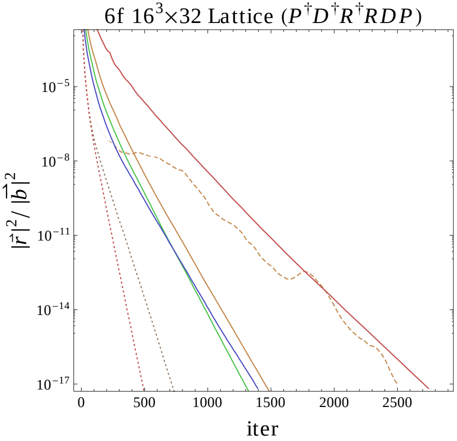
Try $(P^\dagger D^\dagger R^\dagger)(RDP)$ where $\dim R > \dim P$.
In the limit where $\dim R \to \dim D$, this is the known-good algorithm.

$16^3 \times 32 \times 16$ Technicolor lattice

$4^4 \times 16$ blocks
$N_v(R) = 48$
$N_v(P) = 12\text{–}48$

$m_q = 0.01$

Requires fast implementation of rectangular-matrix linear algebra



Summary

- Conclusions
  - Multigrid works for the DWF normal equation.
  - Iteration count improved by factor of 4–8.
- Future Work
  - Scaling test on large lattices (Now running on Kraken)
  - Tuning the algorithm parameters
  - Further study of the unsquared coarse operator
  - Apply to our Disco project
- Distant Future Work
  - Port to GPUs
  - Use TFQMR to make multigrid work without normal equations?