zMobius and other recent developments on Domain Wall Fermions

Tom Blum, Taku Izubuchi, Chulwoo Jung, Christoph Lehner, Sergey Syritsyn

July 14, 2015
Lattice 2015
Kobe International Conference Center, Kobe, Japan

zMobius: Motivation

Mobius fermion(Brower, Neff, Orginos) ($b_5 + c_5 = \alpha, b_5 - c_5 = 1$) has been successfully used for various DWF projects by RBC/UKQCD collaborations, It allows minimal, well bound change from Shamir fermions while allowing for most optimal range of DWF approximation of the step function ($1/L_s \ll (b+c)x \ll L_s$) matches the eigenvalue bounds of $\gamma_5 \frac{D_W}{2+D_W}$. $(b+c)$ between 2 and 4 is typically used.
Recent algorithmic developments(AMA, Greg's talk) made it possible to use cheaper approximation of fermion actions efficiently for both evolution and measurement.

- Smaller L_s allows for more eigenvector generation, more improvement in condition number
- Use in AMA $\rightarrow$ cheaper approximation of the original action
- Eigenvectors with smaller L_s can be used with MADWF(Mawhinney, Yin) to improve on exact solves. The PV inversion necessary for MADWF has been also improved by FFT (Izubuchi,Blum)

Now we are using Mobius as the sea quark action. Can we go beyond Mobius? $\rightarrow$ We can get rational approximations of the Mobius action, which turn out to have complex poles (zMobius).

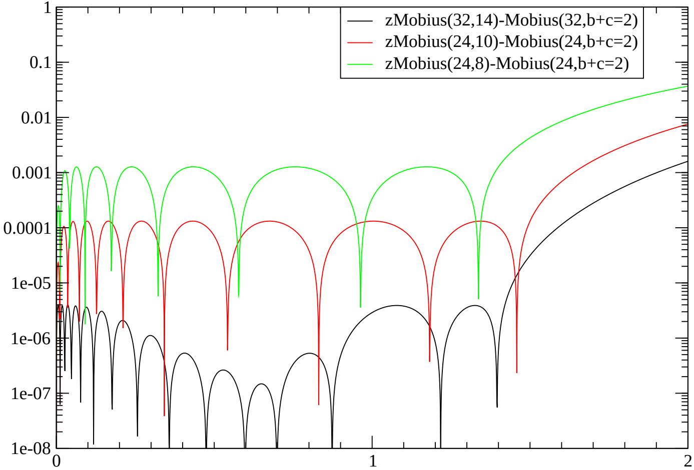
Example: zMobius(L_s=8) for Mobius($L_s = 24, b + c = 2, |x| <1.4$)

$$\epsilon_{zM,8}(x) = \frac{376043x^7 + 329277x^5 + 19977.3x^3 + 95.8779x}{104760x^8 + 507484x^6 + 110975x^4 + 1927.57x^2 + 2}$$
$$= \frac{\prod_{s=0}^{L_s-1}(1+\alpha_s x) - \prod_{s=0}^{L_s-1}(1-\alpha_s x)}{\prod_{s=0}^{L_s-1}(1+\alpha_s x) + \prod_{s=0}^{L_s-1}(1-\alpha_s x)}$$
$$\rightarrow$$

$$b_{0-7} = \{0.87405859026591415, 1.0190035946640621, 1.3658219772391325, 2.0514624685283716, 3.3661546566870935, 5.7994094935966318, 6.7467786395784248 - 3.5543607236727506i, 6.7467786395784248 + 3.5543607236727506i\}$$

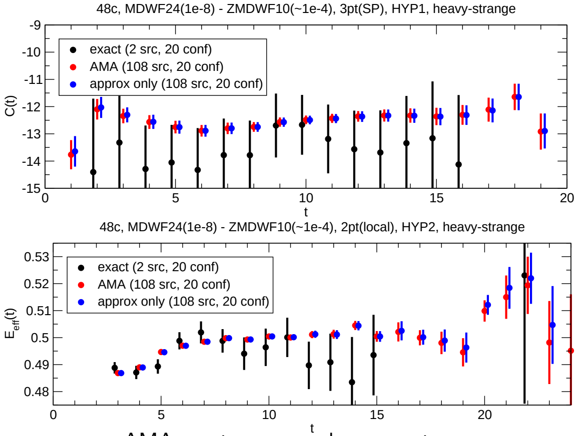
L_s	Max. deviation
6	0.0124435
8	0.00127216
10	1.09868×10^{-4}
12	8.05035×10^{-6}

zMobius


1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
0
1
2
zMobius(32,14)-Mobius(32,b+c=2)
zMobius(24,10)-Mobius(24,b+c=2)
zMobius(24,8)-Mobius(24,b+c=2)

zMobius+AMA example: Static Heavy Quark

T. Ishikawa, 5/16 08:30



AMA on strange quark propagator.
Sloppy: zMobius Ls=10, $N_{iter} = 350$, $|r| = 5 \sim 10 \times 10^{-5}$, no deflation.
Exact: Mobius Ls=24, $N_{iter} =\sim 1000$, $|r| = 10^{-8}$

Preconditioning

In CPS ans QUDA, slightly different conditioning is used. Instead of preconditioning

$$D_{zmob} =$$

$$\begin{pmatrix} b_0 D_W + 1 & (c_0 D_W - 1)P_L & 0 & \cdots & -m_f(c_0 D_W - 1)P_R \\ (c_1 D_W - 1)P_R & b_1 D_W + 1 & (c_1 D_W - 1)P_L & \cdots & 0 \\ 0 & (c_2 D_W - 1)P_R & b_2 D_W + 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ -m_f(c_{L_s-1} D_W - 1)P_L & 0 & \cdots & (c_{L_s-1} D_W - 1)P_R & b_{L_s-1} D_W + 1 \end{pmatrix}$$

directly, CPS multiplies

$$2\kappa_b = diag\left[b_0(4-M)+1, b_1(4-M)+1, \cdots b_{L_s-1}(4-M)+1\right]^{-1}$$

$$2\kappa_b D_{zmob} = \begin{pmatrix} M_5 & -\kappa_b M_{4oe} \\ -\kappa_b M_{4eo} & M_5 \end{pmatrix}$$

$$M_5 = \begin{pmatrix} 1 & \frac{\kappa_{b0}}{\kappa_{c0}} P_L & 0 & \cdots & -m_f \frac{\kappa_{b0}}{\kappa_{c0}} P_R \\ \frac{\kappa_{b1}}{\kappa_{c1}} P_R & 1 & \frac{\kappa_{b1}}{\kappa_{c1}} P_L & \cdots & 0 \\ 0 & \frac{\kappa_{b2}}{\kappa_{c2}} P_R & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ -m_f \frac{\kappa_{bL_s-1}}{\kappa_{cL_s-1}} P_L & 0 & \cdots & \frac{\kappa_{bL_s-1}}{\kappa_{cL_s-1}} P_R & 1 \end{pmatrix}$$

$$M_{4,oe} = \begin{pmatrix} b_0 D_{4,oe} & c_0 D_{4,oe} P_L & 0 & \cdots & -m_f c_0 D_{4,oe} P_R \\ c_1 D_{4,oe} P_R & b_1 D_{4,oe} & c_1 D_{4,oe} P_L & \cdots & 0 \\ 0 & c_2 D_{4,oe} P_R & b_2 D_{4,oe} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ -m_f c_{L_s-1} D_{4,oe} - P_L & 0 & \cdots & c_{L_s-1} D_{4,oe} & \end{pmatrix}$$

Asymmetric and symmetric preconditionings are defined similarly

- Asym$^\kappa$: $M_5 - \kappa_b M_{4,eo} M_5^{-1} \kappa_b M_{4,oe}$
- Sym1$^\kappa$: $1 - M_5^{-1} \kappa_b M_{4,eo} M_5^{-1} \kappa_b M_{4,oe}$
- Sym2$^\kappa$: $1 - \kappa_b M_{4,eo} M_5^{-1} \kappa_b M_{4,oe} M_5^{-1}$

Condition numbers of different preconditioning for zMobius

$24^3 \times 64$, zMobius(L_s =8), m=0.005

Pc	λ_{min}	λ_{max}	λ_{max} / λ_{min}
Asym	$5 \sim 6 \times 10^{-5}$	3.5×10^3	$\sim 7 \times 10^7$
Sym1	$3 \sim 4 \times 10^{-6}$	6.2×10^0	$\sim 2 \times 10^6$
Sym2	$8 \sim 10 \times 10^{-7}$	5.1×10^0	$\sim 6 \times 10^6$
Asym$^\kappa$	2.542×10^{-6}	1.272×10^1	5.00×10^6
Sym1$^\kappa$	2.914×10^{-6}	6.260×10^0	2.15×10^6
Sym2$^\kappa$	4.517×10^{-6}	4.880×10^0	1.08×10^6
Sym1$^\kappa$(MIT)	4.461×10^{-7}	5.086×10^0	1.14×10^7
Sym2$^\kappa$(MIT)	4.208×10^{-7}	2.711×10^1	6.44×10^7

$48^3 \times 96$, zMobius(L_s =10), m=0.00078

Pc	λ_{min}	λ_{max}	λ_{max} / λ_{min}
Asym	1.933×10^{-5}	2.354×10^3	1.218×10^8
Sym1	3.766×10^{-7}	5.946×10^0	1.579×10^7
Sym2	1.988×10^{-7}	5.050×10^0	2.541×10^7
Sym2$^\kappa$	2.086×10^{-7}	5.013×10^0	2.403×10^7

- Condition numbers for symmetric preconditioners (Sym1, Sym2) are in general significantly better than Asymmetric preconditioners, in contrast to scaled Shamir.
- Effect of other details such as Sym1 vs. Sym2, κ_b matrix, and different ordering of b_s, c_s are not clear yet.

Delayed Deflation: Motivation

Algorithmic developments in eigenvector generation, especially Chebyshef-accelerated Lanczos made it possible to generate eigenvectors efficiently, as long as all the vectors can be kept on memory.
$48^3 \times 96$, $m_l = 0.00078$ 2200 zMobius(10) generation: 800K Preconditioned dslash, condition number $2 \times 10^{-7} \to 4 \times 10^{-4}$
1 inversion with Mobius(24,b+c=2) $\sim$ 57K preconditioned dslash.
We can use QUDA on GPU to run the deflated solver efficiently. However, there is a significant mismatch between partitions sizes for evec generation (192 Fermilab pi0, 16 core Ivy Bridge) and GPU solve (12GPU, 3 pi0g nodes)

Mobius QUDA Performance
Fermilab pi0g (K40m), half precision

V	Grid($N_z \times N_t$)	Total	Gflops/GPU
$64^4 \times 12$	2×8	4480	280
$48^3 \times 96 \times 16$	2×8	5170	323
$48^3 \times 96 \times 8$	1×8	2740	343
$48^3 \times 96 \times 16$	1×12	3920	327
$32^3 \times 64 \times 16$	1×4	1690	423

We can save generated 5D eigenvectors efficiently enough to disks on clusters, but reading them before every inversion would be prohibitively expensive. Can we circumvent this $\to$ Delayed deflation.

$R_{\lambda_{max},\lambda_n}(x)$ = Polynomial with pre-calculated coefficients which approximates $1/x$ from λ_{max} to λ_n

$$D^{-1}|\chi\rangle \sim R_{\lambda_{max},\lambda_n}(D)|\chi\rangle + \sum_{\lambda<\lambda_n}\left[1/\lambda - R_{\lambda_{max},\lambda_n}(\lambda)\right]|\lambda\rangle\langle\lambda||\chi\rangle$$
$$= \psi^H_{\lambda_{max},\lambda_n} + \psi^L_{\lambda_{max},\lambda_n} = \psi_{\lambda_{max},\lambda_n}$$

We can do any linear operation on ψ^H and ψ^L separately and combine it later. Examples are:

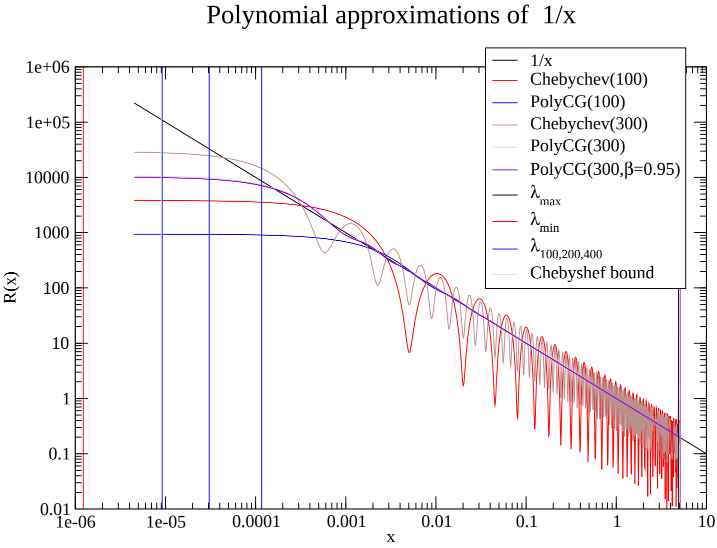
- Self-contraction (disconnected diagrams)
- Contraction with numerically inexpensive propagators (Heavy Quark, etc..)
- 5d$\rightarrow$ 4d and save 4d propagators.

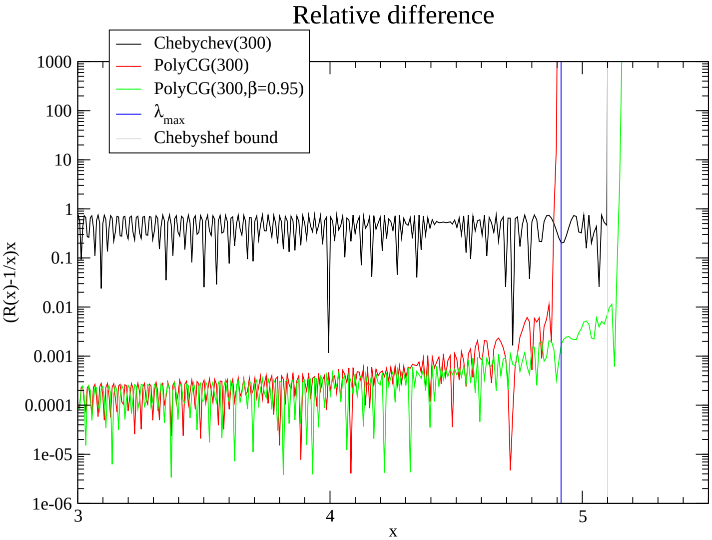
This is also amenable to further optimizations with multiple RHS, and via elimination of global sums.

Delayed Deflation Test

- 32ID (DWF+Iwasaki+DSDR)
  $a^{-1} \sim 1.38$Gev, $m_\pi \sim 170$Mev
  $\lambda_{max} \sim 4.9, \lambda_{min} \sim 1.2 \times 10^{-6}$.
- Choice of polynomials for $R_{\lambda_{max},\lambda_{100}}$
  - Chebyshef: Chebyshef approximation of 1/x over $\sim (\lambda_{100}, \lambda_{max})$
  - PolyCG: Use coefficients from CG on a point source, deflated with 100 eigenvectors, to build R, and use on other sources.
  - Scaled PolyCG: Use same coefficients as PolyCG. $R(x,\beta) = \beta R(\beta x)$, to control very sharp increase of R(x) at λ_{max}
- Observable
  - Residual of $\psi_{\lambda_{max},\lambda_n}$
  - Point source Pseudoscalar meson propagator (1 pt)
  - Point source Nucleon propagator (8 pts) :

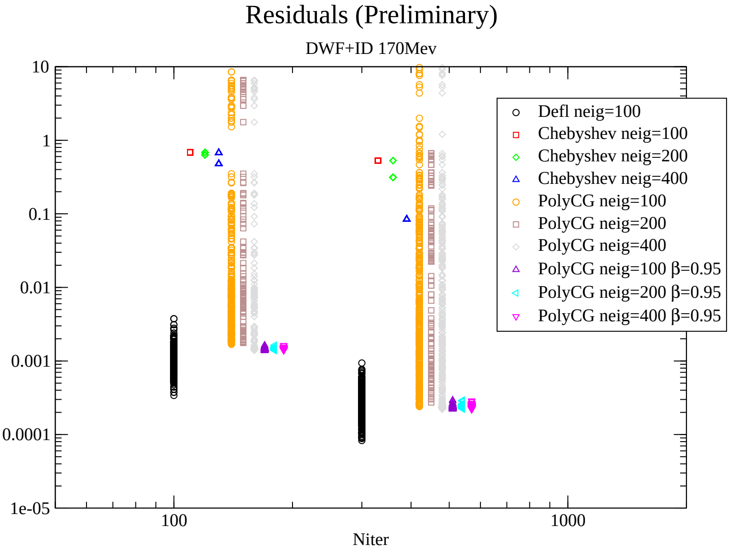
Comparison of approximations



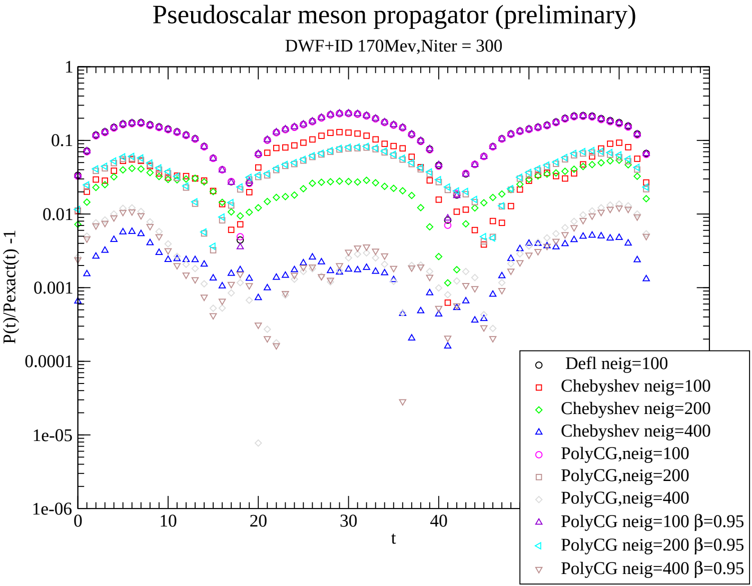


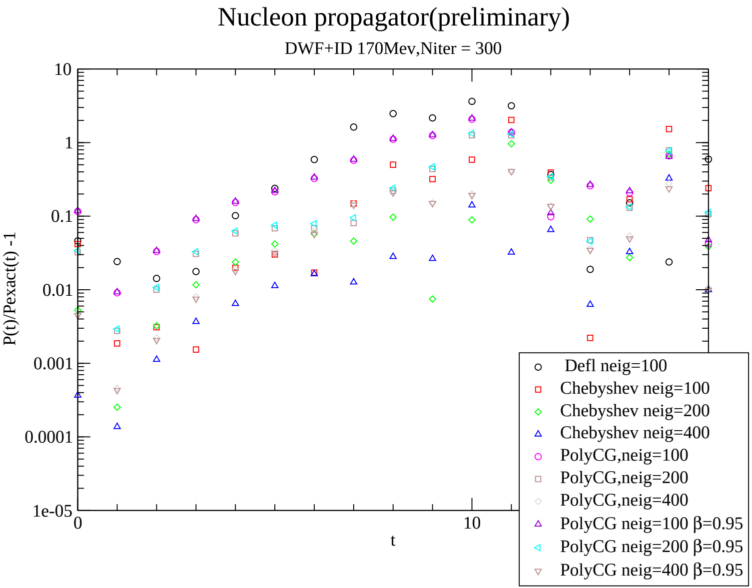
Relative difference
Chebychev(300)
PolyCG(300)
PolyCG(300,β=0.95)
λmax
Chebyshef bound
1000
100
10
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
3
4
5
x
(R(x)-1/x)x

Comparison of residuals for different DD



Comparison of different DD for Niter=300




Nucleon propagator(preliminary)
DWF+ID 170Mev,Niter = 300
P(t)/Pexact(t) -1
t
10
1
0.1
0.01
0.001
0.0001
1e-05
0
10
Defl neig=100
Chebyshev neig=100
Chebyshev neig=200
Chebyshev neig=400
PolyCG,neig=100
PolyCG,neig=200
PolyCG,neig=400
PolyCG neig=100 β=0.95
PolyCG neig=200 β=0.95
PolyCG neig=400 β=0.95

Conclusion/observation

- zMobius provides effective approximation of Mobius DWF with smaller L_s which enables more efficient deflation, measurement, and evolution.
- Symmetric preconditioners for zMobius gives significantly smaller condition number. A more careful comparison between different symmetric preconditioners is under way.
- Delayed deflation with Chebyshef and CG-generated polynomial is studied. Polynomials from CG, with additional deflation and eigenvalue scaling seem able to produce good $R(x)$ for a wide range of observables, although Chebyshef appears competitive in some low-mode dominated quantities. More study is necessary.
- More inspection of CG-generated polynomial can lead to more effective deflation filter.

Thank you!

All Mode Averaging(AMA)

(Blum, Izubichi & Shintani arXiv:1208.4349)
Use the translation invariance of LatticeQCD Lagrangian and replace $\mathcal{O}^{(rest),g}$ with $\mathcal{O}^{(rest)}$ for a set of covariant shifts $\{g\}$.

$$\mathcal{O}^{(imp)} = \frac{1}{N_G}\Sigma_g \left(\mathcal{O}^{(rest),g} + \mathcal{O}^{(approx),g}\right)$$
$$\sim \frac{1}{N_E}\mathcal{O}^{(rest)} + \frac{1}{N_G}\Sigma_g \mathcal{O}^{(approx),g}$$
$$\mathcal{O}^{(rest)} = \mathcal{O}^{(exact)} - \mathcal{O}^{(approx)}$$

This is cost effective when $\mathcal{O}^{(approx)}$ is such that

- $\langle(\Delta(\mathcal{O}^{(rest)})^2\rangle \ll \langle(\Delta(\mathcal{O}^{(approx)})^2\rangle$
- $\mathcal{O}^{(approx)}$ much less expensive than $\mathcal{O}^{(exact)}$
- Covariance: $\mathcal{O}^{(approx),g}[U] = \mathcal{O}^{(approx)}[U^g]$ (However, can be relaxed)