Coarsest-Level Improvements of Multigrid for Lattice QCD on Large-Scale Computers

Gustavo Ramirez
& {Andreas Frommer, Jesus Espinoza, Matthias Rottmann}

Bergische Universität Wuppertal
University of Cyprus
Università degli studi di Ferrara

July 20, 2021

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No' 765048

Main points to be covered

- The problem at hand
- Tackling the coarsest level
- Tuning
- Scaling
- Conclusions

QCD on the lattice

$$\text{forwd. cov. fd.:} \quad \partial^\mu \psi_s(x) = \Big(U_\mu(x)\psi_s(x + ae_\mu) - \psi_s(x) \Big)/a$$

$$\text{backwd. cov. fd.:} \quad \partial_\mu \psi_s(x) = \Big(\psi_s(x) - U_\mu^H(x - ae_\mu)\psi_s(x - ae_\mu) \Big)/a$$

Wilson Discretization:

$$D_W = \underbrace{m_0 I}_{\text{mass shift}} + \frac{1}{2}\sum_{\mu=1}^{4} \Big(\gamma_\mu \otimes \underbrace{(\partial_\mu + \partial^\mu)}_{\text{centralized fd}} - \underbrace{a I_4 \otimes \partial_\mu \partial^\mu}_{\text{stabilization term}} \Big)$$

Clover-improved:

$$D\psi_s(x) = D_W \psi_s(x) - C(x)\psi_s(x),$$

$$\text{with } C(x) = \tfrac{c_{sw}}{32a} \textstyle\sum_{\mu,\nu=0}^{3} (\gamma_\mu \gamma_\nu) \otimes (Q_{\mu\nu}(x) - Q_{\nu\mu}(x))$$

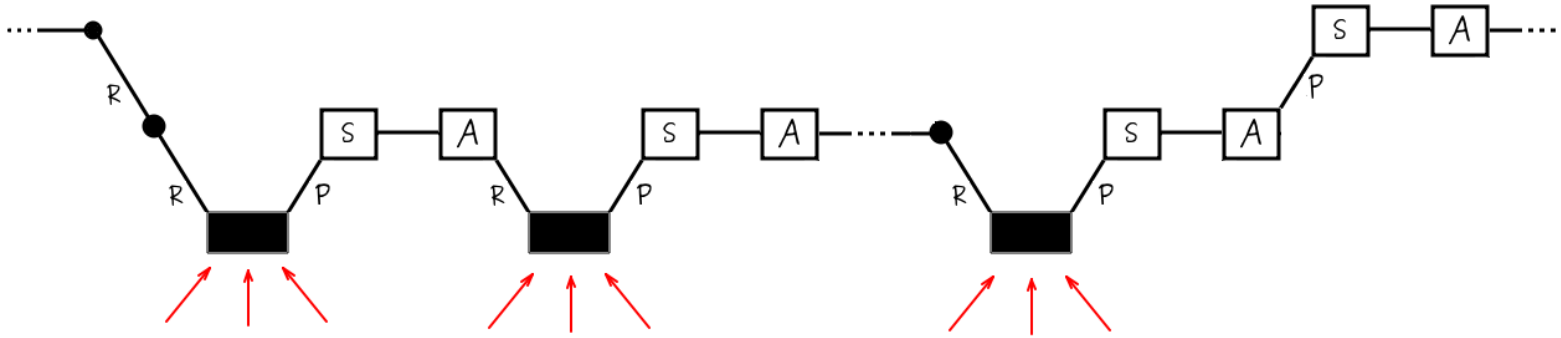
Twisted mass:

$$D_D(\mu) = D \otimes I_2 + i\mu\Gamma_5 \otimes \tau_3$$

The problem at hand : AMG in LQCD

About DD-αAMG:

- the DD-αAMG solver library is an inverter for Wilson-Clover fermions from Lattice QCD.
- $\sim$35k lines of code (before including the improvements discussed in this talk).



Coarsest-level solves up to $\sim 10^{-1}$.

Tackling the coarsest level : block Jacobi preconditioning

Renaming: the matrix of our system of equations i.e. D, will be from hereon called A.

Our system of equations has the form: $Ax = b$. We can split A in diagonal and non-diagonal parts: $A = B + C$, where B is the **block** diagonal part.

In our particular case, A has a particular form:

- $A = A_{ee} - A_{eo}A_{oo}^{-1}A_{oe}$
- Aee and A_{oo}^{-1} are block-diagonal with (small) 6×6 blocks
- we then take: $B = D_{ee} - \mathcal{I}_{eo}D_{oo}^{-1}\mathcal{I}_{oe}$, and B^{-1} as our preconditioner

Our system now takes the form: $B^{-1}Ax = B^{-1}b$.

More on block Jacobi @ Y. Saad. *Iterative Methods for Sparse Linear Systems.*

Tackling the coarsest level : GCRO-DR

Roughly, the GCRO-DR algorithm (our system is $Dx = b$ – matrix not changing):

1. take k approximate eigenvectors of A (initially computed from a first GCRO-DR run on A) $\rightarrow Y_k$
2. compute matrices $U_k, C_k \in \mathbb{C}^{n\times k}$ from Y_k and A, such that : $AU_k = C_k$ and $C_k^H C_k = I_k$ ($QR = AY_k$, $C_k = Q$, $U_k = Y_k R^{-1}$)
3. crucial part : run an "extended" Arnoldi process, at the end of which we have:

$$A[U_k \;\; V_{m-k}] = [C_k \;\; V_{m-k+1}] \begin{bmatrix} I_k & B_k \\ 0 & \bar{H}_{m-k} \end{bmatrix}$$

Any subspace can be recycled for subsequent cycles or linear systems, but we focus here on deflating the smallest eigenvalues (we compute the harmonic Ritz vectors from the extended Arnoldi above).

More on GCRO-DR @ Parks *et al. Recycling Krylov subspaces for sequenes of linear systems.*

Tackling the coarsest level : polynomial preconditioner

As discussed before, we want to solve them system: $B^{-1}Ax = B^{-1}b$.

We add now right preconditioning, with the new preconditioner being a polynomial:

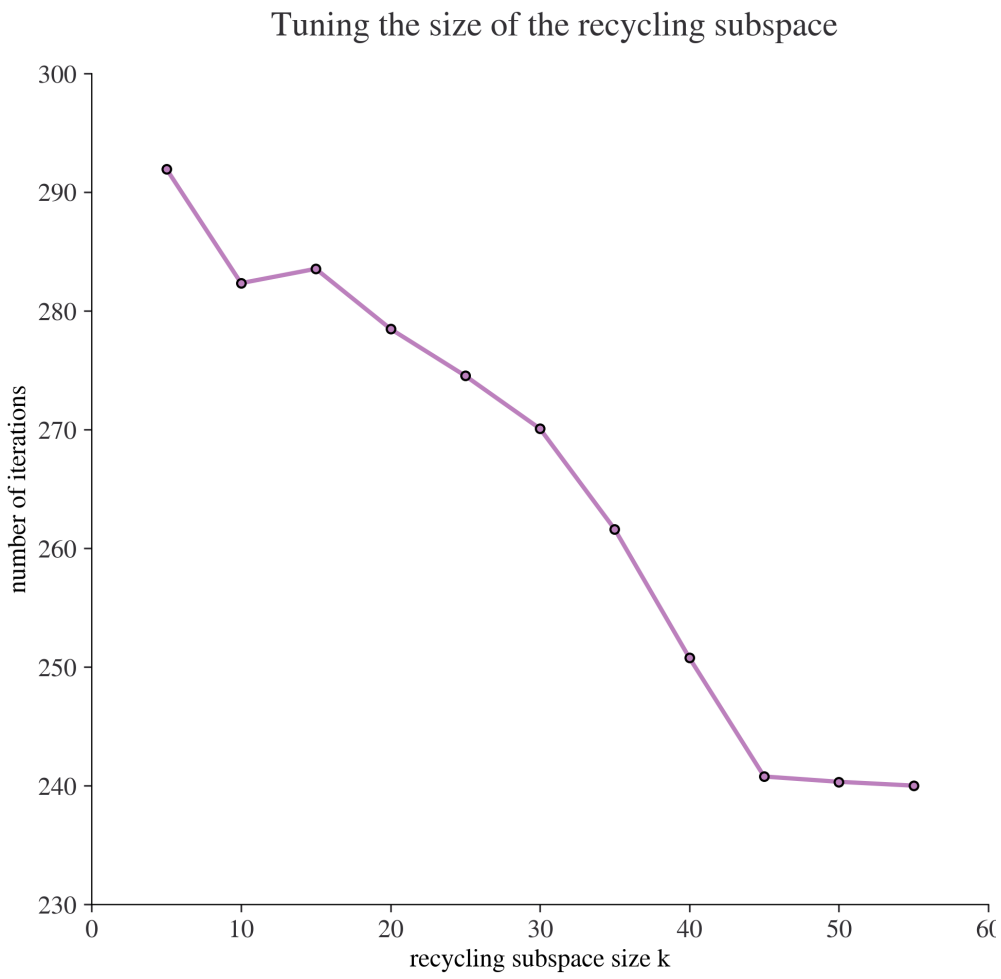
- $B^{-1}Ap(B^{-1}A)y = B^{-1}b$
- $x = p(B^{-1}A)$

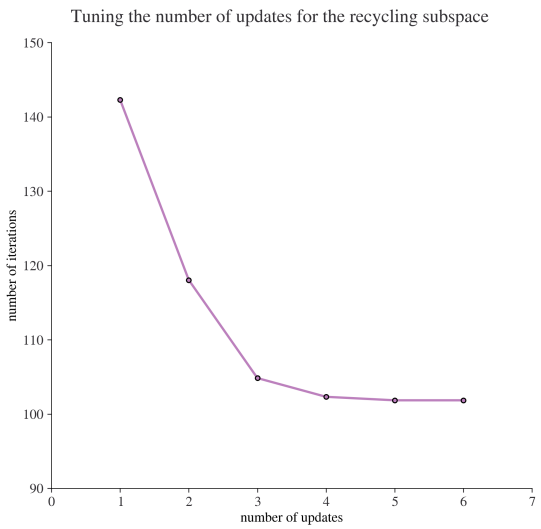
When putting $B^{-1}Ap(B^{-1}A)$ into an Arnoldi process, this allows us to access higher-dimensional subspaces without increasing the calls to orthogonalizations. The con is of course an increasement in the number of matvecs.

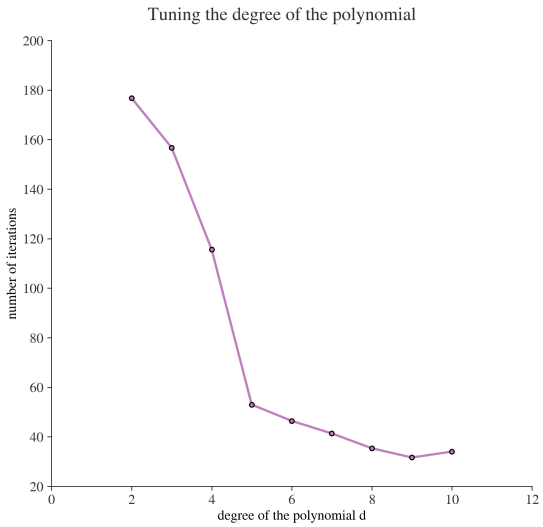
The "trick" in polynomial preconditioner: $\pi(A) = 1 - \phi(A) = 1 - Ap(A)$, and we use the harmonic Ritz values coming from $\pi(A)$.

More on polyn. prec. @ J. Loe and R. Morgan. *et al. Toward efficient and stable polynomial preconditioning for GMRES.*

Tuning : algorithmically $\rightarrow$ k, u and d

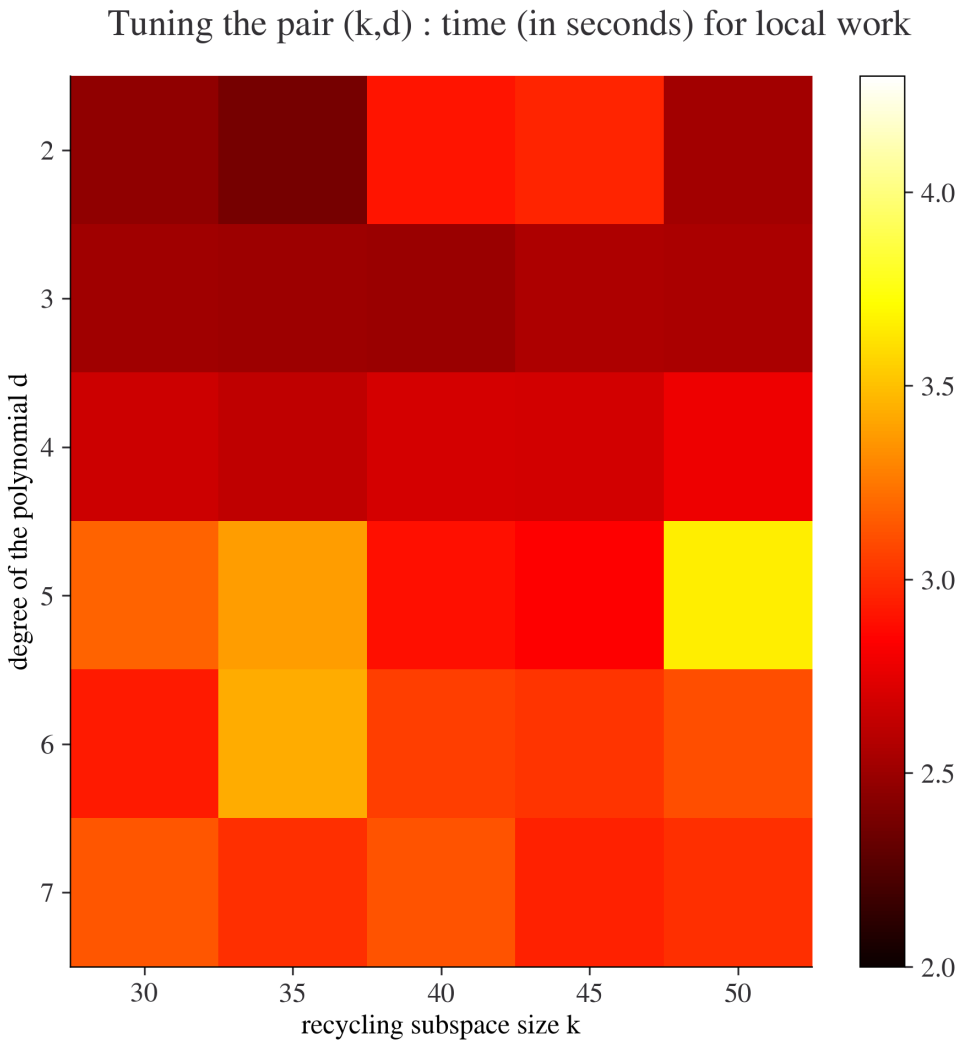




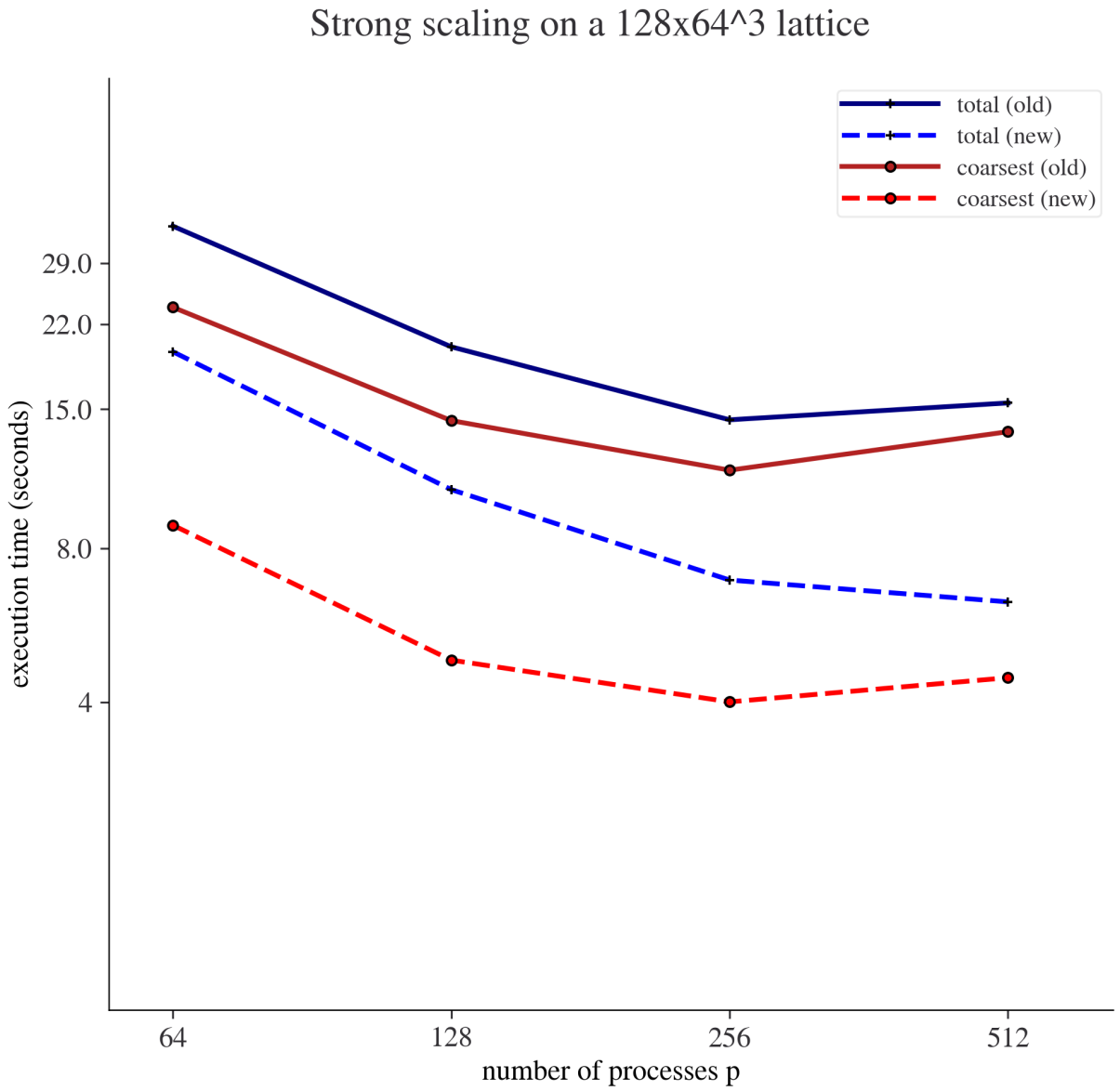


Tuning : computationally

On top of the three previously mentioned methods, we use pipelining (depth=1).



Scaling


Strong scaling on a 128x64^3 lattice
total (old)
total (new)
coarsest (old)
coarsest (new)
29.0
22.0
15.0
8.0
4
execution time (seconds)
64
128
256
512
number of processes p

Conclusions

Conclusions:

- polynomial preconditioner manages to improve the convergence of GCRO-DR and at the same time reducing the work
- the MPI_Wait(..) from nearest-neighbor communications becomes an issue at very large scales due to the reduction of all other contributions
- pipelining interferes with local work and local MPI_Wait(...) at very large scales

Future work:

- further reduce the flops with e.g. more vectorization
- further reduce the wait's overhead by adding e.g. persistent communications
- switch to something different from polynomial preconditioner that (even with some more local work) gives better convergence when combined with GCRO-DR and that doesn't include the wait's overhead

Density of states (coarsest level)

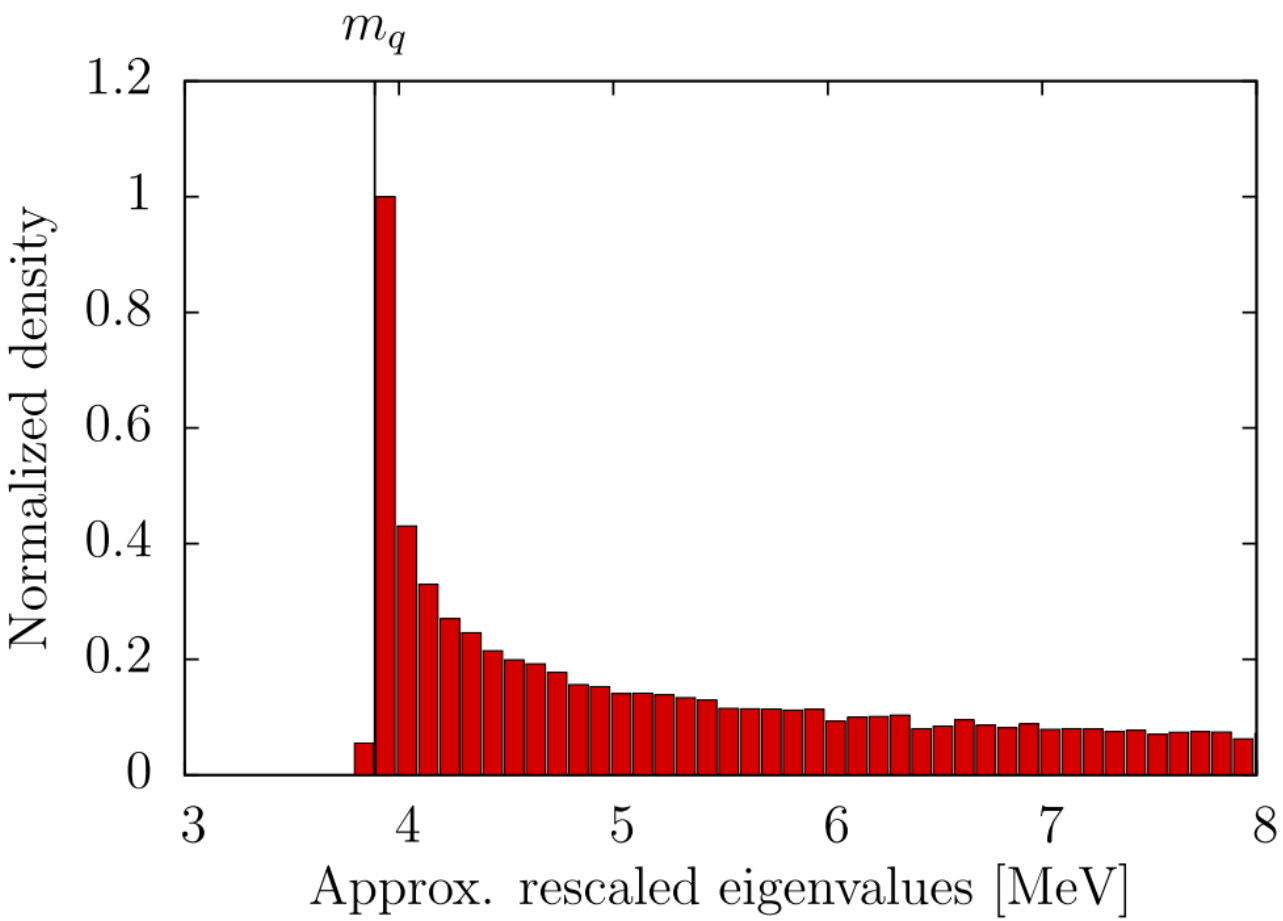


Image taken from : S. Bacchio. PhD thesis : *Simulating Maximally Twisted Fermions at the Physical Point with Multigrid Methods.*