Split Grid and Block Lanczos Algorithm for Efficient Eigenpair Generation

Yong-Chull Jang (ypj@bnl.gov) and Chulwoo Jung (chulwoo@bnl.gov)

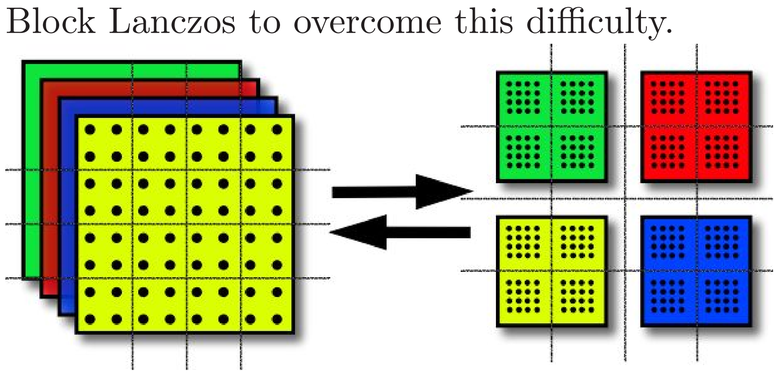
BROOKHAVEN NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

Introduction

Imbalance between the internode bandwidth and the computational capability of each node is increasingly limiting the amount of flops which can be efficiently deployed on a single job. Even when "Embarrassingly parallel" approach is applicable, the need for memory for intermediate data (propagators, A2A vectors...) often forces users to run on more than optimal number of nodes, or rely heavily on Disk I/O.

Implicitly Restarted Lanczos(IRL, Sorenson and Lehoucq) has been used by RBC/UKQCD collaborations (adopted for DWF by T.Blum, T. Izubuchi) to generate lowest-lying exact eigenvectors effectively and allowed a significant reduction in necessary flops for subsequent calculations in combination with AMA, A2A techiniques. However, the need for memory for all the eigenvectors negatively affects the performance of IRL on many machines. Here we explore combining the Split Grid technique with Block Lanczos to overcome this difficulty.



Contractions, LinAlg .. $\longleftrightarrow$ D_{op}..

Block Lanczos(BL)

To calculate lowest eigenvalues of Chebyshev accelerated precondition Dirac matrix $A = T_n(M_{pc})$, BL starts with N_u orthonormal vectors $Y_0 = \{y_0, \cdots y_{N_u-1}\}$.

$n \leftarrow 1, N_r \leftarrow 1$

1: $\tilde{Y}_n = AY_{n-1}$

2: $\tilde{Y}_n = \tilde{Y}_n - Y_{n-2}\boldsymbol{\beta}^\dagger_{n-2} (n > 1)$

3: $\boldsymbol{\alpha}_{n-1} = Y^\dagger_{n-1}\tilde{Y}_n$

4: $\tilde{Y}_n = \tilde{Y}_n - Y_{n-1}\boldsymbol{\alpha}_{n-1}$

5: Orthonormalize $\tilde{Y}_n$: $\tilde{Y}_n = Y_n\boldsymbol{\beta}_{n-1}$

(Orthogonalize $\tilde{Y}_n$ against $\{Y_1, \cdots Y_{n-1}\}$ for numerical stability)

6: $n \leftarrow n + 1$,

repeat 1-5 until $n = (N_k + N_r \times N_p)/N_u$

7: Calculate eigenvector λ'_i of matrix

$$H_{n-1} = \begin{bmatrix} \boldsymbol{\alpha}_0 & \boldsymbol{\beta}^\dagger_0 & \cdots & & 0 \\ \boldsymbol{\beta}_0 & \boldsymbol{\alpha}_1 & \boldsymbol{\beta}^\dagger_1 & \cdots & \\ 0 & \boldsymbol{\beta}_1 & \boldsymbol{\alpha}_2 & \boldsymbol{\beta}^\dagger_2 & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \boldsymbol{\beta}_{n-3} & \boldsymbol{\alpha}_{n-2} & \boldsymbol{\beta}^\dagger_{n-2} \\ 0 & \cdots & 0 & \boldsymbol{\beta}_{n-2} & \boldsymbol{\alpha}_{n-1} \end{bmatrix}$$

7: Construct approx. eigenvectors λ_i from λ'_i.

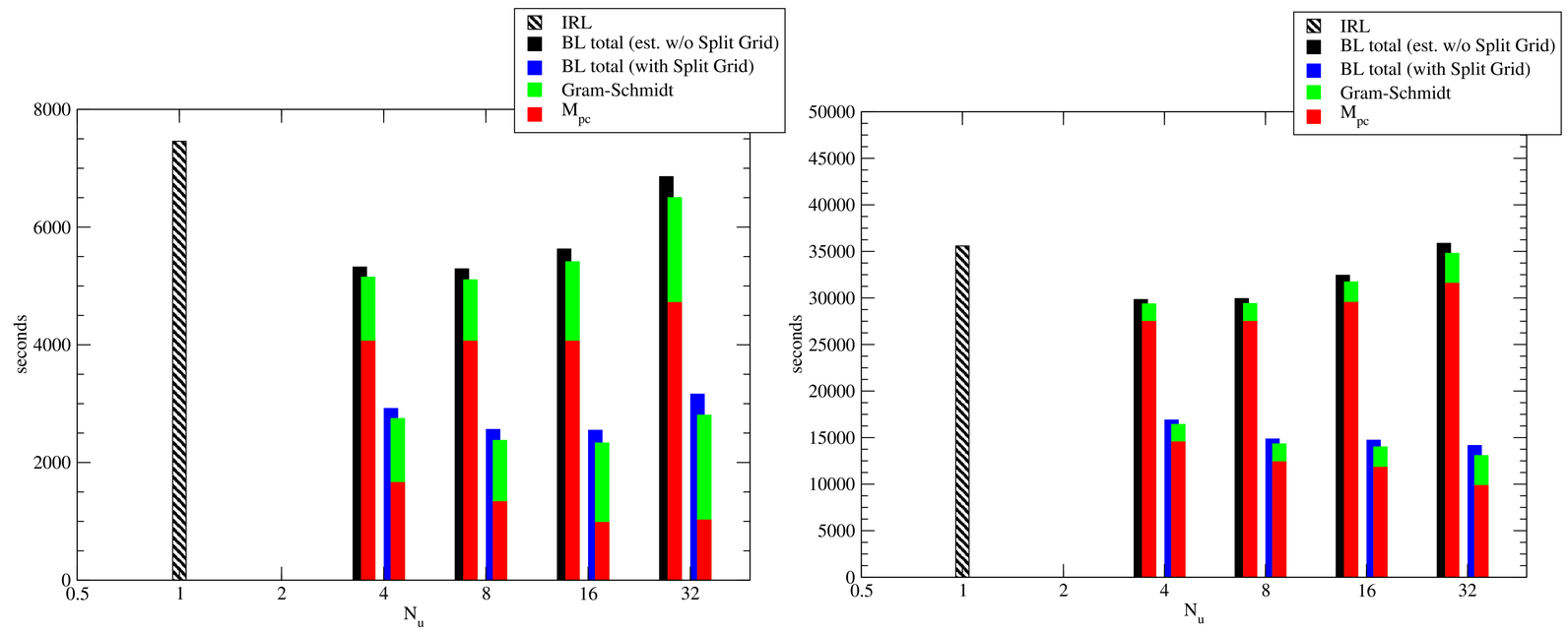
8: Check for convergence.

If less than $N_{stop}(\leq N_k)$ eigenvectors converged, $N_r \leftarrow N_r + 1$,

While it is also possible to use Implicitly Restarted BL (IRBL, Baglama, Calvetti, Reichel and Ruttan), similar to IRL, Only N_p/N_u shift is possible instead of N_p, which makes it less effective.

Block Lanczos Timings

ALCF Theta (KNL 7230, 64 cores/node +Cray XC40)



24ID ensemble (128 nodes) 48ID ensemble (512 nodes)

Factor of ~ 3 improvement seen. Further optimization in GS can give another factor of 2.

Convergence of Block Lanczos on DWF 2+1f ensemble (24ID)

$24^3 \times 64 \times 12$ zMobius , a ~0.2fm. $N_{stop} = 1000, N_k = 1024, N_p = 320$

$N_u = 4$

N_r	1	2	3
N_{conv}	96	832	1056

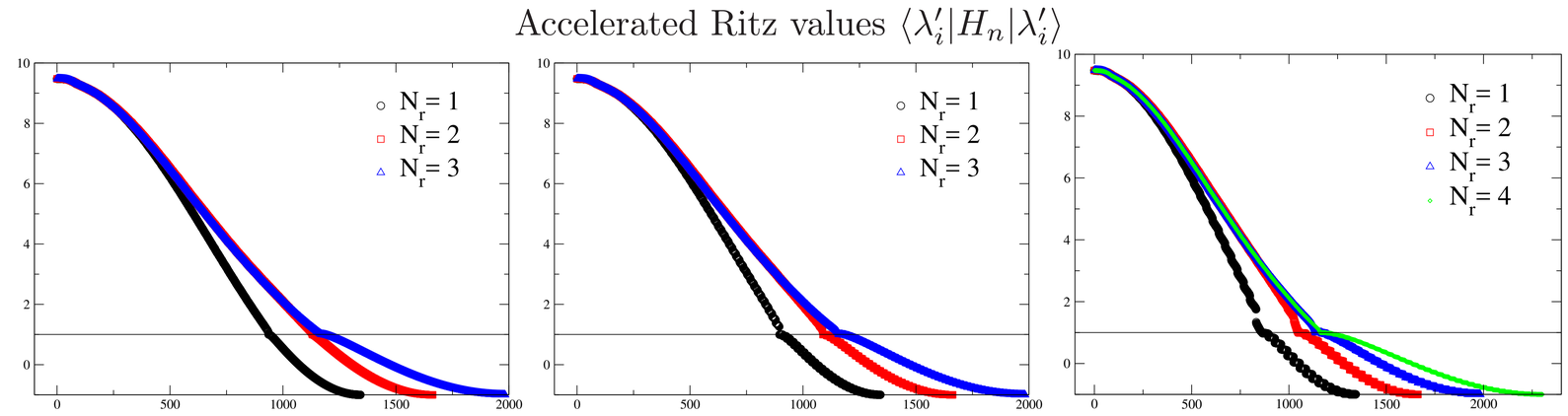
$N_u = 16$

N_r	1	2	3
N_{conv}	0	576	1024

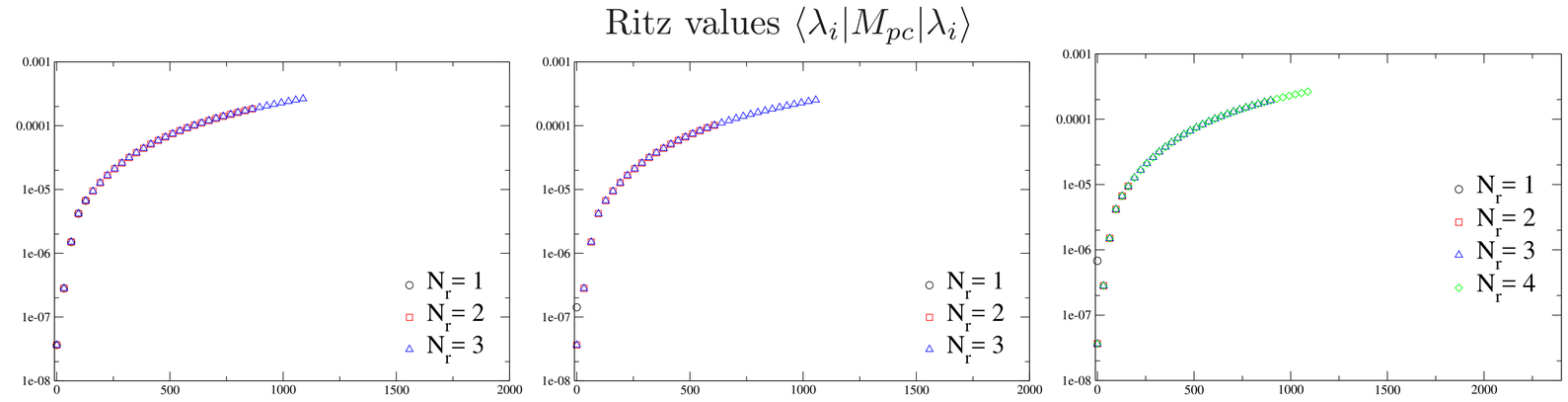
$N_u = 32$

N_r	1	2	3	4
N_{conv}	0	128	864	1056

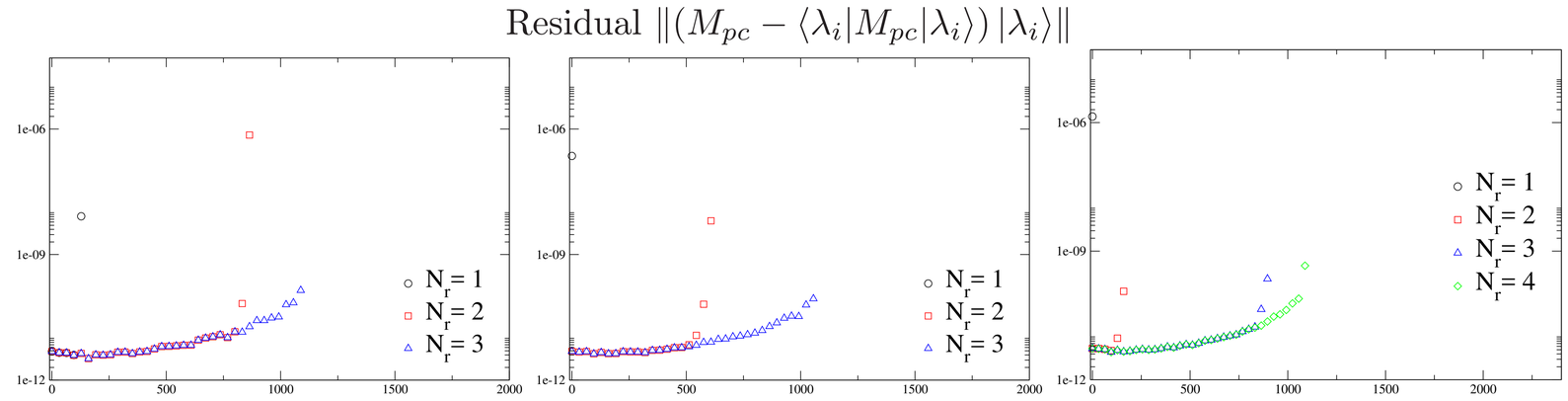
Accelerated Ritz values $\langle\lambda'_i|H_n|\lambda'_i\rangle$



Ritz values $\langle\lambda_i|M_{pc}|\lambda_i\rangle$



Residual $\|(M_{pc} - \langle\lambda_i|M_{pc}|\lambda_i\rangle)\,|\lambda_i\rangle\|$



Summary & Discussion

- Block Lanczos(BL) in combination with Split Grid achieves a significant speed-up compared with IRL. Number of Dirac operator application needed for convergence does not increase for up to $N_u = 16$ for existing DWF ensembles.
- Further optimization is possible, especially in Gram-Schmidt. Optimization is ongoing.
- The change in the Krylov space $\mathcal{K}_{nN_u}(A, y_0) = \text{span}\{y_0, Ay_0, \cdots A^{nN_u-1}y_0\} \rightarrow \text{span}\{y_0, \cdots, y_{N_u-1}, Ay_0, \cdots A^{n-1}y_{N_u-1}\}$ eventually forces BL to require more iterations than IRL. Further tuning of acceleration polynomial could improve the convergence for larger N_u.
- BL has been implemented in a local branch of Grid (https://github.com/paboyle/Grid).