Split Grid and Block Lanczos Algorithm for Efficient Eigenpair Generation

Yong-Chull Jang and Chulwoo Jung[*]
Physics department, Brookhaven National Laboratory, Upton, NY 11973, U.S.A.
E-mail: ypj@bnl.gov, chulwoo@bnl.gov

The increasing imbalance between the computing capabilities of individual nodes and the internode bandwidth makes it highly desirable for any Lattice QCD algorithm to minimize the amount of internode communication. One of the relatively new methods for this is the 'Split Grid' or 'Split Domain' method, where data is rearranged within the running of a single binary, so that the routines which requires significant off-node communications such as Dirac operators are run on multiple smaller partitions in parallel with a better surface to volume ratio, while other routines are run in one partition.
While it is relatively straightforward to utilize Split Grid method for inverters, the typical Lanczos algorithm which has one starting vector does not render itself naturally to Split Grid method. In this report we investigate the Block Lanczos algorithm(BL), which allows multiple starting vectors to be processed in parallel. It is shown that for a moderate number of starting vectors, BL achieves convergence comparable to similarly tuned Implicitly Restarted Lanczos algorithm (IRL)[1] on 2+1-flavor physical DWF/Möbius ensemble.

The 36th Annual International Symposium on Lattice Field Theory - LATTICE2018
22-28 July, 2018
Michigan State University, East Lansing, Michigan, USA.

[*]Speaker.

1. Introduction

One of the characteristics of Lattice QCD(LQCD) which is different from other HPC applications is that the multi-node performance of LQCD applications is often limited by not only the network latency, but also the bandwidth, due to the nature of discretized grid Dirac operators which has relatively low computational intensity. Even when “Embarrassingly parallel” approach is applicable, the need for memory for intermediate data such as eigenvectors, propagators and All-to-all(A2A) vectors often forces users to run on more than optimal number of nodes, or rely heavily on disk I/O.

Implicitly Restarted Lanczos[1] in combination with fine-tuned Chebyshev acceleration has been used by RBC/UKQCD collaborations successfully for some time. The recent development of Multi-grid Lanczos[2] has made it possible to generate even more eigenvectors without corresponding increase in memory requirement, generating up to 8000 lowest-lying exact eigenvectors for 2+1-flavor physical DWF/Möbius emsembles with the numerical cost similar to 20-30 undeflated inversions.

In addition to eliminating the critical slowing down as we approach phyical quark masses, exact eignvectors of preconditioned DWF/Möbius Dirac operators has been used to construct very accurate low-mode base for the All/Low Mode Averaging (AMA/LMA)[3] and All-to-all[4, 5] methods. However, the use of single starting vector makes IRL inherently serial, which limits the performance of the Lanczos kernel.

Recently, RBC/UKQCD collaboration has started to explore the ‘Split Grid’ method, where an application switches between running on one domain defined by single MPI communicator, and on multiple domains, often defined by split MPI communicators or something equivalent. Figure 1 illustrates the placement and movement of data in an application with Split Grid capability. Split Grid allows the communication intensive parts such as Dirac operator applications to run on multiple, smaller domains, while routines without much communications are done on the original domain. Split Grid acheived a significant speed-up in G-parity $K \to \pi\pi$ calculation reported in this conference[6], done on KNL based machines such as NERSC Cori in part. The routines necessary for data rearrangement between the two domains have been implemented in Grid[7], and as an standalone package using MPI RMA routines[8]. For other fermion formulations where memory bandwidth is a limiting factor, utilizing block solvers can lead to a similar speed-up[9].

Here we explore utilizing the Split Grid method for Lanczos by switching to Block Lanczos algorithm where there are multiple starting vectors. A detailed description of the algorithm is presented in Section 2. The performance and convergence of BL in comparison with IRL on 2+1-flavor physical ensemble is presented in Section 3. Summary and discussion is presented in Section 4.

2. Block Lanczos

Block Lanczos(BL) algorithm modifies the normal Lanczos by starting from $N_u(>1)$ orthonormal vectors $Y_0 = \{y_0^0, \cdots y_0^{N_u-1}\}$ and calculate lowest N_{stop} eigenvalues of matrix

$$A = T_n(X(M_{pc}, \lambda_l, \lambda_h)), \tag{2.1}$$

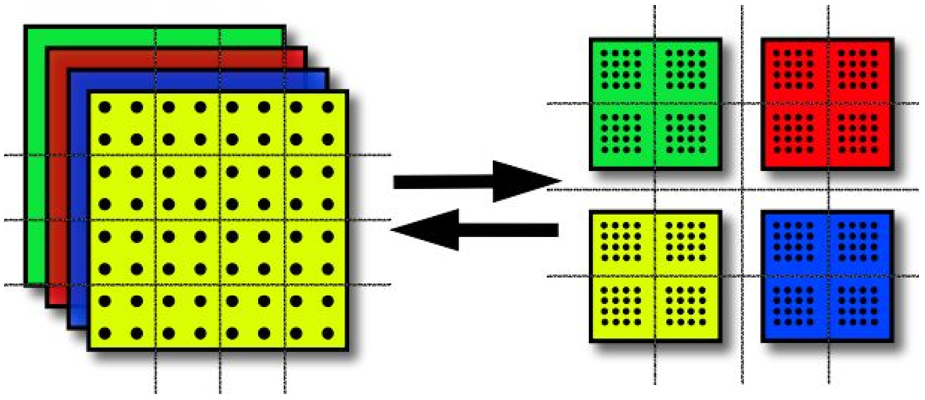


Figure 1: Illustration of Split Grid algorithm

where M_{pc} denotes preconditioned Dirac matrix, $T_n(x)$ the Chebyshev polynomial of the first kind, and $X(x,\lambda_l,\lambda_h)$ is a linear function where $X(\lambda_l) = -1$ and $X(\lambda_h) = 1$. Following a common notation for IRL[1], $N_k(> N_{stop})$ is the number of initial Lanczos iterations before restarts, or convergence checks. N_p denotes the number of the application of matrix between restarts, and N_r is the number of restarts. It is assumed that both N_k and N_p are divisible by N_u.

Similar to the single-vector Lanczos, the basic equation for BL is

$$AY_{n-1} = Y_{n-2}\boldsymbol{\beta}^{\dagger}_{n-2} + Y_{n-1}\boldsymbol{\alpha}_{n-1} + Y_n\boldsymbol{\beta}_{n-1} \tag{2.2}$$

The coefficient $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ now are $N_u \times N_u$ complex matrix instead of real numbers and $\boldsymbol{\beta}$ is an upper triangular matrix by construction via Gram-Schmidt processes.

Besides the number of starting vector(s), the biggest difference between IRL and BL is the restarting strategy: in IRL, after each covergence check, a rotation matrix constructed from QR rotation matrices which suppresses eigenmodes with unwanted eigenvalues reduces the number of Lanczos vectors from $N_k + N_p$ back to N_k, so that it can repeat without continuing increase of memory usage. In BL studied here, all the Lanczos vectors are kept until the convergence is reached. While it can be in principle problematic, in practice it is not a concern: The distribution of eignevalues of a given Lattice QCD ensemble is extremely stable. This means a well tuned Chebyshev polynomial T_n allows the convergence to be reached with a relatively small numeber of extra vectors, i.e. $N_p \times N_r \lesssim N_k$. Convergence is checked by calculating residual

$$R_i = \|(M_{pc} - \langle v_i|M_{pc}|v_i\rangle)\,|v_i\rangle\|. \tag{2.3}$$

We also have studied a more direct extension of IRL in the name of Implicitly Restarted BL (IRBL)[10]. We decided not to pursue this algorithm, as each rotation with QR destroys N_u rows instead of 1, which means the degree of filtering polynomial available at each restart decreases by the factor of N_u, which slows the convergence significantly.

Despite the similarities between the 2 algorithms, there is a qualitative difference between the Krylov space generated:

$$\mathcal{K}_{nN_u}(A,y_0) = \text{span}\{y_0, Ay_0, \cdots A^{nN_u-1}y_0\} \rightarrow \text{span}\{y_0,\cdots,y_{N_u-1}, Ay_0,\cdots,A^{n-1}y_{N_u-1}\} \tag{2.4}$$

As the Chebyshev acceleration is a critical part of the fast convergence of IRL for LQCD, Eq. 2.4 suggests the number of matrix application for converging the same number of vectors will increase for a large enough N_u. The details of BL is given in Algorithm 1.

Algorithm 1 Block Lanczos

```
 1: procedure $BL(A,\{\lambda_i,v_i\})$
 2:     $n \leftarrow 1, N_r \leftarrow 1, Y_0 = \{y_0^0, \cdots y_0^{N_u-1}\}$
 3:     while not converged do
 4:         while $n < (N_k + N_r \times N_p)/N_u$ do
 5:             $\tilde{Y}_n = AY_{n-1}$
 6:             For $n > 1, \tilde{Y}_n \leftarrow \tilde{Y}_n - Y_{n-2}\boldsymbol{\beta}^\dagger_{n-2}$
 7:             Orthogonalize $\tilde{Y}_n$ against $\{Y_1, \cdots Y_{n-2}\}$ (for numerical stability )
 8:             $\boldsymbol{\alpha}_{n-1} = Y^\dagger_{n-1}\tilde{Y}_n$
 9:             $\tilde{Y}_n \leftarrow \tilde{Y}_n - Y_{n-1}\boldsymbol{\alpha}_{n-1}$
10:             Orthonormalize $\tilde{Y}_n$: $\tilde{Y}_n = Y_n\boldsymbol{\beta}_{n-1}$
11:             $n \leftarrow n+1$,
12:         end while
13:         Calculate eigenvector $v'_i$ of matrix
```

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

```
14:         Construct approximate eigenvectors $v_i$ of $A$ from $v'_i$ by $v_i = \boldsymbol{Y}v'_i, \boldsymbol{Y} = \{Y_0, Y_1, \cdots Y_{n-1}\}$.
15:         Check for convergence using Eq. 2.3. If less than $N_{stop}$ eigenvectors converged, $N_r \leftarrow N_r + 1$.
16:     end while
17: end procedure
```

3. Performance of Block Lanczos on DWF $N_f = 2+1r$ ensemble on ALCF Theta

Here we describe the performance and convergence of BL on 2+1-flavor physical DWF+ID ensembles with $V = (24^3, 48^3) \times 64 \times 12, a \sim$0.2 fm[11] which we refer as 24ID and 48ID ensembles respectively. While the evolution was done with Möbius $Ls = 24, b+c = 4$, RBC/UKQCD has been using single precision zMöbius with $Ls = 12$ for the eigenvector generation.

For a realistic comparison, we start from the parameter RBC/UKQCD have been using for the generation of 1000 eigenvectors in fine grid, which forms the basis of Multi-Grid Lanczos[2] for 2000 and 8000 eigenvectors for 24ID and 48ID ensembles respectively. Since the convergence check on every Ritz vector is expensive and our experience with IRL suggest the convergence is

	$24^3 \times 64$ (24ID), 128 nodes					$48^3 \times 64$ (48ID), 512 nodes				
N_u	1(IRL)	4	8	16	32	1(IRL)	4	8	16	32
GF/node	22	53	65	88	99	36	65	76	85	110
$N_p \times N_r$										
320	171	96	0	0	0	32	0	0	0	0
480	621	-	-	-	-	44	0	0	0	0
640	967	832	704	576	128	61	32	0	0	0
800	1040	-	-	-	-	839	768	704	32	0
960		1056	1056	1024	864	1040	928	864	736	608
1120					-		1120	1088	928	736
1280					1056				1088	964
1440										1024

Table 1: Performance of the Chebyshev-accelerated Dirac operator (Eq. 2.1) in GFlops/node and the number of converged eigenvectors from single precision IRL and BL after each convergence check on DWF+ID ensembles. Numbers on the leftmost column below $N_p \times N_r$ are the total number of the operator application after initial Lanczos process for N_k iteration. N_k is 1040 for IRL and 1024 for BL. N_p is 160 for both IRL and BL on 48ID, and 320 for BL on 24ID.

almost always in increasing order in unaccelerated (without Chebyshev acceleration) eigenvalue or in decreasing order in accelerated Ritz value, we check the residual of every 32 Ritz vectors. Table 1 shows the number of converged eigenvectors after each convergence check.

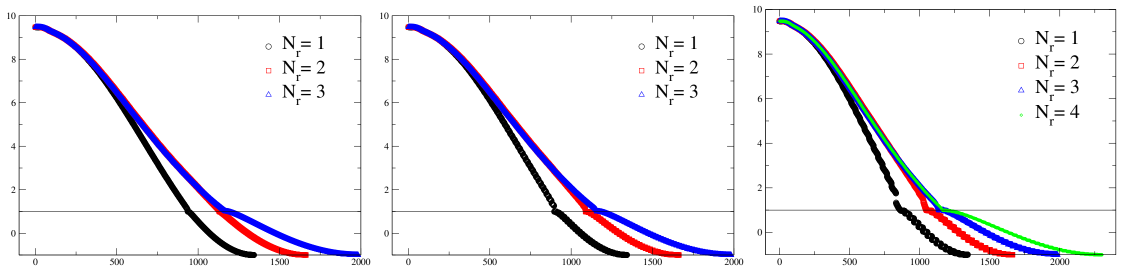


Figure 2: Accelerated Ritz values $\langle v_i'|H_n|v_i'\rangle$ of Eq. 13 for a $24^3 \times 64, a \sim 0.2$fm 2+1f DWF configuration

Figures 2-4 shows the evolution of both accelerated and unaccelerated Ritz values and residuals after each restart. Figure 5 shows the breakdown of different part of BL compared to IRL run on the same number of nodes. The first bargraph for each N_u is and estimate of timing for BL without split grid. They are estimates, as only the performance of Chebyshev kernel without Split Grid was measured, we can reasonably assume timings for the other parts of BL are the same with or without Split Grid.

As also shown in Table 1, the number of converged eigenvalues for a particular N_r decreases as N_u increases. However, the increase in time from the additional matrix application is more than well compensated by the performance increase from increasing local lattice volume. In fact, for the 48ID configuration, the total time is the smallest for the largest block size tried ($N_u =$

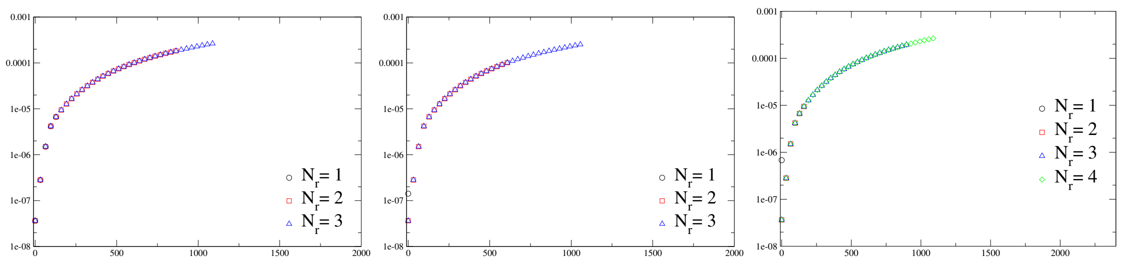


Figure 3: Unaccelerated Ritz values $\langle v_i | M_{pc} | v_i \rangle$

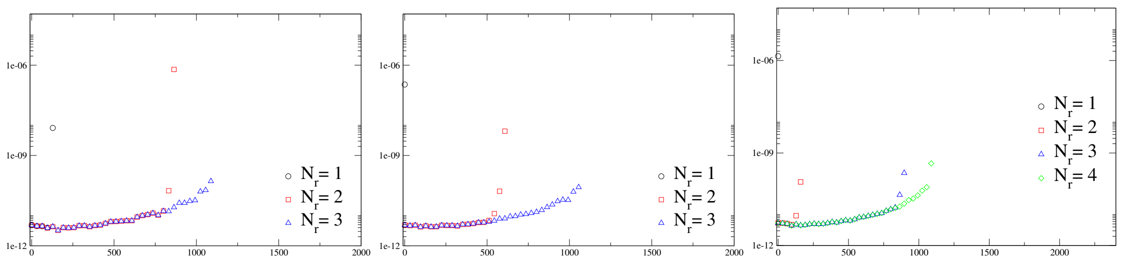
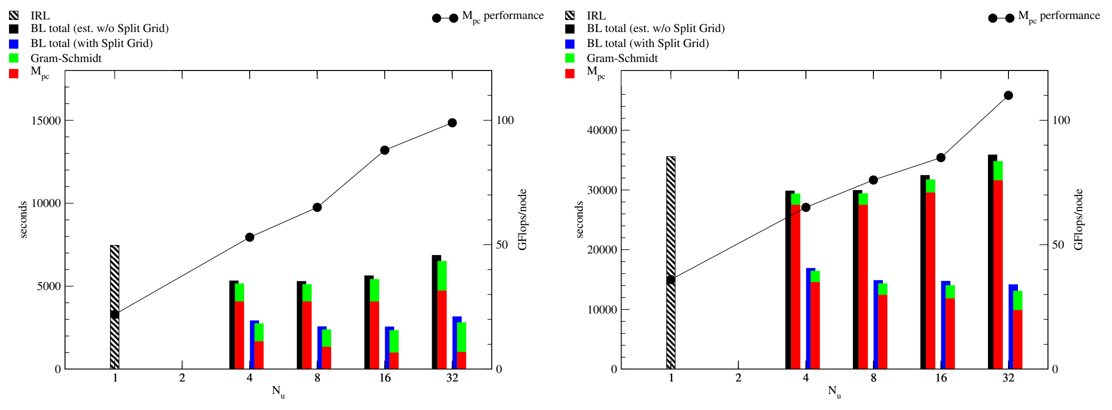


Figure 4: Residual of Ritz vectors (Eq. 2.3) at each convergence check for a Block Lanczos on a 24ID configuration[11].

32), and further increase in N_u may still be beneficial. It should be noted that the time spent on orthogonalization, shown as bars in green in Fig. 5, will grow larger relative to M_{pc} for larger N_{stop}, dminishing gains relative to IRL. However, Gram-Schmidt and other linear algebra routines in BL can be further optimized by cache blocking or similar techniques made possible by the presence of multiple Lanczos vectors. Further optimization is ongoing.



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

Figure 5: Performance of Chebyshev accelerated Dirac operaor $A = T_n(M_{pc})$ and timings of BL and IRL on 2+1-flavor DWF+ID ensemble, from Theta cluster(Cray XC40 with Intel Knights Landing CPUs) at Argonne Leadership Computing Facility(ALCF).

4. Summary & Discussion

Block Lanczos, in combination with Split Grid method, achieves a significant speed-up compared with the well tuned Implicitly Restarted Lanczos for the generation of lowest-lying eigenvectors of DWF/Möbius 2+1-flavor physical ensemble on Intel Knights landing clusters. While the difference in the Krylov space generated by BL and IRL shown in Eq. 2.4 negatively affects the convergence of BL, the increase in the number of Dirac operator application needed for convergence was moderate for up to $N_u = 32$ for the DWF+ID physical ensembles. In fact, the most increase in the total runtime for $N_u > 16$ was from the additional Gram-Schmidt necessary to keep the Lanczos vectors mutually orthogonal. Cache blocking and other well know techniques for local computation routines should extend the range of usability for BL. It is also possible that a different tuning of acceleration polynomial could improve the convergence.

Acknowledgement

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. Block Lanczos investigated here is based on the Grid data parallel C++ mathematical object library[7].

References

[1] D. Calvetti, L. Reichel and D. C. Sorensen, *Electronic Trans. Numer. Anal.* **2** (1994) 1.

[2] M. A. Clark, C. Jung and C. Lehner, *EPJ Web Conf.* **175** (2018) 14023 [1710.06884].

[3] T. Blum, T. Izubuchi and E. Shintani, *Phys. Rev.* **D88** (2013) 094503 [1208.4349].

[4] J. Foley, K. Jimmy Juge, A. O'Cais, M. Peardon, S. M. Ryan and J.-I. Skullerud, *Comput. Phys. Commun.* **172** (2005) 145–162 [hep-lat/0505023].

[5] T. Blum, P. A. Boyle, T. Izubuchi, L. Jin, A. Jüttner, C. Lehner, K. Maltman, M. Marinkovic, A. Portelli and M. Spraggs, *Phys. Rev. Lett.* **116** (2016), no. 23 232002 [1512.09054].

[6] C. Kelly, *PoS* **LATTICE2018** (2018) 277.

[7] https://github.com/paboyle/Grid

[8] https://github.com/chulwoo1/BlockScramble

[9] P. de Forcrand and L. Keegan, *Phys. Rev.* **E98** (2018), no. 4 043306 [1808.01829].

[10] J. Baglama, D. Calvetti and L. Reichel, *SIAM J. Scientific Computing* **24** (2003) 1650–1677.

[11] R.D. Mawhinney and J. Tu, *PoS* **LATTICE2018** (2018).