PROCEEDINGS OF SCIENCE

Optimization of the Oktay-Kronfeld Action Conjugate Gradient Inverter

Yong-Chull Jang[*], **Jon A. Bailey, Weonjong Lee**[1]
Lattice Gauge Theory Research Center, CTP, and FPRD,
Department of Physics and Astronomy, Seoul National University, Seoul, 151-747, South Korea
E-mail: [1]wlee@snu.ac.kr

Carleton DeTar[a,2], **Mehmet B. Oktay**[a,b]
[a]*Department of Physics and Astronomy, University of Utah, Salt Lake City, UT 84112, USA*
[b]*Department of Physics and Astronomy, University of Iowa, Iowa City, IA 52242, USA*
E-mail: [2]detar@physics.utah.edu

Andreas S. Kronfeld
Theoretical Physics Department,
Fermi National Accelerator Laboratory[†]*, Batavia, IL 60510, USA*
E-mail: ask@fnal.gov

SWME, MILC, and Fermilab Lattice Collaborations

Improving the Fermilab action to third order in heavy quark effective theory yields the Oktay-Kronfeld action, a promising candidate for precise calculations of the spectra of heavy quark systems and weak matrix elements relevant to searches for new physics. We have optimized the bi-stabilized conjugate gradient inverter in the SciDAC QOPQDP library and are developing a GPU code. The action is rewritten and the needed gauge-link combinations are precalculated. In tests with a MILC coarse lattice, this procedure accelerates the inverter by a factor of four. The remaining floating-point operations are mostly simple matrix multiplications between gauge links and fermion vectors, which we accelerate by more than an order of magnitude by using CUDA. Further gains could be achieved by using QUDA.

31st International Symposium on Lattice Field Theory - LATTICE 2013
July 29 - August 3, 2013
Mainz, Germany

[*]Speaker.

[†]Operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

PoS(LATTICE 2013)030

1. Introduction

The quantity ε_K describes indirect CP violation in the K^0-$\bar{K}^0$ system and enters tests of CKM unitarity and other searches for new physics. The dominant sources of uncertainty in the Standard Model (SM) value of $|\varepsilon_K|$ are, first, the theory uncertainty in $|V_{cb}|$, which stems from the form factors calculated with lattice QCD, and, second, the uncertainty in the matrix element $\hat{B}_K$, also from lattice QCD. The values for $\hat{B}_K$ and $|V_{cb}|$ used in Ref. [1] were updated at this conference [2, 3]. With these results, the tension between the SM calculation and the experimental measurement of $|\varepsilon_K|$ remains in excess of 3σ. (With the inclusive value [4] of $|V_{cb}|$, the tension vanishes. The exclusive and inclusive values of $|V_{cb}|$ differ by 3σ [3].)

New lattice calculations of the form factors of the exclusive decays $\bar{B} \to D^{(*)}\ell\bar{\nu}$, which are used to determine $|V_{cb}|$, are essential. Heavy-quark discretization errors are the largest source of uncertainty at present, and the Oktay-Kronfeld (OK) action [5] has been designed to reduce them. The OK action was developed by improving the Fermilab action [6] through third order in HQET [5]. With tree-level matching, the third-order improvement terms consist of four dimension-6 and two dimension-7 bilinears; no four-fermion operators arise. The HQET analysis suggests that the charm-quark discretization errors of the OK action are comparable to those of other highly-improved actions, while bottom-quark discretization effects are smaller [5]. In this report, we describe an optimized conjugate gradient (CG) inverter for the OK action. For performance tests we use the tree-level, tadpole-improved action that gave encouraging, albeit preliminary, results for the spectrum [7].

2. Optimization

2.1 Dirac Operator

For a Dirac operator M and source vector ξ, the system of equations

$$\sum_{y\beta b} M_{xy}^{\alpha\beta,ab}\psi_y^{\beta b} = \xi_x^{\alpha a} \tag{2.1}$$

must be solved to construct lattice correlators; x and y label the lattice sites, α and β are spin indices, and a and b are color indices. The solution vector ψ can obtained by the CG method. This algorithm iteratively updates the vector ψ from an initial guess. For each update the matrix multiplication of Eq. (2.1) is required.

We focus on optimizing this matrix multiplication by reducing the number of floating-point operations. We also consider how to exploit the size of local memory and node-to-node communication speed to increase efficiency without sacrificing performance gains from reducing the number of floating-point operations.

We first rewrite the OK action by collecting terms with products of gauge links multiplying the same neighboring fermion field. Suppressing spin and color indices,

$$\begin{aligned}\sum_y M_{xy}\psi_y = W_x^0\psi_x + \sum_\mu \left(W_{\mu,x}^+\psi_{x+\hat{\mu}} + W_{\mu,x}^-\psi_{x-\hat{\mu}}\right) + \sum_i \left(W_{ii,x}^{++}\psi_{x+2\hat{\imath}} + W_{ii,x}^{--}\psi_{x-2\hat{\imath}}\right) \\ + \sum_{j>i}\left\{W_{ij,x}^{++}\psi_{x+\hat{\imath}+\hat{\jmath}} + W_{ij,x}^{--}\psi_{x-\hat{\imath}-\hat{\jmath}} + W_{ij,x}^{+-}\psi_{x+\hat{\imath}-\hat{\jmath}} + W_{ji,x}^{+-}\psi_{x-\hat{\imath}+\hat{\jmath}}\right\},\end{aligned} \tag{2.2}$$

where $\mu = 1,2,3,4$, $\mu = 4$ is the temporal direction, and i runs over the spatial indices. For a given position x, each W is a 12×12 matrix in spin-color space. W consists of sixteen 3×3 color matrices. They are sums of gauge-link products of up to 5 links and a constant. Each gauge-link product carries a factor of ± 1 or $\pm i$ that depends on the involved γ_μ.

The W matrices remain the same for all CG iterations. We precalculate and reuse them to accelerate the CG iteration. In subsequent sections we call them "precalculation matrices."

2.2 Precalculation

Precalculation decreases the number of floating-point operations required for the Dirac operation, but saving the entire set requires too much memory. The full set of precalculation matrices in Eq. (2.2) has 432 color blocks. The size of a gauge configuration is 4 color blocks. It turns out that we do not need to hold everything in memory if we exploit the conjugate relation between opposite direction pairs of precalculation matrices. After introducing explicit representations for γ_μ, we can see that some color blocks are the same or equal zero.

Precalculation matrices multiplied to the off-diagonal($i \neq j$) next-to-nearest neighbor fermion fields satisfy the following relations.

$$W^{--}_{ij,x} = -W^{++\dagger}_{ij,x-\hat{i}-\hat{j}}\,, \quad W^{+-}_{ji,x} = -W^{+-\dagger}_{ij,x-\hat{i}+\hat{j}}\,. \tag{2.3}$$

The operation of Hermitian conjugation is applied in both spin and color spaces. Although the necessary relations are more complicated, the diagonal precalculation matrices $W^{-}_{\mu,x}, W^{--}_{ii,x}$ can be obtained from their positive direction counterparts $W^{+}_{\mu,x-\hat{\mu}}, W^{++}_{ii,x-2\hat{i}}$. Hermitian conjugation, a sign change, and color-block reordering are required, depending on the representation chosen for the Dirac matrices γ_μ. The relations are given explicitly in Eq. (2.10).

Hence, excepting W^0_x, the memory requirement for the precalculation matrices can be reduced by a factor of two. In the end, we can cut the required memory down to 50 color blocks, excluding the identical and vanishing color blocks.[1] As a by-product, the unnecessary floating-point operations required for constructing the precalculation matrices and for multiplying the fermion fields by null color blocks are removed.

Though beneficial in terms of reducing floating-point operations, exploiting the conjugation relations introduces a complicated field access pattern because the operation of Hermitian conjugation is applied to the precalculation matrix shifted in the opposite direction. This pattern can be seen in Eqs. (2.3) and (2.10). To update the fermion field on the site x, the fermion fields on the neighboring sites need to be collected to the site x. Then these and the on-site fermion field are multiplied by the pair of precalculation matrices. (In the temporal direction, only the nearest neighbors are involved. In the spatial directions, all the nearest and next-to-nearest neighbors participate.) However, using the conjugation relations requires collecting not only the fermion fields, but also the precalculation matrices, which reduces off-node performance. Copying the precalculation matrices can be avoided by simplifying the access pattern. We distribute precalculation matrices multiplied by a next-to-nearest neighbor fermion field over the nearest neighbors. This simplification is depicted in Fig. 1.

[1] The mass term in W^0_x is separately treated in practice. It saves memory by 1 more color block, instead of increasing the number of floating-point operations.

The shifted precalculation matrices can be identified by rewriting Eq. (2.2). We have

$$\sum_y M_{xy}\psi_y = W^0_x\psi_x + \sum_\mu \left(W^+_{\mu,x}\psi_{x+\hat{\mu}} + t_{-\mu}W^-_{\mu,x+\hat{\mu}}\psi_x \right) + \sum_i \left(t_i W^{++}_{ii,x-\hat{\imath}}\psi_{x+\hat{\imath}} + t_{-i}W^{--}_{ii,x+\hat{\imath}}\psi_{x-\hat{\imath}} \right)$$
$$+ \sum_{j>i} \left\{ t_j W^{++}_{ij,x-\hat{\jmath}}\psi_{x+\hat{\imath}} - t_{-i}W^{++\dagger}_{ij,x-\hat{\jmath}}\psi_{x-\hat{\jmath}} + t_{-j}W^{+-}_{ij,x+\hat{\jmath}}\psi_{x+\hat{\imath}} - t_{-i}W^{+-\dagger}_{ij,x+\hat{\jmath}}\psi_{x+\hat{\jmath}} \right\}, \quad (2.4)$$

where $t_{\pm\mu}$ are translation operators that shift the function (field) f_x by one lattice spacing in each direction.

$$t_{\pm\mu}f_x = f_{x\pm\hat{\mu}}\,. \quad (2.5)$$

The set of precalculation matrices saved for site x consists of $W^0_x, W^+_{\mu,x}, W^{++}_{ii,x-\hat{\imath}}, W^{++}_{ij,x-\hat{\jmath}}$ and $W^{+-}_{ij,x+\hat{\jmath}}$, $(i<j)$. Using the γ_μ representation in Ref. [6] and the notation of Ref. [7], the explicit form of the precalculation matrices is

$$W^0_x = \frac{u_0}{2\kappa} + \begin{pmatrix} D^0_x & S^0_x \\ -S^0_x & D^0_x \end{pmatrix}, \quad D^0_x = -\frac{c_B\zeta + 16c_5}{2u_0^3}\bar{B}^D_x, \quad S^0_x = -\frac{c_E\zeta}{2u_0^3}\bar{E}^S_x, \quad (2.6)$$

$$\bar{E}^S_x = \sum_{i=1}^3 \sigma_i E_{i,x}, \quad \bar{B}^D_x = i\sum_{i=1}^3 \sigma_i B_{i,x}, \quad W^+_{4,x} = \begin{pmatrix} 0 & S^+_{4,x} \\ S^+_{4,x} & -U_{4,x} \end{pmatrix}, \quad S^+_{4,x} = \frac{c_{EE}}{2u_0^4}(U_{4,x}\bar{E}^S_{x+4} - \bar{E}^S_x U_{4,x}), \quad (2.7)$$

$$W^+_{i,x} = \begin{pmatrix} D^+_{i,x} & S^+_{i,x} \\ S^+_{i,x} & D^+_{i,x} \end{pmatrix}, \quad S^+_{i,x} = \frac{1}{2}(\zeta - 2c_1 - 12c_2)\sigma_i U_{i,x} + \frac{c_3}{2u_0^4}(U_{i,x}\sigma_i\bar{B}^D_{x+i} - \sigma_i\bar{B}^D_x U_{i,x} + 2iB_{i,x}U_{i,x}),$$
$$D^+_{i,x} = -\frac{1}{2}(r_s\zeta + 8c_4)U_{i,x} + i\frac{c_5}{4}\left(u_0^{-2} - u_0^{-4}\right)\sum_{j,k=1}^3 \varepsilon_{ijk}\sigma_j(U_{k,x}U_{i,x+k}U^\dagger_{k,x+i} - U^\dagger_{k,x-k}U_{i,x-k}U_{k,x-k+i})$$
$$+ \frac{c_5}{u_0^4}\left[U_{i,x}\bar{B}^D_{x+i} + \bar{B}^D_x U_{i,x} - i\sigma_i(U_{i,x}B_{i,x+i} + B_{i,x}U_{i,x})\right], \quad (2.8)$$

$$W^{++}_{ii,x} = \begin{pmatrix} D^{++}_{ii,x} & S^{++}_{ii,x} \\ S^{++}_{ii,x} & D^{++}_{ii,x} \end{pmatrix}, \quad D^{++}_{ii,x} = \frac{c_4}{u_0}U_{i,x}U_{i,x+i}, \quad S^{++}_{ii,x} = \frac{c_1+2c_2}{2u_0}\sigma_i U_{i,x}U_{i,x+i}, \quad (2.9)$$

$$W^-_{4,x} = \begin{pmatrix} -U^\dagger_{4,x-4} & -S^{+\dagger}_{4,x-4} \\ -S^{+\dagger}_{4,x-4} & 0 \end{pmatrix}, \quad W^-_{i,x} = \begin{pmatrix} D^{+\dagger}_{i,x-i} & -S^{+\dagger}_{i,x-i} \\ -S^{+\dagger}_{i,x-i} & D^{+\dagger}_{i,x-i} \end{pmatrix}, \quad W^{--}_{ii,x} = \begin{pmatrix} D^{++\dagger}_{ii,x-2i} & -S^{++\dagger}_{ii,x-2i} \\ -S^{++\dagger}_{ii,x-2i} & D^{++\dagger}_{ii,x-2i} \end{pmatrix}, \quad (2.10)$$

$$W^{++}_{ij,x} = \begin{pmatrix} 0 & S^{++}_{ij,x} \\ S^{++}_{ij,x} & 0 \end{pmatrix}, \quad S^{++}_{ij,x} = \frac{c_2}{2u_0}(\sigma_i+\sigma_j)(U_{i,x}U_{j,x+i} + U_{j,x}U_{i,x+j}), \ (i\neq j), \quad (2.11)$$

$$W^{+-}_{ij,x} = \begin{pmatrix} 0 & S^{+-}_{ij,x} \\ S^{+-}_{ij,x} & 0 \end{pmatrix}, \quad S^{+-}_{ij,x} = \frac{c_2}{2u_0}(\sigma_i-\sigma_j)(U_{i,x}U^\dagger_{j,x-j+i} + U^\dagger_{j,x-j}U_{i,x-j}), \ (i\neq j). \quad (2.12)$$

The matrix multiplications in Eq. (2.4) are isolated from shift operations occurring before and after the multiplications. Pre-multiplication shifts gather nearest neighbor fermion fields. Post-multiplication shifts distribute the multiplication results to the nearest neighbors, followed by the fermion vector sum. Through these steps, shown in Fig. 2, each next-to-nearest neighbor fermion field contribution is propagated to the proper destination site. In a parallel computing environment, such as MPI, off-node communications are necessary only for the outermost boundary surfaces.

2.3 Implementation

In the USQCD library [8], a test version of the OK action CG inverter was included as part of QOPQDP. The inverter consists of a general purpose QOPQDP inverter that uses the bi-stabilized CG algorithm and a specific implementation of the OK action Dirac operator. The MILC library serves as our testing environment for the OK action CG inverters. We perform a mixed precision inversion by calling the QOPQDP inverter or Dirac operation module in single or double precision, as appropriate.

For CPU clusters, the optimized OK Dirac operator of Eq.(2.4) is implemented as part of QOPQDP. For GPU clusters, only part of the matrix multiplication in the optimized Dirac operator is replaced with CUDA function calls. To write working GPU code, we do not need to alter the parts of the optimized CPU code responsible for communication and precalculation. As reflected in the performance results, this GPU module can be further optimized.

3. Performance

To measure CG performance, we use a MILC coarse ($a \approx 0.12$ fm) lattice with dimensions $20^3 \times 64$. The lattice is divided to fit 4 nodes of the SNU cluster DAVID1. Each node consists of one core of an Intel i7-920 CPU together with an NVIDIA GTX480 GPU. Each node communicates with a single-rail QLogic InfiniBand network.

Precalculation reduces overall CG time by a factor of 3.9 times. This gain is increased to 13.1 when the precalculation matrix multiplication is performed with CUDA. (See Table 1a.) Table 1b and 1c show the timing details and CG performance in GFLOPS. Counting only the matrix multiplication, the maximum performance is 58.7 GFLOPS. Including memory copy time between host and GPU global memory for the precalculation matrix and the fermion field, the performance is decreased to 18.2 GFLOPS.

Because the CUDA module is called from the QOPQDP side, another overhead of QOPQDP preparation time arises. To pass the QDP data types to the external CUDA function, they should be exposed to the C intrinsic pointer variables. After arithmetic on the GPU, they must be recast

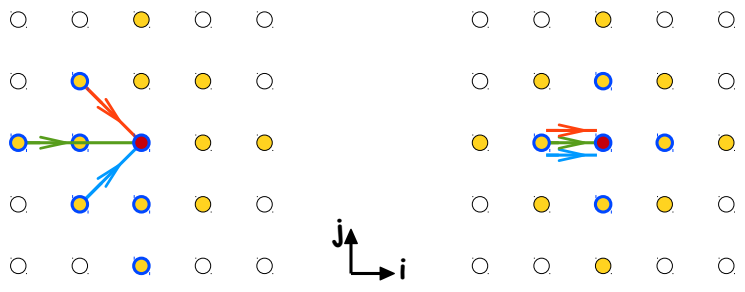

Figure 1: To update the fermion field on the site x (red), one needs in addition the fermion fields defined on the neighboring sites (yellow). The precalculation matrices defined on the blue-circled sites are also needed. By calculating shifted precalculation matrices, the field access pattern is simplified, saving floating-point operations.

Naive	Precalc.	CUDA
11814.8	3048.8	898.7

(a) CG Time

	Precalc.	CUDA
Matrix Multiplication	962[1206]	32[107]
CUDA Memory Copy, W		4[262]
CUDA Memory Copy, ψ		60[121]
QOPQDP Preparation		54[163]

(b) Matrix Multiplication

Communication	2[11]
Gamma Basis Change	16[32]
Spin Decomposition	23[45]
Vector Addition	69[66]

(c) Common Module

Table 1: CG performance: (a) The CG time is measured in seconds. "Naive" means the original CG inverter without any improvement. "Precalc" means the CG inverter with precalculation of W matrices. "CUDA" means the CG inverter with precalculation and with the Dirac opertor programmed in CUDA. (b) The values are measured in the unit of milliseconds. Each value represents the time elapsed per single CG iteration. Values in the bracket $[\cdots]$ correspond to the double precision calculation. (c) The same notation as in (b).

to the previous QDP data types. With the current implementation, the total CUDA overhead time, which consists of memory copy time (= CUDA Memory Copy, W + CUDA Memory Copy, ψ in Table 1b) and QOPQDP preparation time, exceeds the matrix multiplication time by a factor of 3.7 (5.1) for a single Dirac operation of single (double) precision.

The GTX480 has 1.5 GB of global memory, which is not large enough to hold all the necessary precalculation matrices at once. At best we can allocate GPU global memory space for the full set of single precision precalculation matrices. The double precision update is divided into two parts so that the double precision precalculation matrices can be held by the allocated GPU global memory space. For each double precision update, the sets of precalculation matrices are copied in succession from the host memory. Single precision precalculation matrices are copied at the beginning of the iterations and used again in each iteration. When the precision is changed, the precalculation matrices for the other precision are wiped from the GPU global memory. This

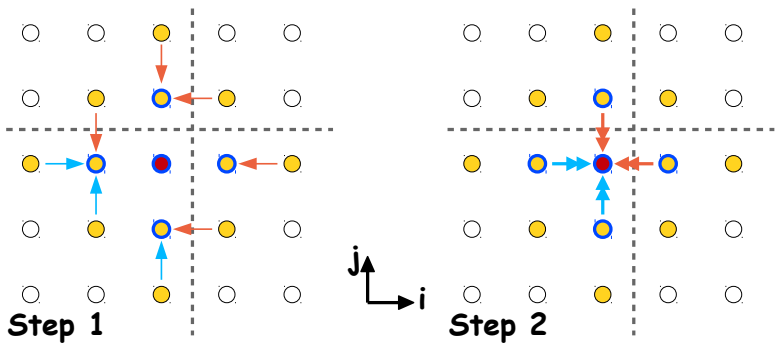

Figure 2: The dashed line is the computing node boundary. To update the fermion field (red), in the first step, only the nearest neighbor fermion fields are gathered and multiplied with the precalculation matrices (blue circles). In the second step, the resulting products are gathered from the nearest neighbor sites and added together.

precalculation matrix copy overhead can be overcome by using a GPU (such as GTX Titan) with global memory sufficient to store the single and double precision precalculation matrices.

The remaining overheads are the fermion field copy time (CUDA Memory Copy, ψ in Table 1b) and QOPQDP preparation time. Together these occupy 96.6% (52.0%) of the total CUDA overhead time in single (double) precision, so we expect more optimization of the GPU inverter can be achieved without new hardware.

4. Future Work

By developing the OK action inverter with QUDA, the QOPQDP preparation time can be removed. Reducing the CPU-GPU communication time requires reducing overheads from copying the fermion fields and the precalculation matrices; the latter could be addressed with hardware with more global memory. Finally, $M^\dagger M$ preconditioning, even-odd preconditioning, and spin projection are commonly used to optimize inversions of the Dirac operator with the CG algorithm. Even-odd preconditioning appears very difficult for the OK action; we have not yet investigated how to implement the remaining two techniques.

Acknowledgments

This work was supported in part by the U.S. Department of Energy under grant No. DE-FC0212ER-41879 (C.D.) and the U.S. National Science Foundation under grant PHY10-67881 (C.D.). The research of W. Lee is supported by the Creative Research Initiatives Program (2013-003454) of the NRF grant funded by the Korean government (MSIP). W. Lee would like to acknowledge the support from KISTI supercomputing center through the strategic support program for the supercomputing application research [No. KSC-2012-G3-08]. Computations were carried out on the DAVID GPU clusters at Seoul National University. The research of J.A.B. is supported by the Basic Science Research Program (2013009149) of the National Research Foundation of Korea (NRF) funded by the Ministry of Education.

References

[1] Y.-C. Jang and W. Lee, PoS(LATTICE2012)269 [hep-lat/1211.0792].

[2] Taegil Bae *et al.* [SWME], PoS(LATTICE2013)476 [hep-lat/1310.7319].

[3] S. Qiu, C. DeTar, A.S. Kronfeld *et al.* [Fermilab Lattice and MILC], these proceedings.

[4] P. Gambino and C. Schwanda, arXiv:1307.4551 [hep-ph], submitted to *Phys. Rev.* **D**.

[5] M. B. Oktay and A. S. Kronfeld, *Phys. Rev.* **D78** (2008) 014504 [hep-lat/0803.0523].

[6] A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, *Phys. Rev.* **D55** (1997) 3933 [hep-lat/9604004].

[7] C. Detar, and A. S. Kronfeld, and M. B. Oktay, PoS(LATTICE2010)234 [hep-lat/1011.5189].

[8] http://usqcd.jlab.org/usqcd-software.