Linear problem of overlap fermion operator in lattice QCD

Hideo Matsufuru for JLQCD Collaboration

High Energy Accelerator Research Organization (KEK),
Oho 1-1, Tsukuba 305-0801, Japan

Email: hideo.matsufuru@kek.jp

July 16, 2009, Ver.1.0

Abstract

This note summarizes the overlap-Dirac operator for those who investigates this operator as a large linear system.

Contents

1 Introduction

The overlap Dirac operator is one of the fermion operators in lattice gauge theories. Among lattice fermion operators, the overlap operator has an attractive feature that it has an exact chiral symmetry on the lattice. However, numerical simulation with the overlap operator has become feasible only recently, because of its large numerical cost. In 2006, the JLQCD Collaboration started a large scale simulation project with dynamical overlap fermions. Despite much progress in algorithm and machine power, improved algorithms of the overlap fermion are still strongly desired. Thus in this note the fundamental structures of the overlap operator and the linear problems to be solved are summarized.

In numerical simulation of lattice QCD, one needs frequently to solve a linear equation

$$\sum_{a,j,x} D_{ij}^{ab}(x,y) x_j^b(y) = b_i^a(x) \tag{1.1}$$

where D is the Dirac operator (here the overlap operator) which has the following indices: color $(a, b = 1, \ldots, 3)$, spinor $(i, j = 1, \ldots, 4)$, and site (x, y)[1]. For example, for a 16^4 lattice, the rank of matrix D is $N = 3 \times 4 \times 16^4 = 785,432$.

In lattice QCD simulations, the above linear equation must be solved mainly in the following two situation.

- *Computation of quark propagator.*
  The observables containing quark fields are constructed with the quark propagator, which is an inverse of the Dirac-operator D.

- *During generation of dynamical gauge configuration.*
  In early days of lattice QCD simulations, the quark determinant $\det(D)$ which represent the dynamical quark effect is often neglected and such simulations are called 'quenched approximation'. Nowadays, it has become popular to included the dynamical quark effect by the hybrid Monte Carlo (HMC) algorithm. In this algorithm, the dynamical quark effect is represented as 'pseudofermion' field ϕ by making use of the relation

  $$\det(D^\dagger D) = \int \mathcal{D}\phi^\dagger \mathcal{D}\phi \exp(-\phi^\dagger (D^\dagger D)^{-1} \phi). \tag{1.2}$$

  In HMC algorithm, first ϕ is given by Gaussian distribution, and then the gauge field is updated under fixed ϕ by molecular dynamics (MD). Since D depends on the gauge field and the force is required to update the gauge field, a linear equation $(D^\dagger D)x = \phi$ must be solved at every step of MD. This linear solver is the most time-consuming part of the HMC algorithm.

In this note, we focus on the latter, *i.e.* the equation to be solved is in the form $(D^\dagger D)x = b$. Of course if two successive solver for $Dx = b$ is faster, one should adopt it.

[1] Here we explicitly assume the SU(3) gauge theory (namely QCD) in 4-dimensional spacetime. In general, one can consider other group structure and spacetime dimension other than 4.

2 Overlap Dirac operator

2.1 Overlap operator

The overlap operator with the bare quark mass m is written

$$D(m) = \left(M_0 + \frac{m}{2}\right) + \left(M_0 - \frac{m}{2}\right)\gamma_5 \cdot \mathrm{sign}(H_W). \tag{2.1}$$

where $H_W(-M_0)$ is the hermitian Wilson-Dirac operator,

$$H_W(-M_0) \equiv \gamma_5 D_W(-M_0), \tag{2.2}$$

and D_W is the Wilson-Dirac operator[2],

$$D_W(-M_0; x, y) = 4 - M_0 - \frac{1}{2a}\sum_\mu \left[(1-\gamma_\mu)U_\mu(x)\delta_{x+\hat{\mu},y} + (1+\gamma_\mu)U_\mu^\dagger(x-\hat{\mu})\delta_{x-\hat{\mu},y}\right]. \tag{2.3}$$

$U_\mu(x) \in$SU(3) are link variables which are 3×3 complex matrices and related to the gauge field $A_\mu(x)$ as $U_\mu(x) = \exp[ig_0 A_\mu(x+\hat{\mu}/2)]$ with g the bare coupling constant. $\hat{\mu}$ is a unit vector in μ-th direction. γ_μ are 4×4 Dirac matrices which satisfies $\{\gamma_\mu, \gamma_\nu\} = 2\delta_{\mu,nu}$. To make the argument explicit, we adopt the chiral representation,

$$\gamma_1 = \begin{pmatrix} & & & -i \\ & & -i & \\ & i & & \\ i & & & \end{pmatrix}, \qquad \gamma_2 = \begin{pmatrix} & & & -1 \\ & & 1 & \\ & 1 & & \\ -1 & & & \end{pmatrix}, \qquad \gamma_3 = \begin{pmatrix} & & -i & \\ & & & i \\ i & & & \\ & -i & & \end{pmatrix}, \tag{2.4}$$

$$\gamma_4 = \begin{pmatrix} & & -1 & \\ & & & -1 \\ -1 & & & \\ & -1 & & \end{pmatrix}, \qquad \gamma_5 = -\gamma_1\gamma_2\gamma_3\gamma_4 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix}. \tag{2.5}$$

These γ_μ's are hermitian. Throughout this note, lattice spacing is set to unity: $a = 1$.

In the following, we use the hopping parameter representation of D_W

$$D_W \to 2\kappa D_W, \qquad \kappa = \frac{1}{2(4 - aM_0)}. \tag{2.6}$$

The normalization of D_W is irrelevant to the overlap operator, since it appears only in the sign-function.[3] In the overlap operator, M_0 needs to satisfy $0 < M_0 < 2$. Throughout this work we adopt $M_0 = 1.6$.

As a general property of the lattice Dirac operator, $D^\dagger = \gamma_5 D \gamma_5$ holds (called γ_5-hermiticity). Thus $H_W = \gamma_5 D_W$ is hermitian. The sign function, $\mathrm{sign}(H_W)$, is the most involved part for the practical implementation of the overlap fermion and will be described in the next subsection.

For practical implementation, $U_\mu(x)$ must be kept on the memory. On the other hand, γ_μ, which is represented as just permutation in spinor components and real/imaginary parts. Also $(1 \pm \gamma_\mu)$ plays a projector onto 2-component spinor. Thus before multiplication of U_μ, the 4-component spinor is projected onto 2-spinor, and afterward 4-spinor is reconstructed.

[2]While we adopt as the kernel D_W the simplest Wilson-Dirac operator, other Wilson-like operators including improved ones are also applicable.

[3]Caution: for the low-mode subtraction, the threshold parameter in the code is given by the hopping parameter representation, and differs in normalization to the value quoted in the paper.

Implementation: In the Fortran code, the Wilson fermion kernel is implemented in file `oprw5_chiral.f`. The overlap fermion kernel is implemented in file `opr_overlap_zolotarev.f`. The input parameters `Rm`, `Rm0` correspond to m and M_0.

2.2 Sign function

As already noted, the implementation of the sign function $\mathrm{sign}(H_W)$ is quite involved. One needs to compute a result of multiplication of $D(m)$ to a vector v. When $\mathrm{sign}(H_W)$ is applied to v, it expands v in terms of the eigenmodes of H_W, and assign ± 1 according the sign of the eigenvalues:

$$\mathrm{sign}(H_W) \cdot v = \sum_{\lambda} \mathrm{sign}(\lambda)(\psi_\lambda, v)\psi_\lambda. \tag{2.7}$$

(ψ_λ, v) is a inner product and given by $\psi_\lambda^\dagger \cdot v$. In numerical application, it is not realistic to determine all the eigenmodes. A standard procedure therefore applies the eigenmode decomposition only to low-lying modes, and employs some approximation formula to $\mathrm{sign}(H_W)$. As the latter, we adopt the Zolotarev's rational approximation [4, 5]. The polynomial approximation is also widely applied procedure. Since these approximations are valid in certain region of λ, highest eigenvalue is also needed to be computed (or to be set to certain value).

The steps to compute $\mathrm{sign}H_W \cdot v$ is as follows:

(i) Determine the low-lying (and highest) eigenmodes of H_W.
(ii) Subtract low-lying eigenmodes from the vector v:

$$\tilde{v} = v - \sum_{i=1}^{n_\lambda} (\psi_{\lambda_i}, v)\psi_{\lambda_i} \tag{2.8}$$

(iii) Multiply approximate sign-function $\epsilon(H_W)$ to $\tilde{v}$.
(iv) Then the total sign function, and thus $D(m)v$, is calculated as

$$\mathrm{sign}(H_W)v \simeq \sum_{i=1}^{n_\lambda} \mathrm{sign}(\lambda_i)(\psi_{\lambda_i}, v)\psi_{\lambda_i} + \epsilon(H_W)\tilde{v}. \tag{2.9}$$

2.3 Low-mode subtraction

It is necessary to determine the low-lying modes to subtract them from H_W. As an eigenvalue solver, we currently adopt the implicitly restarted Lanczos algorithm [2]. This method is based on the Lanczos algorithm. When N_k eigenmodes are desired to determine, this method extends the Krylov subspace $\mathcal{K}_k$ to $(N_k + N_p)$-dimension. Then information of N_p vectors in extra-space is compressed into the N_k-dimension space by applying implicitly shifted QR algorithm. Repeated application of extension and compression of Krylov subspace causes that the N_k vectors approach to the lowest N_k eigenvectors. After enough application of this step, the tridiagonal Lanczos matrix is diagonalized by QR algorithm. The same algorithm is also applied to determine the highest eigenmodes.

In practice, it is useful to determine the eigenvalues whose absolute values are less than certain threshold, V_{th}. Then n_λ is defined as the number of eigenvalues which satisfies $|\lambda| < V_{th}$. This is the policy adopted by the present program. Detailed description of eigenvalue solver will be presented in a separate note.

Implementation:
The eigenvalues of H_W are determined by routines in the files `eigen_wilson5_lex.f` and `qris.f`. The former implements the main part of the implicitly restated Lanczos algorithm, and the latter contains routines for implicitly shifted QR algorithm. In the common file, `eigen_wlex.h`, contains parameter `Nkmax`. This parameter specifies the maximum size of Krylov subspace, $N_k + N_p$, and hence it must be larger than the sum of the input parameters `Nkxmin` and `Nkxmin`, which are respectively N_k and N_p for the determination of low-lying eigenmodes. This is also true for `Nkxmax` and `Nkxmax` for the determination of highest eigenmodes (while practically the highest mode can be easily determined compared to the low-lying ones). The parameter `Nkmax` also appears in other common files, and must be changed simultaneously.

The parameter `Vthrs` corresponds to V_{th}. `Nsbt`, the number of subtracted eigenvectors, is counted accordingly in the program. The parameters `Enorm_eigen` specifies the precision of the eigenvalue relation, $H_W\psi_\lambda = \lambda\psi_\lambda$. The determined eigenvalues and eigenvectors are stored in common variables `TDa(Nkmax)` and `Vk(Nvst,Nkmax)`.

For an acceleration technique, the Chebychev acceleration is implemented. (More detailed description will be supplied.)

To do: The eigenvalue solver has not yet well improved. Within the framework of Lanczos method there may be some improvement. Alternative algorithms such as the CG algorithm [3] should be tested. For the recent simulation, the determination of low-lying eigenmodes of H_W takes about (1/4 of the total HMC for 2+1 flavor simulation). Improving this part is highly desired.

2.4 Zolotarev's rational approximation

The Zolotarev's rational approximation is represented as [4, 5]

$$\frac{1}{\sqrt{H_W^2}} = \frac{d_0}{\lambda_{min}}(h_W^2 + c_{2n})\sum_{l=1}^{n}\frac{b_l}{h_W^2 + c_{2l-1}} \tag{2.10}$$

where $h_W = H_W/\lambda_{min}$. Multiplying H_W to this function, $sign(H_W)$ is computed. This formula is valid for the interval $h_W \in [1, b]$ with $b = \lambda_{max}/\lambda_{min}$. The parameters d_0 and b_l are related to the coefficients c_l as follows.

$$d_0 = \frac{2\lambda}{1+\lambda}\prod_{l=1}^{n}\frac{1+c_{2l-1}}{1+c_{2l}} \tag{2.11}$$

$$b_l = d_0\frac{\prod_{i=1}^{n-1}(c_{2i} - c_{2l-1})}{\prod_{i=1,i\neq l}^{n}(c_{2i-1} - c_{2l-1})} \tag{2.12}$$

To determine d_0 precisely, the parameter λ (which is defined in terms of ϑ function) must also be determined. However, d_0 can practically be determined by enforcing that the approximate sign function exhibits least deviation from unity in the range $[1, b]$.

Alternatively, the following approximate expression is practically useful. By enforcing the sign function is unity at $h_W = 1$, $2\lambda/(1-\lambda) = 1$ follows. Then

$$d_0 = \prod_{l=1}^{n}\frac{1+c_{2l-1}}{1+c_{2l}}. \tag{2.13}$$

After gross setting of parameters, precise value of d_0 can easily be determined since the Zolotarev's formula gives minimax function, which ensures that the maximum deviations from the central value in the valid range are equal in positive and negative directions. Thus obtaining maximum and minimum values of the approximate function, d_0 is easily determined.

The coefficients c_l is given as [5][4]

$$c_l = \frac{\mathrm{sn}^2(lK'/(2n+1);K')}{1-2\mathrm{sn}^2(lK'/(2n+1);K')} \tag{2.14}$$

$$K' = u(1) = \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-\kappa'^2t^2)}}, \tag{2.15}$$

where $\kappa' = \sqrt{1-\kappa^2}$, $\kappa = \lambda_{min}/\lambda_{max}$. K' is the complete elliptic integral of the first kind with modulus κ', *i.e.*, the value of u such that $\mathrm{sn}(u;\kappa') = 1$.

One needs to determine c_l and K' for given $b = \lambda_{max}/\lambda_{mim}$ and n. $\mathrm{sn}(u,\kappa')$ must be computed somehow.

Implementation: First of all, elliptic function sn must be computed. We make use of a routine in Numerical Recipes [6]. In Fortran code, subroutine `Jacobi_elliptic` (`sncndn` in Numerical Recipes) compute $\mathrm{sn}(u,k_c)$, $\mathrm{cn}(u,k_c)$, and $\mathrm{dn}(u,k_c)$ for given u and $k_c = 1-k^2$. In subroutine `Poly_Zolotarev`, first the value of u which satisfies $\mathrm{cn}(u,k_c) = 0$ ($\mathrm{sn}(u,k_c)$=1) is determined to the 14-th digit precision. Then c_l, d_0, and b_l are determined according to Eqs. (2.15), (2.13), and (2.12), respectively. A program to check the Zolotarev approximation formula by giving $\mathrm{sign}(x)$ for real number x is also available.

2.5 Multi-shift CG solver

In the rational approximation of $\mathrm{sign}(H_W)$, one needs to solve equations

$$\left(H_W^2 + c_j\right) x = b, \qquad j = 1, \ldots n \tag{2.16}$$

for a single source vector b. If there is no efficient way to solve these equations, the rational approximation is not tractable.

Multi-shift solver [7, 8] enables to solve these n equations at once. We employ the multi-shift CG algorithm. The multi-shift algorithms make use of the fact that the Krylov subspace $\mathcal{K}_k(A,v_0)$ for a matrix A and initial vector v_0 is unchanged against a shift $A \to A+\sigma$, *i.e.*, $\mathcal{K}_k(A,v_0) = \mathcal{K}_k(A+\sigma,v_0)$. This implies that the residual vector, which is perpendicular to the previous Klylov subspace, can be shared by n equations in Eq. (2.16). Then n solutions of Eq. (2.16) can be determined simultaneously. The algorithm is summarized in Appendix A.

Todo:
On scalar machines, the double-path solver [9, 10] is worth to be considered as an alternative.

3 Overlap solver algorithm

3.1 Nested conjugate Gradient algorithm

Once the approximate $\mathrm{sign}(H_W)$ is at hand, overlap quark propagator can be computed by standard solver algorithms for hermitian or nonhermitian matrices. In HMC algorithm, since

[4] In Ref. [4], c_l is defined $\mathrm{sn}(lK'/2n;K')$ instead of $\mathrm{sn}(lK'/(2n+1);K')$ in Eq. (2.15).

one needs to invert only $D(m)^\dagger D(m)$, which is hermitian and positive definite, the standard conjugate gradient (CG) algorithm is applicable. Of course it is worth to examine which of hermitian algorithms and nonhermitian algorithms (BiCGStab, MR, GMRES, etc.) are efficient (see T. Kaneko's report [16, 18]). For the latter, one needs to solve the linear equations twice in HMC algorithm.

With the rational approximation to the sign function, the CG algorithm is also necessary to implement D_{ov}. In this sense, it is called 'nested' CG algorithm.

Relaxed CG algorithm. The relaxed stopping condition method [11] is based on an idea that, as the outer solver proceeds, the correction to the solution vector, $|x_i - x_{i-1}|$, becomes smaller and one does not have to evaluate $D(m)$ with too much accuracy. Its implementation depends on the outer solver algorithm, and for CG(NE), the condition is loosened as

$$\epsilon_i^{ms} \propto \sqrt{\zeta_i}, \quad \zeta_i = \zeta_{i-1} + \frac{1}{|r_{i-1}|^2}, \tag{3.1}$$

where $r_{i-1} = Dx_{i-1} - b$ and ϵ_i^{ms} the stopping condition for the inner (multishift) solver.

This technique accelerate the convergence measured in multiplication of D_W almost a factor of 2 [12, 17]. The relaxed CG algorithm is summarized as follows.

outer loop:

1: set initial guess x_0

2: $r_0 = b - Ax_0$

3: $p_0 = r_0$

4: $\zeta = 1/|r_0|^2$

5: repeat until $|r_k|/|b| < \epsilon_{out}$

- 5-1: inner loop: calculate q_k so that $|q_k - Ap_k| < \epsilon_{out}|b||p_k|\sqrt{\zeta}/k$
- 5-2: $\alpha = \langle r_k, r_k\rangle / \langle p_k, q_k\rangle$
- 5-3: $x_{k+1} = x_k + \alpha p_k$
- 5-4: $r_{k+1} = r_k - \alpha p_k$
- 5-5: convergence check: exit outer loop if $|r_k|/|k| < \epsilon_{out}$
- 5-6: $\beta = \langle r_{k+1}, r_{k+1}\rangle / \langle r_k, r_k\rangle$
- 5-7: $p_{k+1} = r_k + \beta p_k$
- 5-7: $\zeta = \zeta + 1/|r_{k+1}|^2$

Todo:
- Other outer solver algorithm.
- Adoptive precision.
- Guess of initial approximate solution (chronological estimator or preconditioning).
- Chiral projection.

There are still lots of things to try.

3.2 5-dimensional solver

The 5-dimensional CG solver is based on the Schur decomposition [14, 15]. Let us consider a 5-dimensional block matrix

$$M_5 = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ CA^{-1} & 1 \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} 1 & A^{-1}B \\ 0 & 1 \end{pmatrix} \equiv \tilde{L}\tilde{D}\tilde{U}, \tag{3.2}$$

where $S = D - CA^{-1}B$. S is called the Schur complement. Consider a linear equation

$$M_5 \begin{pmatrix} \phi \\ \psi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ \chi_4 \end{pmatrix}. \tag{3.3}$$

Using $M_5 = \tilde{L}\tilde{D}\tilde{U}$ and multiplying $\tilde{L}^{-1}$ from the left to the above equation, one arrives at

$$\begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} \phi + A^{-1}B\psi_4 \\ \psi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ \chi_4 \end{pmatrix}. \tag{3.4}$$

Namely, by solving 5-dimensional equation (3.3), one can solve

$$S\psi_4 = \chi_4 \tag{3.5}$$

Be an appropriate choice of parameters, as shown below, S can be set to the hermitian overlap operator $H_{ov} = \gamma_5 D_{of}$. Thus by solving Eq. (3.5) twice, one can solve $H_{ov}^2 x = D_{ov}^\dagger D_{ov} x = b$.

Hereafter an example for $N(= N_{pole}) = 2$ case is shown explicitly. Let us consider the 5D operator of a form

$$M_5 = \left(\begin{array}{cccc|c} H_W & -\sqrt{q_2} & & & 0 \\ -\sqrt{q_2} & -H_W & & & \sqrt{p_2} \\ & & H_W & -\sqrt{q_1} & 0 \\ & & -\sqrt{q_1} & -H_W & \sqrt{p_1} \\ \hline 0 & \sqrt{p_2} & 0 & \sqrt{p_1} & R\gamma_5 + p_0 H \end{array}\right) = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right). \tag{3.6}$$

In general, A B, and C has the following structure.

$$A = \begin{pmatrix} A_N & & \\ & A_{N-1} & \\ & & \ddots \end{pmatrix}, \quad B = \begin{pmatrix} B_N \\ B_{N-1} \\ \vdots \end{pmatrix}, \quad C = (C_N, C_{N-1}, \cdots) \tag{3.7}$$

$$A^{-1} = \begin{pmatrix} A_N^{-1} & & \\ & A_{N-1}^{-1} & \\ & & \ddots \end{pmatrix} \tag{3.8}$$

$$A_i^{-1} = \frac{1}{H_W^2 + q_i} \begin{pmatrix} H_W & -\sqrt{q_i} \\ -\sqrt{q_i} & -H_W \end{pmatrix} \qquad (i = 1, \ldots, n) \tag{3.9}$$

Then

$$S = D - CA^{-1}B = D - \sum_i C_i A_i^{-1} B_i \tag{3.10}$$

$$= R\gamma_5 + p_0 H_W + H_W \sum_{i=1}^{N} \frac{p_i}{H_W^2 + q_i}. \tag{3.11}$$

The parameters R, p_0, p_i and q_i $(i = 1, \ldots, N)$ are determined appropriately by comparing with the Zolotarev's partially fractional approximation formula (see below). This technique also applies to other partial fractional approximation and continuum fractional approximation [14, 15].

Low-mode subtraction [13]**.** The low-mode preconditioned hermitian overlap operator is written as

$$H_{ov} = f_1\gamma_5 + f_2 \sum_{j=1}^{N_{sbt}} \mathrm{sign}(\lambda_j) v_j \times v_j^\dagger + f_2 P_H \,\mathrm{sign}(H_W) P_H, \tag{3.12}$$

where $f_1 = M_0 + \frac{m}{2}$, $f_2 = M_0 - \frac{m}{2}$, v_j is an eigenvector of H_W associated to an eigenvalue λ_j, and $P_H = 1 - \sum_{j=1}^{N_{sbt}} v_j \times v_j^\dagger$ is the projector onto the space spanned by the eigenvectors whose eigenvalue $|\lambda| > \lambda_{thrs}$. This implies that the 5D operator is modified as

$$D = R\gamma_5 + p_0 H + f_2 \sum_{j=1}^{N_{sbt}} \mathrm{sign}(\lambda_j) v_j \times v_j^\dagger, \tag{3.13}$$

$$B_i = \begin{pmatrix} 0 \\ \sqrt{p_i} P_H \end{pmatrix}, \qquad C_i = (\, 0, \sqrt{p_i} P_H \,). \tag{3.14}$$

The parameters of the 5D matrix are determined as

$$R = f_1, \tag{3.15}$$

$$p_0 = f_2 \frac{d_0}{\lambda_{thrs}} \sum_l b_l, \tag{3.16}$$

$$p_l = f_2 d_0 b_l (c_{2n} - c_{2l-1}) \lambda_{thrs}, \tag{3.17}$$

$$q_l = c_{2l-1} \cdot \lambda_{thrs}^2. \tag{3.18}$$

Note that in the approximation of $\mathrm{sign}(H_W)$ in Eq. (3.12), λ_{thrs} replaces λ_{min} in Eq. (2.10).

Even-odd preconditioning. 5D solver is not efficient without applying a preconditioning technique. One can apply the even-odd preconditioning by decomposing 4D lattice sites into even and odd sites as

$$M_5 x = \begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} \begin{pmatrix} x_e \\ x_o \end{pmatrix} = \begin{pmatrix} b_e \\ b_o \end{pmatrix}. \tag{3.19}$$

By multiplying

$$\begin{pmatrix} M_{ee}^{-1} & -M_{ee}^{-1} M_{eo} M_{oo}^{-1} \\ 0 & M_{oo}^{-1} \end{pmatrix} \tag{3.20}$$

from the left, one has a closed equation for x_e,

$$(1 - M_{ee}^{-1} M_{eo} M_{oo}^{-1} M_{oe}) x_e = b_e' \equiv M_{ee}^{-1} b_e - M_{ee}^{-1} M_{eo} M_{oo}^{-1} b_o, \tag{3.21}$$

and after solving Eq. (3.21), x_o is provided as

$$x_o = b_o - M_{oo}^{-1} M_{oe} x_e. \tag{3.22}$$

The even-even and odd-odd block matrices must be inverted. The even-even matrix has the form

$$M_{ee} = \left(\begin{array}{cccc|c} \gamma_5 & -\sqrt{q_2} & & & 0 \\ -\sqrt{q_2} & -\gamma_5 & & & \sqrt{p_2}P_{Hee} \\ & & \gamma_5 & -\sqrt{q_1} & 0 \\ & & -\sqrt{q_1} & -\gamma_5 & \sqrt{p_1}P_{Hee} \\ \hline 0 & \sqrt{p_2}P_{Hee} & 0 & \sqrt{p_1}P_{Hee} & D_{ee} \end{array}\right), \tag{3.23}$$

$$D_{ee} = f_1\gamma_5 + f_2 \sum_j \mathrm{sign}\lambda_j v_{je} \times v_{je}^\dagger + p_0 P_{Hee}\gamma_5 P_{Hee}, \tag{3.24}$$

where v_{je} is even-part of the eigenvector v_j. M_{oo}, M_{eo}, and M_{oe} are similarly defined.

The matrix M_{ee} (and also M_{oo}) can be decomposed into left and right triangular matrices,

$$M_{ee} = L_e U_e = \left(\begin{array}{cccc|c} 1 & & & & \\ r_2 & 1 & & & \\ & & 1 & & \\ & & r_1 & 1 & \\ \hline 0 & s_2 & 0 & s_1 & 1 \end{array}\right) \left(\begin{array}{cccc|c} \gamma_5 & -\sqrt{q_2} & & & \\ & v_2 & & & \sqrt{p_2}P_{Hee} \\ & & \gamma_5 & -\sqrt{q_1} & \\ & & & v_1 & \sqrt{p_1}P_{Hee} \\ \hline & & & & u_0 \end{array}\right) \tag{3.25}$$

where

$$r_l = -\sqrt{q_l}\gamma_5 \tag{3.26}$$

$$v_l = -(1+q_l)\gamma_5 \tag{3.27}$$

$$s_l = -\frac{\sqrt{p_l}}{1+q_l} P_{Hee}\gamma_5 \tag{3.28}$$

$$u_0 = D_{ee} + \left(\sum_l \frac{p_l}{1+q_l}\right) P_{Hee}\gamma_5 P_{Hee} \tag{3.29}$$

Multiplication of U^{-1} and L^{-1} is easily implemented by forward and backward substitution. However, one need to solve $x = u_0^{-1}z$ for a given 4D even-vector z at each iteration step of the solver algorithm for Eq. (3.21).

Inversion of u_0. The standard method to solve a linear equation is the iterative Krylov subspace method. The Krylov subspace is, starting with initial residual vector $r_0 = z$, composed as $\mathcal{K}^k(A; z) = \mathrm{span}\{z, Az, \ldots, A^{k-1}z\}$. Because of the structure of the matrix u_0, the Krylov subspace closes at most at $2(N_{sbt}+1)$ dimension, and then is spanned by a non-orthogonal basis $\{z, \gamma_5 z, v_{je}, \gamma_5 v_{je}\}$ $(j = 1, \ldots N_{sbt})$. Thus $u_0^{-1}z$ is also expanded in this basis.

Denoting $w_j = v_{je}$, $w_{N_{sbt}+j} = \gamma_5 v_{je}$ $(j = 1, \ldots, N_{sbt})$, u_0 is expressed as

$$u_0 = \left(a + \sum_{i,j=1}^{2N_{sbt}} c_{i,j} w_j \times w_j^\dagger\right)\gamma_5. \tag{3.30}$$

The coefficients are given as $a = f_1 + p_0 + u_H$,

$$c_{i,j} = u_H\delta_{i,j} \equiv \left(\sum_l \frac{p_l}{1+q_l}\right)\delta_{i,j} \tag{3.31}$$

$$c_{i+N_{sbt},j+N_{sbt}} = u_H\delta_{i,j} \tag{3.32}$$

$$c_{i,j+N_{sbt}} = [f_2\mathrm{sign}(\lambda_i) - p_0\lambda_i]\delta_{i,j} + u_H(v_{je}^\dagger\gamma_5 v_{je}). \tag{3.33}$$

		5D solver		4D solver	
m_q	N_{poly}	time [sec]	D_W mult [k]	time [sec]	D_W mult [k]
0.400	10	16.7	27	41.3	119
0.100	10	41.0	65	161.2	467
0.050	10	75.5	119	322.2	932
0.035	10	104.7	165	458.8	1328
0.025	10	141.2	222	621.5	1801
0.015	10	203.8	321	787.9	2281
0.400	20	28.7	56	55.4	125
0.100	20	65.2	127	216.2	489
0.050	20	121.4	236	431.2	976
0.035	20	168.8	329	614.3	1,390
0.025	20	228.0	444	832.1	1,883
0.015	20	330.1	642	1050.0	2,377

Table 1: The convergence of 5D and 4D solvers on Blue Gene 1024-node class. The convergence criterion is $|r|^2/|b|^2 < 10^{-20}$. Note that the $|b - (D^\dagger D)x|^2$ of the 5D solver is of one order of magnitude larger that of the 4D solver.

for $i, j = 1, \ldots, N_{sbt}$. u_0 is also be expanded as

$$u_0^{-1} = \gamma_5 \left(\bar{a} + \bar{a}' \gamma_5 + \sum_{i,j=1}^{2N_{sbt}} \bar{c}_{i,j} w_j \times w_j^\dagger \right). \tag{3.34}$$

From the condition $u_0 u_0^{-1} = 1$, one easily obtains that $\bar{a} = 1/a$, $\bar{a}' = 0$, and $\bar{c}_{i,j}$ is determined by solving a linear equation

$$\sum_l \left[a\delta_{il} + \sum_k c_{ik} (w_k^\dagger w_l) \right] \bar{c}_{lj} = -\frac{1}{a} c_{ij}. \tag{3.35}$$

This inversion must be done only once before starting the 5D solver. In each step of iteration, required operation is the $2N_{sbt}$ inner products $(w_j^\dagger b)$ and vector operations related to w_j. This is not a heavy operation compared to other parts of the 5D solver.

4 Performance test

In this section, we compare the performance of the 4D and 5D solvers. A comparison is performed on a single configuration of $16^3 \times 48$ lattice generated at $\beta = 2.30$, $N_f = 2 + 1$ with $m_{ud} = 0.100$, $m_s = 0.100$, and at $Q = 0$. The result on Blue Gene 1024-node job class is shown in Table 1 as well as in Fig. 1 for $m_q \leq 0.1$. The convergence time and numbers of D_W mult were observed for various quark masses and $N_{poly} = 10$ and 20. The former N_{poly} corresponding the value used in the productive run. The convergence criterion is $|r|^2/|b|^2 < 10^{-20}$. Note that the $|b - (D^\dagger D)x|^2$ of the 5D solver is of one order of magnitude larger than that of the 4D solver. The threshold of eigenvalues for low-mode subtraction is set as $\lambda_{thrs} = 0.045$, which leads on this configuration $N_{sbt} = 8$.

There are specific features for each of these two algorithms.

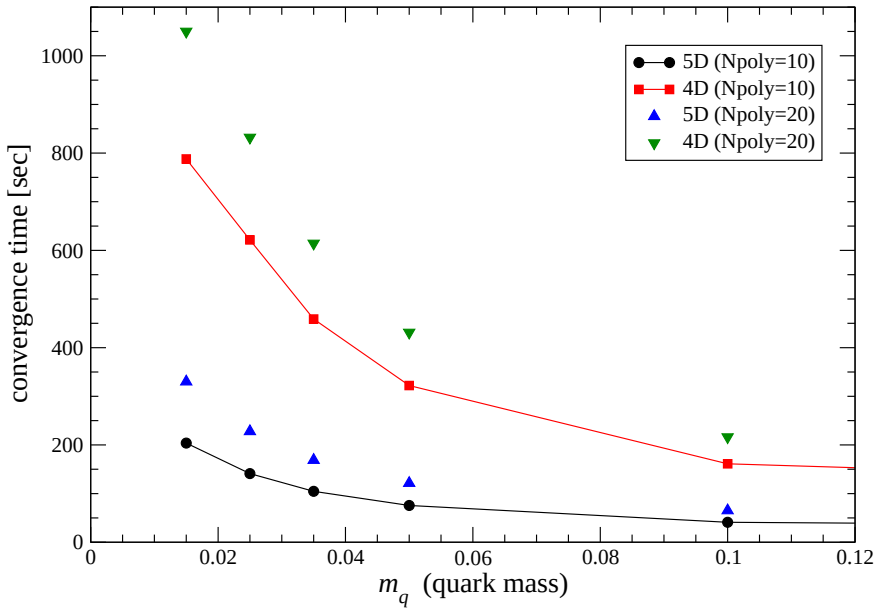

Figure 1: The quark mass dependence of the convergence time of 5D and 4D solvers on Blue Gene 1024-node class. See also the comments of Table 1.

- Increasing the degree of polynomial, N_{poly}, Relaxed CG increases the cost only gradually, because the multishift CG method is applied for the inner loop. In 5D solver, since the size of vector is almost proportional to N_{poly}, the numerical cost increases linearly. (However, these features are not apparent in the above performance test. This may depend also on the machine architecture.)

- By relaxed CG, several masses of overlap fermion can be solver simultaneously by making use of multishift CG for outer loop. The 5D solver can be applied to single quark mass mass at once. Thus we adopt the relaxed CG to the spectroscopy (partially quenched), in which the cost can be largely reduced by solving quark propagator for several masses simultaneously. On the other hand, in HMC the quark mass is fixed to a single value, and hence the 5D solver is employed.

5 Additional topics on the 5D solver

Approximate solution. Suppose that one has an approximate solution $\tilde{\psi}_4$ of a 4D equation,

$$D_{ov}\psi_4 = \chi_4. \tag{5.36}$$

If one can also compose an approximate solution of 5D equation from $\tilde{\psi}_4$, it can provide a good initial guess for the 5D linear problem.

Because of Eq. (3.4), when ψ_4 is already known, ϕ is given by solving

$$A\phi = -B\psi_4. \tag{5.37}$$

More explicitly,

$$\phi = \begin{pmatrix} \phi_N \\ \phi_{N-1} \\ \vdots \end{pmatrix}, \quad \phi_i = \begin{pmatrix} \sqrt{q_i} \\ H_W \end{pmatrix} \frac{\sqrt{p_i}}{H_W^2 + q_i} \psi_4, \tag{5.38}$$

which is easily determined simultaneously by using the multishift CG solver.

There are two straightforward applications:

- Adaptive 5D solver
  At early stage of the CG iteration, one does not require the full precision to D_{ov}. The above method enables us to change the value of N_{poly} at an intermediate stage of CG iteration. Namely, the smaller N_{poly} is used at the earlier CG iterations. The precisions at which N_{poly} is changed are selected in accord with the precisions of the approximation to the sign function with that N_{poly}. This idea was examined in a report [19] on a N_f =2+1 configuration with $m_q = 0.015$ and $m_s = 0.100$. In summary, the adoptive 5D solver achieves about 15% acceleration for the dynamical quark, while almost no improvement for the preconditioner. Present simulation code does not include this technique.

- Chronological estimator in HMC update.
  The chronological estimator technique compose an initial guess of linear equation by previous solutions. To apply it to the 5D solver, one needs to store large 5D solutions. The above method reduces the size of stored vectors from $(2N_{poly} + 1) \times N_q$ to N_q, where $N_q = 2 \cdot N_{color} \cdot N_{spinor} \cdot N_{site}$

Hermitian version of 5D solver. In the implementation in Subsec. 3.2, the even-odd preconditioned matrix in Eq. 3.21 was not hermitian. Therefore the CGNE algorithm is employed to solve it. However, the hermitian (while not positive definite) version of the even-odd preconditioned matrix can also be constructed [20]. Then the algorithms for hermitian matrices, *e.g.* MINRES, are applicable. Unfortunately, this implementation does not improve the cost [20].

6 Conclusion and outlooks

In this note, we described the linear problem for the overlap Dirac operator on the lattice. Two kinds of algorithms, the nested CG (or 4D) solver and 5D solver are described in detail. At present, the latter is faster than the former. These two algorithms seem to still have room for further improvement.

Another challenge concerning the overlap operator is to develop good algorithms for recent architectures such as GPGPU and Cell Broadband Engine. In some cases, making use of the single precision arithmetic units will help to accelerate the solver, while double precision is required for final results. In practice, the severest bottleneck may be the inter-node communication for which the bandwidth of a present architecture is not sufficient for naive implementation. Algorithms with less communication are essential, such as the multi-grid and domain decomposition algorithms.

A Multishift CG algorithm

Multi-shift solver [7, 8] enables to solve these n equations at once. We employ the multi-shift CG algorithm. The multi-shift solver algorithms make use of the fact that the Krylov subspace $\mathcal{K}_k(A, v_0)$ for matrix A and initial vector v_0 is unchanged against a shift $A \to A + \sigma$, *i.e.*, $\mathcal{K}_k(A, v_0) = \mathcal{K}_k(A + \sigma, v_0)$. This implies that the residual vector, which is perpendicular to the previous Klylov subspace, can be shared by n equations in Eq. (2.16). Then n solutions of Eq. (2.16) can be determined simultaneously. The algorithm is written as follows.

Multi-shift CG algorithm:

(i) initial step

$$\begin{array}{ll} x_0 = 0 & x_0^\sigma = 0 \\ r_0 = p_0 = b & p_0^\sigma = b \\ & \zeta_0^\sigma = \zeta_{-1}^\sigma = 1 \end{array}$$

(ii) iteration step

for $i = 0, 1, 2, \ldots$ (repeat until convergence)

$$\begin{array}{ll} \beta_i = -\frac{(r_i, r_i)}{(p_i, Ap_i)} & \\ x_{i+1} = x_i - \beta_i p_i & \\ r_{i+1} = r_i + \beta_i A p_i & \\ \alpha_i = \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)} & \\ p_{i+1} = r_{i+1} + \alpha_i p_i & \\ & \hat{\alpha}_i = 1 + \frac{\alpha_{i-1}\beta_i}{\beta_{i-1}} \\ & \zeta_{i+1}^\sigma = \left[(\hat{\alpha}_i - \sigma\beta_i)/\zeta_i^\sigma \ + \ (1 - \hat{\alpha}_i)/\zeta_{i-1}^\sigma\right]^{-1} \\ & \beta_i^\sigma = \frac{\zeta_{i+1}^\sigma}{\zeta_i^\sigma}\beta_i \\ & \alpha_i^\sigma = \left(\frac{\zeta_{i+1}^\sigma}{\zeta_i^\sigma}\right)\alpha_i \\ & x_{i+1}^\sigma = x_i^\sigma - \beta_i^\sigma p_i^\sigma \\ & p_{i+1}^\sigma = \zeta_{i+1}^\sigma r_{i+1} + \alpha_i^\sigma p_i^\sigma \end{array}$$

The operations in the right column are for the shifted equations.

Implementation: The solutions of Eq. (2.16) for larger c_j converges faster. For the convergence criterion, the squared norm of the vector p^σ is monitored and the iteration is stopped when it become less than the given criterion.

References

[1] H. Neuberger, "Exactly massless quarks on the lattice," Phys. Lett. B **417** (1998) 141 [arXiv:hep-lat/9707022];
"More about exactly massless quarks on the lattice," Phys. Lett. B **427** (1998) 353 [arXiv:hep-lat/9801031].

[2] ARPACK – Arnordi Package –, `http://www.caam.rice.edu/software/ARPACK/`

[3] T. Kalkreuter and H. Simma, "An Accelerated conjugate gradient algorithm to compute low lying eigenvalues: A Study for the Dirac operator in SU(2) lattice QCD," Comput. Phys. Commun. **93**, 33 (1996) [arXiv:hep-lat/9507023].

[4] J. van den Eshof, A. Frommer, T. Lippert, K. Schilling and H. A. van der Vorst, "Numerical methods for the QCD overlap operator. I: Sign-function and error bounds," Comput. Phys. Commun. **146**, 203 (2002) [arXiv:hep-lat/0202025].

[5] T. W. Chiu, T. H. Hsieh, C. H. Huang and T. R. Huang, "A note on the Zolotarev optimal rational approximation for the overlap Dirac operator," Phys. Rev. D **66**, 114502 (2002) [arXiv:hep-lat/0206007].

[6] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *NUMERICAL RECIPES in FORTRAN, 2nd ed.* (Cambridge Univ. Press, 1986,1992).

[7] A. Frommer, B. Nockel, S. Gusken, T. Lippert and K. Schilling, "Many masses on one stroke: Economic computation of quark propagators," Int. J. Mod. Phys. C **6**, 627 (1995) [arXiv:hep-lat/9504020].

[8] B. Jegerlehner, "Krylov space solvers for shifted linear systems," arXiv:hep-lat/9612014.

[9] H. Neuberger, "Minimizing storage in implementations of the overlap lattice-Dirac operator," Int. J. Mod. Phys. C **10**, 1051 (1999) [arXiv:hep-lat/9811019].

[10] T. W. Chiu and T. H. Hsieh, "A note on Neuberger's double pass algorithm," Phys. Rev. E **68**, 066704 (2003) [arXiv:hep-lat/0306025].

[11] N. Cundy, J. van den Eshof, A. Frommer, S. Krieg, T. Lippert and K. Schafer, "Numerical methods for the QCD overlap operator. III: Nested iterations," Comput. Phys. Commun. **165** (2005) 221 [arXiv:hep-lat/0405003].

[12] S. Aoki *et al.* [JLQCD Collaboration], "Two-flavor QCD simulation with exact chiral symmetry," Phys. Rev. D **78** (2008) 014508 [arXiv:0803.3197 [hep-lat]].

[13] S. Hashimoto *et al.* [JLQCD collaboration], "Lattice simulation of 2+1 flavors of overlap light quarks," PoS **LAT2007** (2007) 101 [arXiv:0710.2730 [hep-lat]].

[14] A. Borici, "Computational methods for the fermion determinant and the link between overlap and domain wall fermions," arXiv:hep-lat/0402035.

[15] R. G. Edwards, B. Joo, A. D. Kennedy, K. Orginos and U. Wenger, "Comparison of chiral fermion methods," PoS **LAT2005** (2006) 146 [arXiv:hep-lat/0510086].

[16] T. Kaneko, "GMRES(m) for overlap solver", JLQCD internal report on 8 Dec 2005.

[17] T. Kaneko, "Relaxed overlap solver", JLQCD internal report on 19 Jan 2006.

[18] T. Kaneko, "overlap solvers", JLQCD internal report on 9 Feb 2006.

[19] H. Matsufuru, "Adaptive 5D solver", JLQCD internal report on 12 Dec 2007.

[20] H. Matsufuru, "5D solver with hermitian even-odd preconditioning", JLQCD internal report on 14 Nov 2007.