Cuda Lattice Gauge Document

Ji-Chong Yang

目录

1 Data **5**

1.1 Index of lattice . . . 5

1.1.1 UINT Index of lattice . . . 5

1.1.2 SIndex of lattice . . . 5

1.1.3 Index and boundary condition, a int2 or a uint2 structure . . . 5

1.1.4 Index walking . . . 6

1.2 CParemeters . . . 6

2 Update scheme **7**

2.1 HMC . . . 7

2.1.1 The Fermion action . . . 7

2.1.2 Basic idea, force from gauge field . . . 8

2.1.3 Force of pseudofermions . . . 11

2.1.4 Solver in HMC . . . 14

2.1.5 Leap frog integrator . . . 16

2.1.6 A summary of HMC with pseudofermions . . . 16

2.2 Optimization of HMC . . . 17

2.2.1 Omelyan integrator . . . 17

2.2.2 Omelyan force-gradient integrator . . . 18

2.2.3 Multi-rate integrator (nested integrator) . . . 19

2.2.4 Cached solution . . . 20

2.3 Staggered Fermion . . . 20

2.3.1 The D=1 staggered fermion and Jordan-Wigner transformation . . . 20

2.3.2 The relationship between the naive fermion and staggered fermion . . . 21

2.3.3 Symmetries of the staggered fermions . . . 22

2.4 The RHMC for staggered fermion . . . 23

2.4.1 The rational approximation and Remes algorithm . . . 23

2.4.2 The RHMC with $N_f = 2$ . . . 24

3 Sparse linear algebra solver **28**

3.1 Krylov subspace . . . 28

3.2 GMRES . . . 28

3.3 GCR . . . 32
3.4 TFQMR . . . 34
3.5 GCRO-DR and GMRES-MDR . . . 35
3.5.1 Brief introduction to deflation preconditioner . . . 36
3.5.2 Brief intro to GCRO-DR . . . 37
3.5.3 The choice of deflation subspace . . . 38
3.5.4 Eigen solver . . . 39
3.5.5 Implementation of GCRO-DR . . . 45
3.5.6 Implement of GCRO-DR . . . 46
3.5.7 Implement of GMRES-MDR . . . 48
3.5.8 Test of GCRO-DR and GMRES-MDR . . . 48
3.6 Even-odd preconditioner . . . 48
3.7 The multi-shift solver . . . 50
3.7.1 The multi-shift version of GMRES . . . 50
3.7.2 The multi-shift FOM . . . 51
3.7.3 The multi-shift BiCGStab . . . 52

4 Miscellaneous topics 55
4.1 Gauge Fixing . . . 55
4.1.1 Introduction of FFT before start . . . 55
4.1.2 Cornell Gauge Fixing and FFT accelerated . . . 56
4.1.3 Los Alamos Gauge Fixing and over relaxation . . . 57
4.1.4 Coulomb Gauge . . . 59
4.1.5 Logarithm definition . . . 59

5 Measurement 61
5.1 Plaquette Energy . . . 61
5.2 Meson Correlator . . . 61
5.2.1 Meson Wave Function . . . 61
5.2.2 Meson Correlator . . . 61
5.2.3 Sources . . . 63
5.2.4 Summary of parameters . . . 66
5.2.5 Gauge smearing . . . 67
5.3 Chiral Condensate of Wilson fermion . . . 68

5.3.1 Wall source 68
5.3.2 Decay constant 68
5.3.3 Effective Quark Mass 68
5.4 Stochastic Methods 68
5.4.1 For condensations 68
5.4.2 For densities 69

6 Programming 71
6.1 cuda 71
6.1.1 blocks and threads 71
6.1.2 device member function 71
6.1.3 device virtual member function 74

7 Testing 77
7.1 random number 77

8 Applications 78
8.1 Rotating Frame 78
8.1.1 The rotating gauge action 79
8.1.2 Rotating Fermion action 81
8.1.3 The exponential chemical potential 85
8.1.4 The final action of rotation 87
8.1.5 The force from gauge action 88
8.1.6 The force from fermion action 93
8.1.7 The angular momentum 94
8.1.8 The Current density and Charge density 102
8.1.9 The Topological Density 102
8.1.10 The Polyakov loop 103
8.1.11 The Chiral Condensate 103
8.2 Sample Producer 104
8.3 Data Analyse 104
8.3.1 What is autocorrelation 104
8.3.2 How to calculate autocorrelation, and how to use it to obtain the interval 105

1 Data

1.1 Index of lattice

1.1.1 UINT Index of lattice

Generally, in CLG, we have three kinds of indexes:

- **site index**
- **link index**
- **fat index**

Let the lattice have $V = L_x \times L_y \times L_z \times L_t$ sites.

Note: for $D = 3$, we assume $L_x = 1$, $L_{y,z,t} > 1$; for $D = 2$, we assume $L_x = L_y = 1$, $L_{z,t} > 1$.

For a site at (x, y, z, t)

$$siteIndex = x \times L_y \times L_z \times L_t + y \times L_z \times L_t + z \times L_t + t \tag{1}$$

For a link at direction dir, link with site at (x, y, z, t), and on a lattice with number of directions of links is $dirCount$,

$$linkIndex = siteIndex \times dirCount + dir \tag{2}$$

Note: we do NOT assume dimension equal number of links. For example for $D = 2$ triangle lattice, number of directions of links is 6, for $D = 2$ hexagon number of directions of links is 3. Only for square lattice, number of links equal dimension.

For a link at direction dir, link with site at (x, y, z, t), and on a lattice with number of directions of links is $dirCount$,

$$fatIndex = \begin{cases} siteIndex \times (dirCount + 1); & for\ site. \\ siteIndex \times (dirCount + 1) + (dir + 1); & for\ link \end{cases} \tag{3}$$

1.1.2 SIndex of lattice

1.1.3 Index and boundary condition, a int2 or a uint2 structure

In CLGLib, sometimes, the index function return a uint2 structure.

1.1.4 Index walking

1.2 CParemeters

2 Update scheme

2.1 HMC

HMC is abbreviation for hybrid Monte Carlo.

2.1.1 The Fermion action

Cooperating with HMC, the fermion is usually the 'Pseudofermions'.

We begin with Eq. (1.85) and Eq. (1.86) of Ref. [1].

$$Z=\int\mathcal{D}[U]\prod_{f=1}^{N_f}\mathcal{D}[\bar{\psi}_f]\mathcal{D}[\psi_f]\exp\left(-S_G[U]-\sum_{f=1}^{N_f}\bar{\psi}_f\left(\hat{D}_f\right)\psi_f\right)\tag{4}$$

where $\hat{D}_f=D+m_f$. (Note that, there seems a typo in Eq. (1.85) of Ref. [1] and Eq. (8.31) of Ref. [2], we have $S_F=+\bar{\psi}D\psi$, see also Eqs. (2.5) and (8.39) of Ref. [2] and Eq. (7.6) of Ref. [3], Eq. (3.75) of Ref. [1], etc.)

It can be evaluated as Eq. (1.86) of Ref. [1] (or Eq. (4.19) of Ref. [4]) (Note, there is another minus sign in Eq. (5.28) of Ref. [2])

$$\begin{aligned}&\int\mathcal{D}\bar{\psi}\psi\exp\left(-\bar{\psi}A\psi\right)=\det\left(A\right),\\&Z=\prod_{f=1}^{N_f}\det\left(\hat{D}_f\right)\int\mathcal{D}[U]\exp\left(-S_G[U]\right).\end{aligned}\tag{5}$$

On the other hand, with the help of Gaussian integral of complex vectors Eq. (3.17) of Ref. [4]

$$\int d\mathbf{v}^{\dagger}d\mathbf{v}\exp(-\mathbf{v}^{\dagger}\mathbf{A}\mathbf{v})=\pi^N\left(\det\mathbf{A}\right)^{-1}\tag{6}$$

which is (3.31) of Ref. [1]

$$\frac{1}{\det(\mathbf{A})}=\int\mathcal{D}[\eta]\exp(-\eta^{\dagger}\mathbf{A}\eta)\tag{7}$$

where η now is a complex Bosonic field, and the normalization

$$\mathcal{D}[\eta]=\prod\frac{d\mathrm{Re}(\eta_i)d\mathrm{Im}(\eta_i)}{\pi},\;\;1=\int\mathcal{D}[\eta]\exp(-\eta^{\dagger}\eta)\tag{8}$$

is assumed. With the condition such that

$$\lambda(\mathbf{A}+\mathbf{A}^{\dagger})>0.\tag{9}$$

where $\lambda(\mathbf{M})$ denoted as eigen-values of $\mathbf{M}$.

We now, concentrate on two degenerate fermion flavours. i.e. considering

$$S_F = \bar{\psi}_u \hat{D} \psi_u + \bar{\psi}_d \hat{D} \psi_d. \tag{10}$$

Using $\det(DD^\dagger) = \det(D)\det(D^\dagger)$ and $\det(M^{-1}) = (\det(M))^{-1}$ and $\det(D) = \det(D^\dagger)$ (**Only for Wilson Fermions or γ_5-hermiticity fermions, $\hat{D}^\dagger = \gamma_5 D \gamma_5 + m = \gamma_5(D+m)\gamma_5 = \gamma_5 \hat{D} \gamma_5$, and $\det(\hat{D}^\dagger) = \det(\gamma_5)\det(\hat{D})\det(\gamma_5) = \det(\hat{D})$.** See also Ref. [5].), one can show Eq. (8.9) of Ref. [2] (Eq. (2.77) of Ref. [6])

$$\int \mathcal{D}[\bar{\psi}]\mathcal{D}[\psi] \exp\left(-\bar{\psi}_u \hat{D} \psi_u - \bar{\psi}_d \hat{D} \psi_d\right) = \det(\hat{D}\hat{D}^\dagger) = \int \mathcal{D}[\phi] \exp(-\phi^\dagger \left(\hat{D}\hat{D}^\dagger\right)^{-1} \phi) \tag{11}$$

where ϕ now is a complex Bosnic field. (Note that, there is a sign typo in Eq. (8.31) of Ref. [2], see also Eqs. (8.38) and (8.39) of Ref. [2])

So, generally, we are using HMC to evaluate the action with 'Pseudofermions', or in other words, we are working with an action including only gauge and bosons.

$$S = S_G + S_{pf} = S_G + \phi^\dagger \left(\hat{D}\hat{D}^\dagger\right)^{-1} \phi \tag{12}$$

where pf is short for pseudofermion.

2.1.2 Basic idea, force from gauge field

The basic idea is to use a molecular dynamics simulation, i.e, it is a integration of Langevin equation.

Treating $SU(N)$ matrix U on links as coordinate, HMC will generate a pair of configurations, (P, U), where P is momentum and $P \in \mathfrak{su}(N)$.

One can:

1. Create a random $P = i\sum_a \omega_a T_a$, where $\omega_a \in \mathbb{R}$.
2. Obtain $\dot{P}$, $\dot{U}$. Note that, dot is $d/d\tau$, where τ is 'Markov time'.
3. Numerically evaluate the differential equation, and use a Metropolis accept / reject to update.

- About the randomized P

The randomized P is chosen according to normal distribution $\exp\left(-P^2/2\right)$

Note that, here P corresponds to Q, not U, for $U = \exp\left(i\sum q_a T^a\right)$, there are 8 **real** variables denoting as ω_i.

Using $P = \sum \omega_a T^a$, $tr((T^a)\cdot(T^b)) = \frac{1}{2}\delta_{ab}$. So one have $\frac{1}{2}\sum_a \omega_a^2 = tr[P^2]$.

It is usually written as distribution $\exp\left(-tr(P^2)\right)$ (where P is a matrix, and $tr[P^2] = \frac{1}{2}p^2$ where $p = (\omega_1, \omega_2, \ldots, \omega_8)$).

Using the property of normal distribution

$$\begin{aligned} &if\ \{X\} \sim N(\mu_X, \sigma_X^2),\ \ \{Y\} \sim N(\mu_Y, \sigma_Y^2),\\ &\{X+Y\} \sim N(\mu_X+\mu_Y, \sigma_X^2+\sigma_Y^2). \end{aligned} \tag{13}$$

One can randomize ω_a using $\exp\left(-\omega_a\omega_a\right)$. Then using $P = \frac{1}{\sqrt{8N}}\sum \omega_a T^i$, where N is the number of links.

Note: Here is a difference between Refs. [2] and [1] and Bridge++ [7]

Note, by Eq. (8.16) of Ref. [2], $P^2 = \sum_{n\in\Lambda} P^2(n)$, so when the lattice is large, P become very small. See also the definition of $\langle P, P\rangle$ below Eq. (2.42) of Ref. [1].

However, in Bridge++, it uses distribution $\exp\left(-tr(P^2)/DOF\right)$, where 'DOF' is the degrees of freedom, i.e., number of links.

We use the distribution same as in Bridge++. Imagining that for a very small (hot) $\beta \to 0$, the force is also almost 0 so momentum is unchanged when evolution. Considering a very large lattice such that the momentum is very small when using distribution $\exp\left(-tr(P^2)\right)$, the gauge field will stay near the initial value rather then becoming hot (randomized). So we think it should be $\exp\left(-tr(P^2)/DOF\right)$.

- Force

Defined by Newton, dp/dt is a force, so $\dot{P}$ is called 'force'. See Eqs. (2.53), (2.56) and (2.57) of Ref. [1], for $SU(N)$,

$$\begin{aligned} &S_G[U_\mu(n)] = -\frac{\beta}{N}\mathrm{Retr}\left[U_\mu(n)\Sigma_\mu^\dagger(n)\right]\\ &\Sigma_\mu(n) = \sum_{\mu\neq\nu}\left(U_\nu(n)U_\mu(n+a\nu)U_\nu^{-1}(n+a\mu) + U_\nu^{-1}(n-a\nu)U_\mu(n-a\nu)U_\nu(n-a\nu+a\mu)\right) \end{aligned} \tag{14}$$

Note that $S_G \neq \sum_{\mu,n} S_G[U_\mu(n)]$. $S_G[U_\mu(n)]$ is convenient for derivate which collecting all terms related to the specified bond. For plaquettes with 4 edges, $S_G = \frac{1}{4}\sum_{\mu,n} S_G[U_\mu(n)]$.

S_G the action for a particular $U_\mu(n)$. Σ is the 'staple'(see Eq. (152)). The staple for $U_\mu(n)$ is independent of $U_\mu(n)$, denoting

$$U_\mu(n) = \exp\left(i\sum_a \omega_a(\mu,n)T_a\right)U_\mu^0(n) \tag{15}$$

so

$$\begin{aligned}
&\frac{\partial}{\partial\omega_a(\mu,n)}S_G = -\frac{\beta}{N}\mathrm{Retr}\left[\frac{\partial}{\partial\omega_a}U_\mu(n)\Sigma_\mu^\dagger(n)\right] = -\frac{\beta}{2N}\mathrm{tr}\left[\frac{\partial}{\partial\omega_a}\left(U_\mu(n)\Sigma_\mu^\dagger(n)+\Sigma_\mu(n)U_\mu^\dagger(n)\right)\right]\\
&= -i\frac{\beta}{2N}\mathrm{tr}\left[T_aU_\mu(n)\Sigma_\mu^\dagger(n)-\Sigma_\mu(n)T_a^\dagger U_\mu^\dagger(n)\right] = -i\frac{\beta}{2N}\mathrm{tr}\left[T_a\left(U_\mu(n)\Sigma_\mu^\dagger(n)-\Sigma_\mu(n)U_\mu^\dagger(n)\right)\right]\\
&= \frac{\beta}{N}\mathrm{Im\ tr}\left[T_aU_\mu(n)\Sigma_\mu^\dagger(n)\right]
\end{aligned} \tag{16}$$

This is the Eq. (8.41) of Ref. [2].

Using (Checked by Mathematica that Eq. (8.42) of Ref. [2] is incompatable with our notation, but replacing the $UA - A^\dagger U^\dagger$ of Eq. (8.42) with $\{UA\}_{TA}$ is correct. Also, Eq. (2.58) of Ref. [1] is different from ours, in our formulism, it is correct by replacing $2T_a\mathrm{Re}[tr[T_a\cdot W]]$ of Eq. (2.58) with $2iT_a\mathrm{Im}[tr[T_a\cdot W]]$)

$$\begin{aligned}
&\sum_a \mathrm{tr}\left[T_a\left(U_\mu(n)\Sigma_\mu^\dagger(n)-\Sigma_\mu(n)U_\mu^\dagger(n)\right)\right]T_a = 2i\sum_a \mathrm{Im}\left[T_aU_\mu(n)\Sigma_\mu^\dagger(n)\right]T_a = \{U_\mu(n)\Sigma_\mu^\dagger(n)\}_{TA}\\
&\{W\}_{TA} = \frac{W-W^\dagger}{2} - \mathrm{tr}\left(\frac{W-W^\dagger}{2N}\right)\mathbb{I}
\end{aligned} \tag{17}$$

where $\mathbb{I}$ is identity matrix. Therefor

$$\begin{aligned}
&\dot{\omega}_a = -\frac{\partial}{\partial\omega_a(\mu,n)}S_G\\
&F_\mu(x) = \dot{P}_\mu(x) = i\sum\dot{\omega}_aT_a = -i\frac{\partial}{\partial\omega_a(\mu,n)}S_GT_a = -\frac{\beta}{2N}\{U_\mu(n)\Sigma_\mu^\dagger(n)\}_{TA}
\end{aligned} \tag{18}$$

Note that, $\dot{\omega}_a = \frac{\beta}{N}\mathrm{Im}[tr[T_a\cdot W]]$ is still a **real** number.

Eq. (18) is same as Eqs. (2.53), (2.56) and (2.57) of Ref. [1].

- Integrator

Knowing $\dot{P}$, and $\dot{U}$, to obtain U and P is simply

$$U(\tau+d\tau)\approx\dot{U}d\tau+U(\tau),\quad P(\tau+d\tau)\approx\dot{P}d\tau+P(\tau) \tag{19}$$

A more accurate calculation is done by integrator, for example, the leap frog integrator, the M step leap frog integral is described in Ref. [2],

$$\epsilon = \tfrac{\tau}{M} \tag{20a}$$

$$U_\mu(x,(n+1)\epsilon) = U_\mu(x,n\epsilon) + \epsilon P_\mu(x,n\epsilon) + \tfrac{1}{2}F_\mu(x,n\epsilon)\epsilon^2 \tag{20b}$$

$$P_\mu(x,(n+1)\epsilon) = P_\mu(x,n\epsilon) + \tfrac{1}{2}\left(F_\mu(x,(n+1)\epsilon) + F_\mu(x,n\epsilon)\right)\epsilon \tag{20c}$$

So, knowing $U(n\epsilon)$ we can calculate $F(n\epsilon)$ using Eq. (18). Knowing $U(n\epsilon), P(n\epsilon), F(n\epsilon)$, we can calculate $U((n+1)\epsilon)$ using Eq. (20).b. Then we are able to calculate $F((n+1)\epsilon)$ again using Eq. (18). Then we can calculate $P((n+1)\epsilon)$ using Eq. (20).c.

2.1.3 Force of pseudofermions

For important sampling, one can generate both U and ϕ by e^{-S}. In molecular dynamics simulation, it can be simplified as:

1. Evaluate U use force of U and ϕ on U.
2. Evaluate ϕ use force of U and ϕ on ϕ.

The second step can be simplified as, generating random complex numbers ϕ according to $\exp(-\phi^\dagger\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi) = \exp(-\phi^\dagger(\hat{D}^\dagger)^{-1}\hat{D}^{-1}\phi)$. $D[U]$ is a function of U.

How to get randomized ϕ? Let χ be random **complex** numbers according to $\exp(-\chi^\dagger\chi)$. Let $\hat{D}^{-1}\phi = \chi$, ϕ is the random **complex** number satisfying distribution we want $(\exp(-\phi^\dagger(\hat{D}^\dagger)^{-1}\hat{D}^{-1}\phi))$. So, first get χ and then let $\phi = D\chi$.

Using the Wilson Fermion action

$$\begin{aligned}&\hat{D} = C(D+1)\\ &D = -\kappa\sum_\mu\left((1-\gamma_\mu)U_\mu(x_L)\delta_{x_L,(x+\mu)_R} + (1+\gamma_\mu)U_\mu^{-1}(x_L-\mu)\delta_{x_L,(x-\mu)_R}\right)\end{aligned} \tag{21}$$

with $C = m_f + (4/a) = 1/2a\kappa$ and $\kappa = 1/(2am_f+8)$. One can rescale the field and set $C = 1$.

The force of ϕ on U is obtained as $\partial_{\omega_a}S_{pf}$. The result for Wilson Fermion action is shown in Eqs. (8.39), (8.44) and (8.45) of Ref. [2] as

$$\begin{aligned}
&F = i\sum_a \dot{\omega}_a T_a = i\sum_a \left(-\partial_{\omega_a}\left(S_G[U_\mu(n)] + S_{pf}[U_\mu(n)]\right)\right) T_a = F_G + F_{pf}.\\
&F_{pf} = i\sum_a \left(-\partial_{\omega_a} S_{pf}[U_\mu(n)]\right) T_a = -i\sum_a T^a \frac{\partial}{\partial\omega_a}\left(\phi^\dagger\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right).\\
&\frac{\partial}{\partial\omega_a}\left(\phi^\dagger\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right) = -\left(\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right)^\dagger\left(\frac{\partial D}{\partial\omega_\mu^a}\hat{D}^\dagger + \hat{D}\frac{\partial D^\dagger}{\partial\omega_\mu^a}\right)\left(\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right).\\
&\frac{\partial\hat{D}}{\partial\omega_\mu^a} = \left(\frac{\partial D}{\partial\omega_\mu^a}\right)_{x_L.x_R} = -i\kappa\left\{(1-\gamma_\mu)T^a U_\mu(x)\delta_{x,x_L}\delta_{x,(x+\mu)_R} - (1+\gamma_\mu)U_\mu^{-1}(x)T^a\delta_{x,(x+\mu)_L}\delta_{x,x_R}\right\}\\
&\hat{D}^\dagger = \gamma_5\hat{D}\gamma_5,\ \ \frac{\partial D^\dagger}{\partial\omega_\mu^i} = \gamma_5\frac{\partial D}{\partial\omega_\mu^a}\gamma_5
\end{aligned} \tag{22}$$

where F_G is force from U introduced in Sec. 2.1.2, T^a are $SU(3)$ generators. x_L, x_R are coordinate index of the left and right pseudofermion field. And

$$\begin{aligned}
&U_\mu = \exp(i\sum_a \omega_\mu^a T^a)U_0,\ \ \frac{\partial U_\mu}{\partial\omega_\mu^a} = iT^a U_\mu,\ \ \frac{\partial U_\mu^\dagger}{\partial\omega_\mu^a} = -iU_\mu^\dagger T^a,\\
&(T^a)^\dagger = T^a,\ \ \frac{\partial M^{-1}}{\partial\omega_\mu^a} = -M^{-1}\frac{\partial M}{\partial\omega_\mu^a}M^{-1}
\end{aligned} \tag{23}$$

are used. (Note that, Eq. (8.45) of Ref. [2] has a sign typo, see also Eq. (2.82) of Ref. [6])

We can simplify it further by $\left(\hat{D}^\dagger(\hat{D}\hat{D}^\dagger)^{-1}\phi\right)^\dagger = \left((\hat{D}\hat{D}^\dagger)^{-1}\phi\right)^\dagger\hat{D}$, so

$$\begin{aligned}
&\phi_1 = \left(\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right),\ \ \phi_2 = \hat{D}^\dagger\left(\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right) = D^{-1}\phi,\ \ \phi_1^\dagger D = \phi_2^\dagger,\\
&\frac{\partial}{\partial\omega_a}\left(\phi^\dagger\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right) = -\left(\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right)^\dagger\left(\frac{\partial D}{\partial\omega_\mu^a}\hat{D}^\dagger + \hat{D}\frac{\partial D^\dagger}{\partial\omega_\mu^a}\right)\left(\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right)\\
&= -\left(\phi_1^\dagger\frac{\partial D}{\partial\omega_\mu^a}\phi_2 + \phi_2^\dagger\frac{\partial D^\dagger}{\partial\omega_\mu^a}\phi_1\right) = -2\mathrm{Re}\left[\left(\phi_1^\dagger\frac{\partial D}{\partial\omega_\mu^a}\phi_2\right)\right]
\end{aligned} \tag{24}$$

and

$$\begin{aligned}
&\frac{\partial D}{\partial\omega_\mu^a} = -i\kappa M_a,\\
&(M_a)_{x_L,x_R} = \left\{(1-\gamma_\mu)T^a U_\mu\delta_{x_L,(x+\mu)_R} - (1+\gamma_\mu)U_\mu^{-1}T^a\delta_{(x+\mu)_L,x_R}\right\}\\
&\frac{\partial}{\partial\omega_a}\left(\phi^\dagger\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right) = -2\kappa\mathrm{Im}\left[\left(\phi_1^\dagger M\phi_2\right)\right]
\end{aligned} \tag{25}$$

Again, $\dot{\omega}$ is a **real** number, and

$$F_{pf} = -i \sum_a T^a \frac{\partial}{\partial \omega_a} \left(\phi^\dagger \left(\hat{D} \hat{D}^\dagger \right)^{-1} \phi \right) = 2 i \kappa \sum_a \mathrm{Im} \left[\left(\phi_1^\dagger M_a \phi_2 \right) \right] T_a \tag{26}$$

So we can calculate ϕ_1 first, then $\phi_2 = \hat{D}^\dagger \phi_1$. Then contract the spinor and color space with $\partial D / \partial \omega$.

Note that, D is changing when integrating the Langevin equation.

The last part is how to calculate $(\hat{D}\hat{D}^\dagger)^{-1}$.

- Anti-Hermitian traceless of the force

See from Eq. (18), the force from the gauge field is an anti-Hermitian traceless matrix.

The result above can be further simplified. Note that

$$\begin{aligned} &\phi_{L1}(n) = \phi_1(n), \;\; \phi_{R1}(n) = (1 - \gamma_\mu)\phi_2(n+\mu), \\ &\phi_{L2}(n) = \phi_1(n+\mu), \;\; \phi_{R1}(n) = (1+\gamma_\mu)\phi_2(n), \end{aligned} \tag{27}$$

One have

$$\begin{aligned} &\mathrm{Im} \left[\phi_1^\dagger M \phi_2 \right]_\mu^a (n) = \mathrm{Im} \left[\phi_{L1}^\dagger T^a U_\mu(n) \phi_{R1} \right] - \mathrm{Im} \left[\phi_{L2}^\dagger U_\mu^\dagger(n) T^a \phi_{R2} \right] \\ &= \mathrm{Im} \left[\phi_{L1}^\dagger T^a U_\mu(n) \phi_{R1} \right] + \mathrm{Im} \left[\phi_{R2}^\dagger T^a U_\mu(n) \phi_{L2} \right] \end{aligned} \tag{28}$$

For any vector

$$\mathrm{Im} \left[L^\dagger T U R \right] = \mathrm{Im} \left[\sum_{\alpha,\beta,\rho} L^*_\alpha T_{\alpha\beta} U_{\beta\rho} R_\rho \right] = \mathrm{Im} \left[\sum_{\alpha,\beta,\rho} T_{\alpha\beta} U_{\beta\rho} R_\rho L^*_\alpha \right] = \mathrm{Im} \left[\mathrm{tr} \left[T U (R L^\dagger) \right] \right] \tag{29}$$

So

$$\begin{aligned} &F_\mu^{pf}(n) = 2 i \kappa \mathrm{Im} \left[\phi_1^\dagger M \phi_2 \right]_\mu (n) = \kappa \left(2i \sum_a \mathrm{Imtr} \left[T^a U_\mu(n) \left(\phi_{R1} \phi_{L1}^\dagger + \phi_{R2} \phi_{L2}^\dagger \right) \right] T^a \right) \\ &= i\kappa \left\{ U_\mu(n) \left(\phi_{R1} \phi_{L1}^\dagger + \phi_{R2} \phi_{L2}^\dagger \right) \right\} \Big|_{TA} \end{aligned} \tag{30}$$

which is also an anti-Hermitian traceless matrix.

So, the momentum is always anti-Hermitian traceless..

For anti-Hermitian traceless matrix M, the $\exp(M)$ can be simplified as Appendix. A of Ref. [8].

2.1.4 Solver in HMC

To calculate $(\hat{D}\hat{D}^{\dagger})^{-1}$, we need a solver. The detail of solvers will be introduced in Sec. 3. Here we establish a simple introduction.

Let M be a matrix operating on a vector, for example, $M = (\hat{D}\hat{D}^{\dagger})$, the goal of the solver is to find x such $b = M \cdot x$, and therefor $x = (\hat{D}\hat{D}^{\dagger})^{-1}b$.

We first introduce the CG algorithm for real vector and real matrix, define

$$Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \cdot A \cdot \mathbf{x} - \mathbf{x}^T \mathbf{b}. \tag{31}$$

so that one can try to find the minimum of Q, and at the minimum

$$\frac{\partial}{\partial \mathbf{x}} Q(\mathbf{x}) = 0 = A \cdot \mathbf{x} - \mathbf{b}. \tag{32}$$

To find the minimum, one can use gradient. Starting from a random point on a curve, calculate the falling speed and move it until it is stable.

For complex vector, one can use BiCGStab in Table. 6.2 in Ref. [2]. It can be described as

Algorithm 1 BiCGStab, note that, the numbers are **complex** number.

$\mathbf{x} = \mathbf{b}$ ▷ Use **b** as trail solution and start.
for $i = 0$ to r **do**
  $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ ▷ Restart r times
  $\mathbf{r}_h = \mathbf{r}^*$ (Note, that we use r^* as in [9] which is tested to be better.)
  for $j = 0$ to $itera$ **do**
    $\rho = \mathbf{r}_h^* \cdot \mathbf{r}_j$
    if $j = 0$ **then**
      $\mathbf{p} = \mathbf{r}$
    else
      $\beta = \alpha \times \rho/(\omega \times \rho_p)$
      $\mathbf{p} = \mathbf{r} + \beta\,(\mathbf{p} - \omega\mathbf{v})$
    end if
    $\mathbf{v} = A\mathbf{p}$
    $\alpha = \rho/\left(\mathbf{r}_h^* \cdot \mathbf{v}\right)$
    $\mathbf{s} = \mathbf{r} - \alpha\mathbf{v}$
    if $0 \neq j$ and $0 = \mod(j, 5)$ **then**
      $er = \|\mathbf{s}\|$ ▷ Check deviation every 5 steps
      if $er < \epsilon$ **then**
   return x
      end if
    end if
    $\mathbf{t} = A\mathbf{s}$
    $\omega = \mathbf{s}^* \cdot \mathbf{t}/\|\mathbf{t}\|$
    $\mathbf{r} = \mathbf{s} - \omega\mathbf{t}$
    $\mathbf{x} = \mathbf{x} + \alpha\mathbf{p} + \omega\mathbf{s}$
    $\rho_p = \rho$ ▷ Preserve the last calculated ρ become we still need it
  end for
end for

2.1.5 Leap frog integrator

In Sec. 2.1.2, the basic idea is introduced. However, the implementation is slightly different.

$$U_\mu(0,x) = gauge(x), \;\; P_\mu(0,x) = \sum_a r_a(\mu,x)T_a \tag{33a}$$

$$F_\mu(n\epsilon,x) = -\frac{\beta}{2N}\{U_\mu(n\epsilon,x)\Sigma_\mu(n\epsilon,x)\}_{TA} \tag{33b}$$

$$P_\mu(\tfrac{1}{2}\epsilon,x) = P_\mu(0,x) + \tfrac{\epsilon}{2}F_\mu(0,x) \tag{33c}$$

$$U_\mu((n+1)\epsilon,x) = \exp\left(i\epsilon P_\mu((n+\tfrac{1}{2})\epsilon,x)\right)U_\mu(n\epsilon,x) \tag{33d}$$

$$P_\mu((n+\tfrac{1}{2})\epsilon,x) = P_\mu((n-\tfrac{1}{2})\epsilon,x) + \epsilon F_\mu(n\epsilon,x) \tag{33e}$$

Note that, the sign of F is '+' here which is different from Ref. [2], because in Ref. [2], $F = \partial_{\mu,n}S = -\dot{P}$. Here we define $F = \dot{P} = -\partial_{\mu,n}S$.

Or simply written as

$$P_\epsilon \circ U_\epsilon \circ P_{\frac{1}{2}\epsilon}(P_0, U_0) \tag{34}$$

The pseudo code can be written as

Algorithm 2 leap-frog integration

```
f = CalculateForce(actions, U)
p = p + 0.5 × εf
for i = 1 to n do
    U = exp(εp)U
    f = CalculateForce(actions, U)
    if i = n then
        p = p + 0.5 × εf          ▷ We still need to update p for the Metropolis step.
    else
        p = p + εf
    end if
end for
```

2.1.6 A summary of HMC with pseudofermions

Now, every part is ready. We summary the HMC following the Sec.8.2.3 in Ref. [2]. The HMC with fermions can be divided into 6 steps.

1. Generate a complex Bosonic field with $\chi \sim \exp(-\chi^\dagger \chi)$, and $\phi = \hat{D}\chi$.
2. Generate a momentum field P by $\exp(-tr(P^2))$.
3. Calculate $E = tr(P^2) + S_G(U) + S_{pf}(U, \phi)$.
4. Use U_0 to calculate F, evaluate P and U using integrator. Here, ϕ is treated as a constant field.
5. Finally, use P', U' to calculate Calculate $E' = tr({P'}^2) + S_G(U') + S_{pf}(U', \phi)$. Use a Metropolis to accept or reject the result (configurations) **Note, by Refs. [2] and [6] 'reject' means add a duplicated old configuration.**.
6. Iterate from 1 to 5, until the number of configurations generated is sufficient.

- More on Metropolis step:

If the hybrid Monte Carlo can be implemented exactly, then, when equilibrium is reached, H should be unchanged, so, in some implementation, the Metropolis step can be ignored to archive a better accept rate. The parameter `Metropolis` of parameter `Updator` can be set to 1 if Metropolis step is enabled and 0 otherwise.

2.2 Optimization of HMC

2.2.1 Omelyan integrator

The Omelyan integrator can be simply written as (c.f. Eq. (2.80) of Ref. [1])

$$P_{\lambda\epsilon} \circ U_{\frac{1}{2}\epsilon} \circ P_{(1-2\lambda)\epsilon} \circ U_{\frac{1}{2}\epsilon} \circ P_{\lambda\epsilon} \left(P_0, U_0\right) \tag{35}$$

with

$$\lambda = \frac{1}{2} - \frac{\left(2\sqrt{326}+36\right)^{\frac{1}{3}}}{12} + \frac{1}{6\left(2\sqrt{326}+36\right)^{\frac{1}{3}}} \approx 0.19318332750378364 \tag{36}$$

In practical, the λ is a tunable parameter, and usually, $2\lambda = 0.3 \sim 0.5$ [6]. The `Omelyan2Lambda` parameter of `Updator` is a input parameter to set 2λ, which if left blank is set to be 0.38636665500756728 by default.

Usually, for each sub-step, it is 2 times slower then leap-frog, and for one trajectory, it is 1.5 time faster [6], implying the number of sub-step needed is about 1/3 of leap-frog.

2.2.2 Omelyan force-gradient integrator

Start from the approximation

$$\log\left(\exp(\frac{\epsilon S}{6})\exp(\frac{\epsilon T}{2})\exp\left(\frac{2}{3}\epsilon S+\frac{\epsilon^3}{72}[S,[S,T]]\right)\exp(\frac{\epsilon T}{2})\exp(\frac{\epsilon S}{6})\right)=S+T+\mathcal{O}(\epsilon^4)+\mathcal{O}(\epsilon^6) \tag{37}$$

with [10]

$$\mathcal{O}(\epsilon^4)\sim 10^{-4}\epsilon^4 \tag{38}$$

The other 4 steps are the usual ones, except for $\exp\left(\frac{2}{3}\epsilon S+\frac{\epsilon^3}{72}[S,[S,T]]\right)$ which correspond to

$$\begin{aligned}\omega_i &\to \omega_i-\frac{2}{3}\tau\frac{\partial}{\partial\omega_i}S+\frac{1}{36}\tau^3\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\frac{\partial}{\partial\omega_i}S\\ p_i &= \sum_a \omega_i^a T^a\end{aligned} \tag{39}$$

Use the approximation [11]

$$\begin{aligned}&\frac{2}{3}\tau\frac{\partial}{\partial\omega_i}S-\frac{1}{36}\tau^3\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\frac{\partial}{\partial\omega_i}S\\ &=\frac{2}{3}\tau\exp\left(-\frac{1}{24}\tau^2\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\right)\frac{\partial}{\partial\omega_i}S+\mathcal{O}(\tau^5)\end{aligned} \tag{40}$$

Let U' be a function of U, solving

$$\begin{aligned}&\frac{2}{3}\tau\exp\left(-\frac{1}{24}\tau^2\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\right)\frac{\partial}{\partial\omega_i}S(U)=\frac{2}{3}\tau\frac{\partial}{\partial\omega_i}S(U')\\ &\exp\left(-\frac{1}{24}\tau^2\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\right)\frac{\partial}{\partial\omega_i}U\frac{\partial S(U)}{\partial U}=\frac{\partial}{\partial\omega_i}U'\frac{\partial S(U')}{\partial U'}\\ &\frac{\partial}{\partial\omega_i}\left(\exp\left(-\frac{1}{24}\tau^2\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\right)U\right)=\frac{\partial}{\partial\omega_i}U'\\ &\exp\left(-\frac{1}{24}\tau^2\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\right)U=U',\\ &U'=\exp\left(-\frac{1}{24}\tau^2\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)\frac{\partial}{\partial\omega_j}\right)U=\exp\left(-\frac{1}{24}\tau^2\sum_j\left(\frac{\partial}{\partial\omega_j}S\right)T_i\right)U\end{aligned} \tag{41}$$

This approximation can be divided into 3 steps:

1. Calculate $U' = \exp\left(-\frac{1}{24}\tau^2 \sum_j \left(\frac{\partial}{\partial \omega_j} S\right) T_i\right) U$.
2. Use $S[U']$ and $\frac{2}{3}\tau$ to update P.
3. Restore U.

It is easy to implement, we do not need to calculate second derivative, and $\sum_j \left(\frac{\partial}{\partial \omega_j} S\right) T_i$ is already implemented, it is nothing but the **force**.

It is almost as accurate as force-gradient, see the compare in Ref. [12]

2.2.3 Multi-rate integrator (nested integrator)

Following Ref. [12]

Assuming the action is $S = S_F + S_G$ with $S_G \gg S_F$, one can evaluate S_G more often than S_F. In the case of lattice QCD, often, the S_G is the cheap gauge force, and S_F is the expansive fermion force.

The nested scheme is different for leap-frog Omelyan and force-gradient integrator, but they are similar

- Nested leap-frog

$$\begin{aligned}\Delta(h) &= \exp(\frac{h}{2}S_F)\Delta_m(h)\exp(\frac{h}{2}S_F),\\ \Delta_m(h) &= \left(\exp(\frac{h}{2m}S_G)\exp(\frac{h}{m}T)\exp(\frac{h}{2m}S_G)\right)^m.\end{aligned} \tag{42}$$

- Nested Omelyan

$$\begin{aligned}\Delta(h) &= \exp(\lambda h S_F)\Delta_m(\frac{h}{2})\exp((1-2\lambda)hS_F)\Delta_m(\frac{h}{2})\exp(\epsilon h S_F),\\ \Delta_m(h) &= \left(\exp(\frac{\lambda h}{m}S_G)\exp(\frac{h}{2m}T)\exp(\frac{1-2\lambda}{m}hS_G)\exp(\frac{h}{2m}T)\exp(\frac{\lambda h}{m}S_G)\right)^m.\end{aligned} \tag{43}$$

Note, it is $\Delta_m(\frac{h}{2})$ in the first line.

- Nested force-gradient

$$
\begin{aligned}
&\Delta(h) = \exp(\frac{h}{6}S_F)\Delta_m(\frac{h}{2})\exp(\frac{2}{3}hS_F + \frac{1}{72}h^3C_F)\Delta_m(\frac{h}{2})\exp(\frac{h}{6}S_F),\\
&\Delta_m(h) = \left(\exp(\frac{h}{6m}S_G)\exp(\frac{h}{2m}T)\exp\left(\frac{2}{3}\frac{h}{m}S_G + \frac{1}{72}\left(\frac{h}{m}\right)^3 C_G\right)\exp(\frac{h}{2m}T)\exp(\frac{h}{6m}S_G)\right)^m.
\end{aligned}
\tag{44}
$$

Note about the integrator: when analyzing the error of the integrators, it is assumed e^T, e^{S_G} and e^{S_F} can be calculated accurately. It is almost true for e^{S_G}, and almost true for e^T as long as ϵ is not too large, but it is not true for e^{S_F}. Typically, using an optimized integrator, it needs more accurate criterion for solvers.

2.2.4 Cached solution

The pseudo fermion field is generate only once for a trajectory and is not changed. Also, the gauge field is changing slowly in one trajectory, this make the solutions for $\mathbf{x}_1 = D^{-1}\mathbf{b}$ or $\mathbf{x}_2 = \left(DD^\dagger\right)^{-1}\mathbf{b}$, where D depends on U and $\mathbf{b}$ is the pseudo fermion field, only change slowly.

So, once $\mathbf{x}_{1,2}$ is obtained, in the same trajectory, $\mathbf{x}_{1,2}$ can be set as the initial trail solution for the solver.

2.3 Staggered Fermion

2.3.1 The D=1 staggered fermion and Jordan-Wigner transformation

One way to introduce fermion is to use the Ising action (which is particularly useful in quantum computer simulation of $\mathbb{Z}_2$ lattice gauge [13]). The Jordan-Wigner transformation of $D=1$ Ising model is famous, now considering $D=1$ XY model

$$H = \frac{1}{4a}\sum_n \left(\sigma_x(n)\sigma_x(n+1) + \sigma_y(n)\sigma_y(n+1)\right) \tag{45}$$

with

$$\phi^\dagger(n) = \sigma^+(n)\left\{\prod_{m<n}(-\sigma_z(n))\right\},\quad \phi(n) = \left\{\prod_{m<n}(-\sigma_z(n))\right\}\sigma^-(n),\quad \sigma^\pm = \frac{\sigma_x \pm i\sigma_y}{2} \tag{46}$$

one can verify that ϕ is fermion

$$\{\phi(x),\phi^\dagger(y)\} = \delta_{xy},\quad \{\phi^\dagger(x),\phi^\dagger(y)\} = \{\phi(x),\phi(y)\} = 0 \tag{47}$$

and

$$H = \frac{1}{2a} \sum_n \left(\phi^\dagger(n)\phi(n+1) - \phi^\dagger(n+1)\phi(n) \right) \tag{48}$$

with EoM same as naive discretized fermion

$$\dot{\phi}(n) = -i[\phi(n), H] = \frac{1}{2a} \left(\phi(n+1) - \phi(n-1) \right) \tag{49}$$

2.3.2 The relationship between the naive fermion and staggered fermion

Consider the naive discretized **massless** fermion

$$S_F = a^4 \sum_n \sum_\mu \frac{\bar{\psi}(n)\gamma_\mu U_\mu(n)\psi(n+\mu) + \bar{\psi}(n)\gamma_\mu U_{-\mu}(n)\psi(n-\mu)}{2a} \tag{50}$$

Although γ matrix can mix different component of the spinor, one can however, reorder the component, for example

$$\bar{\psi}(n)\gamma_x\psi(n+x) = \begin{pmatrix} \phi_x(n) \\ \phi_y(n) \\ \phi_z(n) \\ \phi_t(n) \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \phi_t(n+x) \\ \phi_z(n+x) \\ \phi_y(n+x) \\ \phi_x(n+x) \end{pmatrix} \tag{51}$$

One can see ϕ_x only couple to ϕ_x. **It can be shown that, such reorder is independent of path from any given starting site.**

To show that, just start from $\phi_i(n)$, reorder $\phi_i(n+\mu)$ based on $\phi_i(n)$, then $\phi_i(n+\mu+\nu)$ based on $\phi_i(n+\mu)$, then $\phi_i(n+\nu)$ based on $\phi_i(n+\mu+\nu)$ then $\phi_i(n)$ based on $\phi_i(n+\nu)$ (a loop), use the fact that there are always even number of γ_μ for any μ in a loop, and $\gamma_\mu^2 = 1$, so **a loop-reorder will not change the order of component**, which result in **reorder is independent of path**, and therefore $\psi(x)$ **on each site has an ambiguous reorder**.

• An easy reorder

Since it has been proved that reorder is independent of path, the reorder only depend on the coordinate (x, y, z, t). An easy way is to define $\psi(n) = \gamma_1^x\gamma_2^y\gamma_3^z\gamma_4^t\chi(n)$. Note that, for $i \neq j$, $\gamma_i\gamma_j = -\gamma j\gamma_i$, **so there is a sign due to commutate gamma matrix**

$$\begin{aligned} &\chi(n) = \gamma_4^t\gamma_3^z\gamma_2^y\gamma_1^x\psi(n) \\ &\bar{\chi}(n)\chi(n+y) = \bar{\psi}(n)\gamma_1^x\gamma_2^y\gamma_3^z\gamma_4^t\gamma_4^t\gamma_3^z\gamma_2^{y+1}\gamma_1^x\psi(n+y) = (-1)^x\bar{\psi}(n)\gamma_y\psi(n+y) \\ &\bar{\chi}(n)\chi(n+z) = \bar{\psi}(n)\gamma_1^x\gamma_2^y\gamma_3^z\gamma_4^t\gamma_4^t\gamma_3^{z+1}\gamma_2^y\gamma_1^x\psi(n+y) = (-1)^{x+y}\bar{\psi}(n)\gamma_z\psi(n+z) \end{aligned} \tag{52}$$

Defining $\eta_\mu(x)$ so that (**Note that μ in $eta_\mu(x)$ can be $\mu = 5$**)

$$\begin{aligned}&\eta_\mu(x) = (-1)^{\sum_{\nu<\mu} x_\nu}\\ &\eta_\mu(x)\bar{\chi}(n)\chi(n+\mu) = \bar{\psi}(n)\gamma_\mu\psi(n+\mu)\end{aligned} \tag{53}$$

Note that, it hold with massive fermions that the naive discretization can be written as

$$S_F = a^4 \sum_n \left\{ \sum_\mu \eta_\mu(n)\bar{\chi}(n) \frac{U_\mu(n)\chi(n+\mu) - U_{-\mu}(n)\chi(n-\mu)}{2a} + m\bar{\chi}(n)\chi(n) \right\} \tag{54}$$

Here χ is still a 4-component spinor, which is in fact a reorder of $\psi(n)$, the components of χ do NOT mix with each other. Now by the fact each component of χ is equivalent (degenerate),

$$\begin{aligned}S_F &= a^4 \sum_n \sum_\alpha \left\{ \sum_\mu \eta_\mu(n)\bar{\chi}_\alpha(n) \frac{U_\mu(n)\chi_\alpha(n+\mu) - U_{-\mu}(n)\chi_\alpha(n-\mu)}{2a} + m\bar{\chi}_\alpha(n)\chi_\alpha(n) \right\}\\ &= 4a^4 \sum_n \left\{ \sum_\mu \eta_\mu(n)\bar{\chi}(n) \frac{U_\mu(n)\chi(n+\mu) - U_{-\mu}(n)\chi(n-\mu)}{2a} + m\bar{\chi}(n)\chi(n) \right\}\end{aligned} \tag{55}$$

In the last line, we redefine $\chi(x)$ as a scalar (1-component) field.

Kogut-Susskind staggered fermion is just naive discretized fermion.

2.3.3 Symmetries of the staggered fermions

Now we concentrate on

$$D_{st}(n|m) = m\delta_{m,n} + \sum_\mu \eta_\mu(n) \frac{U_\mu(n)\delta_{n,n+\mu} - U_{-\mu}(n)\delta_{n,n-\mu}}{2a} \tag{56}$$

- D_{st} is γ_5-hermiticity

$$D^\dagger_{st}(n|m) = \eta_5(n) D^\dagger_{st}(n|m)\eta_5(m) \tag{57}$$

- **massless** case is **anti-hermitian traceless**

$$D^\dagger_{st}(n|m) = -D_{st}(n|m) \tag{58}$$

- Chiral symmetry

The **massless** S_F is unchanged under transformation

$$\chi(n) \to \exp(i\alpha\eta_5(n))\chi(n), \quad \bar{\chi}(n) \to \bar{\chi}(n)\exp(i\alpha\eta_5(n)) \tag{59}$$

Just note that they are somehow different, the chiral symmetry is from $\gamma_\mu \exp(i\alpha\gamma_5) = \exp(-i\alpha\gamma_5)\gamma_\mu$, and the chiral symmetry of staggered fermion is from $\eta_\mu \exp(i\alpha\eta_5(n+\mu)) = \exp(i\alpha\eta_5(n+\mu))\eta_\mu = \exp(-i\alpha\eta_5(n))\eta_\mu$.

• Sign problem of the staggered fermions

Let $D_{st}(m=0) = U^\dagger \Lambda U$ where Λ is diagonal, U is unitary. $D^\dagger_{st}(m=0) = U^\dagger \Lambda^\dagger U = -D_{st}(m=0) = -U^\dagger \Lambda U$, so Λ is a diagonal matrix with $\Lambda^\dagger = -\Lambda$, Λ should be pure imaginary number.

Then, use $D = U^\dagger \Lambda U + mU^\dagger U = U^\dagger (\Lambda + m) U$, and $\det[D] = \det[U^\dagger]\det[\Lambda+m]\det[U]$. And use $\det[U^\dagger] = \det[U^{-1}] = 1/\det[U]$, one find $\det[D_{st}] = \det[\Lambda + m]$, where Λ is pure imaginary number, and $m \geq 0$ is real number. One find **the eigenvalues of D_{st} are complex number with real part exactly the mass.**

I know the sum of eigenvalues is real number, but how to prove they are complex conjugate pairs?

It has been proved that $\det[D_{st}] \geq m$, so one do not worry about the sign problem, even with a single flavour. And It is a consequence of (i) massless case is anti-hermitian traceless; (ii) the massive case is η_5-hermiticity.

2.4 The RHMC for staggered fermion

From now on, we consider $N_f = 2+1$. Two key quantities should be calculated.

• Calculate the action

$$\exp(-S) = \left(\det[D_{st}(m_{ud})]\right)^{\frac{1}{2}} \left(\det[D_{st}(m_s)]\right)^{\frac{1}{4}} \exp(-S_G) \tag{60}$$

The problem is how to evaluate $(\det[D])^\alpha$. Before that, we have to prepare some tools.

2.4.1 The rational approximation and Remes algorithm

The first thing to do is to find a rational approximation of a function, here we use Remes algorithm as proposed by Ref.

The Remes algorithm is implemented in Mathematica, which finds

$$g(x) = c + \sum_i \frac{a_i}{x + b_i} \approx f(x), \tag{61}$$

in an interval. This have been implemented in Mathematica so we use the result of Mathematica directly. An example is

```
<< FunctionApproximations\;
    res1 = Apart[MiniMaxApproximation[1/Sqrt[x], {x, {0.003, 1}, 3, 3}][[2]][[1]]]
    Join[{Part[res1, 1]}, Table[Numerator[Part[res1, n]], n, 2, 4], Table[Denominator[Part[res1,
    n]] /.  x -> 0, {n, 2, 4}]]
```

2.4.2 The RHMC with $N_f = 2$

Similar as the HMC, the first step is to rewrite the action use a Boson field (pseudo-fermion field), still starting with

$$Z = \int \mathcal{D}[U] \prod_{f=1}^{N_f} \mathcal{D}[\bar{\psi}_f]\mathcal{D}[\psi_f] \left(\det D_{st}\right)^{\frac{1}{4}} \exp\left(-S_G[U]\right) \tag{62}$$

Note that the $1/4$ dose NOT come from the $1/4$ in Eq. (55), it is to deal with the 4 doublers. With

$$D_{st}(n|m) = 2m\delta_{m,n} + \sum_{\mu} \eta_\mu(n) \left(U_\mu(n)\delta_{n,n+\mu} - U_{-\mu}(n)\delta_{n,n-\mu}\right) \tag{63}$$

It is very interesting that, although stated in many literatures, this is **not** the true fourth root method. The true fourth root method is after the 'pseudofermion' step. First, just like the usual HMC to evaluate

$$Z = \int \mathcal{D}[U] \prod_{f=1}^{N_f} \left(\det D_{st}\right) \exp\left(-S_G[U]\right) \tag{64}$$

For $N_f = 2$, before the 4th root, it is

$$Z = \int \mathcal{D}[U] \left(\det D_{st}^\dagger D_{st}\right) \exp\left(-S_G[U]\right) \tag{65}$$

Note that because D_{st0} for massless is anti-Hermitian, $D_{st}^\dagger D_{st} = -\frac{1}{4a^2}D_{st0}^2 + m^2$, for simplicity we denote $A = -D_{st0}^2 + m^2$ (with a rescale of ψ assumed), using

$$\det A = \frac{1}{\det A^{-1}} = \int \mathcal{D}[\phi] \exp(-\phi^\dagger A^{-1}\phi). \tag{66}$$

For $N_f = 2$ it is

$$Z = \int \mathcal{D}[U]\mathcal{D}[\phi] \exp\left(-S_G[U] - \phi^\dagger A^{-1}\phi\right) \tag{67}$$

The true fourth root method is taken here as

$$
\begin{aligned}
Z &= \int \mathcal{D}[U] \left(\det A\right)^{\frac{N_f}{4}} \exp\left(-S_G[U]\right) \\
Z &= \int \mathcal{D}[U]\mathcal{D}[\phi] \exp\left(-S_G[U] - \phi^\dagger A^{-\frac{N_f}{4}}\phi\right)
\end{aligned}
\tag{68}
$$

Note that the fourth root is just to remove doublers, so for $N_f = 2$, it is $A^{1/2}$ in spite that $A = D^\dagger D$ already considered $N_f = 2$.

In the following, we concentrate on

$$
Z = \int \mathcal{D}[U]\mathcal{D}[\phi] \exp\left(-S_G[U] - \phi^\dagger A^{-\frac{1}{2}}\phi\right) \tag{69}
$$

• Pseudofermion refreshment of staggered fermion

Note! The refreshment of staggered fermion use a different action! It can be summarized as $Z_{MC} \approx Z_{MD}$, where Z_{MC} is to refresh the pseudofermion, Z_{MD} is for molecular dynamic.

$$
\begin{aligned}
Z_{MC} &= \int \mathcal{D}[U]\mathcal{D}[\phi] \exp\left(-S_G[U] - \phi^\dagger \left(A^{-\frac{1}{4}}\right)^2 \phi\right) \\
Z_{MD} &= \int \mathcal{D}[U]\mathcal{D}[\phi] \exp\left(-S_G[U] - \phi^\dagger r_2(A)\phi\right)
\end{aligned}
\tag{70}
$$

where $r_2(A) \approx A^{-\frac{1}{2}}$.

Then, the refreshment step is:

Let χ be random **complex** numbers according to $\exp(-\chi^\dagger\chi)$. Let $A^{-\frac{1}{4}}\phi = \chi$, ϕ is the random **complex** number satisfying distribution we want $(\exp(-\phi^\dagger \left(A^{-\frac{1}{4}}\right)^\dagger A^{-\frac{1}{4}}\phi))$. So, first get χ and then let $\phi = A^{+\frac{1}{4}}\chi$.

• MD step of staggered fermion

Let $A^{-\frac{1}{2}} = c_0 + \sum_i \frac{\alpha_i}{A+\beta_i}$, it is

$$
Z = \int \mathcal{D}[U]\mathcal{D}[\phi] \exp\left(-S_G[U] - c_0\phi^\dagger\phi - \sum_i \phi^\dagger \frac{\alpha_i}{A+\beta_i}\phi\right) \tag{71}
$$

then

$$
\begin{aligned}
F &= -iT_a \frac{\partial}{\partial\omega_a}\left(\sum_i \phi^\dagger \frac{\alpha_i}{A+\beta_i}\phi\right) \\
&= -iT_a \sum_i \left(\phi^\dagger \frac{\alpha_i}{(A+\beta_i)^2} \frac{\partial}{\partial\omega_a} A\phi\right)
\end{aligned}
\tag{72}
$$

The order of matrix product is wrong, it is in fact

$$F=-iT_a\sum_i\left(\phi^\dagger\frac{\sqrt{\alpha_i}}{(A+\beta_i)}\left(\frac{\partial}{\partial\omega_a}A\right)\frac{\sqrt{\alpha_i}}{(A+\beta_i)}\phi\right) \tag{73}$$

Now, let $\phi_i=\frac{\sqrt{\alpha_i}}{(A+\beta_i)}\phi$, it is

$$F=-iT_a\sum_i\phi_i^\dagger\left(\frac{\partial}{\partial\omega_a}A\right)\phi_i \tag{74}$$

Note it is not $\phi'=\sum_i\frac{\sqrt{\alpha_i}}{(M+\beta_i)}\phi$ **because generally,** $\phi_i^\dagger A'\phi_j\neq 0$

To calculate the A' we use

$$\begin{aligned}
&\frac{\partial}{\partial\omega_a}A=\left(\frac{\partial}{\partial\omega_a}D^\dagger_{st0}\right)D_{st0}+D^\dagger_{st0}\left(\frac{\partial}{\partial\omega_a}D_{st0}\right)\\
&\phi_i^\dagger\left(\frac{\partial}{\partial\omega_a}A\right)\phi_i=\phi_i^\dagger\left(\frac{\partial}{\partial\omega_a}D^\dagger_{st0}\right)\phi_{i,d}+\phi^\dagger_{i,d}\left(\frac{\partial}{\partial\omega_a}D_{st0}\right)\phi_i\\
&=\left(\phi^\dagger_{i,d}\left(\frac{\partial}{\partial\omega_a}D_{st0}\right)\phi_i\right)^\dagger+\left(\phi^\dagger_{i,d}\left(\frac{\partial}{\partial\omega_a}D_{st0}\right)\phi_i\right)\\
&=2\mathrm{Re}\left[\phi^\dagger_{i,d}\left(\frac{\partial}{\partial\omega_a}D_{st0}\right)\phi_i\right]
\end{aligned} \tag{75}$$

with $\phi_{i,d}=D_{st0}\phi_i$.

$$\begin{aligned}
&\frac{\partial}{\partial\omega_a}D_{st0}=\sum_\mu\left(\eta_\mu(n)\left(\frac{\partial}{\partial\omega_a}U_\mu(n)\right)\delta_{n,n+\mu}-\eta_\mu(n+\mu)\left(\frac{\partial}{\partial\omega_a}U^\dagger_\mu(n)\right)\delta_{n+\mu,n}\right)\\
&=i\eta_\mu(n)\left(\left(T_aU_\mu(n)\right)\delta_{n,n+\mu}+\left(U^\dagger_\mu(n)T^\dagger_a\right)\delta_{n+\mu,n}\right)
\end{aligned} \tag{76}$$

note that the $\eta_\mu(n+\mu)$ follows $\delta_{n+\mu,n}$. The last step we use $\eta_\mu(n+\mu)=\eta_\mu(n)$. The $T_a^\dagger$ is not convenient, so let

$$\begin{aligned}
&\phi_A=\phi_i,\phi_B=T_aU_\mu(n)\phi_i\\
&\phi_C=\phi_{i,d},\phi_D=T_aU_\mu(n)\phi_{i,a}
\end{aligned} \tag{77}$$

it is

$$
\begin{aligned}
&\phi_{i,d}^{\dagger}\left(\frac{\partial}{\partial\omega_a}D_{st0}\right)\phi_i = i\eta(n)\left(\phi_C^{\dagger}(n)\phi_B(n+\mu)+\phi_A^{\dagger}(n)\phi_D(n+\mu)\right)\\
&\phi_i^{\dagger}\left(\frac{\partial}{\partial\omega_a}A\right)\phi_i = i\eta(n)\left(\phi_C^{\dagger}(n)\phi_B(n+\mu)+\phi_A^{\dagger}(n)\phi_D(n+\mu)\right)-i\eta(n)\left(\phi_B^{\dagger}(n+\mu)\phi_C(n)+\phi_D^{\dagger}(n+\mu)\phi_A(n)\right)\\
&=i\eta(n)\left\{\left(\phi_C^{\dagger}(n)\phi_B(n+\mu)-\phi_B^{\dagger}(n+\mu)\phi_C(n)\right)+\left(\phi_A^{\dagger}(n)\phi_D(n+\mu)-\phi_D^{\dagger}(n+\mu)\phi_A(n)\right)\right\}\\
&=-2\eta(n)\mathrm{Im}\left[\phi_C^{\dagger}(n)\phi_B(n+\mu)-\phi_A^{\dagger}(n)\phi_D(n+\mu)\right]\\
&=-2\eta(n)\mathrm{Im}\left[\phi_{i,d}^{\dagger}T_aU_\mu(n)\delta_{n,n+\mu}\phi_i-\phi_i^{\dagger}(n)T_aU_\mu(n)\delta_{n,n+\mu}\phi_{i,d}\right]
\end{aligned}
\tag{78}
$$

let $M(n|m)=U_\mu(n)\delta_{n,n+\mu}$, it is

$$
\phi_{i,d}^{\dagger}\left(\frac{\partial}{\partial\omega_a}D_{st0}\right)\phi_i=-2\eta(n)\mathrm{Imtr}\left[T_a\left((M\phi_i)\phi_{i,d}^{\dagger}-(M\phi_{i,d})\phi_i^{\dagger}\right)\right]
\tag{79}
$$

One can verify that for any matrix

$$
\begin{aligned}
&2i\sum_a T_a\mathrm{Imtr}[T_aM]=-2i\sum_a T_a\mathrm{Retr}[iT_aM]=\{M\}_{TA}\\
&2i\sum_a T_a\mathrm{Imtr}[iT_aM]=2i\sum_a T_a\mathrm{Retr}[T_aM]=\{iM\}_{TA}
\end{aligned}
\tag{80}
$$

so

$$
\begin{aligned}
&F_\mu(n)=-iT_a\sum_i\phi_i^{\dagger}\left(\frac{\partial}{\partial\omega_a}A\right)\phi_i\\
&=\eta_\mu(n)\sum_a\sum_i 2iT_a\mathrm{Imtr}\left[T_a\left((M\phi_i)\phi_{i,d}^{\dagger}-(M\phi_{i,d})\phi_i^{\dagger}\right)\right]\\
&=\eta_\mu(n)\sum_i\left\{(M\phi_i)\phi_{i,d}^{\dagger}-(M\phi_{i,d})\phi_i^{\dagger}\right\}_{TA}
\end{aligned}
\tag{81}
$$

with

$$
\begin{aligned}
&\phi_i=\frac{\sqrt{\alpha_i}}{(D_{st}^{\dagger}D_{st})+\beta_i}\\
&\phi_{i,d}=D_{st0}\phi_i\\
&M(n|m)=U_\mu(n)\delta_{n,n+\mu}
\end{aligned}
\tag{82}
$$

Here is the summary of $N_f=2$ RHMC for staggered fermion

1. Refresh ϕ using $\phi=(D_{st}^{\dagger}D_{st})^{\frac{1}{4}}\chi$, where χ is random Gaussian complex field.

2. Calculate force using Eq. (80).

3. Metropolis step.

Some tips on RHMC of staggered fermions:

- Let $r_1(M) \approx M^a$, $r_2(M) \approx M^b$, $r_1(M)$ is generally less accurate than $(r_2(M))^{1/b}$ when $b > 1$.

- For $N_f = 2 + 1$, there are other optimizations.

3 Sparse linear algebra solver

Given a matrix A and a vector $\mathbf{b}$. The solver works out the solution

$$\mathbf{b} = A\mathbf{x}, \quad \mathbf{x} = A^{-1}\mathbf{b}. \tag{83}$$

3.1 Krylov subspace

In short, the Krylov subspace methods assumes

$$\mathbf{x} \approx \sum_{l=0}^{k-1} C_l A^l \mathbf{b} \in K_k = span\left\{\mathbf{b}, A\mathbf{b}, \ldots, A^{k-1}\mathbf{b}\right\}. \tag{84}$$

with finite k, where C_k are coefficients. The equation $0 = \mathbf{b} - A\mathbf{x}$ becomes

$$\|\mathbf{b} - \sum_{l=0}^{k-1} C_l A^{l+1}\mathbf{b}\| = 0 \tag{85}$$

This is a problem in $k+1$ dimension, where k is independent of the dimension of $\mathbf{b}$, and usually significantly smaller than the dimension of $\mathbf{b}$. The Eq. (84) can be understand that, if $\mathbf{x}_k \approx A^{-1}\mathbf{b}$ is approximation of the solution in k dimension, in the $k+1$ dimension

$$\left(\mathbf{b} - A\mathbf{x}\right)_{k+1} \perp K_k \tag{86}$$

That is, if we have a multi-dimension vector v_n, and its projection in 3-dimension ($D = k+1$) is a vector $\mathbf{v}_3$, if we want to find a plane ($D = k$) such that the projection of v_n in the plane is minimized, the plane is chosen to be the one orthogonal to $\mathbf{v}_3$.

3.2 GMRES

This section we follow Refs. [14] and [9].

Assume a set of basis has been found. For example, if the subspace is found by using modified Gram-Schmidt as

Algorithm 3 Arnoldi with modified Gram-Schmidt

$\mathbf{v}^{(0)} = \mathbf{x}_0/\|\mathbf{x}_0\|$
for $i = 0$ to $k-1$ **do**
  $\mathbf{w} = A\mathbf{v}^{(i)}$
  for $j = 0$ to i **do**
    $c = \mathbf{v}^{(j)^*} \cdot \mathbf{w}$
    $\mathbf{w}- = c\mathbf{v}^{(j)}$
    $h[j,i] = c$
  end for
  $h[i+1,i] = \|\mathbf{w}\|$
  $\mathbf{v}^{(i+1)} = \mathbf{w}/\|\mathbf{w}\|$
end for

Note that $(\mathbf{w} - (\mathbf{v}_i^* \cdot \mathbf{w})\,\mathbf{v}_i)^* \cdot \mathbf{v}_i = 0$, and $\mathbf{x}_0$ is a trail solution, which can be set to be $\mathbf{b}$ at first. Now we obtain $k+1$ unitary orthogonal vectors, such that

$$\mathbf{v}_i^* \cdot \mathbf{v}_j = \delta_{ij}, \quad A\mathbf{v}_{i-1} = \sum_{j=0}^{i} h[j,i-1]\mathbf{v}_j, \tag{87}$$

That is

$$\begin{pmatrix} Av_0 \\ Av_1 \\ Av_2 \\ \dots \\ Av_{k-1} \end{pmatrix} = (v_0, v_1, \dots, v_{k-1}, v_k) \begin{pmatrix} \text{h[0,0]} & \text{h[0,1]} & \dots & \text{h[0,k-2]} & \text{h[0,k-1]} \\ \text{h[1,0]} & \text{h[1,1]} & \dots & \text{h[1,k-2]} & \text{h[1,k-1]} \\ 0 & \text{h[2,1]} & \dots & \text{h[2,k-2]} & \text{h[2,k-1]} \\ 0 & 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \text{h[k-1,k-2]} & \text{h[k-1,k-1]} \\ 0 & 0 & \dots & 0 & \text{h[k,k-1]} \end{pmatrix} \tag{88}$$

which can be written as

$$(Av)_k = v_{k+1}H \tag{89}$$

Assume the solution is

$$\mathbf{x} = \mathbf{x}_0 + \sum_{i=0}^{k-1} y_i \mathbf{v}_i = \mathbf{x}_0 + \mathbf{y} = \mathbf{x}_0 + v_k y, \tag{90}$$

Using $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, to minimize $\|\mathbf{b} - A\mathbf{x}\|$ is to minimize $\|\mathbf{r}_0 - A\mathbf{y}\|$. We always choose $\mathbf{v}_0 = \mathbf{r}_0/\|\mathbf{r}_0\|$, denote $\beta = \|\mathbf{r}_0\|$, it is to minimize

$$\text{argmin}\|\beta\mathbf{e}_0 - Hy\|. \tag{91}$$

Or, to solve an equation in k dimension

$$\beta\mathbf{e}_0 - Hy = 0, \;\; y = H^{-1}\beta\mathbf{e}_0 = H^{-1}g \tag{92}$$

Now, we need to solve H^{-1}, we can do this by applying rotation matrix, defining (This is also called **Givens rotation**)

$$J_0 = \begin{pmatrix} R & 0 \\ 0 & \mathbb{I}_{k-2} \end{pmatrix}_{D=k} = \begin{pmatrix} c_0^* & s_0^* & 0 & \ldots & 0 \\ -s_0 & c_0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ 0 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}_{D=k} \tag{93}$$

Note that c_0^* and s_0^* is necessary to keep unitary. (s_0^* seems not necessary? we only need to keep the length of g unchanged) So that

$$0 = g - Hy \to 0 = J_0 g - J_0 H y \tag{94}$$

with (Note that the first 2 lines are changed entirely)

$$\begin{aligned} H' &= \begin{pmatrix} h'_{0,0} & h'_{0,1} & h'_{0,2} & \ldots & h'_{0,k-1} \\ 0 & h'_{1,1} & h'_{1,2} & \ldots & h'_{1,k-1} \\ 0 & h_{2,1} & h_{2,2} & \ldots & h_{2,k-1} \\ 0 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 0 & 0 & h_{k,k-1} \end{pmatrix} \\ g' &= (c_0^*\beta, -s_0\beta, 0, \ldots) \\ c_0 &= \frac{h_{00}}{\sqrt{h_{00}^2 + h_{10}^2}}, \;\; s_0 = \frac{h_{10}}{\sqrt{h_{00}^2 + h_{10}^2}} \end{aligned} \tag{95}$$

where $\mathbb{I}_l$ is dimension l identity matrix. Similarly, after this, one can rotation matrices

$$J_1 = \begin{pmatrix} \mathbb{I}_1 & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & \mathbb{I}_{k-3} \end{pmatrix}_{D=k}, J_2 = \begin{pmatrix} \mathbb{I}_2 & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & \mathbb{I}_{k-4} \end{pmatrix}_{D=k}, \ldots \tag{96}$$

To make H triangular.

The algorithm is

Algorithm 4 Rotate H

$g[0] = \beta$
for $i = 0$ to $k-1$ **do**
  $d = 1/\sqrt{|h[i,i]|^2 + |h[i+1,i]|^2}$
  $cs = h[i,i] \times d, sn = h[i+1,i] \times d$
  for $j = i$ to $k-1$ **do**
    $h_{ij} = h[i,j]$
    $h[i,j] = cs^* \times h_{ij} + sn^* \times h[i+1,j]$
    $h[i+1,j] = cs \times h[i+1,j] - sn \times h_{ij}$
  end for
  $minus_g = -g[i]$
  $g[i] = cs^* \times g[i]$
  $g[i+1] = sn \times minus_g$
end for

After the rotation, $g[k]$ is the residue. If it is small enough, the last step is to solve $y = H^{-1}g$, where H is a upper triangular matrix. It can be iterated as

$$y[k-1] = \frac{g[k-1]}{h[k-1,k-1]}.\ y[k-2] = \frac{1}{h[k-2,k-2]}\left(g[k-2] - h[k-2,k-1]y[k-1]\right), \ldots \tag{97}$$

The algorithm is backward substitution

Algorithm 5 Solve Y

---

for $i = k-1$ to 0 **do**
  for $j = i+1$ to $k-1$ **do**
    $g[i] -= h[i,j] \times y[j]$
  end for
  $y[i] = g[i]/h[i,i]$
end for
  return $\mathbf{x}_0 + \sum_{i=0}^{k-1} y[i]\mathbf{v}^{(i)}$

Note that, the first step, the modified Gram-Schmidt step will produce more and more unitary normalized vectors, so the GMRES usually has a restart step. Let r denote the restart times, for example, the full algorithm with k is (GMRES(m) means GMRES with modified Gram-Schmidt, there is also GMRES with Household, etc)

Algorithm 6 GMRES(m)

$\mathbf{x}_0 = \mathbf{b}$ ▷ Use **b** as trail and start
for $i = 1$ to r **do**
  $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
  $\beta = \|\mathbf{r}_0\|$
  $\mathbf{v}^{(0)} = \mathbf{r}_0/\beta$
  for $i = 0$ to $k-1$ **do**
    $\mathbf{w} = A\mathbf{v}^{(i)}$
    for $j = 0$ to i **do**
      $c = {\mathbf{v}^{(j)}}^* \cdot \mathbf{w}$
      $\mathbf{w} -= c\mathbf{v}^{(j)}$
      $h[j,i] = c$
    end for
    $h[i+1,i] = \|\mathbf{w}\|$
    $\mathbf{v}^{(i+1)} = \mathbf{w}/\|\mathbf{w}\|$
  end for
  $RotateH(k)$
  $\mathbf{x} = SolveY(k)$
  if $|g[k]| < \epsilon$ **then**
   return x ▷ Succeed, with the solution
  end if
  $\mathbf{x}_0 = \mathbf{x}$ ▷ Use the last solution as trail and restart
end for
   return x ▷ Failed, with the last best solution

where $RotateH(k)$ and $\mathbf{x} = SolveY(k)$ is described in Algorithms. 4 and 5.

3.3 GCR

This section we follow Ref. [9].

The GCR solver is similar to GMRES in Sec. 3.2, but the orthogonal basis are obtained in a different way. If one have a set of orthogonal basis such that

$$A\mathbf{p}_i^* \cdot A\mathbf{p}_j = \delta_{ij}, \tag{98}$$

The solution $\mathbf{x}$ is the residue projected into this basis (Note, here we do NOT assume the basis are normalized)

$$\begin{aligned} \mathbf{r}_0 &= \mathbf{b} - A\mathbf{x}_0 \\ \mathbf{x} &= \mathbf{x}_0 + \sum_{\infty} \frac{\mathbf{r}_0^* \cdot A\mathbf{p}_i}{\|A\mathbf{p}_i\|} \mathbf{p}_i \end{aligned} \tag{99}$$

So, the iteration is

$$\mathbf{x} \approx \mathbf{x}_k = \mathbf{x}_0 + \sum_{i=0}^{k} \frac{\mathbf{r}_0^* \cdot A\mathbf{p}_i}{\|A\mathbf{p}_i\|} \mathbf{p}_i \tag{100}$$

which can be obtained order by order as

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \frac{(\mathbf{b} - A\mathbf{x}_{k-1})^* \cdot A\mathbf{p}_{k-1}}{\|A\mathbf{p}_{k-1}\|} \mathbf{p}_{k-1} \tag{101}$$

There are GCR, ORTHOMIN, ORTHODIR. Both GCR and ORTHOMIN have oscillation (tested with random Gaussian pseudo fermion field and random gauge field), when iterating, sometimes, $\|\mathbf{p}^i\| \gg \|\mathbf{p}^{i-1}$, and $\|\mathbf{p}^{i+1}\| \ll \|\mathbf{p}^i$ and $\|\mathbf{p}^{i+2}\| \gg \|\mathbf{p}^{i+1}$, so we use ORTHODIR. The algorithm is

Algorithm 7 incomplete GCR with restart

```
x = b                                          ▷ Use b as trail and start
for i = 0 to r do                              ▷ restart r times
    r = b − Ax, p_0 = r
    for j = 0 to k − 1 do
        α = (Ap_j)^* · r/‖Ap_j‖^2
        x = x + αp_j
        r = r − αAp_j
        if ‖r‖ < ε then return x               ▷ Success
        end if
        p_{j+1} = Ap_j
        for k = j − l + 1 to j do
            β = (Ap_k)^* · A^2 p_j/‖Ap_k‖^2
            p_{j+1} = p_{j+1} + βp_k
        end for
    end for
end for
    return x                                   ▷ Failed with the closest result
```

Note that, GCR is much slower than GMRES and BiCGStab. A strategy to improve the speed is to restart quickly.

3.4 TFQMR

The QMR and TFQMR is based on Lanczos process [9]

Algorithm 8 Lanczos process

Prepare two vectors $\mathbf{v}_1^\dagger \mathbf{w}_1 = 1$
$\beta = \delta = 0$, $w_0 = v_0 = 1$
for $i = 0$ to m **do**
  $\mathbf{u} = A\mathbf{v}_i$
  $\alpha = \mathbf{u}^\dagger \mathbf{w}_i$
  $\mathbf{v}_{i+1} = \mathbf{u} - \alpha \mathbf{v}_i - \beta \mathbf{v}_{i-1}$
  $\mathbf{w}_{i+1} = A^\dagger \mathbf{w}_j - \alpha \mathbf{w}_j - \delta \mathbf{w}_{i-1}$
  $l = \mathbf{v}_{i+1}^\dagger \mathbf{w}_{i+1}$
  $\delta = \sqrt{|l|}, \beta = l/\delta$
  $\mathbf{w}_{i+1} = \mathbf{w}_{i+1}/\beta, \mathbf{v}_{i+1} = \mathbf{v}_{i+1}/\delta$
  $T[i+1, i+1] = \alpha, T[i+2, i+1] = \beta, T[i+1, i+2] = \delta$
end for
$T[m+1, m+1] = (A\mathbf{v}_{m+1})^\dagger \mathbf{w}_{m+1}$

Let T_m be the $m \times m$ part of $T_{m+1,m+1}$, one have

$$\begin{aligned} AV_m &= V_m T_m + T[m, m+1]\mathbf{e}_m^T \mathbf{v}_{m+1} \\ A^\dagger W_m &= W_m T_m^\dagger + T[m+1, m]\mathbf{e}_m^T \mathbf{w}_{m+1} \\ W_m^T A V_m &= T_m \end{aligned} \tag{102}$$

Use $AV_m = V_m T_m + T[m, m+1]\mathbf{e}_m^T \mathbf{v}_{m+1}$, one can implement a GMRES method, which is called QMR, or one can implement a FOM method, which is called two sided Lanczos method.

In the TFQMR, one no need to calculate $A^\dagger$ (There are also methods making use of the γ_5-hermiticity to simplify this.), which can be summarized as

---

Algorithm 9 TFQMR

$\mathbf{x} = \mathbf{x}_0$ is the initial guess
for $i = 0$ to r **do** ▷ Restart
  $\mathbf{u} = \mathbf{w} = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{d} = 0$, $\mathbf{r} = \mathbf{u}^*$
  $\mathbf{v} = A\mathbf{u}, \mathbf{u}_A = \mathbf{v}$
  $\theta = 0, \tau = |\mathbf{u}|$ ▷ $\theta, \tau \in \mathbb{R}$
  $\rho = \mathbf{r}^\dagger \mathbf{u}, \alpha = \eta = 0$ ▷ $\rho, \alpha, \eta \in \mathbb{C}$
  for $m = 0$ to $\ldots$ **do**
    if m is even **then**
      $\alpha = \frac{\rho}{\mathbf{r}^\dagger \mathbf{v}}$
    end if
    $\mathbf{w} = \mathbf{w} - \alpha \mathbf{u}_A$
    $\mathbf{d} = \mathbf{u} + \frac{\eta\theta}{\alpha}\mathbf{d}$
    $\theta = \frac{|\mathbf{w}|^2}{\tau^2}$, $c = \frac{1}{1+\theta}$, $\tau = \tau \times \sqrt{\theta c}$, $\eta = c\alpha$ ▷ $c \in \mathbb{R}$
    $\mathbf{x} = \mathbf{x} + \eta \mathbf{d}$
    if m is odd **then**
      $\rho' = \mathbf{r}^\dagger \mathbf{w}$, $\beta = \frac{\rho'}{\rho}$, $\rho = \rho'$ ▷ If ρ is too small, stop
      $\mathbf{u} = \mathbf{w} + \beta \mathbf{u}$
      $\mathbf{v} = \beta(\mathbf{u}_A + \beta \mathbf{v})$
      $\mathbf{u}_A = A\mathbf{u}$
      $\mathbf{v} = \mathbf{u}_A + \mathbf{v}$
    else
      $\mathbf{u} = \mathbf{u} - \alpha \mathbf{v}$
      $\mathbf{u}_A = A\mathbf{u}$
    end if
    Use $\mathbf{r}^\dagger \mathbf{v} < \epsilon$ to check whether to stop
  end for
  $\mathbf{x}_0 = \mathbf{x}$
end for

3.5 GCRO-DR and GMRES-MDR

'A comparison with the methods seen in the previous chapter indicates that in many cases, GMRES will be faster if the problem is well conditioned, resulting in a moderate number of steps required to converge. If many steps (say, in the hundreds) are required, then BICGSTAB and TFQMR may perform better. If memory is not an issue, GMRES or DQGMRES, with a large number of directions, is often the most reliable choice. The issue then is one of trading ribustness for memory usage. In general, a sound strategy is to focus on finding a good preconditioner rather than the best accelerator'. [9].

That might because the Krylov space will converge to the domain eigen-vector.

From Fig. 1. 9 of Ref. [15], the low mode is the most critical problem, so CLGLib first implement low mode deflation preconditioner.

In the following, we follow Ref. [16].

3.5.1 Brief introduction to deflation preconditioner

In short, the preconditioner means, one solve

$$M^{-1}Ax=M^{-1}b, \tag{103}$$

or

$$\left\{\begin{array}{l} AM^{-1}u=b \\ x=M^{-1}u \end{array}\right. \tag{104}$$

instead of $Ax=b$. If M is chosen carefully, it is usually faster.

Now, considering $A\in\mathbb{C}^{n\times n}$ and a matrix $Z\in\mathbb{C}^{n\times k}$ such that $Z=(v_1,v_2,\ldots,v_k)$ and each row is a vector $v_i\in\mathbb{C}^n$ such that $v_i^{\dagger}v_j=\delta_{ij}$. So Z acts like a Unitary matrix $Z^{\dagger}Z=\mathbb{I}^{k\times k}$. Then we can use Z to project A on a subspace, as

$$T=Z^{\dagger}AZ,\;\; Z^{\dagger}A=TZ^{\dagger} \tag{105}$$

so

$$\begin{aligned} &Ax=b\Rightarrow\left(\mathbb{I}+Z\left(T^{-1}-\mathbb{I}\right)Z^{\dagger}\right)Ax=\left(\mathbb{I}+Z\left(T^{-1}-\mathbb{I}\right)Z^{\dagger}\right)b\\ &\Rightarrow\left(A-ZZ^{\dagger}A\right)x+ZT^{-1}Z^{\dagger}Ax=b-ZZ^{\dagger}b+ZT^{-1}Z^{\dagger}b \end{aligned} \tag{106}$$

Note that $ZZ^{\dagger}\in\mathbb{C}^{n\times n}$ is not identity matrix (**also not a unitary matrix, but is an Hermitian matrix**). $T\in\mathbb{C}^{k\times k}$ is a small matrix. And then, one can solve

$$\begin{aligned} &ZT^{-1}Z^{\dagger}Ax=ZT^{-1}Z^{\dagger}b,\;\; Z^{\dagger}A=TZ^{\dagger}\\ &ZT^{-1}TZ^{\dagger}x=ZT^{-1}Z^{\dagger}b\\ &x=ZT^{-1}Z^{\dagger}b \end{aligned} \tag{107}$$

exactly, while solving $\left(A - ZZ^{\dagger}A\right)x = b - ZZ^{\dagger}b$ by iteration methods such as GMRES.

This is the so-called **subspace deflation**.

3.5.2 Brief intro to GCRO-DR

Start from Eq. (88). Assume after the first-step GMRES, we have the orthogonal-normal basis v_i which can be written as a matrix $V_m \in \mathbb{C}^{n\times m}, V_{m+1} \in \mathbb{C}^{n\times(m+1)}, H \in \mathbb{C}^{(m+1)\times m}$. On the other hand, will be introduced later, we have a set of deflation vectors, or a matrix $P_k \in \mathbb{C}^{m\times k}$, such that

$$\begin{aligned} &AV_m P_k = V_{m+1} H P_k \\ &\tilde{Y}_k \equiv V_m P_k \in \mathbb{C}^{n\times k} \end{aligned} \tag{108}$$

Then $\tilde{Y}_k$ is the deflation matrix.

Consider the matrix $HP_k = QR$, where QR is the QR factorization, with $Q \in \mathbb{C}^{(m+1)\times k}$ and $R \in \mathbb{C}^{k\times k}$. And define

$$C_k \equiv V_{m+1} Q \in \mathbb{C}^{n\times k}. \tag{109}$$

So, if R which is a small upper triangular matrix such that R^{-1} can be easily calculated, it is

$$\begin{aligned} &AV_m P_k = A\tilde{Y}_k = V_{m+1} H P_k = V_{m+1} Q R = C_k R \\ &C_k = A\tilde{Y}_k R^{-1} = AU, \quad U \equiv \tilde{Y}_k R^{-1} \in \mathbb{C}^{n\times k} \end{aligned} \tag{110}$$

Finally, the problem in GMRES Eq. (88) is changed as

$$\begin{aligned} &\tilde{U}_k = U_k D_k = U_k \left(\begin{array}{cccc} \frac{1}{\|\mathbf{u}_1\|} & 0 & 0 & 0 \\ 0 & \frac{1}{\|\mathbf{u}_2\|} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \frac{1}{\|\mathbf{u}_k\|} \end{array} \right) \in \mathbb{C}^{n\times k} \\ &V_m^{(1)} = (U_k, V_{m-k}) \in \mathbb{C}^{n\times m} \\ &V_{m+1}^{(2)} = (C_k, V_{m-k+1}) \in \mathbb{C}^{n\times(m+1)} \\ &H' = \left(\begin{array}{cc} D_k & B_{m-k} \\ 0 & H_{m-k} \end{array} \right) \in \mathbb{C}^{(m+1)\times m} \\ &AV_m^{(1)} = V_{m+1}^{(2)} H' \end{aligned} \tag{111}$$

where V are orthogonal-normal basis obtained in GMRES, and $B_{m-k} = AV_{m-k}$. Note that $B_{m-k} \in \mathbb{C}^{(m-k)\times k}$ but $H_{m-k} \in \mathbb{C}^{(m-k+1)\times(m-k)}$.

From Eq. (110), we find

- The subspace is k dimension subspace of m dimension Krylov space.
- With H, V and P known, we are able to calculate $QR = HP$, $U = V_m PR^{-1}$, $C = V_{m+1}Q$.

3.5.3 The choice of deflation subspace

In the above, we have assumed $P_k \in \mathbb{C}^{m\times k}$ is already known. Now we concentrate on this part.

Let $A \in \mathbb{C}^{n\times n}, V \in \mathbb{C}^{n\times k}$, **If V is formed as orthogonal normal basis of subspace** S, then, if $(\lambda, w \in \mathbb{C}^m)$ is eigen-pair of $V^\dagger AV$, $(\lambda, u = V^\dagger w \in \mathbb{C}^n)$ is eigen-pair of A.

$$\begin{aligned} & H_m p_i = \lambda_i p_i, \quad H_m = V_m^\dagger A V_m \\ & A\left(V_m p_i\right) = V_m H_m p_i = \lambda_i \left(V_m p_i\right) \end{aligned} \tag{112}$$

Therefor, **P_k is a matrix with k rows, and each row is a eigen-vector of H_m (H_m denoting the first m row of H_{m+1})**, then, VP_k is a matrix with k rows such that each row is a eigen-vector of A (approximately since $AV \approx VH \Rightarrow H \approx V^\dagger AV$).

The first GMRES cycle will generate H_m (denoting the first m row of H_{m+1}), and $H_m\omega = \theta\omega$ is solved. However, starting from the second cycle of GCRO-DR, it is not $AV_m = V_{m+1}H_{m+1}$ but $AV_m^{(1)} = V_{m+1}^{(2)}H'_{m+1}$, such that **$V_{m+1}^{(2)}$ are orthogonal basis but $V_m^{(1)}$ are not orthogonal basis! (Therefor $V^\dagger AV$ does not hold!)**. In this case, it is another eigen-problem which should be solved. This will be listed below without explain.

By Ref. [16], there are three strategies, Ritz eigen-vector (REV), harmonic Ritz eigen vector (HEV) and singular value decomposition (SVD). Either it is $REV > HEV > SVD$ or $SVD > HEV > REV$, so we only list REV and SVD here.

Note that $\tilde{U}_k$ is the normalized U_k, H_{m+1} means the H_{m+1} of GMRES procedure, and H'_{m+1} means H'_{m+1} obtained in GCRO-DR procedure, H_m and H'_m means the upper m rows of H_{m+1} and H'_{m+1}.

- REV

The k small eigen value of m, such that m is

$$\left\{\begin{array}{l} H_m\omega=\theta\omega,\\ \left(\begin{array}{cc}\tilde{U}_k^{\dagger}C_k & \tilde{U}_k^{\dagger}V_{m-k+1}\\ 0 & (I_{m-k},0)\end{array}\right)H'_{m+1}\omega=\theta\left(\begin{array}{cc}\tilde{U}_k^{\dagger}\tilde{U}_k & \tilde{U}_k^{\dagger}V_{m-k}\\ V_{m-k}^{\dagger}\tilde{U}_k & I_{m-k}\end{array}\right)\omega,\end{array}\right. \tag{113}$$

- HEV

The k **larger** eigen value of m, such that m is

$$\left\{\begin{array}{l} H_m{}^{\dagger}\omega=\theta H_{m+1}{}^{\dagger}H_{m+1}\omega,\\ H'_{m+1}{}^{\dagger}\left(\begin{array}{cc}C_k^{\dagger}\tilde{U}_k & 0\\ V_{m-k+1}^{\dagger}\tilde{U}_k & \left(\begin{array}{c}I_{m-k}\\ 0\end{array}\right)\end{array}\right)\omega=\theta H'_{m+1}{}^{\dagger}H'_{m+1}\omega,\end{array}\right. \tag{114}$$

- SVD

The k small eigen value of m, such that m is

$$\left\{\begin{array}{l} H_m^{\dagger}H_m\omega=\theta\omega,\\ H'_{m+1}{}^{\dagger}H'_{m+1}\omega=\theta\left(\begin{array}{cc}\tilde{U}_k^{\dagger}\tilde{U}_k & 0\\ 0 & I_{m-k}\end{array}\right)\omega,\end{array}\right. \tag{115}$$

Although H_m is usually a small matrix, we still need to known how to calculate the eigen-value and eigen-vectors.

Note that the second line of REV and SVD, and both line of HEV, that is a **generalized eigen-value problem (GEV)**.

3.5.4 Eigen solver

The eigen solver is implemented following Ref. [17].

There are many strategies. The most common algorithm is to transform a matrix to a **Hessenberg matrix**.

- Householder reflection

Tested that householder reduction is faster than symmetric or unsymmetric Lanczos method when the matrix is large. On the other hand, for a Hermitian matrix, Householder can also produce Hermitian tri-diagonal matrix.

Note that this might be not true when the matrix is huge, and Hessenberg reduction is not a full reduction. In our case, we concentrate on matrix with $5 < m < 50$**. Tested when about** $7 < m < 30$ ($30 \times 30 \approx 1024$ **is the maximum thread count on test machine), Householder is faster.**

Also, as tested, the quality of QR factorization affects the QR iteration very much. At the same time, compared with QR iteration, the QR factorization is relatively cheap, so we also use Householder to do the QR factorization.

The householder reduction is to insert zeros into a vector, which can be briefly written as

$$\mathbf{v} = \begin{pmatrix} x_1 + e^{i \arg x_1}|\mathbf{x}| \\ x_2 \\ \ldots \\ x_n \end{pmatrix}, \;\; U = \mathbb{I} - \frac{2\mathbf{v}\mathbf{v}^\dagger}{\mathbf{v}^\dagger \mathbf{v}}, \;\; U \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix} = \begin{pmatrix} \frac{|x_1|}{x_1^*}|\mathbf{x}| \\ 0 \\ \ldots \\ 0 \end{pmatrix}, \tag{116}$$

Note that U is at the same time unitary and Hermitian. Since it is unitary, it can be used as QR factorization, $A = QR$ where Q is unitary and R is upper triangular, and to transform a matrix to Henssenberg matrix $A = U^\dagger H U$, where U is unitary and H is upper Henssenberg matrix.

The algorithm is not listed, the procedure can be written as

$$\begin{aligned}
&A_0 = U_0^\dagger U_0 A_0 = U_0^\dagger A_1, \; U_0 A_0 = \left(\mathbb{I} - \frac{2\mathbf{v}\mathbf{v}^\dagger}{\mathbf{v}^\dagger\mathbf{v}}\right) A_0 = A_1 = \begin{pmatrix} + & + & + & + \\ 0 & + & + & + \\ 0 & + & + & + \\ 0 & + & + & + \end{pmatrix} \\
&A_0 = U_0^\dagger U_1^\dagger U_1 A_1, \; U_1 A_1 = \begin{pmatrix} \mathbb{I}_1 & 0 \\ 0 & \mathbb{I} - \frac{2\mathbf{v}\mathbf{v}^\dagger}{\mathbf{v}^\dagger\mathbf{v}} \end{pmatrix} A_1 = A_2 = \begin{pmatrix} + & + & + & + \\ 0 & + & + & + \\ 0 & 0 & + & + \\ 0 & 0 & + & + \end{pmatrix} \\
&A_0 = U_0^\dagger U_1^\dagger U_2^\dagger R, \; R = U_2 A_2 = \begin{pmatrix} \mathbb{I}_2 & 0 \\ 0 & \mathbb{I} - \frac{2\mathbf{v}\mathbf{v}^\dagger}{\mathbf{v}^\dagger\mathbf{v}} \end{pmatrix} A_2 = R = \begin{pmatrix} + & + & + & + \\ 0 & + & + & + \\ 0 & 0 & + & + \\ 0 & 0 & 0 & + \end{pmatrix}
\end{aligned} \tag{117}$$

Similarly, note that if only insert zeroes from the second row

$$UA = \begin{pmatrix} \mathbb{I}_1 & 0 \\ 0 & \mathbb{I} - \frac{2\mathbf{v}\mathbf{v}^\dagger}{\mathbf{v}^\dagger\mathbf{v}} \end{pmatrix} A = \begin{pmatrix} + & + & + & + \\ + & + & + & + \\ 0 & + & + & + \\ 0 & + & + & + \end{pmatrix} \tag{118}$$

then

$$UAU^\dagger = UA \begin{pmatrix} \mathbb{I}_1 & 0 \\ 0 & @ \end{pmatrix} = \begin{pmatrix} + & + & + & + \\ + & + & + & + \\ 0 & + & + & + \\ 0 & + & + & + \end{pmatrix} \begin{pmatrix} \mathbb{I}_1 & 0 \\ 0 & @ \end{pmatrix} = \begin{pmatrix} + & +@ & +@ & +@ \\ + & +@ & +@ & +@ \\ 0 & +@ & +@ & +@ \\ 0 & +@ & +@ & +@ \end{pmatrix} \tag{119}$$

So that it is kept Hensenberg.

- Shifted QR iteration

Let $A = U_0^\dagger H_0 U_0$ where H is a Henssenberg matrix, then, let $H_0 = U_1 R$, it can be shown that $H_1 = RU_1 = U_1^\dagger(U_1 R)U_1$ is still a Henssenberg.

Also, $H = U_1 H_1 U_1^\dagger$, so $A = (U_0^\dagger U_1)H_1(U_1^\dagger U_0)$.

So, H_1 has same eigen-value as A.

Apart from that, H_i can approach a upper triangular matrix. It is noted that, if the QR factorization is performed to a shifted matrix $H - \sigma I$, where σ is an approximate eigen-value of H, it will converge much fast.

In CLGLib, we use Wilkinson shift, which is the eigen-value of the right-bottom 2×2 irreducible matrix, and is the one closer to the right-bottom corner element. It can be written as

Algorithm 10 Shifted QR Iteration

for H is not a triangular **do**

σ be the eigen-value of the 2×2 matrix of right-bottom matrix which is closer to the right bottom element.

$QR = H - \sigma I$

$H' = RQ + \sigma I$

if $H_{n-1,n} \approx 0$ **then**

Reduce to a $n - 1$ Henssenberg matrix problem.

end if

end for

Once the upper triangular matrix is obtained, the eigen-values are just the diagonal elements.

- Implicit shifted QR iteration

The **Implicit shifted QR iteration** sometimes also called **Double shifted QR iteration** or **Double shifted QR iteration**.

The details are not listed here, it uses Householder to chase the zero to the bottom and right, it is a little bit better convergent, and is said to be more stable. It can be written as

Algorithm 11 Implicit shifted QR iteration

for T a Hessenberg matrix with $n \geq 3$. (In the case of $n = 2$, the eigen-value can be directly obtained.) **do**

H a irreducible Hessenberg matrix with $n \geq 3$. $T = \begin{pmatrix} + & + & + \\ 0 & H & + \\ 0 & 0 & + \end{pmatrix}$ ▷ In the case of $n = 2$, the eigen-value can be directly obtained.

$H_{2\times2}$ be the 2×2 matrix of right-bottom matrix. $s = \mathrm{tr}(H_{2\times2})$ and $t = \det(H_{2\times2})$

$x = H_{1,1}(H_{1,1} - s) + H_{1,2}H_{2,1} + t$

$y = H_{2,1}(H_{1,1} + H_{2,2} - s)$

$z = H_{2,1}H_{3,2}$

$H' = RQ + \sigma I$

for $k = 0$ to $n - 3$ **do**

h be Householder matrix to zero $\mathbf{v} = (x, y, z)^T \to (|\mathbf{v}|, 0, 0)^T$.

$q = \max(1, k)$, $H(k + 1 : k + 3, q : n) = hH(k + 1 : k + 3, q : n)$.

$r = \min(k + 4, n)$, $H(1 : 4, k + 1 : k + 3) = H(1 : 4, k + 1 : k + 3)h^\dagger$. ▷ Note that $h^\dagger = h$

$x = H(k + 2, k + 1), y = H(k + 3, k + 1)$

if $k < n - 3$ **then**

$z = H(k + 4, k + 1)$

end if

end for

h be Householder matrix to zero $\mathbf{v} = (x, y)^T \to (|\mathbf{v}|, 0)^T$.

$H(n-1 : n, n-2 : n) = hH(n-1 : n, n-2 : n)$, $H(1 : n, n-1 : n) = H(1 : n, n-1 : n)h^\dagger$.

end for

- Inverse power iteration

Once the eigen-values are obtained, one can calculate the approximate eigen-vector correspond the the eigen-value using inverse power iteration. The inverse power iteration performs well with the original matrix A.

Algorithm 12 Inverse power Iteration

v is a normalized vector.
for $\|(A-\sigma I)\mathbf{v}\| > \epsilon$ **do**
  $QR = (A - \sigma I)$
  $\mathbf{v} = R^{-1}Q^{\dagger}\mathbf{v}$
  $\mathbf{v} = \mathbf{v}/\|\mathbf{v}\|$
end for

The R is upper triangular, so R^{-1} is just a modification of Algorithm. 5.

Algorithm 13 Backward substitution

for $i = k-1$ to 0 **do**
  for $j = i+1$ to $k-1$ **do**
    $\mathbf{y}[i] -= r[i,j]\mathbf{y}[j]$
  end for
  $\mathbf{y}[i] = \mathbf{y}[i]/r[i,i]$
end for
  return $\mathbf{u}[k] = \mathbf{y}[k]$.

- Eigen vector of upper triangular matrix

The inverse power iteration is incompatible with upper triangular matrix, because $R-\lambda I$ is singular, for the inverse power iteration, $R-\lambda I$ is only nearly singular, however, for a upper triangular, it is almost exactly a singular. Although one can shift the eigen value a little bit, but one can also obtain eigen vector exactly. by the procedure below.

Suppose

$$\begin{pmatrix} r_{1,1}-\lambda_k & \dots & r_{1,k-1} \\ 0 & \dots & \dots \\ 0 & 0 & r_{k-1,k-1}-\lambda_k \end{pmatrix} \begin{pmatrix} x_1 \\ \dots \\ x_{k-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ \dots \\ y_{k-1} \end{pmatrix} \tag{120}$$

$(R-\lambda_k\mathbb{I})\mathbf{x}=0$ can be written as

$$\begin{pmatrix} r_{1,1}-\lambda_k & \ldots & r_{1,k-1} & r_{1,k} & \ldots \\ 0 & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & r_{k-1,k-1}-\lambda_k & r_{k-1,k} & \ldots \\ 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & \ldots \end{pmatrix} \begin{pmatrix} x_1 \\ \ldots \\ x_{k-1} \\ 1 \\ 0 \\ \ldots \end{pmatrix} = \begin{pmatrix} y_1 + r_{1,k} \\ \ldots \\ y_{k-1}+r_{k-1,k} \\ 0 \\ \ldots \end{pmatrix} = 0 \quad (121)$$

leads to the equation

$$\begin{pmatrix} r_{1,1}-\lambda_k & \ldots & r_{1,k-1} \\ 0 & \ldots & \ldots \\ 0 & 0 & r_{k-1,k-1}-\lambda_k \end{pmatrix} \begin{pmatrix} x_1 \\ \ldots \\ x_{k-1} \end{pmatrix} = \begin{pmatrix} -r_{1,k} \\ \ldots \\ -r_{k-1,k} \end{pmatrix} \quad (122)$$

which can be solved using backward shift, i.e. Algorithm. 13.

- Generalized eigen-value problem

The generalized eigen-value problem can be transformed to a eigen-value problem

$$A\mathbf{v}=\lambda B\mathbf{v} \Rightarrow B=QR \Rightarrow R^{-1}Q^{\dagger}A\mathbf{v}=\lambda\mathbf{v} \quad (123)$$

3.5.5 Implementation of GCRO-DR

Now, we concentrate on the implementation of GCRO-DR. First of all, we need to know how to apply $\mathbf{x}-AB^{\dagger}\mathbf{v}$, where $A,B\in\mathbb{C}^{n\times k}$ and $\mathbf{v}\in\mathbb{C}^n$.

Algorithm 14 $\mathbf{x}=\mathbf{x}-AB^{\dagger}\mathbf{v}$

for $i=0$ to $k-1$ **do**
  $\mathbf{x}=\mathbf{x}-\left(\mathbf{b}_k^{\dagger}\mathbf{x}\right)\mathbf{a}_k$
end for
  return x

The second thing is QR decompose of $\mathbb{C}^{n\times k}$ and $\mathbb{C}^{(m+1)\times k}$ matrix. For the $\mathbb{C}^{n\times k}$ matrix, the usually Arnoldi with modified Gram-Schmidt, i.e. Algorithm. 3 can be used.

Algorithm 15 modified Gram-Schmidt for QR factorization decompose of $A\tilde{Y}_k$

for $i = 0$ to $k-1$ **do**
  $y_i = Ay_i$
end for
$\mathbf{v}^{(0)} = y_0/\|\mathbf{y}_0\|$
for $i = 0$ to $k-1$ **do**
  $\mathbf{w} = \mathbf{y}_{i+1}$
  for $j = i+1$ to $k-1$ **do**
    $c = \mathbf{v}^{(j)^*} \cdot \mathbf{w}$
    $\mathbf{w} -= c\mathbf{v}^{(j)}$
    $r[j,i] = c$
  end for
  $\mathbf{v}^{(i+1)} = \mathbf{w}/r[i+1,i+1]$
end for
  return $Q = (\mathbf{v}_0, \ldots, \mathbf{v}_{k-1})$, $R = r[i,j]$.

Finally we have to calculate YR^{-1}. This is a forward substitution.

$$U = YR^{-1}, \;\; UR = Y, \;\; R^T U^T = Y^T \;\; U^T = \left(R^T\right)^{-1} Y^T. \tag{124}$$

3.5.6 Implement of GCRO-DR

We present pseudo-code of GCRO-DR can be found in Ref. [16]. The only difference is that we always make sure C_k and V_{m-k+1} are orthogonal to each other. It can be written as

Algorithm 16 GCRO-DR

if U_k is defined from solving a previous linear system **then**

Let $[Q, R] = AU_k$ be QR decomposition or AU_k.

$C_k = Q$.

$U_k = U_k R^{-1}$.

$\mathbf{r}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}$

else

Perform GMRES to get $W_{m+1} = (C_k, V_{m-k+1})$

Update $\mathbf{x}^{(0)}, \mathbf{r}^{(0)}$ as $\mathbf{x}^{(0)} = \mathbf{x}^{(0)} + V_m y, \mathbf{r}^{(0)} = V_{m+1}(\beta \mathbf{e}_1 - H_{m+1} y)$, which is in fact part of GMRES.

Compute eigen-vector problem and obtain $P_k \in \mathbb{C}^{m \times k}$.

$U_k = V_m P_k$

Let $[Q, R] = H_{m+1} P_k$ be QR decomposition.

$C_k = V_{m+1} Q$

$U_k = U_k R^{-1}$

end if

for $\hat{i} = 1$ to r **do** ▷ restart r times.

$\mathbf{x}^{(i-1)} = \mathbf{x}^{(i-1)} + U_k C_k^\dagger \mathbf{r}^{(i-1)}$

$\mathbf{r}^{(i-1)} = \mathbf{r}^{(i-1)} - C_k C_k^\dagger \mathbf{r}^{(i-1)}$

Reset $H_{m+1} = 0$. $W_{m+1}(k) = V_{m-k+1}(0) = \mathbf{r}^{(i)} / \|\mathbf{r}^{(i)}\|$.

$H_{m+1}(k, k) = 1/\|U_k\|$, normalize U_k.

Perform Arnoldi procedure on matrix $(1 - CC^\dagger)A$, to obtain V_{m-k+1}, and set $H_{k:m+1,k:m}$. And $H_{0:k,m} = C_k^\dagger A V_{m-k}$. $\hat{V} = (U_k, V_{m-k})$ and $W_{m+1} = (C_k, V_{m-k+1})$.

Solve arg min $\|\|r\|\mathbf{e}_k - H_{m+1} y\|$.

$\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \hat{V}_m y$, $\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)} - W_{m+1} H_{m+1} y)$. ▷ Check the error here. If reach the criterion, return.

Compute eigen-vector problem and obtain $P_k \in \mathbb{C}^{m \times k}$.

$U_k = V_m P_k$

Let $[Q, R] = H_{m+1} P_k$ be QR decomposition.

$C_k = V_{m+1} Q$

$U_k = U_k R^{-1}$

end for

3.5.7 Implement of GMRES-MDR

The GMRES-MDR is almost the same as GCRO-DR, except for 3 things.

1. It set a threshold on eigen-values to decrease k if a larger k is not necessary.
2. It check the speed of convergence to switch between REV and SVD.
3. At first iteration, if U_k is defined, it use another algorithm to obtain U_k and C_k.

Algorithm 17 First iteration of GMRES-MDR if U_k is defined.

$[Q, R] = U_k$
if REV **then**
  $Q^\dagger AQ\omega = \theta\omega$
end if
if HEV **then**
  $Q^\dagger A^\dagger AQ\omega = \theta Q^\dagger A^\dagger Q\omega$
end if
if SVD **then**
  $Q^\dagger A^\dagger AQ\omega = \theta^2\omega$
end if
$U_k = Q\omega_k$

We only implement the third because the first two can be tunable by parameters.

3.5.8 Test of GCRO-DR and GMRES-MDR

It is tested that, for both GCRO-DR and GMRES-MDR are suitable for the low-mode case.

We run with unitary gauge and $\kappa = 0.1249$. ($\kappa_c = 0.125$). GCRO-DR with $m = 16$ and $k = 4$ will run even faster than $dim = 50$ GMRES.

However, if it is not the low-mode case, GMRES is faster.

3.6 Even-odd preconditioner

The equation for the D operator can be written as

$$\begin{pmatrix} D_{ee} & D_{eo} \\ D_{oe} & D_{oo} \end{pmatrix} \begin{pmatrix} z_e \\ z_o \end{pmatrix} = \begin{pmatrix} \phi_e \\ \phi_o \end{pmatrix} \tag{125}$$

Where D_{oe} means $D_{o\leftarrow e}$. Often, D_{oo}^{-1} and D_{ee}^{-1} is very easy to calculate. In the case of Wilson-Dirac fermion without $\mathcal{O}(a)$ improvement, it is $\mathbb{I}$.

It can be divided into 3 step to solve this equation

1. Calculate $\tilde{\phi}_e = \phi_e - D_{eo}D_{oo}^{-1}\phi_o$.
2. Solve $(D_{ee} - D_{eo}D_{oo}^{-1}D_{oe})\, z_e = \tilde{\phi}_e$.
3. $z_o = D_{oo}^{-1}(\phi_o - D_{oe}z_e)$.

This is because (**Note the order of D's should not change.**)

$$\begin{pmatrix} D_{ee} & D_{eo} \\ D_{oe} & D_{oo} \end{pmatrix}\begin{pmatrix} z_e \\ z_o \end{pmatrix} = \begin{pmatrix} 1 & D_{eo}D_{oo}^{-1} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} D_{ee} - D_{eo}D_{oo}^{-1}D_{oe} & 0 \\ D_{oe} & D_{oo} \end{pmatrix}\begin{pmatrix} z_e \\ z_o \end{pmatrix} = \begin{pmatrix} \phi_e \\ \phi_o \end{pmatrix} \tag{126}$$

Using

$$\begin{pmatrix} 1 & -D_{eo}D_{oo}^{-1} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & D_{eo}D_{oo}^{-1} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{127}$$

So that

$$\begin{pmatrix} D_{ee} - D_{eo}D_{oo}^{-1}D_{oe} & 0 \\ D_{oe} & D_{oo} \end{pmatrix}\begin{pmatrix} z_e \\ z_o \end{pmatrix} = \begin{pmatrix} 1 & -D_{eo}D_{oo}^{-1} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \phi_e \\ \phi_o \end{pmatrix} = \begin{pmatrix} \tilde{\phi}_e \\ \phi_o \end{pmatrix} \tag{128}$$

Which leads to two equations

$$\begin{aligned} &(D_{ee} - D_{eo}D_{oo}^{-1}D_{oe})z_e = \tilde{\phi}_e, \\ &D_{oe}z_e + D_{oo}z_o = \phi_o, \end{aligned} \tag{129}$$

which leads to step 1,2,3.

Because the D matrix is sparse, For n sites, $D\phi$ is not $O(n^2)$ but $O((c+1)n)$ ($c = 2D$ for neighbour sites), so in the even-odd preconditioner, D_{eo} is not $O(n^2/4)$ but $O(cn/2)$. Solving $(D_{ee} - D_{eo}D_{oo}^{-1}D_{oe})\, z_e = \tilde{\phi}_e$, one need to calculate D_{eo} and D_{oe}, so it is not $O(n^2/2)$ but $O(cn)$, as a result the effect of even-odd preconditioner is limited.

There is one more problem: for HMC, one need to solve mainly $(DD^{\dagger})^{-1}$. Which can not be even-odd decomposed, so one need to at first solve D^{-1} then $(D^{\dagger})^{-1}$, the solution in the last step cannot be set as an initial guess.

Note, for some reason I do not know, the GCRO-DR and GMRES-MDR not yet support even-odd preconditioner.

Note, for periodic boundary condition, and odd extent lattice, the $D_{ee} \neq \mathbb{I}$ and $D_{oo} \neq \mathbb{I}$. For example, for $4\times 4\times 3\times 3$, $(0,0,0,1)$ is connected with $(0,0,2,1)$. Note that, D_{oo} is neither a diagonal matrix, so D_{oo}^{-1} need to be solved, which makes the even-odd inefficient, so we do NOT support such case.

3.7 The multi-shift solver

The multi-shift solver are used to solve $(A+a_n)x=b$ for different a_n at once.

Note that, it is only possible when A is on the numerator, for example, when using Arnoldi or Lanczos algorithm.

3.7.1 The multi-shift version of GMRES

We still use Algorithm 3 to build $\mathbf{v}_i$, such that

$$\mathbf{v}_i^*\cdot\mathbf{v}_j=\delta_{ij},\quad A\mathbf{v}_{i-1}=\sum_{j=0}^{i}h[j,i-1]\mathbf{v}_j, \tag{130}$$

holds.

However, now we want to solve

$$((A+a_n)v)_k=v_{k+1}H \tag{131}$$

For that to happen, we need

$$A\mathbf{v}_{i-1}=\sum_{j=0}^{i}h[j,i-1]\mathbf{v}_j\to(A+a_n)\mathbf{v}_{i-1}=\sum_{j=0}^{i}\left(h[j,i-1]+a_n\delta_{j,i-1}\right)\mathbf{v}_j \tag{132}$$

Note that the sum over j will contain $\mathbf{v}_{i-1}$, so simply use $\hat{H}=h[j,i-1]+a_n\delta_{j,i-1}$ instead of $H=h[j,i-1]$ one will arrive at Eq. 130.

When considering the restart, one should assume the residue of the shifted system is collinear to the seed system, $\hat{r}_n=\beta_n r$ which can be guaranteed when $x_0=0$ and for the first iteration, $\hat{r}_n^{k=0}=r=b$, where k is index of restart, r is the residue of seed system.

For the second iteration (after the first restart), to make sure again $\hat{r}_n=\beta_n r$, one need to solve

$$
\begin{aligned}
&\beta = |\mathbf{r}^{k-1}| \\
&\mathbf{z}_{m+1} = \beta \mathbf{e}_1 - H_{(m+1)\times m} \cdot \mathbf{y}_m \\
&\left(\hat{H}_{(m+1)\times m} | \mathbf{z}_{m+1}\right)_{(m+1)\times(m+1)} \cdot \begin{pmatrix} \hat{y}_m \\ \beta_n^k \end{pmatrix} = \beta_n^{k-1} \beta \mathbf{e}_1
\end{aligned}
\tag{133}
$$

where k is the index of restart, $\mathbf{r}$ is the residue of the seed system (zero shift system), $\hat{y}_m$ is the solution of $argmin(\beta^{k-1}\mathbf{e}_1 - Hy_m)$ (GMRES of seed system).

Then, $x_n = x_n + \sum_j \hat{y}_j v_j$.

It can be summarized as [18]. Let the seed system be $Ax = b$, than

---

Algorithm 18 shifted GMRES

$\mathbf{x} = 0, \mathbf{x}_n = 0, \mathbf{r} = \mathbf{b}, \beta_n = 1$
for $i = 0$ to $k - 1$ **do**
  $\mathbf{r} = \mathbf{b} - A\mathbf{x}$, $\beta = |\mathbf{r}|$
  Solve $A\mathbf{x} = \mathbf{b}$, obtain $\mathbf{v}_m$, $H_{m+1,m}$ and y_m
  Update $\mathbf{x}$ use $\mathbf{v}_m$ and y_m as $\mathbf{x} = \mathbf{x} + \sum \mathbf{v}_j y_j$
  $z_{m+1} = \beta\mathbf{e}_1 - H_{m+1,m} y_m$
  for $n = 0$ to n **do**
    The left $(m+1) \times m$ of $\hat{H}_{m+1,m+1}$ is $H_{m+1,m} + a_n \mathbb{I}_{m,m}$
    The right column of $\hat{H}_{m+1,m+1}$ is z_{m+1}
    Solve $\hat{H}_{m+1,m+1}\hat{y}_{m+1} = \beta_n \beta \mathbf{e}_1$ ▷ use QR factorization or Givens rotation
    Update $\beta_n = \hat{y}_{m+1}[m+1]$ ▷ β_n is complex number
    Update $\mathbf{x}_n$ use $\mathbf{v}_m$ and first m element of $\hat{y}_{m+1}$ as $\mathbf{x}_n = \mathbf{x}_n + \sum \mathbf{v}_j \hat{y}_j$
  end for
end for

3.7.2 The multi-shift FOM

FOM is very like GMRES, but for the shifted system, FOM is tested to be much faster. It can be summarized as [19]

Algorithm 19 shifted FOM

$\mathbf{x}_n = 0, \mathbf{r}^{(k=0)} = \mathbf{b}, \beta_n = |\mathbf{b}|$
for $i = 0$ to $k-1$ **do**
  Solve Krylov space for $A\mathbf{x} = \mathbf{b}$, obtain $\mathbf{v}_m, \mathbf{v}_{m+1}$ and $H_{m,m+1}$ ▷ Not solve $\beta\mathbf{e}_1 - Hy$
  $\mathbf{r}^{(k)} = \mathbf{v}_{m+1}$
  for $n = 0$ to n **do**
    if $|\beta_n| > \epsilon$ **then** ▷ β_n is complex number
      Solve $\left(H_{m,m} + a_n\mathbb{I}_m\right) y_m = \beta_n\mathbf{e}_1$ ▷ use Givens rotation
      $\beta_n = y_m[m] - h[m, m+1]$
      Update $\mathbf{x}_n$ use $\mathbf{v}_m$ and y_m as $\mathbf{x}_n = \mathbf{x}_n + \sum \mathbf{v}_j y_j$
    end if
  end for
end for

where $H_{m,m}$ is the upper $m \times m$ square matrix of $H_{m,m+1}$, and $h[m, m+1]$ is the last element of $H_{m,m+1}$.

3.7.3 The multi-shift BiCGStab

The speed of BiCGStab-m is similar as multi-shift GMRES. However, for a fixed number of shift, the memory is fixed (the accuracy is independent of memory)

The algorithm is from Ref. [20], which can be written as

Algorithm 20 multi-shift BiCGStab

$\mathbf{x}^{(n)} = 0, \mathbf{s} = \mathbf{s}^{(n)} = \mathbf{r} = \mathbf{b},\ \mathbf{w}_0 = \mathbf{w} = \mathbf{r}^\dagger,\ \mathbf{s}_A = A\mathbf{s}$

$\beta = \alpha = 0,\ \delta = \mathbf{w}_0^\dagger \mathbf{r},\ \phi = \frac{\mathbf{w}_0^\dagger \mathbf{s}_A}{\delta}$ $\triangleright$ $\beta, \alpha, \delta, \phi \in \mathbb{C}$

$\beta^{(n)} = \zeta^{(n)} = \zeta_p^{(n)} = \rho^{(n)} = \chi^{(n)} = s^{(n)} = 1$ $\triangleright$ $\beta^{(n)}, \zeta^{(n)}, \zeta_p^{(n)}, \rho^{(n)}, \chi^{(n)} \in \mathbb{C}, s^{(n)} \in \mathbb{R}$

for $i = 0$ to m **do**

  $\beta_1 = -\frac{1}{\phi}$

  for $n = 0$ to n with $s^{(n)} > \epsilon$ **do**

    if $i = 0$ **then**

      $\zeta^{(n)} = \frac{1}{1 - a_n \beta_1}$

      $\beta^{(n)} = \zeta^{(n)} \beta_1$

    else

      $\zeta_p = \zeta_p^{(n)},\ \zeta_p^{(n)} = \zeta^{(n)},\ d = \zeta_p \beta$

      $\zeta^{(n)} = \frac{d\zeta_p^{(n)}}{\alpha\beta_1(\zeta_p - \zeta_p^{(n)}) + d(1 - a_n\beta_1)}$

      $\beta^{(n)} = \frac{\beta_1 \zeta^{(n)}}{\zeta_p^{(n)}}$

    end if

  end for

  $\beta = \beta_1$

  $\mathbf{w} = \mathbf{r} - \beta \mathbf{s}_A,\ \mathbf{w}_A = A\mathbf{w}$

  $\chi = \frac{\mathbf{w}_A^\dagger \mathbf{w}}{\mathbf{w}_A^\dagger \mathbf{w}_A}$

  for $n = 0$ to n with $s^{(n)} > \epsilon$ **do**

    $d = \frac{1}{1 + a_n \chi},\ \chi^{(n)} = \frac{\chi}{d}$

    $d_1 = \chi^{(n)} \rho^{(n)},\ d_2 = d_1 \zeta^{(n)}$

    $\mathbf{x}^{(n)} = \mathbf{x}^{(n)} - \beta^{(n)} \mathbf{s}^{(n)} + d_2 \mathbf{w}$

    $\mathbf{s}^{(n)} = \mathbf{s}^{(n)} - \frac{d_2}{\beta^{(n)}} \mathbf{w} + \frac{d_1 \zeta_p^{(n)}}{\beta^{(n)}} \mathbf{r}$

    $\rho^{(n)} = \frac{\rho^{(n)}}{d}$

  end for

  $\mathbf{r} = \mathbf{w} - \chi \mathbf{w}_A$

  $\delta_1 = \mathbf{w}_0^\dagger \mathbf{r},\ \alpha = -\frac{\beta\delta_1}{\delta\chi},\ \delta = \delta_1$

  for $n = 0$ to n with $s^{(n)} > \epsilon$ **do**

    $\alpha^{(n)} = \frac{\alpha \beta^{(n)} \zeta^{(n)}}{\beta \zeta_p^{(n)}}$

    $\mathbf{s}^{(n)} = \zeta^{(n)} \rho^{(n)} \mathbf{r} + \alpha^{(n)} \mathbf{s}^{(n)}$

    $s^{(n)} = |\mathbf{s}^{(n)}|$

  end for

  $\mathbf{s} = \mathbf{r} + \alpha(\mathbf{s} - \chi \mathbf{s}_A)$ $\triangleright$ If $|\mathbf{s}|$ is small, quit.

  $\mathbf{s}_A = A\mathbf{s}$

  $\phi = \frac{\mathbf{w}_0^\dagger \mathbf{s}_A}{\delta}$

end for

It is tested that multi-shift FOM is faster than BiCGStab and GMRES. However, the memory usage of FOM and GMRES grow with m, the dimension of Krylov space, and grow as n vectors, the number of shifted a_n as $m + n$. The memory of BiCGStab does not grow with accuracy but grow with $2n$ vectors.

4 Miscellaneous topics

4.1 Gauge Fixing

4.1.1 Introduction of FFT before start

A brief introduction of FFT.

FFT is to calculate Discrete Fourier Transform (DFT), in 1D, it is

$$\begin{aligned}\tilde{x}_m &= \sum_n x_n W_N^{mn}\\ W_N^j &\equiv \exp(-i\frac{2\pi j}{N})\end{aligned} \tag{134}$$

- Cooley-Tukey mapping

Let $N = N_1 \times N_2$, we first calculate DFT of subset $I_{n_1} = \{n_2 N_1 + n_1\}$, such that

$$\tilde{x}_m = \sum_{n_1} S_{n_1}, \quad S_{n_1} = \sum_{n_2} x_{n_2 N_1 + n_1} W_N^{m(n_2 N_1 + n_1)} \tag{135}$$

Note that, S_{n_1} can be further factorized as

$$\tilde{x}_m = \sum_{n_1} W_N^{mn_1} S'_{n_1}, \quad S'_{n_1} = \sum_{n_2} x_{n_2 N_1 + n_1} W_N^{mn_2 N_1} \tag{136}$$

then, note that, N can be divided by N_1 (the result is N_2), so $W_N^{mn_2 N_1} = W_{N_2}^{mn_2}$, so S' is just DFT of subset I_{n_1}. Then, we can also decompose

$$m = m_1 N_2 + m_2 \tag{137}$$

to write

$$\tilde{x}_{m_1 N_2 + m_2} = \sum_{n_1} W_N^{n_1(m_1 N_2 + m_2)} S'_{n_1} = \sum_{n_1} W_{N_1}^{n_1 m_1} W_N^{n_1 m_2} S'_{n_1} \tag{138}$$

The $W_N^{n_1 m_2}$ is twiddle factor, after 'twiddle', $S''_{n_1} = W_N^{n_1 m_2} S'_{n_1}$, the result is again a DFT with size N_1

$$\tilde{x}_{m_1 N_2 + m_2} = \sum_{n_1} W_{N_1}^{n_1 m_1} S''_{n_1} \tag{139}$$

The FFT is implemented in cuFFT, We may use a batched 3D cuFFT and a batched 1D cuFFT to implement 4D FFT.

4.1.2 Cornell Gauge Fixing and FFT accelerated

The Cornell Gauge Fixing is the steepest descend gauge fixing. The **Landau Gauge** for example. The Landau gauge needs $\partial_\mu A_\mu = 0$. One finds that if

$$F(A) = \sum_n \mathrm{tr}\left[A_\mu^2(n)\right] \tag{140}$$

is minimized, which means $\partial_\mu F(A) = 0$, and leads to $\partial_\mu A_\mu = 0$. In other words, the Landau gauge fixing is to find the minimum of $F(A)$ (using steepest descend method).

The steepest descend method can be simply described as

$$x_{n+1} \to x_n - \alpha \left.\frac{df(x)}{dx}\right|_{x=x_n} \tag{141}$$

where x is a vector, and x_n means iteration for n-times, α is a tunable parameter.

Using the **Cornell gauge fixing**, we follow Ref. [21]

The Cornell gauge fixing is a steepest descend algorithm, which can be described as

Algorithm 21 Cornell gauge fixing

```
for i = 0 to max iteration do
    A_μ(n) = U_μ(n).TA()
    Γ(n) = Σ_μ (A_μ(n − μ) − A_μ(n))
    if Σ_n Γ(n)Γ†(n) < ε then return                ▷ Succeed.
    end if
    G(n) = exp(−α_0 Γ(n))
    U_μ(n) = G(n)U_μ(n)G†(n + μ)
end for
```

Note:

- TA means traceless anti-Hermitian.
- For a traceless anti-Hermitian matrix, $M^\dagger M = 2(|m11{+}m22|^2{+}|m12|^2{+}|m13|^2{+}|m23|^2)$, where $m11$ and $m22$ are pure imaginary numbers.
- For a traceless anti-Hermitian matrix, $\exp(M)$ can be calculated as Appendix. A of Ref. [8].
- α_0 is a tunable parameter usually set to $0.05 - 0.1$.

The Cornell gauge fixing can be Fourier accelerated. At first, prepare the table such that

$$f_p(n)=\begin{cases}\frac{4N_d}{2V\left(N_d-\sum_\mu \cos\left(\frac{2\pi n_\mu}{L_\mu}\right)\right)}, & N_d\neq \sum_\mu \cos\left(\frac{2\pi n_\mu}{L_\mu}\right);\\ \frac{4N_d}{V}, & N_d=\sum_\mu \cos\left(\frac{2\pi n_\mu}{L_\mu}\right).\end{cases} \tag{142}$$

where $N_d=4$ is the number of dimension (Note, for Coulomb gauge, it is not 4), and V is the volume of the FFT transform (just the volume of the lattice, for the case of Coulomb gauge, it is the spatial volume.). L_μ is the extend of the direction.

Then, the step to generate gauge transform is modified by insert a FFT and an inverse FFT such that

$$G(n)=\exp\left(-\alpha_0\Gamma(n)\right)\to G(n)=\exp\left(-\alpha_0\hat{\mathrm{F}}f_p(n)\mathrm{F}\Gamma(n)\right) \tag{143}$$

where the FFT of a matrix is the FFT of each matrix element.

4.1.3 Los Alamos Gauge Fixing and over relaxation

Using the **Los Alamos gauge fixing**, we follow Ref. [22]

The idea is to maximize $F(U)=\sum_n\sum_\mu ReTr[U_\mu(n)]$.

By rewrite

$$\begin{aligned}&F(U)=\sum_n\sum_\mu ReTr[U_\mu(n)]=\frac{1}{2}\sum_n\sum_\mu ReTr[U_\mu(n)+U^\dagger_\mu(n-\mu)]=\frac{1}{2}\sum_n ReTr[\omega(n)],\\&\omega(n)\equiv\sum_\mu\left(U_\mu(n)+U^\dagger_\mu(n-\mu)\right).\end{aligned} \tag{144}$$

Note that, if for the gauge transform such that only even sites or odd sites are non-unity, the transform of ω is $G(n)\omega(n)$ or $\omega(n)G^\dagger(n)$, and $ReTr[G(n)\omega(n)]=ReTr[\omega(n)G^\dagger(n)]$. So we just need to find a $G(n)$ such that $ReTr[G(n)\omega(n)]\geq ReTr[\omega(n)]$, which is known as

Cabibbo-Marinari trick, which is

$$
\begin{aligned}
&G(n) = ABC\\
&a_{11} = \frac{1}{\sqrt{|a_{11}|^2+|a_{12}|^2}}(m_{11}^*+m_{22}),\quad a_{12} = \frac{1}{\sqrt{|a_{11}|^2+|a_{12}|^2}}(m_{21}^*-m_{12})\\
&b_{11} = \frac{1}{\sqrt{|b_{11}|^2+|b_{13}|^2}}(m_{11}^*+m_{33}),\quad b_{13} = \frac{1}{\sqrt{|b_{11}|^2+|b_{13}|^2}}(m_{31}^*-m_{13})\\
&c_{22} = \frac{1}{\sqrt{|c_{22}|^2+|c_{23}|^2}}(m_{22}^*+m_{33}),\quad c_{23} = \frac{1}{\sqrt{|c_{22}|^2+|c_{23}|^2}}(m_{32}^*-m_{23})\\
&A = \begin{pmatrix} a_{11} & a_{12} & 0 \\ -a_{12}^* & a_{11}^* & 0 \\ 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} b_{11} & 0 & b_{13} \\ 0 & 1 & 0 \\ -b_{31}^* & 0 & b_{11}^* \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{22} & c_{23} \\ 0 & -c_{23}^* & c_{22}^* \end{pmatrix}
\end{aligned}
\tag{145}
$$

The Los Alamos gauge fixing with over relaxation ω can be summarized as

Algorithm 22 Los Alamos gauge fixing with over relaxation ω

for $i = 0$ to max iteration **do**
  if $\sum_n \Gamma(n)\Gamma^\dagger(n) < \epsilon$ **then return** ▷ Succeed. Γ is defined in the above.
  end if
  for all odd sites
  $G(n) = \sum_\mu \left(U_\mu(n) + U_\mu^\dagger(n-\mu)\right)$
  $G(n) = (1-\omega)\mathbb{I}_{3\times 3} + \omega G(n)$
  $G(n) = CabibboMarinariProjection(G(n))$
  if n is odd **then**
    $U_\mu(n) = G(n)U_\mu(n)$
  else
    $U_\mu(n) = U_\mu(n)G^\dagger(n+\mu)$
  end if
  for all even sites, do the same thing.
end for

Note:

- There is no need to check convergence every iteration.
- When $\omega = 1$, there is no over relaxation, $\omega = \frac{2}{1+\frac{3}{L}}$ is often used.

4.1.4 Coulomb Gauge

The Coulomb gauge is very similar to Landau gauge, however, note that the gauge transform can be performed time slice by time slice. Usually, the count of iteration to convergence is different for each time slice.

4.1.5 Logarithm definition

The usual definition of U_μ is $U_\mu = e^{iaA_\mu}$, there for the real definition of A should be $iaA_\mu = \log(U_\mu)$.

To calculate take an example of $U^{\frac{1}{k}}$ first, let

$$A = P^{-1}DP \tag{146}$$

where D is a diagonal matrix, let

$$B = P^{-1}D^{\frac{1}{k}}P,\ \ B^k = \left(P^{-1}D^{\frac{1}{k}}P\right)^k = P^{-1}\left(D^{\frac{1}{k}}\right)^k P = P^{-1}DP = A \tag{147}$$

The problem reduce to a eigne system problem.

- Eigenvalue of 3×3 matrix

The eigenvalues of a 3×3 matrix can be obtained by solve the equation

$$A = \det\left[\alpha\mathbb{I}_3 - A\right] = 0 = \alpha^3 - \alpha^2 \mathrm{tr}\left[A\right] - \alpha\frac{1}{2}\left(\mathrm{tr}\left[A^2\right] - \mathrm{tr}^2\left[A\right]\right) - \det\left[A\right] \tag{148}$$

Using the fact that, if $A = aB + b\mathbb{I}_3$, the eigenvalue of A is $\lambda_A = a\lambda_B + b$, we can make a traceless matrix

$$B = \frac{1}{\sqrt{\frac{\mathrm{tr}\left[\left(A - \frac{\mathrm{tr}[A]}{3}\right)^2\right]}{6}}}\left(A - \frac{\mathrm{tr}\left[A\right]}{3}\right) \tag{149}$$

such that

$$\mathrm{tr}\left[B\right] = 0,\ \ \mathrm{tr}\left[B^2\right] = 6 \tag{150}$$

and the eigenvalue equation for B is

$$\lambda_B^3 - 3\lambda_B - \det\left[B\right] = 0 \tag{151}$$

which can be solved analytically (for example, by using Mathematica).

- Eigenvector of 3×3 matrix

If we assume there is no degeneracy due to float point precision, then any column of the matrix $(A - \lambda_2 \mathbb{I}_3)(A - \lambda_3 \mathbb{I}_3)$ is an eigenvector correspond to λ_1, because $(A - \lambda_1 \mathbb{I}_3)(A - \lambda_2 \mathbb{I}_3)(A - \lambda_3 \mathbb{I}_3) = 0$.

- Power, logarithm, and exponential

Similar as the power,

$$\begin{aligned} A &= P^{-1} D P \\ \log(A) &= P \log(D) P^{-1} \\ \exp(A) &= P \exp(D) P^{-1} \end{aligned} \tag{152}$$

where the exponential and logarithm of a diagonal matrix is easy to do.

5 Measurement

5.1 Plaquette Energy

For $SU(N)$, for square lattice, the gauge action can be written as

$$S_G = \beta \frac{1}{N} \sum_n \sum_{\mu > \nu} \left(N - \mathrm{tr}\left[U_\mu(n) U_\nu(n + a\mu) U_\mu^{-1}(n + a\nu) U_\nu^{-1}(n) \right] \right)$$
$$S_G = \frac{1}{4} \beta \frac{1}{N} \sum_n \left((2(D-1))N - \mathrm{tr}\left[U_\mu(n) \Sigma_\mu(n) \right] \right),$$
$$\Sigma_\mu(n) = \sum_{\mu \neq \nu} \left(U_\nu(n) U_\mu(n + a\nu) U_\nu^{-1}(n + a\mu) + U_\nu^{-1}(n - a\nu) U_\mu(n - a\nu) U_\nu(n - a\nu + a\mu) \right) \tag{153}$$

The plaquette energy is defined as

$$\begin{aligned} \langle S \rangle &= \frac{1}{N\Lambda} \sum_n \sum_{\mu > \nu} \left(\mathrm{tr}\left[U_\mu(n) U_\nu(n + a\mu) U_\mu^{-1}(n + a\nu) U_\nu^{-1}(n) \right] \right). \\ &= \frac{1}{N\Lambda} \sum_{n,\mu} \mathrm{tr}\left[U_\mu(n) \Sigma_\mu(n) \right]. \end{aligned} \tag{154}$$

which is the average (average according to configurations) energy of plaquettes per plaquette (average according to plaquettes).

This is also $\langle W^{1\times 1} \rangle$.

5.2 Meson Correlator

5.2.1 Meson Wave Function

We need at first construct an observable which is a bound state of two fermions and **has the same quantum number** as mesons. In short, we want to know

$$O(x) = \bar{\psi}(x) \Gamma \psi(x) \tag{155}$$

where Γ is a (product of) gamma matrix.

5.2.2 Meson Correlator

The correlator is defined as

$$C(x, y) = \langle \bar{O}(x) O(y) \rangle \tag{156}$$

where

$$\begin{aligned}\langle W\rangle &= \frac{1}{Z}\int \mathcal{D}[U,\bar{\psi},\psi]W\exp\left(-S\right)\\ Z &= \int \mathcal{D}[U,\bar{\psi},\psi]\exp\left(-S\right), \quad S=S_G+S_{pf}\end{aligned} \tag{157}$$

- iso-triplet

Denote the variables as C_T and O_T.

We need to calculate (green variables are constant)

$$\begin{aligned}C_T(n,m) &= \langle\bar{\psi}^{f_1}(n)\Gamma\psi^{f_2}(n)\bar{\psi}^{f_2}(m)\Gamma\psi^{f_1}(m)\rangle\\ &= \sum_{a,b,c_i}\Gamma_{a_1,b_1}\Gamma_{a_2,b_2}\langle\bar{\psi}^{f_1}_{a_1,c_1}(n)\psi^{f_2}_{b_1,c_1}(n)\bar{\psi}^{f_2}_{a_2,c_2}(m)\psi^{f_1}_{b_2,c_2}(m)\rangle\end{aligned} \tag{158}$$

Note that, they are all Grassman numbers (exchange three times will introduce a minus sign), and they can be averaged according to different fields, so

$$C_T(n,m) = -\sum_{a,b,c_i}\Gamma_{a_1,b_1}\Gamma_{a_2,b_2}\langle\psi^{f_2}_{b_1,c_1}(n)\bar{\psi}^{f_2}_{a_2,c_2}(m)\rangle_{f_1}\langle\psi^{f_1}_{b_2,c_2}(m)\bar{\psi}^{f_1}_{a_1,c_1}(n)\rangle_{f_2} \tag{159}$$

Using the Wick theorem for Grassman numbers (f is flavour index, c is color index, a,b are spinor index).

$$\begin{aligned}&\langle\ldots\rangle = \frac{1}{Z_f}\int\mathcal{D}\left[\psi\right]\ldots\exp\left(-\sum_{l,m}^{N}\bar{\psi}_l M_{lm}\psi_m\right).\\ &\langle\psi_{i_1}\ldots\psi_{i_n}\bar{\psi}_{j_1}\ldots\bar{\psi}_{j_n}\rangle = \sum_P \mathrm{sign}(P)\prod_n^N\left(M^{-1}\right)_{i_n,j_{P_n}}.\\ &\langle\psi^f(n)_{a,c_1}\bar{\psi}^f_{b,c_2}(m)\rangle = -D^{-1}_{f,a,b,c_1,c_2}(n,m).\end{aligned} \tag{160}$$

Then we can multiply gamma matrix back

$$C_T(n,m) = -\mathrm{tr}_{c,s}\left[\Gamma D^{-1}_{f_1}(n,m)\Gamma D^{-1}_{f_2}(m,n)\right] \tag{161}$$

The trace is for both color and spinor space.

- iso-singlet

Denote the variables as C_S and O_S.

5.2.3 Sources

- Fourier transform

Usually, one need to know the observable in momentum space, which is

$$\tilde{C}(\mathbf{p}, n_t; \mathbf{0}, 0) \equiv \frac{1}{\sqrt{\Lambda_3}} \sum_{\mathbf{n} \in \Lambda_3} \exp(-ia\mathbf{n} \cdot \mathbf{p}) C(\mathbf{n}, n_t; \mathbf{0}, 0) \tag{162}$$

where Λ_3 denotes the spatial lattice.

For hadron spectroscopy,

$$\tilde{C}(\mathbf{p}, n_t; \mathbf{0}, 0) \propto \exp\left(-an_t E_0(\mathbf{p})\right) \times \left(1 + \mathcal{O}\left(e^{-an_t \Delta E}\right)\right) \tag{163}$$

where $E_0(\mathbf{p})$ is the ground state energy (dissipative relation?) and ΔE is the energy gap between ground state and the lowest excitation, and

$$E_0(\mathbf{p}) = \sqrt{m_H^2 + |\mathbf{p}|^2} \times \left(1 + \mathcal{O}\left(a|\mathbf{p}|\right)\right) \tag{164}$$

For zero momentum, we find m_H. That is why the lattice at t-dir is usually larger than the spatial directions.

From Eq. (161), we only need to calculate $C(n, 0)$ for all n. That is a **point source**.

Using $\{\gamma_\mu, \gamma_5\} = 0$ and $\gamma_5^2 = 1$, $\{\gamma_\mu, \gamma_5\} = 0$, so

$$\begin{aligned}
&\left(\Gamma D^{-1}(n,m)\Gamma D^{-1}(m,n)\right) = \left(\Gamma D^{-1}(n,m)\Gamma\gamma_5 \left(D^{-1}(n,m)\right)^\dagger \gamma_5\right) \\
&\mathrm{tr}_{c,s}\left[\Gamma D^{-1}(n,m)\Gamma\gamma_5 \left(D^{-1}(n,m)\right)^\dagger \gamma_5\right] = \mathrm{tr}_{c,s}\left[\gamma_5\Gamma D^{-1}(n,m)\Gamma\gamma_5 \left(D^{-1}(n,m)\right)^\dagger\right] \\
&= \pm\mathrm{tr}_{c,s}\left[\Gamma' D^{-1}(n,m)\Gamma' \left(D^{-1}(n,m)\right)^\dagger\right] \\
&= \pm\mathrm{tr}_{c,s}\left[\Gamma'^\dagger D^{-1}(n,m)\Gamma' \left(D^{-1}(n,m)\right)^\dagger\right]
\end{aligned} \tag{165}$$

where $\Gamma' = \Gamma\gamma_5$ and $\pm$ come from $\gamma_5\Gamma = \pm\Gamma\gamma_5$, and $\pm$ come from both $\gamma_5\Gamma = \pm\Gamma\gamma_5$ and $\Gamma^\dagger = \pm\Gamma^\dagger$. Note that it is in fact a **real** number because

$$\begin{aligned}
&\mathrm{tr}_{c,s}\left[\Gamma'^\dagger D^{-1}(n,m)\Gamma' \left(D^{-1}(n,m)\right)^\dagger\right] = \mathrm{tr}_{c,s}\left[D^{-1}(n,m)\Gamma' \left(D^{-1}(n,m)\right)^\dagger \Gamma'^\dagger\right] \\
&\left(\mathrm{tr}_{c,s}\left[\Gamma'^\dagger D^{-1}(n,m)\Gamma' \left(D^{-1}(n,m)\right)^\dagger\right]\right)^* = \mathrm{tr}_{c,s}\left[\left(D^{-1}(n,m)\Gamma' \left(D^{-1}(n,m)\right)^\dagger \Gamma'^\dagger\right)^\dagger\right] \\
&= \mathrm{tr}_{c,s}\left[\Gamma' D^{-1}(n,m)\Gamma'^\dagger \left(D^{-1}(n,m)\right)^\dagger\right] = \mathrm{tr}_{c,s}\left[\Gamma'^\dagger D^{-1}(n,m)\Gamma' \left(D^{-1}(n,m)\right)^\dagger\right]
\end{aligned} \tag{166}$$

With point source, we need only to calculate $D^{-1}(n,0)$, which is a $12\times 12=144$ elements matrix field on each site, with the matrix element

$$\begin{aligned}&D^{-1}(n,m_0)_{c_1,c_2,s_1,s_2}=\sum_{m,c_3,s_3}D^{-1}(n,m)_{c_1,c_3,s_1,s_3}\left(S(m_0,c_2,s_2;m,c_3,s_3)\right)\\&D^{-1}(n,m_0)_{:,c_2,:,s_2}=D^{-1}\phi^S_{m_0,c_2,s_2}\end{aligned}\tag{167}$$

In the last line, $:,c_2,:,s_2$ denote one column of the 12×12 matrix, and $\phi^S_{m_0,c_2,s_2}$ is pseudo-fermion field with only one none-zero element (the **point source** at m_0, in our case, $m_0=(\mathbf{0},0)$)

$$\phi^S_{m_0,c_2,s_2}(m)_{c,s}=\delta(m-m_0)\delta(c-c_2)\delta(s-s_2)\tag{168}$$

In matrix form it is

$$\begin{pmatrix}D^{-1}_{1,cs}\\D^{-1}_{2,cs}\\D^{-1}_{3,cs}\\\ldots\\D^{-1}_{10,cs}\\D^{-1}_{11,cs}\\D^{-1}_{12,cs_2}\end{pmatrix}=\begin{pmatrix}D^{-1}_{1,1}&D^{-1}_{1,2}&\ldots&\color{red}{D^{-1}_{1,cs}}&\ldots&D^{-1}_{1,11}&D^{-1}_{1,12}\\D^{-1}_{2,1}&D^{-1}_{2,2}&\ldots&\color{red}{D^{-1}_{2,cs}}&\ldots&D^{-1}_{2,11}&D^{-1}_{2,12}\\D^{-1}_{3,1}&D^{-1}_{3,2}&\ldots&\color{red}{D^{-1}_{3,cs}}&\ldots&D^{-1}_{3,11}&D^{-1}_{3,12}\\\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots\\D^{-1}_{10,1}&D^{-1}_{0,2}&\ldots&\color{red}{D^{-1}_{10,cs}}&\ldots&D^{-1}_{10,11}&D^{-1}_{10,12}\\D^{-1}_{11,1}&D^{-1}_{11,2}&\ldots&\color{red}{D^{-1}_{11,cs}}&\ldots&D^{-1}_{11,11}&D^{-1}_{11,12}\\D^{-1}_{12,1}&D^{-1}_{12,2}&\ldots&\color{red}{D^{-1}_{12,cs}}&\ldots&D^{-1}_{12,11}&D^{-1}_{12,12}\end{pmatrix}\begin{pmatrix}0\\\ldots\\0\\1_{idx=cs}\\0\\\ldots\\0\end{pmatrix}\tag{169}$$

So, we need 12 point sources to fill the 12×12 matrix. Now, for each site, we can calculate the trace, and obtain a **real** field defined on sites

$$\Phi(n)=\mathrm{tr}_{c,s}\left(\Gamma D^{-1}\Gamma D^{-1}\right)\tag{170}$$

The final step is to sum the spatial lattice with the weight $e^{-ia\mathbf{n}\cdot\mathbf{p}}$ for each n_t, note that, the result should be calculated for each (assume periodic boundary condition for spatial directions)

$$\mathbf{p}\in\left\{(p_1,p_2,p_3)|p_i=\frac{2\pi}{aN_i}k_i,k_i=-\frac{N_i}{2}-1,\ldots\frac{N_i}{2}\right\}\tag{171}$$

where N_i is the length of the lattice at i direction.

Therefor, $\tilde{C}(\mathbf{p})$ is a complex field defined on spatial **reciprocal space**.

For spectroscopy, we need only the data for $\mathbf{p}=0$, because when $\Delta E\ll 1$

$$\tilde{C}(n_t)\equiv\tilde{C}(\mathbf{0},n_t;\mathbf{0},0)\propto\exp\left(-an_tm_H\right).\tag{172}$$

Note that, for $k_i=0$, we need **even** number of length at spatial directions.

- Detail of implementation

We can write D^{-1} in the form of 4×4 matrices, with elements as 3×3 matrices, as

$$D^{-1} = \begin{pmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ U_{21} & U_{22} & U_{23} & U_{24} \\ U_{31} & U_{32} & U_{33} & U_{34} \\ U_{41} & U_{42} & U_{43} & U_{44} \end{pmatrix} \tag{173}$$

If our pseudo-fermion field is organized as

$$D^{-1}\phi^S \equiv \phi^{s\times 3+c}(n) = (d_0, d_1, d_2, d_3)\,, d_s = (v_0, v_1, v_2) \tag{174}$$

Using Eq. (168), one have

$$\begin{aligned} U_{ij} &= \begin{pmatrix} \phi^{j\times 3+0}(d_i,v_0) & \phi^{j\times 3+1}(d_i,v_0) & \phi^{j\times 3+2}(d_i,v_0) \\ \phi^{j\times 3+0}(d_i,v_1) & \phi^{j\times 3+1}(d_i,v_1) & \phi^{j\times 3+2}(d_i,v_1) \\ \phi^{j\times 3+0}(d_i,v_2) & \phi^{j\times 3+1}(d_i,v_2) & \phi^{j\times 3+2}(d_i,v_2) \end{pmatrix}, \\ U^T_{ij} &= \begin{pmatrix} \phi^{j\times 3+0}(d_i,v_0) & \phi^{j\times 3+0}(d_i,v_1) & \phi^{j\times 3+0}(d_i,v_2) \\ \phi^{j\times 3+1}(d_i,v_0) & \phi^{j\times 3+1}(d_i,v_1) & \phi^{j\times 3+1}(d_i,v_2) \\ \phi^{j\times 3+2}(d_i,v_0) & \phi^{j\times 3+2}(d_i,v_1) & \phi^{j\times 3+2}(d_i,v_2) \end{pmatrix} \end{aligned} \tag{175}$$

The Γ intersect between D^{-1} and $(D^{-1})^\dagger$ is a permutation of rows in spinor space, for example

$$\gamma_1 = \begin{pmatrix} 0 & 0 & 0 & c_1 \\ 0 & 0 & c_2 & 0 \\ 0 & c_3 & 0 & 0 \\ c_4 & 0 & 0 & 0 \end{pmatrix}, \quad \begin{matrix} p(1)=4, & p(2)=3, & p(3)=2, & p(4)=1 \\ c_1=-i, & c_2=-i, & c_3=i, & c_4=i \end{matrix} \tag{176}$$

where c_i are coefficients and $c_i \in \mathbb{Z}_4$ group. $p(i)=j$ denote that the none-zero of the i-th row is j-element, (also, the none-zero of the i-th column is j-element, because $\gamma_\mu^\dagger = \gamma_\mu$.).

So one have such that

$$\begin{aligned} \Gamma' D^{-1} &= \begin{pmatrix} c_1U_{p(1)1} & c_1U_{p(1)2} & c_1U_{p(1)3} & c_1U_{p(1)4} \\ c_2U_{p(2)1} & c_2U_{p(2)2} & c_2U_{p(2)3} & c_2U_{p(2)4} \\ c_3U_{p(3)1} & c_3U_{p(3)2} & c_3U_{p(3)3} & c_3U_{p(3)4} \\ c_4U_{p(4)1} & c_4U_{p(4)2} & c_4U_{p(4)3} & c_4U_{p(4)4} \end{pmatrix}, \\ \Gamma' D^{-1}\Gamma'^\dagger &= \begin{pmatrix} c_1c_1^*U_{p(1)p(1)} & c_1c_2^*U_{p(1)p(2)} & c_1c_3^*U_{p(1)p(3)} & c_1c_4^*U_{p(1)p(4)} \\ c_2c_1^*U_{p(2)p(1)} & c_2c_2^*U_{p(2)p(2)} & c_2c_3^*U_{p(2)p(3)} & c_2c_4^*U_{p(2)p(4)} \\ c_3c_1^*U_{p(3)p(1)} & c_3c_2^*U_{p(3)p(2)} & c_3c_3^*U_{p(3)p(3)} & c_3c_4^*U_{p(3)p(4)} \\ c_4c_1^*U_{p(4)p(1)} & c_4c_2^*U_{p(4)p(2)} & c_4c_3^*U_{p(4)p(3)} & c_4c_4^*U_{p(4)p(4)} \end{pmatrix} \end{aligned} \tag{177}$$

Mason	correlator	Γ	Γ'	mass
Scalar (0^{++})	a_0	$1, \gamma_4$	$\gamma_5, \gamma_4\gamma_5$	
Pseudoscalar (0^{-+})	$\pi^+ = \bar{d}\gamma_5 u$	γ_5	1	139.57061(24)MeV
Vector (1^{--})	$\rho^+ = d\gamma_i\bar{u}$	γ_i	$\gamma_i\gamma_5$	775.11(34)MeV
Axial vector (1^{++})				
Tensor (1^{+-})				

表 1

Finally we have (Note U is not a $SU(3)$ matrix)

$$\mathrm{tr}_{c,s}\left[\Gamma' D^{-1}\Gamma'^{\dagger}\left(D^{-1}\right)^{\dagger}\right] = \sum_{ij} c_i c_j^* \mathrm{tr}_c\left[U_{p(i)p(j)}U_{ij}^{\dagger}\right] = \sum_{ij} c_i c_j^* \mathrm{tr}_c\left[U_{ij}^{\dagger}U_{p(i)p(j)}\right] \tag{178}$$

This can be further simplified, note that the result should be a real number, so for $i \neq j$, if tr $\left[U_{p(i)p(j)}^{\dagger}U_{ij}\right]$ is present, so must be tr $\left[U_{ij}^{\dagger}U_{p(i)p(j)}\right]$ with the same sign. This is guaranteed by symmetric matrix, i.e. if $p(i) = a$, one must have $p(a) = i$, and also $c_i c_j^* = c_{p(i)}c_{p(j)}^*$ as shown below.

To prove $c_i c_j^* = c_{p(i)}c_{p(j)}^*$, we need to consider:

1. $\Gamma^{\dagger} = \pm\Gamma$ and $p(i) = i$. In this case, $c_i = c_{p(i)}$, and $c_i c_j^* = c_{p(i)}c_{p(j)}^*$ is straightforward.

2. $\Gamma^{\dagger} = \Gamma$ and $p(i) \neq i$. In this case, $c_i = c_{p(i)}^*$, so $c_i c_j^* = c_{p(i)}^* c_{p(j)}$. Note that, c_i are either all real or all imaginary, so $c_i c_j^* = c_{p(i)}^* c_{p(j)} = c_{p(i)}c_{p(j)}^*$.

3. $\Gamma^{\dagger} = -\Gamma$ and $p(i) \neq i$. In this case, $c_i = -c_{p(i)}^*$, so $c_i c_j^* = c_{p(i)}^* c_{p(j)} = c_{p(i)}c_{p(j)}^*$.

So, we have two cases, one for $p(1) = 1$, and one for $p(1) \neq 1$

$$\begin{aligned}&\mathrm{tr}_{c,s}\left[\Gamma' D^{-1}\Gamma'^{\dagger}\left(D^{-1}\right)^{\dagger}\right]\\ &= \begin{cases} 2\sum_{i>1,j>i} c_i c_j^* \mathrm{Retr}_c\left[U_{ij}^{\dagger}U_{p(i)p(j)}\right] + \sum_{i=1,2,3,4}\mathrm{tr}_c\left[U_{ii}^{\dagger}U_{ii}\right] & p(i) = i \\ 2\sum_{i>1,j>i} c_i c_j^* \mathrm{Retr}_c\left[U_{ij}^{\dagger}U_{p(i)p(j)}\right] + 2\sum_{i=1,k}\mathrm{Retr}_c\left[U_{ii}^{\dagger}U_{p(i)p(i)}\right] & p(1) \neq 1, k \end{cases}\end{aligned} \tag{179}$$

5.2.4 Summary of parameters

The input gamma matrix is the Γ', here is a summary of gamma matrices.

5.2.5 Gauge smearing

Note: I think I did NOT understand gauge smearing correctly, will go back to this after staggered fermion being implemented.

In HMC with fermions, the computer power is consumed mainly in solving the D^{-1}. The small eigenvalues of the D operator is the main reason to slow down the solver, which is the so called **low mode** or **exceptional configurations**.

Gauge smearing (gauge smoothing) is one of the method to ease the problem by replacing the original configuration with a gauge equivalent but easier configuration. There are several different smearing methods. In CLGLib, only two are implemented.

- APE

It use

$$U'_\mu = \mathcal{P}\left((1-\alpha)U_\mu + \frac{\alpha}{6}\Sigma_\mu\right) \tag{180}$$

where Σ_μ is the staple, (see Eq. (152)). In CLGLib, staples are cached. After smoothing, $\mathcal{P}$ is a projection project to result to $SU(3)$ and can be approximated as

Algorithm 23 $\mathcal{P}(U)$ approximately

```
U = U/sqrt(tr(U†U/3))
for i = 0 to r do                                ▷ iterate r times
    x = U (3/2 − 1/2 U†U)
    U = (1 − i/3 Im(det(x))) x
end for
    return U
```

$U = U/\sqrt{tr(U^\dagger U/3)}$

for $i = 0$ to r **do** ▷ iterate r times

$x = U\left(\frac{3}{2} - \frac{1}{2}U^\dagger U\right)$

$U = \left(1 - \frac{i}{3}\mathrm{Im}(\det(x))\right)x$

end for

return U

Usually, iterate for 4 times, it can archive α accuracy.

- APE stout

In this approach, it construct a $SU(3)$ candidate directly by the staples. (Therefor, no need to project). Using

$$\Omega_\mu = \rho_\mu \Sigma_\mu U_\mu^\dagger,\ Q_\mu = \{\Omega_\mu\}_{TA},\ U'_\mu = \exp(Q_\mu)U_\mu \tag{181}$$

where ρ_μ is usually set to be $\rho_{1,2,3} = \rho, \rho_4 = 0$. Note that, there is no sum over μ in the above equation. Also, note that, exp is not accurate unless ρ is small enough, however, one can iterate the smearing for a few sub-steps.

5.3 Chiral Condensate of Wilson fermion

5.3.1 Wall source

5.3.2 Decay constant

5.3.3 Effective Quark Mass

The quark mass can be measured using

$$m_q = \frac{c_\delta}{2}\frac{\langle 0|\nabla_4 A_4|\pi\rangle}{\langle 0|P_5(n)|\pi\rangle} \tag{182}$$

where

$$\begin{aligned}
&\nabla_4 A_4(t) = A_4(t+1) - A_4(t-1) \to c_\delta = \frac{2m_\pi}{e^{+m_\pi} - e^{-m_\pi}};\\
&\nabla_4 A_4(t) = A_4(t+1) - A_4(t) \to c_\delta = \frac{m_\pi}{1 - e^{-m_\pi}};
\end{aligned} \tag{183}$$

5.4 Stochastic Methods

The stochastic methods is based on

$$\frac{1}{K}\sum_k \phi^{k*}_{c,s}(x)\phi^{k}_{c',s'}(y) \approx \delta_{xy}\delta_{cc'}\delta_{ss'} \tag{184}$$

where ϕ is a Gaussian distributed pseud-fermion, or $\mathbb{Z}_4$ random distributed pseud-fermion (the elements are $1, -1, i, -i$ average distributed).

Therefor, for any matrix,

$$\mathrm{tr}\,[M] \approx \frac{1}{K}\sum_k \phi^{k\dagger} M \psi^k \tag{185}$$

This is useful to measure some objects easily.

5.4.1 For condensations

For example, to measure $\langle\bar{\psi}O\psi\rangle$, where $\Gamma = 1$, γ_5 or γ_i etc.

For Grassman field, and for $S_F = -\bar{\psi}D\psi$

$$\langle \psi_i \bar{\psi}_j \rangle = (D^{-1})_{ji} \tag{186}$$

which means

$$\langle \psi_{c',s'}(n) \bar{\psi}_{c,s}(m) \rangle = (D^{-1})_{c'c,s's}(n|m) \tag{187}$$

Then consider $\langle \bar{\psi} O_{cs,c's'} \psi \rangle$ which is

$$\begin{aligned}\langle \bar{\psi} O_{cs,c's'} \psi \rangle &= \langle \sum_{cs,c's',mn} \bar{\psi}_{cs}(m) O_{cs,c's'}(m|n) \psi_{c's'}(n) \rangle = \sum_{cs,c's',mn} \langle \bar{\psi}_{cs}(m) \psi_{c's'}(n) \rangle O_{cs,c's'}(m|n) \\ &= - \sum_{cs,c's',mn} \langle \psi_{c's'}(n) \bar{\psi}_{cs}(m) \rangle O_{cs,c's'}(m|n) = - \sum_{cs,c's',mn} (D^{-1})_{c'c,s's}(n|m) O_{cs,c's'}(m|n)\end{aligned} \tag{188}$$

The sum over m is matrix multiply, and sum over n is a trace, so

$$\langle \bar{\psi} O_{cs,c's'} \psi \rangle = -\mathrm{tr}_{cs,m} \left[(D^{-1})_{c'c,s's}(n|m) O_{cs,c's'}(m|n) \right] \tag{189}$$

Note that, put D^{-1} right most is more convenient to calculate, use $\mathrm{tr}[ABC] = \mathrm{tr}[BCA]$, it is

$$\langle \bar{\psi} O_{cs,c's'} \psi \rangle = -\mathrm{tr}_{cs,m} \left[O_{cs,c's'}(m|n) (D^{-1})_{c'c,s's}(n|m) \right] \tag{190}$$

For any operator, one can always calculate $D^{-1}\phi^k$ first.

5.4.2 For densities

Similarly, the observables defined as $O(x) = \langle \bar{\psi}(x) O \psi(x) \rangle$ can be calculated, the discretized version is

$$O(n) = \langle \sum_{cs} \bar{\psi}_{cs}(n) \left(O\psi \right)_{cs}(n) \rangle = \langle \sum_{cs} \bar{\psi}_{cs}(n) \left(\sum_{c's',m} O_{cs,c's'}(n|m) \psi_{c's'}(m) \right) \rangle \tag{191}$$

All the same as above, it becomes

$$O(n) = - \sum_{cs,c's',m} (D^{-1})_{c's',cs}(m|n) O_{cs,c's'}(n|m) \tag{192}$$

On the other hand, the n-component of element product of two vectors $\frac{1}{K}\sum_k(\phi^\dagger, OD^{-1}\phi)$ (denoted as $X(n)$) is

$$\begin{aligned}
X(n) &= \frac{1}{K}\sum_k\sum_{cs}\phi^*_{cs}(n)\sum_{c_as_a,c_bs_b,a,b} O_{cs,c_as_a}(n|a)(D^{-1})_{c_as_a,c_bs_b}(a|b)\phi_{c_bs_b}(b)\\
&\approx \sum_{cs}\delta_{nb}\delta_{cc_b}\delta_{ss_b}\sum_{c_as_a,c_bs_b,a,b} O_{cs,c_as_a}(n|a)(D^{-1})_{c_as_a,c_bs_b}(a|b)\\
&= \sum_{cs}\sum_{c_as_a,a} O_{cs,c_as_a}(n|a)(D^{-1})_{c_as_a,cs}(a|n)
\end{aligned} \tag{193}$$

Do the variable rename $a\to m$, $c_a\to c'$, $s_a\to s'$, it is

$$\begin{aligned}
&X(n) \approx \sum_{cs}\sum_{c's',m} O_{cs,c's'}(n|m)(D^{-1})_{c'c,s's}(m|n) = \sum_{cs,c's',m}(D^{-1})_{c's',cs}(m|n)O_{cs,c's'}(n|m)\\
&O(n) \approx -X(n)
\end{aligned} \tag{194}$$

Note, the $\sum_{cs}$ can be written as tr_{cs}, and the sum $\sum_{c's',m}$ is just matrix product, so it is the trace of n-component of element product of two vectors $\frac{1}{K}\sum_k \mathrm{tr}_{cs}\left[(\phi^\dagger, OD^{-1}\phi)\right]$.

It is in fact the median result to calculate the condensation.

6 Programming

6.1 cuda

6.1.1 blocks and threads

6.1.2 device member function

According to https://stackoverflow.com/questions/53781421/cuda-the-member-field-with-device-ptr-and-device-member-function-to-visit-it-i

To call device member function, the content of the class should be on device.

- First, new a instance of the class.
- Then, create a device memory using cudaMalloc.
- Copy the content to the device memory

In other words, it will work as

```
__global__ void _kInitialArray(int* thearray)
{
    int iX = threadIdx.x + blockDim.x * blockIdx.x;
    int iY = threadIdx.y + blockDim.y * blockIdx.y;
    int iZ = threadIdx.z + blockDim.z * blockIdx.z;
    thearray[iX * 16 + iY * 4 + iZ] = iX * 16 + iY * 4 + iZ;
}

extern "C" {
    void _cInitialArray(int* thearray)
    {
        dim3 block(1, 1, 1);
        dim3 th(4, 4, 4);

        _kInitialArray << <block, th >> > (thearray);
        checkCudaErrors(cudaGetLastError());
    }
}

class B
{
public:
    B()
    {
        checkCudaErrors(cudaMalloc((void**)&m_pDevicePtr, sizeof(int) * 64));
```

```
        _cInitialArray(m_pDevicePtr);
    }
    ~B()
    {
        cudaFree(m_pDevicePtr);
    }
    __device__ int GetNumber(int index)
    {
        m_pDevicePtr[index] = m_pDevicePtr[index] + 1;
        return m_pDevicePtr[index];
    }
    int* m_pDevicePtr;
};

__global__ void _kAddArray(int* thearray1, B* pB)
{
    int iX = threadIdx.x + blockDim.x * blockIdx.x;
    int iY = threadIdx.y + blockDim.y * blockIdx.y;
    int iZ = threadIdx.z + blockDim.z * blockIdx.z;
    thearray1[iX * 16 + iY * 4 + iZ] = thearray1[iX * 16 + iY * 4 + iZ] + pB->GetNumber(iX * 16 +
         iY * 4 + iZ);
}

extern "C" {
    void _cAddArray(int* thearray1, B* pB)
    {
        dim3 block(1, 1, 1);
        dim3 th(4, 4, 4);
        _kAddArray << <block, th >> > (thearray1, pB);
        checkCudaErrors(cudaGetLastError());
    }
}

class A
{
public:
    A()
    {
        checkCudaErrors(cudaMalloc((void**)&m_pDevicePtr, sizeof(int) * 64));
        _cInitialArray(m_pDevicePtr);
    }
    ~A()
    {
        checkCudaErrors(cudaFree(m_pDevicePtr));
    }
    void Add(B* toAdd/*this should be a device ptr(new on device function or created by cudaMalloc)
         */)
```

```
    {
        _cAddArray(m_pDevicePtr, toAdd);
    }
    int* m_pDevicePtr;
};



int main(int argc, char * argv[])
{
    B* pB = new B();
    A* pA = new A();
    B* pDeviceB;
    checkCudaErrors(cudaMalloc((void**)&pDeviceB, sizeof(B)));
    checkCudaErrors(cudaMemcpy(pDeviceB, pB, sizeof(B), cudaMemcpyHostToDevice));
    pA->Add(pDeviceB);
    int* res = (int*)malloc(sizeof(int) * 64);
    checkCudaErrors(cudaMemcpy(res, pA->m_pDevicePtr, sizeof(int) * 64, cudaMemcpyDeviceToHost));
    printf("----------- A=");
    for (int i = 0; i < 8; ++i)
    {
        printf("\n");
        for (int j = 0; j < 8; ++j)
            printf("res %d=%d  ", i * 8 + j, res[i * 8 + j]);
    }
    printf("\n");
    //NOTE: We are getting data from pB, not pDeviceB, this is OK, ONLY because m_pDevicePtr is a pointer
    checkCudaErrors(cudaMemcpy(res, pB->m_pDevicePtr, sizeof(int) * 64, cudaMemcpyDeviceToHost));
    printf("----------- B=");
    for (int i = 0; i < 8; ++i)
    {
        printf("\n");
        for (int j = 0; j < 8; ++j)
            printf("res %d=%d  ", i * 8 + j, res[i * 8 + j]);
    }
    printf("\n");
    delete pA;
    delete pB;
    return 0;
}
```

Note: this is a copy of the original instance! It is ONLY OK to change the content of $pDevicePtr->m\_pOtherPtr$, NOT $pDevicePtr->somevalue$

6.1.3 device virtual member function

According to https://stackoverflow.com/questions/26812913/how-to-implement-device-side-cuda-virtual-functions

To call a device virtual member function, unlike Sec. 6.1.2, the pointer to the virtual function table should also be on device,

- First, cudaMalloc a sizeof(void*), for the device pointer.
- Then, use a kernel function to new the instance on device, and assign it to the device pointer created by cudaMalloc.
- One can copy the pointer, by using cudaMemcpy(void**, void**, sizeof(void*), devicetodevice).
- When copy it to elsewhere, one need to copy it back to host, then copy it again to device. The example shows how to copy it to constant.

in other words, it will work as

```

class CA
{
public:
    __device__ CA() { ; }
    __device__ ~CA() { ; }
    __device__ virtual void CallMe() { printf("This␣is␣A\n"); }
};

class CB : public CA
{
public:
    __device__ CB() : CA() { ; }
    __device__ ~CB() { ; }
    __device__ virtual void CallMe() { printf("This␣is␣B\n"); }
};

__global__ void _kernelCreateInstance(CA** pptr)
{
    (*pptr) = new CB();
}

__global__ void _kernelDeleteInstance(CA** pptr)
{
```

```
    delete (*pptr);
}

extern "C" {
    void _kCreateInstance(CA** pptr)
    {
        _kernelCreateInstance << <1, 1 >> >(pptr);
    }

    void _kDeleteInstance(CA** pptr)
    {
        _kernelDeleteInstance << <1, 1 >> >(pptr);
    }
}

__constant__ CA* m_pA;

__global__ void _kernelCallConstantFunction()
{
    m_pA->CallMe();
}


extern "C" {
    void _cKernelCallConstantFunction()
    {
        _kernelCallConstantFunction << <1, 1 >> > ();
    }
}

int main()
{
    CA** pptr;
    cudaMalloc((void**)&pptr, sizeof(CA*));
    _kCreateInstance(pptr);

    //I can NOT use a kernel to set m_pA = (*pptr), because it is constant.
    //I can NOT use cudaMemcpyToSymbol(m_pA, (*pptr)), because * operator on host is incorrect when pptr is a device ptr.
    //I can NOT use cudaMemcpyToSymbol(m_pA, (*pptr)) in kernel, because cudaMemcpyToSymbol is a __host__ function
    //I have to at first copy it back to host, then copy it back back again to constant
    CA* pptrHost[1];
    cudaMemcpy(pptrHost, pptr, sizeof(CA**), cudaMemcpyDeviceToHost);
    cudaMemcpyToSymbol(m_pA, pptrHost, sizeof(CA*));
    _cKernelCallConstantFunction();

```

```
    _kDeleteInstance(pptr);
    cudaFree(pptr);
    return 0;
}
```

7 Testing

7.1 random number

8 Applications

8.1 Rotating Frame

We follow Ref. [23].

The matrix element can be written as

$$\mathcal{M} = \int \mathcal{D}(A_\mu \psi) \exp\left(i \int d^4x \mathcal{L} \right) \tag{195}$$

with

$$\mathcal{L} = \bar{\psi}\left(i\not{D} - m\right)\psi - \frac{1}{4}\left(F^a_{\mu\nu}\right)^2, \;\; D_\mu \equiv \partial_\mu + ig_{YM}\sum_a T_a A^a_\mu \tag{196}$$

The first few steps are as usual, defining

$$A_\mu = g_{YM}\sum_a T_a A^a_\mu, \;\; F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu + i[A_\mu, A_\nu] \tag{197}$$

using $\mathrm{tr}[T_i T_j] = \frac{1}{2}\delta_{ij}$

$$\begin{aligned} F_{\mu\nu} &= g_{YM}\sum_a T^a F^a_{\mu\nu} \\ \frac{1}{4}\left(F^a_{\mu\nu}\right)^2 &= \frac{1}{2g^2_{YM}}\mathrm{tr}\left[F^2_{\mu\nu}\right] \end{aligned} \tag{198}$$

and

$$\begin{aligned} \mathcal{L} &= \bar{\psi}\left(i\not{D} - m\right)\psi - \frac{1}{2g^2_{YM}}\mathrm{tr}\left[F^2_{\mu\nu}\right] \\ D_\mu &= \partial_\mu + iA_\mu \end{aligned} \tag{199}$$

For rotational frame, the metric and frame can be defined as

$$
\begin{aligned}
h_{\mu\nu} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \\
g_{\mu\nu} &= \begin{pmatrix} 1-r^2\Omega^2 & +y\Omega & -x\Omega & 0 \\ y\Omega & -1 & 0 & 0 \\ -x\Omega & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \quad \sqrt{-g_{\mu\nu}} = 1 \\
e_0 &= (1, y\Omega, -x\Omega, 0) \\
e_1 &= (0, 1, 0, 0) \\
e_2 &= (0, 0, 1, 0) \\
e_3 &= (0, 0, 0, 1)
\end{aligned} \tag{200}
$$

8.1.1 The rotating gauge action

Considering the case of pure gauge, the action can be written as

$$
\begin{aligned}
\mathcal{L}_G &= -\sqrt{\det(-g_{\alpha\beta})}\frac{1}{2g_{YM}^2}g^{\mu\nu}g^{\rho\sigma}\mathrm{tr}[F_{\mu\rho}F_{\nu\sigma}] \\
&= -\frac{1}{2g_{YM}^2}\left(\sum_{ijkl=0}^{3} h_{ij}h_{kl}\mathrm{tr}[F_{ik}F_{jl}] + 2\Omega^2\mathrm{tr}\left[(xF_{01}+yF_{02})^2 + r^2F_{03}^2\right]\right. \\
&\left.-4\Omega\left(x\mathrm{tr}[F_{01}F_{12}] + y\mathrm{tr}[F_{02}F_{12}] + y\mathrm{tr}[F_{03}F_{13}] - x\mathrm{tr}[F_{03}F_{23}]\right)\right)
\end{aligned} \tag{201}
$$

- Wick rotation of gauge action

The Wick rotation

$$
t \to -i\tau, \quad \Omega \to i\Omega, \quad A_\mu \to (iA_0, A_1, A_2, A_3), \quad F_{0i} \to -iF_{0i} \tag{202}
$$

and substitute $x = (t, x, y, z) \to x_E = (x, y, z, \tau)$. After Wick rotation, we are expecting

$$
\exp(-S_G) = \exp(i\int d^4x\mathcal{L}_G) \tag{203}
$$

the result is

$$
\begin{aligned}
&-S_G = i\int d^4x\mathcal{L}_G \\
&S_G = \int d^4x_E \frac{1}{2g_{YM}^2}\left(\sum_{ij=1}^{4}\mathrm{tr}[F_{ij}F_{ij}] + 2\Omega^2\mathrm{tr}\left[(xF_{14}+yF_{24})^2 + r^2F_{34}^2\right]\right. \\
&\left.+4\Omega\left(x\mathrm{tr}[F_{14}F_{12}] + y\mathrm{tr}[F_{24}F_{12}] + y\mathrm{tr}[F_{34}F_{13}] - x\mathrm{tr}[F_{34}F_{23}]\right)\right)
\end{aligned}
\tag{204}
$$

Therefor S_G is real. The $\sum_{ij=1}^{4}\mathrm{tr}[F_{ij}F_{ij}]$ is the gauge action in rest frame.

- Discretization of gauge action

The discretized version can be derived using compact gauge group

$$
U_\mu(x) = \exp(iaA_\mu(x)), \;\; U_{-\mu}(x) = U_\mu^{-1}(x-\mu). \tag{205}
$$

As usual,

$$
\begin{aligned}
&U_{\mu,\nu}(n) \equiv U_\mu(n)U_\nu(n+a\mu)U_\mu^{-1}(n+a\nu)U_\nu^{-1}(n) = \exp(ia^2F_{\mu\nu}+\mathcal{O}(a^3)). \\
&\mathrm{Re}\left[U_{\mu\nu}(n)\right] = \mathbb{I}_{N_c\times N_c} - \frac{a^4}{2}F_{\mu\nu}^2 + \mathcal{O}(a^6) \\
&\frac{1}{2g_{YM}^2}\sum_{\mu\neq\nu}\mathrm{tr}[F_{\mu\nu}^2] = \frac{1}{a^4g_{YM}^2}\sum_{\mu\neq\nu}\mathrm{Retr}\left[1-U_{\mu\nu}(n)\right] = \frac{2}{a^4g_{YM}^2}\sum_{\mu>\nu}\mathrm{Retr}\left[1-U_{\mu\nu}(n)\right]
\end{aligned}
\tag{206}
$$

For those plaquette with coordinate as coefficients, we use the average of plaquette

$$
\begin{aligned}
&\bar{U}_{\mu,\nu}(n) \equiv \frac{1}{4}\left(U_{\mu,\nu}(n) + U_{-\mu,\nu}(n) + U_{\mu,-\nu}(n) + U_{-\mu,-\nu}(n)\right) \\
&\mathrm{Retr}[\bar{U}_{\mu,\nu}(n)] = N_c - \frac{a^4}{2}\mathrm{tr}[F_{\mu\nu}^2] + \mathcal{O}(a^6), \;\; \frac{1}{2g_{YM}^2}\mathrm{tr}[F_{\mu\nu}^2] = \frac{2}{a^4g_{YM}^2}\frac{1}{2}\mathrm{Retr}[1-\bar{U}_{\mu,\nu}(n)]
\end{aligned}
\tag{207}
$$

The remaining are those has the form $\mathrm{tr}[F_{ab}F_{bc}]$, using

$$
\begin{aligned}
&U_{\mu\nu}^{-1}(n) = U_\nu(n)U_\mu(n+a\nu)U_\nu^{-1}(n+a\mu)U_\mu^{-1}(n) = U_{\nu\mu}(n) = \exp(-ia^2F_{\mu\nu}) + \mathcal{O}(a^6) \\
&U_{a,b}(n)(U_{b,c}(n) - U_{c,b}(n)) = \exp(-ia^2(F_{ab}+F_{bc})) - \exp(-ia^2(F_{ab}-F_{bc})) \\
&\frac{1}{2}\mathrm{Re}\left[U_{a,b}(n)(U_{c,b}(n) - U_{b,c}(n))\right] = a^4F_{ab}F_{bc} + \mathcal{O}(a^6)
\end{aligned}
\tag{208}
$$

Note that b is not summed.

We can have a more symmetric form to use

$$
U_{c,b}(n) = U_{b,-c}(n) + \mathcal{O}(a^4) \tag{209}
$$

then it is a chair-type.

$$\frac{1}{2}\mathrm{Re}\left[U_{a,b}(n)(U_{b,-c}(n)-U_{b,c}(n))\right]=a^4F_{ab}F_{bc}+\mathcal{O}(a^6) \tag{210}$$

Similarly, we can use the average of chairs, and define

$$\begin{aligned}&V_{\mu\nu\sigma}=\frac{1}{8}\left((U_{\mu,\nu}-U_{-\mu,\nu})(U_{\nu,\sigma}-U_{\nu,-\sigma})+(U_{\mu,-\nu}-U_{-\mu,-\nu})(U_{-\nu,\sigma}-U_{-\nu,-\sigma})\right)\\&\mathrm{Retr}[V_{\mu\nu\rho}]=-a^4\mathrm{tr}\left[F_{\mu\nu}F_{\nu\rho}\right]+\mathcal{O}(a^6),\quad\frac{1}{2g_{YM}^2}\mathrm{tr}\left[F_{\mu\nu}F_{\nu\rho}\right]=-\frac{2}{a^4g_{YM}^2}\frac{1}{4}\mathrm{Retr}[V_{\mu\nu\rho}]\end{aligned} \tag{211}$$

- Finial result of gauge action

  The discretized and Wick rotated gauge action is

$$\begin{aligned}S_G=&\frac{2}{a^4g_{YM}^2}\sum_n\Bigg(\sum_{\mu>\nu}\mathrm{Retr}[1-U_{\mu\nu}(n)]+\Omega\left(x\mathrm{Retr}[V_{412}+V_{432}]-y\mathrm{Retr}[V_{421}+V_{431}]\right)\\&+\Omega^2(x^2\mathrm{Retr}[1-\bar{U}_{14}(n)]+y^2\mathrm{Retr}[1-\bar{U}_{24}(n)]+r^2\mathrm{Retr}[1-\bar{U}_{34}(n)]+xy\mathrm{Retr}[V_{142}]\Big)\\=&\frac{\beta}{N_c}\sum_n\Bigg(\sum_{\mu>\nu}\mathrm{Retr}[1-U_{\mu\nu}(n)]+\Omega\left(x\mathrm{Retr}[V_{412}+V_{432}]-y\mathrm{Retr}[V_{421}+V_{431}]\right)\\&+\Omega^2(x^2\mathrm{Retr}[1-\bar{U}_{14}(n)]+y^2\mathrm{Retr}[1-\bar{U}_{24}(n)]+r^2\mathrm{Retr}[1-\bar{U}_{34}(n)]+xy\mathrm{Retr}[V_{142}]\Big)\end{aligned} \tag{212}$$

with $\frac{\beta}{N_c}\equiv\frac{2}{a^4g_{YM}^2}$.

8.1.2 Rotating Fermion action

$$D_R=\left[i\gamma^\mu\left((\partial_\mu+ieA_\mu)-\frac{i}{4}\sigma^{ij}w_{\mu ij}\right)-m\right] \tag{213}$$

with

$$\begin{aligned}&w_{\mu ij}=g_{\alpha\beta}e_i^\alpha(\partial_\mu e_j^\beta+\Gamma^\beta_{\mu\nu}e_j^\nu)\\&\Gamma^\beta_{\mu\nu}=\frac{1}{2}g^{\beta\alpha}\left(\frac{\partial g_{\alpha\mu}}{\partial x^\nu}+\frac{\partial g_{\alpha\nu}}{\partial x^\mu}-\frac{\partial g_{\mu\nu}}{\partial x^\alpha}\right)\\&\sigma^{ij}=\frac{i}{2}[\gamma^i,\gamma^j]\end{aligned} \tag{214}$$

so

$$\frac{i}{4}\sigma^{ij}w_{\mu ij}=\left(\frac{i}{2}\Omega\sigma^{12},0,0,0\right) \tag{215}$$

and

$$D_R = \left[i\gamma^x(\partial_x + ieA_x) + i\gamma^y(\partial_y + ieA_y) + i\gamma^z(\partial_z + ieA_z) + i\gamma^t(\partial_t + ieA_t - \frac{i}{2}\Omega\sigma^{12}) - m\right] \tag{216}$$

using $\gamma^\mu = \gamma^i e_i^\mu$, it is

$$\begin{aligned} D_R = &\left[i(\gamma^1 + y\Omega\gamma^0)(\partial_x + ieA_x) + i(\gamma^2 - x\Omega\gamma^0)(\partial_y + ieA_y) + i\gamma^3(\partial_z + ieA_z)\right. \\ &\left. + i\gamma^0(\partial_t + ieA_t - \frac{i}{2}\Omega\sigma^{12}) - m\right] \end{aligned} \tag{217}$$

- The Wick rotation of Fermion action

The Wick rotation

$$t \to -i\tau,\ \ \gamma_i^M \to i\gamma_i^E,\ \ \gamma_4 = \gamma_0,\ \ \gamma_5 = \gamma_1\gamma_2\gamma_3\gamma_4,\ \ A_t \to iA_\tau,\ \ \partial_t \to i\partial_\tau,\ \ \Omega \to i\Omega,\ \ \sigma^{12} \to -\sigma^{12} \tag{218}$$

where the superscript of gamma matrix stands for Minkowski or Euclidian. So

$$\begin{aligned} D_R = &-\left[(\gamma_1 + y\Omega\gamma_4)(\partial_x + ieA_x) + (\gamma_2 - x\Omega\gamma_4)(\partial_y + ieA_y) + \gamma_3(\partial_z + ieA_z)\right. \\ &\left. + \gamma_4(\partial_\tau + ieA_\tau + \frac{i}{2}\Omega\sigma^{12}) + m\right] \\ &= -\left[\gamma_1(\partial_x + ieA_x) + \gamma_2(\partial_y + ieA_y) + \gamma_3(\partial_z + ieA_z) + \gamma_4(\partial_\tau + ieA_\tau) + m\right. \\ &\left. + \gamma_4\left(y\Omega(\partial_x + ieA_x) - x\Omega(\partial_y + ieA_y)\right) + \frac{i}{2}\gamma_4\Omega\sigma^{12}\right] \end{aligned} \tag{219}$$

And

$$\begin{aligned} &-S_F = i\int d^4x\sqrt{-g_{\alpha\beta}}\bar{\psi}D_R\psi \\ &S_F = \int d^4x_E\bar{\psi}\left[\gamma_1(\partial_x + ieA_x) + \gamma_2(\partial_y + ieA_y) + \gamma_3(\partial_z + ieA_z) + \gamma_4(\partial_\tau + ieA_\tau) + m\right. \\ &\left. + \gamma_4\left(y\Omega(\partial_x + ieA_x) - x\Omega(\partial_y + ieA_y)\right) + \frac{i}{2}\gamma_4\Omega\sigma^{12}\right]\psi \end{aligned} \tag{220}$$

- The discretization of Fermion action

The naive discretization yields

$$\partial_\mu\psi(n) = \frac{\psi(n + a\mu) - \psi(n - a\mu)}{2a} \tag{221}$$

and

$$\begin{aligned}
&U_\mu(n) = \exp(iaA_\mu) \approx 1 + iaA_\mu(n), \;\; U_\mu^{-1}(n) \approx 1 - iaA_\mu(n) \\
&iA_\mu(n) = \frac{2iaA_\mu(n)}{2a} \approx \frac{(U_\mu(n)-1) - (U_\mu^{-1}(n)-1)}{2a} \approx \frac{(U_\mu(n)-1) - (U_{-\mu}(n)-1)}{2a} \\
&iA_\mu(n)\psi(n) = \frac{(U_\mu(n)-1)\psi(n+a\mu) - (U_{-\mu}(n)-1)\psi(n-a\mu)}{2a} + \mathcal{O}(a)
\end{aligned} \tag{222}$$

Therefor

$$(\partial_\mu + iA_\mu)\psi(n) = \frac{U_\mu(n)\psi(n+a\mu) - U_{-\mu}(n)\psi(n-a\mu)}{2a} + \mathcal{O}(a) \tag{223}$$

The Wilson term is

$$W\psi(n) = -\sum_\mu \frac{U_\mu(n)\psi(n+a\mu) + U_{-\mu}(n)\psi(n-a\mu) - 2\psi(n)}{2a} \to 0 \tag{224}$$

considering the rotation, we add a modified Wilson term as

$$\begin{aligned}
W_R\psi(n) = &-\sum_\mu \frac{U_\mu(n)\psi(n+a\mu) + U_{-\mu}(n)\psi(n-a\mu) - 2\psi(n)}{2a} \\
&- y\Omega \frac{U_x(n)\psi(n+ax) + U_{-x}(n)\psi(n-ax) - 2\psi(n)}{2a} \\
&+ x\Omega \frac{U_y(n)\psi(n+ay) + U_{-y}(n)\psi(n-ay) - 2\psi(n)}{2a}
\end{aligned} \tag{225}$$

Similar as the Wilson term, the last two terms also decouples when approaching the continuum limit. And the Wilson-Dirac operator becomes

$$\begin{aligned}
&S_F = \sum_{n,m} \bar{\psi}(n) D_W(n|m)\psi(m) \\
&D_W(n|m) = \left(m + \frac{4}{a} + \frac{y\Omega}{a} - \frac{x\Omega}{a}\right)\delta_{n,m} - \sum_\mu \frac{(1-\gamma_\mu)U_\mu(n)\delta_{n+a\mu,m} + (1+\gamma_\mu)U_{-\mu}(n)\delta_{n-a\mu,m}}{2a} \\
&- y\Omega \frac{(1-\gamma_4)U_x(n)\delta_{n+ax,m} + (1+\gamma_4)U_{-x}(n)\delta_{n-ax,m}}{2a} \\
&+ x\Omega \frac{(1-\gamma_4)U_y(n)\delta_{n+ay,m} + (1+\gamma_4)U_{-y}(n)\delta_{n-ay,m}}{2a} + \frac{i}{2}\gamma_4\Omega\sigma^{12}\delta_{n,m}
\end{aligned} \tag{226}$$

Note that,

$$\begin{aligned}
&(U(1)\delta_{n+1,m,n=1,m=2})^\dagger = U^\dagger(1)\delta_{n,m+1,n=2,m=1} = U^\dagger(m)\delta_{n,m+1} \; or \; U^\dagger(n-1)\delta_{n-1,m} \\
&(U_\mu(n)\delta_{n+a\mu,m})^\dagger = U_\mu^{-1}(n-a\mu)\delta_{n,m+a\mu} = U_{-\mu}(n)\delta_{n-a\mu,m}
\end{aligned} \tag{227}$$

Let's check whether the periodic condition for gauge field or infinite lattice volume is necessary

$$
\begin{aligned}
&\sum_{n=1,2,3,m=1,2,3}(U_\mu(n)\delta_{n+1,m}+U_{-\mu}(n)\delta_{n-1,m})^\dagger=(U_\mu(1))^\dagger+(U_\mu(2))^\dagger+(U_{-\mu}(2))^\dagger+(U_{-\mu}(3))^\dagger\\
&=U_\mu^{-1}(2-1)+U_\mu^{-1}(3-1)+\left(U_\mu^{-1}(1)\right)^\dagger+\left(U_\mu^{-1}(2)\right)^\dagger\\
&=U_{-\mu}(2)+U_{-\mu}(3)+U_\mu(1)+U_\mu(2)\\
&=\sum_{n=1,2,3,m=1,2,3}(U_\mu(n)\delta_{n+1,m}+U_{-\mu}(n)\delta_{n-1,m})
\end{aligned}
\tag{228}
$$

So we can conclude The γ_5-hermiticity is kept with open(Dirichlet) boundary condition and finite volume.

Both the naive discretization and the Wilson term satisfy the γ_5-hermiticity (separately).

$$
\begin{aligned}
&\gamma_5\gamma_\nu\gamma_5=-\gamma_\nu,\;\;\gamma_\mu^\dagger=\gamma_\nu,\;\;\gamma_5^2=1\\
&\sum_{n,m}(\gamma_\nu U_\mu(n)\delta_{n+a\mu,m}-\gamma_\nu U_{-\mu}(n)\delta_{n,m+a\mu})^\dagger=\sum_{n,m}(\gamma_5\gamma_\nu\gamma_5 U_\mu(n)\delta_{n+a\mu,m}-\gamma_5\gamma_\nu\gamma_5 U_{-\mu}(n)\delta_{n-a\mu,m})\\
&\sum_{n,m}(U_\mu(n)\delta_{n+a\mu,m}+U_{-\mu}(n)\delta_{n,m+a\mu})^\dagger=\sum_{n,m}\left(\gamma_5^2 U_\mu(n)\delta_{n+a\mu,m}+\gamma_5^2 U_{-\mu}(n)\delta_{n-a\mu,m}\right)
\end{aligned}
\tag{229}
$$

Apart from that

$$
\left(\frac{i}{2}\gamma_4\Omega\sigma^{12}\delta_{n,m}\right)^\dagger=\gamma_5\frac{i}{2}\gamma_4\Omega\sigma^{12}\delta_{n,m}\gamma_5 \tag{230}
$$

Therefor, the new Wilson-Dirac operator is also γ_5-hermiticity.

- The doubler problem

Note that the naive action and Wilson term both satisfy the γ_5-hermiticity, the traditional Wilson term will also lead to a γ_5-hermiticity fermion action, with

$$
\begin{aligned}
&D_W(n|m)=\left(m+\frac{4}{a}\right)\delta_{n,m}-\sum_\mu\frac{(1-\gamma_\mu)U_\mu(n)\delta_{n+a\mu,m}+(1+\gamma_\mu)U_{-\mu}(n)\delta_{n-a\mu,m}}{2a}\\
&+y\Omega\frac{\gamma_4 U_x(n)\delta_{n+ax,m}-\gamma_4 U_{-x}(n)\delta_{n-ax,m}}{2a}-x\Omega\frac{\gamma_4 U_y(n)\delta_{n+ay,m}-\gamma_4 U_{-y}(n)\delta_{n-ay,m}}{2a}+\frac{i}{2}\gamma_4\Omega\sigma^{12}\delta_{n,m}
\end{aligned}
\tag{231}
$$

This action also does not suffer from the doubler problem.

- The final action of fermions

As usual, we define the hopping parameter as $\kappa = \frac{1}{2am+8}$, then rescale the fermion field, the action is

$$
\begin{aligned}
&S_F = \sum_{n,m} \bar{\psi}(n) D_W(n|m)\psi(m) \\
&D_W(n|m) = (1 + 2\kappa(y-x)\Omega)\,\delta_{n,m} - \kappa \sum_{\mu} \left((1-\gamma_\mu)U_\mu(n)\delta_{n+a\mu,m} + (1+\gamma_\mu)U_{-\mu}(n)\delta_{n-a\mu,m}\right) \\
&- \kappa y\Omega\left((1-\gamma_4)U_x(n)\delta_{n+ax,m} + (1+\gamma_4)U_{-x}(n)\delta_{n-ax,m}\right) \\
&+ \kappa x\Omega\left((1-\gamma_4)U_y(n)\delta_{n+ay,m} + (1+\gamma_4)U_{-y}(n)\delta_{n-ay,m}\right) + \kappa i\gamma_4 a\Omega\sigma^{12}\delta_{n,m}
\end{aligned}
\tag{232}
$$

8.1.3 The exponential chemical potential

On the other hand, the $i\kappa\gamma_4\hat{\Omega}\sigma^{12}$ term can also be modified. The σ^{12} term can be considered as a chemical potential ($\bar{\psi}\gamma_0\psi$ and then do the Wick rotation $\gamma_0 \to \gamma_4$. The sign is after Wick rotation and relative to the mass term)

$$
\mu\bar{\psi}\gamma_4\psi, \quad \mu = \frac{i\Omega}{2}\sigma^{12} \tag{233}
$$

and discritized as

$$
\begin{aligned}
D_\tau + \mu\bar{\psi}\gamma_4\psi &\to -\kappa\left(e^{\mu a}(1-\gamma_4)U_\tau(n)\delta_{n,n+t} + e^{-\mu a}(1+\gamma_4)U_{-\tau}(n)\delta_{n-t,n}\right) \\
&= -\kappa\left(e^{+\frac{ia\Omega\sigma^{12}}{2}}(1-\gamma_4)U_\tau(n)\delta_{n,n+t} + e^{-\frac{ia\Omega\sigma^{12}}{2}}(1+\gamma_4)U_{-\tau}(n)\delta_{n-t,n}\right)
\end{aligned}
\tag{234}
$$

It looks not satisfy the γ_5 -hermiticity. However, using $(\sigma^{12})^2 = 1$, it is in fact

$$
\begin{aligned}
D_\tau + \mu\bar{\psi}\gamma_4\psi \to -\kappa\Bigg[&\left(\cos(\frac{a\Omega}{2}) + i\sin(\frac{a\Omega}{2})\sigma^{12}\right)(1-\gamma_4)U_\tau(n)\delta_{n,n+t} \\
+ &\left(\cos(\frac{a\Omega}{2}) - i\sin(\frac{a\Omega}{2})\sigma^{12}\right)(1+\gamma_4)U_{-\tau}(n)\delta_{n-t,n}\Bigg]
\end{aligned}
\tag{235}
$$

The 1 in $1 \pm \frac{ia\Omega\sigma^{12}}{2}$ is the usual D_τ. So the additional term is in fact

$$
\begin{aligned}
-\kappa\Bigg[&\left(\cos(\frac{a\Omega}{2}) - 1 + i\sin(\frac{a\Omega}{2})\sigma^{12}\right)(1-\gamma_4)U_\tau(n)\delta_{n,n+t} \\
+ &\left(\cos(\frac{a\Omega}{2}) - 1 - i\sin(\frac{a\Omega}{2})\sigma^{12}\right)(1+\gamma_4)U_{-\tau}(n)\delta_{n-t,n}\Bigg]
\end{aligned}
\tag{236}
$$

Another way to do so

$$
\begin{aligned}
&-\frac{U_\mu(n)\psi(n+a\mu)+U_{-\mu}(n)\psi(n-a\mu)-2\psi(n)}{2a}\to 0 \\
&\psi(n)\to\left(U_\mu(n)\psi(n+a\mu)+U_{-\mu}(n)\psi(n-a\mu)\right)
\end{aligned}
\tag{237}
$$

so

$$
\begin{aligned}
&\kappa\gamma_4 ia\Omega\sigma^{12}\psi(n)=-\kappa\left(-ia\Omega\sigma^{12}\right)\gamma_4\psi(n) \\
&\approx-\kappa\left(\frac{-ia\Omega\sigma^{12}}{2}\right)\left(\gamma_4U_\mu(n)\psi(n+\tau)+\gamma_4U_{-\mu}(n)\psi(n-\tau)\right) \\
&\approx-\kappa\left(\frac{-ia\Omega\sigma^{12}}{2}\right)\left(\gamma_4U_\mu(n)\psi(n+\tau)+\gamma_4U_{-\mu}(n)\psi(n-\tau)-U_\mu(n)\psi(n+\tau)+U_{-\mu}(n)\psi(n-\tau)\right) \\
&=-\kappa\left(\frac{-ia\Omega\sigma^{12}}{2}\right)\left((\gamma_4-1)U_\mu(n)\psi(n+\tau)+(\gamma_4+1)U_{-\mu}(n)\psi(n-\tau)\right) \\
&=-\kappa\left(\frac{-ia\Omega\sigma^{12}}{2}(\gamma_4-1)U_\mu(n)\psi(n+\tau)+\frac{-ia\Omega\sigma^{12}}{2}(\gamma_4+1)U_{-\mu}(n)\psi(n-\tau)\right) \\
&=-\kappa\left(\frac{ia\Omega\sigma^{12}}{2}(1-\gamma_4)U_\mu(n)\psi(n+\tau)+\frac{-ia\Omega\sigma^{12}}{2}(1+\gamma_4)U_{-\mu}(n)\psi(n-\tau)\right)
\end{aligned}
\tag{238}
$$

so

$$
\begin{aligned}
&-\kappa\gamma_4X_4+\kappa\gamma_4 ia\Omega\sigma^{12}\psi(n) \\
&=-\kappa\left((1-\gamma_4)U_\mu(n)\psi(n+\tau)+(1+\gamma_4)U_{-\mu}(n)\psi(n-\tau)\right) \\
&-\kappa\left(\frac{ia\Omega\sigma^{12}}{2}(1-\gamma_4)U_\mu(n)\psi(n+\tau)+\frac{-ia\Omega\sigma^{12}}{2}(1+\gamma_4)U_{-\mu}(n)\psi(n-\tau)\right) \\
&=-\kappa\left(\left(1+\frac{ia\Omega\sigma^{12}}{2}\right)(1-\gamma_4)U_\mu(n)\psi(n+\tau)+\left(1-\frac{ia\Omega\sigma^{12}}{2}\right)(1+\gamma_4)U_{-\mu}(n)\psi(n-\tau)\right) \\
&\approx-\kappa\left(e^{\frac{ia\Omega\sigma^{12}}{2}}(1-\gamma_4)U_\mu(n)\psi(n+\tau)+e^{\frac{-ia\Omega\sigma^{12}}{2}}(1+\gamma_4)U_{-\mu}(n)\psi(n-\tau)\right)
\end{aligned}
\tag{239}
$$

which go back to the σ^{12} term.

We still check the γ_5-hermiticity, using

$$
\begin{aligned}
&\sum_{n=1,2,3,m=1,2,3}\left(U_\mu(n)\delta_{n+1,m}+U_{-\mu}(n)\delta_{n-1,m}\right)^\dagger=\sum_{n=1,2,3,m=1,2,3}\left(U_\mu(n)\delta_{n+1,m}+U_{-\mu}(n)\delta_{n-1,m}\right) \\
&\gamma_5\gamma_4\gamma_5=-\gamma_4^\dagger \\
&\gamma_5\frac{ia\Omega\sigma^{12}}{2}\gamma_5=-\left(\frac{ia\Omega\sigma^{12}}{2}\right)^\dagger
\end{aligned}
\tag{240}
$$

It is γ_5-hermite.

8.1.4 The final action of rotation

Defining $\frac{\beta}{N_c} \equiv \frac{2}{a^4 g_{YM}^2}$, $\kappa \equiv \frac{1}{2am+8}$, $\hat{\mu} \equiv \frac{\mu}{a}$, $\hat{\Omega} \equiv a\Omega$, (here $\mu = x, y, z, t$ is the coordinate) we have

$$Z = \exp(-S_G - S_F) \tag{241}$$

with

$$\begin{aligned}
&S_G = \frac{\beta}{N_c} \sum_n \Bigg(\sum_{\mu>\nu} \mathrm{Retr}[1 - U_{\mu\nu}(n)] + \hat{\Omega}\left(\hat{x}\mathrm{Retr}[V_{412} + V_{432}] - \hat{y}\mathrm{Retr}[V_{421} + V_{431}]\right) \\
&+\hat{\Omega}^2(\hat{x}^2\mathrm{Retr}[1 - \bar{U}_{14}(n)] + \hat{y}^2\mathrm{Retr}[1 - \bar{U}_{24}(n)] + (\hat{x}^2 + \hat{y}^2)\mathrm{Retr}[1 - \bar{U}_{34}(n)] + \hat{x}\hat{y}\mathrm{Retr}[V_{142}]\Bigg) \\
&U_{\mu,\nu}(n) \equiv U_\mu(n)U_\nu(n + a\hat{\mu})U_\mu^{-1}(n + a\hat{\nu})U_\nu^{-1}(n) \\
&\bar{U}_{\mu,\nu}(n) \equiv \frac{1}{4}\left(U_{\mu,\nu}(n) + U_{-\mu,\nu}(n) + U_{\mu,-\nu}(n) + U_{-\mu,-\nu}(n)\right) \\
&V_{\mu\nu\sigma}(n) = \frac{1}{8}\left((U_{\mu,\nu}(n) - U_{-\mu,\nu}(n))(U_{\nu,\sigma}(n) - U_{\nu,-\sigma}(n))\right. \\
&\left.+(U_{\mu,-\nu}(n) - U_{-\mu,-\nu}(n))(U_{-\nu,\sigma}(n) - U_{-\nu,-\sigma}(n))\right)
\end{aligned} \tag{242}$$

and

$$\begin{aligned}
S_F &= \sum_{n,m} \bar{\psi}(n)D(n|m)\psi(m) \\
D(n|m) &= \left(1 + 2\kappa(\hat{y} - \hat{x})\hat{\Omega} + i\kappa\gamma_4\hat{\Omega}\sigma^{12}\right)\delta_{n,m} \\
&- \kappa\sum_\mu [(1 - \gamma_\mu)U_\mu(n)\delta_{n+a\hat{\mu},m} + (1 + \gamma_\mu)U_{-\mu}(n)\delta_{n-a\hat{\mu},m}] \\
&- \kappa\hat{y}\hat{\Omega}\left((1 - \gamma_4)U_x(n)\delta_{n+a\hat{x},m} + (1 + \gamma_4)U_{-x}(n)\delta_{n-a\hat{x},m}\right) \\
&+ \kappa\hat{x}\hat{\Omega}\left((1 - \gamma_4)U_y(n)\delta_{n+a\hat{y},m} + (1 + \gamma_4)U_{-y}(n)\delta_{n-a\hat{y},m}\right)
\end{aligned} \tag{243}$$

such that $\gamma_5 D\gamma_5 = D^\dagger$.

or (As in Ref. [23], the naive discretization is used.)

$$\begin{aligned}
&S_F = \sum_{n,m} \bar{\psi}(n) D(n|m) \psi(m) \\
&D(n|m) = \left(1 + i\kappa\gamma_4 \hat{\Omega}\sigma^{12}\right) \delta_{n,m} \\
&- \kappa \sum_{\mu} \left((1-\gamma_\mu) U_\mu(n) \delta_{n+a\hat{\mu},m} + (1+\gamma_\mu) U_{-\mu}(n) \delta_{n-a\hat{\mu},m}\right) \\
&- \kappa \hat{y}\hat{\Omega} \left((-\gamma_4) U_x(n) \delta_{n+a\hat{x},m} + (+\gamma_4) U_{-x}(n) \delta_{n-a\hat{x},m}\right) \\
&+ \kappa \hat{x}\hat{\Omega} \left((-\gamma_4) U_y(n) \delta_{n+a\hat{y},m} + (+\gamma_4) U_{-y}(n) \delta_{n-a\hat{y},m}\right)
\end{aligned} \tag{244}$$

If using the exponential spin coupling term it is

$$\begin{aligned}
&S_F = \sum_{n,m} \bar{\psi}(n) D(n|m) \psi(m) \\
&D(n|m) = \delta_{n,m} - \kappa \sum_{\mu=1,2,3} \left((1-\gamma_\mu) U_\mu(n) \delta_{n+a\hat{\mu},m} + (1+\gamma_\mu) U_{-\mu}(n) \delta_{n-a\hat{\mu},m}\right) \\
&- \kappa \hat{y}\hat{\Omega} \left((-\gamma_4) U_x(n) \delta_{n+a\hat{x},m} + (+\gamma_4) U_{-x}(n) \delta_{n-a\hat{x},m}\right) \\
&+ \kappa \hat{x}\hat{\Omega} \left((-\gamma_4) U_y(n) \delta_{n+a\hat{y},m} + (+\gamma_4) U_{-y}(n) \delta_{n-a\hat{y},m}\right) \\
&- \kappa \left(e^{\frac{ia\Omega\sigma^{12}}{2}} (1-\gamma_4) U_\tau(n) \delta_{n,n+t} + e^{\frac{-ia\Omega\sigma^{12}}{2}} (1+\gamma_4) U_{-\tau}(n) \delta_{n-t,n}\right)
\end{aligned} \tag{245}$$

8.1.5 The force from gauge action

$$\begin{aligned}
&U_{\mu,\nu}(n) = U_\mu(n) U_\nu(n+a\mu) U_\mu^{-1}(n+a\nu) U_\nu^{-1}(n). \;\; U_{\mu,\nu}^\dagger(n) = U_{\mu,\nu}^{-1}(n) \\
&U_{\mu,\nu}^{-1}(n) = U_\nu(n) U_\mu(n+a\nu) U_\nu^{-1}(n+a\mu) U_\mu^{-1}(n) = U_{\nu,\mu}(n). \\
&\mathrm{tr}[U_{\nu,\mu}(n)] = \mathrm{tr}\left[U_\nu(n) U_\mu(n+a\nu) U_\nu^{-1}(n+a\mu) U_\mu^{-1}(n)\right]. \\
&\mathrm{tr}[U_{-\mu,\nu}(n)] = \mathrm{tr}\left[U_\nu(n-a\mu) U_{-\mu}^{-1}(n+a\nu+a\mu-a\mu) U_\nu^{-1}(n) U_{-\mu}(n)\right] \\
&= \mathrm{tr}\left[U_\nu(n-a\mu) U_\mu(n+a\nu-a\mu) U_\nu^{-1}(n) U_\mu^{-1}(n-a\mu)\right] = \mathrm{tr}[U_{\nu,\mu}(n-a\mu)] = \mathrm{tr}[U_{\mu,\nu}^\dagger(n-a\mu)] \\
&\mathrm{tr}[U_{\mu,-\nu}(n)] = \mathrm{tr}[U_{-\nu,\mu}^\dagger(n)] = (\mathrm{tr}[U_{-\nu,\mu}(n)])^* = (\mathrm{tr}[U_{\mu,\nu}(n-a\nu)])^* = \mathrm{tr}[U_{\mu,\nu}^\dagger(n-a\nu)] \\
&\mathrm{tr}[U_{-\mu,-\nu}(n)] = \mathrm{tr}[U_{\mu,\nu}(n-a\mu-a\nu)]
\end{aligned} \tag{246}$$

so

$$\mathrm{Retr}[\bar{U}_{\mu,\nu}(n)] = \frac{1}{4}\mathrm{Retr}[U_{\mu\nu}(n) + U_{\mu\nu}(n-a\mu) + U_{\mu\nu}(n-a\nu) + U_{\mu\nu}(n-a\mu-a\nu)] \tag{247}$$

and

$$\sum_n f(n)\mathrm{Retr}[1-\bar{U}_{\mu,\nu}(n)] = \sum_n \frac{f(n)+f(n+a\mu)+f(n+a\nu)+f(n+a\mu+a\nu)}{4}\mathrm{Retr}[1-U_{\mu\nu}(n)] \tag{248}$$

so (**Note, this is for infinite lattice size, the boundary condition should be considered**)

$$\begin{aligned}
\sum_n \hat{\Omega}^2\hat{x}^2\mathrm{Retr}[1-\bar{U}_{1,4}(n)] &= \sum_n \hat{\Omega}^2\frac{2\hat{x}^2+2\hat{x}+1}{2}\mathrm{Retr}[1-U_{1,4}(n)]\\
\sum_n \hat{\Omega}^2\hat{y}^2\mathrm{Retr}[1-\bar{U}_{2,4}(n)] &= \sum_n \hat{\Omega}^2\frac{2\hat{y}^2+2\hat{y}+1}{2}\mathrm{Retr}[1-U_{2,4}(n)]\\
\sum_n \hat{\Omega}^2(\hat{x}^2+\hat{y}^2)\mathrm{Retr}[1-\bar{U}_{3,4}(n)] &= \sum_n \hat{\Omega}^2(\hat{x}^2+\hat{y}^2)\mathrm{Retr}[1-U_{3,4}(n)]
\end{aligned} \tag{249}$$

Using

$$\begin{aligned}
&\mathrm{Retr}[U_{\mu,\nu}(n-a\nu)] = \mathrm{Retr}[U_\mu(n-a\nu)U_\nu(n+a\mu-a\nu)U_\mu^{-1}(n)U_\nu^{-1}(n-a\nu)]\\
&= \mathrm{Retr}[U_\nu(n-a\nu)U_\mu(n)U_\nu^{-1}(n+a\mu-a\nu)U_\mu^{-1}(n-a\nu)]\\
&= \mathrm{Retr}[U_\mu(n)U_\nu^{-1}(n+a\mu-a\nu)U_\mu^{-1}(n-a\nu)U_\nu(n-a\nu)]
\end{aligned} \tag{250}$$

$$\begin{aligned}
&\sum_n g(n)\mathrm{Retr}[1-U_{\mu,\nu}(n)] = N\times N_c - \sum_n \mathrm{Retr}\left[U_\mu(n)\Sigma_\mu^\dagger(n)\right]\\
&\Sigma_{\mu,i}(n,\nu) = g_i(n)U_\nu(n)U_\mu(n+a\nu)U_\nu^{-1}(n+a\mu) + g_i(n-a\nu)U_\nu^{-1}(n-a\nu)U_\mu(n-a\nu)U_\nu(n+a\mu-a\nu)
\end{aligned} \tag{251}$$

The product is just same as the definition of staples. However, there are two differences, (i) there is a coefficient function for each term of the sum. (ii) there is no sum over ν.

The following is usual, with the new definition of the staple, one have

$$F_\mu(n) = -\frac{\beta}{2N_c}\left\{U_\mu(n)\Sigma_{\mu,i}^\dagger(n,\nu)\right\}_{TA} \tag{252}$$

with $i = 1,2,3$ and (**Note, this is for infinite lattice size, the boundary condition should be considered**)

$$\begin{aligned}
&g_1(n) = \frac{\Omega^2(2x^2+2x+1)}{2},\;\; g_2(n) = \frac{\Omega^2(2y^2+2y+1)}{2},\;\; g_3(n) = \Omega^2(x^2+y^2),\\
&F_{\mu=1,2,3}(n) = -\frac{\beta}{2N_c}\left\{U_\mu(n)\Sigma_{\mu,\mu}^\dagger(n,4)\right\}_{TA}\\
&F_4(n) = -\frac{\beta}{2N_c}\left\{U_4(n)\sum_{i=1,2,3}\Sigma_{4,i}^\dagger(n,i)\right\}_{TA}
\end{aligned} \tag{253}$$

Now we consider the force of V

$$\begin{aligned} V_{\mu\nu\sigma} = \frac{1}{8}\,(& U_{\mu,\nu}U_{\nu,\sigma} + U_{-\mu,\nu}U_{\nu,-\sigma} + U_{\mu,-\nu}U_{-\nu,\sigma} + U_{-\mu,-\nu}U_{-\nu,-\sigma} \\ & - U_{\mu,\nu}U_{\nu,-\sigma} - U_{-\mu,\nu}U_{\nu,\sigma} - U_{\mu,-\nu}U_{-\nu,-\sigma} - U_{-\mu,-\nu}U_{-\nu,\sigma}) \end{aligned} \tag{254}$$

Using

$$\mathrm{Retr}[U_{\mu,\nu}U_{\nu,\sigma}] = \mathrm{Retr}[U_{\sigma,\nu}U_{\nu,\mu}], \;\; \mathrm{Retr}[U_{\mu,\nu}U_{\nu,-\sigma}] = \mathrm{Retr}[U_{-\sigma,\nu}U_{\nu,\mu}] \tag{255}$$

one have

$$\mathrm{Retr}[V_{\mu\nu\rho}] = \mathrm{Retr}[V_{\rho\nu\mu}] \tag{256}$$

So we only need to calculate $\frac{\partial}{\partial\omega_\mu}V_{\mu\nu\rho}$ and $\frac{\partial}{\partial\omega_\nu}V_{\mu\nu\rho}$.

One can find

$$\sum_n \mathrm{Retr}[g(n)V_{\mu\nu\rho}(n)] \to S[U_\mu(n)] = \mathrm{Retr}[U_\mu(n)M(n)] \tag{257}$$

with

$$\begin{aligned} M(n) = \frac{1}{8}\big(&(g(n)+g(n+a\nu))U_\nu(n+a\mu)U_\mu^{-1}(n+a\nu)U_\rho(n+a\nu)U_\nu^{-1}(n+a\rho)U_\rho^{-1}(n) \\ &+(g(n)+g(n-a\nu))U_\nu^{-1}(n+a\mu-a\nu)U_\mu^{-1}(n-a\nu)U_\rho(n-a\nu)U_\nu(n-a\nu+a\rho)U_\rho^{-1}(n) \\ &+(g(n+a\mu-a\nu)+g(n+a\mu))U_\rho^{-1}(n+a\mu-a\rho)U_\nu^{-1}(n+a\mu-a\rho-a\nu) \\ &\times U_\rho(n+a\mu-a\rho-a\nu)U_\mu^{-1}(n-a\nu)U_\nu(n-a\nu) \\ &+(g(n+a\mu+a\nu)+g(n+a\mu))U_\rho^{-1}(n+a\mu-a\rho)U_\nu(n+a\mu-a\rho) \\ &\times U_\rho(n+a\mu-a\rho+a\nu)U_\mu^{-1}(n+a\nu)U_\nu^{-1}(n) \\ &-(g(n+a\mu)+g(n+a\mu+a\nu))U_\rho(n+a\mu)U_\nu(n+a\mu+a\rho)U_\rho^{-1}(n+a\mu+a\nu)U_\mu^{-1}(n+a\nu)U_\nu^{-1}(n) \\ &-(g(n+a\mu)+g(n+a\mu-a\nu))U_\rho(n+a\mu)U_\nu^{-1}(n+a\mu+a\rho-a\nu) \\ &\times U_\rho^{-1}(n+a\mu-a\nu)U_\mu^{-1}(n-a\nu)U_\nu(n-a\nu) \\ &-(g(n)+g(n+a\nu))U_\nu(n+a\mu)U_\mu^{-1}(n+a\nu)U_\rho^{-1}(n+a\nu-a\rho)U_\nu^{-1}(n-a\rho)U_\rho(n-a\rho) \\ &-(g(n)+g(n-a\nu))U_\nu^{-1}(n+a\mu-a\nu)U_\mu^{-1}(n-a\nu)U_\rho^{-1}(n-a\nu-a\rho)U_\nu(n-a\nu-a\rho)U_\rho(n-a\rho)\big) \end{aligned} \tag{258}$$

It can be simplified as

$$
\begin{aligned}
&M(n) = \frac{1}{8}\big((g(n) + g(n + a\nu))U_\nu(n + a\mu)U_\mu^{-1}(n + a\nu)S_1\\
&+(g(n) + g(n - a\nu))U_\nu^{-1}(n + a\mu - a\nu)U_\mu^{-1}(n - a\nu)S_2\\
&+(g(n + a\mu) + g(n + a\mu + a\nu))S_3U_\mu^{-1}(n + a\nu)U_\nu^{-1}(n)\\
&+(g(n + a\mu) + g(n + a\mu - a\nu))S_4U_\mu^{-1}(n - a\nu)U_\nu(n - a\nu)\big)\\
&S_1 = U_\rho(n + a\nu)U_\nu^{-1}(n + a\rho)U_\rho^{-1}(n) - U_\rho^{-1}(n + a\nu - a\rho)U_\nu^{-1}(n - a\rho)U_\rho(n - a\rho)\\
&S_2 = U_\rho(n - a\nu)U_\nu(n - a\nu + a\rho)U_\rho^{-1}(n) - U_\rho^{-1}(n - a\nu - a\rho)U_\nu(n - a\nu - a\rho)U_\rho(n - a\rho)\\
&S_3 = U_\rho^{-1}(n + a\mu - a\rho)U_\nu(n + a\mu - a\rho)U_\rho(n + a\mu - a\rho + a\nu)\\
&- U_\rho(n + a\mu)U_\nu(n + a\mu + a\rho)U_\rho^{-1}(n + a\mu + a\nu)\\
&S_4 = U_\rho^{-1}(n + a\mu - a\rho)U_\nu^{-1}(n + a\mu - a\rho - a\nu)U_\rho(n + a\mu - a\rho - a\nu)\\
&- U_\rho(n + a\mu)U_\nu^{-1}(n + a\mu + a\rho - a\nu)U_\rho^{-1}(n + a\mu - a\nu)
\end{aligned}
\tag{259}
$$

Similarly, one also have

$$
\sum_n \mathrm{Retr}[g(n)V_{\mu\nu\rho}(n)] \to S[U_\nu(n)] = \mathrm{Retr}[U_\nu(n)N(n)] \tag{260}
$$

where

$$
\begin{aligned}
&N(n) = \frac{1}{8}\big\{(g(n + a\mu) + g(n + a\nu + a\mu))U_\mu(n + a\nu)T_1U_\mu^{-1}(n)\\
&+(g(n - a\mu) + g(n + a\nu - a\mu))U_\mu^{-1}(n + a\nu - a\mu)T_2U_\mu(n - a\mu)\\
&+(g(n + a\rho) + g(n + a\nu + a\rho))U_\rho(n + a\nu)T_3U_\rho^{-1}(n)\\
&+(g(n - a\rho) + g(n + a\nu - a\rho))U_\rho^{-1}(n + a\nu - a\rho)T_4U_\rho(n - a\rho)\big\}\,,\\
&T_1 = U_\rho^{-1}(n + a\nu + a\mu - a\rho)U_\nu^{-1}(n + a\mu - a\rho)U_\rho(n + a\mu - a\rho)\\
&- U_\rho(n + a\nu + a\mu)U_\nu^{-1}(n + a\mu + a\rho)U_\rho^{-1}(n + a\mu),\\
&T_2 = U_\rho(n + a\nu - a\mu)U_\nu^{-1}(n - a\mu + a\rho)U_\rho^{-1}(n - a\mu)\\
&- U_\rho^{-1}(n + a\nu - a\mu - a\rho)U_\nu^{-1}(n - a\mu - a\rho)U_\rho(n - a\mu - a\rho),\\
&T_3 = U_\mu^{-1}(n + a\nu + a\rho - a\mu)U_\nu^{-1}(n + a\rho - a\mu)U_\mu(n + a\rho - a\mu)\\
&- U_\mu(n + a\nu + a\rho)U_\nu^{-1}(n + a\mu + a\rho)U_\mu^{-1}(n + a\rho),\\
&T_4 = U_\mu(n + a\nu - a\rho)U_\nu^{-1}(n - a\rho + a\mu)U_\mu^{-1}(n - a\rho)\\
&- U_\mu^{-1}(n + a\nu - a\mu - a\rho)U_\nu^{-1}(n - a\mu - a\rho)U_\mu(n - a\mu - a\rho),
\end{aligned}
\tag{261}
$$

One can further reduce the dagger operation by defining

$$S[U_\mu(n)] = \mathrm{Retr}[U_\mu(n)M^\dagger(n)], \quad S[U_\nu(n)] = \mathrm{Retr}[U_\nu(n)N^\dagger(n)] \tag{262}$$

with

$$\begin{aligned}
&M(n) = \frac{1}{8}\big((g(n)+g(n+a\nu))S_1U_\mu(n+a\nu)U_\nu^{-1}(n+a\mu) \\
&+(g(n)+g(n-a\nu))S_2U_\mu(n-a\nu)U_\nu(n+a\mu-a\nu) \\
&+(g(n+a\mu)+g(n+a\mu+a\nu))U_\nu(n)U_\mu(n+a\nu)S_3 \\
&+(g(n+a\mu)+g(n+a\mu-a\nu))U_\nu^{-1}(n-a\nu)U_\mu(n-a\nu)S_4\big) \\
&S_1 = U_\rho(n)U_\nu(n+a\rho)U_\rho^{-1}(n+a\nu) - U_\rho^{-1}(n-a\rho)U_\nu(n-a\rho)U_\rho(n+a\nu-a\rho) \\
&S_2 = U_\rho(n)U_\nu^{-1}(n-a\nu+a\rho)U_\rho^{-1}(n-a\nu) - U_\rho^{-1}(n-a\rho)U_\nu^{-1}(n-a\nu-a\rho)U_\rho(n-a\nu-a\rho) \\
&S_3 = U_\rho^{-1}(n+a\mu-a\rho+a\nu)U_\nu^{-1}(n+a\mu-a\rho)U_\rho(n+a\mu-a\rho) \\
&-U_\rho(n+a\mu+a\nu)U_\nu^{-1}(n+a\mu+a\rho)U_\rho^{-1}(n+a\mu) \\
&S_4 = U_\rho^{-1}(n+a\mu-a\rho-a\nu)U_\nu(n+a\mu-a\rho-a\nu)U_\rho(n+a\mu-a\rho) \\
&-U_\rho(n+a\mu-a\nu)U_\nu(n+a\mu+a\rho-a\nu)U_\rho^{-1}(n+a\mu)
\end{aligned} \tag{263}$$

and

$$\begin{aligned}
&N(n) = \frac{1}{8}(N(\mu,\rho)(n) + N(\rho,\mu)(n)) \\
&N(\mu,\rho)(n) = \big\{(g(n+a\mu)+g(n+a\nu+a\mu))U_\mu(n)T_1U_\mu^{-1}(n+a\nu) \\
&+(g(n-a\mu)+g(n+a\nu-a\mu))U_\mu^{-1}(n-a\mu)T_2U_\mu(n+a\nu-a\mu)\big\}, \\
&T_1 = U_\rho^{-1}(n+a\mu-a\rho)U_\nu(n+a\mu-a\rho)U_\rho(n+a\nu+a\mu-a\rho) \\
&-U_\rho(n+a\mu)U_\nu(n+a\mu+a\rho)U_\rho^{-1}(n+a\nu+a\mu), \\
&T_2 = U_\rho(n-a\mu)U_\nu(n-a\mu+a\rho)U_\rho^{-1}(n+a\nu-a\mu) \\
&-U_\rho^{-1}(n-a\mu-a\rho)U_\nu(n-a\mu-a\rho)U_\rho(n+a\nu-a\mu-a\rho),
\end{aligned} \tag{264}$$

Note, instead of $S_G[U_\mu] = -U_\mu\Sigma_\mu^\dagger$, here it is $S_G[U_\mu] = +U_\mu M_\mu^\dagger$ and $S_G[U_\mu] = +U_\mu N_\mu^\dagger$.

8.1.6 The force from fermion action

The first step is to shift the second term to factorize $U_\mu(n)$ out

$$\begin{aligned}&\sum_{m,n} g(n)(1+\gamma_\mu)U_{-\mu}(n)\delta_{n-a\mu,m} = \sum_{m,n} g(n)(1+\gamma_\mu)U_\mu^{-1}(n-a\mu)\delta_{n-a\mu,m}\\&= \sum_{m,n} g(n+a\mu)(1+\gamma_\mu)U_\mu^{-1}(n)\delta_{n,m}\end{aligned} \tag{265}$$

It is more convenient to split the D operator as (note that for yU_x, $g(n)=y$, and $g(n)=g(n+x)$, xU_y is similar. Also, note that, it is also true for open boundary)

$$\begin{aligned}&M^a = \left\{(1-\gamma_\mu)T^a U_\mu \delta_{x_L,(x+\mu)_R} - (1+\gamma_\mu)U_\mu^{-1}T^a\delta_{(x+\mu)_L,x_R}\right\}\\&- y\Omega\delta_{\mu,x}\left\{(1-\gamma_4)T^a U_\mu \delta_{x_L,(x+\mu)_R} - (1+\gamma_4)U_\mu^{-1}T^a\delta_{(x+\mu)_L,x_R}\right\}\\&+ x\Omega\delta_{\mu,y}\left\{(1-\gamma_4)T^a U_\mu \delta_{x_L,(x+\mu)_R} - (1+\gamma_4)U_\mu^{-1}T^a\delta_{(x+\mu)_L,x_R}\right\}\\&F_{pf} = 2i\kappa\sum_a \mathrm{Im}\left[\left(\phi_1^\dagger M_a\phi_2\right)\right]T_a\end{aligned} \tag{266}$$

with

$$\phi_1 = \left(\left(\hat{D}\hat{D}^\dagger\right)^{-1}\phi\right),\ \ \phi_2 = \hat{D}^\dagger\phi_1, \tag{267}$$

similarly, with

$$\begin{aligned}&\phi_{L1}(n) = \phi_1(n),\ \ \phi_{R1}(n) = \{(1-\gamma_\mu) + (x\Omega\delta_{\mu,y} - y\Omega\delta_{\mu,x})(1-\gamma_4)\}\,\phi_2(n+\mu),\\&\phi_{L2}(n) = \phi_1(n+\mu),\ \ \phi_{R1}(n) = \{(1+\gamma_\mu) + (x\Omega\delta_{\mu,y} - y\Omega\delta_{\mu,x})(1+\gamma_4)\}\,\phi_2(n),\end{aligned} \tag{268}$$

$$F_\mu^{pf}(n) = \kappa\left\{U_\mu(n)\left(\phi_{R1}\phi_{L1}^\dagger + \phi_{R2}\phi_{L2}^\dagger\right)\right\}\Big|_{TA} \tag{269}$$

Note that **Both the force from gauge and fermion actions are kept anti-hermitian traceless.**

8.1.7 The angular momentum

The angular momentum operator is defined as

$$
\begin{aligned}
& J \equiv \left.\frac{\delta \mathcal{L}}{\delta \Omega}\right|_{\Omega=0} \\
& = J_G + J_{FL} + J_{FS} \\
& J_G = \frac{\beta}{N_c}\sum_n \left(\hat{x}\mathrm{Retr}[V_{412}(n)+V_{432}(n)] - \hat{y}\mathrm{Retr}[V_{421}(n)+V_{431}(n)]\right) \\
& J_{FL} = \bar{\psi}\left\{-\kappa\hat{y}\left((-\gamma_4)U_x(n)\delta_{n+a\hat{x},m} + (+\gamma_4)U_{-x}(n)\delta_{n-a\hat{x},m}\right)\right. \\
& \left. + \hat{x}\left((-\gamma_4)U_y(n)\delta_{n+a\hat{y},m} + (+\gamma_4)U_{-y}(n)\delta_{n-a\hat{y},m}\right)\right\} \\
& = -\kappa\bar{\psi}\gamma_4(\hat{y}D_x - xD_y)\psi, \\
& J_{FS} = -i\kappa\bar{\psi}\gamma_4\sigma^{12}\psi.
\end{aligned}
\tag{270}
$$

The result is derived as $\delta\mathcal{L}/\delta\hat{\Omega}$, therefor, the result has unit as a^{-3}.

The measurement of $\langle J_G\rangle$ is straightforward. The measurement of J_{FL} and J_{FS} are inertia mass densities of quark-antiquark pairs. So, for the $u-\bar{u}$ pair. On the other hand, J is **NOT** a local operator. $\langle J_F(n|m)\rangle$ can be written as (in the spinor space, where a,b are spinor indices)

$$
\begin{aligned}
& \langle J_F(n|m)\rangle = \langle \bar{u}(n)O(n|m)u(m)\rangle \\
& = \sum_{a,b} O_{a,b}(n|m)\langle \bar{u}_a(n)u_b(m)\rangle = -\sum_{a,b} O_{a,b}(n|m)\langle u_b(m)\bar{u}_a(n)\rangle \\
& = -\sum_{a,b} O_{a,b}(n|m)D^{-1}(m|n)_{b,a} = -\mathrm{tr}_{c,s}\left[O(n|m)D^{-1}(m|n)\right]
\end{aligned}
\tag{271}
$$

So the definition of local angular momentum density should be

$$
\langle J_F(n)\rangle = -\sum_{a,b,m} O_{a,b}(n|m)D^{-1}(m|n)_{b,a} = -\mathrm{tr}_{c,s,m}\left[O(n|m)D^{-1}(m|n)\right] \tag{272}
$$

So, we need to calculate $\sum_n O(m_0|n)D^{-1}(n|m_0)$ as a matrix in color and spinor space. It can be done by introduce the source

$$
\phi^S_{m_0,c_2,s_2}(m)_{c,s} = \delta(m-m_0)\delta(c-c_2)\delta(s-s_2) \tag{273}
$$

and $D^{-1}(n|m_0)_{c,s}$ as a vector $\vec{v}(n)$ can be written as

$$\begin{pmatrix} D^{-1}_{1,cs}(n) \\ D^{-1}_{2,cs}(n) \\ D^{-1}_{3,cs}(n) \\ \dots \\ D^{-1}_{10,cs}(n) \\ D^{-1}_{11,cs}(n) \\ D^{-1}_{12,cs_2}(n) \end{pmatrix} = \begin{pmatrix} D^{-1}_{1,1} & D^{-1}_{1,2} & \dots & D^{-1}_{1,cs}(n|m_0) & \dots & D^{-1}_{1,11} & D^{-1}_{1,12} \\ D^{-1}_{2,1} & D^{-1}_{2,2} & \dots & D^{-1}_{2,cs}(n|m_0) & \dots & D^{-1}_{2,11} & D^{-1}_{2,12} \\ D^{-1}_{3,1} & D^{-1}_{3,2} & \dots & D^{-1}_{3,cs}(n|m_0) & \dots & D^{-1}_{3,11} & D^{-1}_{3,12} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ D^{-1}_{10,1} & D^{-1}_{0,2} & \dots & D^{-1}_{10,cs}(n|m_0) & \dots & D^{-1}_{10,11} & D^{-1}_{10,12} \\ D^{-1}_{11,1} & D^{-1}_{11,2} & \dots & D^{-1}_{11,cs}(n|m_0) & \dots & D^{-1}_{11,11} & D^{-1}_{11,12} \\ D^{-1}_{12,1} & D^{-1}_{12,2} & \dots & D^{-1}_{12,cs}(n|m_0) & \dots & D^{-1}_{12,11} & D^{-1}_{12,12} \end{pmatrix} \begin{pmatrix} 0 \\ \dots \\ 0 \\ 1_{idx=cs,x=m_0} \\ 0 \\ \dots \\ 0 \end{pmatrix} \tag{274}$$

and $\sum_n O(m|n)D^{-1}(n|m_0)_{c,s}$ as a vector $\vec{v}(m)$ can be written as

$$\begin{pmatrix} OD^{-1}_{1,cs}(m) \\ OD^{-1}_{2,cs}(m) \\ \dots \\ OD^{-1}_{11,cs}(m) \\ OD^{-1}_{12,cs_2}(m) \end{pmatrix} = \begin{pmatrix} O_{1,1} & O_{1,2} & \dots & O_{1,11} & O_{1,12} \\ O_{2,1} & O_{2,2} & \dots & O_{2,11} & O_{2,12} \\ O_{3,1} & O_{3,2} & \dots & O_{3,11} & O_{3,12} \\ \dots & \dots & \dots & \dots & \dots \\ O_{10,1} & O_{0,2} & \dots & O_{10,11} & O_{10,12} \\ O_{11,1} & O_{11,2} & \dots & O_{11,11} & O_{11,12} \\ O_{12,1} & O_{12,2} & \dots & O_{12,11} & O_{12,12} \end{pmatrix} \begin{pmatrix} D^{-1}_{1,cs} \\ D^{-1}_{2,cs} \\ \dots \\ D^{-1}_{11,cs} \\ D^{-1}_{12,cs_2} \end{pmatrix} \tag{275}$$

And the trace is just

$$\sum_{i=1}^{12} (OD^{-1})_{i,i}(n) \tag{276}$$

Also, note that, the D and ψ are scaled.

Now, we can consider the physical meaning of J_G. Using

$$F_{0i} = E^i, \quad F_{ij} = 2c\epsilon_{ijk}B_i \tag{277}$$

therefor

$$\begin{aligned} \mathbf{j} &= (x, y, 0) \times (\mathbf{E} \times \mathbf{B}) \\ \mathbf{j} &= (x, y, 0) \times ((F_{01}, F_{02}, F_{03}), F_{23}, F_{31}, F_{12}) \\ j_z &= -2c\left(xF_{01}F_{12} + yF_{02}F_{12} - xF_{03}F_{23} + yF_{03}F_{13}\right) \end{aligned} \tag{278}$$

Details about Angular momentum

In the lattice code, we use

$$F_{\mu\nu} \equiv \partial_\mu A_\nu - \partial_\nu A_\mu + i[A_\mu, A_\nu] = g_{YM} \sum_a T^a F^a_{\mu\nu} \tag{279}$$

and

$$F^a_{0i} = E^a_i, \quad F^a_{ij} = \epsilon_{ijk} B^a_i \tag{280}$$

and

$$\begin{aligned} \mathbf{J}_G &= \sum_a \mathbf{r} \times (\mathbf{E}^a \times \mathbf{B}^a) \\ &= -\begin{pmatrix} y(F^a_{02}F^a_{23} - F^a_{01}F^a_{31}) + z(F^a_{03}F^a_{23} - F^a_{01}F^a_{12}) \\ x(F^a_{01}F^a_{31} - F^a_{02}F^a_{23}) + z(F^a_{03}F^a_{31} - F^a_{02}F^a_{12}) \\ xF^a_{01}F^a_{12} + yF^a_{02}F^a_{12} - xF^a_{03}F^a_{23} + yF^a_{03}F^a_{13} \end{pmatrix} \end{aligned} \tag{281}$$

Using $\mathrm{tr}[T_i T_j] = \frac{1}{2}\delta_{ij}$, one have

$$\frac{2}{g^2_{YM}} \mathrm{tr}\left[F_{\mu\nu} F_{\rho\sigma}\right] = \sum_a F^a_{\mu\nu} F^a_{\rho\sigma} \tag{282}$$

with $\frac{\beta}{N_c} \equiv \frac{2}{g^2_{YM}}$, it is

$$\mathbf{J}_G = -\frac{\beta}{N_c} \mathrm{tr}_c \left[\begin{pmatrix} y(F_{02}F_{23} - F_{01}F_{31}) + z(F_{03}F_{23} - F_{01}F_{12}) \\ x(F_{01}F_{31} - F_{02}F_{23}) + z(F_{03}F_{31} - F_{02}F_{12}) \\ xF_{01}F_{12} + yF_{02}F_{12} - xF_{03}F_{23} + yF_{03}F_{13} \end{pmatrix} \right], \tag{283}$$

After Wick rotation, which is to replace $F_{0i} \to iF_{0i}$, one have

$$\mathbf{J}^E_G = -i\frac{\beta}{N_c} \mathrm{tr}_c \left[\begin{pmatrix} y(F_{02}F_{23} - F_{01}F_{31}) + z(F_{03}F_{23} - F_{01}F_{12}) \\ x(F_{01}F_{31} - F_{02}F_{23}) + z(F_{03}F_{31} - F_{02}F_{12}) \\ xF_{01}F_{12} + yF_{02}F_{12} - xF_{03}F_{23} + yF_{03}F_{13} \end{pmatrix} \right], \tag{284}$$

where tr_c is trace in color space. Then J_{Gz} is

$$J^E_{Gz} = -i\frac{\beta}{N_c} \left(x\mathrm{tr}[F_{01}F_{12}] + y\mathrm{tr}[F_{02}F_{12}] - x\mathrm{tr}[F_{03}F_{23}] + y\mathrm{tr}[F_{03}F_{13}] \right) \tag{285}$$

Then discretize using $\mathrm{Retr}[V_{\mu\nu\rho}] = -a^4 \mathrm{tr}\left[F_{\mu\nu}F_{\nu\rho}\right] + \mathcal{O}(a^6)$, and make the index of time $0 \to 4$, and use dimensionless $\hat{x} = a^{-1}x$, it is

$$J^E_{Gz} = ia^{-3}\frac{\beta}{N_c} \left(\hat{x}\mathrm{Retr}[V_{412} + V_{432}] - \hat{y}\mathrm{Retr}[V_{421} + V_{431}] \right) \tag{286}$$

If we use $\partial\mathcal{L}/\partial\Omega$, it is

$$\begin{aligned} J'_G &= \left.\frac{\partial \mathcal{L}_G}{\partial \Omega}\right|_{\Omega=0} = a\left(\left.\frac{\partial \mathcal{L}_G}{\partial (a\Omega)}\right|_{\Omega=0}\right) \\ &= a^{-3}\left\{\frac{\beta}{N_c}\left(\hat{x}\mathrm{Retr}[V_{412}+V_{432}] - \hat{y}\mathrm{Retr}[V_{421}+V_{431}]\right)\right\} \end{aligned} \tag{287}$$

So, the iJ'_G is just the z component of J_G^E in Ji decomposition [24]

Now, considering

$$\mathbf{L}_F = \frac{1}{i}\psi^\dagger \mathbf{r}\times\mathbf{D}\psi = \frac{1}{i}\bar{\psi}\gamma_0\mathbf{r}\times\mathbf{D}\psi \tag{288}$$

using

$$\langle\bar{\psi}_{c_1,s_1}(m)\psi_{c_2,s_2}(n)\rangle_F \equiv \frac{1}{Z}\int \mathcal{D}[\bar{\psi}\psi]\bar{\psi}_i\psi_j \exp(-\sum_n \bar{\psi}(n)A\psi(n)) = -(A^{-1})_{c_2,s_2;c_1,s_1}(n|m) \tag{289}$$

with $\hat{D} = m - \not{D}$ and Wick rotated $S_F = a^4\sum_n \bar{\psi}(n)\hat{D}\psi(n)$ (here S is not spin but action, once change the integral over a Minkovski space to a summation, the Wick rotation is already assumed. However, Wick rotation will not change $\mathbf{D}$.) Using that

$$\langle\bar{\psi}_{c_1,s_1}(m)\psi_{c_2,s_2}(n)\rangle_F \equiv \frac{1}{Z}\int \mathcal{D}[\bar{\psi}\psi]\bar{\psi}_i\psi_j \exp(-S_F) = -a^{-4}(\hat{D}^{-1})_{c_2,s_2;c_1,s_1}(n|m) \tag{290}$$

therefor

$$\begin{aligned} &\sum_{c_1,c_2,s_1,s_2} \delta_{c_1,c_2}\delta_{s_1,s_2}\langle\bar{\psi}_{c_1,s_1}(m)A_{c_1,s_1;c_2,s_2}(m|n)\psi_{c_2,s_2}(n)\rangle_F \\ &= -a^{-4}(\hat{D}^{-1})_{c_2,s_2;c_1,s_1}(n|m)A_{c_1,s_1;c_2,s_2}(m|n) \end{aligned} \tag{291}$$

consider a local operator it is

$$\begin{aligned} &\sum_{c_1,c_2,s_1,s_2} \delta_{c_1,c_2}\delta_{s_1,s_2}\langle\bar{\psi}_{c_1,s_1}(n)A_{c_1,s_1;c_2,s_2}(n|n)\psi_{c_2,s_2}(n)\rangle_F \\ &= -a^{-4}(\hat{D}^{-1})_{c_2,s_2;c_1,s_1}(n|n)A_{c_1,s_1;c_2,s_2}(n|n) = -a^{-4}\mathrm{tr}_{c,s}\left[A(n)\hat{D}^{-1}(n)\right] \end{aligned} \tag{292}$$

Specifically, we are considering

$$\langle L^E_{F_z}\rangle = ia^{-4}\mathrm{tr}_{c,s}\left[\gamma^0\left(x\left(\partial_y + iA_y(n)\right) - y\left(\partial_x + iA_x(n)\right)\right)\hat{D}^{-1}(n)\right] \tag{293}$$

The D operator and ψ fields are scaled ones. The $\tilde{D}$ and $\tilde{\psi}$ are the original ones.

The discretized Wilson operator is

$$\begin{aligned}
&D_W \approx \frac{1}{m+\frac{4}{a}}\hat{D}\\
&D_W(n|m) = \delta_{n,m} - \kappa\sum_{\mu}\left((1-\gamma_\mu)U_\mu(n)\delta_{n+a\mu,m} + (1+\gamma_\mu)U_{-\mu}(n)\delta_{n-a\mu,m}\right)\\
&+ y\Omega\gamma_4\kappa\left(U_x(n)\delta_{n+ax,m} - U_{-x}(n)\delta_{n-ax,m}\right) - x\Omega\gamma_4\kappa\left(U_y(n)\delta_{n+ay,m} - U_{-y}(n)\delta_{n-ay,m}\right) - i\kappa\gamma_4 a\Omega\sigma^{12}\delta_{n,m}\\
&\kappa = \frac{1}{2am+8}
\end{aligned} \tag{294}$$

so

$$\langle L^E_{F_z}\rangle = ia^{-4}\frac{1}{m+\frac{4}{a}}\mathrm{tr}_{c,s}\left[\gamma^0\left(x\left(\partial_y + iA_y(n)\right) - y\left(\partial_x + iA_x(n)\right)\right)\hat{D}_W^{-1}(n)\right] \tag{295}$$

and use

$$(\partial_\mu + iA_\mu)\psi(n) = \frac{U_\mu(n)\psi(n+a\mu) - U_{-\mu}(n)\psi(n-a\mu)}{2a} + \mathcal{O}(a) \tag{296}$$

so

$$\begin{aligned}
&\langle L^E_{F_z}\rangle = ia^{-3}\kappa\mathrm{tr}_{c,s}\left[\gamma^4\left(\hat{x}\left(U_y(n)\delta_{n+ay,m} - U_{-y}(n)\delta_{n-ay,m}\right)\right.\right.\\
&\left.\left.-\hat{y}\left(U_x(n)\delta_{n+ax,m} - U_{-x}(n)\delta_{n-ax,m}\right)\right)\hat{D}_W^{-1}(n)\right]
\end{aligned} \tag{297}$$

Similarly, using

$$\begin{aligned}
&\left.\frac{\partial\mathcal{L}}{\partial\Omega}\right|_{\Omega=0} = \bar{\psi}\left\{-\kappa y\left((-\gamma_4)U_x(n)\delta_{n+a\hat{x},m} + (+\gamma_4)U_{-x}(n)\delta_{n-a\hat{x},m}\right)\right.\\
&\left.+\kappa x\left((-\gamma_4)U_y(n)\delta_{n+a\hat{y},m} + (+\gamma_4)U_{-y}(n)\delta_{n-a\hat{y},m}\right)\right\}\psi
\end{aligned} \tag{298}$$

and

$$\begin{aligned}
&\langle L'_F\rangle = \langle\left.\frac{\partial\mathcal{L}}{\partial\Omega}\right|_{\Omega=0}\rangle = -a^{-4}\mathrm{tr}\left[-\kappa y\left((-\gamma_4)U_x(n)\delta_{n+a\hat{x},m} + (+\gamma_4)U_{-x}(n)\delta_{n-a\hat{x},m}\right)\right.\\
&\left.+\kappa x\left((-\gamma_4)U_y(n)\delta_{n+a\hat{y},m} + (+\gamma_4)U_{-y}(n)\delta_{n-a\hat{y},m}\right)\hat{D}_W\right]\\
&= \kappa a^{-3}\mathrm{tr}_{c,s}\left[\gamma^4\left(\hat{x}\left(U_y(n)\delta_{n+ay,m} - U_{-y}(n)\delta_{n-ay,m}\right)\right.\right.\\
&\left.\left.-\hat{y}\left(U_x(n)\delta_{n+ax,m} - U_{-x}(n)\delta_{n-ax,m}\right)\right)\hat{D}_W^{-1}(n)\right]
\end{aligned} \tag{299}$$

Again, $i\langle L'_F\rangle$ is the z component of L_F in Ji decomposition [24].

Note that κ entered because of $\hat{D}_W^{-1}$. Later we will calculate D_{pure} acting on A_{phys}, where κ shall not show up.

Similarly, considering (here $\gamma_{1,2}$ have already been Wick rotated)

$$\begin{aligned}
&\mathbf{S}_F = \tilde{\psi}^\dagger \frac{1}{2}\mathbf{\Sigma}\tilde{\psi} \\
&\Sigma_z \equiv \begin{pmatrix} \sigma_z & 0 \\ 0 & \sigma_z \end{pmatrix} = \sigma_{12}^E \equiv \frac{i}{2}\left(\gamma_2^E\gamma_1^E - \gamma_1^E\gamma_2^E\right) \\
&S_{F_z} = \psi^\dagger \frac{1}{2}\sigma_{12}^E\psi = \bar{\psi}\gamma_4\frac{1}{2}\sigma_{12}^E\psi
\end{aligned} \tag{300}$$

therefor

$$\begin{aligned}
&\langle S_{F_z}\rangle = -a^{-4}\mathrm{tr}_{c,s}\left[\gamma_4\frac{1}{2}\sigma_{12}^E\hat{D}^{-1}(n)\right] = -a^{-4}\frac{1}{m+\frac{4}{a}}\mathrm{tr}_{c,s}\left[\gamma_0\frac{1}{2}\sigma^{12}\hat{D}_W^{-1}(n)\right] = -a^{-3}\kappa\mathrm{tr}_{c,s}\left[\gamma_4\sigma^{12}\hat{D}_W^{-1}(n)\right] \\
&\langle S_{F_z}\rangle = ia^{-3}\kappa\mathrm{tr}_{c,s}\left[\gamma_4 i\sigma^{12}\hat{D}_W^{-1}(n)\right]
\end{aligned} \tag{301}$$

On the other hand

$$\begin{aligned}
&\langle \frac{\partial\mathcal{L}}{\partial\Omega}\bigg|_{\Omega=0}\rangle = \langle\bar{\psi}\left(-i\kappa\gamma_4 a\sigma^{12}\right)\psi\rangle \\
&= -a^{-4}\mathrm{tr}\left[-i\kappa\gamma_4 a\sigma^{12}\hat{D}_W^{-1}\right] = a^{-3}\mathrm{tr}\left[i\kappa\gamma_4\sigma^{12}\hat{D}_W^{-1}\right]
\end{aligned} \tag{302}$$

Note that, finally, this term is discretized as an imaginary chiral potential such that

$$\begin{aligned}
&\langle \frac{\partial\mathcal{L}}{\partial\Omega}\bigg|_{\Omega=0}\rangle = \langle\bar{\psi}\left(-\kappa\frac{ia\sigma^{12}}{2}\left((\gamma_4-1)U_\tau(n)\delta_{n,n+t} + (\gamma_4+1)U_{-\tau}(n)\delta_{n-t,n}\right)\right)\psi\rangle \\
&= a^{-3}\mathrm{tr}\left[\frac{i\kappa\sigma^{12}}{2}\left((\gamma_4-1)U_\tau(n)\delta_{n,n+t} + (\gamma_4+1)U_{-\tau}(n)\delta_{n-t,n}\right)\hat{D}_W^{-1}\right]
\end{aligned} \tag{303}$$

We can do it in Coulomb gauge, since in Coulomb gauge, $\mathbf{A} = \mathbf{A}_{phys}$, and $\mathbf{A}_{phys}$ is unchanged under gauge transformation.

Using the fact that

$$\begin{aligned}
&E_{0i} = F_{0i} = g_{YM}\sum_a T^a F_{0i}^a, \quad \mathbf{A} = g_{YM}\sum_a T^a\mathbf{A}^a, \quad \mathrm{tr}[T^aT^b] = \frac{1}{2}\delta_{ab} \\
&\sum_a \mathbf{E}^a\times\mathbf{A}^a = \frac{2}{g_{YM}^2}\mathrm{tr}[\mathbf{F_{0i}}\times\mathbf{A}] \\
&\left(\sum_a \mathbf{E}^a\times\mathbf{A}^a\right)_z = \frac{2}{g_{YM}^2}\mathrm{tr}[F_{0x}A_y - F_{0y}A_x]
\end{aligned} \tag{304}$$

with

$$\begin{aligned}
&\{M\}_{TA} = \frac{1}{2}\left(M - M^\dagger - \frac{1}{3}\mathrm{tr}(M - M^\dagger)\right). \\
&U_\mu(n) = \exp(iaA_\mu(n)) \\
&A_\mu(n) \approx a^{-1}\frac{1}{i}\{U_\mu(n)\}_{TA} \\
&U_{\mu,\nu}(n) \equiv U_\mu(n)U_\nu(n+a\mu)U_\mu^{-1}(n+a\nu)U_\nu^{-1}(n) = \exp(ia^2F_{\mu\nu}(n) + \mathcal{O}(a^3)) \\
&F_{\mu\nu}(n) \approx a^{-2}\frac{1}{i}\left(U_{\mu\nu}(n)\right)_{TA}
\end{aligned} \tag{305}$$

or we can use the Clover gauge field

$$F^{clover}_{\mu\nu}(n) = a^{-2}\frac{1}{4i}\left[U_{\mu,\nu}(n) + U_{\nu,-\mu}(n) + U_{-\mu,-\nu}(n) + U_{-\nu,\mu}(n)\right]_{TA} \tag{306}$$

so after Wick rotation $F_{0i} \to iF_{0i}$ it is

$$\left\langle \left(\sum_a \mathbf{E}^a \times \mathbf{A}^a\right)_z \right\rangle = ia^{-3}\frac{\beta}{N_c}\mathrm{tr}[F^{clover}_{0x}\hat{A}_y - F^{clover}_{0y}\hat{A}_x] \tag{307}$$

where $\hat{A} = aA$

Note that, for all angular momentums, the lattice version is $i\langle\ldots\rangle_{lat} = \langle\ldots\rangle$, therefor, our final form of angular momentum is

$$\begin{aligned}
&\left\langle \left(\sum_a \mathbf{E}^a \times \mathbf{A}^a\right)_z \right\rangle_{lat} = a^{-3}\frac{\beta}{N_c}\mathrm{tr}[F^{clover}_{4x}\hat{A}_y - F^{clover}_{4y}\hat{A}_x] \\
&= -a^{-3}\frac{\beta}{N_c}\mathrm{tr}[\{U^{clover}_{4x}\}_{TA}\{U_{phys,y}\}_{TA} - \{U^{clover}_{4y}\}_{TA}\{U_{phys,x}\}_{TA}]
\end{aligned} \tag{308}$$

The gauge invariant D_{pure} for fermion can be obtained by

$$(\partial_\mu + iA_\mu)\psi(n) = \frac{U_\mu(n)\psi(n+a\mu) - U_{-\mu}(n)\psi(n-a\mu)}{2a} + \mathcal{O}(a) \tag{309}$$

so

$$(\partial_\mu + i(A_\mu - A_{phys,\mu})\psi(n) = \frac{U_\mu(n)\psi(n+a\mu) - U_{-\mu}(n)\psi(n-a\mu)}{2a} - \frac{2iaA_{phys,\mu}(n)}{2a}\psi(n) + \mathcal{O}(a) \tag{310}$$

Note that $2iaA_{phys}$ is just $2\{U_{phys}\}_{TA}$.

Another momentum angular is

$$\sum_{a,i} E^{ai}(\mathbf{x} \times \mathbf{D}_{pure})A^{ai} \tag{311}$$

with $D_{pure} = \partial_\mu + ig[\tilde{A}_{pure,\mu}, \cdot]$ (in our definition, $A = g\tilde{A}$)

After Wick rotation, it is

$$i\sum_{a,i} F^a_{0i}(\mathbf{x}\times\mathbf{D}_{pure})A^{ai} = i\frac{2}{g^2_{YM}}\sum_i \mathrm{tr}\left[F_{0i}(\mathbf{x}\times\mathbf{D}_{pure})A^i\right] \tag{312}$$

Note that

$$\begin{aligned}
&U_\mu(n)A_{phys,\nu}(n+\mu)U^\dagger_\mu(n) - U^\dagger_\mu(n-\mu)A_{phys,\nu}(n-\mu)U_\mu(n-\mu)\\
&= \left(1 + iaA_\mu(n) + \mathcal{O}(a^2)\right)A_{phys,\nu}(n+\mu)\left(1 - iaA_\mu(n) + \mathcal{O}(a^2)\right)\\
&- \left(1 - iaA_\mu(n-\mu) + \mathcal{O}(a^2)\right)A_{phys,\nu}(n-\mu)\left(1 - iaA_\mu(n-\mu) + \mathcal{O}(a^2)\right)\\
&= A_{phys,\nu}(n+\mu) - A_{phys,\nu}(n-\mu)\\
&+ ia\left[A_\mu(n), A_{phys,\nu}(n+\mu)\right] + ia[A_\mu(n-\mu), A_{phys,\nu}(n-\mu)] + \mathcal{O}(a^2)\\
&= A_{phys,\nu}(n+\mu) - A_{phys,\nu}(n-\mu) + 2ia\left[A_\mu(n), A_{phys,\nu}(n)\right] + \mathcal{O}(a^2)
\end{aligned} \tag{313}$$

in the last step, $A(n+\mu) \approx A(n) + \mathcal{O}(a)$ is used. Then

$$\begin{aligned}
&U_\mu(n)A_{phys,\nu}(n+\mu)U^\dagger_\mu(n) - U^\dagger_\mu(n-\mu)A_{phys,\nu}(n-\mu)U_\mu(n-\mu)\\
&= 2a\left(\frac{A_{phys,\nu}(n+\mu) - A_{phys,\nu}(n-\mu)}{2a} + i\left[A_\mu(n), A_{phys,\nu}(n)\right]\right) + \mathcal{O}(a^2)\\
&\approx 2a\left(\partial_\mu A_{phys,\nu} + i\left[A_{pure,\mu}, A_{phys,\nu}\right]\right) = 2aD_{pure,\mu}A_{phys,\nu}
\end{aligned} \tag{314}$$

So

$$\begin{aligned}
&D_{pure,\mu}A_{phys,\nu} = \frac{1}{2a}\left(U_\mu(n)A_{phys,\nu}(n+\mu)U^\dagger_\mu(n) - U^\dagger_\mu(n-\mu)A_{phys,\nu}(n-\mu)U_\mu(n-\mu)\right)\\
&= \frac{1}{2ia^2}\left(U_\mu(n)\{U_{phys,\nu}\}_{TA}(n+\mu)U^\dagger_\mu(n) - U^\dagger_\mu(n-\mu)\{U_{phys,\nu}(n-\mu)\}_{TA}U_\mu(n-\mu)\right)\\
&\equiv \frac{1}{2ia^2}(DA)_{\mu\nu}
\end{aligned} \tag{315}$$

and

$$\begin{aligned}
&i\sum_{a,j} F^a_{0j}(\mathbf{x}\times\mathbf{D}_{pure})A^{aj} = i\sum_j \frac{1}{i}a^{-2}\frac{\beta}{N_C}\mathrm{tr}\left[\{U_{0j}\}_{TA}(\mathbf{x}\times\mathbf{D}_{pure})A^{aj}\right]\\
&= -i\sum_j \frac{1}{2}a^{-4}\frac{\beta}{N_C}\mathrm{tr}\left[\{U_{0j}\}_{TA}\left(x(DA)_{yj} - y(DA)_{xj}\right)\right]\\
&= -i\frac{1}{2}\frac{\beta}{N_C}a^{-3}\sum_j \mathrm{tr}\left[\{U_{4j}\}_{TA}\left(\hat{x}(DA)_{1j} - \hat{y}(DA)_{0j}\right)\right]
\end{aligned} \tag{316}$$

8.1.8 The Current density and Charge density

In the case of exponential σ^{12} term, it is interesting to also measure

$$J_{12} = -i\kappa\langle\bar{\psi}\gamma_4\sigma^{12}\psi\rangle \tag{317}$$

Also the currents defined as

$$J_\mu = \langle\bar{\psi}\gamma_\mu\psi\rangle \tag{318}$$

is measured, such that

$$\begin{aligned} J_x &= \langle\bar{\psi}(\gamma_1 + y\Omega\gamma_4)\psi\rangle \\ J_y &= \langle\bar{\psi}(\gamma_2 - x\Omega\gamma_4)\psi\rangle \\ J_z &= \langle\bar{\psi}\gamma_3\psi\rangle \\ J_\tau &= \langle\bar{\psi}\gamma_4\psi\rangle \end{aligned} \tag{319}$$

we also measure the

$$\begin{aligned} J_1 &= \langle\bar{\psi}\gamma_1\psi\rangle \\ J_2 &= \langle\bar{\psi}\gamma_2\psi\rangle \end{aligned} \tag{320}$$

and the chiral charge density

$$n_5 = a^3\langle\bar{\psi}\gamma_4\gamma_5\psi\rangle \tag{321}$$

8.1.9 The Topological Density

The topological charge is defined as (**This might has to be modified in the rotating frame!**)

$$\begin{aligned} Q &= \frac{1}{32\pi^2}a^4\sum_n \epsilon_{\mu\nu\rho\sigma}\mathrm{tr}\left[C_{\mu\nu}(n)C_{\rho\sigma}(n)\right] \\ C_{\mu\nu}(n) &= \mathrm{Im}\left[U_{\mu\nu}(n)\right] \end{aligned} \tag{322}$$

Another definition is

$$\begin{aligned} Q &= \frac{1}{32\pi^2}a^4\sum_n \epsilon_{\mu\nu\rho\sigma}\mathrm{tr}\left[C^{clover}_{\mu\nu}(n)C^{clover}_{\rho\sigma}(n)\right] \\ C^{clover}_{\mu\nu}(n) &= \frac{1}{4}\mathrm{Im}\left[U_{\mu,\nu}(n) + U_{\nu,-\mu}(n) + U_{-\mu,-\nu}(n) + U_{-\nu,\mu}(n)\right] \end{aligned} \tag{323}$$

Note for both $C_{\mu\nu}$ and $C_{\mu\nu}^{clover}$, one have $C_{\mu\nu} = -C_{\nu\mu}$, alone with $\epsilon_{\mu\nu\rho\sigma}$, it doubles the term. Therefor

$$\sum \epsilon_{\mu\nu\rho\sigma} \mathrm{tr}[C_{\mu\nu}(n) C_{\rho\sigma}(n)] = 8 \left(\mathrm{tr}[C_{12}(n) C_{34}(n)] - \mathrm{tr}[C_{13}(n) C_{24}(n)] + \mathrm{tr}[C_{14}(n) C_{23}(n)] \right) \tag{324}$$

8.1.10 The Polyakov loop

Polyakov loop is measured straight forwardly.

8.1.11 The Chiral Condensate

The Chiral condensate can be calculate by Grassman number integral

$$\langle \bar{u} u \rangle = \mathrm{tr}[D_u^{-1}] \tag{325}$$

We are using two degenerate fermions, so

$$\langle \bar{\psi} \psi \rangle = \mathrm{tr}[D^{-1}] = a^{-4} \frac{1}{m + \frac{4}{a}} \mathrm{tr}\left[\hat{D}^{-1}\right] = a^{-4} \times 2a\kappa \mathrm{tr}\left[\hat{D}^{-1}\right] = 2a^{-3}\kappa \mathrm{tr}\left[\hat{D}^{-1}\right] \tag{326}$$

8.2 Sample Producer

In HMC, the most time-consuming operation is $\left(DD^\dagger\right)^{-1}\phi$, which need to solve the Wilson-Dirac equation, a matrix equation $\mathbf{b} = A\mathbf{x}$, where $A = DD^\dagger$ is a matrix depending on the gauge field and acting on the pseudo-fermion field.

At the same time, applying machine learning algorithms to physics problems has gained more and more attentions. The machine learning algorithms has been applied to solve partial differential equations [25]. In Ref. [26], deep learning is applied to map between potential and energy bypassing the need to solve the Schrödinger equation, in other words, the Schrödinger equation is implicitly solved by the network. So, it is reasonable to ask whether the machine learning can also help to solve the Wilson-Dirac equation? For example, is it possible to train the network to output eigenvectors by inputting a gauge field, or even better output $\mathbf{x}$ by inputting a gauge field and a pseudo-fermion field $\mathbf{b}$?

8.3 Data Analyse

We write the data analyse code based on Ref. [27] in Mathematica.

8.3.1 What is autocorrelation

The problem to address is that, the configurations generated are not statistically independent. So one need to take the relations between configurations into account.

To consider the relation between two sets, correlation functions are used, assuming two sets $a_{\alpha,\beta}$, assume $a_{\alpha,\beta} - \bar{a}_{\alpha,\beta}$ is a normal distribution.

$$\langle (a_\alpha - \bar{a}_\alpha)(a_\beta - \bar{a}_\beta)\rangle = \frac{1}{N^2}\sum_{i,j}\Gamma_{\alpha\beta}(j-i) \tag{327}$$

and

$$C_{\alpha\beta} = \sum_{t=-\infty}^{\infty}\Gamma_{\alpha\beta}(t) \tag{328}$$

Note that, $C_{\alpha\alpha}(0) = N\langle\delta_\alpha^2\rangle$ is the standard error.

For one single observable, one can define a correlation of a set of itself with delayed Markov time as

$$\tau_\alpha = \frac{1}{2\Gamma_{\alpha\alpha}(0)}\sum_{t=-\infty}^{\infty}\Gamma_{\alpha\alpha}(t) \tag{329}$$

For a purely exponential behaviour, $\Gamma_{\alpha\beta}(t) \sim \exp(-|t|/\tau)$. Generally, we can estimate $2\tau_\alpha$ as an interval such that two configurations are effectively independent [27].

Here, $\Gamma_{\alpha\beta}(t)$ is **autocorrelation**.

8.3.2 How to calculate autocorrelation, and how to use it to obtain the interval

Considering, we have already obtained a set of measurements by using configurations generated with Markov chain $\{a_\alpha^{i,r}\}$, where α indicating different observables, $r = 1 \to R$ indicating different replicas (usually, different replicas are obtained by running multi-times starting from same parameters, or running parallelly starting from same parameters), and $i = 1 \to N_r$ is index of each value in the replica.

Assume we measured $a_\alpha^{i,r}$, and want to obtain $F = f(a_\alpha)$.

In Ref. [27], a biased estimator is used such that

$$
\begin{aligned}
N &= \sum_r^R N_r, \\
\bar{a}_\alpha^r &= \frac{1}{N_r}\sum_i a_\alpha^{i,r} \\
\bar{\bar{a}}_\alpha &= \frac{1}{N}\sum_r^R N_r \bar{a}_\alpha^r \\
\bar{F} &= \frac{1}{N}\sum_r^R N_r f(\bar{a}_\alpha^r), \\
\bar{\bar{F}} &= f(\bar{\bar{a}}_\alpha)
\end{aligned}
\tag{330}
$$

and

$$
F_{mean} = \begin{cases} \bar{\bar{F}}, & R = 1 \\ \frac{R\bar{\bar{F}} - \bar{F}}{R-1} & R \geq 2 \end{cases}
\tag{331}
$$

The error is related to a correlation

$$
\bar{\bar{\Gamma}}_{\alpha\beta}(t) = \frac{1}{N - Rt}\sum_{r=1}^{R}\sum_{i=1}^{N_r - t}\left(a_\alpha^{i,r} - \bar{\bar{a}}_\alpha\right)\left(a_\beta^{i+t,r} - \bar{\bar{a}}_\beta\right)
\tag{332}
$$

at first we need to project it onto single variable, for this purpose, we need to calculate gradient as

$$
\begin{aligned}
h_\alpha &= \sqrt{\frac{\Gamma_{\alpha\alpha}(0)}{N}}, \\
\bar{\bar{f}}_\alpha &\approx \frac{1}{2h_\alpha}\left(f(\bar{\bar{a}}_1, \bar{\bar{a}}_2, \ldots, \bar{\bar{a}}_\alpha + h_\alpha, \ldots) - f(\bar{\bar{a}}_1, \bar{\bar{a}}_2, \ldots, \bar{\bar{a}}_\alpha - h_\alpha, \ldots)\right)
\end{aligned}
\tag{333}
$$

and

$$\bar{\bar{\Gamma}}_F(t) = \sum_{\alpha\beta} \bar{\bar{f}}_\alpha \bar{\bar{f}}_\beta \bar{\bar{\Gamma}}_{\alpha\beta}(t) \tag{334}$$

then the sum from $t = -\infty \to \infty$ is approximated as

$$\bar{\bar{C}}_F(W) = \bar{\bar{\Gamma}}_F(0) + 2\sum_{t=1}^{W} \bar{\bar{\Gamma}}_F(t) \tag{335}$$

By definition, if window W is known, then

$$\bar{\bar{\tau}}_{int}(W) = \frac{\bar{\bar{C}}_F(W)}{2\bar{\bar{C}}_F(0)} \tag{336}$$

To get τ one need to use a factor S to fit the exponential, assuming

$$2\bar{\bar{\tau}}_{int}(W) = \sum_{t=-\infty}^{\infty} \exp\left(-\frac{S|t|}{\bar{\bar{\tau}}(W)}\right) \tag{337}$$

with S as a constant usually $S = 1 \to 2$.

Then one can let $\bar{\bar{\tau}}(W) = S\tau_{int}(W)$, and calculate

$$g(W) = \exp\left(-\frac{W}{\bar{\bar{\tau}}(W)}\right) - \frac{\bar{\bar{\tau}}(W)}{\sqrt{WN}} \tag{338}$$

and fix W as the first index such that $g(W) < 0$ change sign.

Once W is obtained, one can calculate $2\bar{\bar{\tau}}_{int}(W)$ which is the Markov time separation such that two configurations can be considered as independent. Also, the error estimate is

$$\delta_F^2 = \frac{\bar{\bar{C}}(W)}{N} \tag{339}$$

索引

APE smearing, 64
APE stout, 64
autocorrelation, 101, 102

backward substitution, 29, 42
BiCGStab, 14

Cabibbo-Marinari trick, 55
chiral condensate, 65
Cornell gauge fixing, 53
correlator, 58

deflation, 34, 35
Double shifted QR iteration, 40

equilibrium, 17
Even-odd preconditioner, 46
exceptional configurations, 64

fat index, 5
FFT, 52
force, 9
force-gradient integrator, 18
forward substitution, 44
Francis QR iteration, 40

gauge smearing, 64
gauge smoothing, 64
GCR, 30
GCRO-DR, 33, 35, 44
generalized eigen-value problem, 37, 43
GEV, 37
Givens rotation, 28
GMRES, 26
GMRES-MDR, 33, 46

harmonic Ritz eigen vector, 36
Hessenberg matrix, 37
hmc, 7
Householder reflection, 37

Implicit shifted QR iteration, 40
Integrator, 10
Inverse power iteration, 41

Ji decomposition, 94

Kogut-Susskind fermion, 20
Krylov subspace, 26

Lanczos process, 32
Landau Gauge, 53
Langevin equation, 8
leap frog, 16
link index, 5
Los Alamos gauge fixing, 54
low mode, 34, 64

meson, 58
Metropolis, 17
molecular dynamics, 8
multi shift solver, 24
multi-rate integrator, 19
multi-shift solver, 48

nested integrator, 19

Omelyan, 17

plaquette energy, 58
point source, 60, 61
preconditioner, 34
pseudofermions, 7, 61

QR factorization, 38

rational approximation, 23
reciprocal space, 61
Remes algorithm, 23
REV, 36
RHMC, 23
Ritz eigen-vector, 36

Shifted QR iteration, 39
singular value decomposition, 36
site index, 5
solver, 14, 26
source, 60
staggered fermion, 20
staple, 10, 64
stochastic methods, 65
stout, 64
SVD, 36

TFQMR, 32

参考文献

[1] Michael Günther Francesco Knechtli and Michael Peardon. Lattice Quantum Chromodynamics Practical Essentials. 2017.

[2] C. Gattringer and C.B. Lang. Quantum Chromodynamics on the Lattice. 2010.

[3] Rajan Gupta. Introduction to Lattice QCD. 1998, arXiv:hep-lat/9807028.

[4] Alexander Altland and Ben Simons. Condensed Matter Field Theory 2nd edition. 2010.

[5] D. H. Weingarten and D. N. Petcher. Monte Carlo integration for lattice gauge theories with fermions. *Phys. Lett. B*, 99(4):333 - 338, 1981.

[6] Martin Lüscher. Computational Strategies in Lattice QCD. 2009, arXiv:1002.4232.

[7] S. Ueda et. al. Development of an object oriented lattice QCD code "Bridge++" on accelerators. *Journal of Physics: Conference Series*, 523:012046, 2014.

[8] Martin Lüscher. Schwarz-preconditioned HMC algorithm for two-flavor lattice QCD. *Computer Physics Communications*, 165:199–220, 2005.

[9] Yousef Saad. Iterative methods for sparse linear systems. 2003.

[10] P. J. Silva A. D. Kennedy, M. A. Clark. Force Gradient Integrators. *PoS LAT*, 2009:021, 2009, arXiv:0910.2950.

[11] Robert D. Mawhinney Hantao Yin. Improving DWF Simulations: the Force Gradient Integrator and the Möbius Accelerated DWF Solver. *PoS LAT*, 2011:051, 2011, arXiv:1111.5059.

[12] Dmitry Shcherbakov et. al. Adapted nested force-gradient integrators: the Schwinger model case. 2015, arXiv:1512.03812.

[13] Yukari Yamauchi Henry Lamm, Scott Lawrence. General Methods for Digital Quantum Simulation of Gauge Theories. *Phys. Rev. D*, 100:034518, 2019, arXiv:1903.08807.

[14] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. 1994, Available at: http://www.netlib.org/templates/Templates.html.

[15] Martin Lüscher. Computational Strategies in Lattice QCD. arXiv:arXiv:1002.4232.

[16] Hussam Al Daas et. al. Recycling Krylov subspaces and reducing deflation subspaces for solving sequence of linear systems. *RR-9206, Inria Paris*, 2018, Available at: https://hal.inria.fr/hal-01886546.

[17] Charles F. Van Loan Gene H. Golub. Matrix computations. 1996.

[18] Andreas Frommer and Uwe Glässner. Restarted GMRES for Shifted Linear Systems. *SIAM Journal on Scientific Computing*, 19:15–26, 1998.

[19] V. Simoncini. Restarted Full Orthogonalization Method for Shifted Linear Systems. *BIT Numerical Mathematics*, 43:459–466, 2003.

[20] B. Jegerlehner. Krylov space solvers for shifted linear systems. 1996, arXiv:hep-lat/9612014.

[21] C. T. H. Davies et. al. Fourier acceleration in lattice gauge theories. I. Landau gauge fixing. *Phys. Rev. D*, 37:1581, 1988.

[22] K. Schilling H. Suman. A Comperative Study of Gauge Fixing Procedures on the Connection Machines CM2 and CM5. 1993, arXiv:hep-lat/9306018.

[23] Y. Hirono A. Yamamoto. Lattice QCD in rotating frames. *Phys. Rev. Lett.*, 111:081601, 2013.

[24] Masashi Wakamatsu. Is gauge-invariant complete decomposition of the nucleon spin possible? *International Journal of Modern Physics A*, 29:1430012, 2014.

[25] Weinan E Jiequn Han, Arnulf Jentzen. Solving high-dimensional partial differential equations using deep learning. *PNAS*, 115(34):8505–8510, 2018.

[26] Isaac Tamblyn Kyle Mills, Michael Spanner. Deep learning and the Schrödinger equation. *Phys. Rev. A*, 96:042113, 2017, arXiv:1702.01361.

[27] Ulli Wolff. Monte Carlo errors with less errors. *Comput. Phys. Commun.*, 156:143–153, 2004, arXiv:1702.01361.