Cuda Lattice Gauge Document

目录

1 Data

1.1 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 $dirCount$,

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

2 Update scheme

2.1 hmc

HMC is short for hybrid Monte Carlo.

2.1.1   Basic idea

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.

(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.

- Force

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

$$
\begin{aligned}
F_\mu(x) &= \dot{P}_\mu(x) = -\frac{\beta}{2N}\{U_\mu(x)\Sigma_\mu(x)\}_{TA} \\
\{W\}_{TA} &= \frac{W - W^\dagger}{2} - \text{tr}\left(\frac{W - W^\dagger}{2N}\right)\mathbf{I}
\end{aligned}
\tag{4}
$$

where $\mathbf{I}$ is identity matrix, Σ is the 'Staple'.

- Integrator

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

$$
U(\tau + d\tau) \approx \dot{U}d\tau + U(\tau), \quad P(\tau + d\tau) \approx \dot{P}d\tau + P(\tau)
\tag{5}
$$

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],

$$
\epsilon = \tfrac{\tau}{M}
\tag{6a}
$$

$$
U_\mu(x, (n+1)\epsilon) = U_\mu(x, n\epsilon) + \epsilon P_\mu(x, n\epsilon) + \tfrac{1}{2}F_\mu(x, n\epsilon)\epsilon^2
\tag{6b}
$$

$$
P_\mu(x, (n+1)\epsilon) = P_\mu(x, n\epsilon) + \tfrac{1}{2}\left(F_\mu(x, (n+1)\epsilon) + F_\mu(x, n\epsilon)\right)\epsilon
\tag{6c}
$$

So, knowing $U(n\epsilon)$ we can calculate $F(n\epsilon)$ using Eq. (4). Knowing $U(n\epsilon), P(n\epsilon), F(n\epsilon)$, we can calculate $U((n+1)\epsilon)$ using Eq. (6).b. Then we are able to calculate $F((n+1)\epsilon)$ again using Eq. (4). Then we can calculate $P((n+1)\epsilon)$ using Eq. (6).c.

2.1.2   Leap frog integrator

In Sec. 2.1.1, the basic idea is introduced. However, the implementation is slightly different.

$$U_\mu(0, x) = gauge(x), \quad P_\mu(0, x) = i \sum_a r_a(\mu, x) T_a \tag{7a}$$

$$F_\mu(n\epsilon, x) = -\frac{\beta}{2N} \{U_\mu(n\epsilon, x)\Sigma_\mu(n\epsilon, x)\}_{TA} \tag{7b}$$

$$P_\mu(\tfrac{1}{2}\epsilon, x) = P_\mu(0, x) + \tfrac{\epsilon}{2} F_\mu(0, x) \tag{7c}$$

$$U_\mu((n+1)\epsilon, x) = \exp\left(\epsilon P_\mu((n+\tfrac{1}{2})\epsilon, x)\right) U_\mu(n\epsilon, x) \tag{7d}$$

$$P_\mu((n+\tfrac{1}{2})\epsilon, x) = P_\mu((n-\tfrac{1}{2})\epsilon, x) + \epsilon F_\mu(n\epsilon, x) \tag{7e}$$

or simply written as

$$P_\epsilon \circ U_\epsilon \circ P_{\frac{1}{2}\epsilon}(P_0, U_0) \tag{8}$$

The pseudo code can be written as

```
1
2    FieldGauge field = gaugeField.copy();
3
4    //sum _i i r_i T_i, where r_i are random numbers generated by Gaussian distribution
5    FieldGauge momentumField = FieldGauge::RandomGenerator();
6
7    //First half update
8    FieldGauge forceField = FieldGauge::Zero();
9    for (int i = 0; i < m_lstActions.Num(); ++i)
10   {
11       forceField += m_lstActions[i]->CalculateForceOnGauge(field);
12   }
13   //momentumField = momentumField + 0.5f * epsilon * forceField
14   momentumField.Axpy(fStep * 0.5f, forceField);
15
16   for (int i = 1; i < steps + 1; ++i)
17   {
18       field = FieldGauge::Exp(fStep * momentumField) * field;
19       forceField = FieldGauge::Zero();
20       for (int j = 0; j < m_lstActions.Num(); ++j)
21       {
22           forceField += m_lstActions[j]->CalculateForceOnGauge(field);
23       }
24       momentumField.Axpy((j < steps) ? fStep : (fStep * 0.5f), forceField);
25   }
```

2.1.3   Omelyan integrator

The Omelyan integrator can be simply written as

$$P_{\lambda\epsilon} \circ U_{\frac{1}{2}\epsilon} \circ P_{(1-2\lambda)\epsilon} \circ U_{\frac{1}{2}\epsilon} \circ P_{\lambda\epsilon} \left(P_0, U_0\right) \tag{9}$$

with

$$\lambda = \frac{1}{2} - \frac{\left(2\sqrt{326} + 36\right)^{\frac{1}{3}}}{12} + \frac{1}{6\left(2\sqrt{326} + 36\right)^{\frac{1}{3}}} \approx 0.19318332750378364 \tag{10}$$

索引

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.