IPCC 2021 赛题介绍

格点量子色动力学
稀疏线性系统求解简介

CNIC-LATTICE

klzhang@cnic.cn

中科院计算机网络信息中心

October 25, 2021

目录

1 格点量子色动力学简介 **2**

1.1 简介 2

2 数学形式 **4**

2.1 求解问题 4

2.2 数学公式 5

2.3 算法 7

3 优化方案，仅供参考 **10**

3.1 CheckerBoarding 10

3.2 Spin Project Trick 11

Chapter 1

格点量子色动力学简介

1.1 简介

目前已知，自然界中存在四种基本相互作用力，分别为引力相互作用、电磁相互作用、弱相互作用和强相互作用。在粒子物理的标准模型中，描述强相互作用的基本理论是**量子色动力学（Quantum Chromodynamics, QCD）**（基于杨振宁先生的Yang-Mills理论）。**格点量子色动力学（Lattice, LQCD, 格点QCD）**既是研究基本粒子夸克和胶子如何结合成质子、中子乃至各种原子核的基本理论，也是目前最为可靠的从第一性原理出发的非微扰研究方法。也是理解可见物质绝大部分质量起源，解释高能物理实验结果必不可少的研究手段。LQCD的本质，是将七大千禧年数学难题之一的“质量缺口问题”也就是可见物质质量起源问题转化为在四维时空格子中千万时空点上的量子态叠加问题。

具体而言，LQCD将连续四维时空离散化，在四维超立方格子的千万时空点和内部自由度空间，基于马尔科夫链蒙特卡洛模拟方法，利用动力学演化，生成样本数据（简称**组态**）；而后对基于组态数据进行物理问题的计算和统计测量，得到物理结果。因此，LQCD研究相当于在超级计算机上进行模拟粒子物理实验。

LQCD的主要计算单元，是循环边界条件的四维超立方格子中的一个稀疏线

性系统。该线性系统表现为四维八点的stencil运算，即在任意点上，是该点以及与之相邻的8个点到该点的9个12x12稠密复数矩阵（该矩阵$U_\mu(x)$是一个SU(3)矩阵，从LQCD组态中读取，最小可压缩为8个实数）映射的组合。上述稀疏线性系统的矩阵结构完全由LQCD理论决定。

同时，LQCD计算的一个核心模块是对上述稀疏线性系统的求解，是一个场景非常明确的类HPCG计算问题。一个完整计算过程需要反复调用该模块，对计算资源需求巨大。因此，一个高效的稀疏矩阵操作是重要且必要的。同时，需要依托其他预处理算法充分利用内存空间降低迭代次数，然后组合线性系统操作、矢量代数变换和全局归约操作并反复迭代。

因此，LQCD计算最主要的热点操作归结为**矩阵乘向量**$M \cdot \chi$，实现中习惯上称之为**Dslash**, 其具体形式见下一章节(chapter.2)。

伴随着国内超算的飞速发展，国内LQCD应用取得了长足的发展，同时面临着机遇与挑战。为适应国内不同类型架构的超算平台，LQCD应用软件的开发、移植与优化工作正在积极开展。目前，已经在多个不同架构超算平台的实现初步开发、移植优化工作。 在国际上，LQCD已经被美国列为E级超级计算机的主要应用，自2016年以来每年投入约10%的超算资源（基于ASCR/HEP Exascale Requirements Review Report[1]，2016年约十亿核小时，预计2025年超过千亿核小时)。

[1] arXiv:1603.09303

Chapter 2

数学形式

2.1 求解问题

大规模稀疏线性系统求解问题

$$M\chi = b \tag{2.1}$$

其中，M 是固定形式的稀疏矩阵，即该矩阵什么位置有非零值是确定的；χ 是待求解向量；b是给出的已知向量，相当于限制条件。

该线性系统规模巨大，难以精确求解，通常使用数值迭代方法（例如共轭梯度算法）求解。 在该问题中，并不能保证M是正定的矩阵，但可以确保$M^{\dagger}M$是正定的。因此，上述式(2.1)可以转化为如下求解问题。

$$M^{\dagger}M\chi = M^{\dagger}b \tag{2.2}$$

不论何种求解方法，通常在迭代求解过程中会反复调用 **矩阵乘向量操作**(Dslash),$M\chi$ 或 $M^{\dagger}M\chi$，因此一个高效的矩阵乘向量操作至关重要。

2.2 数学公式

矩阵M称为费米子矩阵，它的形式根据LQCD理论由四维协变差分改进而来。Wilson 费米子作用量是LQCD中一种较为基础的理论形式。

$$M = (\sum_{\mu} \gamma_\mu D_\mu + m) \tag{2.3}$$

这里的m 代表夸克质量，计算中可视为外部输入参数。Wilson 费米子矩阵M依赖于外部输入的组态数据$U_\mu(x)$,其具体形式如下：

$$M = (4+m)\delta_{x,y} - \frac{1}{2}\sum_{\mu=0}^{4}[(1-\gamma_\mu)U_\mu(x)\delta_{x+\mu,y} + (1+\gamma_\mu)U_\mu(x)\delta_{x-\mu,y}] \tag{2.4}$$

其厄米共轭$M^\dagger$的形式如下：

$$M^\dagger = (4+m)\delta_{x,y} - \frac{1}{2}\sum_{\mu=0}^{4}[(1+\gamma_\mu)U_\mu(x)\delta_{x+\mu,y} + (1-\gamma_\mu)U_\mu(x)\delta_{x-\mu,y}] \tag{2.5}$$

相应的矩阵乘向量操作 $\phi\& = M\chi$

$$\begin{aligned}
\phi_x =& M_{x;y}\chi_y \\
=&(4+m)\delta_{x,y}\chi(y) - \frac{1}{2}\sum_{\mu=0}^{4}\left[[(1-\gamma_\mu)\otimes U_\mu(x)]\delta_{x+\mu,y} + [(1+\gamma_\mu)\otimes U_\mu^\dagger(x-\mu)]\delta_{x-\mu,y}\right]\underline{\chi(y)} \\
=&(4+m)\chi_x - \frac{1}{2}\sum_{\mu=0}^{4}\left[[(1-\gamma_\mu)\otimes U_\mu(x)]\underline{\chi(x+\mu)} + [(1+\gamma_\mu)\otimes U_\mu^\dagger(x)]\underline{\chi(x-\mu)}\right]
\end{aligned} \tag{2.6}$$

其中(以$\{L_x, L_y, L_z, L_t\} = \{8,8,8,16\}$为例):

- x, y 表示四维时空坐标，$x = \{ix, iy, iz, it\}$, $ix = 0, 1, 2, ..., Lx-1$
- $U_\mu(x)$ 是格点规范场（lattice gauge field), 从组态数据文件中读取（或

程序产生）；μ 表示4个时空方向；即对于任意一个时空点x，数据包含4个3×3的复矩阵———4个SU(3)矩阵，内部指标称为色（color）指标。所以一个组态数据的大小为：$Size = L_x * L_y * L_z * L_y * 4 * 3 * 3 * sizeof(complex)$ 字节.

- $\phi(x), \chi(x)$是格点费米子场，对任意时空点x，数据包含(4×3)维的复向量，每个时空点上有12个复数。其中，3表示3个色指标（0，1，2）；4表示Dirac指标（0，1，2，3），注意该指标不是时空坐标。所以，可以理解为每个时空点包含4个3维向量（或3个4维向量）. 因此，在实现中，一个格点费米子场得数据量为：$Size = L_x * L_y * L_z * L_t * 4 * 3 * sizeof(complex)$ 字节

- γ_μ 是 4×4维的 Gamma矩阵，μ 表示4个时空方向，所以γ矩阵有且只有4个，每个矩阵结构和数值简单且固定，每个矩阵都只包含了4个非零复数,如式(2.7)所示.

- 张量乘法: 我们以 $\mu = 0$(x方向)的红色部分$\gamma_\mu \otimes U_\mu(x)$ 为例进行说明（下式中，将表示方向的μ用上标表示）：

$$\gamma^x \otimes U^x(x,y,z,t)\chi(x+\mu) = [\gamma^x_{\alpha,\beta}] * [U^x_{a,b}] * [\chi(x+1,y,z,t)]_{\beta,b} = [\phi(x,y,z,t)_{\alpha,a}$$

在正确对应了时空坐标的情况下，上式中得张量乘法从数学可以理解为一个12维（复数）得矩阵乘向量得计算；在实际计算实现中，并没有存储该矩阵，原因是如上式中的张量乘法公式所示，该矩阵仅有一个3×3的SU(3)矩阵和一个结构简单的γ-矩阵((2.7))构成，所以在计算中根据数据构建出该计算操作更为划算。

$$
\begin{aligned}
\gamma_x &= \begin{bmatrix} 0 & 0 & 0 & i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ -i & 0 & 0 & 0 \end{bmatrix} \\
\gamma_y &= \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \\
\gamma_z &= \begin{bmatrix} 0 & 0 & i & 0 \\ 0 & 0 & 0 & -i \\ -i & 0 & 0 & 0 \\ 0 & i & 0 & 0 \end{bmatrix} \\
\gamma_t &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{2.7}
$$

2.3 算法

如前所述，通常使用数值迭代方法求解。在该问题中，虽然不能保证M是正定的矩阵，但可以确保 $M^\dagger M$是正定的。可以转化为如下求解问题。

$$
M^\dagger M\chi = M^\dagger b \tag{2.8}
$$

现有代码包中采用的是共轭梯度算法（CG）。

求解过程如下：

Algorithm 1: 格点QCD线性系统求解

Brief: 求解问题：$\boldsymbol{M}^{\dagger}\boldsymbol{M}\vec{x} = \boldsymbol{M}^{\dagger}\vec{b}$

Input: 规范场组态数据 (Lattice Gauge) $\boldsymbol{U_{\mu}(x,y,z,t)}$

Output: 费米子场 Lattice Fermion $\vec{x}$

`/* Initialization */`

`/*` 运行提供代码时，须保证进程数与子格子数目相等 `*/`

`/*` 总格子和子格子大小 `*/`

int site_vec[4]

int subgs[4]

读取组态数据 $\boldsymbol{U_{\mu}(x,y,z,t)}$

给定约束条件（make source） $\vec{b}$

$Rsd :=$ 目标精度

$\boldsymbol{M} := Dslash(U_{\mu})$

`/*` 方程两侧同乘以 $\boldsymbol{M\dagger}$ 保证正定 `*/`

$\boldsymbol{M'} := \boldsymbol{M}^{\dagger}\boldsymbol{M}$

$\vec{b'} := \boldsymbol{M}^{\dagger}\vec{b}$

`/*` 求解 $\boldsymbol{M'}\vec{x} = \vec{b'}$, `CG` `*/`

$\vec{x} :=$ 猜测解

for $n\_cg < MaxCG$ **do**

    计算$\vec{x'} = \boldsymbol{M'}\vec{x}$

    计算残差 residual

    if $residual < Rsd$ **then**

        Save $\vec{x}$

    else

        Update $\vec{x}$

Save $\vec{x}$

所采用的共轭梯度算法的伪代码如下：

Algorithm 2: 共轭梯度（CG）算法

Input: $\boldsymbol{M'}, \vec{b}, \vec{x}, i_{max}, \epsilon$

Output: $\vec{x}$

$i \coloneqq 0$

$\vec{r} \coloneqq \vec{b} - \boldsymbol{M'}\vec{x}$

$\vec{d} \coloneqq \vec{r}\ \delta_{new} \coloneqq \vec{r}^{\mathrm{T}}\vec{r}$

$\delta_0 \coloneqq \delta_{new}$

while $i < i_{max}$ **and** $\delta_{new} > \epsilon^2\delta_0$ **do**

  $\vec{q} \coloneqq \boldsymbol{M'}\vec{d}$

  $\alpha \coloneqq \frac{\delta_{new}}{\vec{d}^{\mathrm{T}}\vec{q}}$

  $\vec{x} \coloneqq \vec{x} + \alpha\vec{d}$

  if $i \mod 50 = 0$ **then**

    $\vec{r} \coloneqq \vec{b} - \boldsymbol{M'}\vec{x}$

  else

    $\vec{r} \coloneqq \vec{r} - \alpha\vec{q}$

  $\delta_{old} \coloneqq \delta_{new}$

  $\delta_{new} \coloneqq \vec{r}^{\mathrm{T}}\vec{r}$

  $\beta \coloneqq \frac{\delta_{new}}{\delta_{old}}$

  $\vec{d} \coloneqq \vec{r} + \beta\vec{d}$

  $i \coloneqq i + 1$

Chapter 3

优化方案，仅供参考

以下为格点计算中可能用到的优化方案，仅供参考，参赛队伍可自行设计优化方案。

3.1 CheckerBoarding

格点计算中常用的一种优化策略是Red/Black CheckerBoarding，即相当于根据时空坐标对费米子矩阵M和对应的向量（lattice_fermion）做奇偶预处理（Even-Odd Precondition），会分为四块，对角部分的M_{ee} 和 M_{oo}是实对角的

$$M = \begin{bmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{bmatrix}$$

以上部分代码中以实现，以下部分并未完成，仅为理论推导。

$$\begin{aligned} M &= \begin{bmatrix} 1 & 0 \\ M_{oe}M_{ee}^{-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} M_{ee} & 0 \\ 0 & M_{oo} - M_{oe}M_{ee}^{-1}M_{eo} \end{bmatrix} \cdot \begin{bmatrix} 1 & M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{bmatrix} \\ &= L\tilde{M}U \end{aligned}$$

线性求解问题

$$M\phi = \chi$$

转变为

$$L\tilde{M}U\phi = \chi$$

$$\tilde{M}\phi\prime = \chi\prime$$

$$\chi\prime = L^{-1}\chi \ , \phi = U^{-1}\phi\prime$$

其中，$\tilde{M} = M_{oo} - M_{oe}M_{ee}^{-1}M_{eo}$
于是，原求解问题可转化为以下过程求解:

1 source： $\chi \longrightarrow \chi\prime = L^{-1}\chi$

2 solve $\phi\prime$ by : $\tilde{M}\phi\prime = \chi\prime$

3 reconstruct prop: $\phi = U^{-1}\phi\prime$

3.2 Spin Project Trick

这个方案是依据γ-矩阵形式和性质设计，代码中已基本未实现。以下为理论形式推导。 以 Spin Projection 在 $\mu = 0$ 方向的投影

$$(1+\gamma_0)\begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} = (1+\begin{bmatrix} 0 & i\sigma^1 \\ -i\sigma^1 & 0 \end{bmatrix})\begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} + \begin{bmatrix} i\sigma^1 \begin{pmatrix} z \\ t \end{pmatrix} \\ -i\sigma^1 \begin{pmatrix} x \\ y \end{pmatrix} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + i\sigma^1 \begin{pmatrix} z \\ t \end{pmatrix} \\ \begin{pmatrix} z \\ t \end{pmatrix} - i\sigma^1 \begin{pmatrix} x \\ y \end{pmatrix} \end{bmatrix}$$

只需保留上半部分即可，下半部分可以通过上半部分左乘$-i\sigma^1$恢复

$$i\sigma^1 \times Half_{up} = -i\sigma^1[\begin{pmatrix} x \\ y \end{pmatrix} + i\sigma^1 \begin{pmatrix} z \\ t \end{pmatrix}] = [-i\sigma^1 \begin{pmatrix} x \\ y \end{pmatrix} + 1 \begin{pmatrix} z \\ t \end{pmatrix}] = [\begin{pmatrix} z \\ t \end{pmatrix} - i\sigma^1 \begin{pmatrix} x \\ y \end{pmatrix}] = H_{lower}$$

Spin Projection Trick

- Consider Use of Spin Projection in the Dslash in the 3-direction

$$(1+\gamma_3)U \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} = \begin{bmatrix} U(x+z) \\ U(y+t) \\ \hline U(x+z) \\ U(y+t) \end{bmatrix} \begin{matrix} \} \text{ upper half vector} \\ \} \text{ lower half vector} \end{matrix}$$

- Break down: projection – multiplication- reconstruction

$$\begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} \xrightarrow{SpinProject} \begin{bmatrix} x+z \\ y+t \end{bmatrix} \xrightarrow{Multiply\ U} \begin{bmatrix} U(x+z) \\ U(y+t) \end{bmatrix} \xrightarrow{SpinReconstruct} \begin{bmatrix} U(x+z) \\ U(y+t) \\ \hline U(x+z) \\ U(y+t) \end{bmatrix}$$

- Save two SU(3) multiplications per site
- Reduce communications needed by 2