

格子上の場の理論ハンズオン

part3 Gaugefields.jlについて

東京大学情報基盤センター学際情報科学研究部門

永井佑紀

Gaugefields.jlとは？

 **JuliaQCD**

 **LatticeQCD.jl**



QCDMeasurements.jl



LatticeDiracOperators.jl



Gaugefields.jl

Wilsonloop.jl

CLIME.jl

ゲージ場を取り扱うパッケージ

LatticeQCD.jl

内部ではGaugefields.jlを呼んで、
ゲージ場を取り扱っている

Gaugefields.jlをいじってみよう

ゲージ場

using Gaugefields

4次元の場合。

```
U = Initialize_Gaugefields(NC, Nwing, NX, NY, NZ, NT)
```

Nwingは”のりしろ”を使うかどうかであるが、通常はNwing=0が良い

```
U = Initialize_Gaugefields(NC, Nwing, NX, NY, NZ, NT, condition="hot",  
randomnumber="Reproducible")
```

condition = “cold”, “hot”などがある

NX, NYだけ与えると2次元のゲージ場になる

$U_{\mu}(n)$ Uは4成分配列：リンクの方向

プラケット

あるゲージ場のプラケット期待値

```
plaq_t = calculate_Plaquette(U, temp1, temp2) * factor
```

temp1とtemp2は

```
temp1 = similar(U[1])
```



```
temp2 = similar(U[1])
```

 計算用の一時配列

一時配列などを考えたくなければ、QCDDMeasurements.jlを使う

こちらは簡単に測定できることに特化している

```
m_plaq = Plaquette_measurement(U)
m_poly = Polyakov_measurement(U)
plaq = get_value(measure(m_plaq,U))
poly = get_value(measure(m_poly,U))
println("plaq: $plaq")
println("poly: $poly")
```


他のソフトとの互換性

```
filename = "testconf.txt"  
load_BridgeText!(filename,U,L,NC)
```

```
filename = "testconf.txt"  
save_textdata(U,filename)
```

Bridge++のテキストフォーマットを読んだり書いたり

```
filename = "hoge.ildg"  
ildg = ILDG(filename)  
i = 1  
L = [NX,NY,NZ,NT]  
load_gaugefield!(U,i,ildg,L,NC)
```

```
filename = "hoge.ildg"  
save_binarydata(U,filename)
```

色々なソフトで読み書きできる、ILDGフォーマットにも対応

作用の定義

作用は以下のように定義できる

```
gauge_action = GaugeAction(U)
```

まだ、作用の中身は定義されていない

プラケット作用を入れたい

```
plaqloop = make_loops_fromname("plaquette")
```

右向きプラケット

```
append!(plaqloop, plaqloop')
```

左向きプラケットを足す

```
 $\beta = \beta / 2$ 
```

```
push!(gauge_action,  $\beta$ , plaqloop)
```

よくある定義に合わせるために1/2が入る

長方形作用を入れたい

```
rectloop = make_loops_fromname("rectangular", Dim=Dim)
```

```
append!(rectloop, rectloop')
```

```
 $\beta_{inp} = \beta / 2$ 
```

```
push!(gauge_action,  $\beta_{inp}$ , rectloop)
```

さらに、

Wilsonloop.jlを使えば任意のclosed loop
を作用に足せる

生成子での展開

定義されたUは、4次元格子上の $NC \times NC$ 複素数行列

SU(NC)を厳密に保ちたいときは生成子を使った表現が便利

SU(2)ならパウリ行列、SU(3)ならゲルマン行列

`p = initialize_TA_Gaugefields(U)` “運動量”を扱うときに用いる

`gauss_distribution!(p)` ガウス分布させたい場合

`Sp = p * p / 2` 掛け算も定義されている (全時空で和をとっている)

`Traceless_antihermitian_add!(p[μ], factor, temp)`

作ったリンクをpに代入したりもできる

色々な計算

$$\exp(\epsilon \Delta \tau p_{\mu}(n)) U_{\mu}(n)$$

`expU!(expU, $\epsilon * \Delta \tau$, p[μ], [temp1, temp2])`
`mul!(W, expU, U[μ])`

mul!は積の計算

作用Sの値

`Sg = -evaluate_GaugeAction(gauge_action, U) / NC`

作用Sの微分 $\frac{\partial S}{\partial U_{\mu}(n)}$

`calc_dSdU μ !(dSdU μ , gauge_action, μ , U)`

値を代入

`substitute_U!(U, Uold)`

evaluate_GaugeActionで計算されるSには1/NCが入っていないことに注意

ループフロッグ

```
for itrj = 1:MDsteps
    U_update!(U, p, 0.5, Δτ, Dim, gauge_action)
    P_update!(U, p, 1.0, Δτ, Dim, gauge_action, temp1, temp2)
    U_update!(U, p, 0.5, Δτ, Dim, gauge_action)
end
```

```
function U_update!(U, p, ε, Δτ, Dim, gauge_action)
    temps = get_temporary_gaugefields(gauge_action)
    temp1 = temps[1]
    temp2 = temps[2]
    expU = temps[3]
    W = temps[4]
    for μ = 1:Dim
        exptU!(expU, ε * Δτ, p[μ], [temp1, temp2])
        mul!(W, expU, U[μ])
        substitute_U!(U[μ], W)
    end
end
```

$$U_{\mu}(n) \rightarrow \exp(\epsilon \Delta \tau p_{\mu}(n)) U_{\mu}(n)$$

```
function P_update!(U, p, ε, Δτ, Dim, gauge_action, temp1,
temp2) # p -> p + factor * U * dSdUμ
    NC = U[1].NC
    temp = temp1
    dSdUμ = temp2
    factor = -ε * Δτ / (NC)
    for μ = 1:Dim
        calc_dSdUμ!(dSdUμ, gauge_action, μ, U)
        mul!(temp, U[μ], dSdUμ) # U * dSdUμ
        Traceless_antihermitian_add!(p[μ], factor, temp)
    end
end
```

$$p_{\mu}(n) \rightarrow p_{\mu}(n) - \epsilon \Delta \tau \frac{1}{N_c} T_A \left[\frac{\partial S}{\partial U_{\mu}(n)} U_{\mu}(n) \right]$$

これだけで、MDの部分は終わり！