

# ▼ 1 Lattice QCD ハンズオン

## 1.1 LatticeQCD.jl で $\pi$ 中間子の質量を計算してみよう。

- 2024/08/31 富谷昭夫 [akio@yukawa.kyoto-u.ac.jp](mailto:akio@yukawa.kyoto-u.ac.jp) (<mailto:akio@yukawa.kyoto-u.ac.jp>)

このノートブックは、「[格子場の理論サマースクール2024](https://akio-tomiya.github.io/latticeschool2024/) (<https://akio-tomiya.github.io/latticeschool2024/>)」のハンズオンのために作成されました。以下にも明記してありますが、実行部分では簡単な虫食いになっており、そのままでは動作しません。読みながらうまく動作させてください。

---

Julia およびLatticeQCD.jlインストールの手順は、以下のとおりです。

- Step1. Juliaを Juliaup <https://github.com/JuliaLang/juliaup> (<https://github.com/JuliaLang/juliaup>) でいれる。
- Step2. JupyterNotebook を入れ、julia REPL からカーネルを add IJulia でいれる。
- Step3. LatticeQCD.jl を <https://github.com/akio-tomiya/LatticeQCD.jl> (<https://github.com/akio-tomiya/LatticeQCD.jl>) を参考に入れる (add LatticeQCD で入る)。

ここまで終わってれば次に進んでください。

---

Jupyternotebookの練習も兼ねて、次のセルを実行し、演習で必要なパッケージを入れておきましょう。セル内にカーソルを合わせて「Shiftキー」と「エンターキー」を押すと実行できるはずです。



```
In [1]: 1 import Pkg;
2 println("パッケージを追加します。")
3 Pkg.add("Plots")
4 Pkg.add("LaTeXStrings")
5 Pkg.add("Markdown")
6 Pkg.add("Dates")
7 Pkg.add("Printf")
8 println("最終行まで実行されました。全てインストール済なら「 No Changes to `~/hogehoge/Manifest.toml`」のように出ます。")
```

パッケージを追加します。

```
Resolving package versions...
No Changes to `~/julia/environments/v1.10/Project.toml`
No Changes to `~/julia/environments/v1.10/Manifest.toml`
Resolving package versions...
No Changes to `~/julia/environments/v1.10/Project.toml`
No Changes to `~/julia/environments/v1.10/Manifest.toml`
Resolving package versions...
No Changes to `~/julia/environments/v1.10/Project.toml`
No Changes to `~/julia/environments/v1.10/Manifest.toml`
Resolving package versions...
No Changes to `~/julia/environments/v1.10/Project.toml`
No Changes to `~/julia/environments/v1.10/Manifest.toml`
Resolving package versions...
No Changes to `~/julia/environments/v1.10/Project.toml`
No Changes to `~/julia/environments/v1.10/Manifest.toml`
```

最終行まで実行されました。全てインストール済なら「 No Changes to `~/hogehoge/Manifest.toml`」のように出ます。

次に画面を移り、JuliaのREPLで以下を実行してください。

```
run_wizard()
```

すると `my_parameters.toml` というパラメータファイルを得ることになります。パラメータファイルは、テキストエディタでの編集も、もちろん可能ですし、自分で書いても構いません。

- パラメータの作り方は、[ウィザードの使い方](https://www.dropbox.com/s/nlt3poou8gfd5li/wizardLQCD.pdf?dl=0) (<https://www.dropbox.com/s/nlt3poou8gfd5li/wizardLQCD.pdf?dl=0>)を参考に作ってください。
- ファイルを作るのが面倒な人は、[my\\_parameters.toml](https://www2.yukawa.kyoto-u.ac.jp/~akio.tomiya/filebox/school_handson/my_parameters.zip) ([https://www2.yukawa.kyoto-u.ac.jp/~akio.tomiya/filebox/school\\_handson/my\\_parameters.zip](https://www2.yukawa.kyoto-u.ac.jp/~akio.tomiya/filebox/school_handson/my_parameters.zip)) においておきました、こちらをダウンロードし、このノートブックと同じディレクトリに入れてもらえれば同じように動作します。

以下では、

```
["Physical setting"]
L = [4, 4, 4, 24]
update_method = "Heatbath"
Nsteps = 200
"β" = 5.7

["System Control"]
logfile = "Heatbath_L04040424_beta5.7_quenched.txt"
verboselevel = 2
measurement_dir = "Heatbath_L04040424_beta5.7_quenched"
measurement_basedir = "./measurements"
log_dir = "./logs"
hasgradientflow = false

["Measurement set".measurement_methods.Pion_correlator]
```

以下のセル以降、このJupyter notebook と同じ場所にパラメータファイルがあることを仮定して話を進めます。

## ▼ 2 格子QCDの計算

ここから格子QCDの計算に入ります。LatticeQCD.jl をロードします。

```
In [1]: 1 using LatticeQCD
```

以下のコードはこのままでは動作しません、虫食いになっています。XYZと書かれている部分は適切なコードを入れてください。

パラメータファイル my\_parameters.toml の中身を見えます。

```
In [2]: 1 println(read("my_parameters.toml", String))

["HMC related"]
"Δτ" = 0.05
MDsteps = 20

["Physical setting(fermions)"]
Dirac_operator = "nothing"

["Physical setting"]
useOR = true
L = [4, 4, 4, 24]
Nthermalization = 0
update_method = "Heatbath"
numOR = 3
Nsteps = 200
"β" = 5.7

["System Control"]
logfile = "Heatbath_L04040424_beta5.7_quenched.txt"
verboselevel = 2
measurement_dir = "Heatbath_L04040424_beta5.7_quenched"
measurement_basedir = "./measurements"
log_dir = "./logs"
hasgradientflow = false

["Measurement set".measurement_methods.Pion_correlator]
methodname = "Pion_correlator"

    ["Measurement set".measurement_methods.Pion_correlator.fermion_p
arameters]
    Dirac_operator = "Wilson"

["Measurement set".measurement_methods.Polyakov_loop]
methodname = "Polyakov_loop"

["Measurement set".measurement_methods.Plaquette]
methodname = "Plaquette"

[gradientflow_measurements]
hasgradientflow = false

    [gradientflow_measurements.measurements_for_flow]
```

---

以下から実際のコード実行です。

---

以下のセルはコードの実行が行われるため、時間がかかります。

```
In [4]: 1 # このセルを実行すると、実際にシミュレーションが走ります。
        2 # 時間がかかります！ (m1 mac book air で440秒=7分ちょっと)
        3 run_LQCD("my_parameters.toml") #パラメータファイルを指定して実行。
```

Dict{String, Any} with 2 entries:

```
"Δτ"      => 0.05
"MDsteps" => 20
```

```
inputfile: /Users/akio/Downloads/untitled folder/my_parameters.toml
[HMC related]
```

```
[Physical setting(fermions)]
```

```
[Physical setting]
```

```
[System Control]
```

```
[Measurement set]
[measurement_methods]
```

```
[gradientflow_measurements]
```

```
Heatbath will be used
# /Users/akio/Downloads/untitled folder
```

### 3 $\pi$ 中間子の相関関数を描画

ここでは、 $\pi$ 中間子の相関関数を描画してみます。実際には、モンテカルロ法からくる統計誤差をジャックナイフ法などで評価する必要がありますが、とりあえず中心値を見えます(中心値を計算するコードがあればジャックナイフ法を実装するのは難しくないので興味者ものは試してみてください)。

まず、補助的な関数を定義しておきます。

```
In [3]: 1 using Dates
        2 using Printf
        3
        4 function ls_ltr(directory::String)
        5     # ディレクトリ内のファイル一覧を取得
        6     files = readdir(directory)
        7
        8     # ファイル情報 (タイムスタンプ サイズ) を取得して、タイムスタンプでソート
        9     file_info = [(file, unix2datetime(stat(joinpath(directory, file), statfs{true})))
       10     sorted_files = sort(file_info, by = x -> x[2]) # タイムスタンプでソート
       11
       12     # 結果を表示
       13     for (file, mtime, size) in sorted_files
       14         @printf "%-20s %10d bytes %s\n" file size Dates.format(mtime, "{y}-{m}-{d} {H}:{M}:{S}")
       15     end
       16 end
```

Out[3]: ls\_ltr (generic function with 1 method)

In [4]:

```
1 # ディレクトリを試みる ls -ltr 相当
2 ls_ltr("measurements/")
3
```

```
Heatbath_L04040424_beta5.7_quenched      160 bytes 2024-09-12 02:27:44
.DS_Store                                6148 bytes 2024-09-13 00:40:39
```

Type *Markdown* and LaTeX:  $\alpha^2$

更にファイルを確認すると、

In [5]:

```
1 ls_ltr("measurements/Heatbath_L04040424_beta5.7_quenched/")
2 #;ls -ltr
```

```
Pion_correlator.txt      10885 bytes 2024-09-12 02:35:07
Plaquette.txt            6086 bytes 2024-09-12 02:35:07
Polyakov_loop.txt       10605 bytes 2024-09-12 02:35:07
```

3つファイルがあるはずです。ここでは、Pion\_correlator.txtを指定してみます。

Pion\_correlator.txtの中身を見えます。最後10行を見ると、

In [6]:

```
1 fname = "measurements/Heatbath_L04040424_beta5.7_quenched/Pion_c
```

```
In [7]: 1 # ファイル全体を読み込んで行ごとに分割
        2 lines = readlines("measurements/Heatbath_L04040424_beta5.7_quenc
        3
        4 # 最後の 10 行を表示 (tail -n 10 と同じ)
        5 println(join(last(lines, 10), "\n"))
        6
```

```
139.04621111324613 1.0797610182097874 0.26419403740532554 0.0446034
191079106 0.011266816933202806 0.002662053236684579 0.00058270369224
92386 0.00015067589708806523 3.6423686739692436e-5 1.071293547429295
9e-5 3.097846341092289e-6 1.1135105011309169e-6 2.6316572199784773e-
7 3.9986918393380845e-7 1.5989608802239833e-6 8.04056078654367e-6 2.
9485808353281206e-5 0.00018782898936566957 0.0006918494483918539 0.0
03141352763034348 0.012816354875409027 0.05856416036579575 0.2465982
9122554605 0.726184675898592 #pioncorrelator
138.72838950778663 0.7910962294159757 0.31109682375048614 0.0746787
9774856102 0.021412011470686842 0.006745050098406229 0.0017790228755
031707 0.0005199340402332395 0.00016353421230563283 5.54274083193952
4e-5 1.414172034781564e-5 3.51785822357365e-6 6.728473679144047e-7
2.7262975026164e-7 4.929635762793107e-7 2.092737053590124e-6 1.10146
33318324876e-5 6.512640116003829e-5 0.0002761730643989568 0.00127098
609232967 0.007633769678123072 0.035571329849499775 0.21994538408962
04 1.4101892629702615 #pioncorrelator
139.6775665223994 1.0727287812275015 0.35026187477958826 0.05253377
749412984 0.012248434514822526 0.004023905214494523 0.00130106856775
06118 0.0004483483015103858 0.00012593735275972303 4.085325162252919
e-5 1.4363365967847713e-5 4.95895903502648e-6 1.4259152389074012e-6
3.1561584850798714e-7 3.8135929631118896e-7 1.356512808023021e-6 7.6
19793169420492e-6 3.923733644686655e-5 0.00024032306227463545 0.0013
23406997602224 0.006722297990887953 0.033949704878182635 0.203431374
09495307 1.4762307102754226 #pioncorrelator
136.8054678206421 0.9243442669908161 0.17966063094658416 0.04227209
4601404825 0.007842695749448132 0.0019800821680993224 0.000535783246
0676035 0.00013601627292379747 3.350169927829838e-5 1.02779834678473
51e-5 3.4144719466021013e-6 1.0377720458917313e-6 3.547276647225394e
-7 3.0053014947819104e-7 1.3404562809902918e-6 7.035226776359518e-6
3.0976727378186094e-5 0.000148237403583703 0.0006022512078763791 0.0
024925936963269666 0.01111640507473699 0.054438736527951455 0.189967
04270933135 1.2937661529773883 #pioncorrelator
137.30051381042688 0.9452749836476096 0.24632998754425336 0.0429774
7689957868 0.011223093794067229 0.0033117700569872594 0.001138210962
7573259 0.00036996670039525465 8.259817169503461e-5 1.67391911731955
e-5 3.2508508923980295e-6 5.867211455118336e-7 2.2791789425038434e-7
3.8049873858772697e-7 1.2731043624165372e-6 5.455681664481585e-6 3.0
209128993958132e-5 0.00015576479067316506 0.0007917663950342317 0.00
3991696651316799 0.01618242733048094 0.04670498578160506 0.149344805
42910597 0.9116504613429612 #pioncorrelator
138.63237956705473 1.4034056711333527 0.13717311918474923 0.0372838
7948128758 0.009062992974599645 0.002071282268902401 0.0004544425656
5569974 0.0001224200595471691 2.3994347360782158e-5 4.14691021073143
e-6 7.711564374769546e-7 1.3496193716078578e-7 7.201926601343688e-8
2.1461351911365705e-7 8.214016970056122e-7 3.3057951478120154e-6 1.4
259520325276914e-5 6.578393888269158e-5 0.0002751664006074628 0.0011
939793667339552 0.005588888486733478 0.02816986134452615 0.206074595
55194543 0.735358445843318 #pioncorrelator
138.97489528690704 1.5040977096195642 0.15360663569771954 0.0321657
0591275556 0.008092184550918256 0.0017720256395454025 0.000412102836
50883574 0.00010329619171088184 2.2404662154989186e-5 4.395722130544
363e-6 8.108985817803745e-7 3.7325821270380414e-7 6.573212963160813e
-7 1.6354831162513145e-6 5.474659517382022e-6 1.5082278435531622e-5
5.157176455167082e-5 0.00019217483473491631 0.0006881351262614871 0.
002211216182503624 0.010702984068306843 0.0426764578063929 0.2090276
331239712 1.6856176094963213 #pioncorrelator
135.21512509841557 1.185957940561724 0.17759503186160655 0.05605669
2883930476 0.015516463857372199 0.0036233670488367183 0.000704991730
5599444 0.00012529323859896034 2.387790012666083e-5 4.52674212252377
1e-6 1.0477659122274749e-6 3.528596573788002e-7 5.594858989556658e-7
1.6648435886417957e-6 5.531065599213228e-6 1.5976198619972958e-5 5.7
```



```

7622581657457e-5 0.00029578779251904696 0.001225088724988138 0.00569
7579436850132 0.0221662697122308 0.05840428672146423 0.2350285186940
8703 1.1199353645672918 #pioncorrelator
135.28734537845367 0.9566192764875986 0.3042439465782666 0.06330895
802669358 0.013672739682528353 0.0038162209240164262 0.0013831839883
92987 0.00047250920335419424 0.0001626057934367491 4.379334396678051
e-5 1.2853028430320478e-5 3.7307088940371445e-6 1.1898740870244572e-
6 1.035900812935201e-6 3.825656318094667e-6 1.4646498431026224e-5 5.
912977429458025e-5 0.00022768362001483684 0.000995915369843915 0.004
760576442806712 0.01895332595185582 0.07819720037047138 0.2639136041
6405646 1.0281037238585105 #pioncorrelator
137.3767942283306 0.8602474021193987 0.1977857066462282 0.046508057
62461799 0.016030822713836046 0.004619527087072332 0.001247291840750
1858 0.00034927177121717184 0.00010371690131341567 2.891232382532716
6e-5 6.586879669591567e-6 1.4872003733545615e-6 4.1298936102516943e-
7 4.686939351973488e-7 2.427076916972067e-6 1.3254874532364866e-5 6.
037747210310062e-5 0.0002490375199441326 0.000892543913690909 0.0040
95029941798967 0.016858166408478578 0.0654220829718375 0.30621764505
39151 1.1486902616821646 #pioncorrelator

```

$G(t)$ を $\pi$ 中間子の相関関数として、「トラジェクトリ番号  $G(t = 0) G(t = 1) \dots G(t = 23)$ 」の順に並んでいます。これを解析していきます。

$Nt = 24$  で、かつ熱化で捨てる個数を11個とします。

```

In [8]: 1 Nt = 24;
        2 N_therm = 11; # モンテカルロの熱化、どれくらい捨てるか。

```

```

In [9]: 1 values= []
        2 for u in readlines(fname)
        3     if u[1] == '#'
        4     else
        5         value = parse.(Float64, split(u)[1:Nt] )
        6         push!(values,value)
        7     end
        8
        9 end
       10 # values

```

```

In [10]: 1 pions = zeros(Nt,length(values))
        2 for i = 1:length(values)
        3     pions[1:Nt,i] = values[i][1:Nt]
        4 end
        5 size(pions)
        6 Ndata = size(pions)[2]
        7 println("総データ数は$(Ndata)個。 熱化の分$(N_therm)個だけ放棄するので、

```

総データ数は21個。 熱化の分11個だけ放棄するので、10個だけ解析します。

ここから $\pi$ 中間子の相関関数を図示するために、Plots をロードする。

```

In [11]: 1 using Plots

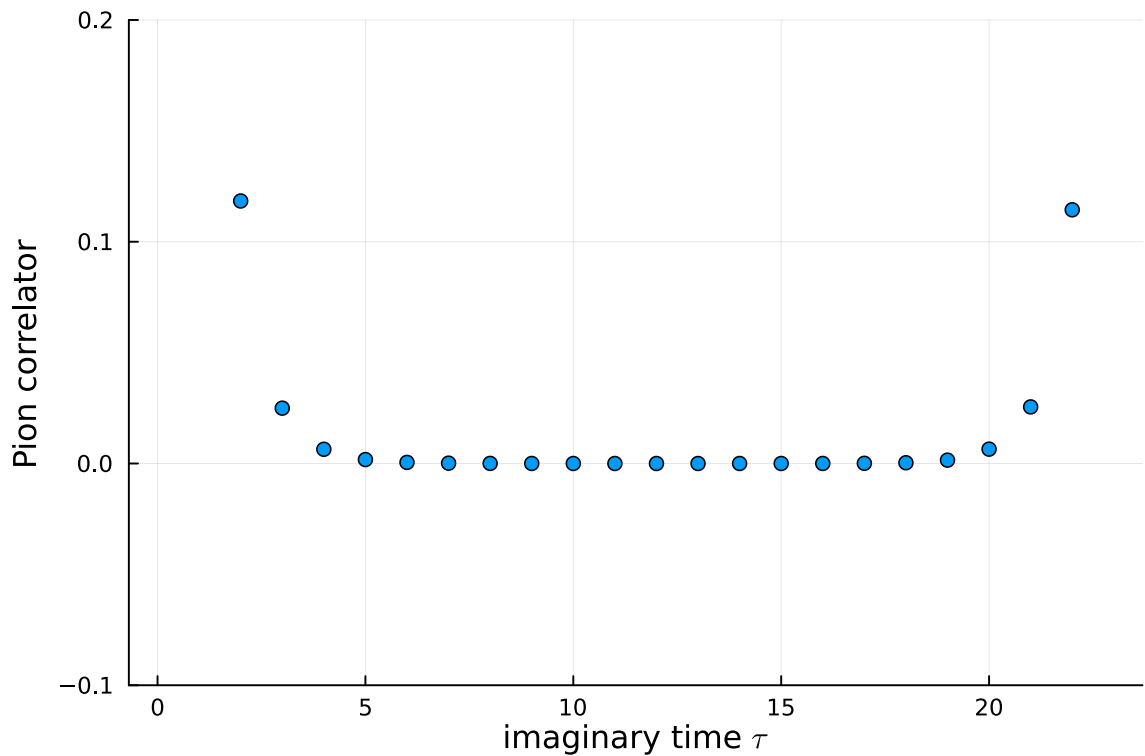
```

```

In [12]: 1 mean_value_pions = zeros(Nt)
          2 for i = N_therm:length(values)
          3     mean_value_pions[1:Nt] += pions[1:Nt,i]
          4 end
          5 mean_value_pions/=Ndata
          6 mean_value_pions
          7 ts = collect(0:Nt-1)
          8 scatter(ts,mean_value_pions,xlabel=raw"imaginary time $\tau$",yl

```

Out [12]:



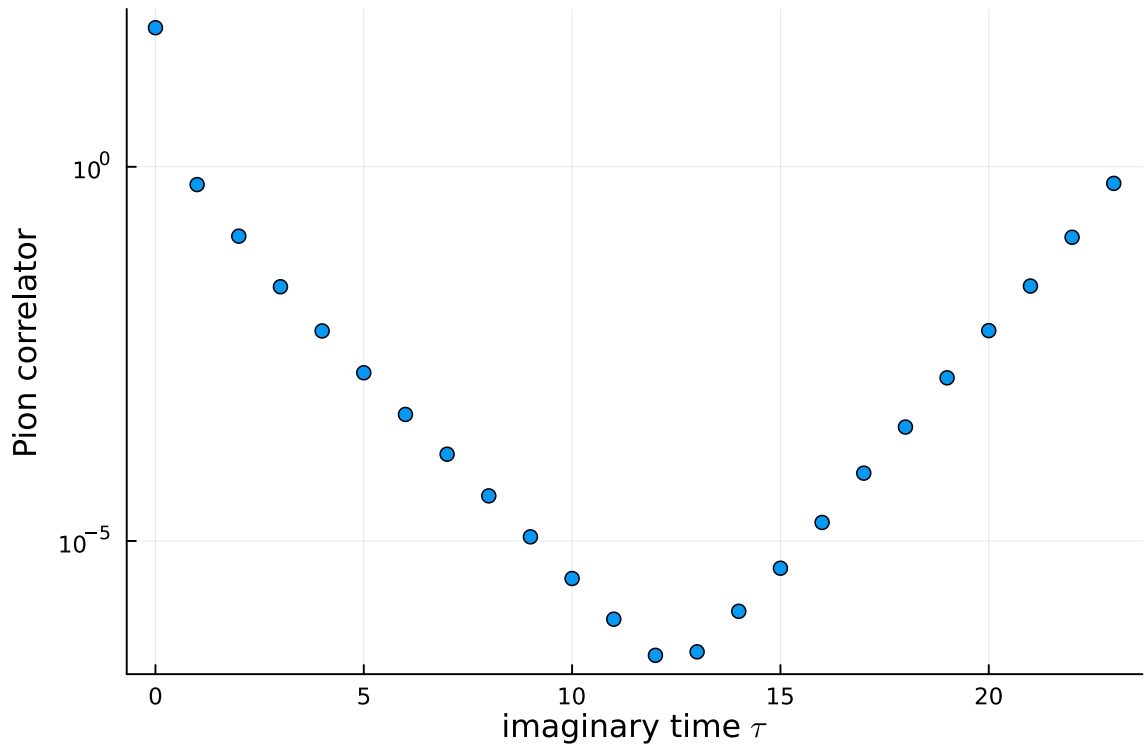
相関関数は、虚時間形式なので虚時間発展を考えると  $e^{-H\tau} \sim e^{-m_\pi\tau}$  となりそうですが(最小固有値が一番長く寄与をのこすからです、大きな固有値は最初に消えていきます。今の場合には $\pi$ 中間子の質量がそれに相当します)、実際には周期的境界条件の影響で反対側からの寄与があるため

$$\langle \pi(\tau)\pi(0) \rangle \sim \cosh(m(\tau - T/2))$$

を得ます。y軸を対数表示すると以下ようになります。

In [13]: 1 `scatter(ts,mean_value_pions,xlabel=raw"imaginary time $\tau$",yl`

Out[13]:



今の場合、一番小さい固有値 (長距離) で効いてくる減衰率がパイ中間子の質量になるので、以下のように質量を推定できます。

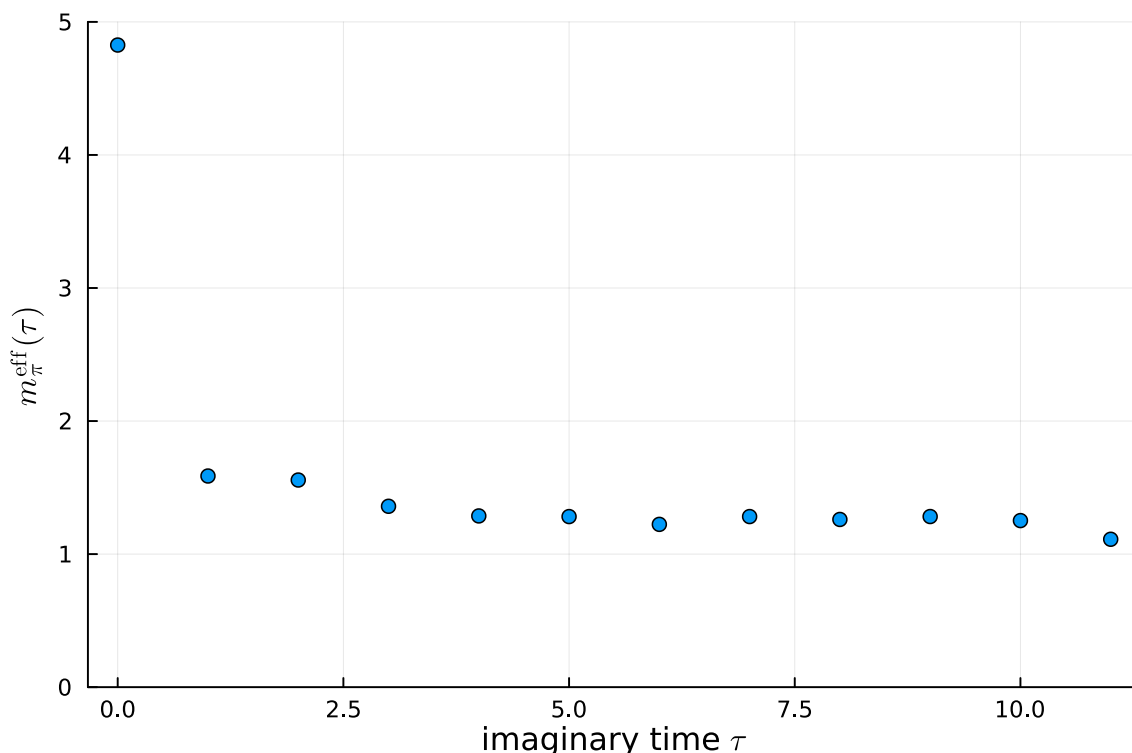
```
In [14]: 1 meffs = []
2 ts2 = []
3 for t=1:div(Nt,2)
4     meff = -log(mean_value_pions[t+1]/mean_value_pions[t])
5     println("(t-1) $meff")
6     push!(ts2,t-1)
7     push!(meffs,meff)
8 end
```

```
0 4.82606433836034
1 1.586924210599467
2 1.5567847174710714
3 1.359422217165435
4 1.2868648766503503
5 1.2820195551758056
6 1.2234917771588376
7 1.2820693175220486
8 1.2604294575671238
9 1.28229473259596
10 1.2519246070059782
11 1.1115371560492464
```

これは以下のように図示できます。

In [15]: 1 `scatter(ts2,meffs,xlabel=raw"imaginary time $\tau$",ylabel=raw"$`

Out [15]:



有効質量が(ほぼ)一定になってるところを取り出します。本当はエラーバー込みで評価する必要がありますが、詳しい解析は割愛する。まず色々とロードします。

In [16]:

```

1 using Statistics
2 using LaTeXStrings
3 using Markdown
4 # もし
5 # ArgumentError: Package LaTeXStrings not found in current path.
6 # - Run `import Pkg; Pkg.add("LaTeXStrings")` to install the LaT
7 # の様なエラーが出ればターミナルでJulia を呼び出して、必要なパッケージをイン
8 # pkg> add LaTeXStrings
9 # として戻って来てください。

```

In [17]:

```

1 m_pi = mean(meffs[3:12])
2 pi_mass_str_lat = L"今のセットアップで $\pi$ 中間子の質量はおおよそ $am_{\pi}$
3
4 # Markdownを使って表示
5 display(Markdown.MD(pi_mass_str_lat))

```

"今のセットアップで  $\pi$  中間子の質量はおおよそ  $am_{\pi} = 1.2897$  です。"

これを格子間隔 $a$ を取り除いて物理的なスケールとして表すなら以下のようにします。格子間隔は、結合定数 $\beta = 6/g^2$ の関数であることを使います。



## 4 現実と比較する

もし、物理的なスケール(MeVなど)に直したければ、 $\beta$ 関数を用いれば良いです。 $\beta$ 関数を使うと結合定数 $g$ の情報をカットオフスケール $1/a$  [MeV]に焼き直すことができます。(  $\beta$ 関数と  $\beta = 6/g^2$ の $\beta$ は関係がない偶然の文字の衝突です、念の為)

[hep-lat/9806005 \(https://arxiv.org/abs/hep-lat/9806005\)](https://arxiv.org/abs/hep-lat/9806005) の(2.18)によると、クエンチ近似の範囲内で、格子間隔 $a$ と $\beta = 6/g^2$ の関係は、

$$\ln(a/r_0) = -1.6805 - 1.7139(\beta - 6) + 0.8155(\beta - 6)^2 - 0.6667(\beta - 6)^3,$$

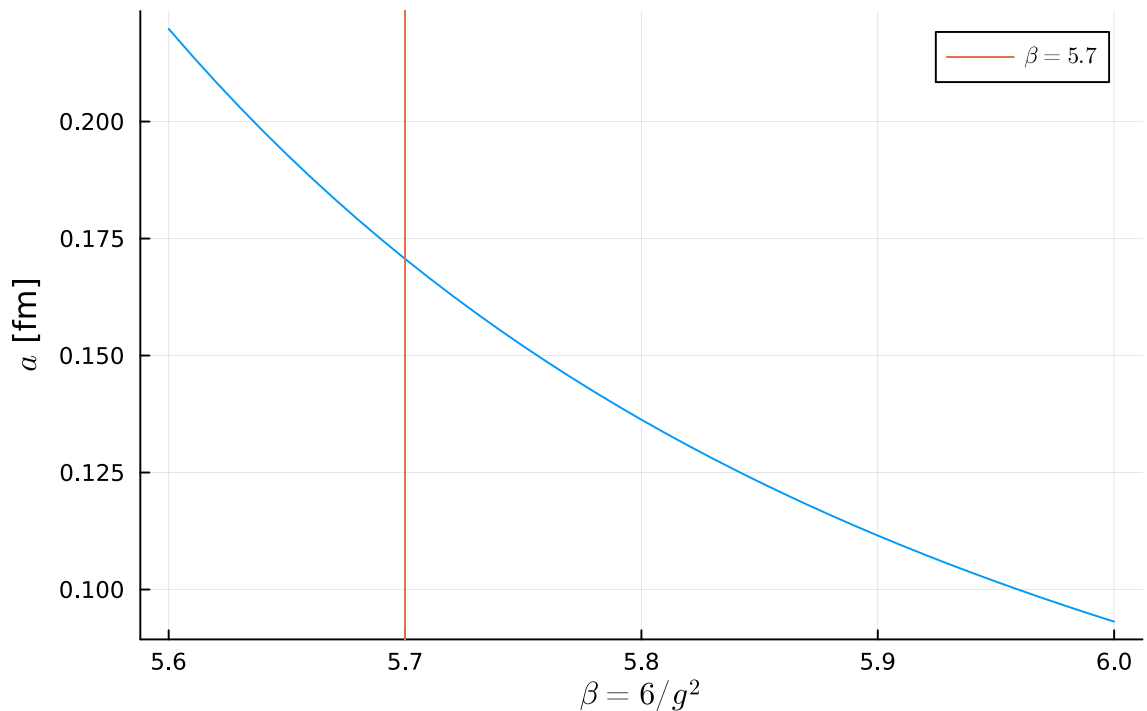
なので、これとSommer scale  $r_0 = 0.5[\text{fm}]$ をつかうと、

```
In [18]: 1 r0=0.5
          2 a(β)= exp(-1.6805-1.7139(β-6)+0.8155(β-6)^2-0.6667(β-6)^3) * r0
```

という結合定数 $\beta = 6/g^2$ から格子間隔 $a$ を計算する関数を定義できます。つまり、ここから格子間隔 $a$  [fm]がわかります。今、シミュレーションに使った $\beta$ を以下に入れます。

```
In [19]: 1 # beta と a の関数形をプロットしてみる。
          2 betas = 5.6:0.01:6.0
          3 abetas = a.(betas)
          4 plot(betas, abetas, label=nothing, xlabel=raw"\beta=6/g^2", ylabel=
          5 plot!(title=raw"\beta = 6/g^2$ vs lattice spacing $a$ [fm]")
          6 vline!([5.7], label=raw"\beta=5.7$")
```

Out [19]:  $\beta = 6/g^2$  vs lattice spacing  $a$  [fm]



```
In [20]: 1 β = 5.7
          2 println("a=$(a(β)) fm")
```

a=0.1706601237820899 fm

すると、カットオフは、 $\hbar c = 197 \text{ MeV fm}$ をつかうと

```
In [21]: 1 β = 5.7
          2 a_inv = 197/a(β)
          3 println("1/a=$(a_inv) MeV")
```

1/a=1154.341129223266 MeV

とわかります。これを $am_\pi$ にかければ良く、以下のようにわかります。

```
In [22]: 1 # LaTeXの文字列を組み立てる
2 pi_mass_str = L"今のセットアップで  $\pi$ 中間子の質量はおおよそ  $m_\pi =$ 
3
4 # Markdownを使って表示
5 display(Markdown.MD(pi_mass_str))
```

"今のセットアップで  $\pi$ 中間子の質量はおおよそ  $m_\pi = 1489.0$  MeV です。"

## ▼ 5 まとめ

今回はLatticeQCD.jl で $\pi$ 中間子の質量を計算しました。得られた値は現実とは異なる値でした。解析に誤差を含めていないという点において、以下のような問題点や疑問点があります。

- Wilson fermion はカイラル対称性がないが、南部ゴールドストーンボゾンである $\pi$ 中間子はどうなるのか。
- ホッピングパラメータ $\kappa$ (おおよそクォーク質量の逆数)は適切だったか
- カットオフはどうか ( $\beta$ の大きさ)。結果はそもそも連続極限を取っていない。
- クエンチ近似ではクォークの動的な効果を無視している

また、他のハドロンは質量などはどうだろうか？ また今回無視した統計誤差はどうだろうか？

現実の値に近づけるというのは一筋縄ではいかない (たとえば [参考](https://www2.ccs.tsukuba.ac.jp/PACS-CS/LQCD/results/kaisetsu.pdf) <https://www2.ccs.tsukuba.ac.jp/PACS-CS/LQCD/results/kaisetsu.pdf>) などをみてください) のですが、この先はそれぞれ学んでください。

```
In [ ]: 1
```