Simulations with (hybrid) Monte Carlo Algorithms

— Introduction for beginners —

André Sternbeck

University of Jena, Germany

Helmholtz International Summer School
August/September 2014, JINR Dubna

Monographs

Particularly useful for the present topic

GL2010 *"Quantum Chromodynamics on the Lattice"* by C. Gattringer and C.B. Lang, Springer 2010.

R2010 *"Lattice Gauge Theories"* by H.J. Rothe, 3rd edition, World Scientific 2005.

MM1994 *"Quantum Fields on a Lattice"* by I. Montvay and G. Münster, Cambridge 1994.

L2010 *"Computational Strategies in Lattice QCD"* by M. Lüscher, [1002.4232], Summer School on *"Modern perspectives in lattice QCD"* Les Houches, August 3–28, 2009.

S2010 *"Simulations with the Hybrid Monte Carlo algorithm: implementation and data analysis"* by S. Schäfer, Summer School on *"Modern perspectives in lattice QCD"* Les Houches, August 3–28, 2009.

Expectation values in lattice QCD

The expectation value of an observable in lattice QCD is given by

$$\langle O \rangle = \frac{1}{Z} \int DU\, D\bar{\psi}\, D\psi\; O[U, \bar{\psi}, \psi]\; e^{-S[U,\bar{\psi},\psi]} \tag{1}$$

- Z denotes the "partition function"

$$Z = \int DU\, D\bar{\psi}\, D\psi\; e^{-S[U,\bar{\psi},\psi]} \tag{2}$$

- S denotes a lattice QCD action, there are many choices (real-valued functional of link and fermionic variables)

$$S[U, \bar{\psi}, \psi] = \underbrace{S_g[U]}_{\text{gauge part}} + \underbrace{S_f[U, \bar{\psi}, \psi]}_{\text{fermionic part}} \tag{3}$$

- O denotes an arbitrary observable, which is a (simple or complicated) functional of U, $\bar{\psi}$ and ψ

Expectation values in lattice QCD

The expectation value of an observable in lattice QCD is given by

$$\langle O \rangle = \frac{1}{Z} \int DU\, D\bar{\psi}\, D\psi\; O[U, \bar{\psi}, \psi]\; e^{-S[U,\bar{\psi},\psi]} \tag{1}$$

- Link variables $U_{x\mu}$ sit on the lattice links from coordinate x to $x + \hat{\mu}$,
- Each $U_{x\mu} \in SU(3)$ is a complex 3×3 matrix
  - $\det U_{x\mu} = 1$, $U_{x\mu}^{\dagger} = U_{x\mu}^{-1}$
  - parametrized by $(N_c^2 - 1) = 8$ real parameters

  Other gauge theories (not QCD):
  - SU(2): $U_{x\mu} \in SU(2)$, parametrized by 3 real parameters.
  - compact U(1): $U_{x\mu} \in \mathbb{C}$, parametrized by 1 real parameter (angle).
- $DU = \prod_{x,\mu} dU_{x\mu}$ the integration measure of link variables $U_{x\mu}$

Expectation values in lattice QCD

The expectation value of an observable in lattice QCD is given by

$$\langle O \rangle = \frac{1}{Z} \int DU\, D\bar{\psi}\, D\psi\; O[U, \bar{\psi}, \psi]\; e^{-S[U,\bar{\psi},\psi]} \tag{1}$$

- At each site x, ψ_x and $\bar{\psi}$ have 3×4 components ($a = 1..3, \alpha = 1..4$)
- $\psi_x = \psi^{a,\alpha}(x)$ and $\bar{\psi}_x = \bar{\psi}^{a,\alpha}(x)$ are Grassmann-valued fields
- $D\psi = \prod_{x,a,\alpha} d\psi_x^{a,\alpha}$ the integration measure, with $d\psi_x^{a,\alpha} d\bar{\psi}_x^{b,\beta}$ the integration measure of a pair of Grassmann numbers
- Integration over Grassmann variables can be done exactly:

$$\det M \sum_{k_1 \cdots k_n} \epsilon^{k_1 k_2 \cdot k_n}_{j_1 j_2 \cdots j_n} M^{-1}_{k_1 i_i} \cdots M^{-1}_{k_n i_n} = \int D\bar{\psi} D\psi\; \psi_{j_1} \bar{\psi}_{i_1} \cdots \psi_{n_1} \bar{\psi}_{i_n} \underbrace{e^{-\bar{\psi} M \psi}}_{e^{-S_f}} \tag{4}$$

- M denotes the Lattice Dirac (Fermion) matrix, det makes action non-local

Expectation values in lattice QCD

After integration over the (Grassmann) fermonic fields

$$\langle O \rangle = \frac{1}{Z} \int DU \; \hat{O}[U] \; e^{-S_{eff}[U]} \tag{5}$$

- Effective action S_{eff} real-valued but non-local functional of link variables only

$$S_{eff} = S_g[U] + \log\det M[U] \tag{6}$$

  - Wilson gauge action:
    $S_g = S_W := \beta \sum_{x,\mu<\nu} \left(1 - \frac{1}{3}\mathfrak{Re}\,\mathrm{Tr}\, U_{x\mu} U_{x+\hat{\mu},\nu} U^\dagger_{x+\hat{\nu},\mu} U^\dagger_{x\nu}\right)$
  - Wilson Dirac matrix: $M^{(q)}_{xy}[U] = \delta_{qq'} \left(\delta_{xy} - \kappa_q \sum_{\pm\mu} \delta_{y,x+\hat{\mu}} (1+\gamma_\mu) U_{x\mu}\right)$

- Observable $\hat{O}[U]$ is now also a functional of links only
  - Average plaquette: $\hat{O}[U] = O[U] = \frac{1}{6V} \sum_{x,\mu<\nu} U_{x\mu} U_{x+\hat{\mu},\nu} U^\dagger_{x+\hat{\nu},\mu} U^\dagger_{x\nu}$
  - Hadronic 2-point function: $\hat{O}[U] \sim \sum_{\text{contractions}} M^{-1}[U]_{n_i n_j} M^{-1}[U]_{n_k n_l}$

Expectation values in lattice QCD

Still, these expectation values are tremendously high-dimensional integrals which have to be solved numerically (stochastically).

Example

State-of-the-art lattice QCD simulations are performed on lattice sizes:

$$L_s^3 \times L_t = 64^3 \times 128$$

Number of integration variables ($N_c = 3, N_d = 4$):

$$\begin{aligned} 64^3 \times 128 \quad & \times (N_c^2 - 1) \times N_d \quad && = 1\,073\,741\,824 \\ 16^3 \times 32 \quad & \times (N_c^2 - 1) \times N_d \quad && = 4\,194\,304 \end{aligned}$$

For each, a numerical integration had to be performed.

Therefore, statistical methods must be employed for the numerical integration.

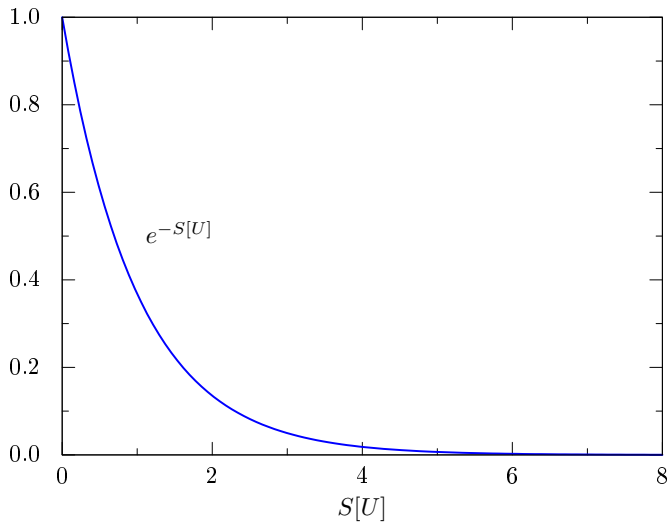
Monte Carlo estimates of expectation values

After integration over the (Grassmann) fermonic fields

$$\langle O \rangle = \frac{1}{Z} \int DU \, \hat{O}[U] \, e^{-S_{eff}[U]} \tag{7}$$

Integral will be dominated by link configurations with small action values

$$P[U] = \frac{1}{Z} e^{-S[U]} \tag{8}$$



Need to employ "important sampling", i.e., generate U with high-probability

Markov chain: $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$

$\langle O \rangle$ can be estimated by Monte-Carlo average $\overline{O}$:

$$\langle O \rangle \approx \overline{O} \equiv \frac{1}{N} \sum_{i=1}^{N} \hat{O}[U^{(i)}] \tag{9}$$

This lectures

This lecture will teach you how these Markov chains $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$ are generated.

- Metropolis algorithm
- Heat-bath algorithm
- **Hybrid Monte Carlo (HMC)**

To better understand these algorithms, it is instructive to look first at the generation of real numbers $x \in \mathbb{R}$ for a given probability distribution $p(x)$.

Random numbers with uniform probability distribution

- Probability of uniformly-distributed random numbers $r \in [x, x + dx]$

$$p(x)dx = \begin{cases} dx & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

  Probability density is normalized such that

$$\int_{-\infty}^{\infty} p(x)dx = 1 \tag{11}$$

- Popular pseudo random number generators (with large period lengths) optimal for parallel computing
  - RANLUX: Lüscher [Comput.Phys.Commun. 79 (1994) 100] http://luscher.web.cern.ch/luscher/ranlux
  - Mersenne-Twister: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

Random numbers with non-uniform probability distribution

For many physics applications, need to sample random numbers

- which are not uniformly distributed
- in high dimensions

Can generate such numbers using uniformly distributed random numbers as input for

1. Inverse transform sampling method
   (possible only for a few cases: Gauss, exponential,...)
2. Accept-reject method (Monte-Carlo technique)
   (generally applicable, efficiency is problem dependent, see important sampling)

Random numbers with non-uniform probability distribution

Transformation method

- Based on the fact that a variable x defined as the value of the cumulative distribution function F

$$x := F(y) \equiv \int_{-\infty}^{y} f(y')dy' = Pr(Y < y) \tag{12}$$

  is a uniformly-distributed random number $x \in [0, 1]$ for any f.
- $F(-\infty) = 0$, $F(\infty) = 1$
- If we know the inverse F^{-1} and define the transformation

$$y(x) := F^{-1}(x) \tag{13}$$

  we take the uniform deviate x into y which is distributed as $f(y)$

Random numbers with non-uniform probability distribution

Geometrical interpretation

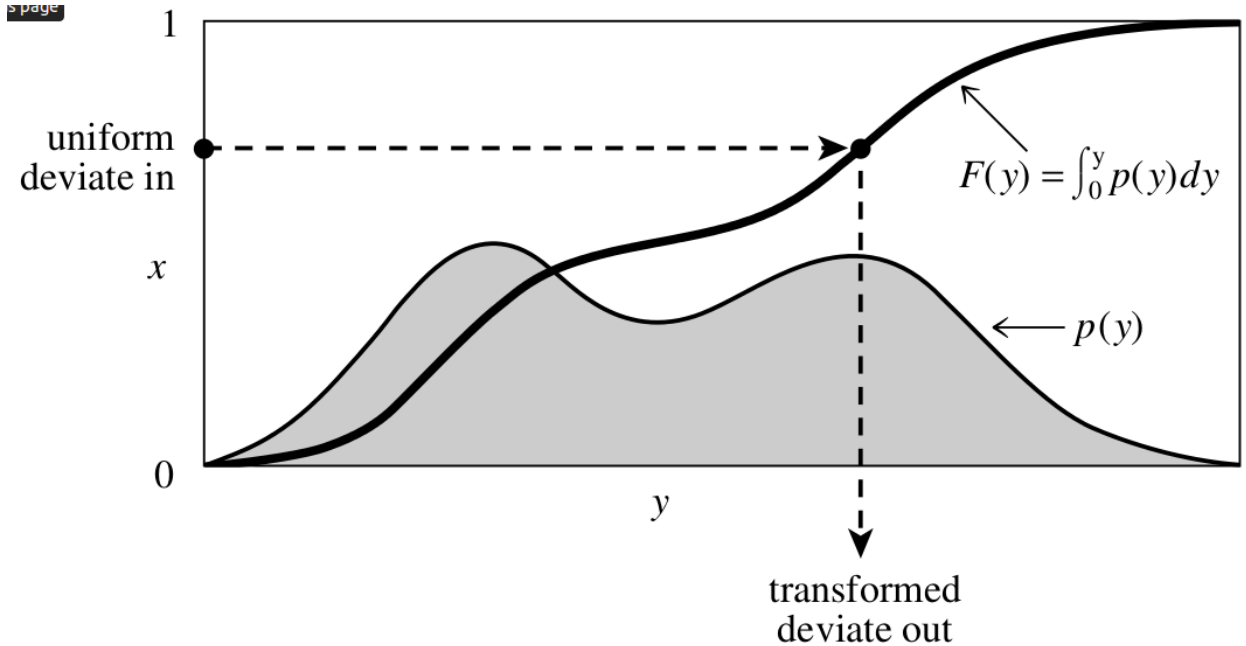


Figure: Transformation method for generating a random deviate y from a known probability distribution $p(y)$. The indefinite integral of $p(y)$ must be known and invertible. A uniform deviate x is chosen between 0 and 1. Its corresponding y on the definite-integral curve is the desired deviate.
Figure and caption taken from Numerical Recipes Vol.1, Figure 7.2.1.

Illustration for exponentially distributed random numbers:

$$p(y) = \begin{cases} e^{-y} & y \in [0, \infty) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The cumulative distribution function is

$$F(y) = \int_{-\infty}^{y} p(y')dy' = \int_{0}^{y} e^{-y'} dy' = 1 - e^{-y} . \tag{15}$$

Use inverse F^{-1} to define the transformation

$$y(x) := F^{-1}(x) = -\ln(1 - x) \tag{16}$$

Random numbers with non-uniform probability distribution

Example: exponential distribution

In practice

1. Generate uniformly distributed random numbers $x \in [0, 1)$
2. Apply transformation

$$x \to F^{-1}(x) = -\ln(1 - x)$$

3. Return transformed x

Fortran function

```
1  subroutine ran_exp(x)
2
3    ! Returns random number x with prob.
4    ! density p(x) = exp(-x)
5
6    real, intent(out) :: x
7
8    call random_number(x)
9    x = -log(1.-x)   ! x = F^{-1}(x)
10
11 end subroutine ran_exp
```

Random numbers with normal (Gaussian) probability distribution

Box-Muller method

Sample uniformly distributed $\vec{x} = (x_1, x_2)$, i.e. $x_i \in [0, 1)$, and consider the transformation $\vec{y}(\vec{x})$

$$y_1(\vec{x}) = \sqrt{-2 \ln x_1} \cos 2\pi x_2 \tag{17a}$$

$$y_2(\vec{x}) = \sqrt{-2 \ln x_1} \sin 2\pi x_2 \tag{17b}$$

whose inverse relation $\vec{x} = F(\vec{y})$ is

$$x_1(\vec{y}) = e^{-\frac{1}{2}(y_1^2 + y_2^2)} \tag{18a}$$

$$x_2(\vec{y}) = \frac{1}{2\pi} \arctan\left(y_2 / y_1\right) . \tag{18b}$$

Since the corresponding Jacobian looks like

$$\left| \frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} \right| = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = - \left[\frac{1}{\sqrt{2\pi}} e^{-y_1^2/2} \right] \left[\frac{1}{\sqrt{2\pi}} e^{-y_2^2/2} \right] \tag{19}$$

with $\vec{y}(\vec{x})$ we get two independent normal-distributed numbers y_1 and y_2.

Random numbers with normal (Gaussian) probability distribution

Box-Muller method

```
1  subroutine ran_normal(x, var)
2
3    ! Returns random number x with probability
4    ! density p(x) = 1/sqrt(2) exp(-x^2/2*var)
5    ! Default: var = 1.0
6
7    real, intent(out)          :: x
8    real, intent(in), optional :: var
9
10   real, parameter    :: pi = 4.*atan(1.0)
11   real               :: u(2), R, phi, twovar
12   complex, save      :: z
13   logical, save      :: scnd = .false.
14
15   twovar = 2.0
16   if (present(var)) twovar = 2.0 * var
17
18   if (scnd) then
19      x = aimag(z)
20      scnd = .false.
21
22   else
23
24      ! Sample two flat random numbers: u(1), u(2)
25      call random_number(u)
26
27      R = sqrt(-twovar * log(1.-u(1)))
28      phi = 2.*pi * u(2)
29
30      z = cmplx(R*cos(phi), R*sin(phi), kind=kind(z))
31      x = real(z)
32      scnd = .true.
33
34   endif
```

Random numbers with non-uniform probability distribution

For many physics applications need to sample random numbers

- which are not uniformly distributed
- in high dimensions

Can generate such numbers using uniformly distributed random numbers as input for

1. Inverse transform sampling method ✓
   (possible only for a few cases: Gauss, exponential,..., where we know F^{-1})
2. Monte-Carlo sampling (accept-reject)
   (generally applicable, efficiency problem dependent)

Random numbers with non-uniform probability distribution

Accept-reject method

If we know the cumulative distribution function F and its inverse F^{-1} we can sample random numbers y with the probability distribution $f(y)$

What if those are difficult to get? All is not lost, all we need is

- the desired probability distribution $f(y)$
- a powerful random number generator for uniformly distributed random numbers

Random numbers with non-uniform probability distribution

Accept-reject method

If we know the cumulative distribution function F and its inverse F^{-1} we can sample random numbers y with the probability distribution $f(y)$

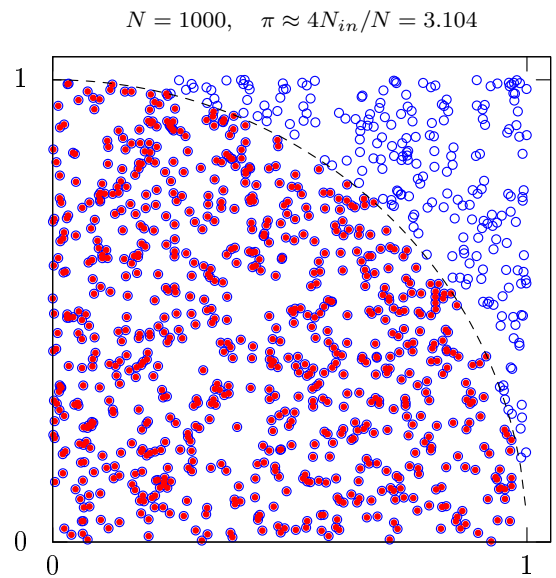
What if those are difficult to get? All is not lost, all we need is

- the desired probability distribution $f(y)$
- a powerful random number generator for uniformly distributed random numbers

Remember: Monte-Carlo method to estimate π

1. Sample N pairs of uniformly distributed random numbers (x, y) in the interval $[0, 1)$
2. Collect in N_{in} the number of pairs with $x^2 + y^2 \leq 1$
3. Ratio $\frac{N_{in}}{N}$ approaches $\frac{\pi}{4}$ when $N \to \infty$

$$\pi = \frac{A_{\bigcirc}}{A_{\square}} = 4 \lim_{N \to \infty} \frac{N_{in}}{N}$$



Similar one does for a probability function $p(x)$, for which $F(x)$ is a complex function or even unknown and we have no chance of getting F^{-1}.

If we enclose $p(x)$ inside a shape C times an easily generated distribution $f(x)$ (for which we know F^{-1}) we can sample random numbers with probability density $p(x)$ as follows:

1. Approximate $p(x)$ by a function $C \cdot f(x) > p(x) \quad \forall x$
2. Sample a uniformly distributed random number $u_1 \in [0, C)$ and calculate $x = F^{-1}(u_1)$.
3. Sample another uniformly distributed random number $u_2 \in [0, 1)$ and test if $u_2 \cdot C \cdot f(x)$ lies below $p(x)$

$$u_2 \leq \frac{p(x)}{C \cdot f(x)} \tag{20}$$

4. If fulfilled, we accept x; otherwise we reject it and start again

Random numbers with non-uniform probability distribution

Accept-reject method: Illustration

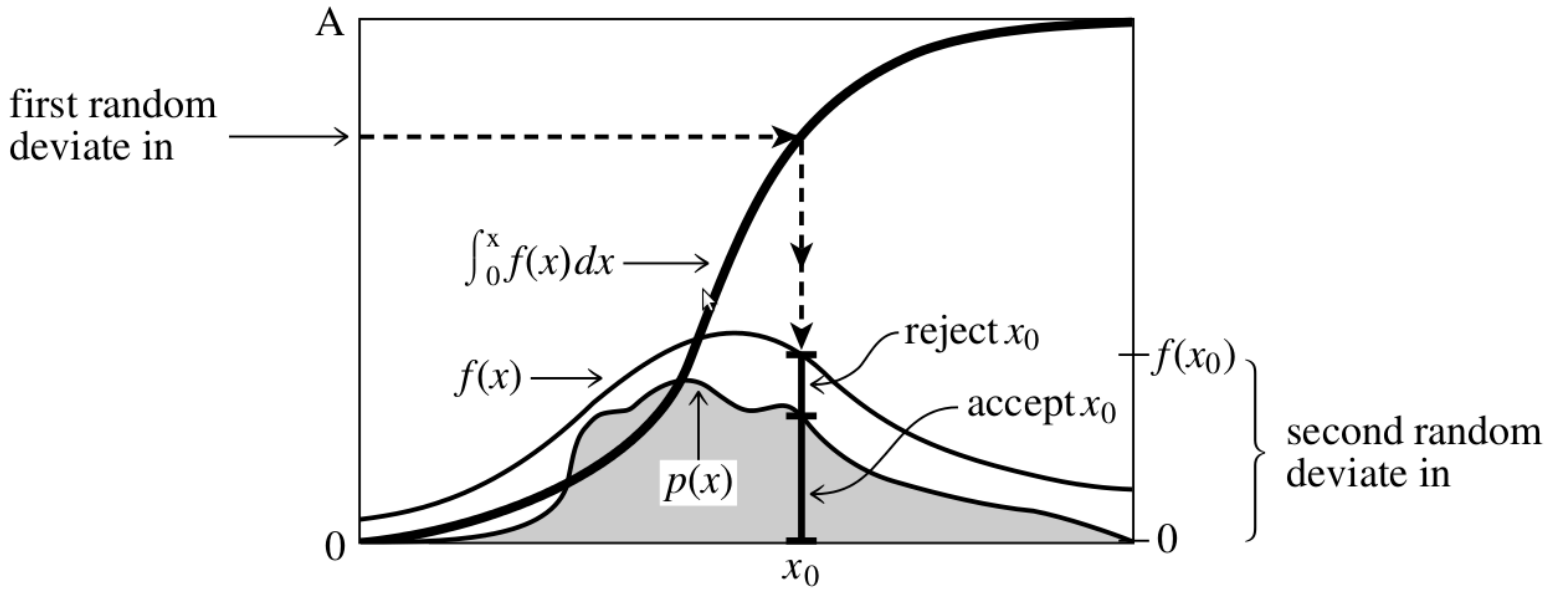


Figure: Rejection method for generating a random deviate x from a known probability distribution $p(x)$ that is everywhere less than some other function $f(x)$. The transformation method is first used to generate a random deviate x of the distribution f (cf. Figure 7.2.1). A second uniform deviate is used to decide whether to accept or reject that x. If it is rejected, a new deviate of f is found; and so on. The ratio of accepted to rejected points is the ratio of the area under p to the area between p and f. Figure and caption taken from Numerical Recipes Vol.1, Figure 7.3.1.

Arbitrary random numbers via accept-reject method

Example

Want to sample $x \in [0, 2]$ with probability density
(see heat-bath algorithm below)

$$p(x) = c_p \cdot \sqrt{2 - x} \qquad \text{where} \quad c_p \equiv \frac{3}{4\sqrt{2}} \quad \text{so that} \quad \int_0^2 dx\, p(x) = 1 \qquad (21)$$

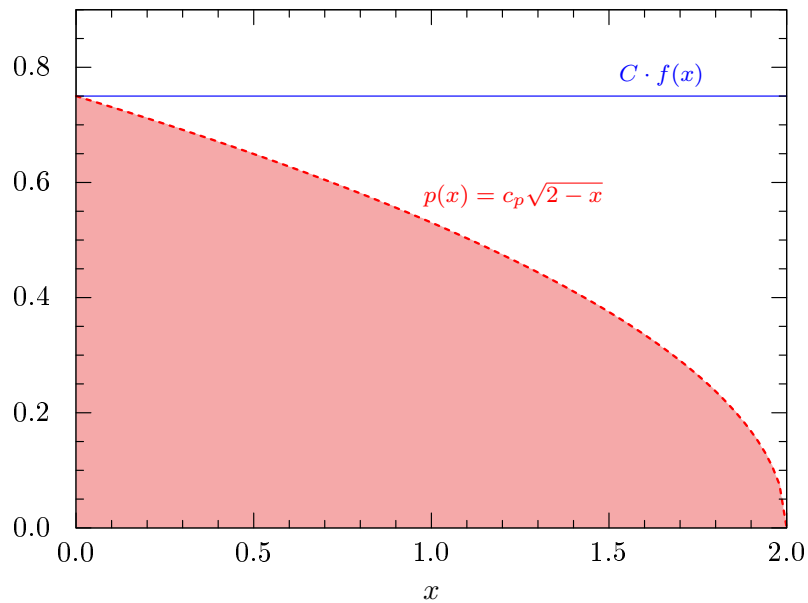
Enclose $p(x)$ by rectangle $C \cdot f(x)$ where

$$f(x) = \frac{1}{2} \quad \forall x \in [0, 2] \quad \text{and } C = \frac{3}{2}$$

Note that

$$F(a) = \int_0^a dx\, f(x) \quad \text{with } F(2) = 1$$

(uniform prob. distribution for $x \in [0, 2]$).



Recipe

1. Sample a uniformly distributed random number $u_1 \in [0, 1)$ and calculate $x = F^{-1}(u_1) = 2 \cdot u_1$.
2. Sample another uniformly distributed $u_2 \in [0, 1)$ and test if

$$u_2 \leq \frac{p(x)}{C \cdot f(x)} = \frac{1}{\sqrt{2}}\sqrt{2 - x} \qquad \Longleftrightarrow \qquad 2u_2^2 \leq 2 - x$$

3. If fulfilled accept x, otherwise go back to step 1.

Arbitrary random numbers via accept-reject method

Example code in Fortran

```fortran
program main

  implicit none
  integer :: i
  real    :: x

  do i = 1,1000000
     call sample(x)
     print *, x
  enddo

contains

  !---------------------------------------
  subroutine sample(x)

    ! Returns x in [0,2] with probability
    ! density p(x), defined further below

    real, intent(inout) :: x
    real                :: u(2)
    real, parameter     :: C = 3./2.

    do
       ! Get two random numbers u(1), u(2)
       call random_number(u)

       ! x = F^{-1}(u_1)
       x = 2.*u(1)

       ! Accept-reject step
       if ( u(2) <= p(x) / (C*f(x)) ) exit
    end do

  end subroutine sample
```

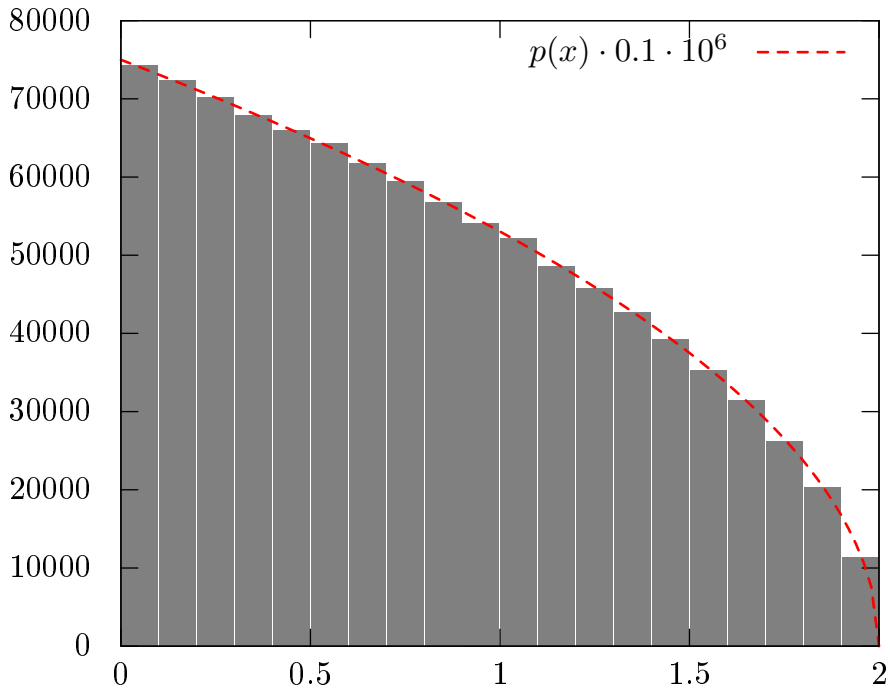
```fortran
  !---------------------------------------
  pure real function p(x)

    real, intent(in) :: x
    real, parameter :: cp = 0.75/sqrt(2.)

    if (x<0 .or. x>2) then
       p = 0
    else
       p = cp * sqrt(2-x)
    endif

  end function p

  !---------------------------------------
  pure real function f(x)

    real, intent(in) :: x

    if (x<0 .or. x>2) then
       f = 0
    else
       f = 0.5
    endif

  end function f

end program main
```

Arbitrary random numbers via accept-reject method

Example code in Fortran

Histogram $H(x)$ of sample data: $x = 0.73478, 0.14750, \ldots$

- $N = 10^6$ random numbers u_1, u_2
- bin size $\Delta H = 0.1$
- $H(x_i) = \underbrace{p(x_i) \cdot \Delta H}_{\textit{probability}} \cdot N$



We will come back to this for the heat-bath algorithm

Random numbers with non-uniform probability distribution

For many physics applications need to sample random numbers

- which are not uniformly distributed
- in high dimensions

Can generate such numbers using uniformly distributed random numbers as input for

1. Inverse transform sampling method ✓
   (possible only for a few cases: Gauss, exponential,..., where we know F^{-1})
2. Accept-Reject method (Monte-Carlo technique) ✓
   (generally applicable, efficiency is problem dependent)

   Have to find $f(x)$ which is a good approximation for $p(x)$
   $\rightarrow$ *important sampling*, otherwise there will be many rejected x

Simulation Algorithms for lattice gauge theories

Expectation values in lattice QCD

The expectation value of an observable in lattice QCD is given by

$$\langle O \rangle = \frac{1}{Z} \int DU\, D\bar{\psi}\, D\psi\; O[U, \bar{\psi}, \psi]\; e^{-S[U,\bar{\psi},\psi]} \tag{22}$$

- Z denotes the "partition function"

$$Z = \int DU\, D\bar{\psi}\, D\psi\; e^{-S[U,\bar{\psi},\psi]} \tag{23}$$

- S denotes a lattice QCD action, there are many choices (real-valued functional of link and fermionic variables)

$$S[U, \bar{\psi}, \psi] = \underbrace{S_g[U]}_{\text{gauge part}} + \underbrace{S_f[U, \bar{\psi}, \psi]}_{\text{fermionic part}} \tag{24}$$

- O denotes an arbitrary observable, which is a (simple or complicated) functional of U, $\bar{\psi}$ and ψ

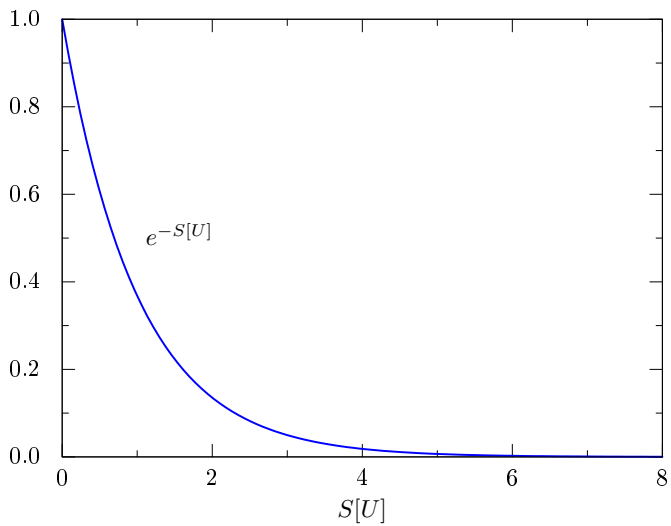
Monte Carlo estimates of expectation values

After integration over the (Grassmann) fermonic fields

$$\langle O \rangle = \frac{1}{Z} \int DU \, \hat{O}[U] \, e^{-S_{eff}[U]} \tag{7}$$

Integral will be dominated by link configurations with small action values

$$P[U] = \frac{1}{Z} e^{-S[U]} \tag{8}$$



Need to employ "important sampling", i.e., generate U with high-probability

Markov chain: $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$

$\langle O \rangle$ can be estimated by Monte-Carlo average $\overline{O}$:

$$\langle O \rangle \approx \overline{O} \equiv \frac{1}{N} \sum_{i=1}^{N} \hat{O}[U^{(i)}] \tag{9}$$

This lectures

This lecture will teach you how these Markov chains $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$ are generated.

- Metropolis algorithm
- Heat-bath algorithm
- Hybrid Monte Carlo (HMC)

To better understand these algorithms, it is instructive to look first at the generation of real numbers $x \in \mathbb{R}$ for a given probability distribution $p(x)$. ✓

Metropolis algorithm

Markov process and detailed balance

Let us denote a configuration of link variables $C = \{\ldots, U_{x\mu}, \ldots\}$.

Markov process

- Stochastic process which generates a finite set of configurations (*Markov chain*) one after the other according to some transition probability $P(C \to C')$
- $P(C \to C')$ is independent of all previous states except for C
- If $P(C \to C')$ satisfies detailed balance

$$e^{-S[C]} P(C \to C') = e^{-S[C']} P(C' \to C) \tag{25}$$

configurations are sampled with probability distribution $\propto e^{-S(C)}$
For a proof see, e.g., the book by H. Rothe "Lattice Gauge Theories..."

$\Longrightarrow$ "Simulation Time" average $\overline{O} = \frac{1}{N} \sum_{i=1}^{N} \hat{O}[C_i]$ equals ensemble average corresponding to given Boltzmann distribution

Metropolis algorithm

- Assume we have a configuration $C = \{\ldots, U_{x\mu}, \ldots\}$
- Propose a new configuration $C' = \{\ldots, U'_{x\mu}, \ldots\}$ with transition probability $T(C \to C)$ satisfying

$$T(C \to C') = T(C' \to C) \qquad \text{(reversibility)} \tag{26}$$

- Local, extended or global changes of U's are allowed
- We accept C' if the action $S(C') < S(C)$ is lowered (i.e., $\Delta S < 0$)
- If action is increased, we accept C' with probability $e^{-S(C')}/e^{-S(C)}$
- If C' is rejected, we keep C (i.e., $C' = C$) and start again

In practise one generates a random number $r \in [0, 1]$ and accepts C' if

$$r \leq e^{-\Delta S} \equiv \frac{e^{-S(C')}}{e^{-S(C)}} \tag{27}$$

The Metropolis method

Advantages

- Applicable to any system
- Will generate link configurations U with probability $P \propto e^{-S[U]}$
- Powerful, if combined with other methods which provide good candidates for new configurations (see HMC below)

Disadvantages

- If the proposal for a new configuration C' is unguided/random a large change of action will typically result in low acceptance rates
- Small changes of C will increase acceptance rate, but will also cause large autocorrelation times

$$C \sim C' \sim C'' \sim \cdots \sim C^{(n)}$$

$\rightarrow$ have to skip many C's in between

Satisfies detailed balance

- If $\Delta S < 0$

$$P(C \to C') = T(C \to C') \quad \equiv \text{"probability for suggesting } C\text{'"} \tag{28}$$

$$P(C' \to C) = T(C' \to C)\frac{e^{-S[C]}}{e^{-S[C']}} \tag{29}$$

- If $\Delta S > 0$

$$P(C \to C') = T(C \to C')\frac{e^{-S[C']}}{e^{-S[C]}} \tag{30}$$

$$P(C' \to C) = T(C' \to C) \quad \equiv \text{"probability for suggesting } C\text{"} \tag{31}$$

Since $T(C \to C') = T(C' \to C)$ detailed balance is satisfied for both cases.

Pure lattice gauge theory

Wilson gauge action

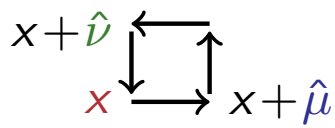
$$S_W[U] = \beta \sum_{plaq} \left(1 - \frac{1}{N_c} \mathfrak{Re}\, \mathrm{Tr}\, U_{plaq}\right) \tag{32}$$

where

$$\sum_{plaq} \equiv \sum_{x} \sum_{1 \le \mu < \nu \le 4}$$

$$U_{plaq} \equiv U_{x\mu}\, U_{x+\hat{\mu},\nu}\, U^{\dagger}_{x+\hat{\nu},\mu}\, U^{\dagger}_{x\nu}$$

Plaquette:



Degrees of freedom

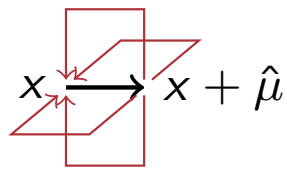
	compact $U(1)$	$SU(2)$	$SU(3)$
$U_{x\mu}$	$e^{i\phi_{x\mu}} \in \mathbb{C}$	$\mathbb{C}^{2\times 2}$	$\mathbb{C}^{3\times 3}$
	$(U^{\dagger}_{x\mu} = U^{-1}_{x\mu}, \det U_{x\mu} = 1)$		
free parameters	1 angle	3 real	8 real

Local update

- Consider change of link variable $U_{x\mu} \to U'_{x\mu}$
- All other links $U_{y\nu}$ with $(y,\mu) \neq (x,\mu)$ kept fix
- Will change S_W everywhere $U_{x\mu}$ joins a plaquette

$$S_W[U] \longrightarrow S_W[U'] = -\frac{\beta}{N_c}\mathfrak{Re}\,\mathrm{Tr}(U'_{x\mu}W_{x\mu}) - const.$$

W denotes sum of staples attached to $U_{x\mu}$:



$$W_{x\mu} := \sum_{\nu\neq\mu} \underbrace{U_{x+\hat{\mu},\nu}U^\dagger_{x+\hat{\nu},\mu}U^\dagger_{x\nu}}_{\text{"upper" staple}} + \underbrace{U^\dagger_{x+\hat{\mu}-\hat{\nu},\nu}U^\dagger_{x-\hat{\nu},\mu}U_{x-\hat{\nu},\nu}}_{\text{"lower" staple}} \qquad (33)$$

The Metropolis method in action

Code snippet for compact U(1)

```
!....
complex :: w, u(volume,4)
!....

iacc = 0
do mu = 1,4
   do i = 1, volume

      w = staple(u, i, mu)
      c = w * u(i,mu)

      ! Propose rotation u(i,mu) --> dc * u(i,mu)
      call random_number(r)
      dphi = eps * TWOPI * (r - 0.5) ! reversible: T(C->C') = T(C'->C)
      dc = cmplx(cos(dphi), sin(dphi))

      ! Change of action for proposed new link

      dS = - Re(c) + Re(c) * Re(dc) - Im(c)*Im(dc) ! -Re(c) + Re(c*dc)
      exp_dS = exp(beta * dS)

      ! Accept/reject step

      call random_number(r)
      if ( exp_dS >= r ) then
         u(i,mu) = dc * u(i,mu)
         iacc = iacc + 1
      endif

   end do
end do
!....
```

HOR algorithm (heatbath + overrelaxation)

Consider first case of SU(2) gauge theory ($N_c = 2$)

- Parametrization of SU(2) link variable $U_{x\mu}$ via real vector $a_{x\mu}$

$$U_{x\mu} = a_0 \mathbb{I} + i\vec{\sigma} \cdot \vec{a} \qquad \text{where} \quad a_{x\mu} = (a_0, a_1, a_2, a_3)_{x\mu} \in \mathbb{R}^4 \tag{34}$$

- Pauli matrices: $\vec{\sigma} = (\sigma_1, \sigma_2, \sigma_3)$

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{35}$$

satisfy $\sigma_j \sigma_k = \delta_{jk} + i\epsilon_{jkl}\sigma_l$

- Sum of staples is proportional to SU(2) element (only for SU(2))

$$W_{x\mu} = \omega_0 \mathbb{I} + i\vec{\sigma} \cdot \vec{\omega} \qquad \text{where} \quad \omega_{x\mu} = (\omega_0, \omega_1, \omega_2, \omega_3)_{x\mu} \in \mathbb{R}^4$$

$$W_{x\mu} \propto SU(2) \qquad \text{exactly:} \quad \frac{W_{x\mu}}{\sqrt{\det W_{x\mu}}} \in SU(2)$$

Heat-bath algorithm for pure SU(2) lattice gauge theory

Idea

- Consider local change of a single link variable $U_{x\mu}$, all others fixed
- Combine proposal-of-new-$U_{x\mu}$ and accept/reject step of Metropolis algorithm into one step

Boltzmann distribution of link variable $U_{x\mu} \in SU(2)$

$$dP(U_{x\mu}) = \frac{1}{Z} e^{\frac{\beta k}{2} \mathfrak{Re}\, \mathrm{Tr}\, U_{x\mu} \hat{W}_{x\mu}} \underbrace{\frac{1}{\pi^2} \delta(a_{x\mu}^2 - 1) d^4 a_{x\mu}}_{dU_{x\mu}} \tag{36}$$

where $\hat{W}_{x\mu} = W_{x\mu}/k \in SU(2)$ with $k \equiv \sqrt{\det W}$

We see: transition probability $P(U \to U') = dP(U'_{x\mu})$ is independent of U

Heat-bath algorithm for pure SU(2) lattice gauge theory

Since the Haar measure is invariant under right multiplication $dU = d(U\hat{W})$ we can define

$$V \equiv U_{x\mu}\hat{W}_{x\mu} = \mathbb{I} \cdot v_0 + \vec{\sigma} \cdot \vec{v} \quad \in SU(2) \ , \tag{37}$$

sample V with the probability distribution

$$dP(V) \ \sim \ e^{\frac{\beta k}{2} \mathfrak{Re}\ \mathrm{Tr}\ V} dV \tag{38}$$

and finally get the new link $U_{x\mu} \to V \cdot \hat{W}^{\dagger}_{x\mu} = V \cdot W^{\dagger}_{x\mu} / \sqrt{\det W_{x\mu}}$.

Again:

- V only depends on $\rho \equiv \beta \cdot k$
- new link $U \to U'$ is independent of the old, it only depends on the surrounding links and the "temperature" $1/\rho k$ (heat-bath)

Heat-bath algorithm for pure SU(2) lattice gauge theory

Since $V \in SU(2)$, the Monte-Carlo sampling is straightforward

$$e^{\frac{\beta k}{2} \mathfrak{Re}\, \mathrm{Tr}\, V} dV = e^{\rho v_0} \frac{1}{\pi^2} \delta(v^2 - 1) d^4 v \tag{39a}$$

$$= \frac{1}{2\pi^2} \underbrace{\sqrt{1 - v_0^2}\, e^{\rho v_0} dv_0}_{P(v_0)} \underbrace{d(\cos\theta) d\phi}_{\text{unit sphere in 3D}} \tag{39b}$$

That is, we are looking for vectors $v = (v_0, \vec{v})$ of unit length $|v| = 1$ where

- $v_0 \in [-1, 1]$ is a real number with the probability distribution

$$dP(v_0) \sim \sqrt{1 - v_0^2} e^{\rho v_0} dv_0 \tag{40}$$

- $\vec{v} = \sqrt{1 - v_0^2}\, \hat{\vec{v}}$, where $\hat{\vec{v}}$ is uniformly distributed point on the surface of the unit sphere in 3D.

Heat-bath algorithm for pure SU(2) lattice gauge theory

To sample v_0 we introduce the variable $\lambda \in [0, 1)$ and set

$$v_0 = 1 - 2\lambda^2$$

This gives

$$dP(v_0) \sim \sqrt{1 - v_0^2}\, e^{\rho v_0}\, dv_0 \quad \longrightarrow \quad dP(\lambda) \sim d\lambda \lambda^2 \sqrt{1 - \lambda^2}\, e^{-2\rho\lambda^2} \tag{41}$$

One option to sample λ (simple, but there are more efficient ways):

1. Generate 3 uniformly-distributed random numbers r_1, r_2, r_3 and set

$$\lambda^2 = -\frac{1}{2\rho} \ln r_1 + \cos^2(2\pi r_2) \ln(r_3) \tag{42}$$

   This generates λ^2 distributed as: $P(\lambda^2) \sim d\lambda\, \lambda^2 e^{-2\rho\lambda^2}$

2. We then accept $v_0 = 1 - 2\lambda^2$ by choosing another random number r_4 and testing if

$$r_4^2 \leq 1 - \lambda^2 \tag{43}$$

Heat-bath algorithm for pure SU(2) lattice gauge theory

Generating $\hat{\vec{v}}$ is simple: We sample two further (uniformly-distributed) random numbers r_5 and r_6 and set

$$v_1 = \sin(\theta)\cos(\phi) \qquad \cos(\theta) = 2r_5 - 1$$
$$v_2 = \sin(\theta)\sin(\phi) \qquad \sin(\theta) = \sqrt{1 - \cos^2(\theta)}$$
$$v_3 = \cos(\theta) \qquad \phi = 2\pi r_6$$

To define V we set

$$V = v_0 \mathbb{I} + \sum_{i=1,3} \sigma_i v_i \sqrt{1 - v_0^2} \tag{44}$$

New link: $U_{x\mu} \to U'_{x\mu} = V \cdot \hat{W}^{\dagger}_{x\mu} = V \cdot W^{\dagger}_{x\mu} / \sqrt{\det W_{x\mu}}$.

Microcanonical overrelaxation step

Before continuing with pure SU(3) lattice gauge theory, let's also consider microcanonical update steps of $SU(2)$ link variables.

- Change gauge links but leave the action invariant
- Reduces autocorrelation between subsequent configurations
- Not ergodic, therefore has to be combined with the heatbath step.

Consider $\mathrm{Tr}\,\alpha \cdot \omega^\dagger$ and try to find a matrix α which fulfills $\mathrm{Tr}\,\alpha\omega^\dagger = \mathrm{Tr}\,\omega^\dagger$

$$\alpha \equiv U_{x\mu} = a_0 \mathbb{I} + i\vec{\sigma}\cdot\vec{a} \quad \in SU(2) \tag{45}$$

$$\omega \equiv W_{x\mu} = \omega_0 \mathbb{I} + i\vec{\sigma}\cdot\vec{\omega} \quad \propto SU(2) \tag{46}$$

Solution:

$$\alpha = -1 + \frac{4\omega_0}{\mathrm{Tr}\,\omega\omega^\dagger}\omega \tag{47}$$

Replaces the corresponding step in the heatbath, the rest stays the same.

Note the $\mathrm{Tr}\,\omega\omega^\dagger$ can become very small$\rightarrow$ no update if $\mathrm{Tr}\,\omega\omega^\dagger < 10^{-10}$ or so.

The HOR algorithm and the Cabbibo-Marinari trick

Typical update algorithm for pure gauge theory: Hybrid overrelaxation (HOR)

$$1 \text{ HOR step } = 1 \text{ heatbath step} + \text{a few microcanonical update steps}$$

Very successful for $SU(2)$ gauge theory, more complicated if applied to $SU(n)$ directly.

Elegant solution by Cabbibo-Marinari: Do updates in $SU(2)$ subgroups and apply them subsequently.

An Update starts with $U^{(0)} = U$ and ends with $U' = U^{(m)}$, where after each update step $U^{(k)} = A^{(k)} U^{(k-1)}$ with $A^{(k)} \in SU(2)$, $U^{(k)} \in SU(n)$

Consider $SU(3)$

$$A^{(1)} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & 0 \\ \alpha_{21} & \alpha_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; A^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \alpha_{11} & \alpha_{12} \\ 0 & \alpha_{21} & \alpha_{22} \end{pmatrix}, \; A^{(3)} = \begin{pmatrix} \alpha_{11} & 0 & \alpha_{12} \\ 0 & 1 & 0 \\ \alpha_{21} & 0 & \alpha_{22} \end{pmatrix}$$

The HOR algorithm and the Cabbibo-Marinari trick

General case: loop over $m = N(N-1)/2$ matrices A where $i \in [1, N-1]$, $j \in [i+1, N]$

$$A^{(k)} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & \cdots & 0 & \cdots & 0 \\ 0 & \alpha_{ii} & 0 & \cdots & \cdots & \alpha_{ij} & \cdots & 0 \\ 0 & : & 1 & \cdots & \cdots & 0 & \cdots & 0 \\ 0 & : & 0 & 1 & \cdots & 0 & \cdots & 0 \\ 0 & \alpha_{ji} & 0 & \cdots & \cdots & \alpha_{jj} & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & \cdots & 1 \end{pmatrix} \tag{48}$$

For the (local) action it follows

$$S[A^{(k)} U^{(k-1)}_{x\mu}] = -\frac{\beta}{N} \mathfrak{Re}\, \mathrm{Tr}\, A^{(k)} X^{(k-1)}_{x\mu} + C$$

where $X^{(k-1)}_{x\mu} = U^{(k-1)}_{x\mu} W_{x\mu}$

Note that: $\mathfrak{Re}\, \mathrm{Tr}\, AX = \sum_{k \neq i,j} X_{kk} + \alpha_{ii} X_{ii} \alpha_{ij} X_{ji} + \alpha_{ji} X_{ij} \alpha_{jj} X_{jj}$

Hybrid Monte Carlo preliminaries

HOR algorithm (Heatbath + OR steps)

- Works well for local updates of gauge links (pure $SU(N)$ gauge theories)
- Shorter autocorrelation lengths compared to Metropolis algorithm
- Restricted range of application: Not an option for unquenched gauge theories (det M makes the action non-local)

HOR algorithm (Heatbath + OR steps)

- Works well for local updates of gauge links (pure $SU(N)$ gauge theories)
- Shorter autocorrelation lengths compared to Metropolis algorithm
- Restricted range of application: Not an option for unquenched gauge theories (det M makes the action non-local)

Hybrid Monte Carlo algorithm

- "Work horse" for unquenched QCD
- Produces Markov chain: $U^{(1)}, U^{(2)}, \ldots, U^{(n)}$ with probability density $e^{S_{eff}[U]}$

$$S_{eff}[U] = S_g[U] + \log \det M[U]$$

- Used in different variants nowadays
  (different improvement strategies and fermionic actions)

Main "ingredients" for Metropolis algorithm

1. Propose a new configuration $C' = \{\ldots, U'_{x\mu}, \ldots\}$ with transition probability $T(C \to C)$ satisfying

$$T(C \to C') = T(C' \to C) \qquad \text{(reversibility)} \tag{49}$$

   (local, extended or global changes of C's are allowed)

2. Accept new configuration if $e^{-\Delta S} \geq r \in [0, 1)$ (random number)

$\Rightarrow$ Detailed balance $\Longrightarrow$ Markov chain of $C, C', C'', \ldots$ with $e^{S[C]}$

Monte-Carlo simulations with fermions

Main "ingredients" for Metropolis algorithm

1. Propose a new configuration $C' = \{\ldots, U'_{x\mu}, \ldots\}$ with transition probability $T(C \to C)$ satisfying

$$T(C \to C') = T(C' \to C) \qquad \text{(reversibility)} \tag{49}$$

   (local, extended or global changes of C's are allowed)

2. Accept new configuration if $e^{-\Delta S} \geq r \in [0, 1)$ (random number)

$\Rightarrow$ Detailed balance $\Longrightarrow$ Markov chain of $C, C', C'', \ldots$ with $e^{S[C]}$

Problem of Metropolis algorithm

- Choose C' randomly ΔS will be very large $\Longrightarrow$ low acceptance rate
- Choose $C' = C + \delta C$ where δC is random but small change to C $\Longrightarrow$ large autocorrelation lengths

Goal: Global update which keeps ΔS moderate

The Hybrid Monte Carlo algorithm

1. Introduce auxiliary canonical momenta for each (dynamical) degree of freedom and set up a Hamiltonian H
   - Scalar field theory $S(\phi)$: $H(\pi, \phi) = \sum_x \pi_x^2 + S(\phi)$
   - Pure gauge theory $S_g(U)$: $H(P, U) = \sum_{x\mu} P_{x\mu}^2 + S_g(U)$

The Hybrid Monte Carlo algorithm

1. Introduce auxiliary canonical momenta for each (dynamical) degree of freedom and set up a Hamiltonian H
   - Scalar field theory $S(\phi)$: $H(\pi, \phi) = \sum_x \pi_x^2 + S(\phi)$
   - Pure gauge theory $S_g(U)$: $H(P, U) = \sum_{x\mu} P_{x\mu}^2 + S_g(U)$

2. Evolve, e.g., all π_x and ϕ_x along a trajectory τ in the phase space (p, ϕ) using Hamilton's equation of motion (EoM):

$$\frac{d}{d\tau}\phi_x = \frac{\partial H}{\partial \pi_x} \qquad \frac{d}{d\tau}\pi_x = -\frac{\partial H}{\partial \phi_x} \tag{50}$$

   - Evolution must be reversible $(\pi, \phi) \leftrightarrow (\pi', \phi')$ (see step 1 of Metropolis)
   - Numerical evolution: $\Delta H = H' - H \neq 0$ (small changes are welcome)

The Hybrid Monte Carlo algorithm

1. Introduce auxiliary canonical momenta for each (dynamical) degree of freedom and set up a Hamiltonian H
   - Scalar field theory $S(\phi)$: $H(\pi, \phi) = \sum_x \pi_x^2 + S(\phi)$
   - Pure gauge theory $S_g(U)$: $H(P, U) = \sum_{x\mu} P_{x\mu}^2 + S_g(U)$

2. Evolve, e.g., all π_x and ϕ_x along a trajectory τ in the phase space (p, ϕ) using Hamilton's equation of motion (EoM):

$$\frac{d}{d\tau}\phi_x = \frac{\partial H}{\partial \pi_x} \qquad \frac{d}{d\tau}\pi_x = -\frac{\partial H}{\partial \phi_x} \tag{50}$$

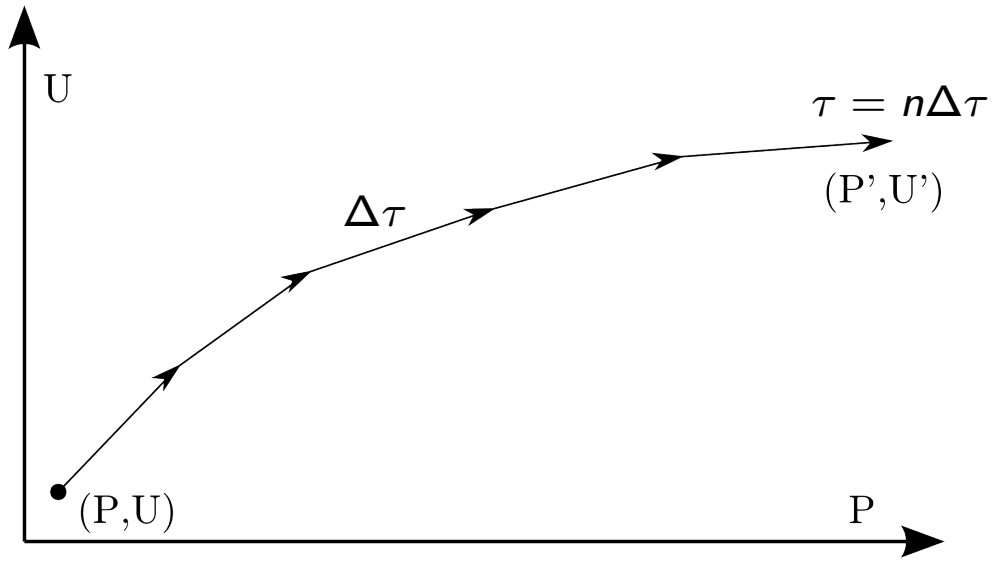
   - Evolution must be reversible $(\pi, \phi) \leftrightarrow (\pi', \phi')$ (see step 1 of Metropolis)
   - Numerical evolution: $\Delta H = H' - H \neq 0$ (small changes are welcome)

3. After one trajectory, accept/reject new (p', ϕ') using step 2 of Metropolis
   - Corrects for small $\Delta H \neq 0$ due to numerical integration of EoM

Reversibility and Metropolis step make HMC algorithm exact
(remember ingredients for Metropolis algorithm to ensure detailed balance).

The Hybrid Monte Carlo algorithm

Discrete evolution



Tunning parameter: $\Delta\tau = \frac{\tau}{n}$

- If small, $\Delta H \approx 0$ (large autocorrelation)
- If large, $|\Delta H| \gg 0$, acceptance rate decreases.

Optimum: Set τ such that acceptance rate $\sim 70 - 80\%$

HMC algorithm for scalar field theory

HMC for a scalar field theory

As an example, we consider a simple case: Scalar field theory

Lattice action (unrenormalized)

$$S[\phi] = \sum_x a^4 \left[\frac{1}{2} \sum_{\mu=1}^{4} (\Delta_\mu^f \phi_x)^2 + \frac{m_0^2}{2} \phi_x^2 + \frac{g_0}{4!} \phi_x^4 \right] \tag{51}$$

where $\Delta_\mu^f \phi_x := \frac{1}{a} \left(\phi_{x+\hat{\mu}} - \phi_x \right)$ defines a forward derivative on the lattice.

For numerical simulations it is advantageous to express S such that it only contains dimensionless quantities. We achieve this by

$$a\phi \to \sqrt{2\kappa}\phi, \qquad a^2 m_0 = \frac{1 - 2\lambda}{\kappa} - 8, \qquad g_0 = \frac{6\lambda}{\kappa^2} \tag{52}$$

which gives us

$$S[\phi] = \sum_x \left[-2\kappa \sum_{\mu=1}^{4} \phi_x \phi_{x+\mu} + \phi_x^2 + \lambda(\phi_x^2 - 1)^2 - \lambda \right] \tag{53}$$

HMC for a scalar field theory

To set up the HMC we use the last form of the action (repeated here)

$$S[\phi] = \sum_x \left[-2\kappa \sum_{\mu=1}^{4} \phi_x \phi_{x+\mu} + \phi_x^2 + \lambda(\phi_x^2 - 1)^2 - \lambda \right]$$

and define the Hamiltonian of a fictitious phase space (π, ϕ)

$$H[\pi, \phi] = \frac{1}{2} \sum_x \pi_x^2 + S[\phi] \,. \tag{54}$$

H is a real-valued function of $\phi_x \in \mathbb{R}$ and their conjugate/canonical momenta $\pi_x \in \mathbb{R}$.

We will also need the force

$$F_x := \frac{\partial S[\phi]}{\partial \phi_x} = 2\phi_x + 4\lambda(\phi_x^2 - 1)\phi_x - 2\kappa \sum_{\mu=1}^{4} (\phi_{x+\mu} + \phi_{x-\mu}) \tag{55}$$

HMC for a scalar field theory

The HMC algorithm consist of the following three steps:

1. A trajectory is started by choosing a set of Gaussian distributed momenta $\pi = \{\pi_x\}$, i.e.,

$$P[\pi] \propto e^{-\frac{1}{2}\sum_x \pi_x^2}$$

Why? Those momenta are auxiliary fields we need for step 2 and 3. We therefore have to integrate over them

$$\int [D\phi] \underbrace{\underbrace{[D\pi]}_{\prod_x d\pi_x} e^{-\sum_x \pi_x^2}}_{V\text{-dim Gaussian integral}} e^{-S[\phi]}$$

for which we use important sampling again. It is sufficient to refresh π at the start at every trajectory.

The HMC algorithm consist of the following three steps:

2. A configuration (π, ϕ) is then evolved along a discretized trajectory in the phase space until it reaches a new configuration (π', ϕ') after $n\tau$ steps. The evolution is governed by the EoM

$$\frac{d}{d\tau}\pi_x = -\frac{\partial H}{\partial \phi_x} = -\frac{\partial S}{\partial \phi_x}\,, \qquad \frac{d}{d\tau}\phi_x = \frac{\partial H}{\partial \pi_x} = \pi_x \tag{56}$$

which are

- deterministic, i.e., (π', ϕ') is unique
- reversible, i.e., $(\pi, \phi) \leftrightarrow (\pi', \phi')$
- area-preserving, i.e., $[D\pi', D\phi'] = [D\pi, D\phi]$

Important: any (discrete) implementation of the evolution must fulfill these properties, as for example the leap-frog integrator (see below).

The HMC algorithm consist of the following three steps:

3. After a full trajectory $(\pi, \phi) \to (\pi', \phi')$, that is after n steps of integration, the new configuration (π', ϕ') is either accepted or rejected using the acceptance probability of the Metropolis algorithm

$$P = \min\{1, e^{-\Delta H}\} \quad \text{where} \quad \Delta H = H' - H$$

Again one chooses a random number $r \in [0, 1)$ and accepts (π', ϕ') if

$$r \leq P$$

If $r > P$ the last ϕ is kept as initial ϕ for the next trajectory.

Note that if the evolution (step 2) was exact, i.e., we could solve the EoM exactly, H would be conserved and so all new configurations would be accepted.

However, we can solve/integrate these equations only approximately.

A popular choice to integrate the discretized Hamiltonian equations is the leapfrog algorithm, which for any finite $\Delta\tau$ is

- Simple
- Deterministic
- Reversible (apart from numerical rounding errors)
- Area preserving

Since $\Delta\tau$ is finite, numerical errors are unavoidable ($\Delta H \neq 0$). The half-size steps of the leapfrog (see below) introduce $O(\Delta\tau^2)$ errors, the full steps errors of $O(\Delta\tau^3)$. These errors are however corrected by step 3 (accept-reject step)

The leapfrog algorithm for a scalar field theory

Starting point ($\tau = 0$): $(\pi^{(0)}, \phi^{(0)})$

Elementary operations:

$$I_\pi(\epsilon): \qquad \pi^{(i)} \to \pi^{(i+1)} = \pi^{(i)} - \epsilon \overbrace{F[\phi^{(i)}]}^{Eq.(55)} \tag{57a}$$

$$I_\phi(\epsilon): \qquad \phi^{(i)} \to \phi^{(i+1)} = \phi^{(i)} + \epsilon\, \pi^{(i+1)} \tag{57b}$$

End point ($\tau = n \cdot \Delta\tau$):

$$(\pi^{(n)}, \phi^{(n)}) = \underbrace{\left[I_\pi\left(\frac{\Delta\tau}{2}\right) I_\phi(\Delta\tau)\, I_\pi\left(\frac{\Delta\tau}{2}\right) \right]^n}_{I(\Delta\tau, n)} (\pi^{(0)}, \phi^{(0)}) \tag{58}$$

The leapfrog algorithm for a scalar field theory

Remember for detailed balance the molecular dynamic step needs to be

- Reversible: $I(\Delta\tau, n) \cdot I(-\Delta\tau, n) = 1$
  (invertible mapping of phase space)
- Area-preserving: $[D\pi^{(0)}, D\phi^{(0)}] = [D\pi^{(N)}, D\phi^{(N)}]$
  integration measure preserved

We will now show that the leapfrog algorithm satisfies both

For a single step $i \to i+1$

$$(\pi^{(i+1)}, \phi^{(i+1)}) = \left[I_\pi\left(\frac{\Delta\tau}{2}\right) I_\phi(\Delta\tau)\, I_\pi\left(\frac{\Delta\tau}{2}\right) \right] (\pi^{(i)}, \phi^{(i)}) \tag{59}$$

we have for the Jacobian

$$\begin{aligned}
\det\left[\frac{\partial(\pi^{(i+1)}, \phi^{(i+1)})}{\partial(\pi^{(i)}, \phi^{(i)})}\right] &= \det\left[\frac{\partial(\pi^{(i+1)}, \phi^{(i+1)})}{\partial(\pi^{(i+1/2)}, \phi^{(i+1)})} \frac{\partial(\pi^{(i+1/2)}, \phi^{(i+1)})}{\partial(\pi^{(i+1/2)}, \phi^{(i)})} \frac{\partial(\pi^{(i+1/2)}, \phi^{(i)})}{\partial(\pi^{(i)}, \phi^{(i)})}\right] \\
&= \det\left[\begin{pmatrix} 1 & \dots \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \Delta\tau & 1 \end{pmatrix}\begin{pmatrix} 1 & \dots \\ 0 & 1 \end{pmatrix}\right] \\
&= 1
\end{aligned}$$

If for simplicity we combine the elementary operations, I_π and I_ϕ, of one integration step

$$\phi^{(i+1)} = \phi^{(i)} + \epsilon\pi^{(i)} - \frac{\epsilon^2}{2}F[\phi^{(i)}] \tag{60a}$$

$$\pi^{(i+1)} = \pi^{(i)} - \frac{\epsilon}{2}\left(F[\phi^{(i)}] + F[\phi^{(i+1)}]\right) \tag{60b}$$

we easily see that if we integrate backwards, i.e. we start the integration from $(\phi^{(i+1)}, \pi^{(i+1)})$ with $-\epsilon$ as step size we reach again $(\phi^{(i)}, \pi^{(i)})$:

$$\begin{aligned}\phi^{(i+1)} &\to \phi^{(i+1)} - \epsilon\pi^{(i+1)} - \frac{\epsilon^2}{2}F[\phi^{(i+1)}] \\ &= \phi^{(i+1)} - \epsilon\pi^{(i)} + \frac{\epsilon^2}{2}\left(F[\phi^{(i)}] + F[\phi^{(i+1)}]\right) - \frac{\epsilon^2}{2}F[\phi^{(i+1)}] \\ &= \phi^{(i+1)} - \epsilon\pi^{(i)} + \frac{\epsilon}{2}F[\phi^{(i)}] \quad = \phi^{(i)}\end{aligned} \tag{61a}$$

$$\pi^{(i+1)} \to \pi^{(i+1)} + \frac{\epsilon}{2}\left(F[\phi^{(i+1)}] + F[\phi^{(i)}]\right) \quad = \pi^{(i)} \tag{61b}$$

q.e.d.

HMC for lattice QCD simulations

HMC for lattice QCD simulations

The HMC algorithm for lattice QCD proceeds similarly.

The involved expressions (lattice fields, force term) are however more complicated and computationally expensive, which is due to the non-abelian nature of the link variables $U = \{U_{x\mu}\}$.

Comparison of lattice fields

ϕ^4	*LQCD*
$\phi_x \in \mathbb{R}$	$U_{x\mu} = e^{i\omega^a_{x\mu} T^a} \in SU(3), \quad \omega^a_{x\mu} \in \mathbb{R}, \quad T^a \in \mathfrak{su}(3), T^a = T^{a\dagger}$
$\pi_x \in \mathbb{R}$	$P_{x\mu} = iP^a_{x\mu} T^a \in \mathfrak{su}(3), \quad P^a_{x\mu} \in \mathbb{R}$

Comparison of force terms

$$\phi^4: \quad F_x := \frac{\partial S[\phi]}{\partial \phi_x}$$

$$= 2\phi_x + 4\lambda(\phi_x^2 - 1)\phi_x - 2\kappa \sum_{\mu=1}^{4} (\phi_{x+\mu} + \phi_{x-\mu}) \tag{62}$$

$$LQCD: \quad F^a_{x\mu} := \frac{\partial S[U]}{\partial U_{x\mu}} \equiv \left. \frac{\partial S[e^{i\omega} U]}{\partial \omega^a_{x\mu}} \right|_{\omega=0} \qquad \text{where} \quad \omega \equiv \{\omega^a_{x\mu} T^a\}$$

$$= \left. \frac{\partial S_g[e^{i\omega} U]}{\partial \omega^a_{x\mu}} \right|_{\omega=0} + \left. \frac{\partial}{\partial \omega^a_{x\mu}} \ln \det M^2[e^{i\omega} U] \right|_{\omega=0} \tag{63}$$

(for simplicity we consider $N_f = 2$ degenerate fermions)

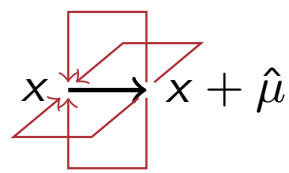
Let's focus first on pure SU(3) lattice gauge theory with the standard Wilson gauge action (i.e., no fermions).

$$S_g = S_W := \beta \sum_{x,\mu<\nu} \left(1 - \frac{1}{3}\mathfrak{Re}\,\mathrm{Tr}\, U_{x\mu} U_{x+\hat{\mu},\nu} U^\dagger_{x+\hat{\nu},\mu} U^\dagger_{x\nu}\right) \tag{64}$$

The force term reads

$$F^a_{x\mu} := \left.\frac{\partial S_W[e^{i\omega} U]}{\partial \omega^a_{x\mu}}\right|_{\omega=0} = -\frac{\beta}{3}\mathfrak{Re}\,\mathrm{Tr}\, iT^a U_{x\mu} W_{x\mu} = \frac{\beta}{3}\mathfrak{Im}\,\mathrm{Tr}\, T^a U_{x\mu} W_{x\mu} \tag{65}$$

W denotes the sum of staples at x in direction μ:



$$W_{x\mu} := \sum_{\nu\neq\mu} U_{x+\hat{\mu},\nu} U^\dagger_{x+\hat{\nu},\mu} U^\dagger_{x\nu} + U^\dagger_{x+\hat{\mu}-\hat{\nu},\nu} U^\dagger_{x-\hat{\nu},\mu} U^\dagger_{x-\hat{\nu},\nu} \tag{66}$$

HMC for pure lattice gauge theory

For the Wilson gauge action the HMC reads:

1. A trajectory is started choosing a set of Gaussian distributed momentum components $P^a_{x\mu} \in \mathbb{R}$. These define the set of initial momenta $P = \{P_{x\mu}\}$ with

$$P_{x\mu} = \sum_a i P^a_{x\mu} T^a \quad \in \mathfrak{su}(3)$$

conjugate to the link variables $U = \{U_{x\mu}\}$.

2. This start configuration (P, U) is evolved along a discretized trajectory applying n leapfrog steps of size $\Delta\tau = \tau/n$ (typically $\tau = 1$)

$$(P', U') = \underbrace{\left[I_P\left(\frac{\Delta\tau}{2}\right) I_U(\Delta\tau)\, I_P\left(\frac{\Delta\tau}{2}\right)\right]^n}_{I(\Delta\tau, n)} (P, U) \tag{67}$$

Hamiltonian: $H[P, U] = \frac{1}{2}\sum_{x\mu a}(P^a_{x\mu})^2 + S_W[U]$

The elementary operations for each leapfrog step are

$$I_P(\epsilon): \quad P^a_{x\mu} \to P^a_{x\mu} - \epsilon F^a_{x\mu}[U] \tag{68a}$$

$$I_U(\epsilon): \quad U_{x\mu} \to e^{i\epsilon P^a_{x\mu} T^a} U_{x\mu} \tag{68b}$$

Remember, the force term components read $F^a_{x\mu} = \frac{\beta}{3} \mathfrak{Im}\,\mathrm{Tr}\, T^a U_{x\mu} W_{x\mu}$, and that we split $I_P(\epsilon)$ into two half steps. P after the first half step is used for $I_U(\epsilon)$

3. After n leapfrog steps the new configuration (P', U') is either accepted or rejected using the Metropolis step: We choose a random $r \in [0, 1)$ and accept (P', U') if $r < P$ with

$$P = \min\{1, e^{-\Delta H}\} \quad \text{where} \quad \Delta H = H' - H$$

Otherwise (P', U') is rejected and becomes the old $(P', U') = (P, U)$.

Useful simplification for implementation of step 2:

$$
\begin{aligned}
(P', U') &= \left[I_P\left(\frac{\Delta\tau}{2}\right) I_U(\Delta\tau)\, I_P\left(\frac{\Delta\tau}{2}\right) \right]^n (P, U) \\
&= I_P\left(\frac{\Delta\tau}{2}\right) I_U(\Delta\tau) \underbrace{I_P\left(\frac{\Delta\tau}{2}\right) I_P\left(\frac{\Delta\tau}{2}\right)}_{I_P(\Delta\tau)} I_U(\Delta\tau) \cdots I_U(\Delta\tau) I_P\left(\frac{\Delta\tau}{2}\right)(P, U) \\
&= \underbrace{I_P\left(\frac{\Delta\tau}{2}\right) I_U(\Delta\tau)}_{\text{final step}} \underbrace{I_P(\Delta\tau) I_U(\Delta\tau) \cdots I_P(\Delta\tau) I_U(\Delta\tau)}_{(n-1)\text{ full steps}} \underbrace{I_P\left(\frac{\Delta\tau}{2}\right)}_{\text{first half step}} (P, U)
\end{aligned}
$$

HMC for lattice QCD with fermions

Up to now we have neglected contributions to S from dynamical quarks. Adding them will change the action, in particular the force, in a nontrivial way (most expensive part of HMC calculation).

For step 3, the change of H is harmless, but the force F becomes non-local and renders the calculation of $F^a_{x\mu}[U]$ computationally expensive for each of the many iterations (step 2)

$$\mathcal{I}_P(\epsilon): \quad P^a_{x\mu} \;\to\; P^a_{x\mu} - \epsilon F^a_{x\mu}[U]$$

Calculation of force F

- For plain Wilson action: only staples are needed
- With fermions: an inversion of the Dirac operator is needed for each leapfrog step $\mathcal{I}_P(\epsilon)$.

Let us look at this now in more detail.

HMC for lattice QCD with fermions

Lattice action with dynamical fermions

$$S[U,\psi,\bar{\psi}] = S_g[U] + S_f[U,\psi,\bar{\psi}] \tag{69}$$

Simplest form: Wilson gauge action for the gauge part $S_g = S_W$ and plain Wilson fermions for the fermionic part

$$S_f[U,\psi,\bar{\psi}] = \sum_{q=u,d,s,..} \underbrace{\sum_x \left\{ \bar{\psi}^q_x \psi^q_x - \kappa_q \sum_{\pm\mu} \left(\bar{\psi}^q_{x+\hat{\mu}} [1+\gamma_\mu] U_{x\mu} \psi^q_x \right) \right\}}_{\sum_{xy} \bar{\psi}_x M^{(q)}_{xy} \psi_x} \tag{70}$$

$$\text{where} \quad M^{(q)}_{xy}[U] = \delta_{qq'} \left(\delta_{xy} - \kappa_q \sum_{\pm\mu} \delta_{y,x+\hat{\mu}} (1+\gamma_\mu) U_{x\mu} \right) \tag{71}$$

is the Wilson Dirac operator; notation: $\gamma_{-\mu} \equiv -\gamma_\mu$ and $U_{x,-\mu} = U^\dagger_{x-\hat{\mu},\mu}$

HMC for lattice QCD with fermions

The expectation value of an observable in lattice QCD is given by

$$\langle O \rangle = \frac{1}{Z} \int DU \, D\bar{\psi} \, D\psi \; O[U, \bar{\psi}, \psi] \; e^{-S[U,\bar{\psi},\psi]} \tag{72}$$

After integration over the (Grassmann) fermonic fields

$$\langle O \rangle = \frac{1}{Z} \int DU \; \hat{O}[U] \; e^{-S_{eff}[U]} \tag{73}$$

with

$$S_{eff}[U] = S_g[U] - \ln \det M \tag{74}$$

We got rid off the Grassmann numbers but our effective action is non-local.

How to deal with that?

Dealing with the fermionic determinant

What about treating the determinant as part of an observable?

$$\langle O \rangle = \frac{\langle \det M_u \det M_d O[U] \rangle_U}{\langle \det M_u \det M_d \rangle_U} \tag{75a}$$

where

$$\langle (\bullet) \rangle_U \equiv \frac{1}{Z} \int [DU] \, e^{S_g[U]} \, (\bullet) \tag{75b}$$

is the path integral of pure gauge theory

Sounds like a nice idea, but values for the determinant covering several orders of magnitude

$\Longrightarrow$ large fluctuations. Not an option for QCD.

Dealing with the fermionic determinant

Fermionic determinant is considered as part of the probability weight factor for the Markov chain

$$P[U] = \frac{1}{Z}\, e^{-S_g[U]} \det M_u \det M_d \tag{76}$$

To this end: determinant must be (a) real and (b) positive

(a) M is γ_5-symmetric: $M_{yx} = \gamma_5 M^\dagger_{xy} \gamma_5$ and so

$$\det M = \det M^\dagger = (\det M)^* \qquad \text{is real}$$

(b) M can be negative: Consider two mass-degenerate fermion flavors:

$$M \equiv M^{(u)} = M^{(d)} \qquad (\kappa_u = \kappa_d)$$

Then: $\det M^{(u)} \det M^{(d)} = \det M \det M \geq 0$ is positive.

It is useful to consider: $\det M \det M = \det M \det M^\dagger = \det MM^\dagger$ which is Hermitian.

Dealing with the fermionic determinant

Ok fine. But still how to calculate the determinant? M is sparse but nonetheless a huge matrix.

Dealing with the fermionic determinant

Ok fine. But still how to calculate the determinant? M is sparse but nonetheless a huge matrix.

We use the exact relation [GL]:

$$\frac{\pi^N}{\det A} \cdot e^{\sum_{kl} \chi_k^\dagger (A)_{kl}^{-1} \chi_l} = \int \prod_{k=1}^{N} d\phi_R d\phi_I \, e^{-\sum_{kl} \phi_k A_{kl} \phi_l + \sum_k (\phi_k^\dagger \chi_k + \chi_k^\dagger \phi_k)} \tag{77}$$

- A has eigenvalues which all have positive real parts
- $\phi = \phi_R + i\phi_I \in \mathbb{C}$; similar for χ

Dealing with the fermionic determinant

Ok fine. But still how to calculate the determinant? M is sparse but nonetheless a huge matrix.

We use the exact relation [GL]:

$$\frac{\pi^N}{\det A} \cdot e^{\sum_{kl} \chi_k^\dagger (A)_{kl}^{-1} \chi_l} = \int \prod_{k=1}^{N} d\phi_R d\phi_I \, e^{-\sum_{kl} \phi_k A_{kl} \phi_l + \sum_k (\phi_k^\dagger \chi_k + \chi_k^\dagger \phi_k)} \tag{77}$$

- A has eigenvalues which all have positive real parts
- $\phi = \phi_R + i\phi_I \in \mathbb{C}$; similar for χ

This is very useful since

1. if $A = MM^\dagger$ we can get M_{kl}^{-1} and $\mathrm{Tr}\, M^{-1}$ from a Gaussian integral (see stochastic noise integration, e.g., of disconnected diagrams)
2. if $A = (MM^\dagger)^{-1}$ we can express $\det MM^\dagger$ as a Gaussian-type integral (this is what we need for the HMC)

HMC for lattice QCD with two mass-degenerate fermions

Introducing a bosonic integral [k, l are multi-indices: $k = (x, \alpha, a)$]

$$\det MM^\dagger = \frac{1}{\pi^N} \int [D\phi_R][D\phi_I] \underbrace{e^{-\sum_{kl} \phi_k^\dagger [MM^\dagger]^{-1}_{kl} \phi_l}}_{= e^{-\eta_k \eta_k} \text{ where } \phi = M\eta} \tag{78}$$

With this the path integral for the expectation values becomes

$$\begin{aligned} \langle O \rangle &= \frac{1}{Z} \int [DU]\, \hat{O}[U]\, e^{-S_{eff}[U]} \\ &= \frac{1}{Z'} \int [DU][D\phi_R][D\phi_I]\, e^{-S_g + \phi^\dagger [MM^\dagger]^{-1} \phi}\, \hat{O}[U] \end{aligned} \tag{79a}$$

where

$$Z' = \int [DU][D\phi_R][D\phi_I]\, e^{-S_g + \phi^\dagger [MM^\dagger]^{-1} \phi} \tag{79b}$$

HMC for lattice QCD with two mass-degenerate fermions

That is, we have to generate a Markov chain of U's with the probability weight

$$P(U) = \int [D\phi_R][D\phi_I] e^{-S_g[U] + \phi^\dagger [M(U) M^\dagger(U)]^{-1} \phi} \tag{80}$$

Remember, we want to generate this Markov chain using the HMC algorithm, and for the step

$$I_P(\epsilon): \qquad P^a_{x\mu} \;\to\; P^a_{x\mu} - \epsilon F^a_{x\mu}[U]$$

we need the fermionic part to force F

$$F^a_{x\mu} = \underbrace{\left.\frac{\partial S_W[e^{i\omega} U]}{\partial \omega^a_{x\mu}}\right|_{\omega=0}}_{\frac{\beta}{3}\,\mathfrak{Im}\,\mathrm{Tr}\, T^a U_{x\mu} W_{x\mu}} + \underbrace{\left.\frac{\partial}{\partial \omega^a_{x\mu}} \ln\det M^2[e^{i\omega} U]\right|_{\omega=0}}_{??}$$

$$= \quad \ldots \quad \ldots \qquad + \int [D\phi_R][D\phi_I] \left.\frac{\partial}{\partial \omega^a_{x\mu}} e^{\phi^\dagger [M(e^{i\omega} U) M^\dagger(e^{i\omega} U)]^{-1} \phi}\right|_{\omega=0}$$

The contribution from the (two-flavour) fermionic part is ($\mathcal{M} \equiv MM^\dagger$)

$$\partial^a_{x\mu}\left(\phi^\dagger \mathcal{M}^{-1}\,\phi\right) = -\phi^\dagger \mathcal{M}^{-1}\left(\partial^a_{x\mu}\mathcal{M}\right)\mathcal{M}^{-1}\phi \tag{81}$$

$$= -(\mathcal{M}^{-1}\phi)^\dagger\left[(\partial^a_{x\mu}M)M^\dagger + M(\partial^a_{x\mu}M^\dagger)\right](\mathcal{M}^{-1}\phi) \tag{82}$$

where for the derivative of M we have (next slide)

$$(\partial^a_{x\mu}M)_{yz} = -\kappa\left[\delta_{xz}\delta_{y,z+\hat{\mu}}(1+\gamma_\mu)iT^a U_{x\mu} - \delta_{xy}\delta_{y,z-\hat{\mu}}(1-\gamma_\mu)iT^a U^\dagger_{x\mu}\right]. \tag{83}$$

Thus, we sample Gaussian-distributed complex ϕ^a_x at the beginning of each trajectory and solve for each leapfrog step $\mathcal{M}[U^{(i)}]\chi^{(i)} = \phi$

- using the CG-algorithm or any more advanced inversion algorithm.
- $\chi^{(i)}\left[(\partial^a_{x\mu}M)M^\dagger + M(\partial^a_{x\mu}M^\dagger)\right]\chi^{(i)}$ then gives the fermionic part to $F^a_{x\mu}$

Derivative of the Wilson-Dirac operator

$$\partial^a_{x\mu} f[U] \equiv \frac{\partial}{\partial \omega^a_{x\mu}} f[e^{i\omega} U]\Big|_{\omega=0}$$

$$\Rightarrow \partial^a_{x\mu} U_{z\nu} = \delta_{xz}\delta_{\mu\nu} \frac{\partial}{\partial \omega^a_{x\mu}} e^{+iT^b \omega^b_{x\mu}} U_{x\mu}\Big|_{\omega=0} = +\delta_{xz}\delta_{\mu\nu} iT^a U_{x\mu}$$

$$\Rightarrow \partial^a_{x\mu} U^\dagger_{z\nu} = \delta_{xz}\delta_{\mu\nu} \frac{\partial}{\partial \omega^a_{x\mu}} e^{-iT^b \omega^b_{x\mu}} U^\dagger_{x\mu}\Big|_{\omega=0} = -\delta_{xz}\delta_{\mu\nu} iT^a U^\dagger_{x\mu}$$

For the derivative of the Wilson-Dirac operator we thus get

$$\begin{aligned}
(\partial^a_{x\mu} M)_{yz} &= \partial^a_{x\mu} \left(\delta_{yz} - \kappa \sum_{\pm\nu} \delta_{y,z+\hat{\nu}} (1+\gamma_\nu) U_{z\nu} \right) \\
&= \partial^a_{x\mu} \left(\delta_{yz} - \kappa \sum_{\nu} \delta_{y,z+\hat{\nu}} (1+\gamma_\nu) U_{z\nu} + \delta_{y,z-\hat{\nu}} (1-\gamma_\nu) U^\dagger_{z-\nu,\nu} \right) \\
&= -\kappa \left[\delta_{xz}\delta_{y,z+\hat{\mu}} (1+\gamma_\mu) iT^a U_{x\mu} - \delta_{xy}\delta_{y,z-\hat{\mu}} (1-\gamma_\mu) iT^a U^\dagger_{x\mu} \right]
\end{aligned}$$

1. A trajectory is started choosing

   (a) Gaussian distributed momentum components $P^a_{x\mu} \in \mathbb{R}$. These define the set of initial momenta $P = \{P_{x\mu}\}$ conjugate to the link variables $U = \{U_{x\mu}\}$

   $$P_{x\mu} = \sum_a i P^a_{x\mu} T^a \quad \in \mathfrak{su}(3)$$

   conjugate to the link variables $U = \{U_{x\mu}\}$.

   (b) Gaussian distributed momentum complex numbers $\eta^{a,\alpha}_x \in \mathbb{C}$ for the *pseudo fermion fields* $\phi = M\eta$, needed for the fermionic force

2. (P, U) is evolved along a discretized trajectory applying n leapfrog steps of size $\Delta\tau = \tau/n$ (no evolution of ϕ is needed)

$$= \underbrace{T_P\left(\frac{\Delta\tau}{2}\right) T_U(\Delta\tau)}_{\text{final step}} \underbrace{T_P(\Delta\tau) T_U(\Delta\tau) \cdots T_P(\Delta\tau) T_U(\Delta\tau)}_{(n-1)\text{ full steps}} \underbrace{T_P\left(\frac{\Delta\tau}{2}\right)}_{\text{first half step}} (P, U)$$

2. ...
   The elementary operations for each leapfrog step are

$$I_P(\epsilon): \quad P^a_{x\mu} \rightarrow P^a_{x\mu} - \epsilon F^a_{x\mu}[U] \tag{84a}$$

$$I_U(\epsilon): \quad U_{x\mu} \rightarrow e^{i\epsilon P^a_{x\mu} T^a} U_{x\mu} \tag{84b}$$

   where the force consists of F (Wilson gauge + $N_f = 2$ Wilson fermions)

$$\text{gauge part:} \quad \frac{\beta}{3} \mathfrak{Im}\, \mathrm{Tr}\, T^a U_{x\mu} W_{x\mu} \tag{85}$$

$$\text{fermionic part:} \quad \chi^{(i)} \left[(\partial^a_{x\mu} M) M^\dagger + M(\partial^a_{x\mu} M^\dagger) \right] \chi^{(i)} \tag{86}$$

   with χ a solution of $\mathcal{M}[U^{(i)}]\chi^{(i)} = \phi$ we get from an inversion using the CG or any more advanced inverter.

3. After n leapfrog steps the new configuration (U') is either accepted or rejected if for a random $r \in [0, 1)$

$$r \leq \min\{1, e^{-\Delta H(U, P \rightarrow U', P')}\}$$

HMC diagnostics

HMC diagnostics

Acceptance rate

- Should be at $70\% - 85\%$
- If acceptance rate $= 0\%$, something is wrong with the MD evolution
- If acceptance rate $= 100\%$, configurations are changed only little (large autocorrelation lengths)
- Adjust $\Delta\tau$ to tune acceptance rate

Change of Hamiltonian

- ΔH will fluctuate above and below zero. Check that on average

$$\langle e^{-\Delta H} \rangle = 1$$

- Check also that ΔH scales with $\Delta\tau^2$

HMC diagnostics

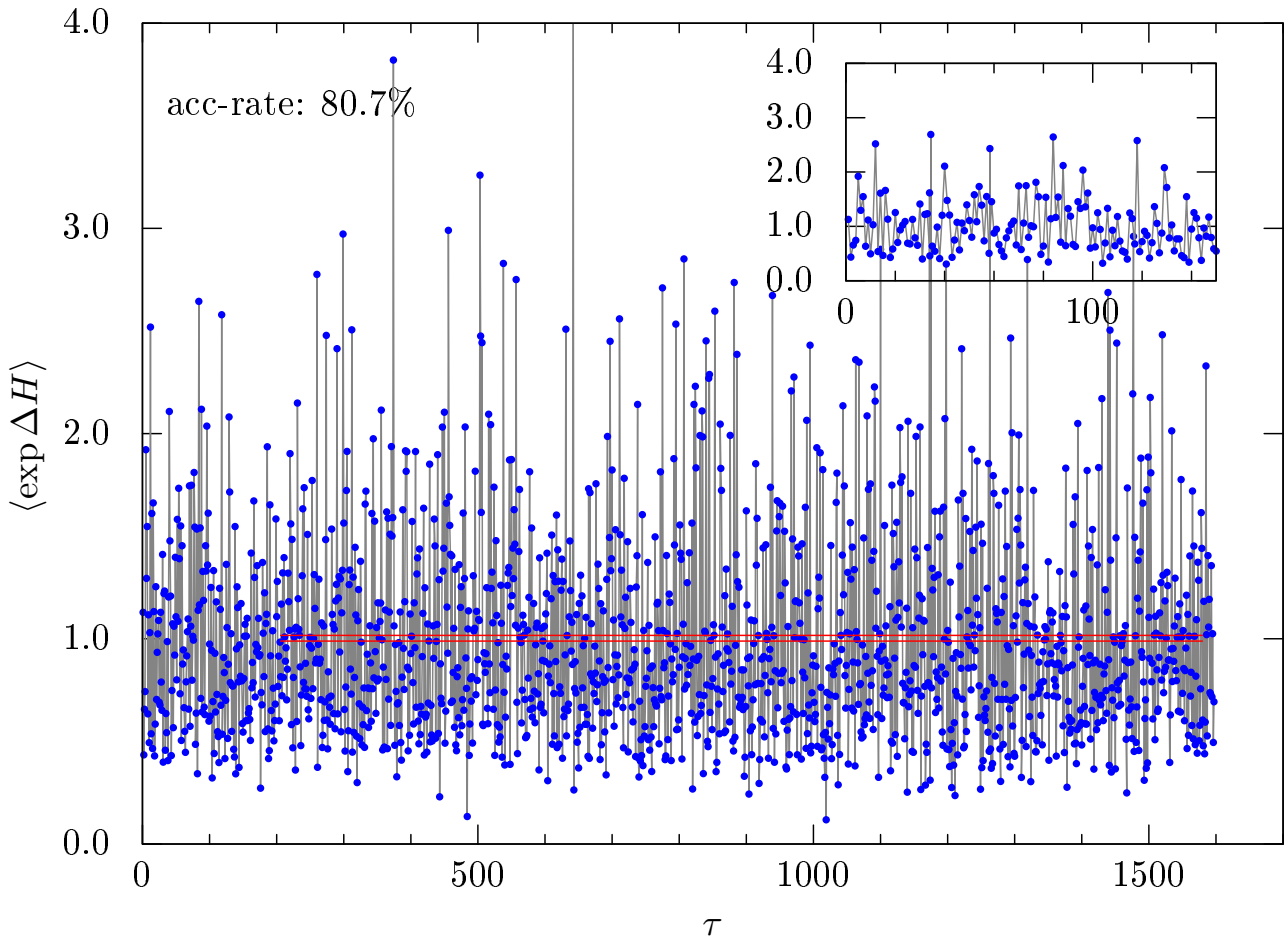


Figure: History of $\langle e^{-\Delta H} \rangle$

HMC diagnostics

Reversibility test

- To get the right probability distribution the MD evolution needs to be reversible
- Check reversibility by integrating forward and backward ($-\epsilon$)

Plaquette

- If one starts the thermalisation of new configurations, there will be a thermalisation phase
- During this phase the average plaquette $\langle P \rangle$ will increase or decrease
- Check that the HMC saturates at the same $\langle P \rangle$, starting from "cold", "hot" or another configuration

HMC diagnostics

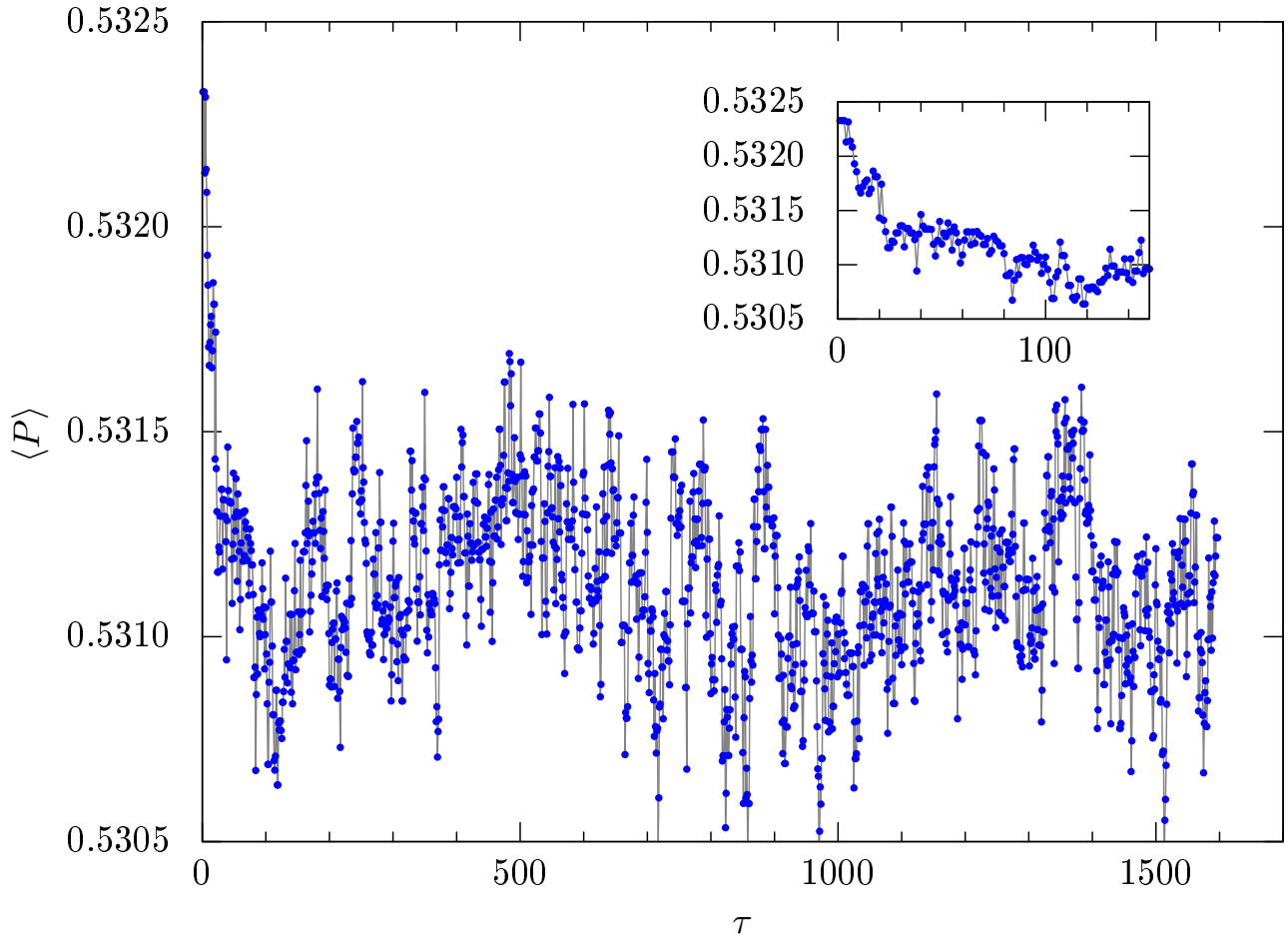


Figure: History of plaquette values

HMC diagnostics

Be aware:

- Autocorrelation times are observable-dependent.
- Monitoring the history of values of your observable and check if thermalisation has reached
- If the “moving average“ has saturated, thermalisation has been reached.
- Drop the unthermalised values and start “measuring” your observable
- Get an estimate for the autocorrelation time.

Inverting the fermion matrix using the CG

Inverting the Fermion matrix

There are various algorithms to invert the (sparse) fermion matrix, in our case the hermitian product $\mathcal{M} = MM^\dagger$ for two mass-degenerate fermions.

Popular solvers are

(Bi)CG: (Bi-)Conjugate gradient algorithm for (non-) symmetric matrices

GCR: Generalized Conjugate Residual for non-symmetric matrices

GMRES: Generalized Minimal REsidual method for non-symmetric matrices

These are so-called Krylov subspace methods.

Do not provide the inverse of a matrix, say the full matrix A^{-1} of A, but the action of A^{-1} on some given vector $\vec{b}$:

$$\vec{x} = A^{-1}\vec{b} \qquad \text{by solving the linear system:} \quad A\vec{x} = \vec{b}$$

This is fully sufficient, because we need $\mathcal{M}^{-1}\phi$ and not $\mathcal{M}^{-1}$ itself.

Inverting the Fermion matrix

In practise, pre-conditioned systems are solved. With the conjugate gradient (CG), for example, instead of solving $A\vec{x} = \vec{b}$, one solves

$$[A\widetilde{A}^{-1}]\,\vec{y} = \vec{b} \qquad \text{where} \quad \vec{y} = \widetilde{A}\vec{x}\,.$$

The level of efficiency increases with the level $\widetilde{A}^{-1}$

- is a good approximation of A^{-1}
- its application to $\vec{y}$ does not introduce significant additional cost

For an illustration, how these "inverters" work we look at the CG algorithm, but leave any pre-conditioning aside.

System of conjugate vectors

If for two vectors $\vec{u}$ and $\vec{v}$ it holds

$$0 = \langle \vec{u}, A\vec{v} \rangle \equiv \sum_{kl} u_k A_{kl} v_l \tag{87}$$

we say $\vec{u}$ and $\vec{v}$ are conjugate to each other with respect to the (symmetric) matrix A.

Supposed we had a set $\left\{ \vec{p}_i : \langle \vec{p}_i, A\vec{p}_k \rangle = 0, \ \forall i \neq k \in [1, n] \right\}$ of n mutually conjugate "direction" vectors $\vec{p}_k$, we could write the solution to $A\vec{x} = \vec{b}$ as

$$\vec{x} = \sum_{i}^{n} \alpha_i \vec{p}_i \qquad \text{where} \quad \alpha_i = \frac{\langle \vec{p}_i, \vec{b} \rangle}{\langle \vec{p}_i, A\vec{p}_i \rangle} \tag{88}$$

because

$$\langle \vec{p}_i, \vec{b} \rangle = \langle \vec{p}_i, A\vec{x} \rangle = \sum_{k} \alpha_k \langle \vec{p}_i, A\vec{p}_k \rangle = \alpha_i \langle \vec{p}_i, A\vec{p}_i \rangle$$

System of conjugate vectors

If for two vectors $\vec{u}$ and $\vec{v}$ it holds

$$0 = \langle \vec{u}, A\vec{v} \rangle \equiv \sum_{kl} u_k A_{kl} v_l \tag{87}$$

we say $\vec{u}$ and $\vec{v}$ are conjugate to each other with respect to the (symmetric) matrix A.

Supposed we had a set $\left\{ \vec{p}_i : \langle \vec{p}_i, A\vec{p}_k \rangle = 0, \ \forall i \neq k \in [1, n] \right\}$ of n mutually conjugate "direction" vectors $\vec{p}_k$, we could write the solution to $A\vec{x} = \vec{b}$ as

$$\vec{x} = \sum_i^n \alpha_i \vec{p}_i \qquad \text{where} \quad \alpha_i = \frac{\langle \vec{p}_i, \vec{b} \rangle}{\langle \vec{p}_i, A\vec{p}_i \rangle} \tag{88}$$

because

$$\langle \vec{p}_i, \vec{b} \rangle = \langle \vec{p}_i, A\vec{x} \rangle = \sum_k \alpha_k \langle \vec{p}_i, A\vec{p}_k \rangle = \alpha_i \langle \vec{p}_i, A\vec{p}_i \rangle$$

System solved!

The conjugate gradient algorithm

But how to get the vectors $\vec{p}_i$? Also, n may be very large?

It turns out, we do not need all $\vec{p}_i$'s. A subset is sufficient, if it gives a good approximation to $\vec{x}$. That is $|r| < \epsilon$ where

$$\vec{r} = \vec{b} - A\vec{x}' \quad \text{is the residual of} \quad \vec{x}' = \sum_{i \ll n} \alpha_i \vec{p}_i \tag{89}$$

Note

The full solution $\vec{x}$ to $A\vec{x} = b$ will minimize the function

$$f(\vec{x}) = \vec{x}^T A\vec{x} - \vec{b}\vec{x}, \tag{90}$$

because its Gradient

$$\nabla f(\vec{x}) \equiv A\vec{x} - b = 0$$

The conjugate gradient algorithm

For a symmetric matrix A, the CG algorithm tries to find all the relevant direction vectors $\vec{p}_i$ which are conjugate to each other

$$\left\{\vec{p}_i : \langle \vec{p}_i, A\vec{p}_k \rangle = 0,\ \forall i \neq k \in [1, n]\right\}$$

The solution (within the desired precision) is then

$$\vec{x} = \sum_i^n \alpha_i \vec{p}_i \qquad \text{where} \quad \alpha_i = \frac{\langle \vec{p}_i, \vec{b} \rangle}{\langle \vec{p}_i, A\vec{p}_i \rangle} \tag{91}$$

The conjugate gradient algorithm

1. Initial guess: $\vec{x}^{(0)} = 0$
2. Initial residual: $\vec{r}^{(0)} = \vec{b} - A\vec{x}^{(0)}$ "minus the Gradient"
3. Initial direction: $\vec{p}_0 = \vec{r}^{(0)}$ "in direction minus the Gradient"
4. Next (better) solution $\vec{x}^{(1)} = \vec{x}^{(0)} + \alpha_0 \vec{p}_0$ where

$$\alpha_0 = \frac{\langle p_0, \vec{b} \rangle}{\langle p_0, A\vec{p}_0 \rangle}$$

All the remaining $\vec{p}_{i>0}$ are now chosen conjugate to the gradient $r^{(i>0)}$ and to each other.

For this, we use the Gram-Schmidt orthogonalization, where $\langle \vec{p}_k, A\vec{p}_i \rangle$ reads $\langle \vec{u}, \vec{v} \rangle$.

Reminder on the Gram-Schmidt orthogonalization/orthonormalization

Goal: sequence of orthonormal vectors (normalized and orthogonal)

$$\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n \qquad |e_i| = 1,\ \vec{e}_i \vec{e}_k = \delta_{ik} \tag{92}$$

Gram-Schmidt orthonormalization

Starts with a set of n vectors: $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$

orthogonal	orthonormal
$\vec{u}_1 = \vec{v}_1$	$\vec{e}_1 = \vec{u}_1/\lvert\vec{u}_1\rvert$
$\vec{u}_2 = \vec{v}_2 - \mathcal{P}_{\vec{u}_1}(\vec{v}_2)$	$\vec{e}_2 = \vec{u}_2/\lvert\vec{u}_2\rvert$
$\vec{u}_3 = \vec{v}_3 - \mathcal{P}_{\vec{u}_1}(\vec{v}_3) - \mathcal{P}_{\vec{u}_2}(\vec{v}_3)$	$\vec{e}_3 = \vec{u}_3/\lvert\vec{u}_3\rvert$
$\vdots \quad \vdots$	$\vdots$
$\vec{u}_n = \vec{v}_n - \sum_{j<n} \mathcal{P}_{\vec{u}_j}(\vec{v}_n)$	$\vec{e}_n = \vec{u}_n/\lvert\vec{u}_n\rvert$

where

$$\mathcal{P}_{\vec{u}}(\vec{v}) \equiv \frac{\langle \vec{u}, \vec{v} \rangle}{\langle \vec{u}, \vec{u} \rangle} \tag{93}$$

The conjugate gradient algorithm

1. Residual of current solution $\vec{x}^{(i)}$: $\vec{r}^{(i)} = \vec{b} - A\vec{x}^{(i)}$
2. Choose next direction orthogonal to this and the other $\vec{p}_{k<i}$

$$\vec{p}_i = \vec{r}^{(i)} - \sum_{k<i} \frac{\langle \vec{p}_k, A\vec{r}^{(i)} \rangle}{\langle \vec{p}_k, A\vec{p}_k \rangle} \vec{p}_k \quad \text{(Gram-Schmidt)}$$

3. Next $\vec{x}^{(i+1)} = \vec{x}^{(i)} + \alpha_i \vec{p}_i$ where

$$\alpha_i = \frac{\langle p^{(i)}, \vec{b} \rangle}{\langle p_i, A\vec{p}_i \rangle} = \frac{\langle p_i, \vec{r}^{(i-1)} - A\vec{x}^{(i-1)} \rangle}{\langle p_i, A\vec{p}_i \rangle} = \frac{\langle p_i, \vec{r}^{(i-1)} \rangle}{\langle p_i, A\vec{p}_i \rangle}$$

4. The algorithm stops if $|\vec{r^{(i)}}|^2 < \epsilon$

Note $\vec{r}^{(i)}$ points into the direction of steepest descent of $f(\vec{x}^{(i)})$. Going into that direction would be the *gradient descent* method.

The conjugate gradient algorithm

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$\mathbf{p}_0 := \mathbf{r}_0$ (first direction $=-$ Gradient)

$k := 0$

repeat

$$\alpha_k := \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}$$

$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$

if r_{k+1} is sufficiently small then exit loop

$$\beta_k := \frac{\mathbf{r}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1}}{\mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k}$$

$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$

$k := k + 1$

end repeat

The result is $\mathbf{x}_{k+1}$

HMC: Adding another flavor

"Adding flavor"

The standard HMC algorithm uses the fact that $M^{(u)} = M^{(d)}$ and so $\det MM^\dagger$ is positive and real.

Good approximation of two-flavor QCD, because $m_u \approx m_d$

What about adding heavier quark flavors to the simulation, which do not come in (almost mass-degenerate) pairs? That is, how do we simulate the $+1$ quark in current $N_f = 2 + 1$ simulations?

"Adding flavor"

The standard HMC algorithm uses the fact that $M^{(u)} = M^{(d)}$ and so $\det MM^\dagger$ is positive and real.

Good approximation of two-flavor QCD, because $m_u \approx m_d$

What about adding heavier quark flavors to the simulation, which do not come in (almost mass-degenerate) pairs? That is, how do we simulate the $+1$ quark in current $N_f = 2 + 1$ simulations?

This is more difficult, but we are lucky because for the relevant regime $\det M^{(q \geq s)}$ is positive and can be replaced in the MD evolution by a non-negative hermitean operator.

Two popular approximations are:

1. Rational approximation (see, e.g., lecture by Lüscher [L2010])
2. Polynomial approximation (see, e.g., book by Gattringer and Lang [GL2010]).

Polynomial Approximation

Here we follow [GL2010]: If $\det M$ is positive one can approximate it by a non-negative hermitean operator

$$M^{-1} = TT^{\dagger} \tag{94}$$

and correct this approximation during the accept/reject step.

Polynomial approximation $(z_k = 1 - e^{2\pi ik/(2n-1)})$:

$$M^{-1} \approx P_{2n} \equiv \prod_{k=1}^{n} (M - z_{2k-1})(M - z_{2k-1}^{*}) \tag{95}$$

Due to γ_5 Hermiticity $\det[M - z_k^{*}] = \det[\gamma_5 M^{\dagger} \gamma_5 - z_k^{*}] = \det[M - z_k]^{*}$, and so

$$P_{2n} = T_n T_n^{\dagger} \qquad \text{where} \quad T = \prod_{k=1}^{n} (M - z_{2k-1}) \tag{96}$$

Polynomial Approximation

$P_{2n} = T_n T_n^\dagger$ is just an approximation, because the correct determinant is

$$\det M = C / \det[T_n^\dagger T_n] \qquad \text{with} \quad C = \det[M T_n^\dagger T_n] \tag{97}$$

However, it is sufficient to use

$$S_f = -\phi^\dagger T_n^\dagger T_n \phi \tag{98}$$

for the MD evolution (setp 2 of HMC) and then correct for C once in the Metropolis accept-reject step at the end of the trajectory. This makes it exact again.

HMC: Improvements

Improving the efficiency of the HMC

Most expensive part of HMC is the fermionic force in particular solving $\mathcal{M}[U^{(i)}]\chi^{(i)} = \phi$ for each leapfrog step of a trajectory

Typical targets for improvements of the HMC in the recent years:

- Reducing the number of if $\mathcal{I}_P$ steps, in particular those where an inversion of the fermion matrix is needed (larger $\Delta\tau$, multiple-time scale)
- Reduce the cpu-time for a single inversion (preconditioning, chronological inverter)

It is beyond the scope of this lecture to cover all the improvements achieved over the last years. We will just highlight a few standard tricks.

Improving the efficiency of the HMC

Achievements over the last years

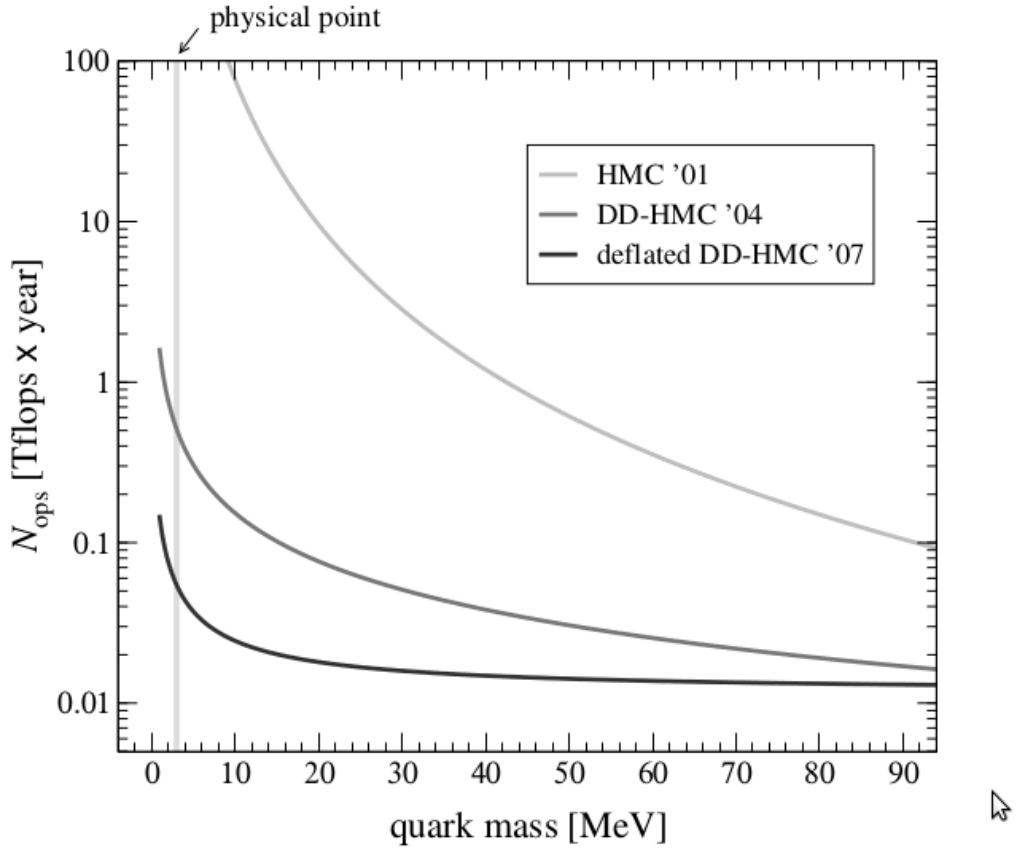


Figure: From [L2010]: *"Number of floating-point operations required for the generation of 100 statistically independent gauge-field configurations in $O(a)$-improved two-flavour QCD on a 64×32^3 lattice with spacing $a = 0.08$ fm. The top curve (Ukawa, 2002) represents the status reported at the memorable Berlin lattice conference in 2001, the middle one was obtained a few years later, using the so-called domain-decomposed HMC algorithm (Lüscher, 2005; Del Debbio et al., 2007), and the lowest curve shows the performance of a recently developed deflated version of the latter (Lüscher, 2007b)"*

Even-Odd Preconditioning

Standard trick for Wilson-Dirac operator

$$M_{xy}[U] = \left(\delta_{xy} - \kappa \sum_{\pm\mu} \delta_{y,x+\hat{\mu}} (1+\gamma_\mu) U_{x\mu} \right)$$

Even-Odd Preconditioning

Standard trick for Wilson-Dirac operator

$$M = \begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & -M_{eo}M_{oo}^{-1} \\ 0 & 1 \end{pmatrix}}_{U} \begin{pmatrix} \tilde{M}_{ee} & 0 \\ 0 & M_{oo} \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 \\ -M_{oo}^{-1}M_{oe} & 1 \end{pmatrix}}_{L}$$

where

$$\tilde{M}_{ee} \equiv M_{ee} - M_{eo}M_{oo}^{-1}M_{oe} \tag{99}$$

Determinant

$$\det M = \det \tilde{M}_{ee} \det M_{oo} \tag{100}$$

Need to invert only $\det\left(\tilde{M}_{ee}\tilde{M}_{ee}^{\dagger}\right)$, the inverse of M_{oo} is simple.

Advantages:

- Condition number of $\tilde{M}$ typically less then half that of M
- less memory because ϕ_e is only half as long.

Multiple step size integration

Force of MD evolution: $F = F_g + F_f$. Empirically one finds $F_g \gg F_f$. The most expensive is F_f, can accelerate MD evolution by introducing different step sizes

$$\Delta\tau_g = \frac{\tau}{N_g N_f} \qquad \text{and} \qquad \Delta\tau_f = \frac{\tau}{N_f} \tag{101}$$

and split the MD evolution ($n = N_g N_f$)

$$(P', U') = \underbrace{\left[I_P\left(\frac{\Delta\tau}{2}\right) I_U(\Delta\tau)\, I_P\left(\frac{\Delta\tau}{2}\right)\right]^n}_{I(\Delta\tau, n)} (P, U) \tag{102}$$

Multiple step size integration

Force of MD evolution: $F = F_g + F_f$. Empirically one finds $F_g \gg F_f$. The most expensive is F_f, can accelerate MD evolution by introducing different step sizes

$$\Delta\tau_g = \frac{\tau}{N_g N_f} \qquad \text{and} \qquad \Delta\tau_f = \frac{\tau}{N_f} \tag{101}$$

and split the MD evolution ($n = N_g N_f$) into two parts

$$(P', U') = \left[I_P^f \left(\frac{\Delta\tau_f}{2} \right) I_g(\Delta\tau_g)\, I_P^f \left(\frac{\Delta\tau_f}{2} \right) \right]^{N_f} (P, U) \tag{103}$$

where $I_P^f(\epsilon) : P \to P - \epsilon F_f[U]$, $I_P^g(\epsilon) : P \to P - \epsilon F_g[U]$ and

$$I_g(\Delta\tau_g) = \left[I_P^g \left(\frac{\Delta\tau_g}{2} \right) I_U(\Delta\tau_g)\, I_P^g \left(\frac{\Delta\tau_g}{2} \right) \right]^{N_g} \tag{104}$$

Much less computing time because the fermionic part is not needed in I_P^g

Frequency splitting of determinant

Popular factorizations

1. RHMC [Kennedy et al. '99, Clark et al. '07]

$$\det M^2 = \prod_i^N \det(M^2)^{1/n} \tag{105}$$

2. Hasenbusch-trick [Hasenbusch et al 2001,'03] (mass preconditioning)

$$\det M^2 = \det \frac{M^2}{M^2 + \mu^2} \det M^2 + \mu^2 \tag{106}$$

3. Domain decomposition of lattice into blocks (Lüscher 2004)

$$\det M = \det M_{blocks} \det M_{Rest} \tag{107}$$

Represent the different parts by separate pseudo fermions: ϕ_1, ϕ_2,... and do the MD evolution of fermionic force again with a multiple-time scale integrator.

Thank you for your attention!