Simulation of harmonic oscillator through path integral and lattice QCD

Alessandro Nervo

Università di Bologna

22 dicembre, 2024

Table of contents

Path Integral on the Harmonic Oscillator

Path Integral

In one-dimensional quantum mechanics we have:

$$\langle x_f | \, e^{-H(t_f - t_i)} \, | x_i \rangle = \int \mathcal{D}\left[x(t)\right] e^{-S[x]}$$

With the action:

$$S[x] = \int_{t_i}^{t_f} dt \left(\frac{m\dot{x}(t)^2}{2} + V\left(x(t)\right) \right)$$

We can see the space and time as a discretized lattice, the action becomes:

$$S[x] = \sum \left(\frac{m}{2a}(x_{j+1} - x_j)^2 + aV(x_j) \right)$$

Where $a = \frac{t_f - t_i}{N}$ and N is the lattice size.
The approximation for the derivative is the easiest one but can be improved.

Correlation function and Energy gap

The vacuum expectation value can be written as the propagator

$$\langle x(t_1)x(t_2)\rangle = \frac{\int \mathcal{D}\left[x(t)\right] x(t_2)x(t_1)e^{-S[x]}}{\int \mathcal{D}\left[x(t)\right] e^{-S[x]}}$$

Using euclidean time in Schrodinger picture, after some manipulation we get to

$$\langle x(t_1)x(t_2)\rangle = \langle E_0| xe^{-(H-E_0)t}x |E_0\rangle$$

Where $T = t_f - t_i$ and $t = t_2 - t_1$ and $T >> t$.
With this assumptions, if the ground state dominates we eventually write:

$$G(t) = |\langle E_0| x |E_1\rangle|^2 e^{-(E_1-E_0)t}$$

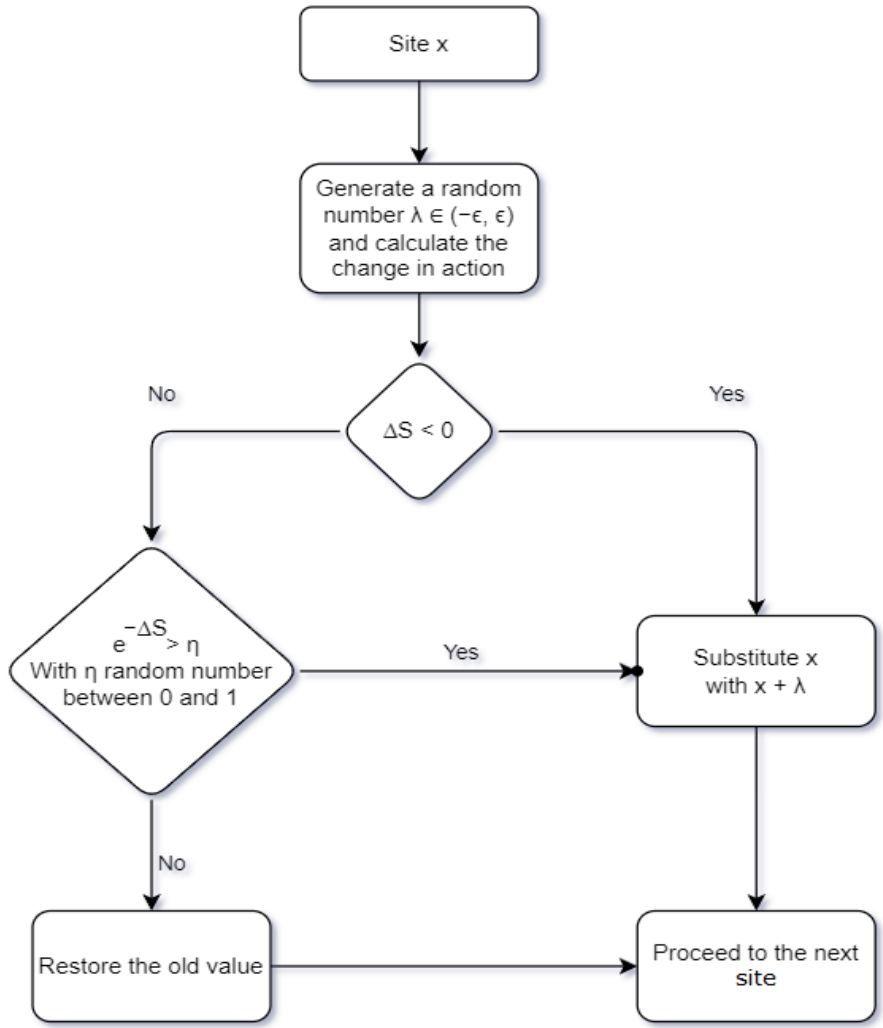
Where $G(t) = \langle x(t_1)x(t_2)\rangle$. From that we can extract the first energy gap:

$$E_1 - E_0 = \frac{1}{a} log\left(\frac{G(t)}{G(t+a)}\right)$$

Metropolis

We can study the sistem with the Monte Carlo procedure, to evolve it we use the simplest one, the Metropois algoritm:

- We have to assume ergodicity and detailed balance
- Typically ϵ should be tuned so that $40\% - 60\%$ of the sweeps are accepted.
- After some sweeps we can take the measurment of correlation functions $G_n = \frac{1}{N} \sum_j \langle x_{j+n} x_j \rangle$
- We need to skip some measurment because of the correlation $N_{corr} \propto \frac{1}{a^2}$

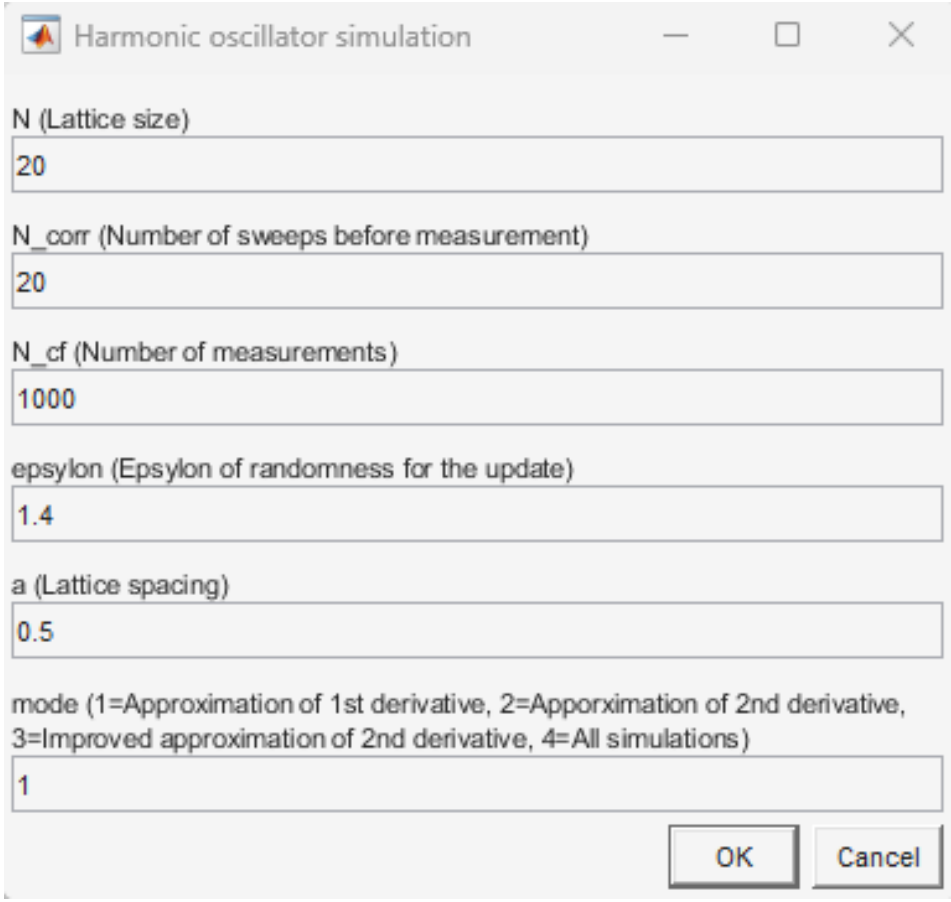


Code's functions

```
function [acceptance_ratio,elapsed_time]=main(setup) ...

function [d_E,acceptance_ratio]=simulation(therm,N,epsylon,a,mode,N_corr,N_cf) ...

function [teo,plotted]=plotting(d_E,N,a) ...

function dE=delta_E(G1,a) ...

function [x,partial,total]=update_x(N,x,epsylon,a,partial,total,mode) ...

function dS=delta_S(x,x_old,k,a,N,mode) ...

function G = compute_G(x,n,N) ...
```

Variables discussion

- $T = \frac{N}{a}$ is the euclidean time and has to be as large as possible
- $N_{corr} \propto \frac{1}{a^2}$
- N_{cf} has to be large in order to reduce statistical error
- ϵ should be tuned so that $40\% - 60\%$ of the sweep are accepted
- We start our measurments after $N_{therm} = N_{corr} * 5$ thermalization steps



Code's functions (detail)

Simulation

```
for j = 1:therm
    [x,par,tot]=update_x(N,x,epsylon,a,par,tot,mode);
end
% Simulation and measurement
for alpha= 1:N_cf
     for j=1 : N_corr
        [x,par,tot]=update_x(N,x,epsylon,a,par,tot,mode);
     end

     for k=0:N
        G(alpha,k+1)=compute_G(x,k,N);
     end
end

avg_G=sum(G(:,:))/N_cf;          % Averege of G
acceptance_ratio = 1- par/tot;
d_E= delta_E(avg_G,a);
```

Update lattice

```
x_old=x;

% Metropolis algoritm
for k = 1:N
    x(k)=x_old(k) + epsylon*(2*rand-1);
    dS=delta_S(x,x_old,k,a,N,mode);
    if dS>0 && exp(-dS)<rand
        x(k)=x_old(k);
        partial=partial+1;
    end
    total=total+1;
end
```

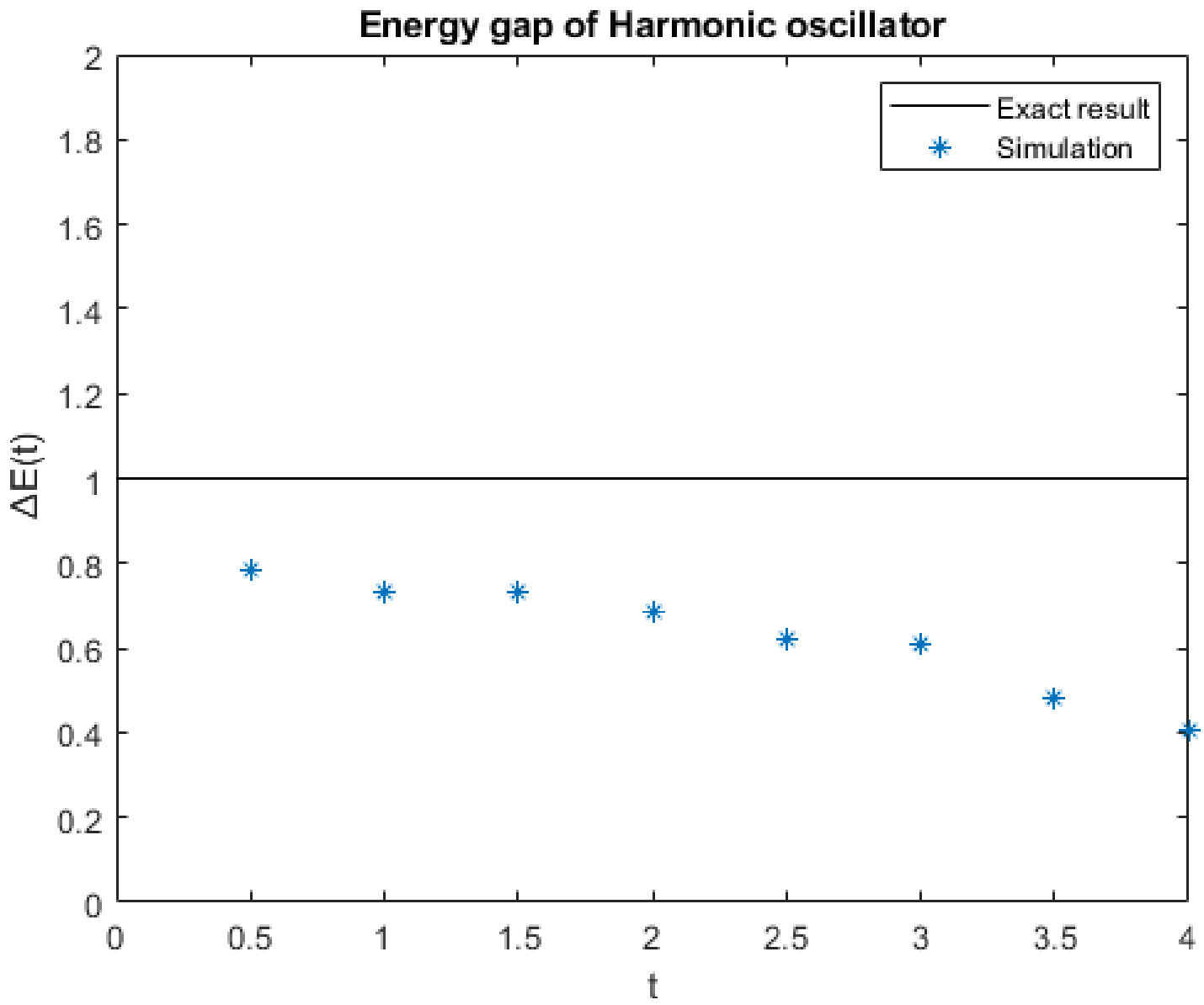
Compute energy gap

```
G2=G1;
G1(end)=[];
G2(1)=[];
dE=log(abs(G1./G2))/a;
```

Compute correlation function

```
A = eye(N);
B = circshift(A, [0, n]);
G = x' * B * x/ N;
```

Result 1


Energy gap of Harmonic oscillator
Exact result
Simulation
ΔE(t)
t
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
0.5
1
1.5
2
2.5
3
3.5
4

Derivative improvement

By integrating by part we can rewrite our action:

$$S[x] = \int_{t_i}^{t_f} dt \left(-\frac{m}{2} x(t)\ddot{x}(t) + V\left(x(t)\right) \right)$$

The second derivative $\ddot{x}(t)$ can be discrteized as:

$$\ddot{x}(t_j) \to \Delta^{(2)} x_j = \frac{x_{j+1} - 2x_j + x_{j-1}}{a^2}$$

We can improve the discretization by using the corrected approximation:

$$\Delta^{(2)} x_j \to \left(\Delta^{(2)} - \frac{a^2}{12} \left(\Delta^{(2)} \right)^2 \right) x_j$$

Thus the full lagrangian for harmonic oscillator look like this:

$$\mathcal{L} = a x_j \left(\Delta^{(2)} + \frac{m\omega^2}{2} \right) x_j$$

Where in ω we need to add a correction caused by the finite lattice spacing

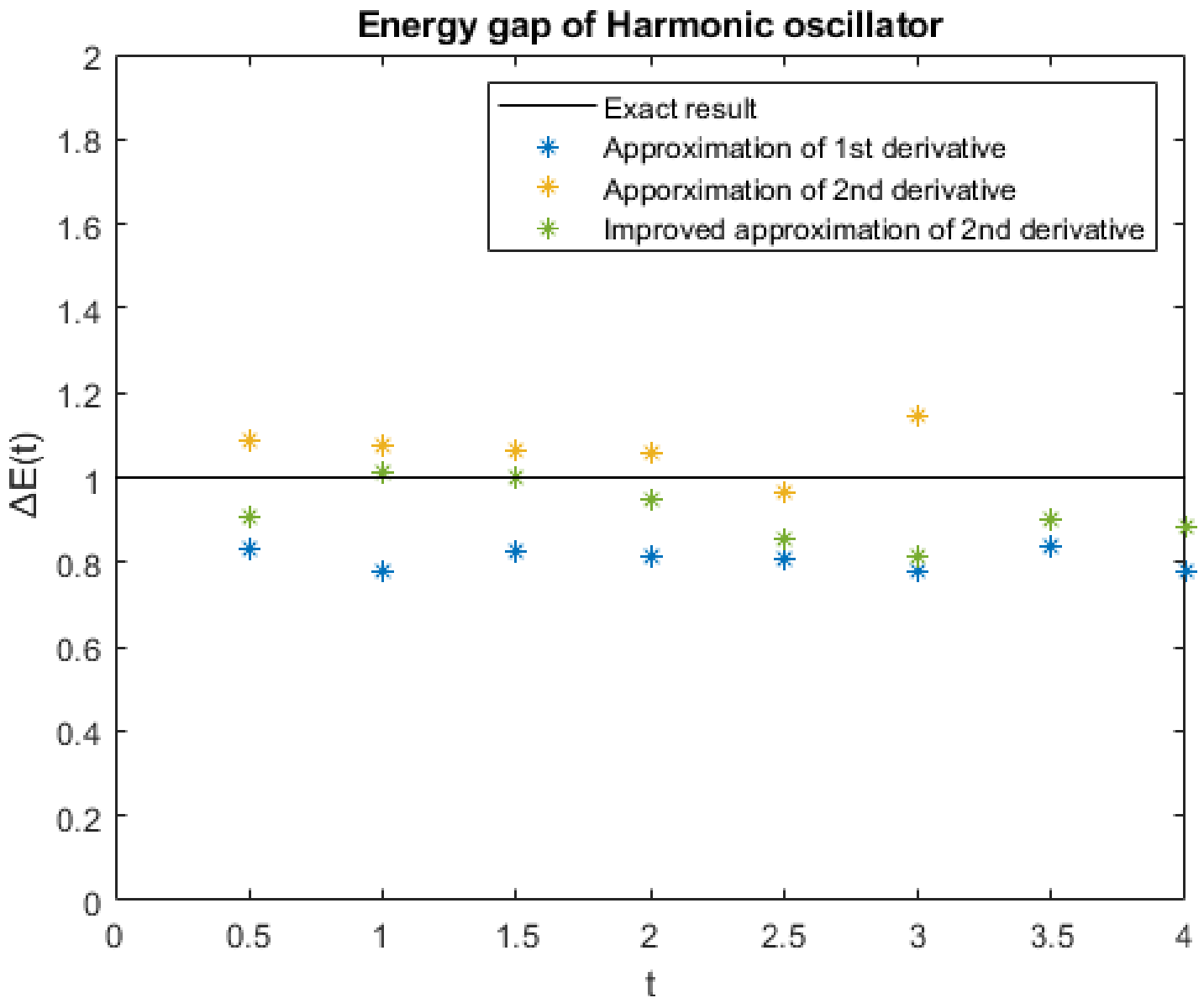
$$\omega^2 = \omega_0^2 \left(1 - \frac{(a\omega_0)^2}{12} + \mathcal{O}\left((a\omega_0)^4 \right) \right)$$

Improved derivative implementation

Function to calculate the delta in action

```
if mode==1
    S       = 0.5*a*x(k)*x(k)         + x(k)*(x(k) - x(mod(k-2,N)+1) - x(mod(k,N)+1))/a;
    S_old   = 0.5*a*x_old(k)*x_old(k) + x_old(k)*(x_old(k) - x_old(mod(k-2,N)+1) - x_old(mod(k,N)+1))/a;

elseif mode==2
    A=eye(N);
    D2=(eye(N,N)*(-2)+circshift(eye(N,N), [0, 1])+circshift(eye(N,N), [0, -1]))/(a^2);
    z1=A(:,k);
    z2=z1+A(:,mod(k-2,N)+1)+A(:,mod(k,N)+1);

    S     = (-0.5*a)*(z2.*x)'*(D2*x)         + (a*0.5)*(z1.*x)'*x;
    S_old = (-0.5*a)*(z2.*x_old)'*(D2*x_old) + (a*0.5)*(z1.*x_old)'*x_old;

elseif mode==3
    A=eye(N);
    D2=(eye(N,N)*(-2)+circshift(eye(N,N), [0, 1])+circshift(eye(N,N), [0, -1]))/(a^2);
    z1=A(:,k);
    z2=z1+A(:,mod(k-2,N)+1)+A(:,mod(k,N)+1);
    z3=z2+A(:,mod(k-3,N)+1)+A(:,mod(k+1,N)+1);

    S     = (-0.5*a)*(z3.*x)'*(D2-(a^2/12)*D2^2)*x         + (a*0.5)*(z1.*x)'*x*(1+a^2/12);
    S_old = (-0.5*a)*(z3.*x_old)'*(D2-(a^2/12)*D2^2)*x_old + (a*0.5)*(z1.*x_old)'*x_old*(1+a^2/12);

end
dS = S-S_old;
```

Result 2


Energy gap of Harmonic oscillator
Exact result
Approximation of 1st derivative
Apporximation of 2nd derivative
Improved approximation of 2nd derivative
ΔE(t)
t
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
0.5
1
1.5
2
2.5
3
3.5
4

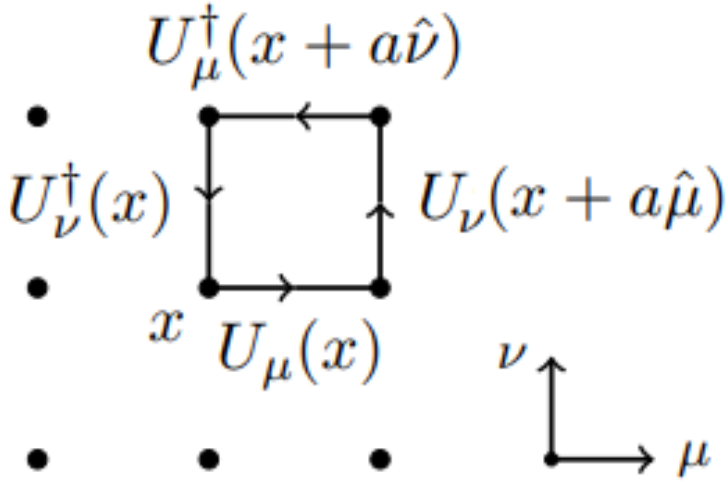
Lattice QCD

Lattice QCD

Instead of the one-dimensional space coordinates $x(t)$ we have a four-dimensional lattice $x = (\vec{x}, t)$, where fields live.
The field $A_\mu(x)$ represent the gluon at the site x, but since we can't formulate a gauge invariant discrtized version of QCD, we use the link variables instead:

$$U_\mu(x) = \mathcal{P}exp\left(-i\int_x^{x+a\hat{\mu}} gA_\mu dy\right)$$

Where $\mathcal{P}$ path-orders the integral and $\hat{\mu}$ is the direction of the link.
Wilson lines are $SU(3)$ matrices, and since they transform in a gauge convariant way, when we loop path, we get a gauge invariant quantity, for example, the plaquette operator:



$$P_{\mu\nu} = \frac{1}{3}Re\ Tr\left(U_\mu(x)U_\nu(x+a\hat{\mu})U_\mu^\dagger(x+a\hat{\nu})U_\nu^\dagger(x)\right)$$

Wilson loops and QCD action

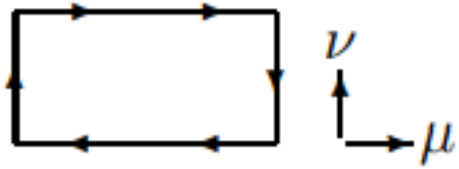
The plaquette operator $P_{\mu\nu}$ can be expanded around 1:

$$P_{\mu\nu} = 1 - \frac{a^4}{6} Tr \left(gF_{\mu\nu}(x_0)\right)^2 + \mathcal{O}(a^6)$$

So we can recover the QCD action with this operator

$$S = \int dx^4 \frac{1}{2} \sum_{\mu,\nu} TrF^2_{\mu\nu}(x) = \beta \sum_x \sum_{\mu>\nu} (1 - P_{\mu\nu}(x))$$

With $\beta = \frac{6}{g^2}$ This is true only at $\mathcal{O}(a^6)$, if we want to cancel out higher-order terms we can introduce the rectangle operator $R_{\mu\nu}$ and the improved action becomes:

$R_{\mu\nu} = \frac{1}{3}\,\mathrm{Re\ Tr}$ 

$$S_{imp} = -\beta \sum_x \sum_{\mu>\nu} \left(\frac{5}{3} P_{\mu\nu} - \frac{1}{12} \left(R_{\mu\nu} + R_{\nu\mu}\right) \right)$$

Tadpole improvment and evolution

In order to deal with the perturbations in the quantum theory we have to do an improvment to the action, that is the Tadpole improvment.
We renormalize the action by dividing by the mean link value u_0.

$$U_\mu(x) \to \frac{U_\mu(x)}{u_0}$$

This will change the action to:

$$S_{imp} = -\beta \sum_x \sum_{\mu>\nu} \left(\frac{5}{3u_0^4} P_{\mu\nu} - \frac{1}{12u_0^6} \left(R_{\mu\nu} + R_{\nu\mu} \right) \right)$$

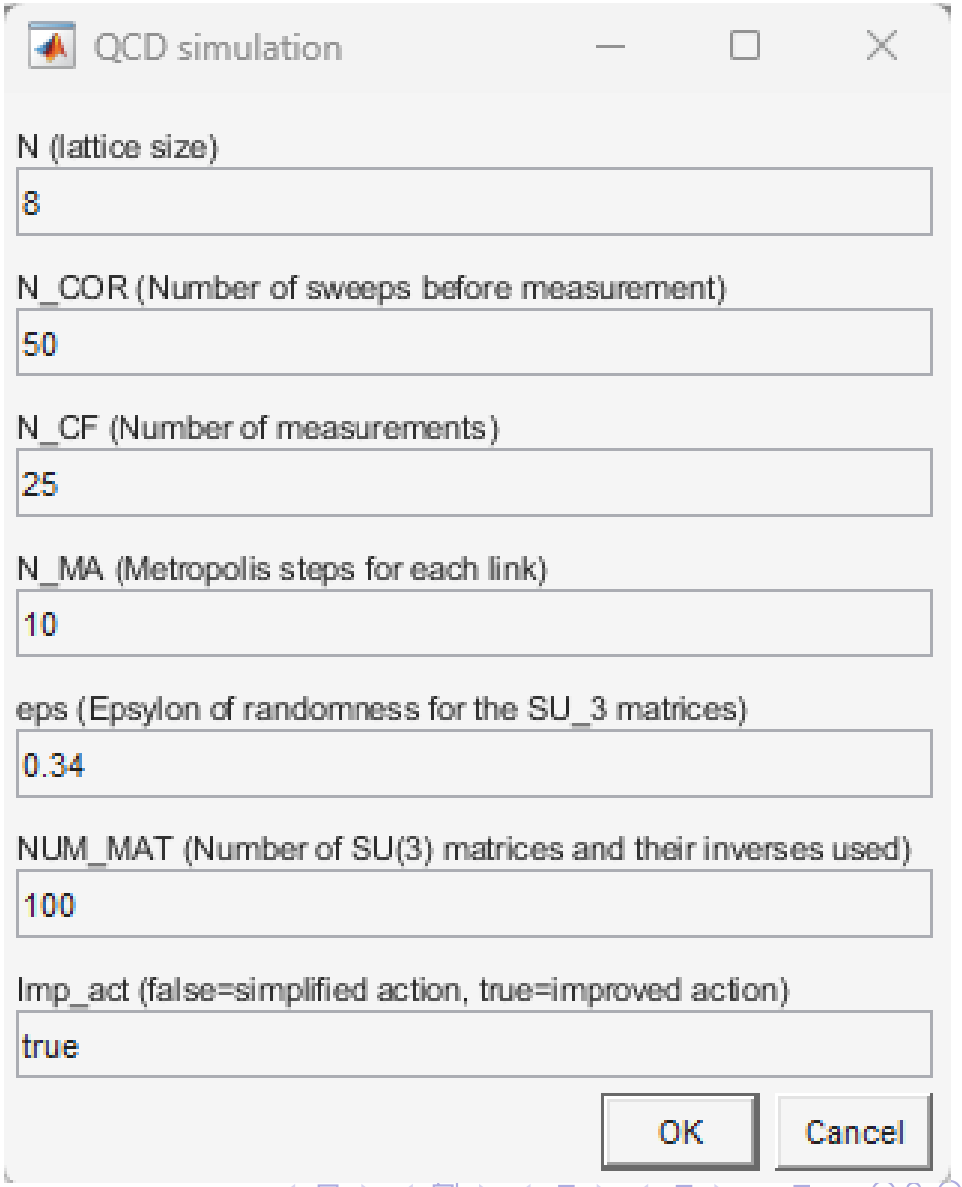
To evolve the system we take all the the link variables $U_\mu(x)$, that are initializated to the identity, and multiply them by a random $SU(3)$ matrix.
Since Generating a random $SU(3)$ matrix is heavy, we generate a list of matrices at the start of the simulation and randomly select one of those each time.

Code's functions

```
function [tot_avg1_link,err_avg1_link,tot_avg2_link,err_avg2_link,acceptance_ratio,elapsed_time]=main(setup) ...
function SU3_LIST=create_SU3_list(NUM_MAT,dim,eps_herm) ...
function [L,par,tot]=sweep(L,N_ma,N,SU3_LIST,BETA,par,tot,NUM_MAT,dim,c_1,c_2,imp_act) ...
function L=create_lattice(N,dim) ...
function SU_N=generate_suN(N, eps_herm) ...
function H=generate_hermitian(N, eps_herm) ...
function [staple,staple_conj]=plaq(L,x,y,z,t,n,N) ...
function z=m(x,N,change) ...
function delta_S=dS(L,x,y,z,t,n,staple,staple_conj,staple_rect,staple_rect_conj,M,BETA,dim,c_1,c_2,imp_act) ...
function plaq_size=measure_plaq(L,plaq_size,N) ...
function rect2_size=measure_rect2(L,rect2_size,N) ...
function [staple_rect,staple_rect_conj]=rect(L,x,y,z,t,n,N) ...
```

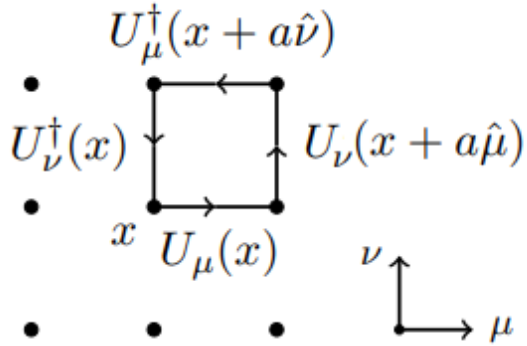
Variables discussion

- The lattice has a total of $N^4 * 4$ indipendent variables
- 50 correlation steps are enough to do independent measurments
- N_{MA} It's to let the link variable reach an equilibrium with its neighbors before moving on to the next link
- eps should be tuned so that $40\% - 60\%$ of the sweeps are accepted
- The products of $SU(3)$ matrices should cover the entire space. The inverse of each matrix should also be included in the set.
- The improved action will more than double the simulation time



Sweep

```
staple_rect=0;
staple_rect_conj=0;
   for x=1:N
       for y=1:N
           for z=1:N
               for t=1:N
                   for n=1:4
                       [staple,staple_conj]=plaq(L,x,y,z,t,n,N);
                       if imp_act
                           [staple_rect,staple_rect_conj]=rect(L,x,y,z,t,n,N);
                       end

                       for i=1:N_ma
                           M=SU3_LIST{randi(NUM_MAT*2)};
                           delta_S = dS(L,x,y,z,t,n,staple,staple_conj,staple_rect,staple_rect_conj,M,BETA,dim,c_1,c_2,imp_act);

                           if delta_S < 0 || rand < exp(-delta_S)
                               L(:,:,x,y,z,t,n) = M*L(:,:,x,y,z,t,n);
                               par=par+1;
                           end
                           tot=tot+1;
                       end
                   end
               end
           end
       end
   end
```



Plaquette staple

```
M=10;
a=zeros(1,M);
a(n)=1;

staple=0;
staple_conj=0;

for i=1:(4-n)
    % Staple construction for (n=1) xy,xz,xt
    %                         (n=2) yz,yt
    %                         (n=3) zt
    staple = staple + ...
    L(:,:,m(x,N,a(1)),m(y,N,a(2)),m(z,N,a(3)),t,n+i)*...
    L(:,:,x,m(y,N,a(m(1,M,-i+1))),m(z,N,a(m(2,M,-i+1))),m(t,N,a(m(3,M,-i+1))),n)'*...
    L(:,:,x,y,z,t,n+i)';

    staple_conj= staple_conj +...
    L(:,:,x,m(y,N,-a(m(1,M,-i+1))),m(z,N,-a(m(2,M,-i+1))),m(t,N,-a(m(3,M,-i+1))),n+i)'*...
    L(:,:,x,m(y,N,-a(m(1,M,-i+1))),m(z,N,-a(m(2,M,-i+1))),m(t,N,-a(m(3,M,-i+1))),n)*...
    L(:,:,m(x,N,a(1)),m(y,N,a(2)-a(m(1,M,-i+1))),m(z,N,a(3)-a(m(2,M,-i+1))),m(t,N,-a(m(3,M,-i+1))),n+i);

end

for i=1:(n-1)
    % Staple construction for (n=2) xy
    %                         (n=3) xz,yz
    %                         (n=4) xt,yt,zt
    staple = staple + ...
    L(:,:,m(x,N,-a(i+n-1)),m(y,N,a(2)-a(i+n-2)),m(z,N,a(3)-a(m(i,M,n-3))),m(t,N,a(4)),i)'*...
    L(:,:,m(x,N,-a(i+n-1)),m(y,N,-a(i+n-2)),m(z,N,-a(m(i,M,n-3))),t,n)'*...
    L(:,:,m(x,N,-a(i+n-1)),m(y,N,-a(i+n-2)),m(z,N,-a(m(i,M,n-3))),t,i);

    staple_conj= staple_conj +...
    L(:,:,x,y,z,t,i)*...
    L(:,:,m(x,N,+a(i+n-1)),m(y,N,+a(i+n-2)),m(z,N,+a(m(i,M,n-3))),t,n)*...
    L(:,:,x,m(y,N,a(2)),m(z,N,a(3)),m(t,N,a(4)),i)';

end
```

Plaquette measurment

```
for x=1:N
    for y=1:N
        for z=1:N
            for t=1:N

                plaq_size(x,y,z,t,1) = real( trace( L(:,:,x,y,z,t,1)*L(:,:,m(x,N,1),y,z,t,2)*...
                                                    L(:,:,x,m(y,N,1),z,t,1)'*L(:,:,x,y,z,t,2)' ) );
                plaq_size(x,y,z,t,2) = real( trace( L(:,:,x,y,z,t,1)*L(:,:,m(x,N,1),y,z,t,3)*...
                                                    L(:,:,x,y,m(z,N,1),t,1)'*L(:,:,x,y,z,t,3)' ) );
                plaq_size(x,y,z,t,3) = real( trace( L(:,:,x,y,z,t,1)*L(:,:,m(x,N,1),y,z,t,4)*...
                                                    L(:,:,x,y,z,m(t,N,1),1)'*L(:,:,x,y,z,t,4)' ) );
                plaq_size(x,y,z,t,4) = real( trace( L(:,:,x,y,z,t,2)*L(:,:,x,m(y,N,1),z,t,3)*...
                                                    L(:,:,x,y,m(z,N,1),t,2)'*L(:,:,x,y,z,t,3)' ) );
                plaq_size(x,y,z,t,5) = real( trace( L(:,:,x,y,z,t,2)*L(:,:,x,m(y,N,1),z,t,4)*...
                                                    L(:,:,x,y,z,m(t,N,1),2)'*L(:,:,x,y,z,t,4)' ) );
                plaq_size(x,y,z,t,6) = real( trace( L(:,:,x,y,z,t,3)*L(:,:,x,y,m(z,N,1),t,4)*...
                                                    L(:,:,x,y,z,m(t,N,1),3)'*L(:,:,x,y,z,t,4)' ) );

            end
        end
    end
end

plaq_size=plaq_size/3; % We divide it by 3 for the definition of wilson Loop
```

Results

Simulation results			
Action used	Plaquette size	Literature result	Time elapsed
Unimproved	0.497 ± 0.005	0.50	37 minutes
Improved	0.541 ± 0.002	0.54	95 minutes
Action used	Rectangular size	Literature result	Time elapsed
Unimproved	0.260 ± 0.005	0.26	37 minutes
Improved	0.284 ± 0.003	0.28	95 minutes

Wilson loops are unrenormalized and so these values need not agree with those from the Wilson action.
However these results are usefull for all the next simulations.

Conclusions

Conclusions

- We used the Harmonic oscillator as a framework to test the metropolis algoritm, via 3 possible discrete approximations of the action through path integral.
- Then we used the Metropolis algoritm to simulate a N^4 lattice, with 2 different actions, we calculated the average size of the plaquette operator and the 2 x 1 Wilson loop, with good agreement with the literature.
- This is just the starting point for a QCD simulation, static quark, anti-quark potential can be calculated for istance.
- Also the path integral part had some acceptable results, even though more accurate results can be achived with better variables tuning
- This tool can be used to other potentials
- Matlab turned out to be a good choice, since in the lattice simulation there are a lot of matrix multiplications.