Path integral Monte Carlo evaluation of the static quark potential

Emiliano Pezzini

May 2023

All codes can be found on my GitHub repository [3].

1 Method and results

1.1 Discretizing the path integral

The first exercise is the numerical integration of the path integral for a harmonic oscillator (then extended to an anharmonic oscillator potential).
The euclidean ($t \to it$) action

$$S[x] \equiv \int_{t_i}^{t_f} dt L(x, \dot{x} \equiv \int_{t_i}^{t_f} dt \Big[\frac{m\dot{x}(t)}{2} + V(x(t))\Big] \tag{1}$$

can be discretized as

$$S^j_{lat} = \int_{t_j}^{t_{j+1}} dt L \approx a\Big[\frac{m}{2}\big(\frac{x_{j+1} - x_j}{a}\big)^2 + \frac{1}{2}(V(x_{j+1} + V(x_j))\Big] \tag{2}$$

Then the lattice action is

$$S_{lat}[x] = \sum_{j=0}^{N} S^j_{lat} \tag{3}$$

Following Lepage[1] we extract the quantum mechanical propagator from this by numerically calculating the following integral

$$\langle x|e^{-\tilde{H}T}|x\rangle = A\int_{-\infty}^{\infty} dx_1...dx_{N-1}e^{-S_{lat}[x]} \tag{4}$$

where $A = \left(\frac{m}{2\pi a}\right)^{\frac{N}{2}}$.

Using the Monte Carlo integrator **vegas**[2], the definition of the function and the integration is as such:

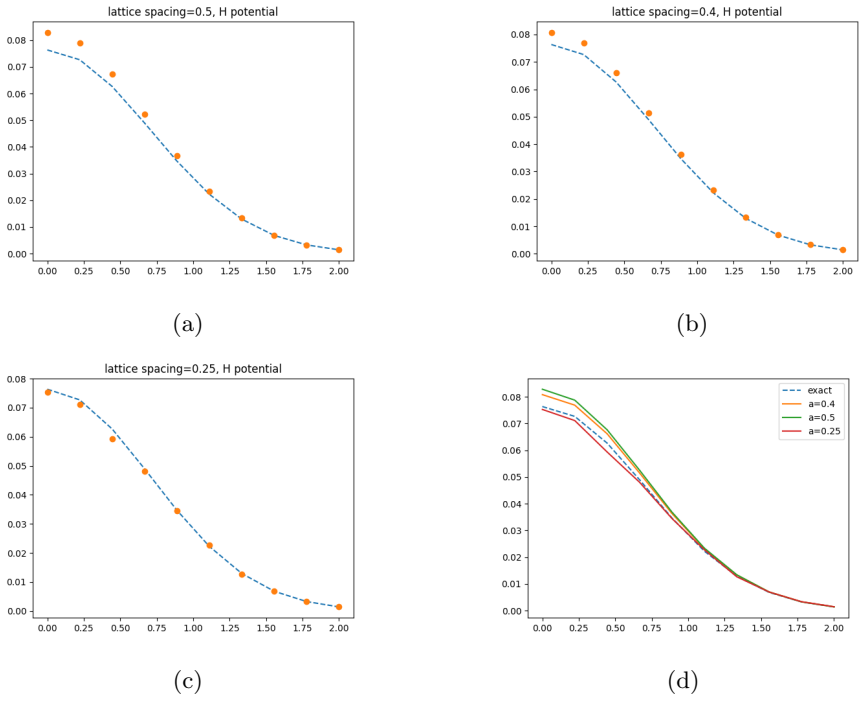
```
def I(x):
    S_E = 0
    for j in range(len(t)):
        if j == 0:
            S_E += (0.5/a)*(x[j]-x_i)**2 + a*0.5*V(x_i) + a*V(x[j])
        elif j == len(t)-1:
            S_E += (0.5/a)*((x_f-x[j])**2 + (x[j]-x[j-1])**2) + a*V(x[j])
                + 0.5*a*V(x_f)
        else:
            S_E += (0.5/a)*(x[j]-x[j-1])**2 + a*V(x[j-1])
    return ((2*np.pi*a)**(-(len(t)+1)/2))*np.exp(-S_E)

#now integrate exp(-S) in the list of variables x trough values x_t
integ = vegas.Integrator((len(t)) * [[x_min, x_max]])

#train the integrator, discard results
integ(I, nitn=20, neval=1e6, alpha=0.5)

#integrate, keep results
result = integ(I, nitn=20, neval=1e6, alpha=0.4)
```

which is looped over many initial and final values (x_i, x_f) which are set to be the same for the calculation of the propagator.
The integration is done over all x_j's in between the initial and final and in the interval $[-5, 5]$, larger intervals don't affect the result a lot as suggested in [1] and are also computationally too demanding. The integrator is trained for 20 iterations and 10^6 Monte Carlo evaluations with an adapting speed **alpha** of 0.5 (which is default), the results are discarded after this adaptation run and a run with slightly lower **alpha** for fine convergence is used. The results are shown below.



(a) (b) (c) (d)

Figure 1: Harmonic oscillator propagator for lattice spacing (a): 0.5, (b): 0.4, (c): 0.25. Meanwhile (d) sums up these results showing the asymptotic behaviour

The computational time grows extremely fast when reducing the lattice spacing, hence I stop at a value of 0.25 but it is clear that the approximation gets better and better as the spacing is reduced.

The results for the quartic potential $V(x) = \frac{x^4}{2}$ are shown in figure 2.

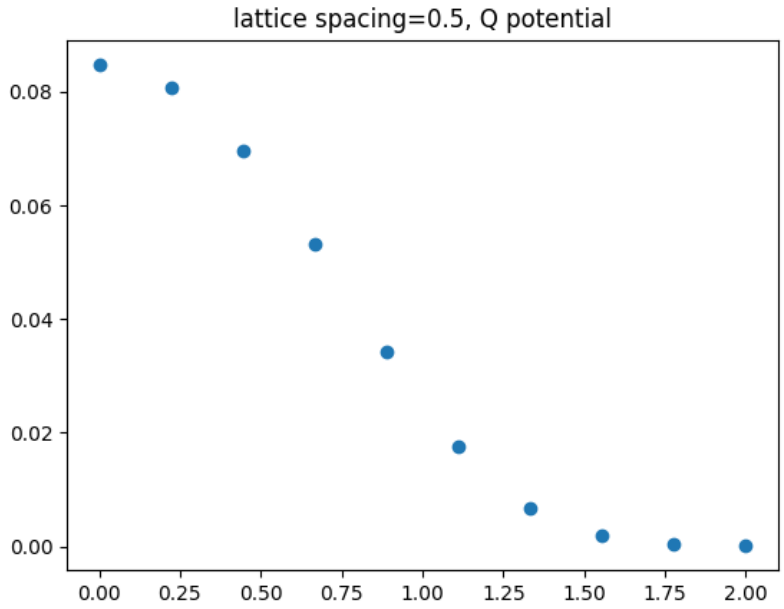


Figure 2: Quartic oscillator potential, lattice spacing 0.5

1.2 Metropolis Monte Carlo algorithm calculation of first excited state energy of the harmonic oscillator

The previous exercise was about the ground state, since in quantum field theories the ground state is the vacuum we are generally interested in excited states. To do that we introduce sources and sinks states in our propagator formula.

$$G(t) \equiv \langle\langle x(t_1)x(t_2)\rangle\rangle = \frac{\int Dx x(t_2)x(t_1)e^{-S[x]}}{\int Dx e^{-S[x]}} \tag{5}$$

where now also initial and final states are integrated over. Following [1] and using quantum mechanics one finds

$$\langle\langle x(t_1)x(t_2)\rangle\rangle = \frac{\sum e^{-E_n T}\langle E_n|\tilde{x}e^{-(\tilde{H}-E_0)t}\tilde{x}|E_n\rangle}{\sum e^{-E_n T}} \tag{6}$$

and for large evolution time T we recover

$$G(t) \to |\langle E_0|\tilde{x}|E_1\rangle|^2 e^{-(E_1-E_0)t} \tag{7}$$

and from this we extract a formula for the evaluation of the energy difference between first excited state and ground state.

$$\Delta E = \frac{1}{a} \ln \frac{G(t)}{G(t+a)} \tag{8}$$

And given this one can recover the quantum mechanical transition matrix element $\langle E_0|\tilde{x}|E_1\rangle$.

The algorithm used to find the path which minimizes the action is the Metropolis algorithm. A number N_{cf} of configurations is generated and stored after an even longer period of thermalisation ($10 \times N_{cf}$ Monte Carlo steps). Only paths that are a number N_{cor} of steps apart are stored to avoid correlation. The estimation of the correlation is done trough a binning procedure. The data is binned and several values of N_{cor} are tested until the standard deviation isn't importantly affected by this parameter. A bootstrap procedure is also applied to the data since it is computationally inexpensive and helps clear out a lot of the noise.

The metropolis path update algorithm is, inspired by the code suggested in [1]:

```
@njit
def update(x):
    for j in range(N):
        old_x = x[j]                                # save original value
        old_Sj = S(j, x)
        x[j] = x[j] + np.random.uniform(-eps, eps) # update x[j]
        dS = S(j, x) - old_Sj                       # change in action
        if dS > 0 and np.exp(-dS) < np.random.uniform(0,1):
            x[j] = old_x                            # restore old value

@njit
def S(j, x):                                        # harm. osc. S
    jp = (j+1)%N                                    # next site
    jm = (j-1)%N                                    # previous site
    jm2 = (j-2)%N
    jp2 = (j+2)%N
    if imp == 'n':
        return a*m*(w**2)*x[j]**2/2 + m*x[j]*(x[j]-x[jp]-x[jm])/a
    elif imp == 'y':
        return a*m*(w**2)*x[j]**2/2 -
            (m/(2*a))*x[j]*(-(x[jm2]+x[jp2])/6+(x[jm]+x[jp])*(8/3)-x[j]*(5/2))
    elif imp == 'noghost':
        return a*m*(w**2)*(1 + (a*w)**2/12)*x[j]**2/2 +
            m*x[j]*(x[j]-x[jp]-x[jm])/a
```

Just in time computation decorators from the **numba** Python package are used to speed up the computation allowing me to quickly run higher statistics such as in Figure 4. My results are the following:

1.2.1 Harmonic potential

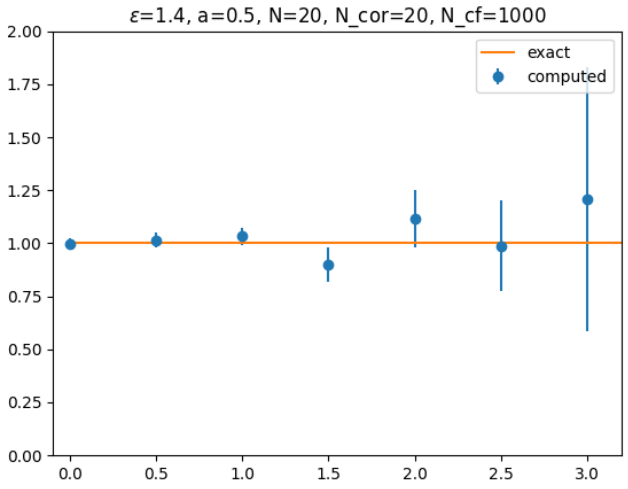


Figure 3: ΔE results

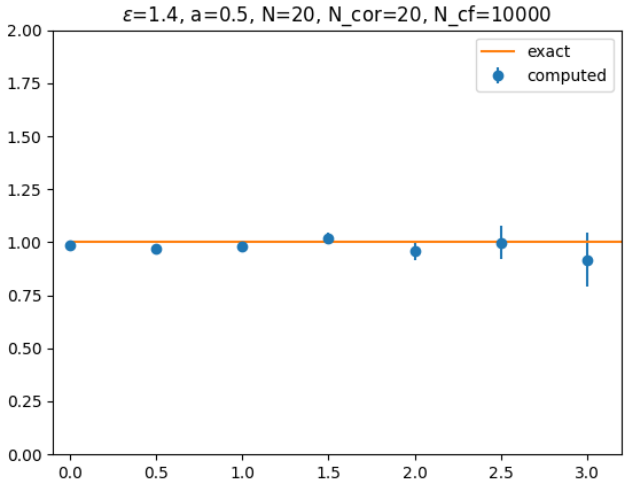


Figure 4: ΔE results with higher N_{cf}

Using a different operator such as x^3 to create and destroy our excited state I find the following results:

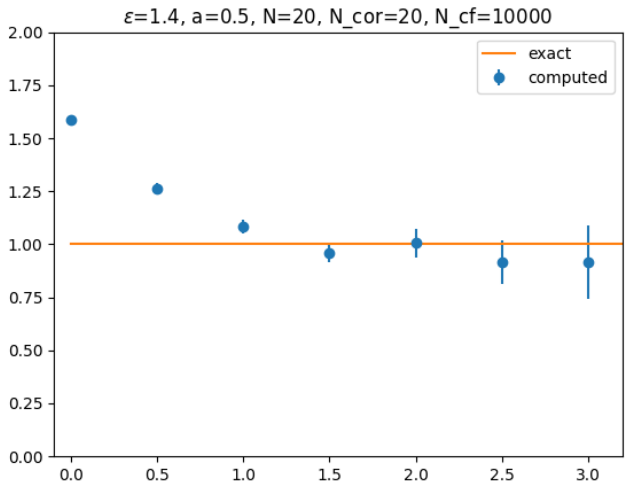


Figure 5: ΔE results for x^3 source terms

As expected the convergence is much slower and high values of N_{cf} were necessary to reduce statistical errors and accuracy.

As one can see in the source code provided, the function S which generates the action has 2 more options which one can choose from, these are the improved discretization of the derivative (which is explained in the appendix) which gives the following plot

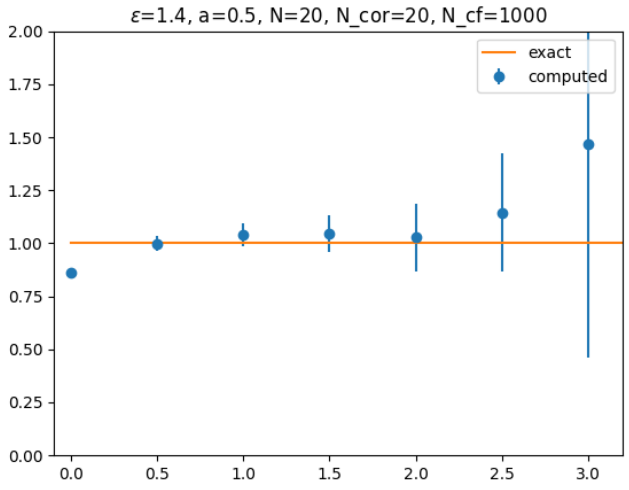


Figure 6: ΔE results for x source terms and improved action discretization

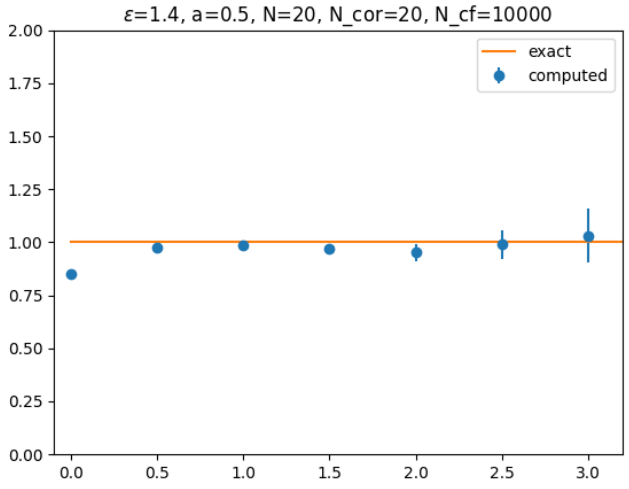


Figure 7: ΔE results for x source terms and improved action discretization and higher N_{cf}

Note that this improved discretization has the disadvantage of making the values approach the true value from below, this means we don't get a reliable upper bound on our data. Lastly this issue can be fixed by avoiding the improved

discretization and applying a field transformation trick (the option 'noghost') which will be explained better in the next subsection. I present here the results, note that it again approaches the line $\Delta E = 1$ from above:

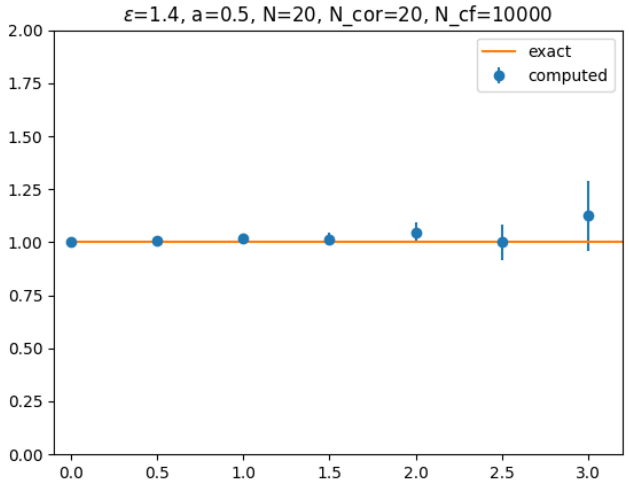


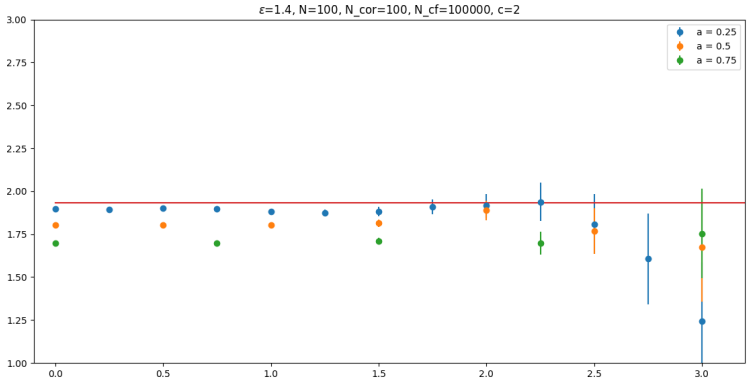
Figure 8: ΔE results for x source terms and applied field transformation trick

1.2.2 Anharmonic potential

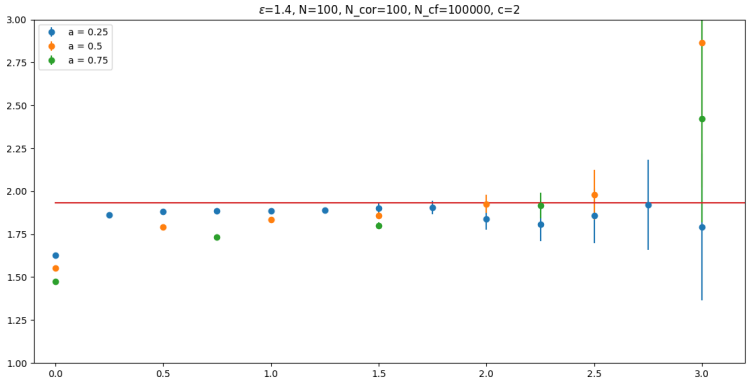
Using a potential of the form

$$V(x) = \frac{1}{2} m\omega_0^2 x^2 \left(1 + cm\omega_0 x^2\right) \tag{9}$$

The results are the following:

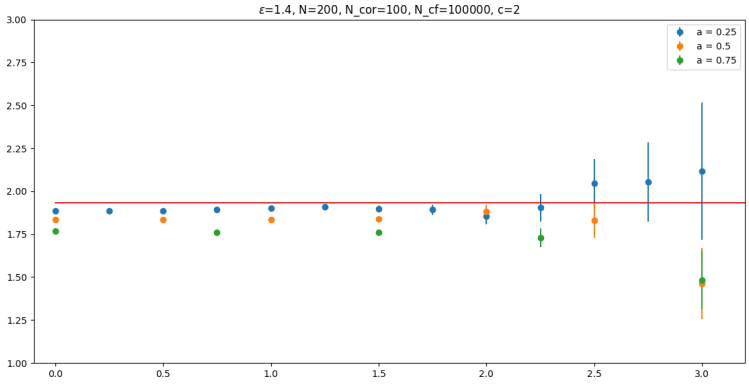


(a)

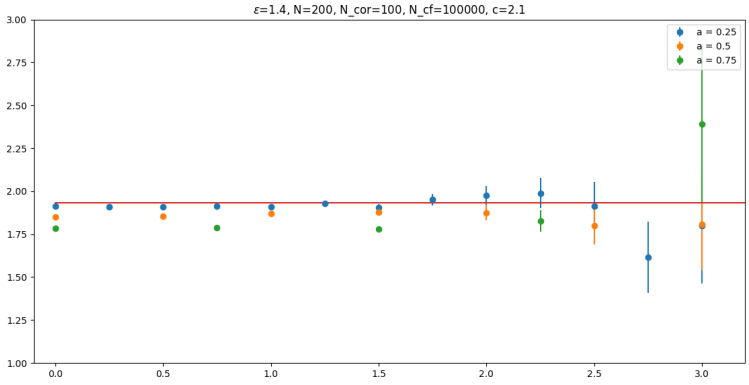


(b)

Figure 9: Anharmonic oscillator excited state energy differences sweep on lattice spacing values in the case of (a): naive discretization, (b): improved discretization



(a)



(b)

Figure 10: Anharmonic oscillator excited state energy differences sweep on lattice spacing values in the case of (a): field transformation trick with $c = 1$, (b) field transformation trick with $c = 2.1$

Trough the field transformation trick again the energies estimates are bound from above but as one can see an error of order $O(a)$ is present in all graphs, adjusting the value of the adimensional constant c in the potential according to the spacing used can balance this effect but of course this also means we are changing the potential itself.

1.3 Comparison between various discretization schemes

We now wish to compare various discretization schemes by actually numerically solving the equations of motion for the harmonic oscillator, it is known that the Euclidean e.o.m. are derived from

$$\frac{\partial S[x]}{\partial x_j} = 0 \tag{10}$$

which gives

$$m\Delta^{(2)} x_j = \frac{dV(x_j)}{dx_j} \tag{11}$$

for the usual second derivative and

$$m\Big(\Delta^{(2)} - a^2 \frac{(\Delta^{(2)})^2}{12}\Big) x_j = \frac{dV(x_j)}{dx_j} \tag{12}$$

for the improved case (see appendix for a clearer derivation and a definition of $\Delta^{(2)}$).

We write these discretized derivative operators in matrix form as ($C = -(2 + (a\omega)^2)$):

$$\begin{bmatrix} C & 1 & 0 & 0 & \dots & 0 \\ 1 & C & 1 & 0 & \dots & 0 \\ 0 & 1 & C & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & C & 1 \\ 0 & 0 & \dots & 0 & 1 & C \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} -x_{-1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -x_N \end{bmatrix} \tag{13}$$

for the unimproved case and ($K = -(\frac{5}{2} + (a\omega)^2)$

$$\begin{bmatrix} K & 4/3 & -1/12 & 0 & \dots & 0 \\ 4/3 & K & 4/3 & -1/12 & \dots & 0 \\ -1/12 & 4/3 & K & 4/3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 4/3 & K & 4/3 \\ 0 & 0 & \dots & -1/12 & 4/3 & K \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} 1/12\, x_{-2} - 4/3\, x_{-1} \\ 1/12\, x_{-1} \\ 0 \\ \vdots \\ 1/12\, x_N \\ 1/12\, x_{N+1} - 4/3\, x_N \end{bmatrix} \tag{14}$$

for the improved case.

Solving these linear systems with appropriate initial conditions for various values of lattice spacing a, we look for the euclidean version of an oscillatory solution, i.e. $x_j = e^{-\omega t_j}$.

We suppose the frequency ω to be close to ω_0 (which here I always set to 1), in this case, substituting $x_j \to e^{-\omega_0 t_j}$ and using the harmonic potential $V(x_j) = m\omega^2 x_j$, we expect the following errors in our numerical solution:

$$\omega^2 = \omega_0^2 \Big[1 - \frac{(a\omega_0)^2}{12} + O((a\omega_0)^4) \Big] \tag{15}$$

for the unimproved case and

$$\omega^2 = \omega_0^2 \Big[1 + \frac{(a\omega_0)^4}{90} + O((a\omega_0)^6) \Big] \tag{16}$$

for the improved case. Clearly the latter is better, it's lowest error is of order $(a\omega_0)^4$.

If we were to not assume $\omega \approx \omega_0$ we would, in solving numerically the exact equation (12), find a spurious solution, called a "numerical ghost" of the form

$$\omega^2 \approx \Big(\frac{0.005}{a} \Big)^2$$

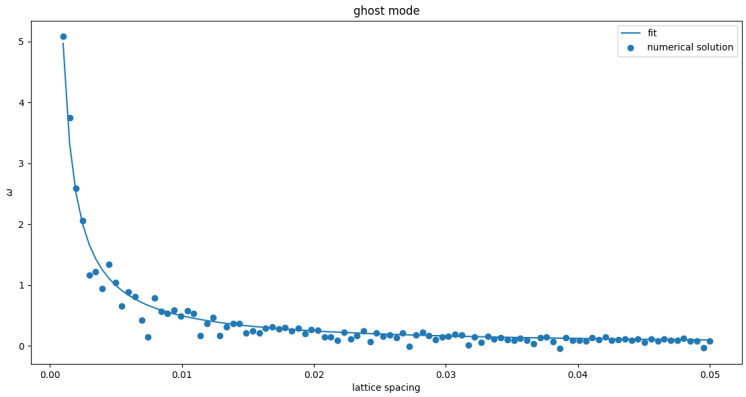


Figure 11: ghost mode

Such solution emerge from finite difference discretization schemes, especially improved ones, and can be problematic as we saw previously since the value of our ω is not bound from above. The field transformation trick mentioned above can solve this: in that case, the expected behaviour of the frequency with respect to a can be shown to be

$$\omega^2 = \omega_0^2 \Big[1 - \frac{(a\omega_0)^4}{360} + O((a\omega_0)^6) \Big] \tag{17}$$

note it is decreasing with a.

My final results are the following:

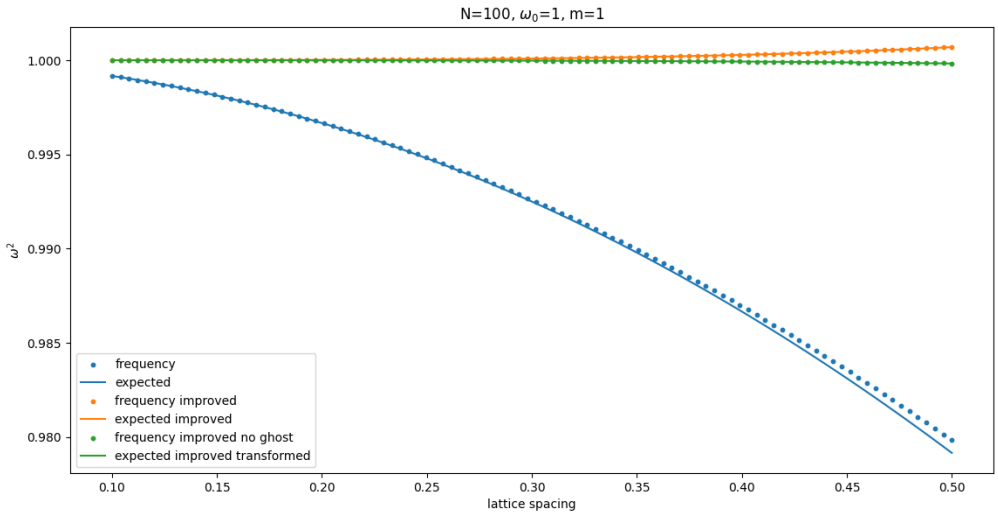


Figure 12: Unimproved action discretization

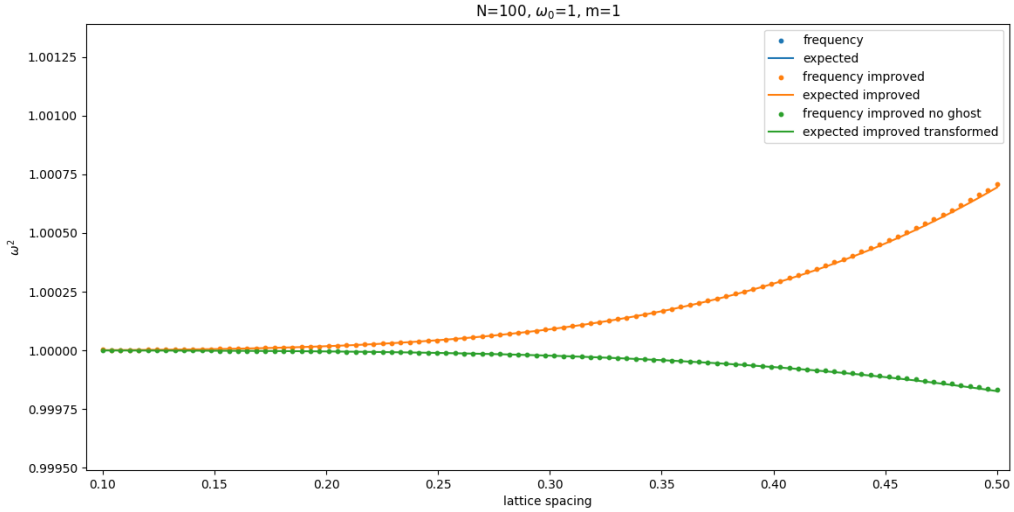


Figure 13: Improved action discretization

Figure 13 is a zoom on figure 12 to better appreciate the data fit. The green dots are obtained trough the field transformation trick which consists in noting

that a change of coordinates of the form

$$x_j \to x'_j = x_j + \delta x_j \tag{18}$$

where

$$\delta x_j = \xi_1 a^2 \Delta^{(2)} x_j + \xi_2 a^2 \omega_0^2 x_j \tag{19}$$

leaves the action unchanged, in fact

$$S[x] \to S'[x'] = S'[x + \delta x] = S'[x] + \sum_j \delta x_j \frac{\partial S'[x]}{\partial x_j} + O(a^4) = S'[x] \tag{20}$$

Where in the last step the equations of motion were used to set the terms in the sum to zero. Thus one reduces the improved five-point discretization of the second derivative back to the three-point discretization, removing the numerical ghost modes at the cost of redefining the potential:

$$S'[x] = \sum_{j=0}^{N-1} a\Big[-\frac{1}{2} m x_j \Delta^{(2)} x_j + V'(x_j)\Big] \tag{21}$$

with

$$V'(x_j) = \frac{1}{2} m \omega_0^2 x_j^2 \Big(1 + \frac{(a w_0)^2}{12}\Big) \tag{22}$$

is precise to order $O(a^4)$. The transformation being linear the Jacobian is not affected.
Jacobian effects are found when the same trick is performed on an anharmonic potential such as in the previous section since the change of coordinates need include non linear terms, this introduces renormalization errors.

1.4 Quantum Gluons

Now we wish to apply what we learned trough the previous simple model of a 1D harmonic oscillator (effectively a 1-temporal dimension field theory) to

actual 4D QCD:

The lattice is initialized by creating an empty $7D$ array: 4 dimensions are the euclidean space-time dimensions while at every point in this 4D space we assign for every direction a $3x3$ identity matrix, which is an element of SU(3), hence

$$4D \oplus 1D \oplus 2D$$

shows why 7 dimensions are needed. Now that we have at each point the 4 link variables U we create a set of SU(3) matrices which will be used in the metropolis algorithm to update a link before choosing if to accept or reject the change. The matrices are generated by first creating a matrix M with random complex numbers with real and imaginary parts both contained in the interval $[-1, 1]$, then making them hermitian by transforming them to

$$H = \frac{1}{2}(M^\dagger + M) \tag{23}$$

then converting them to unitary by transforming them to the matrix exponent of themselves to a sufficient order in the series expansion (30 in my case), such that

$$U = \sum_{n=0}^{30} \frac{i\epsilon^n}{n!} H^n \tag{24}$$

and finally making their determinant one by diving them by the cube root of their determinant

$$SU = \frac{U}{\det\{U\}^{\frac{1}{3}}} \tag{25}$$

After having created a pool of such matrices large enough to cover the whole SU(3) group (100 plus their hermitian conjugates in my case) we proceed to the lattice upgrade procedure, the part of the Wilson action (see appendix) regarding one link in direction μ at a point x in the lattice which is changed by the application of an SU(3) matrix to said link is

$$\Delta S(x, \mu) = \frac{\tilde{\beta}}{3} ReTr\Big(\sum_{\nu \neq \mu} P_{\mu\nu}\Big) = \frac{\tilde{\beta}}{3} ReTr(U_\mu(x)\Gamma_\mu(x)) \tag{26}$$

where Γ is the sum of staples in all directions orthogonal to μ. A staple is the three links completing a plaquette of which one side is the link which is being updated U_μ, meanwhile $\tilde{\beta} = \frac{\beta}{u_0^4}$ includes the tadpole improvement factor.

The metropolis update proceeds straightforwardly from here.

The lattice is then thermalised trough this process a sufficient number of times (I chose $2N_{cor}$, hence 100) and I proceed to calculate the expectation values of plaquettes, rectangles and big plaquettes, i.e. $a \times a$, $2a \times a$ and $2a \times 2a$ Wilson loops, with great agreement with [1].

N_{cf}	$a \times a$	$2a \times a$	$2a \times 2a$
1	0.5021	0.2732	0.0818
2	0.5001	0.2633	0.0802
3	0.4962	0.2676	0.0813
4	0.4959	0.2608	0.0794
5	0.4983	0.2580	0.0735
6	0.4955	0.2549	0.0828
7	0.4999	0.2587	0.0782
8	0.4952	0.2575	0.0811
9	0.5026	0.2554	0.0812
10	0.5048	0.2645	0.0828
mean	0.4991	0.2614	0.0802
σ	0.0032	0.0055	0.0026

Table 1: Unimproved action results

I then extracted the same result using the improved version of the action:

$$S = -\beta \sum_{x,\mu>\nu} \left\{ \frac{5}{3} \frac{P_{\mu\nu}}{u_0^4} - \frac{R_{\mu\nu} + R_{\mu\nu}}{12 u_0^6} \right\} \tag{27}$$

Where the R's are the rectangles containing the link $U_\mu(x)$, the new variation in the action is then:

$$\Delta S(x,\mu) = -\frac{\beta_{imp}}{3} \left\{ \frac{5}{3u_0^4} ReTr(U_\mu \Gamma_P) - \frac{1}{12u_0^6} ReTr(U_\mu \Gamma_R) \right\} \tag{28}$$

where β_{imp} takes the value given in [1] (which is determined by matching simulation results with perturbation theory calculations or experimental results) and Γ_P is as before while Γ_R is the sum of rectangular "staples" having link $U_\mu(x)$ as any edge.

This particular combination of square and rectangular Wilson loops removes order a^2 errors and helps restore a certain degree of rotational invariance, this will be important later. Arbitrarily higher corrections can be done by adding more exotic shapes of Wilson loops with appropriate constant factors, such as non-planar loops.
The results are the following

N_{cf}	$a \times a$	$2a \times a$	$2a \times 2a$
1	0.5438	0.2879	0.0791
2	0.5408	0.2927	0.0819
3	0.5383	0.2843	0.0789
4	0.5393	0.2801	0.0822
5	0.5418	0.2817	0.0789
6	0.5408	0.2855	0.0829
7	0.5380	0.2870	0.0742
8	0.5433	0.2824	0.0792
9	0.5428	0.2818	0.0821
10	0.5405	0.2811	0.0851
mean	0.5409	0.2844	0.0804
σ	0.0019	0.0037	0.0029

Table 2: Improved action results

1.5 Static quark potential

The final part of my analysis consists in the determination of the static quark potential to find evidence of the phenomenon of confinement.

The lattice creation, thermalisation and update procedure is the same as in the previous part, in addition a smearing procedure is introduced.

The smearing procedure involves averaging links with their neighbouring staples to suppress high energy gluons and hasten the convergence of results. This is important since to extract the static quark potential from the lattice we need to evaluate Wilson loops of varying duration and length, duration refers to the extending of the loop in the 0-th direction (time) while length refers to its extent in the radial direction (space).

$$W(r,t) = \langle 0|\frac{1}{3}ReTr\{L_{tr}\}|0\rangle \tag{29}$$

where L_{tr} is the said Wilson loop with dimensions t in time and r in space.

Similarly to the previous sections, for large time T in the infinitely massive quark approximation:

$$\ln\big(W(r,t)/W(r,t+a)\big) \to aV(r) \tag{30}$$

The smearing procedure (APE smearing [6][7]), which I apply before any computation of Wilson loop expectation value consists of the following transformation in the link variables:

$$\tilde{U}_\mu(x) = \mathbf{Proj}_{SU(3)}\Big\{\big(1+\epsilon a^2\Delta^{(2)}\big)^n U_\mu(x)\Big\} \tag{31}$$

where this ϵ is not the same used in the creation of the SU(3) matrices (eq. (24)), here I will use $\epsilon = \frac{1}{12}$ and (the 0-th axis is time)

$$\Delta^{(2)}U_\mu(x) \equiv \sum_{\rho\neq 0,\mu} \Delta^{(2)}_\rho U_\mu(x) \tag{32}$$

and $\Delta^{(2)}_\rho$ is a discretized gauge covariant derivative defined as:

$$\begin{aligned}\Delta^{(2)}_\rho U_\mu(x) \equiv \frac{1}{u_0^2 a^2}\big(&U_\rho(x)U_\mu(x+a\hat{\rho})U^\dagger_\rho(x+a\hat{\mu}) \\ &- 2u_0^2 U_\mu(x) + U^\dagger_\rho(x-a\hat{\rho})U_\mu(x-a\hat{\rho})U_\rho(x-a\hat{\rho}+a\hat{\mu})\big)\end{aligned} \tag{33}$$

And this smearing is applied only on the r-direction links, time-pointing links are not to be smeared.

The projection back onto SU(3) is necessary to avoid the link variables to grow out of control if more than one smearing is applied (and I will use $n = 4$ smearings). The projection is done as suggested in [4], by first performing a unitary projection on the smeared link V_μ:

$$W_\mu(x) = \frac{V_\mu(x)}{\sqrt{V^\dagger_\mu(x)V_\mu(x)}} \tag{34}$$

and then normalizing it to make it's determinant one:

$$\tilde{U}_\mu = \frac{W_\mu(x)}{\det(W_\mu(x))^{\frac{1}{3}}} \tag{35}$$

The data is fitted with a potential of coulombic/linear combination of the form

$$V(r) = \sigma r - \frac{b}{r} + c \tag{36}$$

The data shows the emergence of confinement, while I don't have data points for the low distance, coulumbic behaviour (which is usually calculated form perturbation theory since in this regime one has the so-called asymptotic freedom).

I show here the results, which took approximately from half an hour up to 3 hours of computational time depending on action improvement and smearing choices.

The fit parameters are

Improved action	**Smearing**	**Variable**	**Fit result**	**est. standard deviation**
No	No	σ	0.1124	1.3625
		b	0.4763	3.1721
		c	1.0000	4.5121
No	Yes	σ	0.1736	0.0867
		b	0.5822	0.2500
		c	1.0000	0.3307
Yes	No	σ	0.2825	0.6461
		b	0.6477	1.3235
		c	1.0000	1.9666
Yes	Yes	σ	0.2146	0.1261
		b	0.6130	0.3327
		c	0.9850	0.4543

Table 3: Best fit parameters

As one can see smearing indeed betters the data and lowers the errors considerably. This is also because it allows us to use smaller times t in equation (30) since, as we said, it hastens the convergence to the asymptotic results. This is important since other errors sum up quickly as we consider larger loops. The errors in the unsmeared cases are larger than the parameters themselves which is of course a disaster, the smeared data has still large errors but acceptable, which gives us confidence in the shape of the potential that was fitted. From my plots the improved action might seem unnecessary, this is because I didn't calculate non-planar loops which would give information about distances which aren't a multiple of a, if we did that, we would notice the convergence of same radius but different site results given by the partial restoration of rotational invariance from improving the discretization.

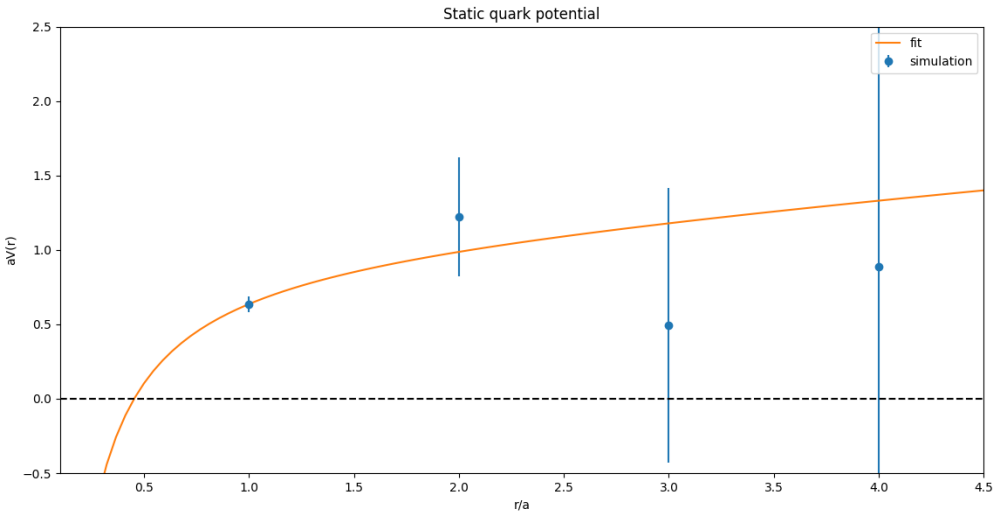


Figure 14: Unimproved action with unsmeared links

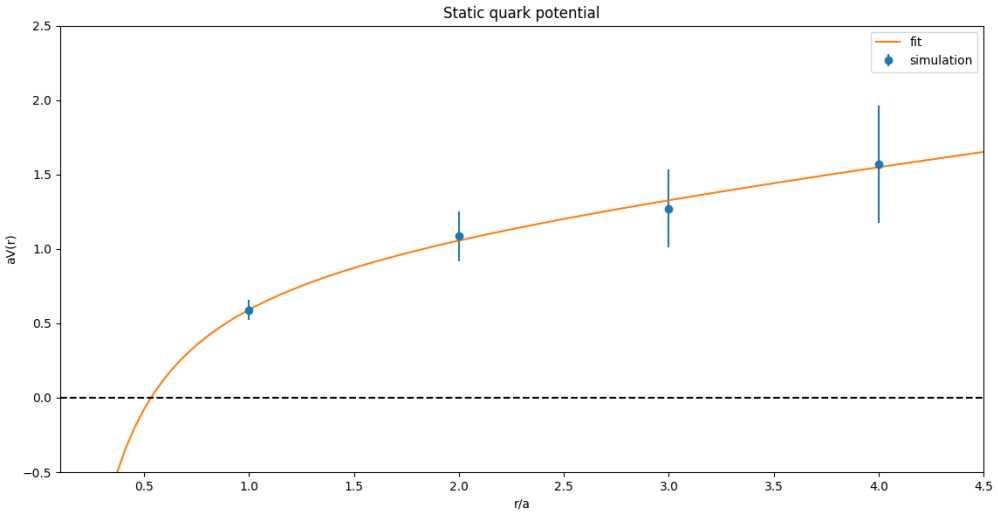


Figure 15: Unimproved action with smeared links

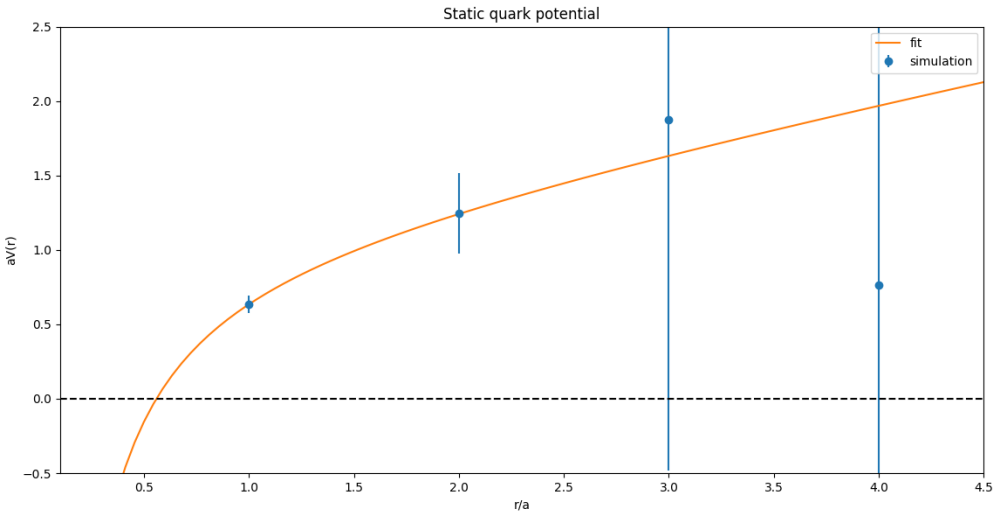


Figure 16: Improved action with unsmeared links

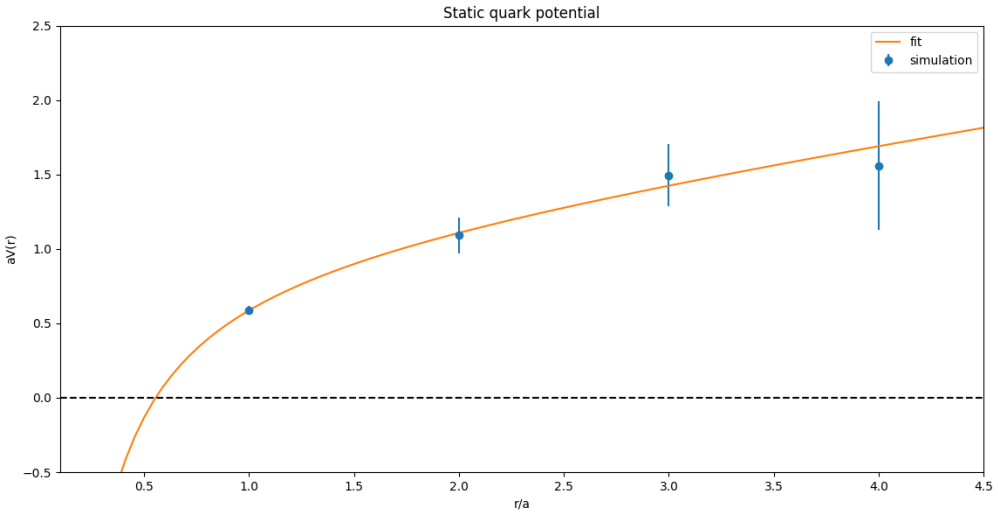


Figure 17: Improved action with smeared links

2 Appendix

2.1 Wilson action for lattice QCD

The pure gauge part of the argument in the exponential when calculating path integrals for QCD is

$$S = -\beta \int d^4x \; Tr[F_{\mu\nu}F^{\mu\nu}] \tag{37}$$

where $\beta = \frac{2N_C}{g_0^2} = \frac{2N_C}{4\pi\alpha_S}$ is related to the bare coupling constant g_0 and N_C is the number of colors. Because of asymptotic freedom the theory should be renormalizable, as such we expect $a \to 0$ as the coupling $g \to 0$ ($\beta \to \infty$). Also

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - ig[A_\mu A_\nu] \tag{38}$$

and as for the generators we use the following convention

$$T^a = T^{a\dagger} \tag{39}$$

$$Tr[T^a T^b] = \frac{1}{2}\delta^{ab} \tag{40}$$

$$F_{\mu\nu} = F^a_{\mu\nu} T^a \tag{41}$$

And work in euclidean 4D spacetime, hence the symmetry is not the Lorentz group but $SO(4)$, the group of rotations in 4D. The gauge transformation is

$$G(x) = e^{i\Lambda(x)} \tag{42}$$

where Lambda is an element of the group s.t.

$$\Lambda(x) = \Lambda(x)^a T^a \tag{43}$$

Then the fields transform as

$$A_\mu^G = GA_\mu G^\dagger + iG\partial_\mu G^\dagger \tag{44}$$

On the lattice this becomes the Wilson action trough the following procedure: Let's define the links on the lattice as the transformation of the gauge field along the link:

$$U(x)_\mu = Pe^{-i\int_x^{x+a\hat{\mu}} gA\cdot dx} \approx e^{-iaA_\mu(x+\hat{\mu})} \tag{45}$$

which is a connection between 2 nodes on the lattice, these are the degrees of freedom over which one integrates. $\hat{\mu}$ is a versor between the links while P is the path ordering operator. This last expression transforms as

$$U(x)_\mu^G = G(x)U(x)_\mu G(x+\hat{\mu})^\dagger \tag{46}$$

A plaquette is defined as the smallest loop starting from a given link and is formed from four links around a square in the $\mu-\nu$ plane

$$U(x)_{\mu\nu} = U(x)_\mu U(x+\hat{\mu})_\nu U(x+\hat{\nu})_\mu^\dagger U(x)_\nu^\dagger \tag{47}$$

Note that this is gauge invariant. The Wilson action for the gauge field is

$$S_G[U] = -\beta\sum_x\sum_{\mu<\nu}[1-\frac{1}{3}Re\{Tr\{U_{\mu\nu}(x)\}\}] \tag{48}$$

One can see that the Wilson action reduces to the continuum action using the BCH formula to first order

$$e^A e^B = e^{\{A+B+\frac{1}{2}[A,B]+\ldots\}} \tag{49}$$

It follows that using the definition of U_μ one gets

$$\begin{aligned} U_{\mu\nu}(x) = \exp\Big\{-ia[A_\mu(x)+A_\nu(x+\hat{\mu})-\frac{ia}{2}[A_\mu(x),A_\nu(x+\hat{\mu}]+\ldots]\Big\}\times \\ \exp\Big\{-ia[A_\mu(x+\hat{\nu})+A_\nu(x)-\frac{ia}{2}[A_\mu(x+\hat{\nu}),A_\nu(x)]+\ldots]\Big\} \end{aligned} \tag{50}$$

Now define a discretized version of the derivative as what is called the "forward derivative"

$$\nabla_\mu b(x) = \frac{b(x+\hat{\mu}) - b(x)}{a} \tag{51}$$

Using again the BCH formula on the two remaining exponentials and using the definition of the forward derivative one gets

$$U_{\mu\nu}(x) = \exp\Big\{-ia[\tilde{F}_{\mu\nu} + O(a)]\Big\} \tag{52}$$

where the discretized field strength tensor is

$$\tilde{F}_{\mu\nu} = \nabla_\mu A_\nu - \nabla_\nu A_\mu - ig[A_\mu(x), A_\nu(x)] \tag{53}$$

Now insert this expansion in the expression for the gluon action

$$S_G[U] = -\beta \sum_x \sum_{\mu\nu} \Big[1 - \frac{1}{6} Tr\{1 - ia^2(F_{\mu\nu} + O(a)) - \frac{1}{2}a^4 F_{\mu\nu}F_{\mu\nu} + \\ 1 + ia^2(F_{\mu\nu} + O(a)) - \frac{1}{2}a^4 F_{\mu\nu}F_{\mu\nu} + O(a^5)\}\Big] \tag{54}$$

cancelling terms and taking the limit of $a \to 0$ one reproduces the continuum Yang-Mills action.

Hence the euclidean path integral for the dicretized action is

$$Z = \int dU \exp\{-S_G[U]\} \tag{55}$$

where

$$dU = \prod_{x,\mu} dU_\mu(x) \tag{56}$$

2.2 Improved action discretization

One-dimensional quantum mechanics follows the usual action

$$S[x] = \int_{t_i}^{t_f} dt[-\frac{1}{2}m\dot{x}^2(t) + V(x(t))] \tag{57}$$

which can be recast in the following form trough integration by parts

$$S[x] = \int_{t_i}^{t_f} dt[-\frac{1}{2}mx(t)\ddot{x}(t) + V(x(t))] \tag{58}$$

the second derivative can be dicretized as

$$\ddot{x}(t_j) \to \Delta^{(2)}x_j \equiv \frac{x_{j+1} - 2x_j + x_{j-1}}{a^2} \tag{59}$$

which is the lattice Laplace operator. A better approximation comes from considering the following Taylor expansions

$$f(x-2a) \approx \frac{2}{3}a^4 f^{(4)}(x) - \frac{4}{3}a^3 f^{(3)}(x) + 2a^2 f''(x) - 2af'(x) + f(x) \tag{60}$$
$$f(x-a) \approx \frac{1}{24}a^4 f^{(4)}(x) - \frac{1}{6}a^3 f^{(3)}(x) + \frac{1}{2}a^2 f''(x) - af'(x) + f(x) \tag{61}$$
$$f(x+a) \approx \frac{1}{24}a^4 f^{(4)}(x) + \frac{1}{6}a^3 f^{(3)}(x) + \frac{1}{2}a^2 f''(x) + af'(x) + f(x) \tag{62}$$
$$f(x+2a) \approx \frac{2}{3}a^4 f^{(4)}(x) + \frac{4}{3}a^3 f^{(3)}(x) + 2a^2 f''(x) + 2af'(x) + f(x) \tag{63}$$

Then these relationships can be inverted to give

$$f'(x) \approx \frac{f(x-2a) - 8f(x-a) + 8f(a+x) - f(2a+x)}{12h} \tag{64}$$
$$f''(x) \approx \frac{-f(x-2a) + 16f(x-a) + 16f(a+x) - f(2a+x) - 30f(x)}{12a^2} \tag{65}$$
$$f^{(3)}(x) \approx \frac{-f(x-2a) + 2f(x-a) - 2f(a+x) + f(2a+x)}{2a^3} \tag{66}$$
$$f^{(4)}(x) \approx \frac{f(x-2a) - 4f(x-a) - 4f(a+x) + f(2a+x) + 6f(x)}{a^4} \tag{67}$$

Which can also be found by applying a second order contribution of the discretized second derivative operator as

$$\ddot{x}(t_j) \to \Big(\Delta^{(2)} - \frac{a^2}{12}(\Delta^{(2)})^2\Big)x_j \tag{68}$$

which we implement trough the following form of a discretized action

$$S_{imp}[x] = \sum_{j=0}^{N-1} a\Bigg[-\frac{1}{2}mx_j\Big(\Delta^{(2)} - \frac{a^2}{12}(\Delta^{(2)})^2\Big)x_j + V(x_j)\Bigg] \tag{69}$$

where

$$\Big(\Delta^{(2)} - \frac{a^2}{12}(\Delta^{(2)})^2\Big)x_j = \frac{1}{a^2}\Bigg[-\frac{1}{12}(x_{j-2}+x_{j+2}) + \frac{4}{3}(x_{j-1}+x_{j+1}) - \frac{5}{2}x_j\Bigg] \tag{70}$$

2.3 Exploring the lattice

In this section I'd like to showcase my method of computing Wilson loops and lines in the lattice. Two simple functions for moving a predefined point up and down a certain direction are the following

```
#move a coordinate point up a direction in the lattice
def up(coordinate, direction):
    coordinate[direction] = (coordinate[direction] + 1)%N
    return coordinate

#move a coordinate point down a direction in the lattice
def down(coordinate, direction):
    coordinate[direction] = (coordinate[direction] - 1)%N
    return coordinate
```

whose function is straightforward, once at the appropriate point, we can call a link variable departing that point from the lattice by setting the **dagger** argument to **False**, or a link variables reaching said point by setting **dagger**=**True** in case we want to move in a negative direction. This is achieved by the following function:

```
#call a link SU(3) at a certain point in the lattice given a direction
    or its hermitian conjugate if direction is negative
def call_link(point, direction, lattice, dagger:bool):
    if dagger == False:
        return lattice[point[0], point[1], point[2], point[3], direction]
    elif dagger == True:
        return dag(lattice[point[0], point[1], point[2], point[3],
            direction])
```

Using these methods a function to calculate a generic planar Wilson loop can be written easily, here I show an example which calculates loops of a certain duration in the time direction and a specified length in the spatial direction (this is done in all 3 spatial direction and then averaged), it is a trivial generalization to extract any kind of loop trough this method and even non planar loops would only require an extra **if** statement.

```
def planar_loops(lattice, point, length, duration):
    W_planar=0
    for space_direction in range(1, 4, 1):
        loop = np.identity(3, np.complex128)
        for time in range(duration):
            link = call_link(point, 0, lattice, dagger=False)
            link = np.ascontiguousarray(link)
            loop = loop @ link
            up(point, 0)
        for space in range(length):
            link = call_link(point, space_direction, lattice,
                dagger=False)
            link = np.ascontiguousarray(link)
            loop = loop @ link
            up(point, space_direction)
        for time_reverse in range(duration):
            down(point, 0)
            link = call_link(point, 0, lattice, dagger=True)
            link = np.ascontiguousarray(link)
            loop = loop @ link
        for space_reverse in range(length):
            down(point, space_direction)
            link = call_link(point, space_direction, lattice, dagger=True)
            link = np.ascontiguousarray(link)
            loop = loop @ link
        W_planar += (1/3)*np.real(np.trace(loop))
    return W_planar/3
```

3 Bibliography

[1] Lepage, G. Peter, Lattice QCD for Novices, 2005, arXiv, arXiv:hep-lat/0506036

[2] https://vegas.readthedocs.io/en/latest/

[3] https://github.com/indoor-industry/LQCD

[4] Smoothing algorithms for projected center-vortex gauge fields arXiv:2203.09764 [hep-lat]

[5] Lattice QCD - A guide for people who want results arXiv:hep-lat/0509046

[6] Bacilieri, P., Bernaschi, M., Cabasino, S., Cabibbo, N., Coppola, F., Fernández, L. A., ... & APE Collaboration. (1989). On the order of the deconfining phase transition in SU (3) LGT. Nuclear Physics B-Proceedings Supplements, 9, 315-319.

[7] Albanese, M., Costantini, F., Fiorentini, G., Flore, F., Lombardo, M. P., Tripiccione, R., ... & Ape Collaboration. (1987). Glueball masses and string tension in lattice QCD. Physics Letters B, 192(1-2), 163-169.

[8] QCD forces and heavy quark bound states arXiv:hep-ph/0001312

[9] (Lecture notes in physics 788) Christof Gattringer, Christian B. Lang (auth.) - Quantum chromodynamics on the lattice: an introductory presentation-Springer-Verlag Berlin Heidelberg (2010)

[10] Quark action for very coarse lattices, Mark Alford, Timothy R. Klassen, and G. Peter Lepage Phys. Rev. D 58, 034503 – Published 29 June 1998

[11] Redesigning Lattice QCD - G. Peter Lepage arXiv:hep-lat/9607076

[12] Analytic Smearing of SU(3) Link Variables in Lattice QCD Colin Morningstar, Mike Peardon, arXiv:hep-lat/0311018