# Path Integral Monte Carlo

## Lattice QCD for novices

Matilde Grassi

Department of Physics - University of Bologna
Theoretical and Numerical Aspects of Nuclear Physics

July 2024

# Table of Contents

# Discretizing Path Integrals for 1-dim Quantum Mechanics to evaluate the ground state

**Propagator** from $t_i$ to $t_f$: $\langle x_f | e^{-\hat{H}(t_f - t_i)} | x_i \rangle = \int Dx(t) e^{-S[x]}$

w/ **Classical Action**: $S[x] \equiv \int_{t_i}^{t_f} L(x, \dot{x})\, dt \equiv \int_{t_i}^{t_f} dt \left( \frac{m\dot{x}(t)^2}{2} + V(x(t)) \right)$

Setting $x_i = x_f = x$ and $t_f - t_i = T$ we have

$$\langle x | e^{-\hat{H}T} | x \rangle = \sum_k \langle x | E_k \rangle e^{-E_k T} \langle E_k | x \rangle.$$

At $T \to \infty$ only the **groundstate** contributes so we can extract it by integrating over x: $\int dx \langle x | e^{-\hat{H}T} | x \rangle \quad \to_{T \to \infty} \quad e^{-E_0 T}$.

We can also determine the **ground state wavefunction**: $\Psi_{E_0}(x) = \langle x | E_0 \rangle$.

# Discretizing Path Integrals for 1-dim Quantum Mechanics to evaluate the ground state

**How to develop a numerical procedure for evaluating the propagator using path integral**

- We approximate $x(t)$ only at the nodes on a **discretized** t axis:
  $\Rightarrow t_j = t_i + ja$ with $j = 0, \ldots, N$ and Lattice spacing $a = \frac{t_f - t_i}{N}$
  $\Rightarrow x(t) = \{x(t_0), x(t_1), \ldots x(t_N)\}$
  $\Rightarrow \int Dx(t) \;\; \rightarrow \;\; A \int_{-\infty}^{+\infty} dx_1\, dx_2 \ldots dx_{N-1}$ with $x_0 = x_N = x$
- The action for $t_j \leq t \leq t_{j+1}$ is

$$S_{lat}^{j} \approx a \left[ \frac{m}{2} \left( \frac{x_{j+1} - x_j}{a} \right)^2 + \frac{1}{2} \left( V(x_{j+1}) + V(x_j) \right) \right]$$

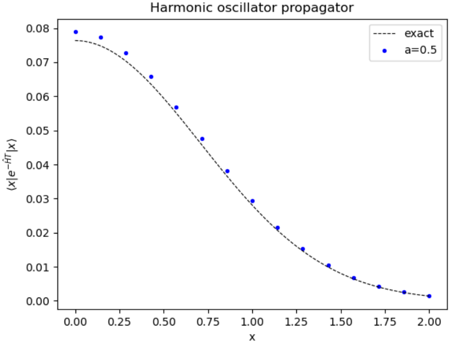
$$\langle x | e^{-\hat{H}(t_f - t_i)} | x \rangle \approx A \int_{-\infty}^{+\infty} dx_1 dx_2 \ldots dx_{N-1} e^{-S_{lat}[x]}$$

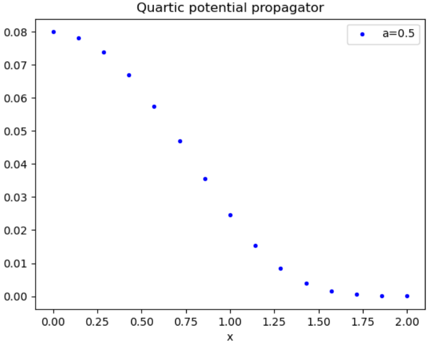
# Exercise 1: Evaluation of harmonic and quartic potential propagator for several values of $x_0 = x_N$

```python
#Integrand function
def I(x):
    S=0 #constructing the action
    for i in range(N-1):
        if i == 0:
            #this is the action from x_i to x[0]
            S += (1/(2*a))*(x[i]-x_i)**2 + (a/2)*(V(x_i) + V(x[i]))
        elif i == N-2:
            #this is the action from x[5] to x[6] and x[6] to x_f
            S += (1/(2*a))*((x_f-x[i])**2 + (x[i]-x[i-1])**2) + (a/2)*(V(x[i])+
            V(x[i-1])) + (a/2)*(V(x_f)+V(x[i]))
        else:
            #this is the action from x[i-1] to x[i]
            S += (1/(2*a))*(x[i]-x[i-1])**2 + (a/2)*(V(x[i-1])+V(x[i]))
    return ((1/(2*np.pi*a))**(N/2))*np.exp(-S)

#Specify integration details, including integration intervals for each of the
#independent variables.
integ = vegas.Integrator((N-1)*[[x_min, x_max]])

#Integration results of the integrand function I
result = integ(I, nitn=10, neval=100000)
```

Code extracted from github.com/MatildeGrassi/Lattice-QCD-for-novices—project

# Exercise 1: Results for $a = 0.5$, $N = 8$, $x_0 = x_N = np.linspace(0, 2, 15)$



# Monte Carlo evaluation of path integral to extract states beyond the ground state

**Path integral averages** $\langle\langle\Gamma(x)\rangle\rangle$ of arbitrary functionals $\Gamma(x)$

$$\langle\langle\Gamma(x)\rangle\rangle = \frac{\int Dx(t)\Gamma(x)e^{-S[x]}}{\int Dx(t)e^{-S[x]}}$$

are used to compute physical property of the excited states in quantum theory.

**e.g. in exercise 2**: $\Delta E = E_1 - E_0 = \frac{1}{a}\log\frac{G(t)}{G(t+a)}$

with $G(t) = \frac{1}{N}\sum_j \langle\langle x(t_{(j+n)modN})x(t_j)\rangle\rangle$

# Metropolis algorithm

To evaluate the path integrals we need to generate a large number of configurations $x^\alpha = \{x_0^\alpha, x_1^\alpha, \ldots, x_N^\alpha\}$ with $\alpha = 1, 2, \ldots, N_{cf}$, in order to compute the **Monte Carlo estimator**: $\langle\langle \Gamma(x) \rangle\rangle \approx \overline{\Gamma} = \frac{1}{N_{cf}} \sum_{\alpha=1}^{N_{cf}} \Gamma[x^\alpha]$.

```
@njit
def update(x): #to update each site of path
    for j in range(N):
        #saving original values and compute the action
        old_x = x[j]
        old_Sj = S(j,x)
        #update the x[j] using a uniformly distributed probability
        x[j] += np.random.uniform(-eps,eps)
        #compute the difference between the old and new action
        dS = S(j,x) - old_Sj
        #condition to restore the old value
        if dS>0 and np.exp(-dS)<np.random.uniform(0,1):
            x[j] = old_x
@njit
def S(j,x): #Action
    jp = (j+1)%N  #taking into account the periodicity xN=x0
    jm = (j-1)%N
    return a*V(x[j])+x[j]*(x[j]-x[jp]-x[jm])/a
```

Code extracted from github.com/MatildeGrassi/Lattice-QCD-for-novices—project

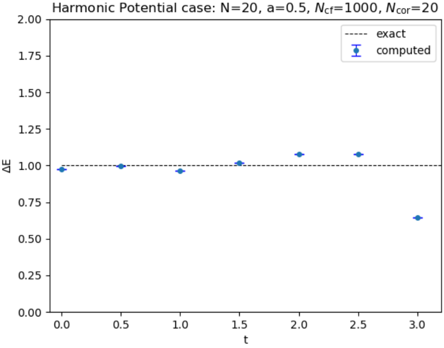
# Metropolis algorithm

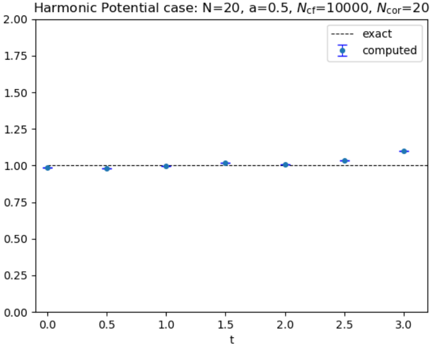
Successive paths generated by the Metropolis algorithm will be correlated. We need to **thermalize** them at the beginning and we need to keep only every $N_{cor}$ updates.

```python
@njit
def MCaverage(x,G): #erase correlations and thermalize the G's
    for j in range(0,N):
        x[j] = 0 #initialization of the path as all zeros
    for j in range(10*N_cor): #thermalization
        update(x)
    for alpha in range(len(G)):
        for j in range(N_cor): #erasing correlations
            update(x)
        for n in range(0,N):
            G[alpha][n] = compute_G(x,n) #filling the G
    return G
```

Code extracted from github.com/MatildeGrassi/Lattice-QCD-for-novices—project

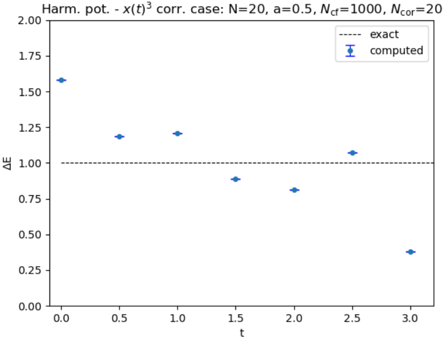
# Exercise 2: Results for Harmonic potential and correlator $\frac{1}{N}\sum_j \langle\langle x(t_{(j+n)modN})x(t_j)\rangle\rangle$

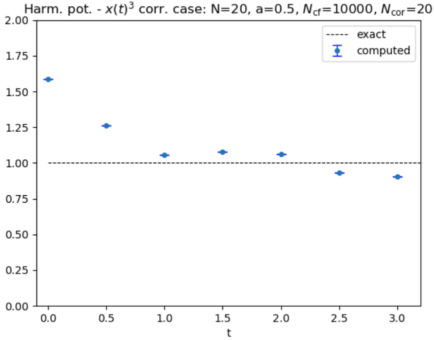


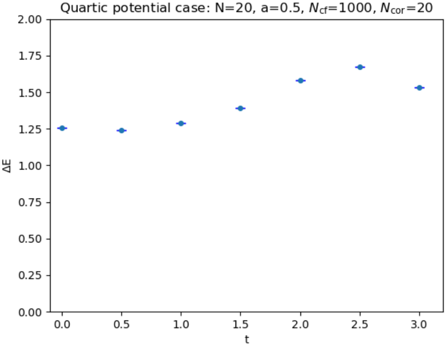
# Exercise 3: Results for Harmonic potential and correlator

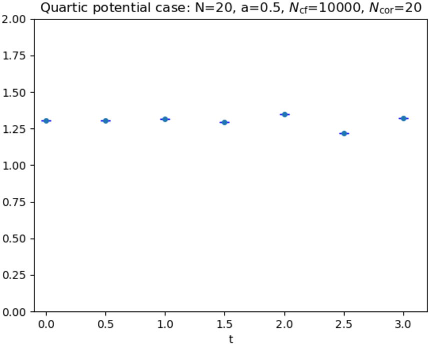
$$\frac{1}{N}\sum_j \langle\langle x(t_{(j+n)modN})^3 x(t_j)^3\rangle\rangle$$



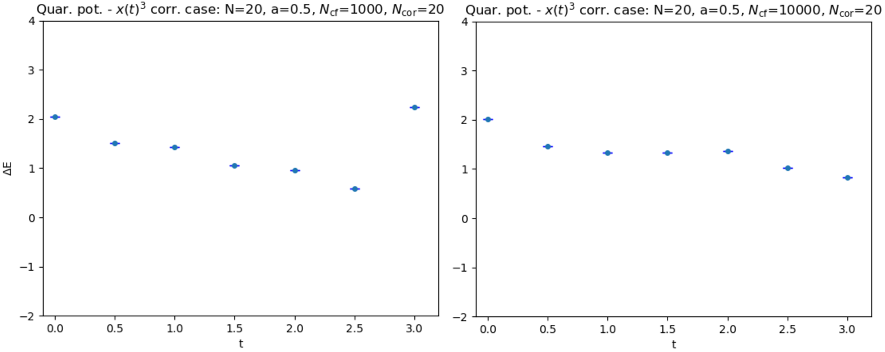



# Exercise 2: Results for Quartic potential and correlator $\frac{1}{N}\sum_j \langle\langle x(t_{(j+n)modN})x(t_j)\rangle\rangle$



# Exercise 3: Results for Quartic potential and correlator $\frac{1}{N}\sum_j \langle\langle x(t_{(j+n) modN})^3 x(t_j)^3\rangle\rangle$



# Estimation of errors - Statistical bootstrap

Given an ensemble $G^{\alpha}$ with $\alpha = 1, \ldots, N_{cf}$, we assemble a bootstrap copy of that ensemble by selecting $G^{\alpha\prime}s$ at random from the original ensemble, allowing duplications and omissions.

```
@njit
def bootstrap(G): #bootstrapping the G
   #new ensemble
   G_bootstrap = np.zeros((len(G), N))
   for i in range(len(G)):
     #choose random config
     k = int(np.random.uniform(0,len(G)))
     #keep G[k]
     G_bootstrap[i]=G[k]
   return G_bootstrap
```

```
@njit #DeltaE+errors
def bootstrap_deltaE(G,nbstrap=100):
   bootstrap_E = np.zeros((nbstrap,N-1))
   #bs copies of deltaE
   for i in range(nbstrap):
      g = bootstrap(G)
      g_avg=average(g)
      bootstrap_E[i]=deltaE(g_avg)
    #spread of deltaEs
    sdevE = stdDev(bootstrap_E)
    #deltaEs
    avgE=average(bootstrap_E)
    return avgE,sdevE

```

Code extracted from github.com/MatildeGrassi/Lattice-QCD-for-novices—project

# Estimation of errors - Binning

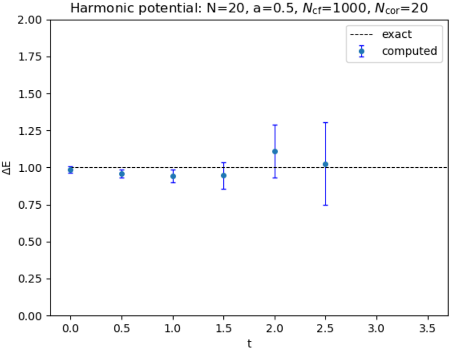
**Binning** allows us to save a lot of disk space, RAM and CPU time and it also reduces correlations.

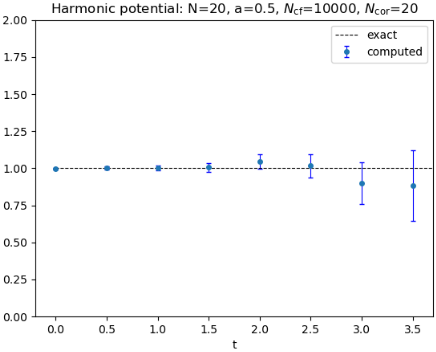
```
def bin(G,binsize): #binning the G's
    G_binned = np.zeros((int(len(G)/binsize), N))
    k=0
    for i in range(0,len(G),binsize):
        G_avg = 0
        for j in range(binsize): #summing for each bin
            G_avg = G_avg + G[i+j]
        G_binned[k]=(G_avg/binsize) #bin average
        k+=1
    return G_binned
```

Code extracted from github.com/MatildeGrassi/Lattice-QCD-for-novices—project

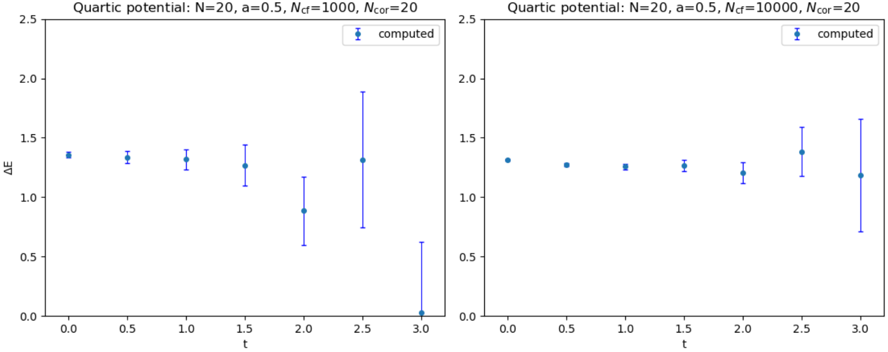
# Exercise 4: Results for harmonic oscillator and correlator

$$\frac{1}{N}\sum_j \langle\langle x(t_{(j+n)modN})x(t_j)\rangle\rangle$$



# Exercise 4: Results for quartic potential and correlator $\frac{1}{N}\sum_j \langle\langle x(t_{(j+n)modN})x(t_j)\rangle\rangle$



# Improved discretization of action

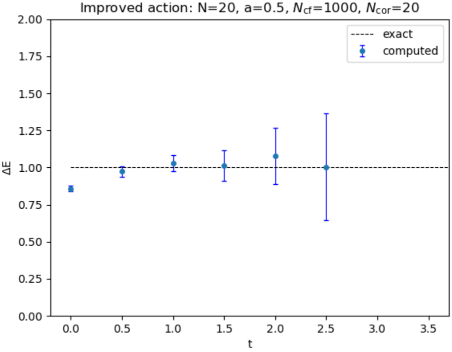
We use the **4th order central difference method** applied to the 2nd derivative:

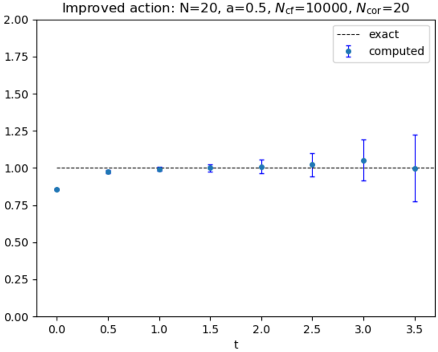
$$\ddot{x}(t) \rightarrow \Delta^{(2)} x_j - \frac{a^2}{12}(\Delta^{(2)})^2 x_j \quad w/\ \ \Delta^{(2)} x_j = \frac{x_{j+1} - 2x_j + x_{j-1}}{a^2}$$

Thus, we obtain

$$S_{imp}[x] = a \sum_{j=0}^{N-1} \left[ -\frac{1}{2} m x_j \left( \frac{-x_{j+2} + 16x_{j+1} - 30x_j + 16x_{j-1} - x_{j-2}}{12a^2} \right) + V(x(t)) \right]$$

# Exercise 6: Results with improved action for harmonic case



# Avoiding ghost states

Let's derive the **discretized euclidean eq. of motion** from $\frac{\partial S[x]}{\partial x_j} = 0$.

Using the **unimproved S[x]** w/ harmonic pot. we get $\frac{x_{i+1} - 2x_i + x_{x-1}}{a^2} = \omega_0^2 x_i$.

Let's plug in the ansatz solution $x_i = e^{-\omega t_j}$ we obtain the frequency

$$\omega^2 = \omega_0^2 \left[ 1 - \frac{(a\omega_0)^2}{12} + \mathcal{O}((a\omega)^4) \right] \quad \textit{error of the order of } a^2$$

Using the **improved S[x]** we obtain two solutions.
One is

$$\omega^2 = \omega_0^2 \left[ 1 + \frac{(a\omega_0)^4}{90} + \mathcal{O}((a\omega)^6) \right] \quad \textit{error of the order of } a^4 \ (\textit{Improved})$$

# Avoiding ghost states - Field transformation

The other solutions comes from setting $\omega_0 = 0$:

$$\omega^2 \approx \left(\frac{2.6}{a}\right)^2$$

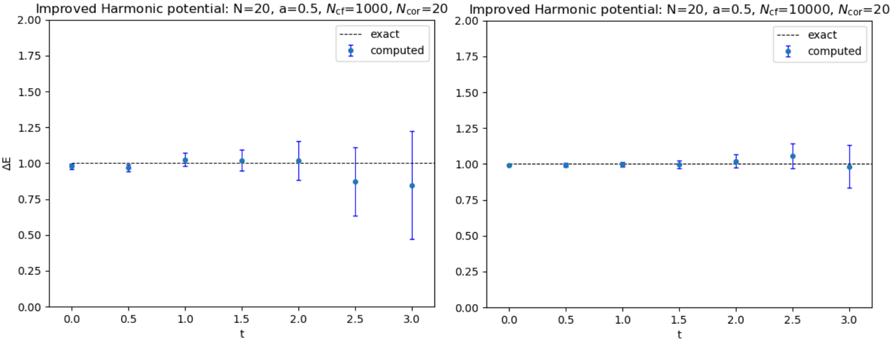
It corresponds to new oscillation modes which are an **artifact of the improved lattice theory**, called **ghost modes**.
To avoid them we operate a change of variables:

$$x_j \ \rightarrow \tilde{x}_j \quad w/\ \ x_j = \tilde{x}_j + \delta\tilde{x}_j \ \ and \ \ \delta\tilde{x}_j = \xi_1 a^2 \Delta^{(2)}\tilde{x}_j + \xi_2 a^2 \omega_0^2 \tilde{x}_j$$

$$\Rightarrow \tilde{S}_{imp}(\tilde{x}) = \frac{1}{2} m\tilde{x}_j \Delta^{(2)}\tilde{x}_j + \tilde{V}_{imp}(\tilde{x}_j)$$

$$with \quad \tilde{V}_{imp}(\tilde{x}) = \frac{1}{2} m\omega_0^2 \tilde{x}_j{}^2 \left(1 + \frac{(a\omega_0)^2}{12}\right)$$

# Exercise 7: Avoiding ghost states for harmonic case



# Avoiding ghost states - Field transformation

Generalizing this trick to the anharmonic case:

$$V(x) = \frac{1}{2} m\omega_0^2 x^2 (1 + cm\omega_0 x^2) \quad \textit{with c dimensionless parameter}.$$

We do the following variable change:

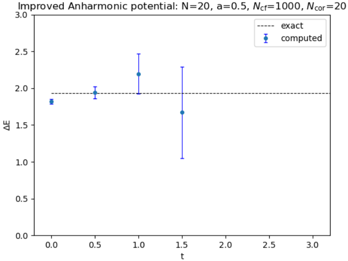
$$\delta\tilde{x}_j \equiv \xi_1 a^2 \Delta^{(2)} \tilde{x}_j + \xi_2 a^2 \omega_0^2 \tilde{x}_j + \xi_3 a^2 m\omega_0^3 \tilde{x}_j^3.$$

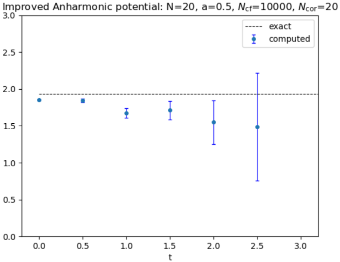
We obtain the new $\tilde{V}_{imp}$:

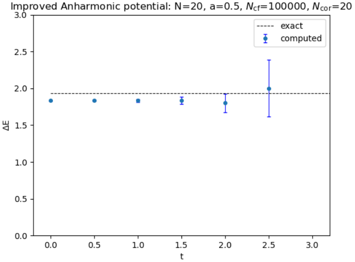
$$\tilde{V}_{imp}(\tilde{x}) = \frac{m\omega_0^2}{2} \tilde{x}^2 (1 + cm\omega_0 \tilde{x}^2) + \frac{a^2 m\omega_0^4}{24} (\tilde{x} + 2cm\omega_0 \tilde{x}^3)^2 - a\delta v(\tilde{x}) + \frac{a^3}{2} \delta v(\tilde{x})^2$$

$$\textit{with} \quad \delta v(\tilde{x}) \equiv cm\omega_0^3 \tilde{x}^2 / 4.$$

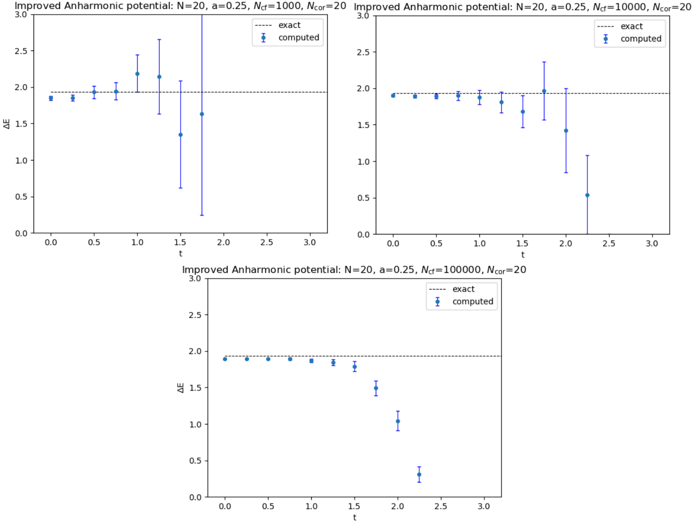
# Exercise 8: Avoiding ghost states for anharmonic case



# Exercise 8: Avoiding ghost states for anharmonic case



# Bibliography


- https://github.com/MatildeGrassi/Lattice-QCD-for-novices—project
- https://arxiv.org/abs/hep-lat/0506036
- https://github.com/paolofinelli/latticeQCD_novices