Numerical simulations of Lattice QCD

Riccardo Bianconcini, Francesco Marini

Marzo 2025

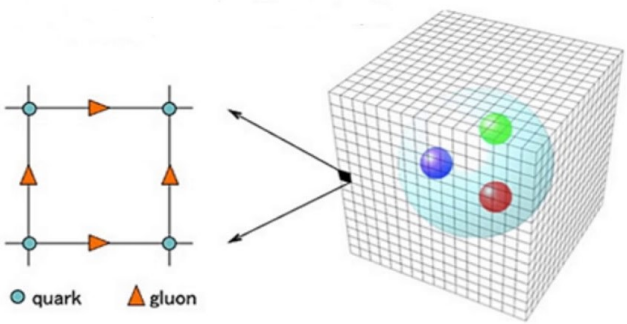
Quantum Chromodynamics

QCD The most fundamental Quantum Field Theory to describe the Strong Force.

Problem Calculations of probability amplitudes and other observables are not possible with the perturbative approach in the $\leq 1\ GeV$ scale :

Strong coupling constant $\alpha_s \sim 1$

Alternative framework Lattice QCD



Feature and Advantages Rightful employment of Path Integral calculations on the Euclidian grid (which allows implementation of computer simulations) and cut-off provided by the spacing of the lattice a.

Numerical Path Integral

Path Integral for Fields are computed summing over trajectories in the space of field configurations, and present some complexities for numerical analysis.

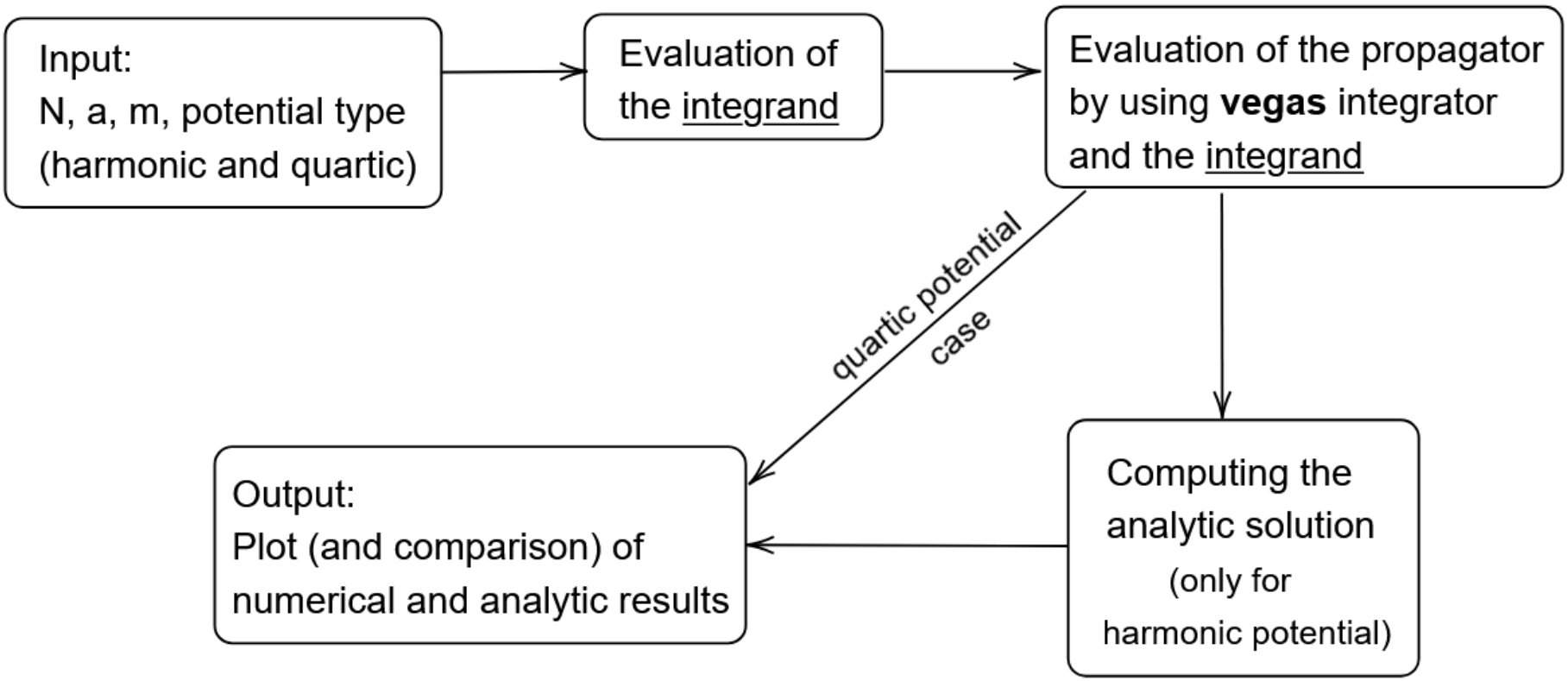
Computer feasibility is achieved discretizing properly the lattice and considering a practicable number of field configurations.

First approach We consider a point-like particle path, describing its trajectory with a function of time $x(t)$ evaluated on a discrete time axis.

$$S_{\text{lat}}[x] \sim a \sum_{j=0}^{N-1} \left[\frac{m}{2} \left(\frac{x_{j+1} - x_j}{a} \right)^2 + \frac{1}{2} \left(V(x_{j+1}) + V(x_j) \right) \right] \tag{2.1}$$

The Euclidian Action after Time Slicing

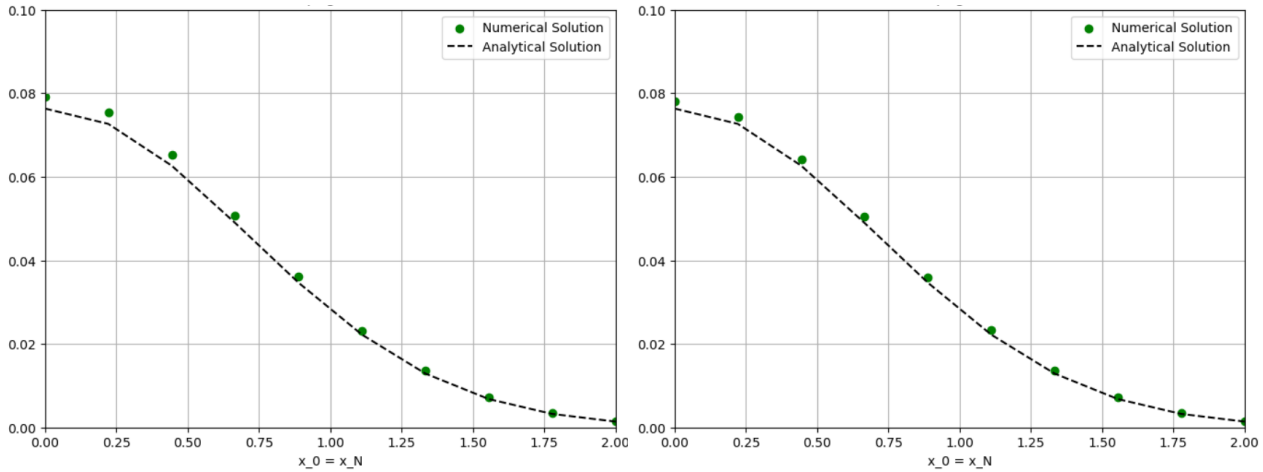
NumericalPathIntegral.py



Harmonic Potential results

Addressing the discretization issue we used the Monte Carlo integrator *vegas* to evaluate the propagator (2.2).

$$K = A \int_{-\infty}^{\infty} dx_0\, dx_1 \ldots dx_{N-1}\, e^{-S_{\text{lat}}[x]} \tag{2.2}$$

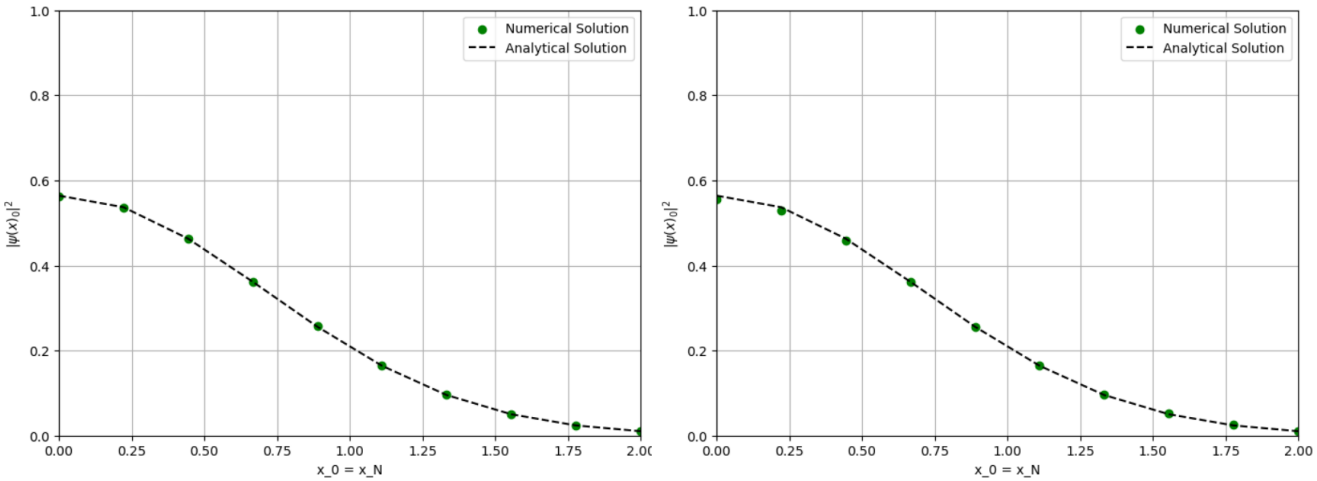


Harmonic oscillator propagator for: (left) a = 0.5 and N = 8,
(right) a = 0.4 and N = 10

Harmonic Potential results

This implementation can be checked calculating the Ground State Energy expected to be $E_0 = 1/2$, using

$$\int dx \langle x \,|\, e^{-\hat{H}T} \,|\, x \rangle \quad \underset{T\to\infty}{\longrightarrow} \quad e^{-E_0 T} = e^{-aN/2} \tag{2.3}$$

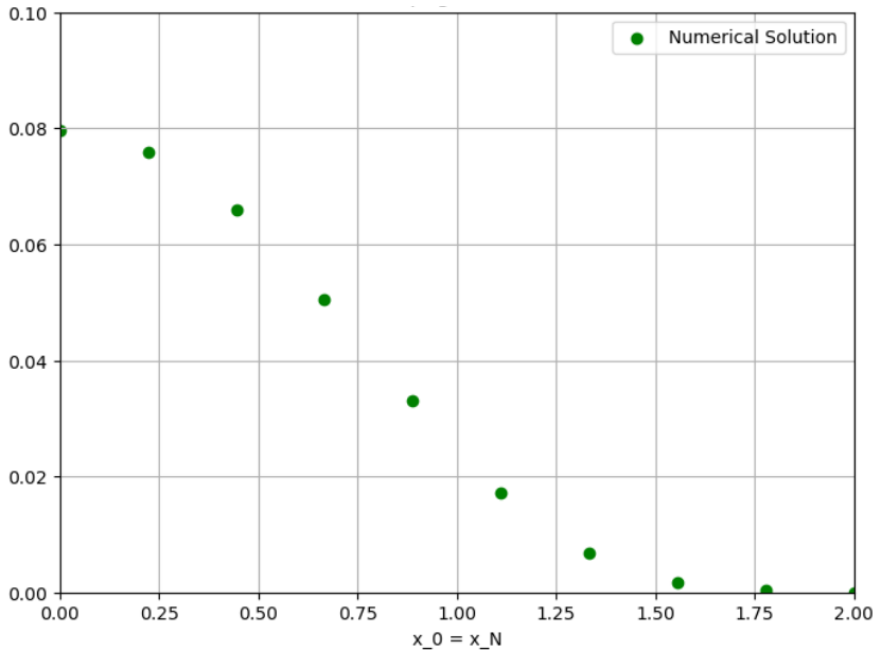


Harmonic oscillator ground state wavefunction for: (left) a = 0.5 and N = 8, (right) a = 0.4 and N = 10. The analytical results are obtained by exploiting

$$\langle x|n\rangle = \frac{2^n n!}{\pi^{1/4}} e^{-x^2/2} H_n(x) \quad \rightarrow \quad \langle x|0\rangle = \frac{1}{\pi^{1/4}} e^{-x^2/2}. \tag{2.4}$$

Quartic Potential results

The propagator is evaluated also for the quartic potential, using $V(x) = x^4/2$ inside the equation (2.1).



Numerical results for a = 0.5 and N = 8

Monte Carlo evaluation of Path Integral

For Quantum Field Theories the ground state correspond to the vacuum, our interest is therefore directed in the first excited state.

To achieve this, one way is to introduce the discretized 2-point Correlation Function

$$\langle x(t_2)x(t_1)\rangle = \frac{\sum e^{-E_n T}\langle E_n|\tilde{x}e^{-(\hat{H}-E_n)t}\tilde{x}|E_n\rangle}{\sum e^{-E_n T}} \tag{2.5}$$

and assume $t << T$, such that the propagator can be written as:

$$G(t) \underset{t \text{ large}}{\longrightarrow} |\langle E_0|\tilde{x}|E_1\rangle|^2 e^{-(E_1-E_0)t}, \tag{2.6}$$

and the first excitation energy can be extracted from its large-t dependance.

$$\Delta E \equiv E_1 - E_0 = \frac{1}{a}\log\left(\frac{G(t)}{G(t+a)}\right), \tag{2.7}$$

Metropolis Algorithm

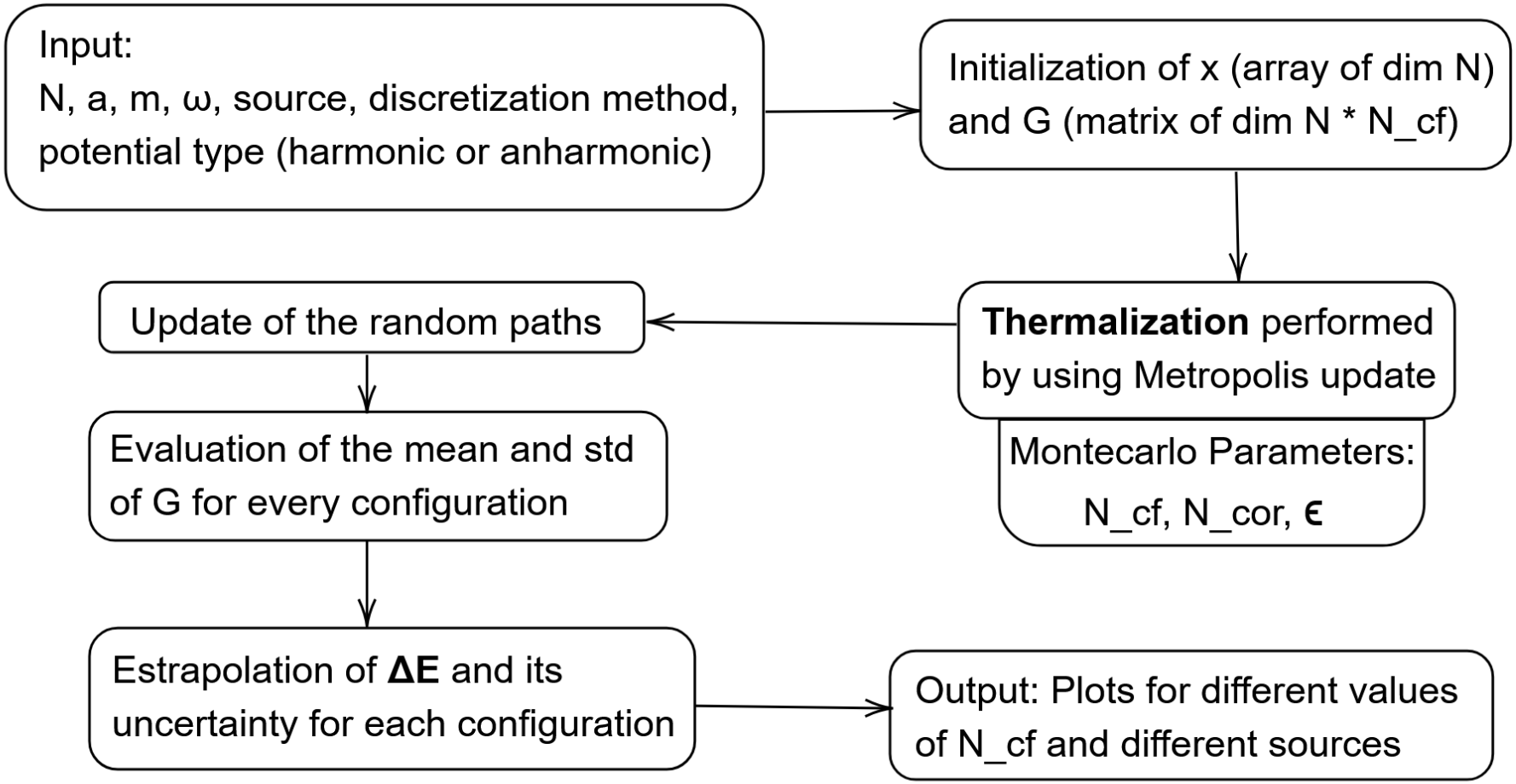
In this case a direct sampling is difficult to manage (computer unfeasibility).

To address this issue is useful to employ the Metropolis Monte Carlo algorithm.

This method generate a sequence of random samples (paths) from a probability distribution $P[x^{(\alpha)}] \propto e^{-S[x^{(\alpha)}]}$, specifying the iterations of the algorithm N_{cf} and the acceptance rate of the new samples ($48\% - 50\%$).

```
    def update(self):
        accepted_moves = 0  # Counter for accepted moves
        total_moves = 0    # Counter for total moves

        for j in range(self.N):
            old_x = self.x[j]
            old_Sj = self.S(j, self.x)
            self.x[j] += np.random.uniform(-self.eps, self.eps)  # Propose a new value for x[j]
            dS = self.S(j, self.x) - old_Sj  # Compute the change in the action
            total_moves += 1  # Increment the total moves counter
            if dS <= 0 or np.exp(-dS) >= np.random.uniform(0, 1):
                accepted_moves += 1  # Increment the accepted moves counter if accepted
            else:
                self.x[j] = old_x  # Revert to the previous value if not accepted

        acceptance_rate = accepted_moves / total_moves if total_moves > 0 else 0

        return acceptance_rate
```

PathIntegralMonteCarlo.py



Statistical Errors analysis

To improve the statistical errors values one can increase N_{cf}, raising also the computational time. A more clever procedure is to introduce the Bootstrap function, which generates multiple resampled ensambles reducing also correlation between paths.

```
    def bin(self, G, binsize):
        return np.array([np.mean(G[i:i + binsize], axis=0) for i in
            range(0, len(G), binsize)])

    def bootstrap(self, G):
        N_bs = len(G)
        return G[np.random.randint(0, N_bs, size=N_bs)]

    def bootstrap_deltaE(self, G, nbstrap=100):
        bsE = np.empty((nbstrap, self.N - 1))
        for i in range(nbstrap):
            g = self.bootstrap(G)
            bsE[i] = self.deltaE(self.avg(g))
        return self.avg(bsE), self.sdev(bsE)
```

This process is repeated numerous times ($N_{bstrap} = 1000$), recomputing the functional for each resampled ensambles

Statistical Errors analysis

We adopted also the Binning Procedure, to reduce again the correlation between paths.

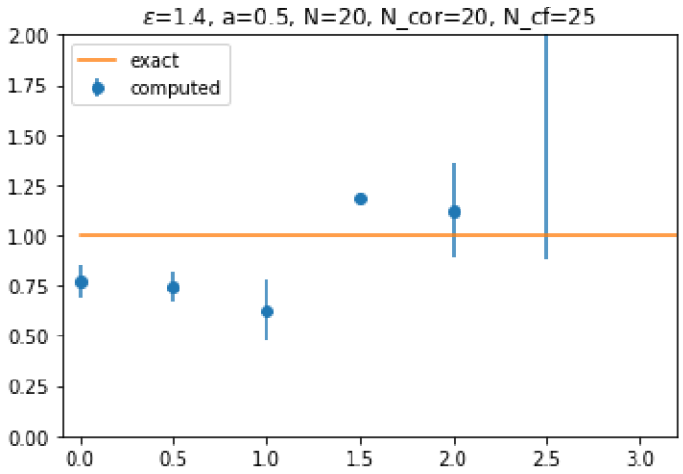
$$\bar{G}^{(1)} \equiv \frac{G^{(1)} + G^{(2)} + G^{(3)} + G^{(4)}}{4} \quad ... \tag{2.8}$$

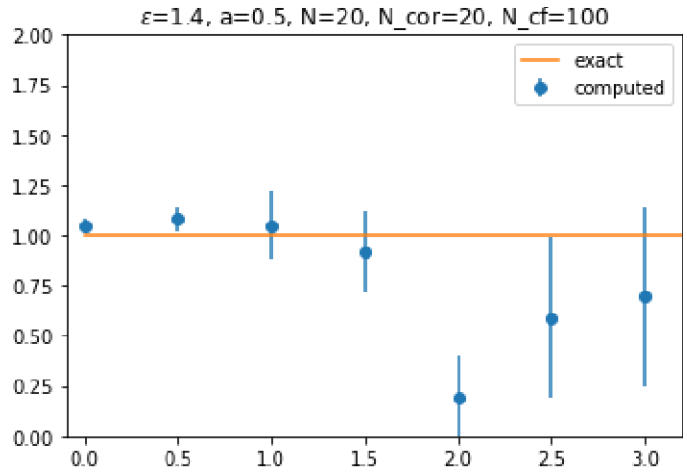
This storing process reduce indeed the size of the ensemble, but mantain the statistical properties.

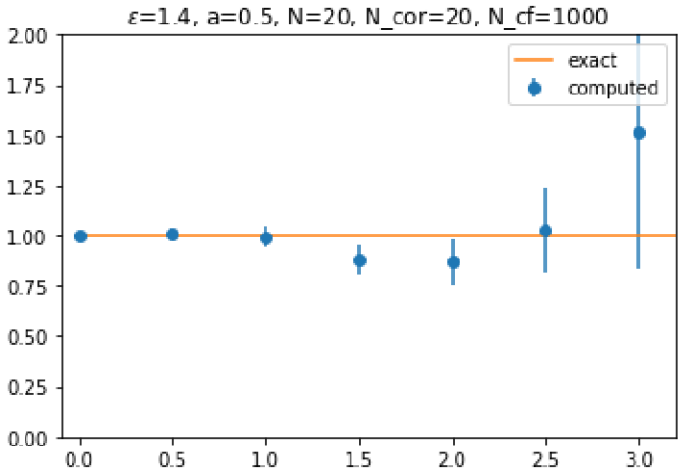
The last step consist in Thermalizing the Lattice, i.e. discard the atypical initial configurations.

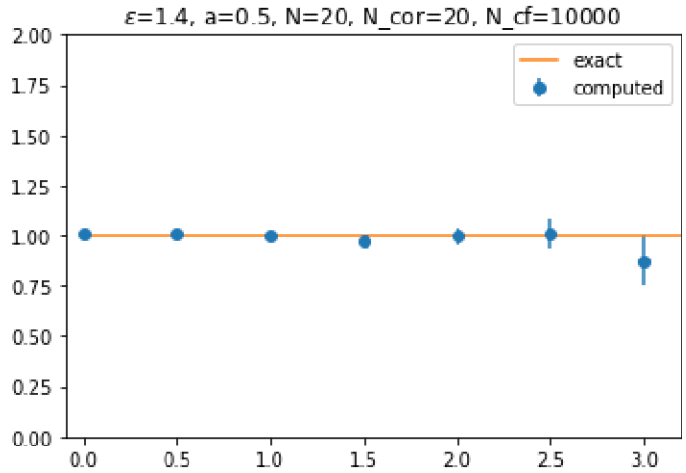
```
    def MCaverage(self, x, G):
        x.fill(0)
        total_acceptance_rate = 0  # Accumulator for acceptance rates
        num_updates = 0  # Counter for the number of updates

        # Thermalization step
        for _ in range(10 * self.N_cor):
            total_acceptance_rate += self.update()
            num_updates += 1

        # Compute configurations
        for alpha in range(self.N_cf):
            for _ in range(self.N_cor):
                total_acceptance_rate += self.update()
                num_updates += 1
            for n in range(self.N):
                G[alpha][n] = self.compute_G(x, n)
```

ΔE Results





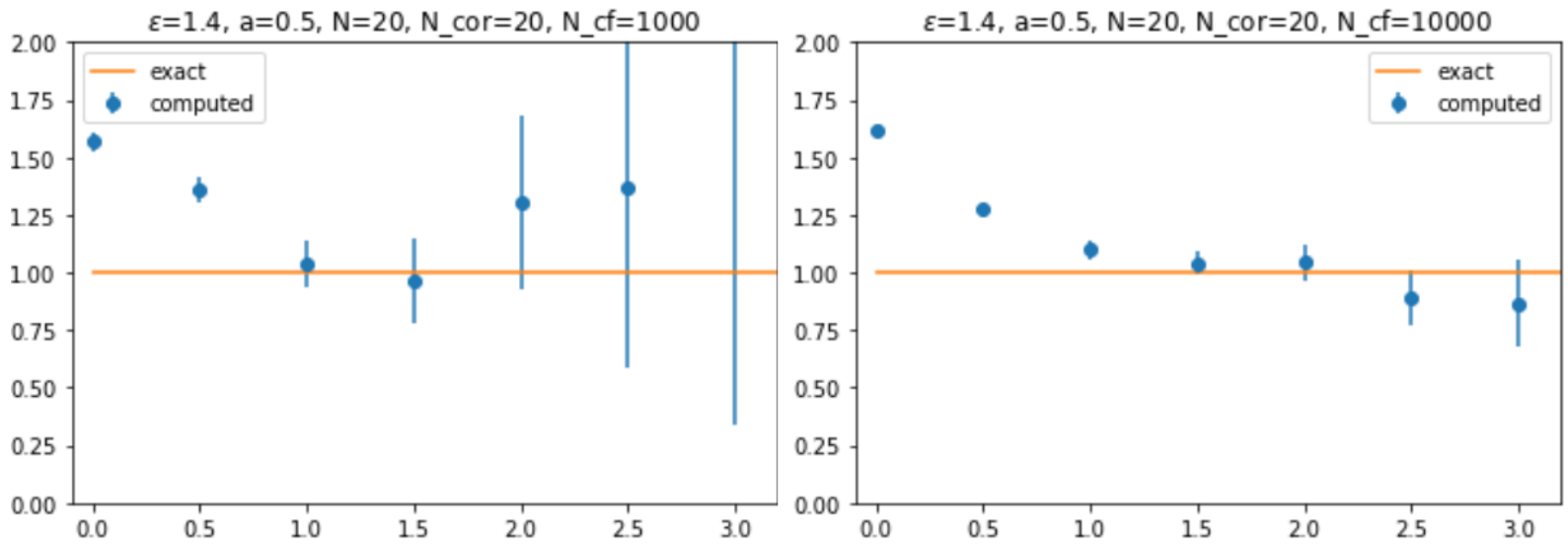




This results show, as expected, more accurate sampling for higher values of N_{cf}

ΔE Results for the x^3 case

Repeating the computation with x^3 as source and sink, instead of x, gives ΔE results that converge slower in the same value of the previous case.



The results obtained with $N_{cf} = 25, 100$ are still too random

The Different Discretizations Used

The Lattice approximation entails a discretization of the field and, as a consequence, of its derivatives.

$$\left.\frac{\partial^2\phi(x)}{\partial x^2}\right|_{x=x_j} \approx \Delta_x\phi(x_j) + \mathcal{O}(a^2) = \left.\frac{\phi(x+a) - 2\phi(x) + \phi(x-a)}{a^2}\right|_{x=x_j} + \mathcal{O}(a^2) \tag{2.9}$$

An improved discretization (2.10) obviously reduce the magnitude of discretization errors, but also carry some relevant differences.

$$\left.\frac{\partial^2\phi(x)}{\partial x^2}\right|_{x=x_j} \approx \Delta_x\phi(x_j) - \frac{a^2}{12}(\Delta_x)^2\phi(x_j) + \mathcal{O}(a^4) \tag{2.10}$$

Improved Action

$$\left.\frac{\partial S[x]}{\partial x}\right|_{x=x_j} = 0 \tag{2.11}$$

The Equation of Motion are derived from the minimized Action (2.11), the results are indeed dependent from the approximation used in the discretization.

The Unimproved Action admits a unique solution:

$$\omega^2 \approx \omega_0^2 \left(1 - \frac{(a\omega_0)^2}{12}\right) \tag{2.12}$$

The Improved Action instead admits two solutions, of which the physical one is:

$$\omega^2 \approx \omega_0^2 \left(1 + \frac{(a\omega_0)^4}{90}\right) \tag{2.13}$$

Improved Action

There are two key differences between the iterative solution for the improved action and the iterative solution for the unimproved one:

The errors are of fourth order in $a\omega_0$ instead of second order.

Freq. ω_0 now represents a lower bound for ω instead of an upper bound.

There exists indeed a second solution, called "numerical ghost".

Such states have negative norm, which lower the ΔE values.

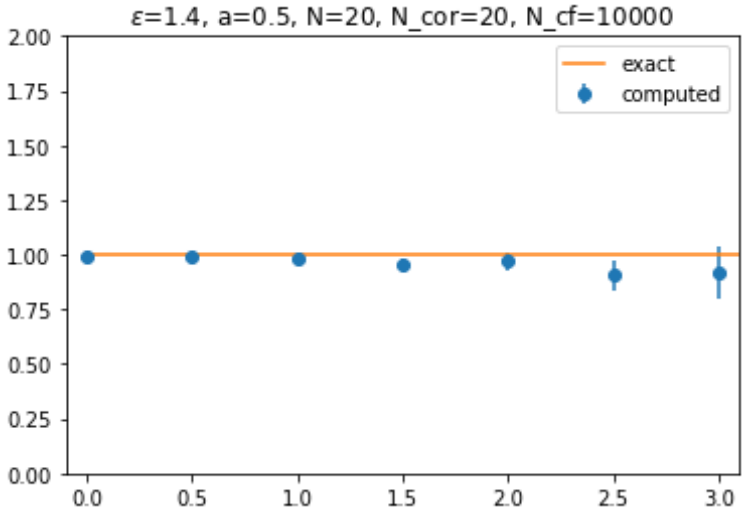
To eliminate these extra states, a change of variables is performed in the path integral:

$$x_j = \tilde{x}_j + \delta\tilde{x}_j \equiv \tilde{x}_j + \xi_1 a^2 \Delta\tilde{x}_j + \xi_2 a^2 \omega_0^2 \tilde{x}_j, \tag{2.14}$$

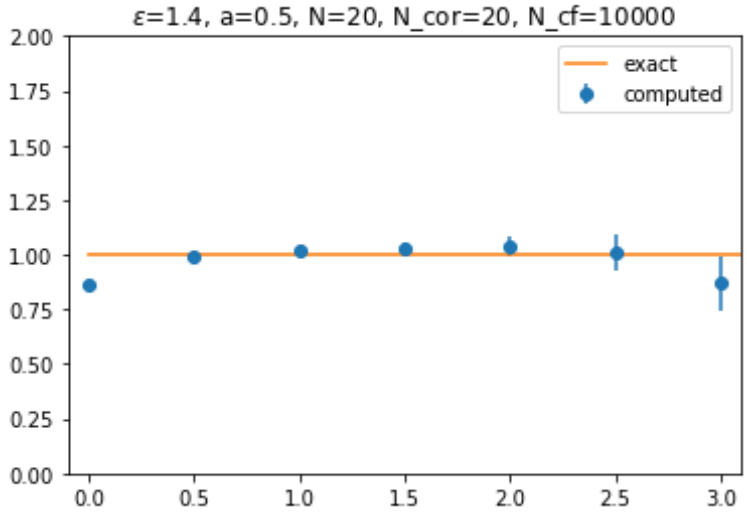
which leads to an iterative solution of the form:

$$\omega^2 \approx \omega_0^2 \left(1 - \frac{(a\omega_0)^4}{360}\right). \tag{2.15}$$

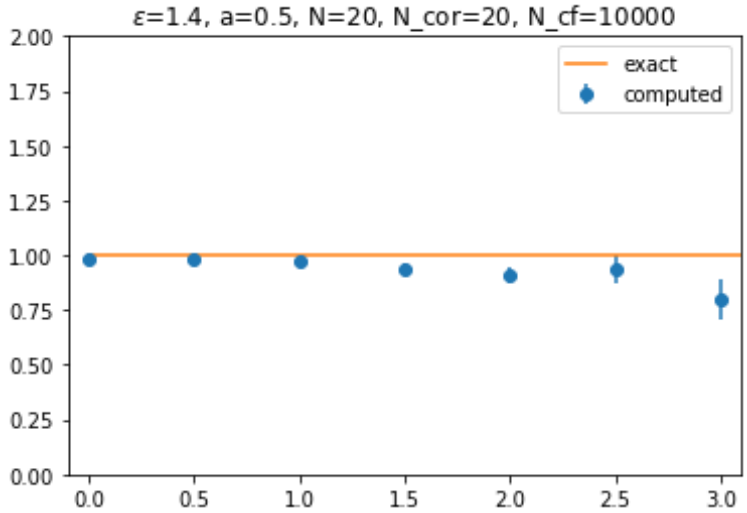
Discretization Methods Comparison



Not improved

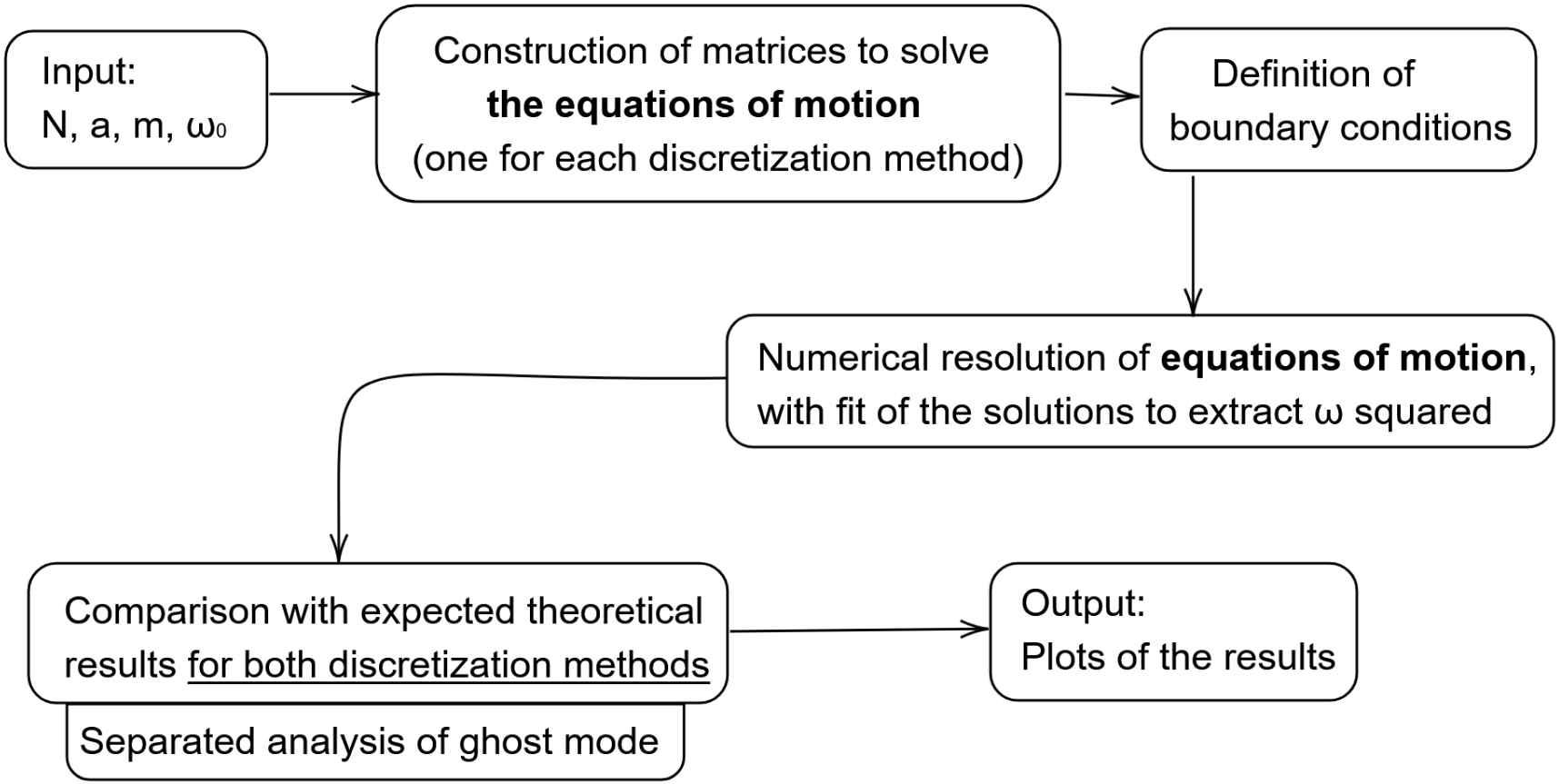


Improved

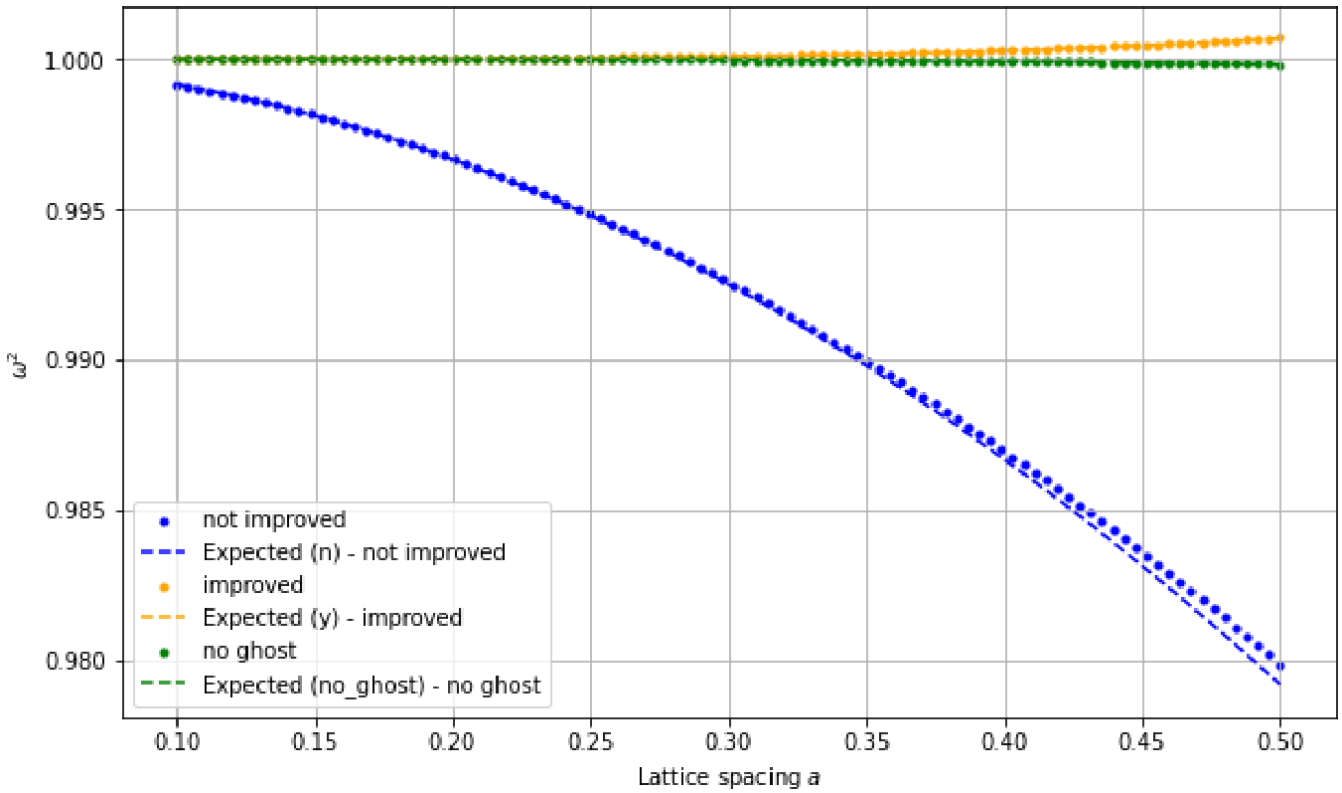


No ghost states

EOMSolver.py

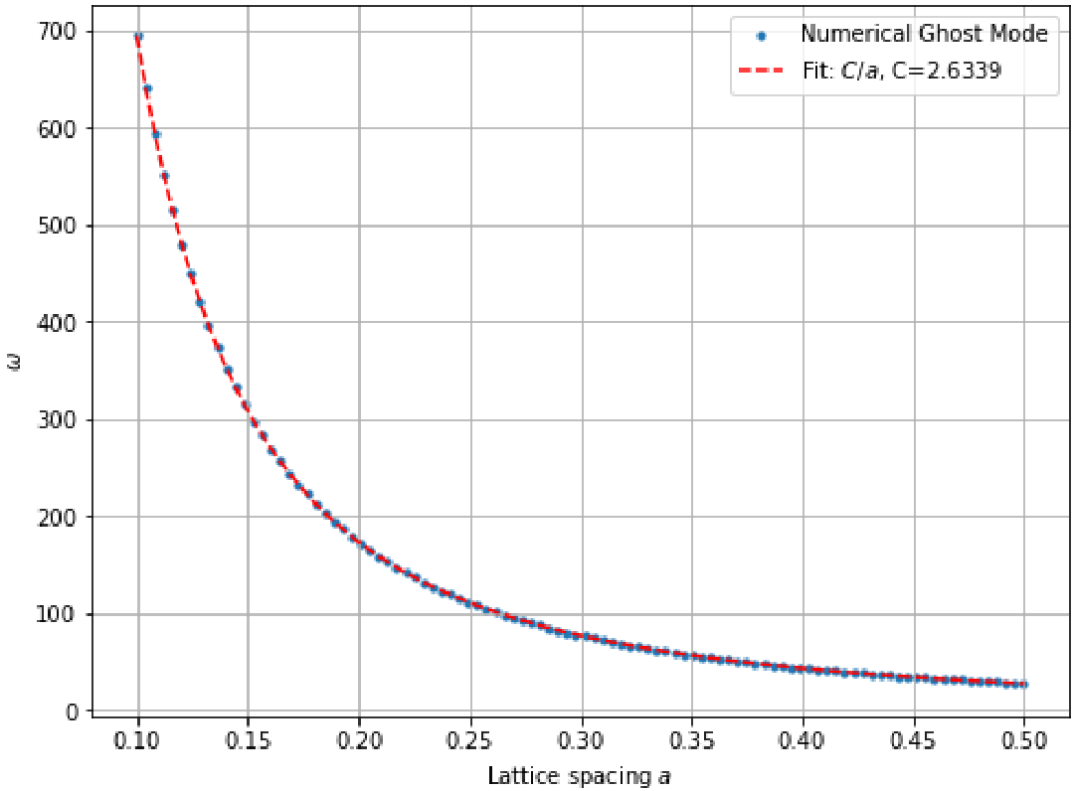


Harmonic Potential



This results are generated with the `build_matrix` and the `boundary_conditions` functions present in the `EOMSolver` class. They represent, with very good agreement, the ω^2 behaviour of a harmonic oscillator.

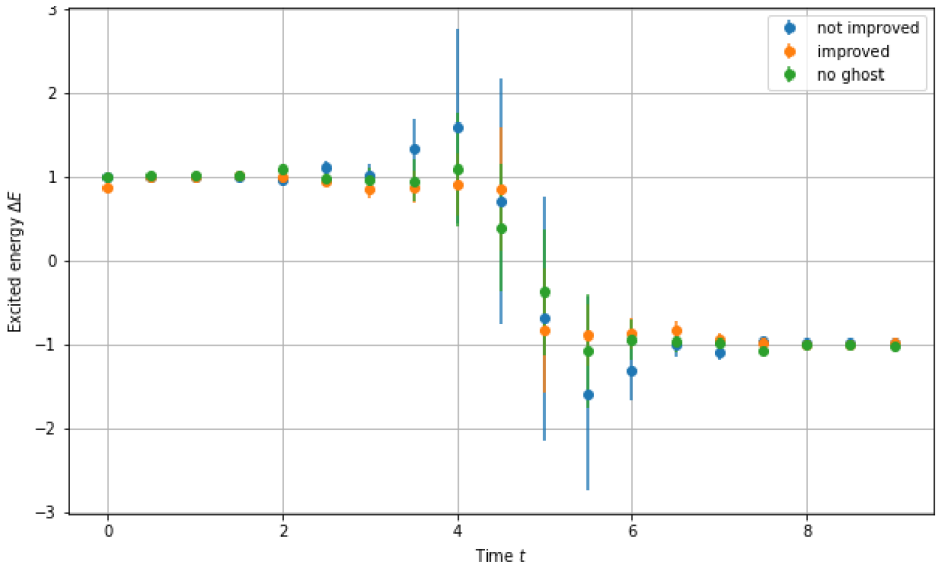
Ghost Mode Analysis



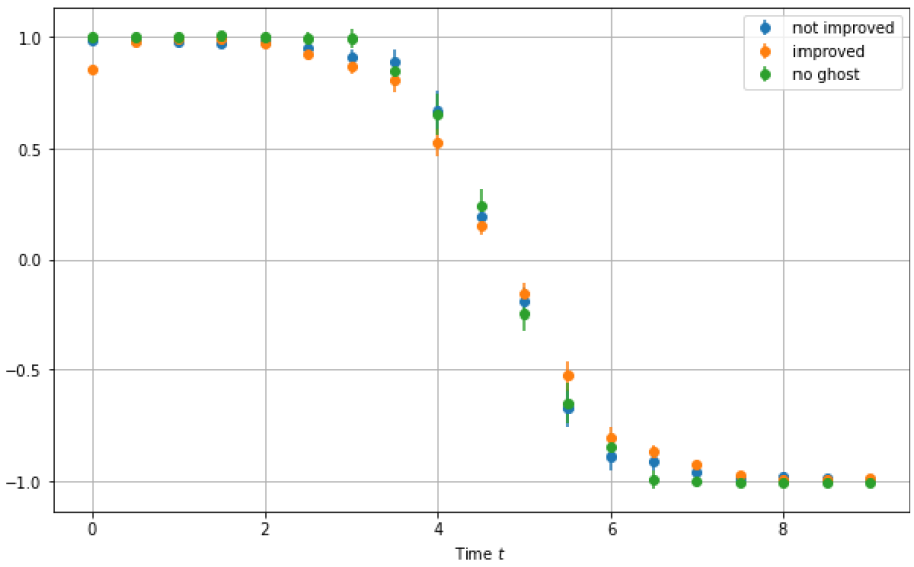
To further validate the discussion above we implemented the function `compute_ghost_mode`, in order to test the behaviour of "numerical ghost". The fit of the solution, in the form $\omega \approx C/a$, are in perfect agreement with the prediction, evaluating C as ≈ 2.6339

Improved ΔE Results

We can now apply the improved methods to the `PathIntegralMonteCarlo` class and obtain better results for the excitation energy.



$N_{cf} = 10^4$



$N_{cf} = 10^5$

Anharmonic Potential

Now we want to extend the procedures validated above to an Anharmonic Potential of form:

$$V_{an} = \frac{1}{2} m\omega^2 x^2 (1 + cm\omega_0 x^2) \tag{2.16}$$

Solving the equations of motion for this potential makes some complications explicit:

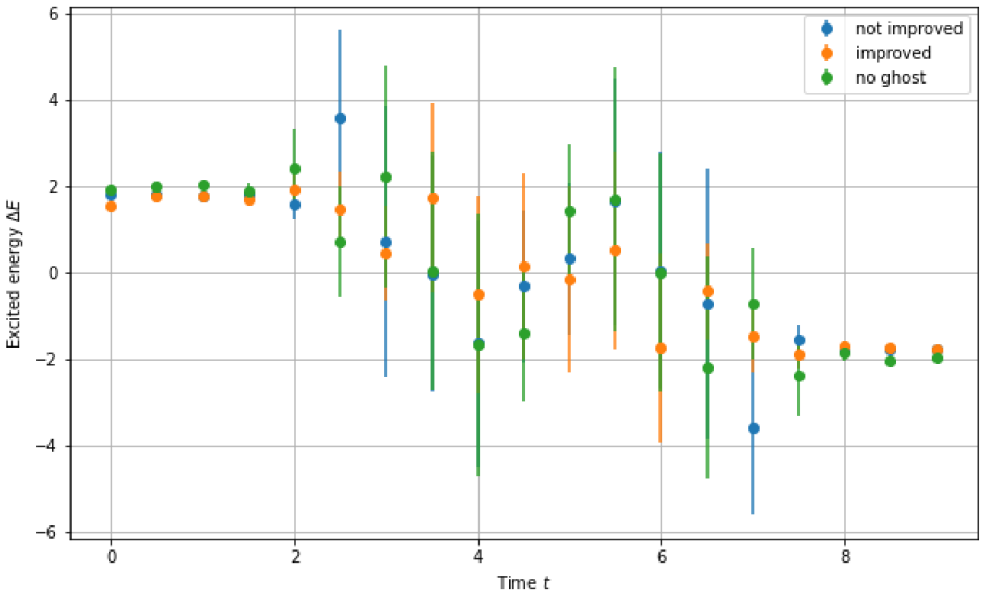
The x^3 dependence of this EoM impedes the use of a matrix-based solution,

Numerical ghost states can be eliminated by performing a change of variables, however this procedure results in an extra term in the $\tilde{V}_{imp}$.

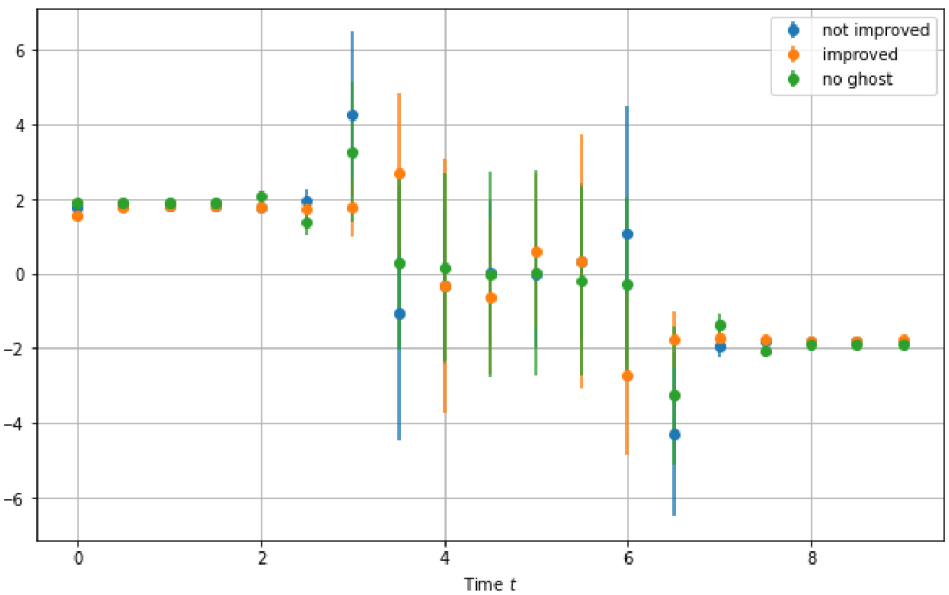
Even without a full analysis of this issues we can already compute the first excitation energy.

Anharmonic ΔE Results

Using the improved `PathIntegralMonteCarlo` class for the anharmonic potential (2.16) we obtain this results for the excitation energy.



$N_{cf} = 10^4$



$N_{cf} = 10^5$

Lattice QCD

Finally we attempt to generalize the methods applied to a point-like particle path to a $(3+1)D$ Field Theory, thus entering the realm of Lattice QCD.

We will follow the Euclidian gauge theories for lattice formulated by Wilson with some extra steps to numerically implement the path integral approach.

The most crucial step is the costruction of the QCD action in Euclidean space-time.

In the continuum, a fermion moving from x to y picks up a phase factor given by

$$\psi(y) = \mathcal{P}\, e^{\int_x^y igb_\mu(x)dx_\mu}\psi(x), \tag{3.1}$$

From (3.1) we can infer that the field is specified not by the sites, but by the variables on the links joining them. To discretize the path-ordered product of a fermion changing sites, maintaining gauge invariance, we reformulate the theory in terms of $SU(3)$ matrices.

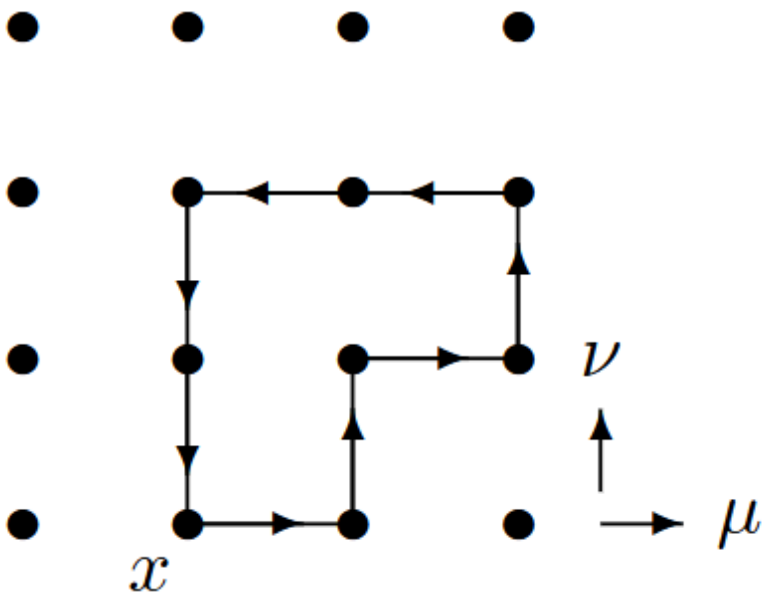
Wilson Loop

The link variables can be written as:

$$U(x, x+\mu) \equiv U_\mu(x) = e^{iagb_\mu\left(x+\frac{\mu}{2}\right)}. \tag{3.2}$$

From (3.2) we can construct a gauge-invariant quantity, the Wilson Loop.

$$W(\mathcal{C}) = \frac{1}{3}\mathrm{Tr}(U_\mu(x)U_\nu(x+a\mu)\ \ldots\ U_\nu^\dagger(x)). \tag{3.3}$$



Wilson Action

The simplest Wilson Loop is the $a \times a$ loop, defined by the plaquette operator :

$$W_{\mu\nu}^{1\times 1} \equiv P_{\mu\nu} = \mathrm{Re}\{\mathrm{Tr}(U_\mu(x)U_\nu(x+\mu)U_\mu^\dagger(x+\nu)U_\nu^\dagger(x))\}. \tag{3.4}$$

We can exploit the gauge-invariance of (3.4) to construct the gauge action in terms of closed loops.

$$S_{\mathrm{Wil}} = \int d^4x \sum_{\mu<\nu} \left\{ \frac{1}{2}\mathrm{Tr}G_{\mu\nu}^2 + \frac{a^2}{24}\mathrm{Tr}\left[G_{\mu\nu}(D_\mu^2 + D_\nu^2)G_{\mu\nu}\right] + \dots \right\}. \tag{3.5}$$

We can cancel the a^2 error in (3.5) by adding other Wilson loops.

In our case we add the $2a \times a$ loop via the rectangle operator :

$$W_{\mu\nu}^{2\times 1} \equiv R_{\mu\nu} = 1 - \frac{2}{3}a^4\mathrm{Tr}(gG_{\mu\nu})^2 - \frac{1}{18}a^6\mathrm{Tr}\left[gG_{\mu\nu}(4D_\mu^2 + D_\nu^2)gG_{\mu\nu}\right] - \dots \tag{3.6}$$

Improved Wilson Action

Combining the plaquette and rectangle operators we obtain the Improved Classical Lattice Action :

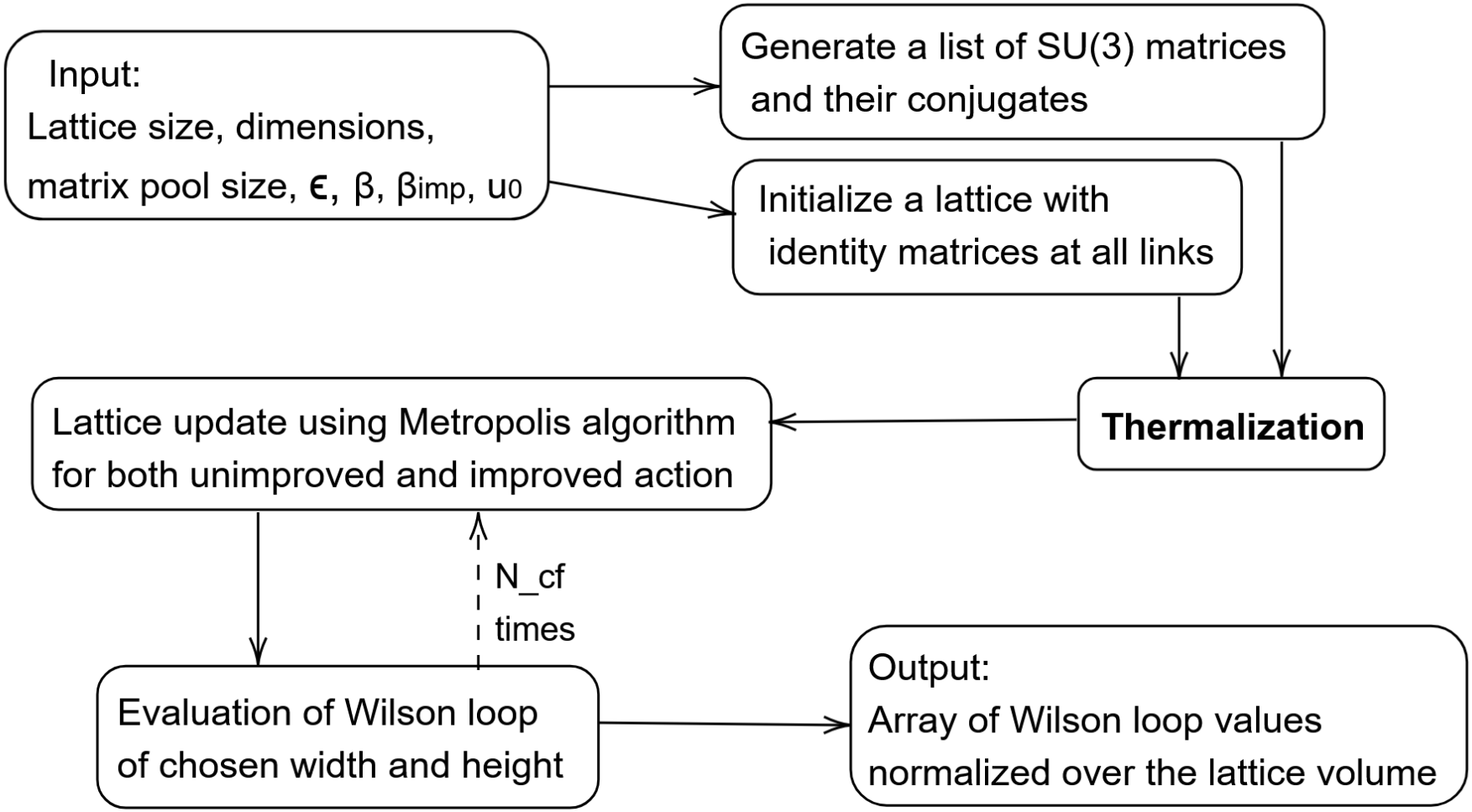
$$S \equiv -\frac{6}{g^2}\sum_x\sum_{\mu,\nu}\left\{\frac{5P_{\mu\nu}}{3} - \frac{R_{\mu\nu}+R_{\nu\mu}}{12}\right\} + \text{const} \tag{3.7}$$

$$= \int d^4x \sum_{\mu<\nu}\left\{\frac{1}{2}\text{Tr}G^2_{\mu\nu}\right\} + \mathcal{O}(a^4), \tag{3.8}$$

$$\text{with} \qquad G_{\mu\nu} = \partial_\mu b_\nu - \partial_\nu b_\mu + ig[b_\mu, b_\nu]. \tag{3.9}$$

To go from classical to quantum analysis of the lattice one must cancel tadpole contribution. This is achieved through tadpole improvement, the normalization of the gauge links with the mean link value, namely u_0, dependent only on lattice spacing a.

WilsonLatticeUtils.py



Code Implementation

In a gluonic path integral the fundamental variables are the link variables $U_\mu(x)$, which are updated using randomly generated $SU(3)$ matrices.

The Lattice is initialized as a *7D array*, where 4 dimensions correspond to space-time coordinates, 1 to μ directions and the remaining 2 store an identity matrix as initial condition of $U_\mu(x)$.

The Metropolis algorithm implemented in `WilsonLatticeUtils` multiply the link variables of the entire lattice by random matrices from the $SU(3)$ pool, leading to a *variation in Wilson Action*.

$$\Delta S(x,\mu) = \frac{\beta}{3u_0^4}\,\mathrm{Re}\{\mathrm{Tr}\,(U_\mu(x)\Gamma_\mu(x))\}, \tag{3.10}$$

$$\Delta S_{\mathrm{imp}}(x,\mu) = -\frac{\beta_{\mathrm{imp}}}{3}\left\{\frac{5}{3u_0^4}\,\mathrm{Re}\,\{\mathrm{Tr}\,(U_\mu(x)\Gamma_\mu(x))\} - \frac{1}{12u_0^6}\,\mathrm{Re}\left\{\mathrm{Tr}\left(U_\mu(x)\Gamma_\mu^R(x)\right)\right\}\right\}. \tag{3.11}$$

Wilson Loop results

At last, after setting the lattice size ($N = 8$) we compute Monte Carlo averages of some Wilson Loop.

	Unimproved a×a	Unimproved a×2a	Improved a×a	Improved a×2a
1	0.503784	0.258365	0.540964	0.277024
2	0.493801	0.258380	0.536787	0.283862
3	0.493650	0.255885	0.545100	0.283298
4	0.492839	0.262779	0.536370	0.278848
5	0.494994	0.258511	0.541172	0.283591
...				
20	0.494206	0.254229	0.539708	0.284301
Mean	0.494975	0.258150	0.539485	0.280911
σ	0.004015	0.003929	0.002439	0.002927

Confinement study

At last we want to use the Lattice QCD model to study charge confinement.

To do so we consider the potential energy between two static quark.

The Action of an external quark added in a gauge field is given by:

$$S_J = \int d^4x J^k_\mu(x) b^k_\mu. \tag{3.12}$$

The simplest ansatz for a confining potential is the Cornell Potential, i.e.

$$V(r) = \sigma r - \frac{b}{r} + V_0, \tag{3.13}$$

which exhibits both confinement and asymptotic freedom.

The term σr dominates at large r while the Coulomb term $\frac{b}{r}$ is relevant at small r.

We will obtain our results computing Wilson Loop along one spatial and one temporal dimension for a variety of T, and fit them to this type of curve.

Static Potential analysis

The majority of the methods needed are already present in `WilsonLatticeUtils`, therefore we implement the subclass `StaticPotentialeUtils`, adding the remaining methods required.

The new most crucial method performs the smearing procedure, it both improve the convergence rate and act as a momentum cut-off.

This procedure consists in replacing the link variables with a smeared counterpart:

$$\tilde{U}_\mu(x) \equiv (1 + \epsilon_{\text{sm}} a^2 \Delta^{(2)})^n U_\mu(x), \tag{3.14}$$

where $\Delta^{(2)} U_\mu(x)$ is the gauge covariant-derivative.

The function `smear_lattice()` is iterated 4 times over the entire lattice, and include also a projection procedure to ensure that the new matrices $\tilde{U}_\mu(x)$ belong to $SU(3)$.

Smearing procedure

```
def smear_lattice(self, lattice):
for t in range(self.N):
for x in range(self.N):
for y in range(self.N):
for z in range(self.N):
point = np.array([t, x, y, z])
for direction in range(1, self.dim):
smeared_link = lattice[t, x, y, z, direction]
+ self.smeared_eps * (self.a ** 2)
* self.gauge_covariant_derivative(lattice, point, direction)

#SU(3) projection process
U, S, Vh = np.linalg.svd(smeared_link)  # SVD
lattice[t, x, y, z, direction] = np.dot(U, Vh)

detU = np.linalg.det(lattice[t, x, y, z, direction])
lattice[t, x, y, z, direction] /= detU**(1/3)

return lattice
```

Static Potential results

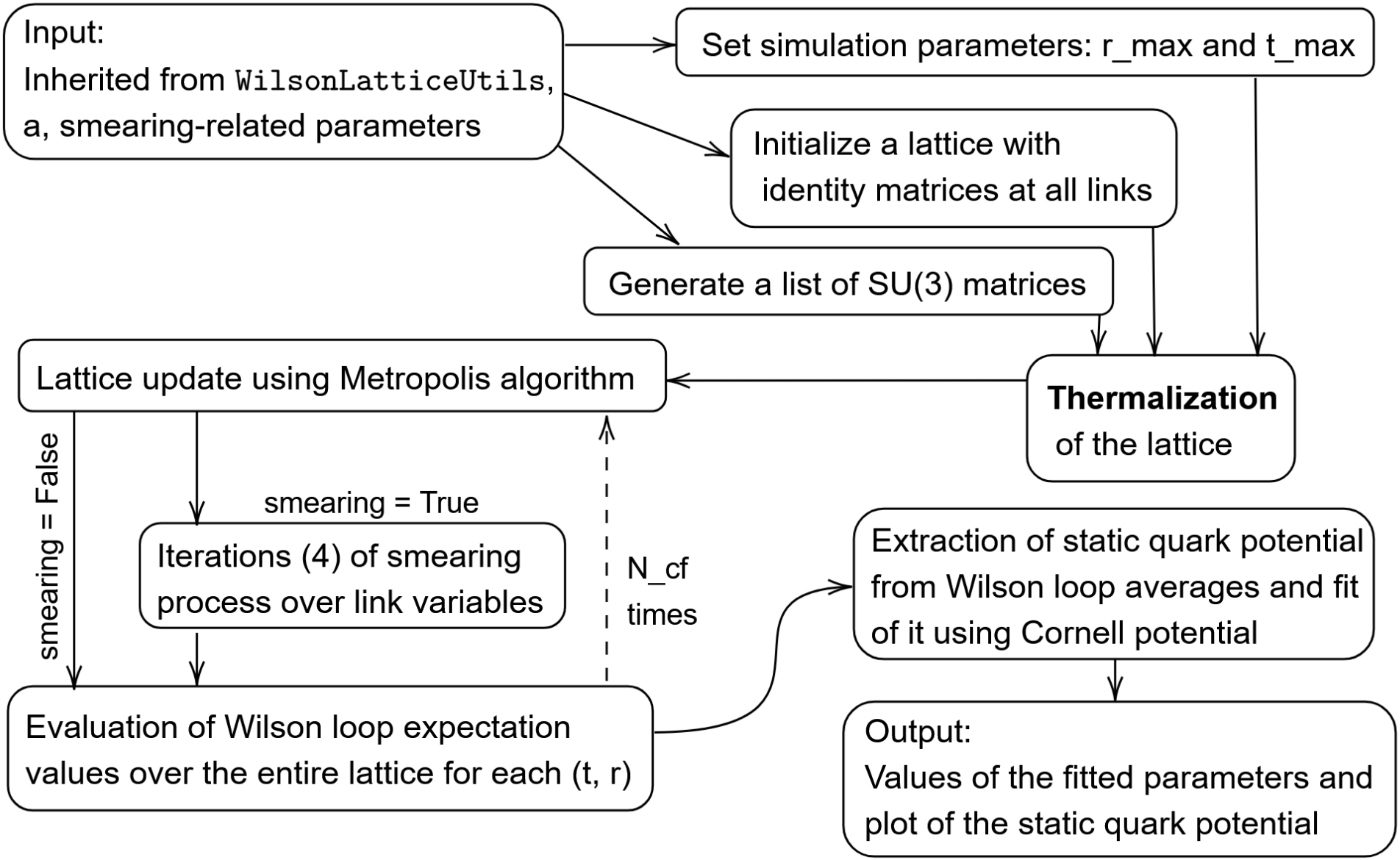
Finally, after evaluating the Wilson loops needed, we take the large T limit to compute the static potential as

$$aV(r) \approx \frac{W(r,T)}{W(r,T+a)}. \tag{3.15}$$

We iterate the computation for both action (3.10, 3.11) and for both the unsmeared and smeared case, fitting them to the Cornell Potential (3.13).

The fitted parameters and the plotted results are shown below.

StaticPotentialUtils.py

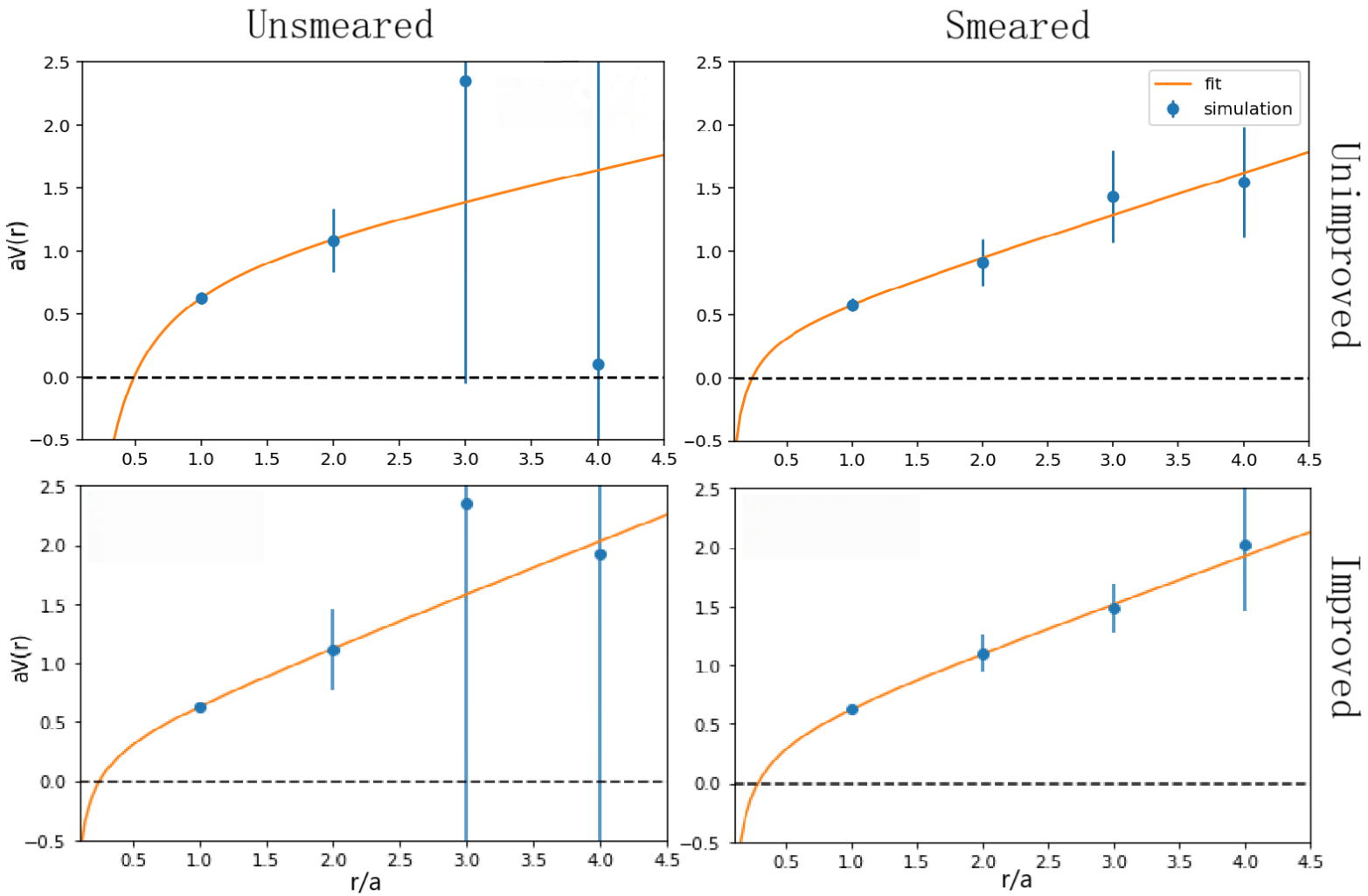


Static Potential results

Improved action	Smearing	Parameters Fit values
No	No	$\sigma = 0.2124(6) \pm 1.2342(6)$ $b = 0.5007(6) \pm 2.5241(0)$ $V_0 = 0.9161(2) \pm 3.7532(5)$
No	Yes	$\sigma = 0.3234(7) \pm 0.1595(7)$ $b = 0.1000(0) \pm 0.4465(3)$ $V_0 = 0.3499(2) \pm 0.5980(6)$
Yes	No	$\sigma = 0.4435(5) \pm 0.8645(2)$ $b = 0.1000(0) \pm 1.7413(6)$ $V_0 = 0.2839(4) \pm 2.6048(4)$
Yes	Yes	$\sigma = 0.3999(4) \pm 0.0800(0)$ $b = 0.1400(6) \pm 0.2165(1)$ $V_0 = 0.3627(9) \pm 0.2943(2)$

As expected the errors magnitude decrease in the smeared case and for the improved action. However the errors are still very large, probably due to the highly fluctuating behavior observed in the evaluation of the largest loops.

Static Potential plotted results



The plotted potential explicitly exhibit the expected confinement behaviour $V(r) \propto r$, showing also graphically the bigger uncertainty of the largest loops.

Fin