FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation:

Mathematics and physics / physics

The spring semester, 20.22.

Open / ~~Confidential~~

Henrik B. Hagen
(signature author)

Author: Henrik Bekkely Hagen

Course coordinator:

Supervisor(s): Alexander Karl Rothkopf

Thesis title:

Exploring modern simulation algorithms for complex Langevin

Credits (ECTS): 60

Keywords:

Markov chain Monte Carlo, Metropolis, Langevin, Complex Langevin

Pages: 59

+ appendix: Julia code

Stavanger, 06/2022
date/year

Vedtatt av dekan 30.09.21
Det teknisk-naturvitenskapelige fakultet

Exploring modern simulation algorithms for complex Langevin

Hagen, Henrik Bekkely

June 2022

Contents

Introduction

This thesis is a study of Markov chain Monte Carlo methods used to simulate a wide class of quantum systems. Markov chain Monte Carlo methods is a class of algorithms designed to sample from probability distributions according to observables in the simulated system. For many-body systems and systems with strong interactions, where analytical or approximation methods like perturbation theory break down, Markov chain Monte Carlo methods have proved themselves usefull. The samples generated from these methods can be used to estimate statistical expectation values for the system. The first chapter provides a introduction to the concepts that will be used to find analytical expectation values which will then be compared to the results from the Monte Carlo methods.

Out of the Markov chain Monte Carlo methods in this thesis, the Metropolis algorithm will be introduced first in Chapter 2. The Metropolis algorithm samples from a probability distribution which form is known up to a constant, by accepting or rejecting proposed samples based on their relative probability to the last accepted sample. By using the Euclidean path integral as an example, we estimate the expectation values for first and second moment. Since Markov chain Monte Carlo methods have inherent autocorrelation, a derivation and investigation is provided in this chapter. The thesis comes with the Julia code [1] which I, the author, has written and used in the development of the thesis.

From the data generated in Chapter 2, the One-Point function and Two-Point functions are introduced and computed in Chapter 3. The One-Point function for the harmonic oscillator system simulated gives an expectation for the modulo squared wave function of the system. From the Two-Point Correlation function expectation values about the energy levels of the system can be estimated.

In Chapter 4 another Markov chain Monte Carlo method, Langevin, is intro-

duced. The Langevin equation is derived and used to derive a corresponding Fokker-Planck equation for a simple system. The Fokker-Planck equation for a given Langevin equation expresses how the distribution the Langevin method samples from for different observables, evolves in time. If the Fokker-Planck arrives at a stationary distribution for an observable as the Langevin method is run to infinity, expectation values can be extracted directly. Since it is often very computationally expensive to find the Fokker-Planck equation for a complicated Langevin equation, expectation values for the simple system is also estimated by simulating with the Langevin method in Section 4.5.1.

The Metropolis method can only simulate systems with real actions. So in Section 4.5 the Complex Langevin equation is derived and used to simulate the simple system with a complex contribution to the action. In certain limits the samples generated by the Langevin and complex Langevin methods can start to run of to infinity. In Section 4.5.2 and 4.5.3 some solutions to this problem is explained more closely.

In Chapter 5 I use the code implemented by myself, and that of Alvestad et al. [2], to compare the efficiency of different stochastic differential equation solvers, which are an integral part of the Langevin and complex Langevin methods introduced earlier.

Chapter 1

Quantum mechanics

This chapter contains a short introduction to the quantum mechanical derivations needed to estimate the expectation values of the Euclidean path integral. The quantum harmonic oscillator and Euclidean time path integrals being the central topics.

The constant $\hbar$ will be set to 1 for most of the derivations and results, it can be reinserted by dimensional analysis.

1.1 Gaussian integrals

The next two sections are inspired by *Quantum Field Theory and the Standard Model* [3, p. 254–259]

For later reference, the integral over a gaussian is derived.

Consider a gaussian integral of the form:

$$\int_{-\infty}^{\infty} exp[-(ax^2 + bx + c)]dx \tag{1.1}$$

with $a, b, c, x \in \mathbb{R}$. Since e^{-c} is independent of x it can be moved outside the integral (will be omitted for now).

By completing the square

$$\begin{aligned} I &= \int_{-\infty}^{\infty} exp[-(ax^2 + bx)]dx = \int_{-\infty}^{\infty} exp[-a(x^2 + \frac{b}{a}x)]dx \qquad (1.2)\\ &= \int_{-\infty}^{\infty} exp\Big[-a\Big(x^2 + \frac{b}{a}x + \big(\frac{b}{2a}\big)^2\Big) + \frac{b^2}{2^2 a}\Big]dx \\ &= \int_{-\infty}^{\infty} exp\Big[-a\Big(x + \frac{b}{2a}\Big)^2 + \frac{b^2}{4a}\Big]dx \qquad (1.3) \end{aligned}$$

again the constant can be pulled ut, and then shifting $x \to x - \frac{b}{2a}$, which does not change the measure.

$$I = e^{\frac{b^2}{4a}} \int_{-\infty}^{\infty} e^{-ax^2} dx \tag{1.4}$$

Again temporarily omitting the constant, then doing the common gaussian integral trick, in which:

$$\begin{aligned}(I')^2 = \int_{-\infty}^{\infty} e^{-ax_1^2} dx_1 \int_{-\infty}^{\infty} e^{-ax_2^2} dx_2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-ax_1^2} e^{-ax_2^2} dx_1 dx_2 \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-a(x_1^2+x_2^2)} dx_1 dx_2 \end{aligned} \tag{1.5}$$

Changing to polar coordinates:

$$\int_0^{2\pi}\int_0^{\infty} re^{-ar^2} dr d\theta = 2\pi \int_0^{\infty} re^{-ar^2} dr = 2\pi\big(\frac{1}{2a} e^{-ar^2}\big)\Big|_{\infty}^{0} = \frac{\pi}{a} \tag{1.6}$$

Which implies that the original integral I':

$$I' = \int_{-\infty}^{\infty} e^{-ax^2} dx = \sqrt{(I')^2} = \sqrt{\frac{\pi}{a}} \tag{1.7}$$

And the full solution to the gaussian integral stated in the beginning 1.1 becomes:

$$\int_{-\infty}^{\infty} exp[-(ax^2 + bx + c)]dx = exp\Big[\frac{b^2}{4a} - c\Big]\sqrt{\frac{\pi}{a}} \tag{1.8}$$

Using a complex a means still that the real part must be more than zero, see Sec. 4.4.2.

1.2 The path integral

Following the derivations of [3, p. 255–256].
Considering the one-dimensional non-relativistic quantum harmonic oscillator with the Hamiltonian given by

$$\hat{H}(t) = \frac{\hat{p}^2}{2m} + V(\hat{x}, t). \tag{1.9}$$

where $\hat{H}, \hat{p}, \hat{x}$ are operators and t is time.

If the initial state $\big|i\rangle = \big|x_i\rangle$ is localized at x_i at time t_i, to compute the projection onto the final state $\langle f\big| = \langle x_f\big|$ localized at x_f at time t_f, and

assuming the system with Hamiltonian $\hat{H}$ being independent of time, the solution is given by the matrix element as

$$\langle f|i\rangle = \langle x_f|e^{-i(t_f-t_i)\hat{H}}|x_i\rangle. \tag{1.10}$$

setting $\hbar$ to 1. If instead $\hat{H}(t)$ is a smooth function of t, the solution is given in the limit of $\delta t \to 0$ of the matrix element broken down to n small time intervals δt, where $t_j = t_i + j\delta t$ and $t_n = t_f$. Then, expanding to first order in δt:

$$\langle f|i\rangle = \int dx_n ... dx_1 \langle x_f|e^{-i\hat{H}(t_f)\delta t}|x_n\rangle\langle x_n|...|x_2\rangle\langle x_2|e^{-i\hat{H}(t_2)\delta t}|x_1\rangle\langle x_1|e^{-i\hat{H}(t_1)\delta t}|x_i\rangle \tag{1.11}$$

Each matrix element evaluated by inserting a complete set of momentum eigenstates, and using $\langle p|x\rangle = e^{-ipx}$:

$$\begin{aligned}\langle x_{j+1}|e^{-i\hat{H}(t_{j+1})\delta t}|x_j\rangle &= \int \frac{dp}{2\pi}\langle x_{j+1}|p\rangle\langle p|e^{-i\left[\frac{\hat{p}^2}{2m}+V(\hat{x}_j,t_j)\right]\delta t}|x_j\rangle \\ &= e^{-iV(x_j,t_j)\delta t}\int \frac{dp}{2\pi}e^{-\left(\frac{i}{2m}p^2\delta t - i(x_{j+1}-x_j)p\right)}.\end{aligned} \tag{1.12}$$

And as was shown in section 1.1, Eq. 1.8, it is possible to evaluate this gaussian integral.
Setting $a = \frac{i}{2m}\delta t$, $b = -i(x_{j+1}-x_j)$ and $c = 0$:

$$\begin{aligned}\int \frac{dp}{2\pi}e^{-\left(\frac{i}{2m}p^2\delta t - i(x_{j+1}-x_j)p\right)} &= exp\Big[\frac{b^2}{4a}\Big]\frac{1}{2\pi}\sqrt{\frac{\pi}{a}} \\ &= exp\Big[\frac{-m(x_{j+1}-x_j)^2}{i2\delta t}\Big]\sqrt{\frac{m}{2\pi i\delta t}} \qquad (1.13)\\ &= \sqrt{\frac{m}{2\pi i\delta t}}e^{i\frac{m}{2}\delta t\frac{(x_{j+1}-x_j)^2}{\delta t^2}} \qquad (1.14)\end{aligned}$$

The matrix element becomes (still in the limit where δt is very small, such that $O(\delta t^2)$ can be omitted):

$$\langle x_{j+1}|e^{-i\hat{H}(t_{j+1})\delta t}|x_j\rangle = Ne^{-iV(x_j,t_j)\delta t}e^{i\frac{m}{2}\delta t\frac{(x_{j+1}-x_j)^2}{\delta t^2}} = Ne^{iL(x,\dot{x})\delta t} \tag{1.15}$$

where $N = \sqrt{\frac{m}{2\pi i\delta t}}$ is a normalization constant which usually is ignored.
The Lagrangian takes the form

$$L(x,\dot{x}) = \frac{m}{2}\frac{(x_{j+1}-x_j)^2}{\delta t^2} - V(x_j,t_j) \tag{1.16}$$

In the limit $\delta t \to 0$

$$L(x, \dot{x}) = \frac{1}{2} m \dot{x}^2 - V(x, t) \tag{1.17}$$

is the Lagrangian of the system, generated by the Legendre transformation from the Hamiltonian.
As the final steps, since an expression for the matrix elements is derived, the product in 1.11 reduces to

$$\langle f \big| i \rangle = N^n \int dx_n ... dx_1 e^{iL(x_n, \dot{x}_n)\delta t} ... e^{iL(x_1, \dot{x}_1)\delta t} \tag{1.18}$$

and when taking the limit $\delta t \to 0$, the exponentials combine into an integral over dt, and

$$\langle f \big| i \rangle = N \int_{x(t_i)=x_i}^{x(t_f)=x_f} \mathcal{D}x(t) e^{iS[x]} \tag{1.19}$$

where $\mathcal{D}x$ stands for a sum over the paths with correct boundary conditions, and the action is $S[x] = \int dt \mathcal{L}[x(t), \dot{x}(t)]$. Note, as $\delta t \to 0$, $n \to \infty$, and the redefined N (N^n) is formally infinite.

The final result of Eq. 1.19 shows that, integrating over different paths with a complex phase, enables different paths to constructively or destructively interfere. And it is only the final sum, the superposition of these possible states, that gives the expectation value.

1.3 Euclidean time Path integrals

The path integral in the previous section (Eq. 1.19) gives an action multiplied by i, which does not give a well defined probability measure for a Metropolis algorithm. Instead, moving to Euclidean time $\tau = it$, one can do the Euclidean time Path integral, which in the end returns a real exponent. This can be used to define a well defined probability measure to use in a Metropolis update algorithm.

Following the derivations and explanations of [4]. For imaginary time path integrals there are two main applications. Firstly, in statistical mechanics one can use τ as $\frac{\hbar}{k_B T}$, such that for equal initial and final positions x, the partition function for the system can be evaluated:

$$Z = \int \langle x | e^{-\hat{H}\tau/\hbar} | z \rangle = Tr(e^{-\hat{H}\tau/\hbar}) \tag{1.20}$$

Which can be used in Markov chain Monte Carlo methods to estimate statistical expectation values of the system. As an example a system of a massive

particle in an idealized fluid can be used to simulate Brownian motion in this way.

The second main application, which will be the application in this thesis, is the energy spectrum of a quantum system. By using the identity $1 = \sum_n |n\rangle\langle n|$, where n is eigenfunctions of $\hat{H}$ that give the energies E_n, when acted upon by the Hamiltonian, $\hat{H}|n\rangle = E_n|n\rangle$, the partition function can be written as:

$$Z = \int \langle x|e^{-\hat{H}\tau}|x\rangle dx = \int \sum_n \langle x|e^{-\hat{H}\tau}|n\rangle\langle n|x\rangle dx \tag{1.21}$$

having set $\hbar$ to 1.
By using normalized wavefunctions $\psi_n(x)$ it can be re-expressed as:

$$Z = \int \sum_n e^{-E_n\tau}\psi_n(x)\bar{\psi}_n(x)dx = \sum_n e^{-E_n\tau} \tag{1.22}$$

The propagator $\langle x_f|x_i\rangle$, where $|x_i\rangle$ is the initial state i at time t_i and $\langle x_f|$ is the final state f at time t_f, can be expanded in terms of the eigenfunctions $|n\rangle$:

$$\langle x_f|x_i\rangle = \sum_{n=0}^{\infty} e^{-E_n(t_f-t_i)}\langle f|n\rangle\langle n|i\rangle \tag{1.23}$$

Then to derive the Euclidean time path integral, the same procedure as with the path integral previously derived are followed. The imaginary element i is multiplied by the factor $-i$ from the change $t \to -i\tau$, and the matrix element corresponding to 1.15 becomes

$$\begin{aligned}\langle x_{j+1}\big|e^{-\hat{H}(t_{j+1})\delta t}\big|x_j\rangle &= Ne^{-V(x_j,t_j)\delta t}e^{-\frac{m}{2}\delta t\frac{(x_{j+1}-x_j)^2}{\delta t^2}} \\ &= Ne^{-\delta t\left(\frac{m}{2}\frac{(x_{j+1}-x_j)^2}{\delta t^2}+V(x_j,t_j)\right)}\end{aligned} \tag{1.24}$$

where $N = \sqrt{\frac{m}{2\pi\delta\tau}}$. In the limit of infinite time intervals as $\delta\tau \to 0$, the partition function can be expressed as

$$Z = Tr(e^{-\hat{H}(\tau_f-\tau_i)} = \int_{x(\tau_i)=x_i}^{x(\tau_f)=x_f} \mathcal{D}x(\tau)e^{-S[x]} \tag{1.25}$$

where the Euclidean action, S, over a path $x(\tau)$ from τ_i to τ_f is

$$S = \int_{\tau_i}^{\tau_f} L(x(\tau))d\tau = \int_{\tau_i}^{\tau_f} \left[\frac{m}{2}\left(\frac{dx}{d\tau}\right)^2 + V(x(\tau))\right]d\tau \tag{1.26}$$

The exponent in Eq. 1.25 is real (choosing τ as new time). The Metropolis algorithm can therefore be used to estimate expectation values from many systems of this form, since there exist a well defined probability measure.

1.4 The harmonic oscillator in quantum mechanics

Here will the expectation values for the harmonic oscillator be solved for algebraically, following the derivations of Ch. 2.3.1 [5, p. 39–44].
The Hamiltonian for the harmonic oscillator can be expressed on the form:

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2\hat{x}^2 \tag{1.27}$$

where $\hat{x}$ and $\hat{p}$ are the position and momentum operator, and ω is the natural frequency. Defining the ladder operators

$$\hat{a}_\pm = \frac{1}{\sqrt{2\hbar m\omega}}(\mp i\hat{p} + m\omega\hat{x}) \tag{1.28}$$

for which the product is

$$\hat{a}_-\hat{a}_+ = \frac{1}{2\hbar m\omega}[\hat{p}^2 + (m\omega\hat{x})^2] - \frac{i}{2\hbar}[\hat{x}, \hat{p}] \tag{1.29}$$

where the commutator of $\hat{x}$ and $\hat{p}$:

$$[\hat{x}, \hat{p}] = i\hbar \tag{1.30}$$

This allows for the Hamiltonian to be rewritten in terms of the ladder operators:

$$\hat{H} = \hbar\omega\left(\hat{a}_+\hat{a}_- + \frac{1}{2}\right) \tag{1.31}$$

since $[\hat{a}_-, \hat{a}_+] = 1$. To find the normalized ground state wave function of the harmonic oscillator, consider the wave function ψ_0 that obeys:

$$\hat{a}_-\psi_0 = \frac{1}{\sqrt{2\hbar m\omega}}(i\hat{p} + m\omega\hat{x})\psi_0 = 0 \tag{1.32}$$

$$\implies \frac{d\psi_0}{dx} = -\frac{m\omega}{\hbar}x\psi_0 \tag{1.33}$$

The solution of this differential equation for ψ_0 is:

$$\psi_0(x) = Ae^{-\frac{m\omega}{2\hbar}x^2} \tag{1.34}$$

Normalizing $\int_{-\infty}^{\infty}|\psi_0|^2 = 1$, (For the Gaussian integral see Sec. 1.1):

$$1 = |A|^2\int_{-\infty}^{\infty}e^{-\frac{m\omega}{\hbar}x^2}dx = |A|^2\sqrt{\frac{\pi\hbar}{m\omega}} \tag{1.35}$$

The normalization constant A is then $(\frac{m\omega}{\pi\hbar})^{\frac{1}{4}}$. And the harmonic oscillator normalized ground state wave function becomes:

$$\psi_0 = \Big(\frac{m\omega}{\pi\hbar}\Big)^{1/4} e^{-\frac{m\omega x^2}{2\hbar}} \tag{1.36}$$

Where expectation values can be computed as:

$$\langle x_n \rangle = \int_{-\infty}^{\infty} \psi_0^* x_n \psi_0 dx \tag{1.37}$$

The expectation values for the first and second order of x:

$$\begin{aligned} \langle x \rangle = \int_{-\infty}^{\infty} \psi_0^* x \psi_0 dx &= \int_{-\infty}^{\infty} \sqrt{\frac{m\omega}{\pi\hbar}} e^{-\frac{m\omega x^2}{2\hbar}} x e^{-\frac{m\omega x^2}{2\hbar}} dx \\ &= \sqrt{\frac{m\omega}{\pi\hbar}} \int_{-\infty}^{\infty} e^{-\frac{m\omega x^2}{\hbar}} x dx = \sqrt{\frac{m\omega}{\pi\hbar}} \frac{\hbar}{2m\omega} \big(0 - 0\big) \\ &= 0 \end{aligned} \tag{1.38}$$

$$\begin{aligned} \langle x^2 \rangle = \int_{-\infty}^{\infty} \psi_0^* x^2 \psi_0 dx &= \sqrt{\frac{m\omega}{\pi\hbar}} \int_{-\infty}^{\infty} e^{-\frac{m\omega x^2}{\hbar}} x^2 dx \\ &= \sqrt{\frac{m\omega}{\pi\hbar}} \frac{\hbar}{2m\omega} \sqrt{\frac{\pi\hbar}{m\omega}} \\ &= \frac{\hbar}{2m\omega} \end{aligned} \tag{1.39}$$

Which, by setting $\hbar = 1$, becomes $\langle x^2 \rangle = \frac{1}{2m\omega}$.

1.5 The discretized harmonic oscillator

To do simulations on a quantum system, consider the Euclidean time harmonic oscillator Hamiltonian for a particle of mass m, and natural frequency $\omega = \sqrt{k/m}$, where k is the force constant:

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{m\omega^2 \hat{x}^2}{2}. \tag{1.40}$$

To simulate the system, it is necessary to discretize the system into N_τ slices of euclidean time $\tau = it$. By integrating over a small constant time step $d\tau$ in euclidean time and expressing it for each time slice i:

$$L_i = \int_{\tau_i}^{\tau_f} \frac{\hat{p}^2}{2m} + \frac{m\omega^2 \hat{x}^2}{2} d\tau = \frac{m}{2} \Big(\frac{x_{i+1} - x_i}{\delta\tau} \Big)^2 + \frac{m\omega^2 x_i^2}{2} \tag{1.41}$$

This is then the discretized Euclidean Lagrangian for the system:

$$S = \sum_{i=1}^{N_\tau - 1} L_i \tag{1.42}$$

The metropolis algorithm proposes a change to a single coordinate of this path through euclidean time, and accepts or rejects the proposal based on the change in action. To make the algorithm efficient, it is beneficial to only calculate the part of the sum that changes when a single coordinate, x_j, changes.

In the simulation, a periodic boundary condition is imposed, such that the proposal step goes through all $i = (1, N_\tau)$, with $i = N_\tau + 1$ being the same as $i = 1$. This leads to the sum of the action time-slices: $S = \sum_{i=1}^{N_\tau} L_i$

From [4, p. 8], expanding out the dimensionless variables, the discretized action is:

$$\begin{aligned} S &= \sum_{i=1}^{N_\tau} \Big[\frac{1}{2}\tilde{m}(\tilde{x}_{i+1} - \tilde{x}_i)^2 + \frac{1}{2}\tilde{m}\tilde{\omega}^2\tilde{x}_i^2\Big] && (1.43) \\ &= \sum_{i=1}^{N_\tau} \Big[\frac{1}{2}\delta\tau m\Big(\frac{x_{i+1} - x_i}{\delta\tau}\Big)^2 + \frac{1}{2}\delta\tau^3 m\omega^2 x_i^2/\delta\tau^2\Big] \\ &= \frac{m}{2}\delta\tau \sum_{i=1}^{N_\tau} \Big[\Big(\frac{x_{i+1} - x_i}{\delta\tau}\Big)^2 + \omega^2 x_i^2\Big] && (1.44) \end{aligned}$$

Then, finding the part of the action that depends on only a single x_j:

$$\begin{aligned} S(x_j) &= \frac{m}{2}\delta\tau \sum_{i=1}^{N_\tau} \Big[\Big(\frac{x_{i+1} - x_i}{\delta\tau}\Big)^2 \delta_{i,j} + \omega^2 x_i^2 \delta_{i,j}\Big] \\ &= \frac{m}{2}\delta\tau \sum_{i=1}^{N_\tau} \Big[\Big(\frac{x_{i+1}^2 - 2x_{i+1}x_i + x_i^2}{\delta\tau^2}\Big)\delta_{i,j}\Big] + \frac{m}{2}\delta\tau\omega^2 x_j^2 \\ &= \frac{m}{2}\delta\tau\Big[\frac{x_j^2 - 2x_j x_{j-1} - 2x_{j+1}x_j + x_j^2}{\delta\tau^2} + \omega^2 x_j^2\Big] \\ &= \frac{m}{\delta\tau}\big[x_j^2 - x_j(x_{j+1} + x_{j-1})\big] + \frac{m}{2}\delta\tau\omega^2 x_j^2 && (1.45) \end{aligned}$$

where $\delta_{i,j}$ is the Kronecker delta, which is 1 when $i = j$, and 0 otherwise. Taking the sum over j will give back the full discretized action.

To calculate changes in the action, only

$$\begin{aligned}\delta S(x_j) &= \frac{m}{\delta\tau}\left[(x_j')^2 - x_j'(x_{j+1}+x_{j-1})\right] + \frac{m}{2}\delta\tau\omega^2(x_j')^2 \\ &- \left(\frac{m}{\delta\tau}\left[x_j^2 - x_j(x_{j+1}+x_{j-1})\right] + \frac{m}{2}\delta\tau\omega^2 x_j^2\right) \\ &= \frac{m}{\delta\tau}\Big[(x_j')^2 - x_j^2 + (x_j - x_j')(x_{j+1}+x_{j-1})\Big] \\ &+ \frac{m}{2}\delta\tau\omega^2\Big((x_j')^2 - x_j^2\Big) \end{aligned} \tag{1.46}$$

has to be calculated, where x_j' is the proposed new x_j.

For this system it is possible to compute analytically the expectation values. Since this is a harmonic oscillator with no shift in the centering (See 1.4):

$$\langle x\rangle = 0 \tag{1.47}$$

So every x_i for $i = (1, 2, ..., N_\tau)$ the mean of saved configurations will approach 0 for many samples created by the Metropolis algorithm in Ch. 2. The distribution around 0 will also be symmetric for the harmonic oscillator action.

The expectation values $\langle x^2\rangle$ and $\langle x^4\rangle$ can be computed by the transfer operator $\hat{T}$. Following derivations of appendix C through H in [4]. Since the periodic discretized action for this system is given in Eq. 1.44, the discretized path integral is given by:

$$Z = \int \prod_{j=1}^{N_\tau} dx_j e^{-\frac{m}{2}\delta\tau\sum_{i=1}^{N_\tau}\left[\left(\frac{x_{i+1}-x_i}{\delta\tau}\right)^2 + \omega^2 x_i^2\right]} \tag{1.48}$$

The transfer operator $\hat{T}$ is then defined by its matrix elements between its position eigenstates,

$$\langle x'|\hat{T}|x\rangle = e^{-\frac{m}{2\delta\tau}(x'-x)^2 - \frac{m\omega^2\delta\tau}{4}(x^2+(x')^2)} \tag{1.49}$$

The transfer operator describes the deviations of the S matrix from the free theory.
By imposing that for the periodic boundary conditions $Z = Tr(\hat{T}^{N_\tau})$, and computing the commutator of $\hat{T}$ with $\hat{x}$ and $\hat{p}$. The commutator of the lowering operator $\hat{a} = \frac{1}{\sqrt{2m\omega}}(\hat{p} - im\omega\hat{x})$ (defined in Eq. (G1a) [4]) (in Euclidean time) with the transfer operator

$$[\hat{a}, \hat{T}] = \hat{T}\hat{a}\Big(\frac{\delta\tau^2\omega^2}{2} - \delta\tau\omega\Big(1 + \frac{\delta\tau^2\omega^2}{4}\Big)^{1/2}\Big) \tag{1.50}$$

Which can be equivalently written as:

$$\hat{a}\hat{T} - \hat{T}\hat{a} = \hat{T}\hat{a}\Big(\frac{\delta\tau^2\omega^2}{2} - \delta\tau\omega\Big(1 + \frac{\delta\tau^2\omega^2}{4}\Big)^{1/2}\Big) \tag{1.51}$$

$$\hat{a}\hat{T} = \hat{T}\hat{a}\Big(1 + \frac{\delta\tau^2\omega^2}{2} - \delta\tau\omega\Big(1 + \frac{\delta\tau^2\omega^2}{4}\Big)^{1/2}\Big) = \hat{T}\hat{a}R, \tag{1.52}$$

for the constant R from Eq. (G11) in [4, p. 35].

$$R = 1 + \frac{\delta\tau^2\omega^2}{2} - \delta\tau\omega\sqrt{1 + \frac{\delta\tau^2\omega^2}{4}} \tag{1.53}$$

Since the first element of the Two-Point Correlation Function (Sec. 3.2) is the same as the expectation value for x^2:

$$\langle x^2\rangle = \langle x_i x_i\rangle_i \tag{1.54}$$

$$\langle x^2\rangle = \frac{1}{2m\omega}\Big(\frac{1 + R^{N_\tau}}{1 - R^{N_\tau}}\Big) \tag{1.55}$$

from Eq. (H8b) in [4, p. 37]. The variance depends on the lattice spacing, m, ω, and the number of points in the lattice N_τ. Which is intuitive since $\delta\tau$ and N_τ also determines the temperature of the system, therefore $\langle x^2\rangle$. The inverse temperature β being:

$$\beta = \delta\tau N_\tau \tag{1.56}$$

The solution of $\langle x^2\rangle$ converges to a different solution because of the discretization, but as $\delta\tau \to 0$, the true solution is recovered. For $\delta\tau > 0$, the correcting factor $\sqrt{1 + \delta\tau^2\omega^2/4}$ is included in the denominator:

$$\langle x^2\rangle = \frac{1}{2m\omega\sqrt{1 + \delta\tau^2\omega^2/4}}\Big(\frac{1 + R^{N_\tau}}{1 - R^{N_\tau}}\Big) \tag{1.57}$$

From Eq. (42) in [4, p. 8].

1.6 The discretized anharmonic oscillator

To test the methods explained here, it would be useful to see if problems without analytic solutions give reasonable results. Since this is the main purpose of the methods, to solve problems that are impossible or hard analytically.

To simulate a system which does not have an analytical solution, consider discretizing the anharmonic oscillator action:

$$S_{AHO} = \int_{t_i}^{t_f}\Big[\frac{m}{2}\Big(\frac{dx}{dt}\Big)^2 + V(x(t))\Big]dt, \qquad V(x(t)) = \frac{m}{2}\omega^2x^2 + \frac{\lambda}{4!}x^4 \tag{1.58}$$

$$S = \frac{m}{2}\delta\tau \sum_{i=1}^{N_\tau} \Big[\Big(\frac{x_{i+1} - x_i}{\delta\tau} \Big)^2 + \omega^2 x_i^2 \Big] + \sum_{i=1}^{N_\tau} \frac{\lambda}{4!} \Big(\frac{x_i}{\delta\tau} \Big)^4 \tag{1.59}$$

The first sum is just Eq. 1.44.
Finding the parts of the action depending on x_j:

$$S(x_j) = \frac{m}{2}\delta\tau \sum_{i=1}^{N_\tau} \Big[\Big(\frac{x_{i+1} - x_i}{\delta\tau} \Big)^2 \delta_{i,j} + \omega^2 x_i^2 \delta_{i,j} \Big] + \sum_{i=1}^{N_\tau} \frac{\lambda}{4!} \Big(\frac{x_i}{\delta\tau} \Big)^4 \delta_{i,j} \tag{1.60}$$

$$= \frac{m}{\delta\tau} \big[x_j^2 - x_j(x_{j+1} + x_{j-1}) \big] + \frac{m}{2}\delta\tau\omega^2 x_j^2 + \frac{\lambda}{24} \frac{x_j^4}{\delta\tau^4} \tag{1.61}$$

And so the difference in action by a change in x_j will be:

$$\begin{aligned} \delta S(x_j) = \frac{m}{\delta\tau} \Big[(x_j')^2 - x_j^2 + (x_j - x_j')(x_{j+1} + x_{j-1}) \Big] \\ + \frac{m}{2}\delta\tau\omega^2 \Big((x_j')^2 - x_j^2 \Big) + \frac{\lambda}{24} \frac{(x_j')^4 - x_j^4}{\delta\tau^4} \end{aligned} \tag{1.62}$$

The anharmonic action induces a skewness to the probability distribution of the samples x_j, meaning the distribution no longer will be symmetric around 0 for $\lambda \neq 0$.

In the limit $\lambda \to 0$ the action becomes the harmonic oscillator action, and the previously derived statistical expectations of the probability distribution holds.

Chapter 2

Metropolis update algorithm

This chapter provides an introduction to the simulation algorithm Metropolis-Hastings, and gives an example simulating the harmonic oscillator in Euclidean time. In part inspired by [4].

The configurations saved from a Metropolis-Hastings algorithm are inherently correlated. To estimate the correct error-bars a section on autocorrelation is provided in 2.4.

2.1 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a class of algorithms used to sample configurations from a probability distribution p, which form, f, is known up to a constant.

The MCMC method introduced in this chapter is the Metropolis algorithm, a less general case of the Metropolis-Hastings algorithm. In the Metropolis algorithm an initial condition, a sample from the domain of p, is drawn at random or chosen. New samples are drawn randomly according to the distribution f, with the same form as p up to a constant, and are accepted or rejected with relative probability to the previous sample. After an initial burn in time, the samples are drawn according to the target distribution p. The Metropolis algorithm is an efficient algorithm for sampling from high dimensional probability distributions, relative to simple accept/reject sampling.

Later, in Ch. 4, the Langevin and Complex Langevin methods are introduced. New samples are drawn by adding noise to the drift term of the action of the harmonic oscillator. The drift term pushes the action towards the classical limit, while the appropriate noise makes the sampled configurations follow the statistical probability distribution of the quantum harmonic

oscillator.

The complex Langevin method makes it possible to simulate certain complex actions.

2.2 Metropolis-Hastings algorithm

The metropolis-hastings algorithm is a Markov chain Monte Carlo method of sampling from a distribution, $p(x)$, which form $f(x)$ is known up to a constant.

$$p(x) = N_1 f(x) \tag{2.1}$$

Starting from a random initial configuration, the algorithm generates a new configuration based on a proposal step, and an accept/reject step.

In the Metropolis-Hastings algorithm, a new configuration is randomly generated by modifying the previous sample. The algorithm goes through each element of the existing sample, and proposes changes to the element. If the change generates a relatively more probable configuration it is accepted, if the proposed configuration is relatively less probable it is accepted with a probability. If the change is rejected, the element remains as it was. The generation of proposed changes and the accept reject step, when done for each element in the configuration, constitutes one "sweep", a Monte Carlo step for the whole configuration. After each sweep the new configuration is saved as a sample from the distribution.
As the name Markov chain Monte Carlo suggests, this generates a Markov chain of configurations:

$$a_i \to a_1 \to a_2 \to a_3 \to ... \tag{2.2}$$

This implies that each new configuration depends on the previous configuration.
The algorithm creates new configurations such that it satisfies the condition:

$$T(x_i \to x_i')p(x_i) = R(x_i' \to x_i)p(x_i') \tag{2.3}$$

where the transition probability T is the probability of x_i' being generated from x_i and then accepted. It is crucial for the simulation that the transition operator allows for samples to change, i.e not the identity operation $\mathbb{I}$, a delta peak at x_i. For the algorithm to work it has to give the procedure the possibility to explore the whole configuration-space. The choice of T is important for the procedure to satisfy ergodicity. It is also possible to choose different $T(x_i \to x_i')$ for different steps $a_j \to a_{j+1}$ as long as they satisfy the conditions prescribed bellow.

If the transition probability T and R is the same, then the stronger detailed balance condition is met:

$$T(x_i \to x_i')p(x_i) = T(x_i' \to x_i)p(x_i') \tag{2.4}$$

So that the transition amplitude of going from x_i to x_i', times the probability density at x_i, is the same as the transition amplitude of going back, times the probability density at x_i'. The detailed balance condition implies that after the Markov chain has thermalized, it will sample from a stationary distribution, namely the target distribution:

$$\sum_{x_i} T(x_i \to x_i')p(x_i) = p(x_i') \tag{2.5}$$

and new samples x_i' can be considered as drawn from $p(x)$.

The first step in the algorithm generates a random element, η_i, from a distribution, D, centered at the previous element of $f(x)$. Which is the same as sampling from the distribution centered at 0, for example a gaussian distribution with finite variance σ^2:

$$D = N(0, \sigma^2) \tag{2.6}$$

and then adding this change to the previous element:

$$x_i' = x_i + \eta_i. \tag{2.7}$$

The new sample from f(x) is proposed and accepted based on its acceptance probability, $A(x_i \to x_i')$. The transition amplitude then becomes:

$$T(x_i \to x_i') = g(x_i'|x_i)A(x_i \to x_i') \tag{2.8}$$

where $g(x_i'|x_i)$ is the probability of proposing the new element, x_i', given the current element, x_i.

The choice of σ^2 of η_i is very important to the efficiency of the algorithm. If the variance of the random number is too large, the accept/reject step will throw away many proposals, and the Markov chain will for many proposal-accept/reject steps (sweeps) stay the same. It is not detrimental to the algorithm as long as any proposals are accepted every once in a while, as the probability to picking a configuration in the chain will approach the probability of picking a configuration from the target distribution as the length of the chain approaches infinity. More specifically, since only pseudo-random numbers and float precision is considered, the generator with the given seed can not repeat before a new sample is accepted, with the new

sample being the same as one already in the chain, and the generator having the same state it had after generating that sample.

If the variance of the random number is too low, many proposals close to the dense parts of the distribution f will be accepted right away, but the chain will be heavily autocorrelated (see Sec. 2.4), and it will take a long time for the simulation to sample in the low density parts of f. The general shape of parts of the distribution will be sampled after some time, but for the chain to sample from all low-density parts, and move between high density parts separated by low density, many more samples than necessary has to be accepted to the chain.

To avoid these two extremes a preset ideal acceptance rate *idrate* is chosen. The random number with finite variance σ^2 is multiplied by h, initially set to 1 (or some other finite, non-zero value). After each sweep, where at least one proposal was accepted, h is updated to approach a value that makes the acceptance rate closer to the ideal. Written in Julia, the update is as follows:

```
h *= accepted_rate/idrate
```

where h is now multiplied by the ratio of the real and ideal rate. The proposals created will then be accepted at a rate close to *idrate*, and new configurations are added to the chain regularly. The ideal accept rate varies for different systems, often somewhere between 0.4 and 0.8. In [4] an ideal accept rate of 0.8 is used, although the authors mention that a lower accept rate might be more efficient for the harmonic oscillator simulated there. A Metropolis algorithm run with 20'000 sweeps, with the update of h above and $idrate = 0.8$, returned a mean accept rate of 0.806.

From the detailed balance condition, Eq. 2.4, using Eq. 2.8:

$$\frac{A(x_i \to x_i')}{A(x_i' \to x_i)} = \frac{f(x_i')}{f(x_i)} \frac{g(x_i|x_i')}{g(x_i'|x_i)} = f_i g_i \tag{2.9}$$

where f_i and g_i is shorthand for the two fractions. Notice that the constant N_1 dropped out, which hints that:

$$\frac{A(x_i \to x_i')}{A(x_i' \to x_i)} = \frac{p(x_i')}{p(x_i)} \frac{g(x_i|x_i')}{g(x_i'|x_i)} \tag{2.10}$$

with p being the target distribution from before.
Since the target distribution is only known up to the constant, the method works by exploiting the relative probabilities $f(x_i')/f(x_i)$.
The algorithm gives the acceptance probability for two cases:

If the proposed new configuration is more probable than the last, then it is accepted:

$$A(x_i \to x_i') = 1, \qquad \text{if } f_i g_i >= 1 \tag{2.11}$$

And if the new configuration is less probable, it is accepted with the probability:

$$A(x_i \to x_i') = f_i g_i, \quad \text{if } f_i g_i < 1 \tag{2.12}$$

Where the probability of accepting the inverse being:

$$A(x_i' \to x_i) = \begin{cases} \frac{1}{f_i g_i}, & \text{if } f_i g_i >= 1 \\ 1, & \text{if } f_i g_i < 1 \end{cases} \tag{2.13}$$

Which is consistent with Eq. 2.9.

For the Metropolis algorithm (without the Hastings), the less general case is considered. The distribution D has to be symmetric, which implies that the probability of proposing the new sample given the current and the other way around is the same:

$$\frac{g(x_i|x_i')}{g(x_i'|x_i)} = 1 \tag{2.14}$$

Then,

$$A(x_i \to x_i') = \begin{cases} 1, & \text{if } f_i >= 1 \\ f_i, & \text{if } f_i < 1 \end{cases} \tag{2.15}$$

In other words:

$$A(x_i \to x_i') = min(1, f_i) \tag{2.16}$$

To simulate a physics system, given an action S, the fraction

$$f_i = \frac{f(x_i')}{f(x_i)} = e^{-\delta S} \tag{2.17}$$

which is the exponent of the change of the action, that depends on the system simulated. It is important to note that a complex action makes no well defined probability measure. Therefore, the Metropolis algorithm are only able to simulate systems with real actions.

The accept/reject step will then be on the form:

```
rng = MersenneTwister(11111)
if rand(rng) < exp(-dS)
    x_i = x_new
else
    x_i = x_old
end
```

Where $rand(rng)$ gets a random number uniformly distributed on the interval $[0, 1]$, given by the psedo-random generator MersenneTwister with seed 11111. The seed of the generator can be set to a less deterministic value, like the computer time, temperatures, other values or combinations of these, to create a different set of random numbers. If no seed is given to the MersenneTwister from the Random package of Julia, a seed is generated "using entropy from the system".
If the action is increased with the proposal, it is only accepted with a probability depending on the magnitude of the increase. And if it is decreased (i.e. $e^{-\delta S} > 1$), the change is always accepted.

After each Metropolis sweep, each coordinate x_j, $j \in \{1, 2, ..., N_\tau\}$, has been proposed changed. For a visual representation, consider the imaginary time axis, with the equidistant points $j\delta\tau$:

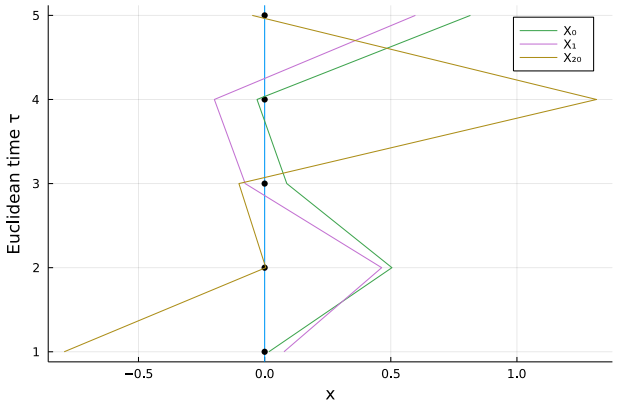


Figure 2.1: Shows a visual representation of configurations with $N_\tau = 5$ and $\delta\tau = 1$. X_0 is a configuration after the Markow chain has thermalized, X_1 is the next configuration in the Markow chain, and is very similar to X_0. X_{20} is the 20th configuration in the chain after X_0, and differs more from X_0. Inspired by Figure 1. [4, p. 8].

As seen in the plot, new configurations are generated by small random changes proposed to each element in the previous configuration.

2.3 Results from the metropolis algorithm

The samples generated in the burn-in time will skew the distribution towards samples close to the initial configuration. Any initial configuration skews the sample distribution, but configurations that have high probability in the target distribution skew the sample distribution less, since new samples are

drawn with relative probability close to equal to the target distribution. For initial configurations that have very small relative probability, the proposals are drawn from only the neighborhood, and the relative probability of the initial configuration is not taken into account, as they would have if the algorithm had drawn random proposals and with low probability ended up at the configuration. When the number of samples goes to infinity, this skewness goes to zero, as the proposals eventually will be drawn from the target distribution and the whole configuration space have been sampled. For a finite number of samples, one must take care to not choose initial configurations far away from the dense probability areas. Giving an example of the burn-in which should be discarded before saving configurations:

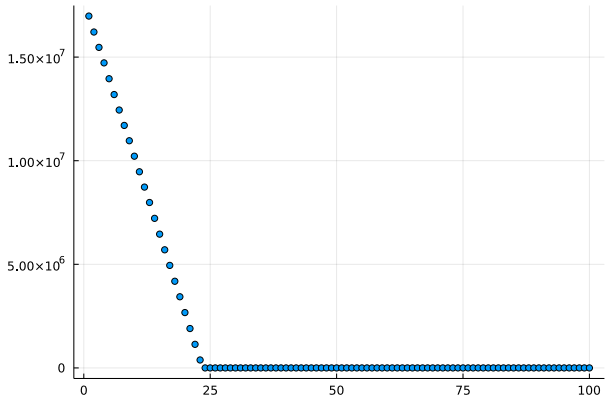


Figure 2.2: Action of configurations generated by the metropolis algorithm. For each sample saved, 20'000 were discarded. The initial condition was initialized to an array of 16 elements of value 500. The algorithm uses 460'000 Metropolis sweeps to sample a configuration with close to zero action.

This results in observables of the system being estimated to wrong values for finite sample-sizes, if not the burn-in is discarded:

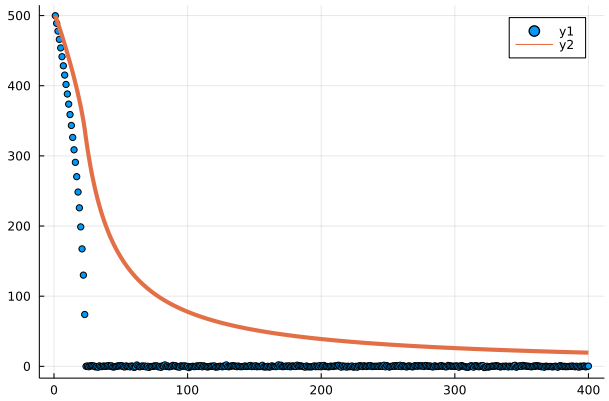


Figure 2.3: Running average of x_1 for a badly initialized simulation.

As is seen in Fig. 2.2, the initial condition is very unlikely, and the probability density is skewed toward the initial configuration. The relative probability of samples being drawn from 500 or -500 is very low, but not 0. After an infinite amount of time, samples can be proposed and accepted from $(-\infty, \infty)$ and the scewness imposed by the initial configuration will approach 0. By discarding the first samples, where the configurations are not thermalized, the error induced by the first configurations in the new data set can be ignored.

By saving configurations according to the distribution, expectation values like $\langle x \rangle$ and $\langle x^2 \rangle$ can be estimated. For calculating the errors, the naive error estimator

$$err_O = \sqrt{\frac{\frac{1}{N-1}\sum_{i=1}^{N}\left[O_i - E(O)\right]^2}{N}} = \sqrt{\frac{\sigma^2_{O,std}}{N}} \tag{2.18}$$

is used. This is the same as the standard deviation of the variables divided by the square root of the number of variables, N. This is just the naive estimator for the error, since it is assumed the samples are uncorrelated, which is not always the case. An estimator for the statistical error for autocorrelated data is derived in Sec. 2.4.5.

Running a simulation with $N_\tau = 16$ and $\beta = 8 \implies \delta\tau = 0.5$, gathering 1'000'000 samples of possible configurations, shown in the following plots:

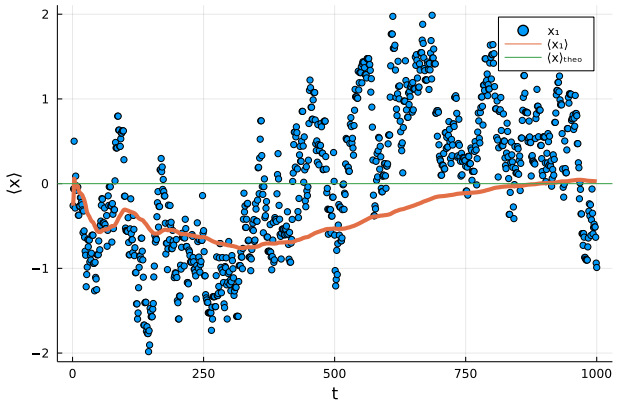


Figure 2.4: Average of x_1 for the first 1000 samples in Metropolis time, $\langle x_1 \rangle$, shown with the individual samples of x_1.

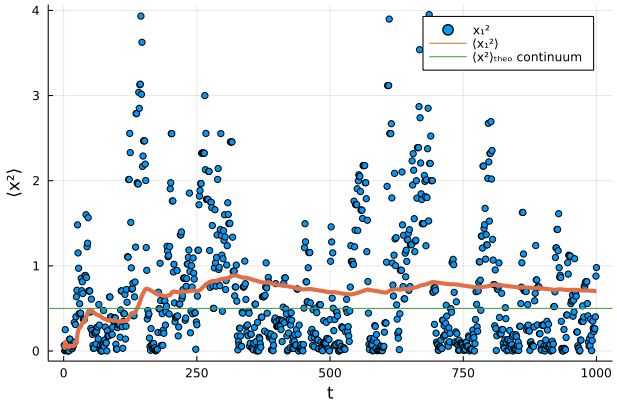


Figure 2.5: Average of x_1^2 for the first 1000 samples in Metropolis time, $\langle x_1^2 \rangle$, shown with the individual samples of x_1^2.

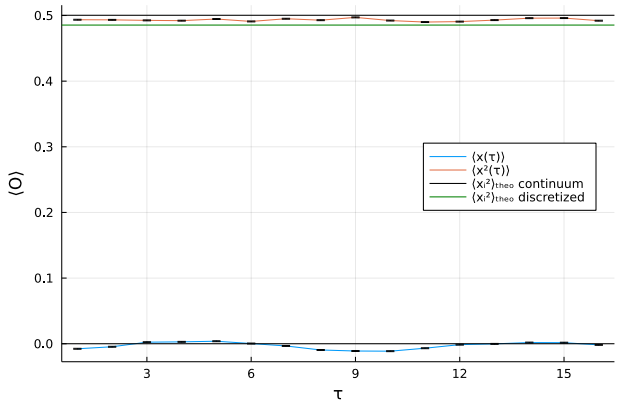


Figure 2.6: Final estimated mean of x_i and x_i^2 for each index $i = 1, ..., N_\tau$, $\langle x_i \rangle$ and $\langle x_i^2 \rangle$ for 1'000'000 samples. The naively estimated error bars are far away from the realistic error.

Fig.2.4 and Fig. 2.5 capture the evolution of the mean for the first element x_1 in simulation time after burn in. In Fig. 2.6 the final mean for each element in the configuration, x_i where $i = [1, ..., N_\tau]$ is shown. As expected, x is centered around 0, while x^2 is centered around some other value. The naively estimated error bars are far off from the analytical expectation values, since the samples are heavily correlated, and many more samples are needed to get reliable estimates of the mean. To estimate appropriate error bars Jackknife resampling is introduced in Sec. 2.4.5.
Autocorrelation of the samples can be removed or mediated by removing intermediate samples between each saved (See Sec. 2.4.1).

The one-point correlation function (See Sec. 3.1) for the samples generated is plotted:

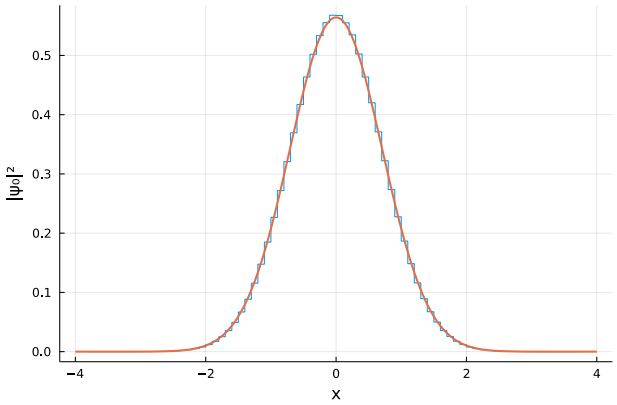


Figure 2.7: Probability density plot for x over all indexes in N_τ, in unfilled histogram form, with the smooth line being the analytical expectation for the ground state $\beta \to \infty$ in orange.

Fig. 2.7 shows the distribution of sampled realizations of x_i for all i in the configurations. The PDD is centered about 0, which is the analytical result for the expectation value of x.

2.4 Autocorrelation

This chapter contains an introduction to the correlations along simulation time in section 2.4.1. A derivation of the autocorrelation is provided in section 2.4.2. To compute the autocorrelation of a data set more efficiently a fast Fourier transformation of the original equation is derived in section 2.4.3. Results and plots from a Metropolis simulation without correlations is provided in 2.4.4.
Finally how to estimate error bars of correlated data using Jackknife analysis is shown in 2.4.5.

2.4.1 Correlations in simulation time

Every new configuration generated by the Metropolis Update only differ from the previous by a small amount, since they are the previous coordinates some of which may have added to it a random change. In other words the data is correlated. Since correlated data contains less information of the distribution of created configurations, with many successive configurations being similar to the previous one, to approach a reliable distribution will require many more samples than if the data was uncorrelated.
To remove or mediate the autocorrelation of the samples, one can remove

some number of intermediate samples between each saved. By removing correlated configurations in between two uncorrelated ones, the configurations saved approaches the correct distribution faster. Defining an integer value n_{skip} which removes every $n_{skip} - 1$ sample between each saved, the following expectation values can be estimated from a Metropolis algorithm:

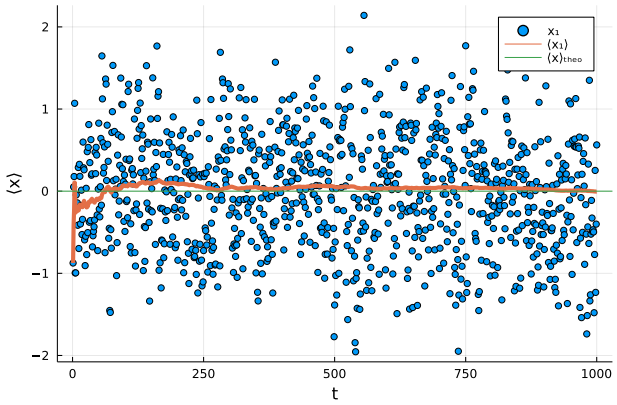


Figure 2.8: Running average of x at index 1, $\langle x_1 \rangle$, shown with the individual samples. For each sample saved, 19 are discarded to reduce autocorrelation

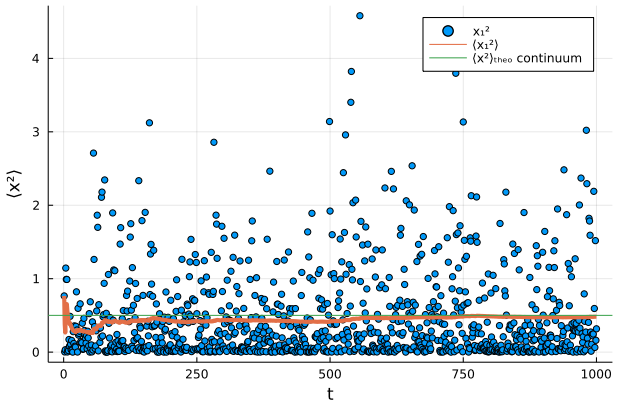


Figure 2.9: Running average of x^2 at index 1, $\langle x_1^2 \rangle$, shown with the individual samples. For each sample saved, 19 are discarded to reduce autocorrelation

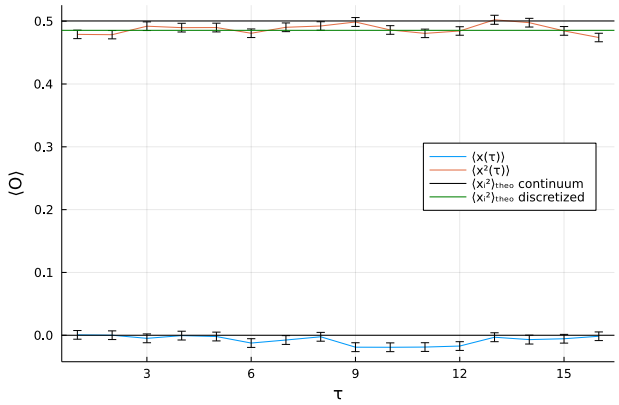


Figure 2.10: Final estimated mean of x_i and x_i^2 for each index $i = 1, ..., N_\tau$, $\langle x_i \rangle$ and $\langle x_i^2 \rangle$. For each sample saved, 39 are discarded to reduce autocorrelation. The jackknife estimated error bars are only marginally larger than the naively estimated error bars.

Compare to the figures of Sec. 2.3. Since the samples are now approximately uncorrelated, fewer samples are needed to get better estimates of the mean. The simulation needs to run for n_{skip} longer to generate the same amount of samples. If a system with a finer lattice is simulated, decreasing $\delta\tau$ keeping $N_\tau \delta\tau$ constant, the samples are more autocorrelated. To remove autocorrelation n_{skip} must increase with the inverse of the lattice spacing, which will cause simulation times to increase. To estimate expectation values in this limit, a fit is made through expectation values of decreasing lattice spacing's.

To calculate the amount of samples to discard between every two observations, the autocorrelation is computed. If the autocorrelation goes to zero very quickly (exponential decrease in the first 1-2 Δt, then noise) one can approximate the calculations on the data as if the samples are independent. If not, an estimate of n_{skip} can be made by calculating the integrated autocorrelation time (Sec. 2.4.2).

Comparing the naive and unbiased estimator of the mean (derived in section 2.4.5), the effective number of independent measurements is $N_{eff} = N/(2\tau_{O,int})$ [4, p. 15]

Over-Relaxation is another method used to reduce autocorrelation in Metropolis simulations of systems similar to the harmonic oscillator. The method works by introducing a trial change in the Metropolis sweep that changes x_i far from the old value while keeping the action constant. The method is only applicable for actions where the solution for x_i giving zero contribution to the action can be solved for analytically. See Ch. VI. [4,

p. 15–16].

2.4.2 The Autocorrelation derivation

Autocorrelation describes how samples in a signal or data set is correlated with themselves as a function of a time shift. If for example data of the outdoor temperature is considered, the temperature data for every minute of a day will typically display high positive autocorrelation for small time shifts. Since the temperature from hour to hour does not change much, temperatures measured depend on the temperature a short while ago. The autocorrelation gives a value for how much the next value is dependent on the previous. Temperature measurements will also display seasonality, since the temperature change follows the same patterns over intervals of 24 hours and intervals of 365 days. This causes the autocorrelation to go up from zero for every integer multiple of the time intervals of the seasonality.

In MCMC the Markov Chain makes small changes to existing configurations, therefore new configurations depend on the previous. No seasonality is expected to occur in the generation of data, since the proposals are generated by random changes to only the previous sample. Still, when calculated for finite samples, the autocorrelation can display small signs of seasonality because of the random samples coincidentally aligning. This is referred to as noise in the autocorrelation.

When extracting expectation values from the data, the assigned naive errors does not account for autocorrelation. In other words, the samples are not independent, and errors underestimates the true statistical errors of correlated samples. In section 2.4.5 an unbiased estimator of the variance of the mean is derived, to get better estimates of the statistical error in autocorrelated data.

Since the simulation and generation of data is in under full control, certain methods can decrease autocorrelation time in the generated samples, and naive error will approach the true statistical error. For the simulation to be resource efficient, less autocorrelation imply more of the samples collected are effectively independent, which imply better statistics with fewer samples. The MCMC algorithms samples more often in areas probable to be accepted, but induces autocorrelation in succeeding samples, by only saving samples at regular intervals, the previously saved configuration will go through multiple proposal and accept/reject steps before a new is saved. There will be a higher computational cost per sample, but each sample will be less correlated.

Autocorrelation for continuous real and bounded data (or signal), $x(t)$,

is defined as:

$$A_x(\Delta t) = \int_{-\infty}^{\infty} x(t)x(t+\Delta t)dt \tag{2.19}$$

where the integration limits imply integrating over all data existing on the interval.
If the data is periodic the cyclic autocorrelation can be computed from this equation, where $x(t_{max} + \delta t) = x(t_{min})$. If the data is not periodic, the existing data is padded to $\pm\infty$ by the empty signal. This gives the linear autocorrelation. In this thesis it is the latter that will be used for calculating correlations in simulation time.

In cases where the data is discrete, use the discrete autocorrelation:

$$A_x(\Delta t) = \int_{-\infty}^{\infty} x(t)x(t+\Delta t)\delta_t dt \tag{2.20}$$

$$= \frac{1}{N-\Delta t}\sum_{t=1}^{N-\Delta t} x(t)x(t+\Delta t) \tag{2.21}$$

The covariance (which also is referred to as "autocorrelation") of an observable $\langle O \rangle$ for discrete data is defined as:

$$A_O(t_{MC}) = \Big\langle \big(O_i - \langle O_i \rangle\big)\big(O_{i+t_{MC}} - \langle O_{i+t_{MC}} \rangle\big) \Big\rangle \tag{2.22}$$

where $t_{MC} = \Delta t$, is the difference in Markov chain (or Metropolis) time between each sample for which the correlation is computed. This is Eq. (61) [4, p. 13], and will be the definition of "Autocorrelation" that will be used in the following sections. Note that, since the harmonic oscillator discussed previously has the expectation value for x, $\langle x \rangle = 0$, the autocorrelation for this observable will be approximately equal to the covariance. To compare autocorrelations of different data sets it is easier to work with the normalized autocorrelation:

$$A_{O,norm}(t_{MC}) = \frac{A_O(t_{MC})}{A_O(0)} \tag{2.23}$$

where the autocorrelation is normalized by the sample variance $\sigma^2_{O,std} = A_O(0)$. The normalized autocorrelation will then always start at 1 for $t_{MC} = 0$.

To get an estimate for how correlated the samples are, the integrated autocorrelation time will be computed:

$$\tau_{O,int} = \frac{1}{2} + \sum_{t_{MC}=1}^{N_\tau - 1} A_{O,norm}(t_{MC}) \tag{2.24}$$

This gives a measure of how correlated an observable O is. To compute the integrated autocorrelation, the sum is cut of at an clearly defined time interval t_{MC} where the exponential relation for the autocorrelation breaks down. In [4] and in the Julia code of this thesis, this is taken to be when the autocorrelation first becomes negative.

2.4.3 Fast Fourier transform AC

For a real and bounded signal, the autocorrelation can be defined as:

$$A(\tau) = \int_{-\infty}^{\infty} x(t)x(t+\tau)dt \tag{2.25}$$

where $\tau = \delta t$ is the time shift between the evaluations of $x(t)$.
Since the signal is real, the complex conjugates:

$$x^*(t) = x(t) \tag{2.26}$$
$$x^*(t+\tau) = x(t+\tau) \tag{2.27}$$

The Fourier transforms of x:

$$x(t) \to X(\omega) \tag{2.28}$$
$$x(t+\tau) \to e^{i\omega\tau}X(\omega) \tag{2.29}$$
$$x^*(t+\tau) \to e^{-i\omega\tau}X^*(\omega) \tag{2.30}$$

Applying the Parseval's theorem for Fourier transformations:

$$\int_{-\infty}^{\infty} x(t)y^*(t)dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)Y^*(\omega)d\omega \tag{2.31}$$

to Eq. 2.25 using Eq. 2.27,2.28, and 2.30

$$\int_{-\infty}^{\infty} x(t)x^*(t+\tau)dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{-i\omega\tau}X^*(\omega)d\omega \tag{2.32}$$

Which is the same as:

$$\int_{-\infty}^{\infty} x(t)x(t+\tau)dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 e^{-i\omega\tau}d\omega \tag{2.33}$$

A new equation for the autocorrelation of a real and bounded signal is then:

$$\begin{aligned} A(\tau) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 e^{-i\omega\tau}d\omega \\ &= \mathcal{F}^{-1}\big(|\mathcal{F}(x)|^2\big)[\tau] \end{aligned} \tag{2.34}$$

where x is the continuous data stream or signal.

For data sets of discrete data with elements O_i, as those generated by MCMC methods, the discrete Fourier transformation can be used to calculate the discrete autocorrelation function:

$$A_O(t_{MC}) = F^{-1}(|F(\{O_i\})|^2)[t_{MC}] \tag{2.35}$$

with F being the discrete Fourier transform.

To get the covariance of the data the mean is subtracted from the data before doing the first Fourier transform:

$$A_O(t_{MC}) = F^{-1}(|F(\{O_i - \langle O \rangle\})|^2)[t_{MC}] \tag{2.36}$$

This is the discrete Fourier transform equation to calculate the symmetric autocorrelation of the data. To get the standard linear autocorrelation the input data must be padded at the end with 0's to double its length.

$$A_O(t_{MC}) = A_O(t_{MC})/A_O(0) \quad , A_O(0) = \sigma^2_{O,std} \tag{2.37}$$

The normalized autocorrelation, which we will use, is normalized by the sample variance.

Since the discretized Fourier Transform easily can be computed by use of vectors and matrices, it has also been given the name Fast Fourier Transform (FFT), and is frequently used for efficient computation in computer graphics and many other fields. This allows for vastly decreased computation time of autocorrelation.
To estimate the efficiency, for a single variable x_1 in a set of 10'000 samples, implementations of the computation of the autocorrelation by summation and Fourier transforms were benchmarked. The autocorrelation by summation implementation took 493 ± 4.8 ms, while the Fourier transform implementation took 2.56 ± 0.44 ms.

2.4.4 Autocorrelation for a Metropolis simulation

For a Metropolis simulation which has not run for very long, and where burn in and samples between each saved were removed looks like:

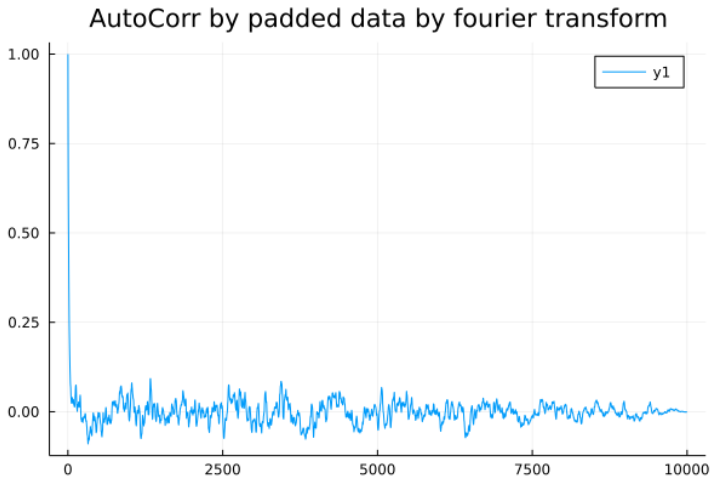


Figure 2.11: Plot of autocorrelation for Monte Carlo time $t_{MC} = [0 : 9999]$

As expected for uncorrelated samples, the Autocorrelation is 1 for $\tau_{MC} = 0$, then drops of into noise. The noise decreases with simulation lenght.

Julia has packages to calculate the autocorrelation of data in an array. To verify the results above, the autocorrelation derived here, being the normalized covariance, has been compared to the $autocor()$ function provided from the Julia package $StatsBase.jl$.
To visualize the autocorrelation of the Markov chain generated by the Metropolis algorithm, the autocorrelation for $\Delta t = [0, 300]$ of the samples generated in 2.3 is plotted:

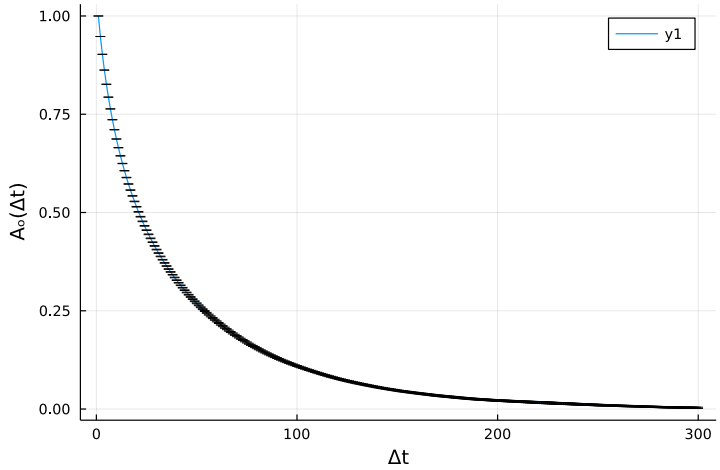


Figure 2.12: Autocorrelation of 1'000'000 samples generated by the Metropolis algorithm. After burn in, no samples are removed in between saved samples.

Excluding burn time, the autocorrelation has an exponential decrease. When not removing any intermediate samples the saved samples are heavily autocorrelated.
When removing intermediate samples, the autocorrelation of the samples is decreased, as the configurations go through multiple Metropolis sweeps between being saved. The samples generated in 2.4.1 have the following autocorrelation:

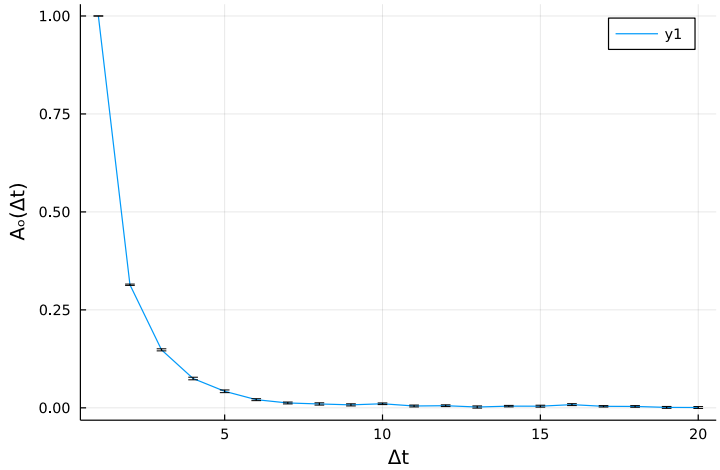


Figure 2.13: Autocorrelation of 10'000 samples generated by the Metropolis algorithm. After burn in, 39 samples are removed in between every saved sample.

The samples are much less autocorrelated. When estimating errors for nearly uncorrelated samples the naive error can be used without heavily underestimating the true statistical errors. The noise in the autocorrelation is reduced by increasing the number of samples.

2.4.5 Jackknife

As has been shown, the configurations in the Markov chain are correlated. To do analysis with reliable estimate of the error of autocorrelated data, certain statistical methods have been developed. Here the Jackknife procedure will be introduced.

Jackknife resampling is a statistical method for estimating mean and variance of the mean or variance of correlated data. It is fully deterministic contrary to newer resampling methods such as the Bootstrap, which picks samples randomly from the data set to estimate the mean and variance. The Jackknife will give the same results when run on the same data, while the Bootstrap, because of its random component, gives different results each time. The Jackknife is less computationally expensive, and because of giving the same results for the same data set, preferred when publishing reproducible results.

As a resampling method, smaller samples from the full data set is drawn, and estimators are computed for each.

Following the derivation of [4, p. 13]: Block estimators of bin-size B

samples from the data are given by $k = (1, ..., N_B)$ of

$$o_k = \frac{1}{B}\sum_{i=1}^{B} O_{(k-1)B+i} \tag{2.38}$$

where N_B is the total number of block estimators. The bin-size should be larger than the autocorrelation time to ensure that the N_B values of o_k can be treated as uncorrelated. o_1 contains the first B samples from the data set, o_2 the next B samples and so on.
The bin-based variance of the mean is given by

$$\sigma^2_{E(O),bins} = \frac{1}{N_B(N_B-1)}\sum_{k=1}^{N_B}\big[o_k - E(O)\big]^2 \tag{2.39}$$

As the estimators o_k are average over only the N_Bth fraction of all measurements, it may prevent the determination of o_k for some k (A fit on too few configurations may occasionally fail to converge). By using complimentary bins to the block estimators of length $N - B$, this problem is overcome:

$$\tilde{o}_k = \frac{1}{N-B}\Big(\sum_{i=1}^{N} O_i - Bo_k\Big) = \frac{1}{N-B}\Big(\sum_{i=1}^{N}\big[O_i\big] - \sum_{i=1}^{B}\big[O_{(k-1)B+i}\big]\Big) \tag{2.40}$$

where N is the total number of samples in the data set. $\tilde{o}_1$ is an estimate of all but the B first samples of the data set, $\tilde{o}_2$ an estimate of all but the next B samples and so on. Therefore, the number of samples averaged over in the N_B new Jackknife estimators, $N - B$, is much larger than the original B for the block estimators, and the estimators converges as long as there is enough samples in the data set.
The resulting jackknife variance of the mean is

$$\sigma^2_{E(O),jack} = \frac{N_B - 1}{N_B}\sum_{k=1}^{N_B}\big[\tilde{o}_k - E(O)\big]^2 \tag{2.41}$$

An implementation of the Jackknife resampling method in Julia is given here:

```
function Jackknife1(array1::AbstractVector,binsize::Integer)
    N_B = floor(Int64,length(array1)/binsize) # ommiting last not-full bin
    leng2 = N_B*binsize
    array2 = array1[1:leng2]
    sum1 = sum(array2)
```

```
    jf = Vector{Float64}(undef,N_B)
    fill!(jf,sum1)
    for k = 1:N_B
        jf[k] -= sum(array2[(k-1)*binsize+1:k*binsize])
    end
    jf = ((jf ./ (leng2 - binsize)) .- (sum1/leng2)).^2
    jfvm = mean(jf)
    jfvm *= (N_B - 1)
    return [mean(array1),sqrt(jfvm)]
end
```

Instead of summing over all but some samples for each Jackknife estimator, the sum of all samples is computed in the beginning, and only the sum of elements to remove is computed for each estimator. This is much more efficient. Benchmarking 10'000 samples, choosing the best of three runs, of an implementation of the prior for an array of 15'001 elements, choosing Jackknife bin size $N - B = N - 1$. This gave a mean evaluation time of $455ms \pm 7.3ms$ (mean $\pm\sigma$), while an implementation for the latter gives a mean evaluation time of $96.7\mu s \pm 116.78\mu s$ for the same array.

If the Jackknife bin size decreases, $N-B$ where B is larger, the efficiency of the first implementation should increase. Usually if the autocorrelation time is larger than half the size of the data set, Jackknife bin size chosen to $N - B < N/2$, more samples are needed to get good estimates.

Taking the data set from the Metropolis simulation shown in Sec. 2.3 under the Jack-knife gives an unbiased estimator of the variance of the mean, from which the error can be calculated by swapping the variance in Eq. 2.18 by the jackknife estimated variance.

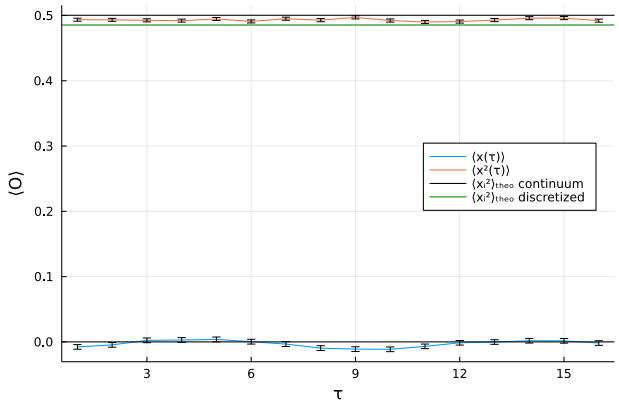


Figure 2.14: Final estimated mean of x_i and x_i^2 for each index $i = 1, ..., N_\tau$, $\langle x_i \rangle$ and $\langle x_i^2 \rangle$. The jackknife estimator of the variance of the mean gives more reliable error bars.

Comparing to Fig. 2.6, the error bars are more reliably estimated.
As is seen in the figure, the discretization of the harmonic oscillator causes a different system to be simulated. It is only in the continuum limit $\delta\tau \to 0$ that the true system is recovered. In the same limit autocorrelation increases inversely proportional to $\delta\tau$. To collect effectively independent samples on smoother lattices comes at increased computational cost. To extract expectation values from the intended system, a fit must be made against data of different lattice spacing's.

Chapter 3

n-Point Correlation

For the quantum system simulated with the Metropolis algorithm in Sec. 2.3, the ratio $\delta\tau = \frac{\beta}{N_\tau}$ corresponds to a coupling to a heat bath by the inverse temperature $\beta = \frac{1}{T}$. Keeping $\beta = \delta\tau N_\tau$ constant while changing N_τ and $\delta\tau$ the lattice can be made coarser (N_τ smaller) and finer (N_τ larger). A choice of β changes how samples are distributed around $\langle x_i \rangle$, specifically the value of the second moment, $\langle x^2 \rangle$. This will be shown by plotting the probability distribution diagram (PDD) for samples generated by two different Metropolis simulations in Sec. 3.1.

To simulate how the system changes as a function of the inverse temperature, the number of lattice points, N_τ, has been kept constant and the lattice spacing a has been changed as a function of β. For larger β (lower temperature) the lattice spacing increases, and the system approaches the ground state.
To see the correlations within the lattice the two-point correlation function (TPCF) will be computed in Sec 3.2.

From the TPCF, the effective mass and energy differences in the system can be estimated. This will be the topic of Sec. 3.3.

3.1 One-Point Function

When sampling is done, configurations with values x_i for $i = [1, 2, ..., N_\tau]$ are saved. To see how the samples are distributed, the One-Point function is computed. This corresponds to binning the number of values that fall between values $[a_j, b_j]$ then taking the norm, such that a probability density distribution (PDD) is obtained.
As was already shown in Sec. 2.3, the probability distribution of a system with inverse temperature $\beta = 8$ almost matches the thermalized limit.

Running a simulation with $\beta = 2$, $N_\tau = 16$ ($\delta\tau = 0.125$ and one with $\beta = 32$, $N_\tau = 64$ (to make $\delta\tau = 0.5$ smaller than 1) gives the following PDD's for x:

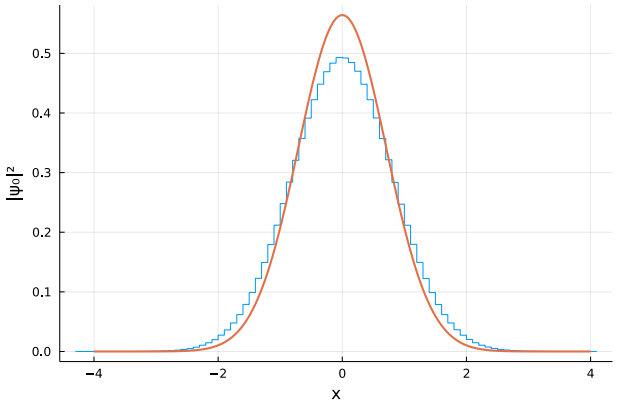


Figure 3.1: Probability density plot for x over all indexes in N_τ, in unfilled histogram form. The inverse temperature is here $\beta = 2$. The smooth line is the analytical expectation for the ground state $\beta \to \infty$ in orange.

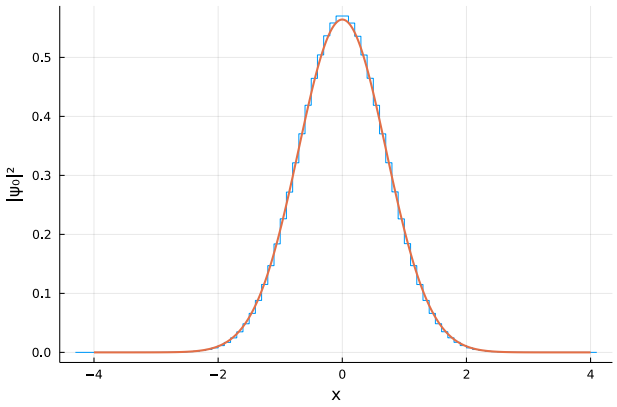


Figure 3.2: Probability density plot for x over all indexes in N_τ, in unfilled histogram form. The inverse temperature is here $\beta = 32$ ($N_\tau = 64$ to set $\delta\tau = 0.5$. The smooth line is the analytical expectation for the ground state $\beta \to \infty$ in orange.

As can be seen from Fig. 3.1, the system is not in the ground state, and therefore will the probability distribution of x be much wider. In Fig. 3.2 the system is much cooler, and the probability distribution of x approaches the ground state.

3.2 Two-Point Correlation

The two-point correlation function is used to measure correlations within the configurations. Since there is a momentum coupling in the action (See Eq. 1.44), there is information stored in this correlation. In the configurations, consisting of components $x_j = x(\tau)$ for different τ, the Two-Point correlation is found by taking the $x(\tau)$ minus their average $\langle x(\tau)\rangle$ and multiplying them with themselves shifted by an interval $\Delta\tau$ and taking the average:

$$G(\Delta\tau) = \big\langle (x(\tau) - \langle x(\tau)\rangle)(x(\tau+\Delta\tau) - \langle x(\tau+\Delta\tau)\rangle)\big\rangle \tag{3.1}$$

As one can see from this expression, for $\Delta\tau = 0$ the TPCF is the same as the standard deviation of the configuration.
To get an expression that is easier to compute, rewrite the equation:

$$\begin{aligned}
G(\Delta\tau) &= \big\langle (x(\tau) - \langle x(\tau)\rangle)(x(\tau+\Delta\tau) - \langle x(\tau+\Delta\tau)\rangle)\big\rangle \\
&= \frac{1}{N}\sum_{\tau=1}^{N} x(\tau)x(\tau+\Delta\tau) + \langle x(\tau)\rangle\langle x(\tau+\Delta\tau)\rangle - x(\tau)\langle x(\tau+\Delta\tau)\rangle - x(\tau+\Delta\tau)\langle x(\tau)\rangle \\
&= \langle x(\tau)x(\tau+\Delta\tau)\rangle + \langle x(\tau)\rangle\langle x(\tau+\Delta\tau)\rangle \\
&\quad - \langle x(\tau+\Delta\tau)\rangle\frac{1}{N_\tau}\sum_{\tau=1}^{N}\big[x(\tau)\big] - \langle x(\tau)\rangle\frac{1}{N_\tau}\sum_{\tau=1}^{N}\big[x(\tau+\Delta\tau)\big] \\
&= \langle x(\tau)x(\tau+\Delta\tau)\rangle + \langle x(\tau)\rangle\langle x(\tau+\Delta\tau)\rangle - \langle x(\tau+\Delta\tau)\rangle\langle x(\tau)\rangle - \langle x(\tau)\rangle\langle x(\tau+\Delta\tau)\rangle \\
&= \langle x(\tau)x(\tau+\Delta\tau)\rangle - \langle x(\tau)\rangle\langle x(\tau+\Delta\tau)\rangle
\end{aligned} \tag{3.2}$$

The last expression for $G(\Delta\tau)$ is the one given in Eq. (46) [4, p. 9]:

$$G(\Delta\tau) = \big\langle (x(\tau)x(\tau+\Delta\tau)\big\rangle - \big\langle x(\tau)\big\rangle\big\langle x(\tau+\Delta\tau)\big\rangle$$

For the harmonic oscillator, the $\langle x(\tau)\rangle = 0, \quad \forall\tau$, so only the first term on the right hand side contribute. The equation for the connected TPCF then becomes:

$$G(\Delta\tau) = \big\langle (x(\tau)x(\tau+\Delta\tau))\big\rangle = \frac{1}{N_\tau}\sum_{i=1}^{N_\tau}\sum_{j,(j-i)modN_\tau=\Delta\tau} x(i)x(j) \tag{3.3}$$

where N_τ is the total number of elements x_i in the configuration.
As an example $G(0) = \frac{1}{N_\tau}\sum_{i=1}^{N_\tau}[x(i)]^2$, which is the mean of the squares of the elements $x(i)$ in the configuration. $G(\delta\tau)$ is largest for $\tau = 0$ (that is where the largest numbers are multiplied by the largest numbers, ie. itself,

and we expect it to decrease as $|\tau|$ goes to $N_\tau/2$ since the action only contains a next neighbor coupling term. The two-point correlation is symmetric for $\Delta\tau \to -\Delta\tau$, which can be seen from the above equation.

The two-point correlation of a harmonic oscillator with $\beta = 8$ and $N_\tau = 16$, simulated with the Metropolis algorithm is shown below.

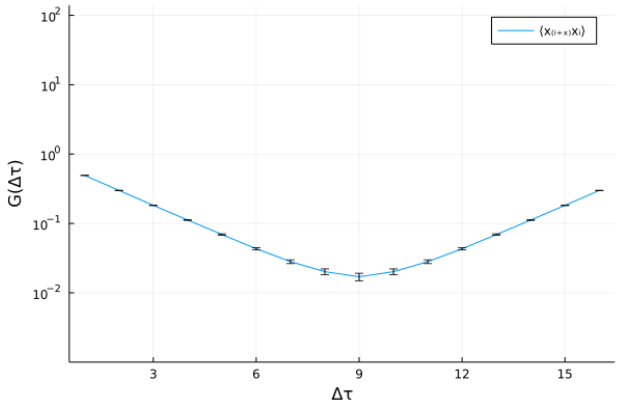


Figure 3.3: Two-Point Correlation of a Monte Carlo simulated Harmonic Oscillator action in a $N_\tau = 16$ lattice with inverse temperature $\beta = 8$.

From the TPCF it is possible to fit two exponentials, one for each slope, or a cosine hyperbolic.

$$G(\Delta\tau) = Ae^{-\Delta\tau/\xi} + Ae^{-(T-\Delta\tau)/\xi} \tag{3.4}$$

where the second term comes from the periodic boundary conditions. Eq. (49) [4, p. 10]. ξ is the correlation length of the lattice. From this it is possible to estimate energy differences of the system of the form $e^{-(E_1-E_0)\Delta\tau/\hbar}$, which will be shown in Sec. 3.3.

The true solution for the TPCF for the thermal harmonic oscillator is:

$$\langle x_i x_{i+j}\rangle = \frac{1}{2m\omega}\left(\frac{R^j + R^{N_\tau - j}}{1 - R^{N_\tau}}\right)^2 \tag{3.5}$$

with R defined in Eq. 1.53. This is Eq. (H8b) from [4, p. 37].
Below is a plot of the estimated Two-Point Correlation the previous simulation with the analytical solution:

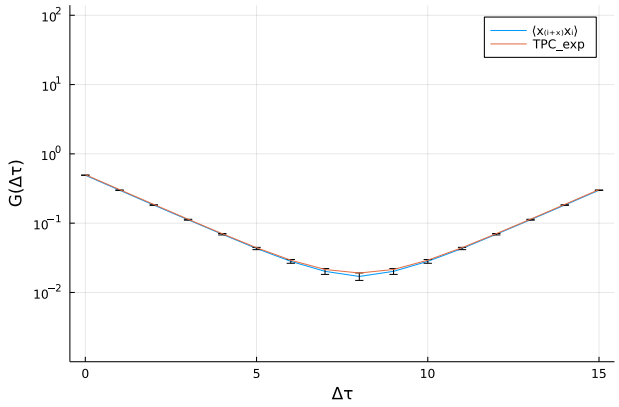


Figure 3.4: Two-Point Correlation Function of a simulated harmonic oscillator, with $N_\tau = 16$ timesteps and inverse temperature $\beta = 8$, with the analytical expected value TPC_exp.

Using a cosine hyperbolic function $f(x) = a * cosh(b * (x - 4))$, the TPCF displayed in Fig. 3.3 has been fitted using gnuplot. For the fit these were the results:

Variable	Value	Asymptotic Standard Error
a	0.0185058	± 8.909e-05
b	0.993218	± 0.001366

Table 3.1: cosh fit for the TPCF. From the cosh fit the effective mass can be estimated.

3.3 Effective Mass and Energy differences

The effective mass of the system can be calculated from the two-point correlation function. The effective mass is estimated by the local logarithmic slope of the TPCF:

$$m_{eff} = \frac{1}{\xi} = \frac{1}{2} log\left[\frac{G(\Delta\tau - 1)}{G(\Delta\tau + 1)}\right] \tag{3.6}$$

For the previous simulation the following estimation of the effective mass can be made:

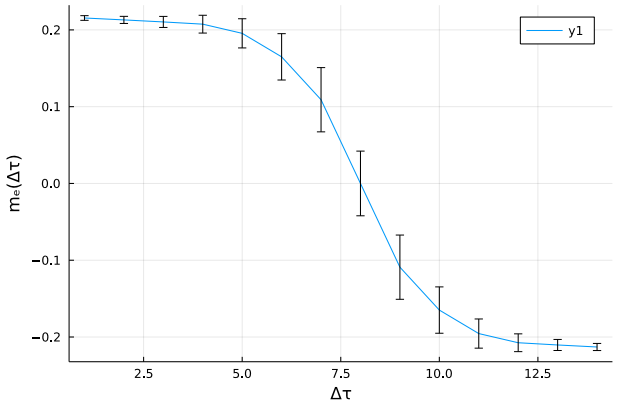


Figure 3.5: The effective mass estimated from the Two-Point Correlation Function of a simulated harmonic oscillator, with $N_\tau = 16$ time steps and inverse temperature $\beta = 8$.

The quantum harmonic oscillator (with $\hbar = 1$) has the energy spectrum $E_n = (n+\frac{1}{2})\omega$, $n = (0, 1, 2, ...)$. This can be verified by the lattice simulations described in Sec. 2.2 and 4.3. The energies themselves cannot be estimated, but the energy differences can by the two-point correlation function. Following derivations of Ch. VII B [4]. Rewriting the two-point correlation function, $G(\Delta\tau)$, in terms of the lowering and raising operators $\hat{a}$ and $\hat{a}^\dagger$ respectively, by using the following substitution for $\hat{x}$:

$$\hat{x} = i\frac{1}{\sqrt{2m\omega\delta\tau^2}}\frac{1}{\left(1+\frac{1}{4}\omega^2\delta\tau^2\right)^{1/4}}\left(\hat{a}-\hat{a}^\dagger\right) \tag{3.7}$$

the convention for the ladder operators being

$$\hat{a} = \frac{1}{\sqrt{2m\omega}}\frac{1}{(1+\frac{1}{4}\omega^2\delta\tau^2)^{1/4}}\left(\hat{p} - i\hat{x}m\omega\sqrt{1+\frac{1}{4}\omega^2\delta\tau^2}\right) \tag{3.8}$$

$$\hat{a}^\dagger = \frac{1}{\sqrt{2m\omega}}\frac{1}{(1+\frac{1}{4}\omega^2\delta\tau^2)^{1/4}}\left(\hat{p} + i\hat{x}m\omega\sqrt{1+\frac{1}{4}\omega^2\delta\tau^2}\right) \tag{3.9}$$

Eq. (G1) in [4, p. 34]. Since the ladder operators act on normalized eigenstates $|n\rangle$ of the number operator $\hat{a}^\dagger\hat{a}$, the lowering operator acting on the ground state $\hat{a}|0\rangle = 0$ and n number of raising operators acting on the ground state $(\hat{a}^\dagger|0\rangle = \sqrt{n!}|n\rangle$.

Then the two-point correlator takes the form

$$G(\Delta\tau) = \frac{1}{2m\omega\delta\tau^2\sqrt{1+\frac{1}{4}\omega^2\delta\tau^2}}\langle 0|\hat{a}(\tau)\hat{a}^\dagger(\tau+\Delta\tau)|0\rangle \tag{3.10}$$

after collecting the remaining terms in the substitution. After inserting a complete set of states, $I = \sum |n\rangle\langle n|$, between the two operators in 3.10, only the state $|1\rangle\langle 1|$ contributes (since $\hat{a}|1\rangle = |0\rangle$, $\hat{a}^\dagger|0\rangle = |1\rangle$ and $\langle n|m\rangle = \delta_{n,m}$).

Finally, writing $\hat{A}(t) = e^{\hat{H}t}\hat{A}e^{-\hat{H}t}$ the two-point correlator $G(\Delta\tau)$ can be expressed as

$$\begin{aligned} m\delta\tau G(\Delta\tau) &= \frac{1}{2m\omega\delta\tau^2\sqrt{1+\frac{1}{4}\omega^2\delta\tau^2}}\langle 0|\hat{a}(\tau)|1\rangle\langle 1|e^{-E_1\Delta\tau}\hat{a}^\dagger(\tau)e^{E_0\Delta\tau}|0\rangle \\ &= \frac{1}{2m\omega\delta\tau^2\sqrt{1+\frac{1}{4}\omega^2\delta\tau^2}}e^{-(E_1-E_0)\Delta\tau}\langle 0|\hat{a}(\tau)|1\rangle\langle 1|\hat{a}^\dagger(\tau)|0\rangle \\ &= e^{-(E_1-E_0)\Delta\tau}G(0) \end{aligned} \tag{3.11}$$

From this equation, one can see that the logarithmic slope of $G(\Delta\tau)$ as a function of $\Delta\tau$ measures the energy difference $E_1 - E_0$. See Fig. 3.4.

Higher order correlation functions (like the 4-point function) can be used to determine the energy differences $E_n - E_0$ for higher n.

Returning to the Effective mass, by using the new expression for the two-point correlator, Eq. 3.6 can be expressed as:

$$\begin{aligned} m_{eff} &= \frac{1}{2}log\Big(\frac{G(\Delta\tau-1)}{G(\Delta\tau+1)}\Big) = \frac{1}{2}log(e^{-(E_1-E_0)(\Delta\tau-1)}e^{(E_1-E_0)(\Delta\tau+1)}) \\ &= \frac{1}{2}log(e^{2(E_1-E_0)}) = E_1 - E_2 \end{aligned} \tag{3.12}$$

which holds when the slope of $log(G(\Delta\tau))$ is linearly decreasing.

Chapter 4

Langevin

The Langevin method can be used to sample from the probability distribution governing statistical properties of a physical system. As a classical example, consider Brownian motion of a heavy particle submerged in an ideal fluid. The particle experiences a drag of the fluid, and at the same time random fluctuations due to collisions with the particles of the fluid. If the fluid is still, i.e. the gradient of finite pieces of the fluid is zero, the particle has an expectation position of where it was initially set with momentum 0. As time progress, the particle will move about the original position. To calculate expectation values of the position and its variance, a Langevin equation for the system can be used.

An introduction to the Langevin equation is given in Sec. 4.1.

To introduce the concept the Langevin equation for a simple system, and the corresponding Fokker-Planck equation, is derived to find the distribution generated by the Langevin equation in Sec. 4.2.

To compute expectation values using the Langevin method in a computer program, the problem will first be discretized as previously. The Langevin method for the Euclidean time harmonic oscillator is demonstrated in Sec. 4.3.

4.1 The Langevin Equation

Following chapter 34 of [6], the Langevin equation can be defined as a first order stochastic differential equation of the form:

$$\dot{\boldsymbol{q}} = -\frac{1}{2}\boldsymbol{f}(\boldsymbol{q}(t), t) + \boldsymbol{\nu}(t) \tag{4.1}$$

where Newton's notation has been used, $\dot{\boldsymbol{q}} = d\boldsymbol{q}(t)/dt$, where t is time and $\boldsymbol{q}(t)$ is a trajectory in $\mathbb{R}^d$. $\boldsymbol{f}(\boldsymbol{q}(t), t)$ is the drift force, assumed to be a smooth

function of $\boldsymbol{q} = (q_1, q_2, ..., q_d)$, and $\boldsymbol{\nu}(t)$ a stochastic vector function called the noise.

The noise is characterized by a functional probability measure $[d\rho(\nu)]$. The following derivations will be specialized to Gaussian white noise given by:

$$[d\rho(\nu)] = [d\nu]e^{-\frac{1}{2\Omega}\int dt\nu^2(t)}, \qquad \int [d\rho(\nu)] = 1 \tag{4.2}$$

where the positive constant Ω characterizes the width of the noise distribution. The measure is normalized to 1 for any finite time interval.

Gaussian noise can be equivalently described by its expectation and two-point function:

$$\langle \nu_i(t) \rangle = 0, \quad \langle \nu_i(t)\nu_j(t') \rangle = \Omega\delta_{i,j}\delta(t-t'), \quad 1 <= i, j <= d, \tag{4.3}$$

where again $\delta(t)$ is the Dirac delta function. Since the noise is uncorrelated in time, the Langevin process generates a Markov chain, since every value only depend on the first preceding value.

To determine whether the distributions generated by the Langevin equation approaches a stationary distribution, the corresponding Fokker-Planck equation is computed and used. The Fokker-Planck equation is introduced in the following section.

4.2 The Langevin and Fokker-Planck equations for a simple system

To demonstrate the Langevin method, consider the 1d harmonic potential

$$S = \frac{1}{2}\mu\phi^2 \tag{4.4}$$

where $\mu = m\omega^2$.

Following the derivations of [7]. The expectation values for this system will be:

$$Z = \int_{-\infty}^{\infty} e^{-\frac{1}{2}\mu\phi^2} \tag{4.5}$$

$$\langle \phi \rangle = \frac{1}{Z}\int_{-\infty}^{\infty} \phi e^{-\frac{1}{2}\mu\phi^2} = 0 \tag{4.6}$$

$$\begin{aligned} \langle \phi^2 \rangle &= \frac{1}{Z}\int_{-\infty}^{\infty} \phi^2 e^{-\frac{1}{2}\mu\phi^2} \\ &= \frac{1}{Z}\int_{-\infty}^{\infty} \phi e^{-\frac{1}{2}\mu\phi^2} + \frac{1}{\mu}\frac{1}{Z}\int_{-\infty}^{\infty} e^{-\frac{1}{2}\mu\phi^2} = \frac{1}{\mu} \end{aligned} \tag{4.7}$$

To estimate the expectation values, first find the derivative of S

$$\frac{dS}{d\phi} = \mu\phi \tag{4.8}$$

and insert it into the Langevin equation

$$\begin{aligned}\frac{d\phi}{dt} &= -\frac{dS}{d\phi} + \eta \\ &= -\mu\phi + \eta\end{aligned} \tag{4.9}$$

where the second term on the right hand side is the drift term. The noise term must be normalized:

$$\langle \eta(t)\eta(t')\rangle = 2\delta_{t,t'} \tag{4.10}$$

The noise term will have the normalization constant:

$$A = \sqrt{2} \tag{4.11}$$

and the Langevin equation becomes:

$$\frac{d\phi}{dt} = -\mu\phi + \sqrt{2}\eta \tag{4.12}$$

To check whether or not the sampled distribution approaches a stationary distribution, the corresponding Fokker-Planck equation is derived.

For this Langevin Equation the corresponding Fokker-Planck equation is:

$$\frac{\partial}{\partial t}P = \mu\frac{\partial}{\partial\phi}\phi P + \frac{\partial^2}{\partial\phi^2}P \tag{4.13}$$

To get an equation for $\langle\phi\rangle$, multiply both sides by ϕ, and integrate over all possible values of ϕ:

$$\frac{\partial}{\partial t}\int_{-\infty}^{\infty} d\phi\phi P(\phi,t) = \mu\int_{-\infty}^{\infty} d\phi\phi\frac{\partial}{\partial\phi}\phi P(\phi,t) + \sqrt{2}\int_{-\infty}^{\infty} d\phi\phi\frac{\partial^2}{\partial\phi^2}P(\phi,t) \tag{4.14}$$

Now, integrating by parts the therms on the right hand side, noting that $P(\phi,t)$ is normalizable and its moments are well defined (at least up to second), which means P decays "fast enough" at the boundaries, the boundary terms in the integration by parts vanish. The first moment is then given by:

$$\frac{d\langle\phi(t)\rangle}{dt} = -\mu\langle\phi(t)\rangle \tag{4.15}$$

Which can be solved for $\langle\phi(t)\rangle$:

$$\langle\phi(t)\rangle = \phi_0 e^{-\mu t} \tag{4.16}$$

where ϕ_0 is the initial condition of ϕ. This implies that $\langle\phi(t)\rangle = 0$ as $t \to 0$.

The same can be done for the second moment. By multiplying on both sides of the Fokker-Planck equation by x^2 and integrating by parts. The diffusive term will also contribute in this case, since the second derivative $\frac{d^2}{d\phi^2}\phi^2 = 2$ and not 0. If the initial condition, ϕ_0, is set to 0, the equation evaluates to:

$$\frac{d\langle\phi^2(t)\rangle}{dt} = -2\mu\langle\phi^2(t)\rangle + 2 \tag{4.17}$$

Which gives the following solution for $\langle\phi^2(t)\rangle$ ($\langle\phi^2(0)\rangle = 0$ since $P(x, t = 0) = \delta(x - x_0)$):

$$\langle\phi^2(t)\rangle = \frac{1}{\mu}\Big[1 - e^{-2\mu t}\Big] \tag{4.18}$$

As $t \to \infty$, $\langle\phi^2(t)\rangle$ approaches $\frac{1}{\mu}$. The expectation values obtained from the Fokker-Planck equation corresponds to those derived in Eq. 4.6 and 4.7.

4.3 The discrete Langevin equation for Euclidean time Path integrals

To use the Langevin method to simulate a system with discrete action, the discrete Langevin equation must be derived. From the discretized action derived in 1.5:

$$S = \frac{1}{2}m\delta\tau \sum_{i=1}^{N_\tau} \Big[\Big(\frac{\phi_{i+1} - \phi_i}{\delta\tau}\Big)^2 + \omega^2\phi_i^2\Big] \tag{4.19}$$

Take the functional derivative:

$$\frac{\partial S}{\partial \phi_j} = \frac{1}{2} m\delta\tau \sum_{i=1}^{N_\tau} \Big[\frac{1}{\delta\tau^2} \frac{\partial}{\partial \phi_j} (\phi_{i+1} - \phi_i)^2 + \omega^2 \frac{\partial}{\partial \phi_j} \phi_i^2 \Big] \tag{4.20}$$

$$= \frac{1}{2} m\delta\tau \Big(\frac{1}{\delta\tau^2} \sum_{i=1}^{N_\tau} 2(\phi_{i+1} - \phi_i) \big(\frac{\partial}{\partial \phi_j} \phi_{i+1} - \frac{\partial}{\partial \phi_j} \phi_i \big) + \sum_{i=1}^{N_\tau} 2\omega^2 \phi_i \frac{\partial}{\partial \phi_j} \phi_i \Big)$$

$$= m\delta\tau \Big(\frac{1}{\delta\tau^2} \sum_{i=1}^{N_\tau} (\phi_{i+1} - \phi_i) \big(\frac{\partial}{\partial \phi_j} \phi_{i+1} - \frac{\partial}{\partial \phi_j} \phi_i \big) + \sum_{i=1}^{N_\tau} \omega^2 \phi_i \frac{\partial}{\partial \phi_j} \phi_i \Big)$$

$$= m\delta\tau \Big(\frac{1}{\delta\tau^2} \sum_{i=1}^{N_\tau} (\phi_{i+1} - \phi_i) \big(\delta(\delta\tau(j - (i+1))) - \delta(\delta\tau(j - i)) \big) + \sum_{i=1}^{N_\tau} \omega^2 \phi_i \delta(\delta\tau(j - i)) \Big)$$

$$= m\delta\tau \Big(\frac{1}{\delta\tau^2} \sum_{i=1}^{N_\tau} (\phi_{i+1} - \phi_i) \big(\frac{1}{|\delta\tau|} \delta(i - (j-1)) - \frac{1}{|\delta\tau|} \delta(i - j) \big) + \sum_{i=1}^{N_\tau} \omega^2 \phi_i \frac{1}{|\delta\tau|} \delta(i - j) \Big)$$

$$= m \Big(\frac{1}{\delta\tau^2} \sum_{i=1}^{N_\tau} (\phi_{i+1} - \phi_i) \big(\delta_{i,j-1} - \delta_{i,j} \big) + \sum_{i=1}^{N_\tau} \omega^2 \phi_i \delta_{i,j} \Big)$$

$$= m \frac{1}{\delta\tau^2} (2\phi_j - \phi_{j+1} - \phi_{j-1}) + \mu\phi_j. \tag{4.21}$$

with $\mu = m\omega^2$.
Then, to arrive at the discrete Langevin equation, write on the discretized form for each j:

$$\phi_j^{n+1} = \phi_j^n - \frac{\partial S}{\partial \phi_j} \delta t + \eta_j^n \delta t$$

$$= \phi_j^n - \Big(\frac{m}{\delta\tau^2} \big(2\phi_j^n - \phi_{j+1}^n - \phi_{j-1}^n \big) + \mu\phi_j^n \Big) \delta t + \eta_j^n \delta t \tag{4.22}$$

Where $\langle \eta \rangle = 0$, and the random Gaussian distributed numbers, η, are fully uncorrelated

$$\langle \eta_j^n \eta_k^m \rangle = \Omega \delta(j-k) \delta(n-m) = \frac{2}{|\delta\tau|} \delta_{j,k} \frac{1}{|\delta t|} \delta_{n,m} \tag{4.23}$$

where $\Omega = 2$.

Using Gaussian distributed random numbers with $\langle \eta \rangle = 0$ and $\langle \eta^2 \rangle = 1$ for η gives the normalization:

$$A = \sqrt{\frac{2}{\delta\tau \delta t}}$$

Which makes Eq. 4.22:

$$\phi_j^{n+1} = \phi_j^n - \Big(\frac{m}{\delta\tau^2}\big(2\phi_j^n - \phi_{j+1}^n - \phi_{j-1}^n\big) + \mu\phi_j^n\Big)\delta t + \sqrt{\frac{2\delta t}{\delta\tau}}\eta_j^n \tag{4.24}$$

Which is the discrete Langevin equation for the harmonic oscillator system from before. It is important to note that $\delta\tau$ is still the lattice spacing, and not the same as the Langevin-time δt.

By increasing the Langevin time step δt, the samples will be less correlated, but at the same time, the numerical approximation to the solution of the differential equation will be affected by larger errors. To remove these errors on the other hand, $\delta t \to 0$ and samples will be heavily autocorrelated, many samples will need to be collected to approach the target distribution.

To simulate the system, new paths ϕ^{n+1} are generated from the previous ϕ^n, by looping through the coordinates in Euclidean time as before. The difference from the Metropolis algorithm is that now a drift term pulls the path towards a smaller action, and it is the complex numbers η that makes the configurations "jump about" the equilibrium. Whereas in the Metropolis update, random changes were proposed, then accepted or rejected with a probability based on the relative change in action.

As with other approximations to ordinary differential equations, the result may diverge from the correct solution, or even diverge to $\pm\infty$, if the time spacing, δt, is chosen too large, depending on the solver used. At the same time a smaller time spacing does imply a higher numerical computational cost.

A balance must be found between each step being very fine but taking many steps to cover samples from the whole configuration space, and jumping far each step, covering different areas of the space all the time, but being so grainy that it converges to a skewed distribution or completely diverging!

Below is the TPCF for the samples obtained from a Langevin simulation with this in action, with $N_\tau = 16$, $m = 1$, $\mu = 1$, $\delta\tau = \frac{\beta}{N_\tau} = 0.5$ as in the previous systems, going $N = 1000$ steps in Langevin time with a Langevin time step of $\delta t = 0.1$:

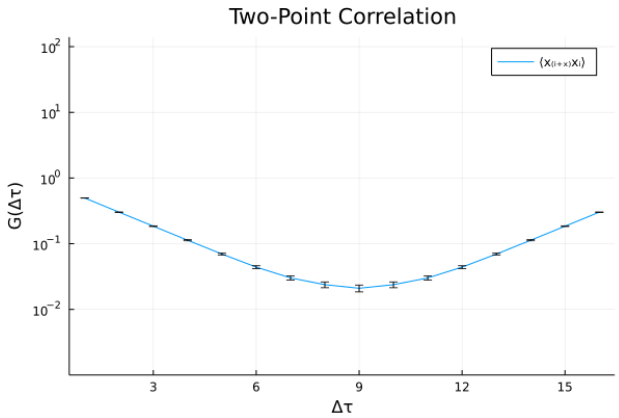


Figure 4.1: Two-Point Correlation of a Langevin simulated Harmonic Oscillator action in a 16 point lattice with nearest-neighbor coupling.

4.4 Complex Langevin

To simulate systems with complex actions, which does not have a well defined probability measure to be used in the Metropolis algorithm, the complex Langevin method can be used for some of these. As seen in the previous section, the Langevin equation is able to sample from the target distribution for real actions. To demonstrate the complex Langevin method for complex actions, a modification to the previously shown simple example will be used, where now μ can be a complex number.

For certain $\mu \in \mathbb{C}$, the samples does not converge to a stationary distribution, so in Sec. 4.4.2 reasons for why in certain limits will be explained, and for some μ the evolution can be made more stable.

Sec. 4.4.3 shows an example for how the evolution can be made more stable for the μ that should make $\langle x^2 \rangle$ converge to a solution.

The last two chapters go through the complex μ harmonic oscillator and how it is possible to extend the Euclidean time evolution to also allow for real-time dynamics by the Schwinger-Keldysh contour.

4.4.1 Complex Langevin simulations for a simple system

Recall the action from Sec. 4.2, specifically Eq. 4.4:

$$S = \frac{1}{2}\mu\phi^2 \tag{4.25}$$

To consider the case of a complex action, set $\mu = \mu_r + i\mu_i$. For $\mu_i \neq 0$ the action is complex. If the Langevin method is used to sample from the distribution according to the corresponding Langevin equation, the samples ϕ get complex. Instead of working with a single Langevin equation for ϕ, the equation is split into a real and an imaginary part Langevin equation for ϕ_r and ϕ_i respectively, where $\phi = \phi_r + i\phi_i$. Then the evolution in Langevin time is given by the two coupled, real valued Langevin equations.

To simulate this simple system, separate the corresponding discretized Langevin equation into the two coupled, real valued, discretized Langevin equations. The discretized Langevin equation for this system:

$$\begin{aligned}\phi^{n+1} &= \phi^n - \frac{dS}{d\phi}|_n \delta t + \sqrt{2\delta t}\eta^n \\ &= \phi^n - \mu\phi^n\delta t + \sqrt{2\delta t}\eta^n \end{aligned} \tag{4.26}$$

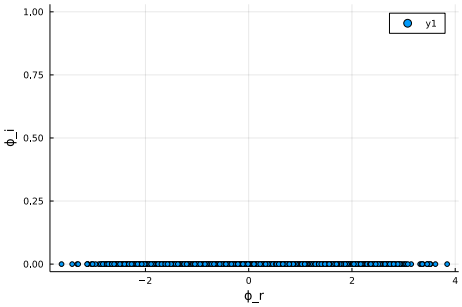
Then separating the real and imaginary evolution:

$$\begin{aligned}\phi_r^{n+1} &= \phi_r^n - \frac{dS}{d\phi}\Big|_n \delta t + \sqrt{2N_r\delta t}\eta^n \\ &= \phi_r^n - (\mu_r\phi_r^n - \mu_i\phi_i^n)\delta t + \sqrt{2\delta t}\eta^n \end{aligned} \tag{4.27}$$

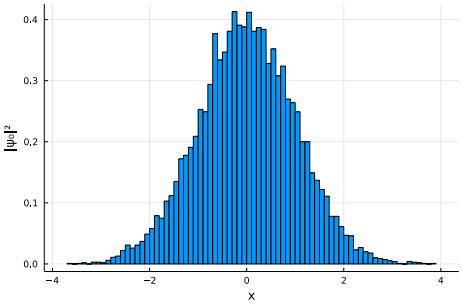
$$\begin{aligned}\phi_i^{n+1} &= \phi_i^n - \frac{dS}{d\phi}\Big|_n \delta t + \sqrt{2N_i\delta t}\eta^n \\ &= \phi_i^n - (\mu_i\phi_r^n + \mu_r\phi_i^n)\delta t \end{aligned} \tag{4.28}$$

The random noise is chosen to contribute to the real part of the evolution here, $N_r = 1$ while satisfying $N_r^2 + N_i^2 = 1$.

Using a real valued μ with initial condition on the complex part $\phi_i = 0$ in Eq. 4.27 should give the same results as a simulation using the single Langevin equation in Sec. 4.2. Each sample will then have real part according to the real Langevin equation, and the imaginary part will stay 0. Running a simulation using the set of coupled evolution equations, Eq. 4.27 and 4.28, with $\mu = 1 + 0i$, and Langevin time step $dt = 0.001$. A total number of 10'000 samples of ϕ were collected.



(a) Scatter plot of the samples.

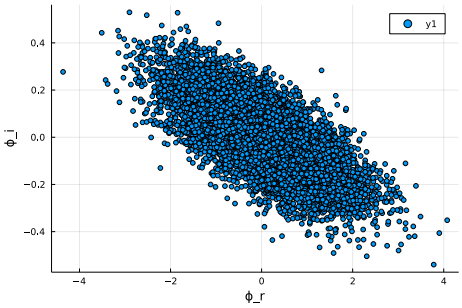


(b) Plot of the probability density distribution ($x = \psi$).

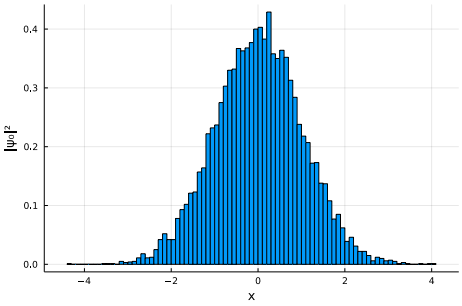
Figure 4.2: Plots of samples from a complex Langevin simulation of the Gaussian system with action $S = \mu x^2/2$, $\mu = 1 + 0i$. The 10'000 samples were collected after 3000 removed as burn in, as well as 3000 in between each saved. An explicit forward Euler solver was used.

The estimated value for $\langle \phi^2 \rangle$ from the simulation was 1.003 ± 0.014 standard error.

If instead $\mu_i \neq 0$, the second Langevin equation has a non-zero drift term (as long as $\phi \neq 0$) and ϕ^i changes over time. It would be useful to see how samples collected using the coupled complex Langevin equations to simulate the system for complex μ is distributed in the complex plane. Running a simulation using the set of coupled evolution equations, Eq. 4.27 and 4.28, with $\mu = 1 + 0.2i$, and Langevin time step $dt = 0.001$:



(a) Scatter plot of the samples.



(b) Plot of the probability density distribution for only the real part ($x = \psi_r$).

Figure 4.3: Plots of samples from a complex Langevin simulation of the Gaussian system with action $S = \mu x^2/2$, $\mu = 1 + 0.2i$. The 10'000 samples were collected after 3000 removed as burn in, as well as 3000 in between each saved. An explicit forward Euler solver was used.

In Fig. 4.3a one can see that the samples are distributed in the complex plane, the scatter plot corresponds to looking at the two dimensional discrete probability distribution from above. In Fig. 4.3b, by ignoring the imaginary part of ϕ_i, the probability distribution for the real part of ϕ_i is shown. As can be seen, for this small change in μ it does not change much, compared to Fig. 4.2b.

In the limit where $\mathrm{Re}(\mu) \to 0$ there is a problem. The Langevin evolution starts to run of to some infinity limit. The system where $\mathrm{Re}(\mu) < 0$ diverges. Convergence of the Langevin equation is partially determined by the associated Fokker-Planck equation. Additionally, convergence may not be satisfied since the problem is not well-defined, or the evolution is too coarse or the numerical solver induces errors, round-off by floating point precision and propagated truncation error. (Subsection 4.4.3) The first step, if possible, is to confirm that the system under investigation is well-defined, which will be considered in the following subsection.

4.4.2 Complex Gaussian integrals

The integral evaluated by the method in the previous section will be the complex Gaussian integral:

$$Z = \int_{-\infty}^{\infty} e^{-\mu x^2/2} dx \tag{4.29}$$

Which is defined as long as $\mu_r = \mathrm{Re}(\mu) > 0$, since then: $\sqrt{\frac{2\pi}{\mu}} e^{-\frac{\mu}{2} r^2} \big|_0^\infty$ is finite. The phase in the integral ($e^{\mu_i x^2/2}$) is a sum over infinitely swirling phases, which goes to zero compared to something with a constant phase. [3, p. 259]

$$Z = \int_{-\infty}^{\infty} e^{-\frac{\mu_r}{2} x^2} e^{-\frac{\mu_i}{2} x^2} dx \tag{4.30}$$

The limit $\mu_r \to 0$ can be evaluated as $\sqrt{\frac{2\pi}{\mu}}$. This limit is like going back to the Minkowski path integral, where the integral is the same as the real time integral. The bottomless system ($\mu_r < 0$) is not well defined, but can be extrapolated to by introducing a complex kernel in the complex Langevin equation, which stabilizes the evolution. [8]

The expectation values of this integral:

$$\langle x \rangle = \frac{1}{Z} \int_{-\infty}^{\infty} x e^{-\mu x^2/2} dx = 0 \tag{4.31}$$

As long as the integral is well-defined, $\mathrm{Re}(\frac{\mu}{2}x^2) > 0$, the integrand is odd and the integral evaluates to 0.
By partial integration:

$$\langle x^2 \rangle = \frac{1}{Z}\int_{-\infty}^{\infty} x^2 e^{-\mu x^2/2} dx \tag{4.32}$$

$$= \frac{1}{Z}\Big(\int_{-\infty}^{\infty} x e^{-\mu x^2/2} dx + \int_{-\infty}^{\infty} \frac{1}{\mu} e^{-\mu x^2/2} dx\Big) \tag{4.33}$$

$$= \frac{1}{\mu}\frac{\int_{-\infty}^{\infty} e^{-\mu x^2/2} dx}{Z} = \frac{1}{\mu} \tag{4.34}$$

The following is inspired by the paper of Okamoto et al. [8].

To see why the complex Langevin method fails for this system, consider a plot of the results for $\mu = exp(n\pi i/6)$ with values $n = (0, 11)$.

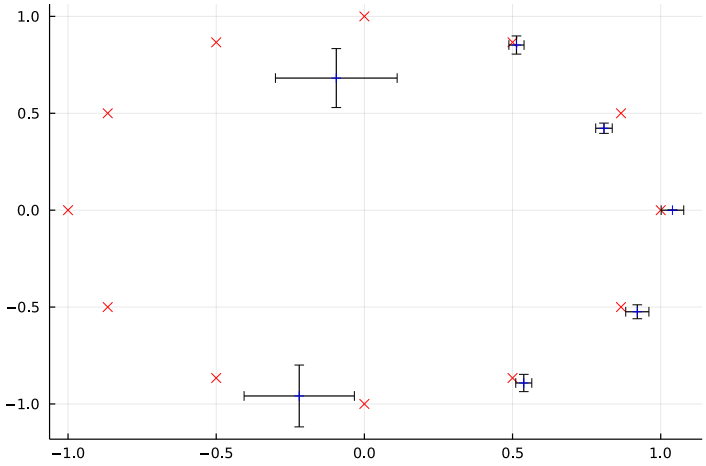


Figure 4.4: Plot of the Gaussian system with action $S = \mu x^2/2$, $\mu = exp(n\pi i/6)$. The data points and statistical error was estimated by expectation values of 64 runs, each with 2000 updates, of a Langevin simulation using an explicit forward Euler solver.

When $\mathrm{Re}(\mu) > 0$ the system is stable and the solution converges to $1/\mu$, but for $\mathrm{Re}(\mu) = 0$ the simulated expectation value blows up if left simulating for too long. The data was acquired by numerical simulation with 64 runs of 2000 updates to calculate mean and statistical error. If each run is allowed more updates, the solution for $\mathrm{Re}(\mu) = 0$ usually runs of, which indicate this is an unstable solution. For $\mathrm{Re}(\mu) < 0$ the result runs of from the beginning.

4.4.3 Stable Solvers

In the previous subsection, the Langevin method started to run of for $\mathrm{Re}(\mu) \approx 0$, and completely run of for $\mathrm{Re}(\mu) < 0$. Since the integral is not defined when the real part $\mathrm{Re}(\mu) < 1$, it is not possible to evaluate it either, but when the integral is defined ($\mathrm{Re}(\mu) > 0$), the solution exists, and the Langevin evolution should not run of.

The numerical solver introduces truncation errors which propagates through the solutions. If the error is large enough this can cause the evolution to run of. Typically the Forward Euler solver overestimates the solution, which when combined with propagated errors caused the evolution to run of after only a few time steps in the case of $\mathrm{Re}(\mu)$ being close to 0 but still positive. By choosing another solver the evolution can be made more stable.

For the implicit Euler, which solves for the new step as:

$$\phi^r_{n+1} = \phi^r_n + \frac{dS}{d\phi_{n+1}}\delta t + \sqrt{2N_r\delta t}\eta = \phi^r_n + \mu^r\phi^r_{n+1} - \mu^i\phi^i_{n+1} + \sqrt{2\delta t}\eta \quad (4.35)$$

$$\phi^i_{n+1} = \phi^i_n + \frac{dS}{d\phi_{n+1}}\delta t + \sqrt{2N_i\delta t}\eta = \phi^i_n + \mu^i\phi^r_{n+1} + \mu^r\phi^i_{n+1} \quad (4.36)$$

the solutions are typically undershot, and the running away caused by propagated errors in the solver is more often overcome.

Using the Euler-Maruyama method from StochasticDiffSolvers.jl, ImplicitEM(θ), with $\theta = 1$ for the implicit Euler method, the solutions converge to the expected solutions for the well-defined systems, as long as the propagated errors are low enough.

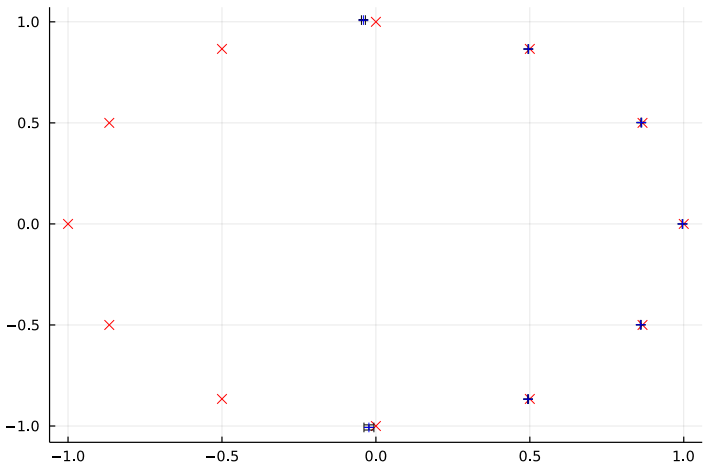


Figure 4.5: Plot of the Gaussian system with action $S = \mu x^2/2$, $\mu = exp(n\pi i/6)$. The data points and statistical error was estimated by expectation values of 64 runs, each with 2000 updates, of a Langevin simulation using the Euler-Maruyama solver from the Julia package StochasticDiffSolvers.jl.

Comparing to Fig. 4.4, the values of the expectation values are much more reliably estimated.

4.4.4 Complex Harmonic Oscillator

Running a short complex Langevin simulation for the complex action case of the harmonic oscillator system in Sec. 4.3, with complex μ.

As before, the system must thermalize. Initializing an array of 16 ϕ_r and 16 ϕ_i, all at $20. + 0i$:

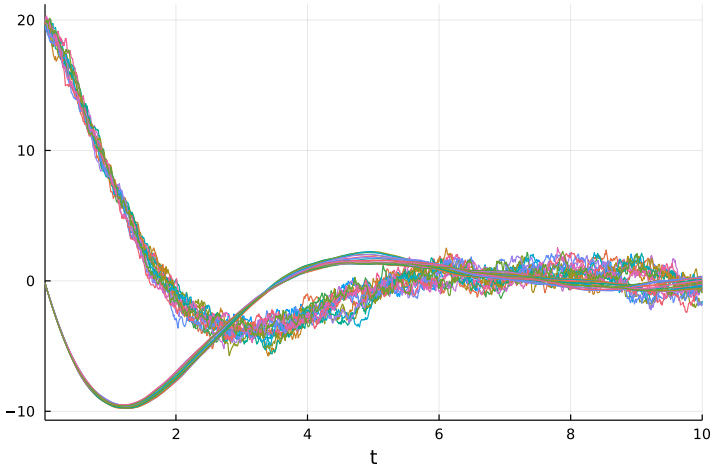


Figure 4.6: Thermalization of a Complex Langevin algorithm. The real and imaginary parts of the 16 time points in euclidean time separated by $\delta\tau = 0.5$ evolved with Complex Langevin step size 0.0001. The real parts initialized at 20, and the imaginary parts initialized at 0. As can be seen, the real parts are fluctuating around a slope because of the random contributions to the evolution, while the imaginary part is much more clean because of no direct noise. The real parts goes to about -3 at $t \approx 3$ because of the imaginary part couplings.

Chapter 5

Numerical solvers for Complex Langevin

Stable numerical solvers for complex Langevin is very central to current developments in the field. To find numerical Stochastic Differential Equation solvers that are precise enough to not cause instabilities in the Langevin evolution, which also are efficient. For stiff problems regular solvers are inefficient, therefore "stiff solvers" are best applied for these problems.

From [9], recommended methods depend on the problem:
For non-stiff problems, with non-commutative noise, difficult problems usually require adaptive time stepping to be efficient. In this case, "LambaEM" is adaptive and handle general non-diagonal problems. If adaptivity is not necessary, "EM" (Euler-Maruyama) is a good choice.

For stiff problems with additive noise, the higher order adaptive method "SKenCarp" is highly preferred and will solve problems with similar efficiency as ODEs.

If only an estimation for the expected value of the solution is required, i.e., if one is only interested in an accurate draw from the distribution induced by a given SDE, the use of high weak order solvers is recommended. Specifically, "DRI1" is preferred for a high number of Wiener processes.

For later implementations of the algorithms discussed in this thesis, it would be beneficial to test the different solver algorithms against each other for different problems. The criteria for good solvers are computation time and same step-size convergence for a given problem.

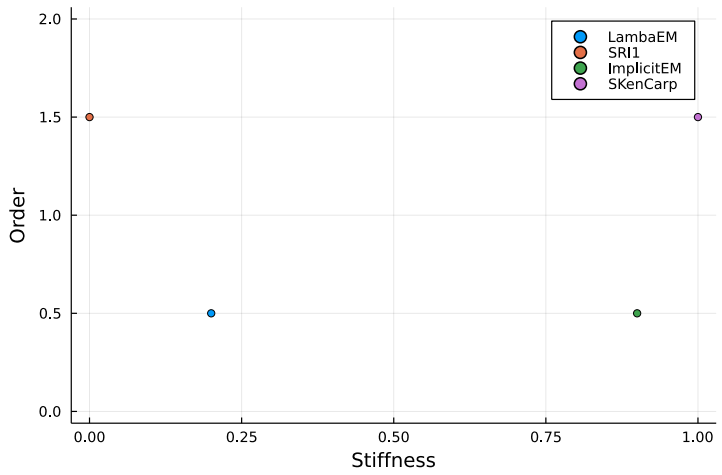


Figure 5.1: Plot of order vs. approximate stiffness of different chosen solvers relevant to Langevin and Complex Langevin simulations. The value $1/\mu$ is plotted bottom right.

By using the harmonic and anharmonic oscillators as the systems to be simulated, testing will be done using different solvers from the StochasticDiffEq julia package.

To test the performance of the solvers, only the time to solve, $solve(SDEProblem)$, will be measured. The time to set up parameters and building the SDEProblem will be the same in all the different cases for the same system. Some solvers have the option of allowing adaptive step size to more easily meet the tolerance error set. For testing against methods without adaptive stepsize, the adaptive stepsize option will be turned off.

Since Julia uses a lot of time when a new function not part of the standard library (like $Plots.plot()$) is run for the first time, the first test run time is always discarded. For example, solving a stochastic differential equation of a harmonic oscillator using the DifferentialEquations solver Euler() used 64.7 seconds from a fresh Julia REPL, while the second time, it only used 0.56 seconds. It should be noted that the initial compile time for functions from packages can be reduced or removed by creating and using an image of a Julia REPL with the compiled functions. See [10] for more information.

The initial condition of the systems used while testing the solvers will be the same for all. All tests will be done on a personal desktop computer with OS Microsoft Windows 10 Home, in WSL2 (Windows Subsystem for Linux) with Ubuntu.

5.1 Solvers for Complex Langevin

Different solvers will be tested for the complex Gaussian integral $e^{-\mu x^2}$, setting $\mu = e^{-i\pi/4}$. As a good first test, a solver which is already known, the Euler-Maruyama will be used. Setting $\theta = 0$ uses a forward Euler type solver, while a setting of $\theta = 1$ uses a fully implicit Euler type solver. A choice of $\theta = 0.5$ uses a partially implicit Euler type solver.

Using Langevin time step $\delta t = 0.01$, and time span $t = (0.0, 3000)$ and $saveat = \delta t$, collecting 30'000 correlated samples. For the Gaussian system with action $S = \frac{1}{2}(e^{-i\pi/4})\phi$ the solvers time til completion and expectation values are given:

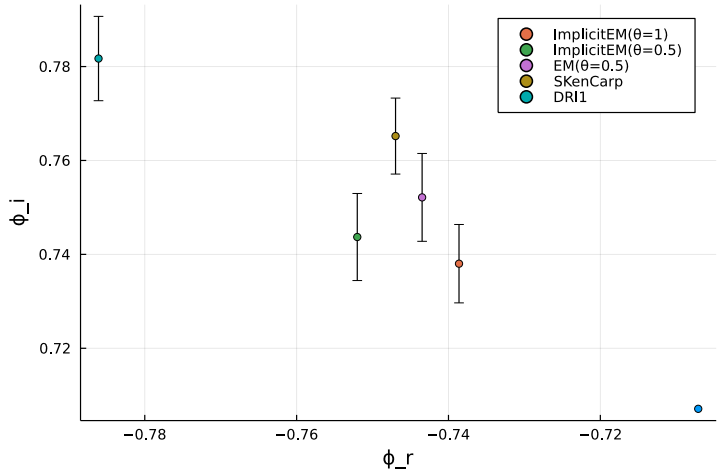


Figure 5.2: Plot of results for a single run for different solvers.

The ImplicitEM() solver with different arguments used 4.0 seconds, the EM() solver used 0.98 seconds, the SKenCarp() solver used 9.66 seconds, and the DRI1() solver used 2.73 seconds, in a single run.

A proper benchmark where different runs are averaged would be more appropriate in the future.

Acknowledgements

I would like to sincerely thank my supervisor Alexander Karl Rothkopf, Associate Professor in Physics at the University of Stavanger (UiS), for the great guidance throughout the work on this thesis. These two last semesters have taught me so much, and I am grateful for all the time you have spent teaching me these new concepts.

I wish to thank Daniel Alvestad, PhD student at UiS, for introducing me to and helping me in the process of working with the programming language Julia.

I also would like to express my thanks to Prof. Rothkopf and Postdoc. Rasmus Normann Larsen together with PhD students Daniel Alvestad and Gaurang Yatin Parkar for great group meetings about complex Langevin throughout these last two semesters.

Lastly, I wish to thank Ellen Mong for supporting me through thick and thin during the work on this thesis.

Bibliography

[1] "Reference to the julia code used in this thesis." (), [Online]. Available: https://github.com/HenrikHag/myProject (visited on 06/15/2022).

[2] D. Alvestad, R. Larsen, and A. Rothkopf, "Stable solvers for real-time complex langevin," 2021. DOI: https://doi.org/10.48550/arXiv.2105.02735.

[3] M. D. Schwartz, *Quantum Field Theory and the Standard Model*. Cambridge University Press, 2014, ISBN: 978-1-107-03473-0.

[4] M. J. E. Westbroek, P. R. King, D. D. Vvedensky, and S. Dürr, "User's guide to monte carlo methods for evaluating path integrals," *American Journal of Physics*, vol. 86, no. 4, pp. 293–304, Apr. 2018. DOI: 10.1119/1.5024926. [Online]. Available: https://doi.org/10.1119%2F1.5024926.

[5] D. J. Griffiths and D. F. Schroeter, *Introduction to Quantum Mechanics, Third Edition*. Cambridge University Press, 2018, ISBN: 9781107189638.

[6] J. Zinn-Justin, *Quantum Field Theory and Critical Phenomena. 5th ed.* Oxford University Press, 2021, pp. 831–839. DOI: 10.1093/oso/9780198834625.001.0001.

[7] "Working with langevin and fokkerplanck equations online lecture notes." (), [Online]. Available: https://www2.ph.ed.ac.uk/~dmarendu/ASP/Section16.pdf (visited on 06/15/2022).

[8] H. Okamoto, K. Okano, L. Schülke, and S. Tanaka, "The role of a kernel in complex langevin systems," *Nuclear Physics B*, vol. 324, no. 3, pp. 684–714, 1989, ISSN: 0550-3213. DOI: https://doi.org/10.1016/0550-3213(89)90526-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0550321389905269.

[9] "Differentialequations.jl - sde solvers documentation." (), [Online]. Available: https://diffeq.sciml.ai/stable/solvers/sde_solve/ (visited on 05/30/2022).

[10] "Julia documentation, system image." (), [Online]. Available: `https://docs.julialang.org/en/v1/devdocs/sysimg/` (visited on 06/03/2022).