KU LEUVEN

Advanced Monte Carlo Methods

Final Report

Bert Depoorter
MSc. Master of Physics
Option Theoretical Physics
Academic Year 2024-2025

Faculty of Science
Department of Physics & Astronomy
Institute for Theoretical Physics
G0U08a: Computational Physics: Advanced Monte Carlo Methods
Instructor: Prof. Dr. Enrico Carlon

Introduction

In this work I present the solutions to the exercises at the end of the lecture notes for the course on Advanced Monte Carlo Methods. All simulations were done using code written in python.

4.1 Non-uniform Random Number Generators

This first exercise is concerned with non-uniform RNGs. We are asked to develop biased RNGs according to some probability distribution. There are multiple ways to do this.

Using inverse cumulative distribution function Here we first calculate the inverse cumulative distribution function of the given probability density distribution. Suppose we want to sample $f(x)$. We first calculate

$$F(x) = \int_{-\infty}^{x} f(x')dx'$$

We then invert this function to $F^{-1}(x)$. Then, if we generate $x_i \sim \mathcal{U}(0,1)$ then the numbers $y_i = F^{-1}(x_i)$ will be distributed according to $f(x)$.

Uniform sampling in $[-2,1]$ corresponds to the probability density

$$f(x) = \begin{cases} \frac{1}{3} & x \in [-2,1] \\ 0 & \text{else} \end{cases} \tag{1}$$

Integrating this gives the cumulative probability density

$$F(x) = \begin{cases} 0 & x < -2 \\ \frac{x-2}{3} & x \in [-2,1] \\ 1 & x > 1 \end{cases} \tag{2}$$

Implementing this gives the histogram of sampled values shown in Figure 1.

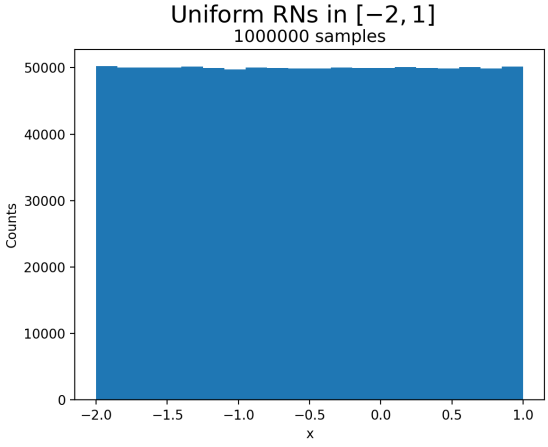


Figure 1: Uniform distribution on $[-2,1]$. Sample size is 10^6.

for the exponential distribution we have $f(x) = \exp(-x) \forall x \geq 0, f(x) = 0$ elsewhere. This corresponds to

$$F^{-1}(x) = \begin{cases} -\ln(1-x) & x \in [0,1] \\ 0 & \text{else} \end{cases} \tag{3}$$

This gives the normalized histogram shown in Figure 2.

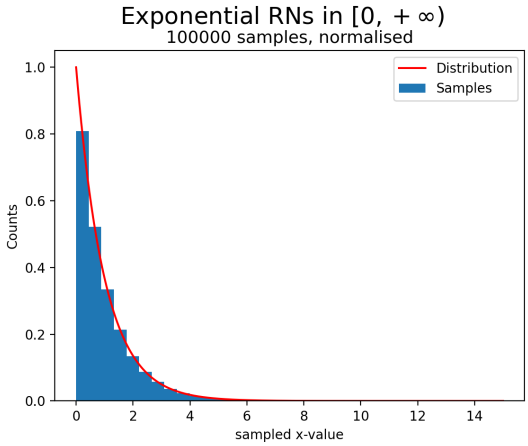


Figure 2: Normalized histogram for the exponential distribution. The analytical PDF is plotted as well for comparison. Number of samples is 10^6.

The third function we are asked to sample is the following:

$$f(x) = \begin{cases} \frac{x+1}{2} & x \in [-1, 1] \\ 0 & \text{else} \end{cases} \tag{4}$$

Transforming this to $F^{-1}(x)$ gives

$$F^{-1}(x) = \{. \tag{5}$$

This gives rise to the histogram shown in Figure 3

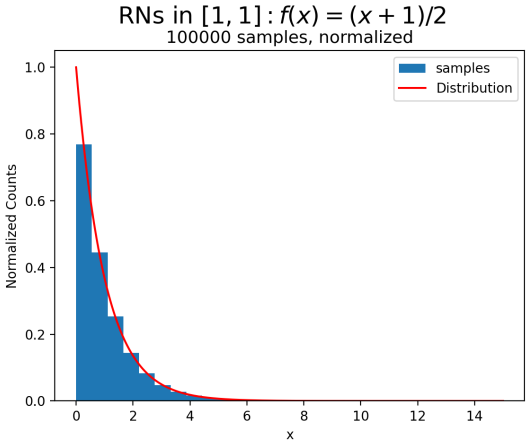


Figure 3: Caption

Hit and Miss Method We can also use the hit and miss method to sample random numbers according to some distribution $f(x)$. Suppose $f(x)$ is defined on $[a, b]$ and we have $M > f(x) \forall x \in [a, b]$. We then generate 2 random numbers t, s uniformly in respectively $[a, b]$ and $[0, M]$. If $s > f(t)$ this is a miss and the t-value is discarded. If $s \leq f(t)$ we keep the t-value as the random number. This method is computationally more expensive since not all numbers are used. For a bad choice of M this can take a lot of time to sample many points of the distribution.

We use this method to sample again from the distribution

$$f(x) = \begin{cases} \frac{x+1}{2} & x \in [-1, 1] \\ 0 & \text{else} \end{cases} \tag{6}$$

We choose $M = 1.1$ such that it is always larger than any value of $f(x)$. This gives the histogram shown in Figure 4.

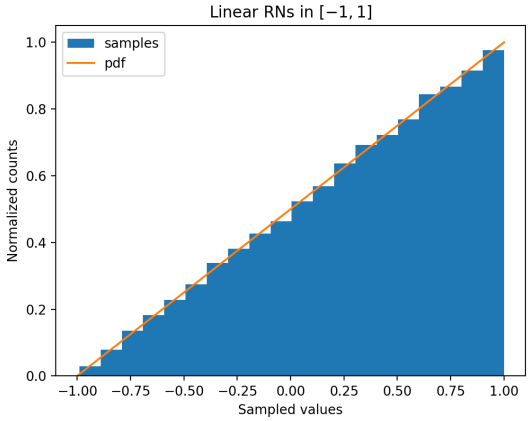


Figure 4: The histogram for the Hit and Miss method for the probability distribution $f(x) = (x + 1)/2$ on $[-1, 1]$. We used $M = 1.1$, $N = 100000$.

We also do this for the following probability distribution.

$$f(x) = \begin{cases} -x(1-x)e^{-x^2}\log(1-x) & x \in [0, 1] \\ 0 & \text{else} \end{cases} \tag{7}$$

This function is not yet normalized. We can compute the normalization factor by the integral

$$Z = \int_0^1 -x(1-x)e^{-x^2}\log(1-x)dx = 0.0901135 \tag{8}$$

With this normalization taken into account, we show the histogram with the PDF plotted on top in Figure 5.

Points inside unit circle We use the hit and miss method to generate points distributed uniformly in the unit circle, using 2 different methods.

First of all we use the hit and miss method. We generate $x, y \sim \mathcal{U}(-1, 1)$ and then impose the condition $x^2 + y^2 \leq 1$ for acceptance. Points outside the unit circle are discarded.

Secondly, we can transform to polar coordinates (r, θ). That means we sample $\theta \sim \mathcal{U}(0, 2\pi)$ and $r' \sim \mathcal{U}(0, 1)$, where we then apply the transformation $r = \sqrt{r'}$ to evenly sample points close to the origin as those close to the unit circle. The result of both methods is shown in Figure 6.

4.2 Gaussian Random Number Generator

In this exercise we generate random numbers distributed according the density function

$$f(x, y)dxdy = \frac{1}{2\pi}\exp\left(-\frac{x^2 + y^2}{2}\right)dxdy \tag{9}$$

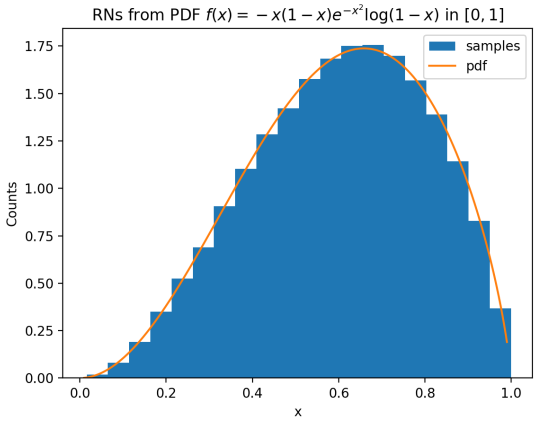


Figure 5: Generated histogram of samples obtained by using the hit and miss method for the function defined in Equation (7). We used $M = 2$, $N = 100000$.

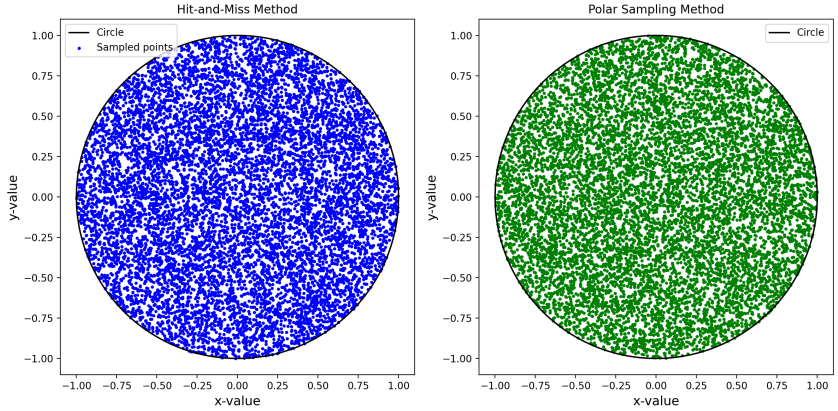


Figure 6: The sampled points for both methods described above. We see that the points are sampled evenly over the full circle. Both figures contain $N = 10000$ points.

This can be transformed to polar coordinates. We then need to sample $\theta \sim \mathcal{U}(0, 2\pi)$ and r according to

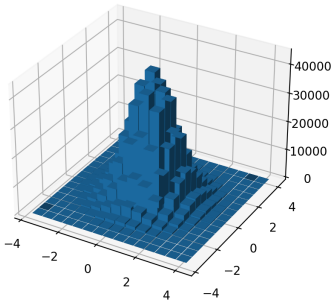
$$F(r) = \int_0^r \exp\left(-\frac{z^2}{2}\right) z dz$$

Then transforming back to cartesian coordinates result in 2 gaussianly distributed numbers.

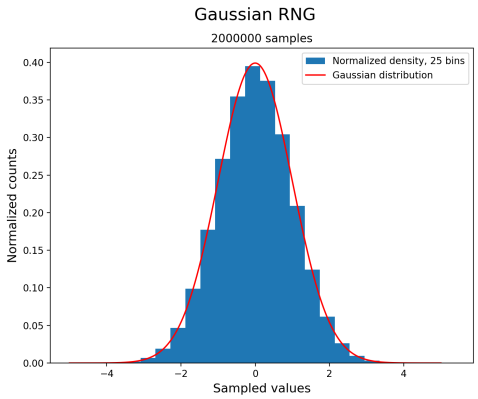
The procedure we use is the following:

- Generate $u, v \sim \mathcal{U}(0, 1)$
- Compute $r = \sqrt{-2\ln(u)}$ and $\theta = 2\pi v$
- Compute $x = r\cos(\theta)$, $y = r\sin(\theta)$

This results in the distribution shown on Figure ??



(a) This 3D histogram shows the distribution of the (x, y)-pairs generated by the Box-Muller transformation. We see the typical bell shape arise.



(b) The x and y values generated using the Box-Muller transformation are all added into one array and plotted on this histogram. We also normalize and compare to the analytical expression.

In order to sample random numbers according to a more general normal distribution $x, y \sim \mathcal{N}(\mu, \sigma^2)$ we slightly adapt the algorithm. The probability density is given as

$$f(x, y)dxdy = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-\mu)^2 + (y-\mu)^2}{2\sigma^2}\right) dxdy$$

Now we first do a transformation

$$t = \frac{x-\mu}{\sigma}, s = \frac{y-\mu}{\sigma}$$

This gives

$$f(t, s)dtds = \frac{1}{2\pi} \exp\left(-\frac{t^2 + s^2}{2}\right) dtds$$

Transforming this into polar coordinates gives

$$f(r, \theta)drd\theta = \frac{1}{2\pi} \exp\left(-\frac{r^2}{2}\right) drd\theta$$

The procedure we use is the following:

-
- Generate $u, v \sim \mathcal{U}(0, 1)$
- Compute $r = \sqrt{-2\ln(u)}$ and $\theta = 2\pi v$
- Compute $t = r\cos(\theta)$, $s = r\sin(theta)$
- Transform to $x = \sigma t + \mu$ and $y = \sigma s + \mu$

We can plot the x and y values separately, to show that both sets of generated random numbers also obey the gaussian distribution. Figure 8 shows this for the general case.

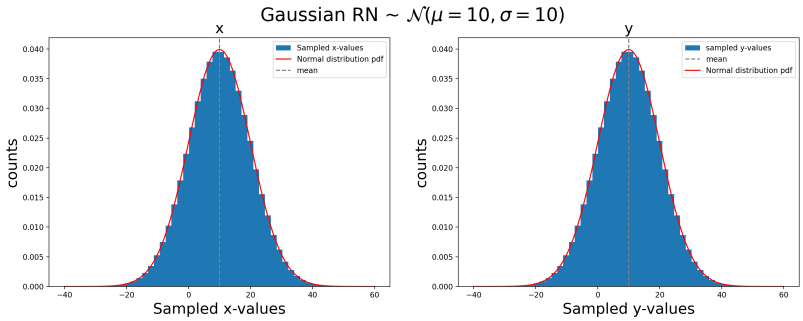


Figure 8: Histogram of the sampled x and y values. Here we have made a particular choice for the mean μ and the standard deviation σ. The sample size is 10^5.

4.3 Monte Carlo Integration (I)

We are asked to compute the value of the following integral:

$$\mathcal{J} = \int_{-\infty}^{+\infty} \exp\left(-\frac{r^2}{2}\right)(x+y+z)^2 dxdydz$$

where $r^2 = x^2 + y^2 + z^2$. To perform the calculation we take the factor $\exp(-r^2/2)$ as a density function on the domain of the integral. When doing so, one has to be careful with the normalization factor. This integral can also be calculated analytically, and also using another type of Monte Carlo sampling based on the Metropolis algorithm. We compare these different methods.

Analytical Solution

The integral is analytically solvable. For this we first transform to polar coordinates, using the following set of transformations:

$$(x,y,z) \mapsto (r,\theta,\phi) : \begin{cases} x &= r\cos(\theta)\sin(\phi) \\ y &= r\sin(\theta)\sin(\phi) \\ z &= r\cos(\phi) \end{cases} \tag{10}$$

where we have the bounds $r \in [0, +\infty)$, $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. The determinant of the Jacobian needs to be taken into account, this term gives $dxdydz = r^2\sin(\theta)drd\theta d\phi$. Now we can work out the integral analytically

$$\begin{aligned} \mathcal{J} &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \exp\left(-\frac{x^2+y^2+z^2}{2}\right)(x+y+z)^2 dxdydz \\ &= \int_0^{+\infty}\int_0^{\pi}\int_0^{2\pi} e^{-\frac{r^2}{2}}\left(r\cos(\theta)\sin(\phi) + r\sin(\theta)\sin(\phi) + r\cos(\phi)\right)^2 r^2\sin(\theta)drd\theta d\phi \\ &= \int_0^{+\infty} e^{-\frac{r^2}{2}} r^4 dr \int_0^{\pi}\int_0^{2\pi} \left(\cos(\theta)\sin(\phi) + \sin(\theta)\sin(\phi) + \cos(\phi)\right)^2 \sin(\theta)d\theta d\phi \end{aligned} \tag{11}$$

The radial integral is a variation on the standard gaussian integral, which we can look up to be

$$\int_0^{+\infty} e^{-\frac{r^2}{2}} r^4 dr = 3\sqrt{\frac{\pi}{2}} \tag{12}$$

For the angular integral we have to do some more work. We work out the square and then split the integral by linearity.

$$\begin{aligned}
&\int_0^{\pi}\int_0^{2\pi} \left(\cos(\theta)\sin(\phi)+\sin(\theta)\sin(\phi)+\cos(\phi)\right)^2 \sin(\theta)d\theta d\phi \\
&= \int_0^{\pi}\int_0^{2\pi} \Big(\cos^2(\theta)\sin^2(\phi)+\sin^2(\theta)\sin^2(\phi)+\cos^2(\phi)+2\cos(\theta)\sin(\phi)\sin(\theta)\sin(\phi) \\
&\qquad 2\sin(\theta)\sin(\phi)\cos(\phi)+2\cos(\theta)\sin(\phi)\cos(\phi)\Big)\sin(\theta)d\theta d\phi
\end{aligned} \tag{13}$$

Using some goniometric formulae this simplifies to

$$\begin{aligned}
&= \int_0^{\pi}\int_0^{2\pi} \Big(1+\sin(2\theta)\sin^2(\phi)+\sin(\theta)\sin(2\phi)+\cos(\theta)\sin(2\phi)\Big)\sin(\theta)d\theta d\phi \\
&= \int_0^{\pi}\int_0^{2\pi} \sin(\theta)d\theta d\phi + \int_0^{\pi}\int_0^{2\pi} \sin(\theta)\sin(2\theta)\sin^2(\phi)d\theta d\phi \\
&\quad + \int_0^{\pi}\int_0^{2\pi} \sin(\theta)(\sin(\theta)+\cos(\theta))\sin(2\phi)d\theta d\phi
\end{aligned} \tag{14}$$

We can again split the integral in a $\theta-$ and $\phi-$integral for each of the three terms. We compute these separately.

- $\int_0^{2\pi}\sin(2\phi)d\phi = -\frac{\cos(2\phi)}{2}\Big|_0^{2\pi} = 0$
- $\int_0^{2\pi}\sin^2(\phi)d\phi = \int_0^{2\pi}\frac{1-\cos(2\phi)}{2}d\phi = \frac{\phi-\sin(2\phi)/2}{2}\Big|_0^{2\pi} = \pi$
- $\int_0^{\pi}\sin(\theta)d\theta = -\cos(\theta)\big|_0^{\pi} = 2$
- $\int_0^{\pi}\sin(\theta)\sin(2\theta)d\theta = 2\int_0^{\pi}\sin^2(\theta)cos(\theta)d\theta = 2\int_0^{\pi}\sin^2(\theta)d\sin(\theta) = 2\,\frac{\sin^3(\theta)}{3}\Big|_0^{\pi} = 0$

Combining these results for Equation (14) only leaves the first integral

$$\begin{aligned}
&\int_0^{\pi}\int_0^{2\pi} \Big(1+\sin(2\theta)\sin^2(\phi)+\sin(\theta)\sin(2\phi)+\cos(\theta)\sin(2\phi)\Big)\sin(\theta)d\theta d\phi \\
&= \int_0^{\pi}\int_0^{2\pi} \sin(\theta)d\theta d\phi = \int_0^{\pi}\sin(\theta)d\theta\int_0^{2\pi} d\phi \\
&= 2\times 2\pi = 4\pi
\end{aligned} \tag{15}$$

So finally we find the analytical solution for this integral as the product of the radial and angular integrals:

$$\mathcal{J} = 3\sqrt{\frac{\pi}{2}}\times 4\pi = 6\sqrt{2}\sqrt{\pi}^3 \simeq 47.2488... \tag{16}$$

Use Gaussian RNG

We use a gaussian RNG to generate x, y, z and we use these to compute the integral using this method.

The biased random number generator is in this case the following function:

$$W(x_n, y_n, z_n) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{x_n^2}{2})\frac{1}{\sqrt{2\pi}}\exp(-\frac{y_n^2}{2})\frac{1}{\sqrt{2\pi}}\exp(-\frac{z_n^2}{2})$$

which simplifies to

$$W(\vec{x}_n) = \frac{1}{\sqrt{2\pi}^3} \exp(-\frac{\vec{x}_n^2}{2})$$

Now we approximate the integral as

$$\mathcal{J} \simeq \frac{1}{N} \sum_{n=1}^{N} \frac{f(\vec{x}_n)}{W(\vec{x}_n)} = \frac{1}{N} \sum_{n=1}^{N} \frac{\exp\left(-\frac{x_n^2+y_n^2+z_n^2}{2}\right)(x_n+y_n+z_n)^2}{\frac{1}{\sqrt{2\pi}}\exp(-\frac{x_n^2}{2})\frac{1}{\sqrt{2\pi}}\exp(-\frac{y_n^2}{2})\frac{1}{\sqrt{2\pi}}\exp(-\frac{z_n^2}{2})}$$

reducing this to

$$\mathcal{J} \simeq \frac{1}{N} \sum_{n=1}^{N} \sqrt{2\pi}^3 (x_n^2 + y_n^2 + z_n^2)$$

For this we only need to sample x_n, y_n, z_n from a normal distribution with $\mu = 0, \sigma = 1$. Doing this results in the following block of code.

[1]:
```
import numpy as np

# generate gaussian random numbers
n_sam = 1E6
x = np.random.normal(0, 1, int(n_sam))
y = np.random.normal(0, 1, int(n_sam))
z = np.random.normal(0, 1, int(n_sam))

integral = 0
for i in range(int(n_sam)):
    integral += (x[i]+y[i]+z[i])**2
val_sam = 1/n_sam*(integral)*np.sqrt(2*np.pi)**3

val_exact = 12*np.pi*np.sqrt(2*np.pi)/2

print(r"The sampled value is: ", round(val_sam,5))
print(r"The exact value is:   ", round(val_exact,5))
```

```
The sampled value is:  47.28118
The exact value is:    47.24883
```

We see that by sampling the gaussian RNs we get a pretty good estimate for the integral value. Better accuracy is achieved when we draw more random numbers, but this increases the computational cost. This code snippet takes less than a second to run.

One could imagine using the factor $(x+y+z)^2$ as the density distribution from which we draw random numbers, and $e^{-r^2/2}$ as the integrand. However, we require that

$$\int_{\mathbb{R}^3} W(x,y,z)dxdydz = 1 \tag{17}$$

for $W(x,y,z)$ to be a probability density and this is not satisfied for the choice $W(x,y,z) = (x+y+z)^2$. This function is not normalizable over $\mathbb{R}^3$.

Lastly, we plot the variance σ^2 for different sample sizes of N. This should decay with $\sigma \sim 1/\sqrt{N}$. For this we make a list of different sample sizes:

$$N = [10^3, 5\times10^3, 10^4, 5\times10^4, 10^5, 5\times10^5, 10^6, 5\times10^6, 10^7, 5\times10^7]$$

For each N-value we sample the integral value 10 times. We plot the resulting variance of these 10 runs together with the reference behaviour on

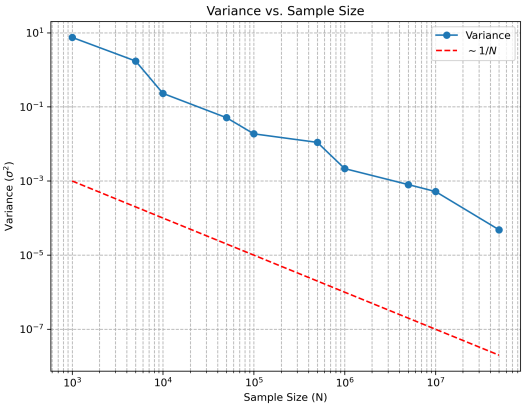


Figure 9: Variance for different sample sizes. For each sample size the algorithm runs 10 times, and from these values we compute the variance.

4.4 Monte Carlo integration (II)

We compute the same integral again, this time using the Metropolis-Hastings algorithm instead of gaussian random numbers. If we require detailed balance

$$q(\vec{r})P(\vec{r} \to \vec{r}') = q(\vec{r}')P(\vec{r}' \to \vec{r})$$

for the rates for any pairs of points $\vec{r}$ and $\vec{r}'$, then the dynamics will converge to the distribution $q(\vec{r})$ for a large number of sampled points. We choose the transition probabilities such that we have

$$\frac{P(\vec{r} \to \vec{r}')}{P(\vec{r}' \to \vec{r})} = \frac{\omega(\vec{r}')}{\omega(\vec{r})} = \exp\left(-\frac{r'^2 - r^2}{2}\right)$$

The transition probabilities are split as $P(\vec{r} \to \vec{r}') = g(\vec{r} \to \vec{r}')A(\vec{r} \to \vec{r}')$ where g is the selection probability and A the acceptance ratio.

The g is generated as follows: starting from an initial point $\vec{r}$ we select a new point $\vec{r}' = \vec{r} + \vec{\delta}$ where $\delta = (\delta_x, \delta_y, \delta_z)$ is a random shift and the three components are independently chosen from $W_h(\delta)$ a uniform distribution centered in 0 and of width h

$$W_h(\delta) = \begin{cases} 1/h & \text{if } |\delta| < h/2 \\ 0 & \text{otherwise} \end{cases}$$

The acceptance ratio for the Metropolis algorithm is given by

$$A(\delta) = \begin{cases} 1 & \text{if } r'^2 - r^2 < 0 \\ \exp\left(-\frac{r'^2 - r^2}{2}\right) & \text{otherwise} \end{cases}$$

In short this means that we jump to a new site and accept the jump if the new site is closer to the origin. If we are further away from the origin we accept the jump with a probability $\exp(-\frac{r'^2 - r^2}{2}) \leq 1$.

First of all we implement the Monte Carlo algorithm as described in python. Parameters for the simulation are:

- $N = 10^4$ MC steps
- initial position: $\vec{r}_0 = (0, 0, 0)$ and $\vec{r}_0 = (10, 10, 10)$

- We repeat the sampling for $h = 1$ and $h = 0.1$.

The resulting trajectories are plotted in the x-y plane. For clarity we plot the initial and final position as well. The results are shown in Figure 10 If we plot the histogram we obtain the

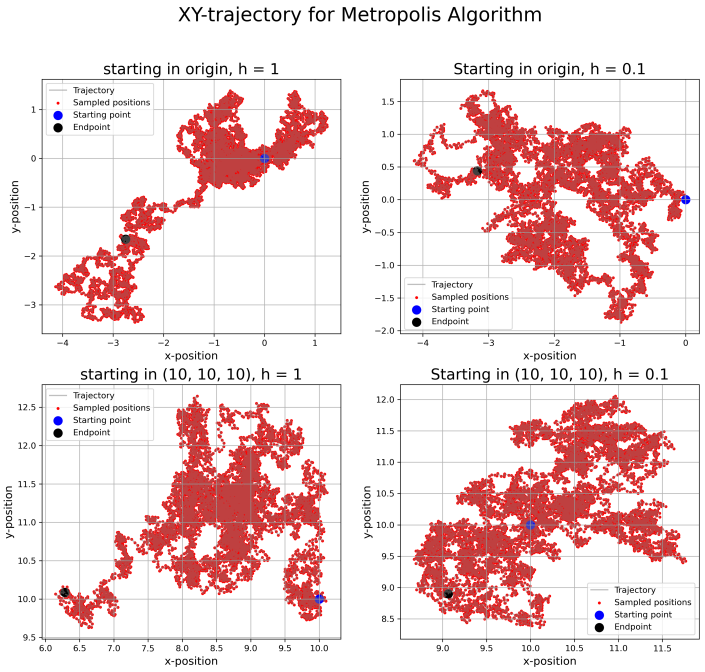


Figure 10: The Trajectories sampled by the Metropolis method. Titles on the figures indicate the value of h used, as well as the initial position. All trajectories are built from 10.000 MC steps. In terms of time the algorithm takes a few seconds to run.

result shown on Figure 11. Here the analytical model plotted on the plot is the function

$$f(r) = \frac{4\pi}{\sqrt{2\pi}^3} r^2 e^{-r^2/2} \tag{18}$$

where the r^2 come from the jacobian when going to spherical coordinates.

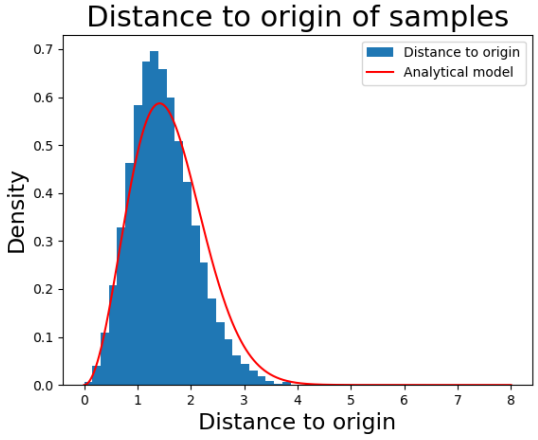


Figure 11: distance to the origin of sampled random walk, plotted against the analytical model we would expect.

Estimate Integral We again want to estimate the integral value. This can be done in the following way using the probability distribution found in 11

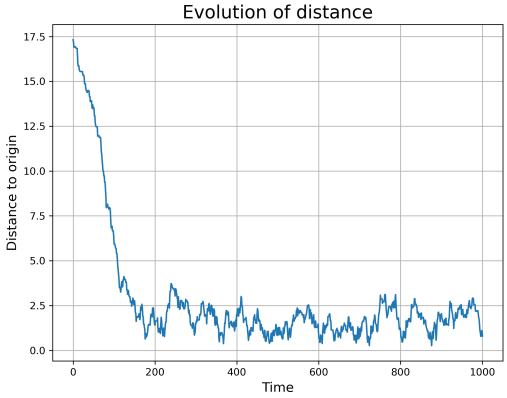
$$I = \frac{1}{N}\sum_{i=1}^{N}\frac{f(r_i)}{W(r_i)} = \frac{1}{N}\sum_{i=1}^{N}\frac{r_i^4\sin(\theta_i)(\cos(\theta_i)\sin(\phi_i)+\sin(\theta_i)\sin(\phi_i)+\cos(\phi_i))^2\exp(-\frac{r_i^2}{2})}{\frac{1}{\sqrt{2\pi}^3}r_i^2\exp(-\frac{r_i^2}{2})}$$

where we have transformed $f(x) = (x+y+z)**2 = r^2(\cos(\theta)\sin(\phi)+\sin(\theta)\sin(\phi)+\cos(\phi))^2$ to spherical coordinates. This gives for us to compute the corresponding angles θ_i, ϕ_i and then the integral can be computed as

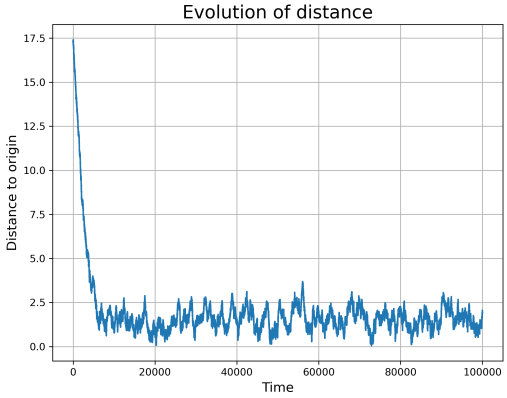
$$\mathcal{J} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sqrt{2\pi}^3}{4\pi}r_i^2\sin(\theta_i)(\cos(\theta_i)\sin(\phi_i)+\sin(\theta_i)\sin(\phi_i)+\cos(\phi_i))^2$$

If we do this then we find the value $\mathcal{J} = 31.33101103722458$. I suppose I have forgotten a factor somewhere but I have no idea where.

Equilibration time Now we plot the distance against time. Both plots are shown on Figure 12. We estimate the equilibration number to be $N_{eq} \simeq 160$ for $h = 1$ and $N_{eq} \simeq 5000$ for $h = 0.1$.



(a) Equilibration for $h = 1$ and initial starting point in $(10, 10, 10)$. $N_{eq} \simeq 160$.



(b) Equilibration for $h = 1$ and initial starting point in $(10, 10, 10)$. $N_{eq} \simeq 160$.

Figure 12: .

I think that the gaussian algorithm will perform better: to converge with the metropolis algorithm on the actual solution, the system first needs to equilibrate before we can use the smapled numbers ar really independent. Afterwards, we will need to consider the correlation since we are only taken small steps so that means subsequent samples are heavily correlated.

4.5 Stochastic Matrices

In this exercise we prove some important properties of stochastic matrices. These will be important later on.

Theorem 0.0.1. *For any non-negative initial vector w, the stochastic matrix generates non-negative vectors at all later times.*

Proof. We prove this by induction. First, let us take a vector w which is non-negative: $\forall i \in \{1, ..., N\} : w_i \geq 0$. This forms the basis for the induction.

Now we assume $w(t)$ is non-negative. Under the stochastic matrix this evolves to $w(t + \Delta t) = Pw(t)$. In terms of components this becomes

$$\forall i \in \{1, ..., N\} : w_i(t + \Delta t) = \sum_{j=1}^{N} P_{ij} w_j(t)$$

In this sum each term consists of the product of $P_{ij} \geq 0$ and $w_j(t) \geq 0$, so the sum contains only non-negative terms. We conclude that $w(t + \Delta t)$ is a stochastic matrix as well. By the principle of induction this statement is valid at all later times if all evolution is determined by the stochastic matrix. □

Theorem 0.0.2. *The product of two stochastic matrices is again a stochastic matrix.*

Proof. Suppose we have two stochastic matrices P and Q. We are concerned with the product of both, PQ. We first show that all elements of this product matrix are non-negative as well.

$$(PQ_{ij} = \sum_{k=1}^{N} P_{ik} Q_{kj} \tag{19}$$

Since P and Q are stochastic matrices we have that all their elements are non-negative, so

$$\forall i, j, k \in \{1, ..., N\} : P_{ik} Q_{kj} \geq 0 \tag{20}$$

and so the sum $(PQ_{ij} = \sum_{k=1}^{N} P_{ik} Q_{kj} \geq 0$. So all elements of the product matrix are non-negative as well.

Now we prove that all columns sum to one. Pick a random $j \in \{1, ..., N\}$. We show that the j-th column sums to 1.

$$\begin{aligned} \sum_{i=1}^{N} (PQ)_{ij} &= \sum_{i=1}^{N} \sum_{j=1}^{N} P_{ik} Q_{kj} \\ &= \sum_{k=1}^{N} Q_{kj} \left(\sum_{i=1}^{N} P_{ik} \right) \\ &= \sum_{k=1}^{N} Q_{kj} = 1 \end{aligned} \tag{21}$$

where in the last two steps we used that the columns of P, Q sum to one since these are stochastic matrices. We conclude that the product is a stochastic matrix as well. □

Theorem 0.0.3. *Introduce the norm $||w||_1 = \sum_j |w_j|$. Consider an arbitrary $N-$dimensional vector, whose entries can be negative as well. Then we have the following inequality:*

$$||Py||_1 \leq ||y||_1 \tag{22}$$

. Then, if λ is an eigenvalue of P, then $|\lambda| \leq 1$

Proof. Take an arbitrary N-dimensional vector y. Under the 1-norm we have

$$\begin{aligned} ||Py||_1 &= \sum_{j=1}^{N} |(Py)_j| \\ &= \sum_{j=1}^{N} \left| \sum_{i=1}^{N} P_{ji} y_i \right| \\ &\leq \sum_{j=1}^{N} \sum_{i=1}^{N} |P_{ji} y_i| \\ &= \sum_{j=1}^{N} \sum_{i=1}^{N} P_{ji} |y_i| \\ &= \sum_{i=1}^{N} \left(\sum_{j=1}^{N} P_{ji} \right) |y_i| \\ &= \sum_{i=1}^{N} |y_i| = ||y||_1 \end{aligned} \tag{23}$$

Here the inequality comes from the triangle inequality So the first part of the theorem holds. Now suppose λ is an eigenvalue of P with eigenvector y. Then we have $Py = \lambda y$. So the inequality tells us that

$$||y||_1 \geq ||Py||_1 = ||\lambda y||_1 = |\lambda| \times ||y||_1 \tag{24}$$

So we can conclude that in this case $|\lambda| \leq 1$ must hold in order not to violate the inequality. □

Theorem 0.0.4. *A stochastic matrix has at least one eigenvalue* $\lambda = 1$.

Proof. We construct the following left eigenvector:

$$y = (1, 1, 1, ..., 1) \tag{25}$$

When taking the product with P we get

$$yP = (1, ..., 1) \begin{pmatrix} P_{11} & ... & P_{1N} \\ ... & & ... \\ P_{N1} & ... & P_{NN} \end{pmatrix} = (\sum_{i=1}^{N} P_{i1}, \sum_{i=1}^{N} P_{i2} ..., \sum_{i=1}^{N} P_{iN}) \tag{26}$$

Since P is a stochastic matrix, the sum of all entries in one column equals one. So we have

$$\forall j \in \{1, ..., N\} : \sum_{i=1}^{N} P_{ij} = 1 \tag{27}$$

So the resulting vector in Equation (26) equals

$$yP = (1, ..., 1) = y \tag{28}$$

And so $y = (1, ..., 1)$ is a left eigenvector with eigenvalue $\lambda = 1$ for any stochastic matrix. □

4.6 Detailed Balance

In this exercise we consider a system with three distinct energy levels $E1 < E_2 < E_3$. We are given the following transition probabilities:

- $P(1 \to 2) = \alpha > 0$

- $P(2 \to 3) = \delta > 0$
- $P(3 \to 1) = \gamma > 0$

We also have the option to stay in the same state under a time step. We construct the stochastic matrix as follows:

$$P = \begin{pmatrix} 1-\alpha & 0 & \gamma \\ \alpha & 1-\delta & 0 \\ 0 & \delta & 1-\gamma \end{pmatrix} \tag{29}$$

This matrix obeys the criteria to be a stochastic matrix.

- All columns sum to 1.
- All entries are non-negative.

Detailed Balance We show that this stochastic matrix does not satisfy detailed balance. Suppose we have a stationary vector $\omega = (\omega_1, \omega_2, \omega_3)$ In the case of detailed balance, this vector would have to satisfy the condition

$$\begin{aligned} \omega_\nu P(\nu \to \mu) &= \omega_\mu P(\mu \to \nu) \\ \frac{P(\nu \to \mu)}{P(\mu \to \nu)} &= \frac{\omega_\mu}{\omega_\nu} \end{aligned} \tag{30}$$

where we choose $\mu, \nu \in \{1, 2, 3\}$. Now we have for $\mu \neq \nu$:

$$P(\mu \to \nu) = 0 \quad \text{or} \quad P(\nu \to \mu) = 0 \tag{31}$$

So detailed balance is not satisfied, unless we have $\omega = (0, 0, 0)$ which is not a stochastic vector.

Ergodicity We have non-zero probabilities to go from state $1 \to 2$, $2 \to 3$ and $3 \to 1$. That means that all states are accesible from any other state; there are no disjunct sets of states which are unreachable from some specific states. Therefore the system is ergodic.

Boltzmann distribution

We can however show that there is still a possibility to obtain the Boltzmann distribution for the equilibrium of the dynamics. We can then conclude that detailed balance is not a necessary condition to have this equilibrium distribution.

The Boltzmann distribution is defined as

$$p_i = \frac{e^{-\beta E_i}}{Z} \quad \text{where} \quad Z = \sum_{i=1}^{3} e^{-\beta E_i} \tag{32}$$

For a stochastic vector ω to describe the equilibrium distribution means that the condition

$$\omega_j = \sum_{i=1}^{3} \omega_i P(i \to j) \tag{33}$$

has to hold. (This is the global balance condition, a less strict requirement.) For the Boltzmann distribution we then get the following set of equations:

$$\begin{cases} e^{-\beta E_1} &= (1-\alpha)e^{-\beta E_1} + \gamma e^{-\beta E_3} \\ e^{-\beta E_2} &= (1-\delta)e^{-\beta E_2} + \alpha e^{-\beta E_1} \\ e^{-\beta E_3} &= (1-\gamma)e^{-\beta E_3} + \delta e^{-\beta E_2} \end{cases} \tag{34}$$

We dropped the inverse partition function since each of the terms carries this extra factor. This set of equations can be rewritten to

$$\begin{cases} 0 &= \gamma e^{-\beta E_3} - \alpha e^{-\beta E_1} \\ 0 &= \alpha e^{-\beta E_1} - \delta e^{-\beta E_2} \\ 0 &= \delta e^{-\beta E_2} - \gamma e^{-\beta E_3} \end{cases} \implies \begin{cases} \gamma e^{-\beta E_3} &= \alpha e^{-\beta E_1} \\ \alpha e^{-\beta E_1} &= \delta e^{-\beta E_2} \\ \delta e^{-\beta E_2} &= \gamma e^{-\beta E_3} \end{cases} . \tag{35}$$

This tells us that the Boltzmann distribution is indeed the equilibrium distribution when we choose the rates such that

$$\begin{cases} \frac{\gamma}{\alpha} = e^{-\beta(E_1 - E_3)} \\ \frac{\alpha}{\delta} = e^{-\beta(E_2 - E_1)} \\ \frac{\delta}{\gamma} = e^{-\beta(E_3 - E_2)} \end{cases} \tag{36}$$

If the rates α, δ, γ satisfy this set of equations for some given energy differences then we will have an equilibrium state which

4.7 Ising Model: Uniform Sampling

I combine both exercise 4.7 and 4.8 here, since I wrote a general code for the Ising model. We discuss three different sampling methods to study the evolution of the spin lattice

Energies per bond We sample the Ising model for a 10×10 lattice with uniform sampling, using helical boundary conditions and calculate the energy for each sampled configuration. This energy is given by the following Ising Hamiltonian

$$E\left(\{s_i\}\right) = -J \sum_{\langle ij \rangle} s_i s_j \tag{37}$$

where the sum runs over neighbouring spins and the spins can take values ± 1.

Uniform sampling means that we generate for each step a completely random configuration of spins: for each lattice siet we choose $s_i = \pm$ with 50% probability each.

Instead of plotting the total energy for each drawn sample, we plot the energy per bond, which is a rescaling of the energy E: $e = E/2N^2$. The result for a MC chain with $N = 10.000$ is shown on Figure 13.

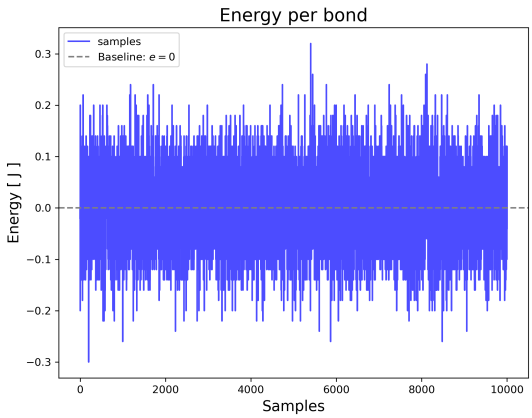


Figure 13: Energy per bond for uniform sampling. This looks like white noise, as we expect. The MC chain drew 10.000 samples uniformly for a spin lattice of 10×10.

Creating the histogram of the normalized energies shows that for uniform sampling the energies are indeed distyributed like white noise, i.e. gaussian. See Figure 14

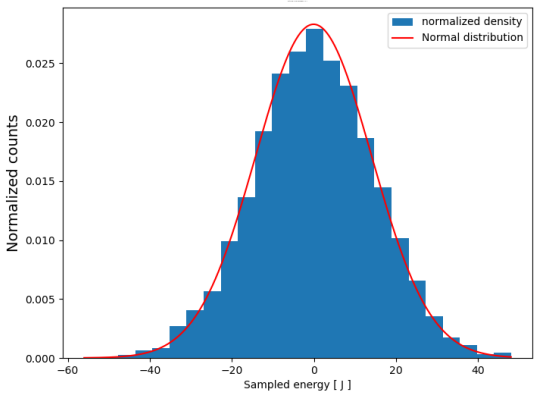


Figure 14: Normalized histogram of 10.000 spin configurations simulated using uniform sampling. We see that the histogram starts to approach the normal distribution centered around 0, with standard deviation $\sigma \sim 1/\sqrt{N}$.

Hit and Miss Method Here we use the hit and miss method to generate samples. The algorithm is as follows:

- generate a random configuration of N^2 spins.
- Calculate energy E of sample.
- Generate $r \sim \mathcal{U}(0, 1)$
- Accept new configuration if $r \leq \exp(-\beta(E - E_0))$, otherwise reject.

We sample 10.000 times using this method, using four different temperatures: $T \in \{10^5, 10^3, 10^2, 10\}$. Then we calculate the fraction of hits over the total number of attempts. This is done by the following block of code:

[4]:
```
T = [10E5, 10E3, 10E2, 100, 10]

for i in range(len(T)):
    model = IsingModel(
        10,
        J=1,
        T=T[i],
        sampling_method='hit and miss',
        boundary_condition='helical'
    )
    num_sam = 10000
    frac = model.fraction_hits(num_sam)
    print('T = ', T[i], '       Fraction of hits: ', frac)
```

```
T =  1000000.0        Fraction of hits:  0.9995
T =  10000.0        Fraction of hits:  0.9842
T =  1000.0        Fraction of hits:  0.82
T =  100        Fraction of hits:  0.1331
T =  10        Fraction of hits:  0.0
```

We conclude that for large temperatures the hit and miss method is reasonable, but the lower the temperature, the worse the al;gorithm starts to perform. This is due to the fact that for a random configuration, $E \simeq 0$. So we need $\beta 2N^2 = 2N^2/T < 1$ to have a reasonable probability for acceptance. Since we chose $N = 10$, this means $T > 100$ to have good performance.

Reweighing Technique Here we generate the histogram of total energy generated by uniform sampling $P(E)$ we generate that for a given temperature by $P_T(E) = \exp(-E/T)P(E)$.

Here we first create the histogram for the uniform sampling at a certain temperature: we used the energies obtained in Figure14.

This gives rise to the following reweighted histogram of Figure 15 for 4 different temperatures: This looks like a pretty reasonable method: for $T = 2$ we are below the critical temperature and

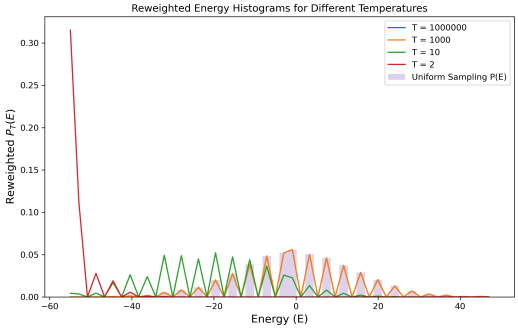


Figure 15: Histograms for some different temperatures. We started from 10.000 uniform samples that are generated at a temperature $T = 4$, with helical boundary conditions for a 10×10 lattice.

for $T < T_c$ we expect to find the model in one of the ferromagnetic states where all spins tend to align.

4.8 Ising Model: Metropolis Sampling

In this section we investigate the Ising model once again, but this time using the metropolis algorithm. This is expected to perform better than the uniform sampling and the heat bath method.

The metropolis algorithm is the following:

- Select 1 random spin at lattice site i.
- Calculate $\Delta E = E_{\text{new}} - E_{\text{old}}$, where E_{new} is the energy of the configuration with the spin at the selected lattice site flipped.
- Keep the spin flip if $\Delta E \leq 0$ or if $r \leq \exp(-\beta \Delta E)$ where $r \sim \mathcal{U}(0, 1)$.
- Repeat N times.

Magnetization We create a 50×50 lattice with helical boundary conditions. The magnetization per spin is defined as

$$m = \frac{1}{N^2} \left| \sum_{i=0}^{N^2} s(i) \right| \tag{38}$$

We can sample different MC chains using the metropolis algorithm and then plot the magnetization for each sampled configuration. We take $T = 2$ and let 4 chains run for 1000 sweeps, so we create a figure similar to Figure 3 of the lecture notes.

On this figure also the exact magnetization is plotted as the gray, dashed line. The value can be calculated using the transfer matrix method as shown by Onsager and the formula for th 2D Ising model is

$$m(T) = \left[1 - \sinh^{-4}\left(\frac{2J}{T}\right)\right]^{1/8} \tag{39}$$

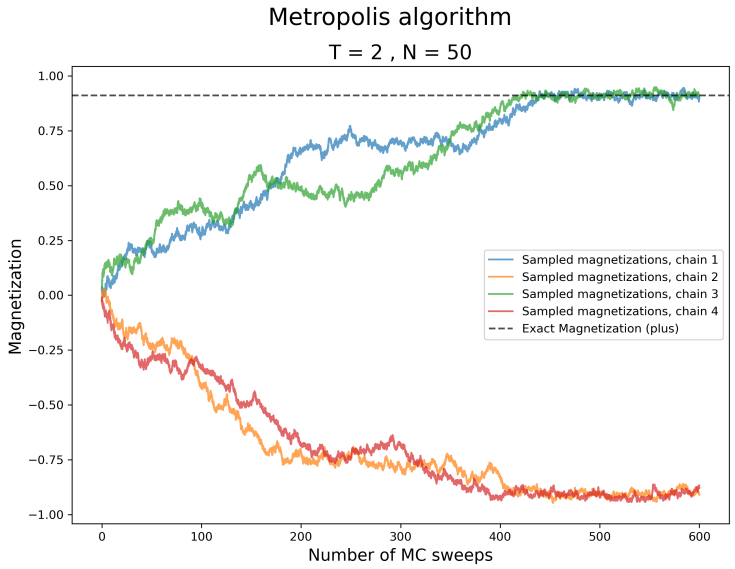


Figure 16: Magnetization per spin for the 2D Ising model simulated using metropolis sampling. We sampled each chain for 1000 sweeps, at a temperature of $T = 2$ for a 50×50 lattice with helical boundary conditions.

This means we create $4 \times 1000 \times 50^2 = 10$ million different spin configurations. In order to speed up the computation we precomputed the Boltzmann factors and updated the same lattice each time, to not run into memory problems.

From this plot we can estimate the equilibration time to be about $\tau_{eq} \simeq 200$ sweeps

Energy Histogram We run the metropolis algorithm at $T = 2$, and throw away all sampled energies before τ_{eq}. That way we can compare the histogram with that obtained using the reweighting technique. On Figure 17 the histograms are compared. We see that the reweighting technique is not too bad, we get quite close to the metropolis sampling energies.

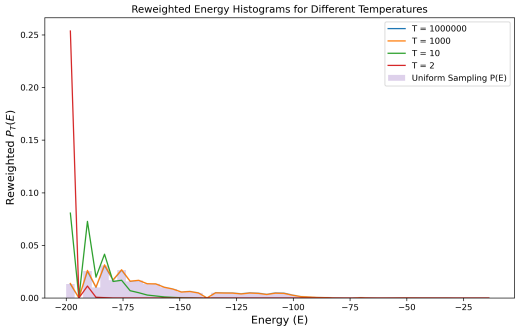


Figure 17: Comparison of the reweighting technique with the metropolis sampling method. We each time sampled 100.000 different energies and then plotted them all on the same histogram

Autocorrelation Function We can compute the autocorrelation function which is defined as follows:

$$\chi(t) = \frac{1}{t_f} \sum_{s=\tau_{eq}}^{\tau_{eq}+t_f} [m(s) - \langle m \rangle][m(s+t) - \langle m \rangle] \tag{40}$$

Here we again use a $n \times N$ lattice and use the equilibration time $\tau_{eq} =$, which we determined using Figure 16. For $T = 2.2$ this results in an autocorrelation function that is plotted on Figure 18. The calculated function is normalized with

$$\chi(0) = \frac{1}{t_f} \sum_{s=\tau_{eq}}^{\tau_{eq}+t_f} [m(s) - \langle m \rangle]^2 \tag{41}$$

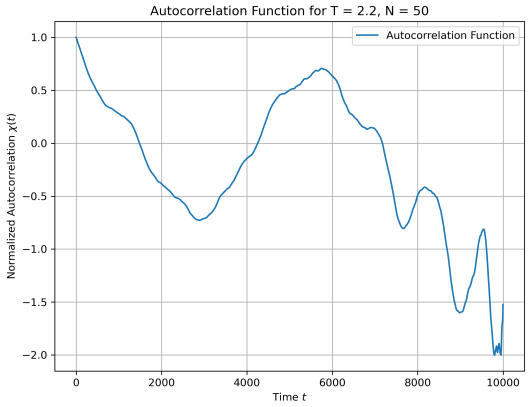


Figure 18: The autocorrelation function calculated for a 50×50 lattice with helical boundary conditions at $T = 2.2$. We normalized the function by dividing over $\chi(0)$

Lastly, we are asked to compute the correlation time τ. We do this by fitting to the data with the following function:

$$\chi(t) \sim e^{-t/\tau} \tag{42}$$

where τ denotes the leading order correlation time. This characterizes the time it takes to uncorrelate different samples. Since in each step of the algorithm we at most flip one spin, two subsequent samples will be heavily correlated. The leading behaviour of the autocorrelation

function is modulated by this decaying exponential characterized by a time τ. We do this fitting and obtain a value

$$\tau \simeq 751.54 \tag{43}$$

The fit is shown on Figure 19.

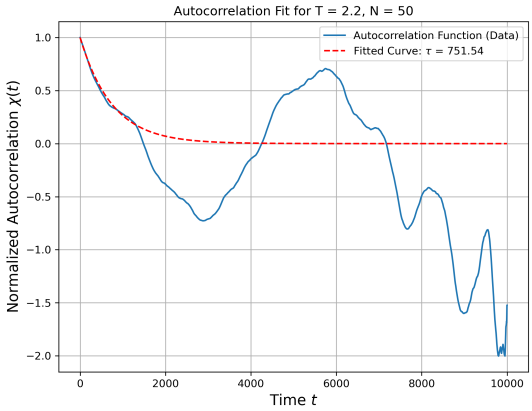


Figure 19: The correlation function plotted with the fit for the correlation time. We see that the leading behaviour in the beginning is indeed governed by this decaying exponential behaviour.

4.9 The q-dimensional Potts Model: Heat Bath

This exercise covers a generalization of the Ising model. In the code two sampling methods are implemented, the heat bath algorithm and a more general version of the Metropolis algorithm.

Detailed Balance and Ergodicity

First we show that the defined selection probability and transition probability obeys the requirements for detailed balance and ergodicity.

The transition probabilities for the heat bath algorithm are given as

$$P(\mu \to \nu) = g(\mu \to \nu)A(\mu \to \nu) \tag{44}$$

where $g(\mu \to \nu)$ denotes the probability to select state ν when we are in state μ. The selection probability is defined as

$$g(\mu \to \nu) = \frac{e^{-\beta E_n}}{\sum_{m=0}^{q-1} e^{-\beta E_m}} \tag{45}$$

Here I have left out the $\frac{1}{qM}$ factor since I think this results in a total selection probability that does not sums to 1.

First we show that the system is ergodic. Suppose we have an initial configuration μ and a final configuration ν. We can then list all lattice sites for which the spin value in μ and ν is different. Choose any of these lattice sites. In each step we choose the lattice site we want to flip uniformly so all have non-zero probability to be picked. If we have picked on of these lattice sites, we select any of the spin values within $q \in \{0, ..., q-1\}$ with probability $g(\mu \to \mu')$, which is non-zero for all choices of q. That means we can always pick select the spin value for ν for any lattice location, including all the locations where μ and ν differ. So the system must be

ergodic, since any state is reachable from any state. There exists no set of states that cannot evolve towards another set of states.

Now we turn to detailed balance. Consider μ and ν two states that differ in just one lattice site. We see that

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{\frac{e^{-\beta E_\nu]}}{\sum_{m=0}^{q-1} e^{-\beta E_m}}}{\frac{e^{-\beta E_\mu]}}{\sum_{m=0}^{q-1} e^{-\beta E_m}}} = \frac{e^{-\beta E_\nu}}{e^{-\beta E_\mu}} = e^{-\beta(E_\nu - E_\mu)} \tag{46}$$

So with equation (27) from the lecture notes in mind, we can conclude that if the dynamics obey the Boltzmann distribution then detailed balance

$$\omega_\mu P(\mu \to \nu) = \omega_\nu P(\nu \to \mu) \tag{47}$$

holds for the q-dimensional Potts model.

Equilibration time for heat bath and Metropolis In the second part of this exercise we estimate the equilibration time for the het bath algorithm and the metropolis algorithm. We fix the temperature to $T = 0.5$, the lattice size to $L = 10$ and $q = 10$. Then we start from a randomly sampled configuration that shows the energy per sampling step. The MC sampling is run for both the heat bath model as well as the metropolis model. The resulting behaviour is shown on Figure 20.

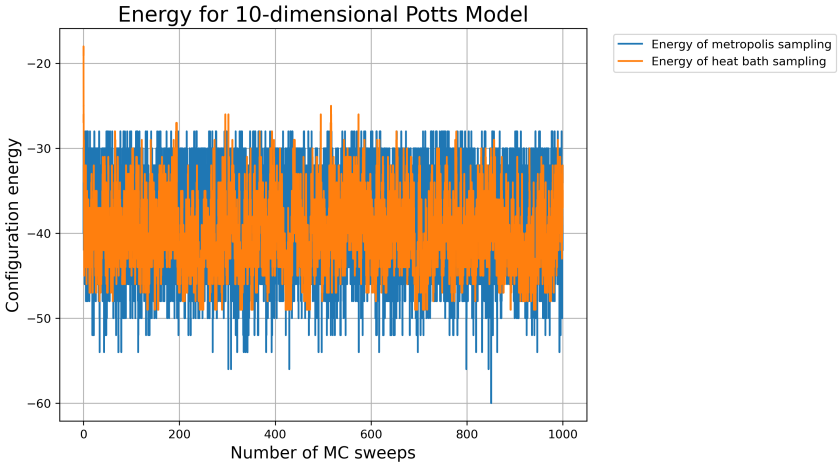


Figure 20: This is the plot I get when calculating the sampled energies for a MC chain using both metropolis and heat bath method. The parameters for this simulation are $q = 10$, $L = 10$ and $T - 0.5$.

On this figure we see that the jtwo algorithms balance around the same mean. I chose a smaller grid to reduce the computational cost. I also used the method of precomputing the Boltzmann factors to speed up the algorithm.

We find that for simulating 100000 samples using each method, the heat bath method performs slightly better than the metropolis method.

- Metropolis: $T = 39.1053900718689$
- heat Bath: $T = 35.418583154678345$

4.12 Hard Disks

Packing density

The packing fraction of a fluid is defined as

$$\eta = \frac{N\pi R^2}{L^2} \tag{48}$$

which equals the total surface of all hard disks in the system divided by the total area of the of the system. The maximal packing fraction for hard disks can be calculated analytically.

For this we assume the maximal packing density will be achieved when arranging all hard disks in a hexagonal manner. The idea is shown in Figure 21.

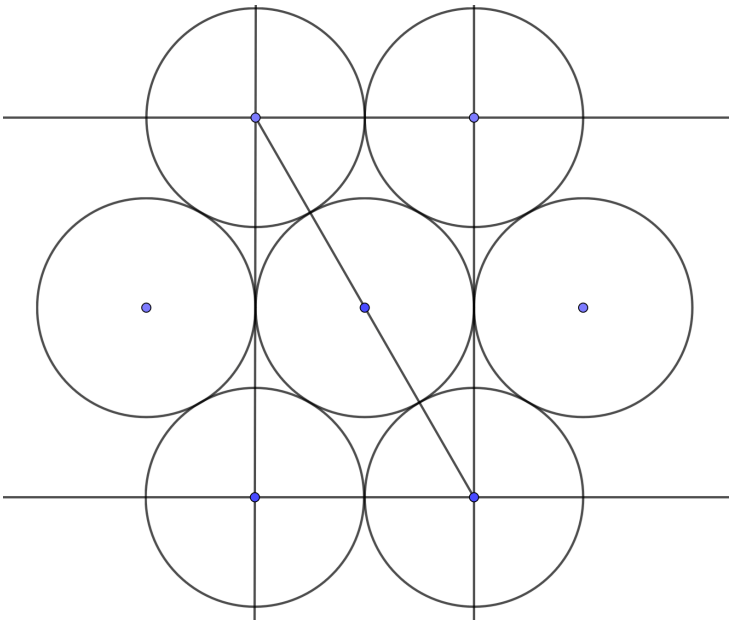


Figure 21: Hexagonal packing of hard disks in 2D. We compute the packing fraction by selecting a unit cell, a repeating pattern that will fill the whole surface of the system. This unit cell is the rectangular box as shown on the figure.

Now we can calculate the packing frequency for this configuration of the hard disks.

$$A_{\text{Disks}} = 2 \times (\pi\sigma^2) \tag{49}$$

The total surface of the unit cell is given as

$$A_{\text{Unit cell}} = b \times h = 2\sigma * \sqrt{(4\sigma)^2 - (2\sigma)^2} = 2\sqrt{12}\sigma^2 \tag{50}$$

So the packing density is in this case the given by

$$\eta = \frac{A_{\text{Disks}}}{A_{\text{Unit cell}}} = \frac{2\pi\sigma^2}{2\sqrt{12}\sigma^2} = \frac{\pi}{\sqrt{12}} \simeq 0.9069 \tag{51}$$

Average Acceptance Ratio Now in this part we will try to get a grip on the acceptance rate of the metropolis algorithm as a function of the packing fraction η and the characteristic length q. We do this by creating an array with different $N-$values, which we initialize in a rectangular grid. we also create an array with different $q-$values, which denotes the characteristic length. for all combinations the metropolis algorithm is nran for 1000 steps and we count all accepted metropolis moves. We divide by the 1000 proposed moves and this gives the acceptance rate for the run. This is plotted as a density on the grid, which is colorcoded. The result is shown on figure 22.

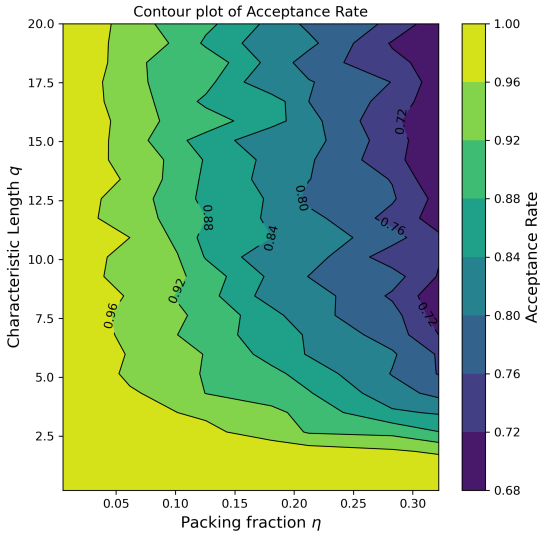


Figure 22: The colormap of the acceptance rate for a Monte Carlo runn for each of the combinations. We see that for low q, almost all moves are accpeted.

Pair Correlation Function In this last part of the exercise we implement the pair correlation function. This function is defined as

$$g(r) = \frac{L^2}{2\pi r N} \langle \sum_i \sum_{j \neq i} \delta(r - r_{ij}) \rangle \tag{52}$$

This function denotes the probability of finding a particle at distance r, given that there is a particle at the origin. For a fluid at low density, this well be a near constant function. For fluids with higher density the behaviour can be non-trivial, revealing information about the local structure and arrangement.

We can compute it by building up the histogram of distances for any two pairs of particles as

$$g(r) \simeq \frac{\langle N(r, \Delta r) \rangle}{\rho 2\pi r \Delta r}. \tag{53}$$

Here Δr is some discrete distance and $\langle N(r, \Delta r) \rangle$ denotes the average number of pairs with distnace between $[t, t+\Delta t]$. $\rho = N/L^2$ denotes the particle density. Plotting the behaviour of this function for a series of packings $\eta = [4, 9, 16, 36, 49, 64, 81]$ gives the result shown on Figure 23. For the other parameters we initialized the grid rectangularly with $L = 100, \sigma = 4, q = 2.5$.

4.13 Monte Carlo in Continuous Time

We consider a 1D lattice $x \in \{..., -2, -1, 0, 1, 2, ...\}$ with one particle whose dynamics can be described as a random walk. At each time step the particle either jumps to one of the neighbouring positions with probability $P(x \to x \pm 1) = \alpha$ or stays on the same lattice site with probability $P(x \to x) = 1 - 2\alpha$. If α is small the probability to jump becomes small and the particle will spend most of the time in a certain position.

We consider therefore a time T_γ, which describes the average time a particle stays in the same

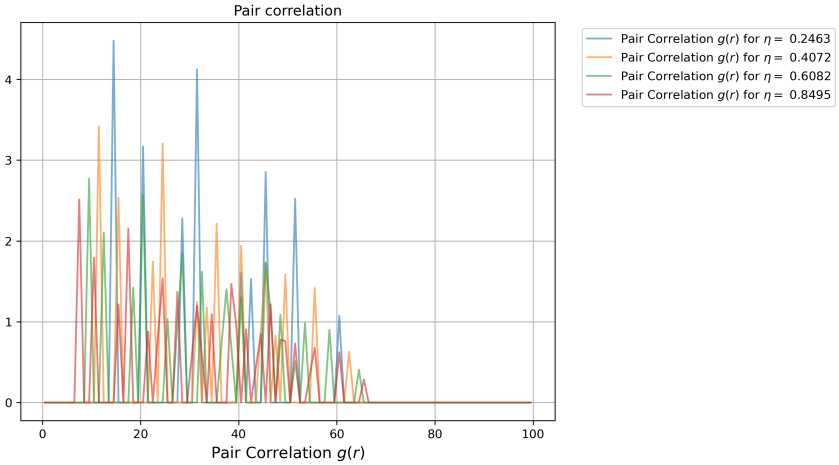


Figure 23: Behaviour of different packing fractions for the pair correlation function $g(r)$. We see that there arise peaks for larger packings, indicating that the distance between pairs obtains some structure due to the larger density of particles in the fluid.

lattice site without jumping. We can estimate $\Delta t'$ analytically as follows.

$$\begin{aligned}\langle \Delta t' \rangle &= \Delta t \sum_{n=0}^{+\infty} n(1-2\alpha)^n(2\alpha) \\ &= \Delta t \sum_{n=0}^{+\infty} nP(x \to x)^n P(x \to x \pm 1) \\ &= \frac{\Delta t}{2\alpha}\end{aligned} \tag{54}$$

where we made use of the geometric distribution: this distribution described to probability for a Bernoulli experiment to yield n times a failure (= no jump) before resulting in a success (= a jump). For this distribution the mean is give by $1/p$, where p denotes the probability to have a success (= a jump). So we have $\langle \Delta t' \rangle = 1/(2\alpha)$. For $\alpha = 0.01$ this gives $\langle \Delta t' \rangle = 1/(2\alpha) = 50$. Here we have set $\Delta t = 1$ which can always be done by rescaling our time units.

For the continuous time algorithm, we select with equal probability a jump to either left or right. The time step is updated with the analytically calculated $\langle \Delta t' \rangle$. Here we move the particle each time step, which is updated with $t \to t + \langle \Delta t' \rangle$ each time.

We can compare the two algorithms using the following block of code, where we time the runtime for each algorithm. We see that the discrete time-algorithm takes much longer to reach $x = 20$ than the continuous time algorithm, by a large factor.

[5]:
```
import time

alpha = 0.01
target_x = 20
num_simulations = 100

# Measure runtime for discrete-time
start = time.time()
discrete_time_avg = discrete_time_simulation_numpy(alpha, target_x,␣
 ↪num_simulations)
end = time.time()
```

```
print(f"Discrete-Time Average: {discrete_time_avg}, Runtime: {end - start:.
  ↪2f}s")

# Measure runtime for continuous-time
start = time.time()
continuous_time_avg = continuous_time_simulation_numpy(alpha, target_x,␣
  ↪num_simulations)
end = time.time()
print(f"Continuous-Time Average: {continuous_time_avg}, Runtime: {end -␣
  ↪start:.2f}s")
```

```
Discrete-Time Average: 2554604.69, Runtime: 141.51s
Continuous-Time Average: 878279.0, Runtime: 0.85s
```

Theoretically the continuous time algorithm should be be about 50 times as fast since it takes steps in time that are a factor 50 larger. When running both algorithms a 100 times and computing the average time it takes to reach $T = 10^6 \Delta t$, we find a speedup of a factor 25:

$$\frac{\text{Average time discrete time method}}{\text{Average time continuous time method}} = \frac{2.84576416015625e-05}{1.1396408081054688e-06} \simeq 24.94 \tag{55}$$

4.14 Birth-Annihilation Process

Exercise 4.14, 4.15, 4.16 are all solved by a general code I wrote for solving reactions with the Gillespie algorithm.

The results are plotted here.

Setting of the Problem

We have a single particle type, that obeys the following dynamics:

$$\begin{aligned} A &\xrightarrow{\lambda} \emptyset \\ A &\xrightarrow{\mu} AA \end{aligned} \tag{56}$$

Deterministic Rate Equations We can write down deterministic differential equations that describe the evolution of the average number of particles. We have only 1 type of particle in the system, A. Then we can construct the rate equation for A as follows:

$$\frac{d\langle A\rangle}{dt} = -\lambda\langle A\rangle + \mu\langle A\rangle \tag{57}$$

This is a very simple differential equation which can be solved as

$$\begin{aligned} \frac{d\langle A\rangle}{\langle A\rangle} &= (\mu-\lambda)dt \\ \langle A\rangle &= A(t=0)e^{(\mu-\lambda)t} \end{aligned} \tag{58}$$

This describes the evolution of the average number of particles. We can check whether this is correct by evolving the system under the Gillespie algorithm.

We start from $N = 100$ $A-$particles and evolve with $\lambda = 1, \mu = 0.2$. The result is shown in Figure **??**

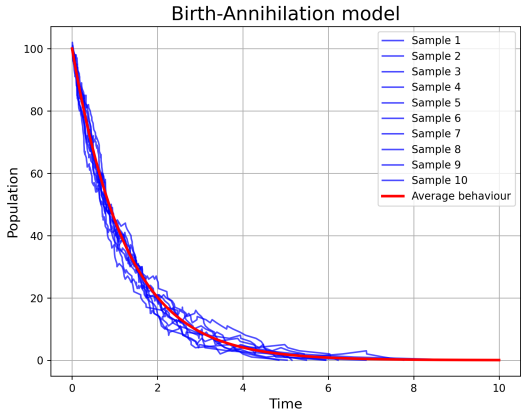


Figure 24: Evolution of the Birth-Annihilation model for $\lambda = 1, \mu = 0.2$ and $N(0) = 100$. We sampled 10 different chains and see that the rate equation prediction matches the sampled behaviour.

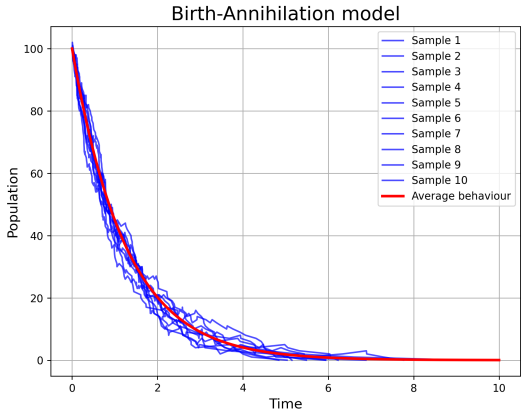


Figure 25: Evolution of the Birth-Annihilation model for $\lambda = 1, \mu = 0.2$ and $N(0) = 10.000$. We sampled 10 different chains and see that the rate equation prediction matches the sampled behaviour.

We can do the same but for 10.000 initial A-particles. This gives the result shown in Figure 25

We are also asked to plot the histogram of the annihilation time for each simulation. This is the time at which all particles have disappeared from the system. From the rate equation solution we see that this will eventually happen since $\mu - \lambda = -0.8$, so the exponential is decaying in time. The histogram is traced for a 1000 chains, and is shown in Figure 26.

Lotka-Volterra Model

In this problem we consider the Lotka-Volterra model. This is a system of reaction equations:

$$\begin{aligned} Y1 &\xrightarrow{k_1} 2Y_1 \\ Y_1 + Y_2 &\xrightarrow{k_2} 2Y_2 \\ Y_2 &\xrightarrow{k_3} \emptyset \end{aligned} \tag{59}$$

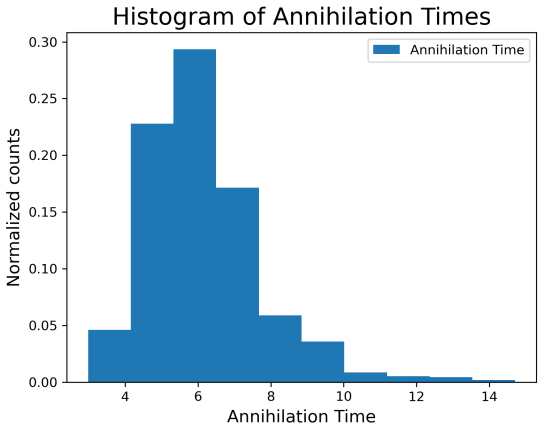


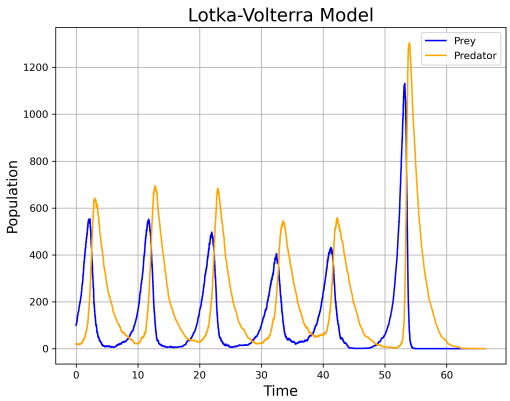
Figure 26: Histogram of annihilation times for 1000 runs. The parameters are $\lambda = 1, \mu = 0.2, N = 100$.

Here Y_1 is the prey and Y_2 is the predator. We choose the rates as

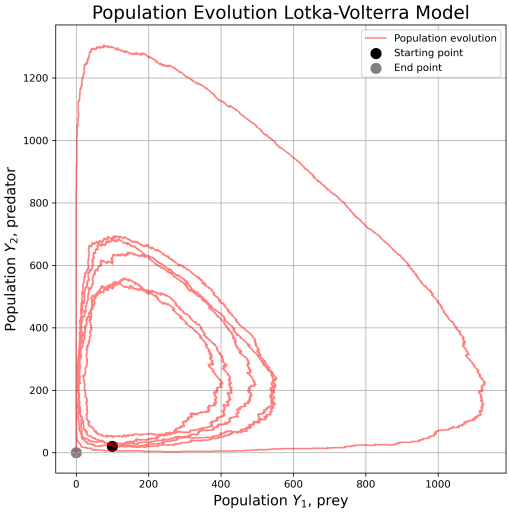
- $k_1 = 1$
- $k_2 = 0.005$
- $k_3 = 0.6$.

The initial populations are chosen as $N_1 = 100$ and $N_2 = 20$

Under the Gillespie algorithm we can evolve this system. We plot the evolution of $X_1(t)$ and $X_2(t)$ against time (see Figure 27a) as well as against each other (see Figure 27b).



(a) Evolution of the populations against time in the Lotka-Volterra model described above.



(b) Evolution of Figure 27a but now plotted against each other. For convenience, we have plotted the starting and ending points as well.

Figure 27: Comparison of Lotka-Volterra population dynamics. (a) shows the evolution of the populations against time, while (b) shows their relationship when plotted against each other.

Average number of oscillations We run the simulation 20 times and count the number of oscillations that the population levels undergo before ending up in one of the absorbing states. We count the oscillation by counting the number of times the population crosses its average value and we divide by two since in one oscillation the species number will cross the mean value two times. This results in

- Prey population: $\langle$number of oscillations, prey$\rangle = 6.45$
- Predator population: $\langle$number of oscillations, predator$\rangle = 5.85$

This is expected: in a small number of cases, the system will evolve to a situation where the prey population starts growing while the predator population has been extincted.

The majority of the times the Lotka-Volterra system evolves to the $X_1 = X_2 = 0$ state, where both populations go extinct.

4.16 Brusselator

This exercise again treats a model of a set of reaction equations. In this case we have

$$\begin{aligned} \emptyset &\xrightarrow{k_1} Y_1 \\ Y_1 &\xrightarrow{k_2} Y_2 \\ 2Y_1 + Y_2 &\xrightarrow{k_3} 3Y_1 \\ Y_1 &\xrightarrow{k_4} \emptyset \end{aligned} \tag{60}$$

As in [1], we fix the rates k_i to

- $k_1 = 5 \times 10^3$
- $k_2 = 50$
- $k_3 = 5 \times 10^{-5}$
- $k_4 = 5$.

We can compute the trajectory in the $X_1(t) - X_2(t)$ plane for initial conditions of $X_1(0) = 1000, X_2(0) = 2000$. This results in the trajectory shown on Figure 28.

Different Initial Conditions

Now we initialize the populations differently. We plot each time the behaviour that arises and compare it to the original Gillespie paper.

$X_1(0) = 1000, X_2(0) = 4000$**.** We see that the results are comparable to those found in the Gillespie paper. Both nicely show the epicycle behaviour for the Brusselator model, which is the near periodic behaviour of the population evolution.

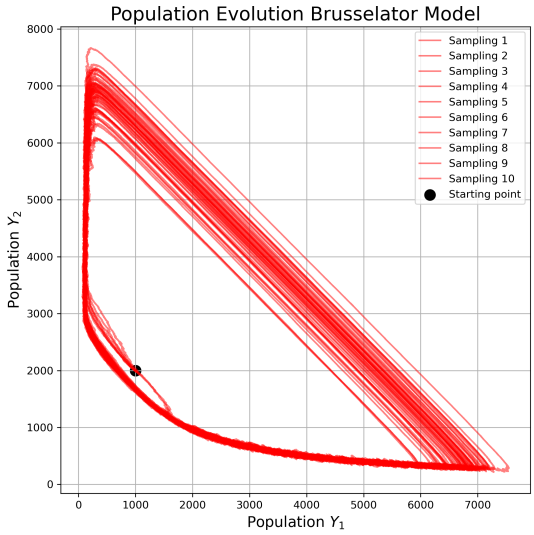
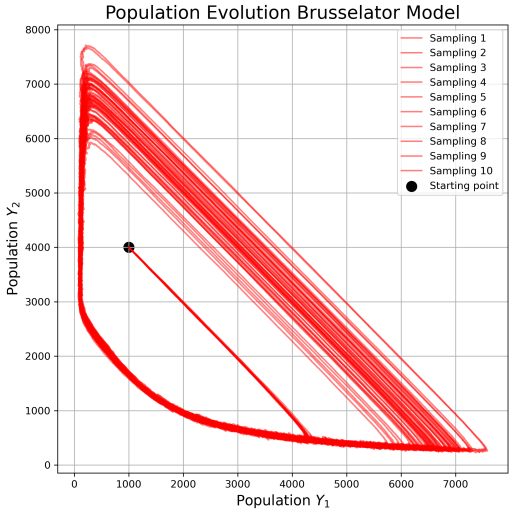
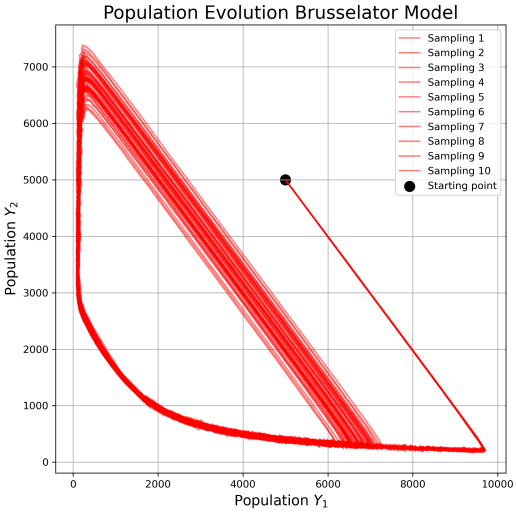


Figure 28: Trajectory in the $X_1(t) - X_2(t)$ plane. We sampled the evolution 10 times, and indicated the starting point.



(a) Trajectory plot for $X_1(t) - X_2(t)$ plane. Initial populations are $X_1(0) = 1000, X_2(0) = 4000$.



(b) Trajectory plot for $X_1(t) - X_2(t)$ plane. Initial populations are $X_1(0) = 5000, X_2(0) = 5000$.

Figure 29: Trajectory plots for $X_1(t) - X_2(t)$ plane with different initial populations. (a) shows $X_1(0) = 1000, X_2(0) = 4000$, and (b) shows $X_1(0) = 5000, X_2(0) = 5000$.

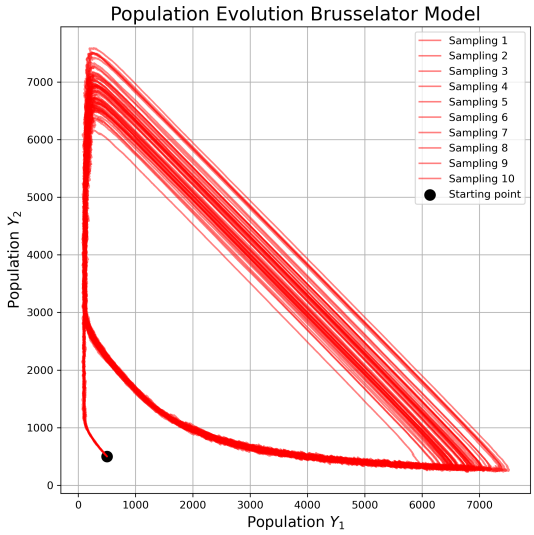


Figure 30: Trajectory plot for $X_1(t) - X_2(t)$ plane. Initial populations are $X_1(0) = 500, X_2(0) = 500$

REFERENCES

[1] *https://www.cl.cam.ac.uk/teaching//2223/Bioinfo/papers/DanielGillespie1.pdf*. URL: https://www.cl.cam.ac.uk/teaching//2223/Bioinfo/papers/DanielGillespie1.pdf (visited on 11/28/2024).