# Application of Brownian Motion in Particle Separation

TFY4235 Assignment 2, Solfrid Hagen Johansen

*Discussed with Jeanette Bonden Isachsen and Kristine Biering Mohr*

**Abstract**

A numerical solution of the overdamped Langevin equation was obtained for multiple particles in a Ratchet potential, in order to separate them based on their size. It was found that the optimum the system with $r = r_1$ was found to be $\tau_{\text{op}} = 0.78$, which gave a drift velocity of $\langle v_1 \rangle = 0.027 \mu\text{m/s}$. It was found for particles of size $r_2$ that $\tau_{\text{op, 2}} = 1.95$.

## 1. Introduction

To analyze substances of different sizes, we can use a ratchet potential to separate the particles. A ratchet potential is asymmetric, and by turning it on and of (flashing it), the Brownian motion of the particles will make it so that the particles tend to the right. The drift velocity of particles of different sizes will differ, so that we may in theory sort the constituent particles by size. Here, a numerical experiment is developed, which is based on the Langevin-approach [1].

The code is written in C++ using Armadillo [2], which is a library for linear algebra and scientific computing. The data computed is exported to Python, and visualized using matplotlib.

## 2. Theory and Numerical Method

The system consists of objects of two sizes in Brownian motion. The particles are assumed to be spherical, with radius $r_1$ and $r_2$, and mass $m_1$ and $m_2$. The equation of motion for a 1D problem, assuming the particles do not interact, from Newton's second Law [1], is

$$m_i \frac{\mathrm{d}^2 x_i}{\mathrm{d}t^2} = -\frac{\partial U}{\partial x}(x_i, t) - \gamma_i \frac{\mathrm{d}x_i}{\mathrm{d}t} + \xi(t), \tag{1}$$

where the terms have the same meaning as in [1].

This is the Langevin equation, which by neglecting the inertial term, becomes

$$\gamma_i \frac{\mathrm{d}x_i}{\mathrm{d}t} = -\frac{\partial U}{\partial x}(x_i, t) + \xi(t), \tag{2}$$

which is a stochastic differential equation.

In this experiment, the potential energy is on the form $U(x,t) = U_r(x) f(t)$, where

$$\hat{U}_{\text{r}}(x) = \begin{cases} \frac{x\Delta U}{\alpha L} & \text{if } 0 \leq x < \alpha L \\ \frac{L-x}{L(1-\alpha)}\Delta U & \text{if } \alpha L \leq x < L \end{cases} \tag{3}$$

and

$$\hat{f}(t) = \begin{cases} 0 & \text{if } 0 \leq t < \frac{3\tau}{4} \\ 1 & \text{if } \frac{3\tau}{4} \leq t < \tau \end{cases} \tag{4}$$

So solve the above equation, a forward Euler Scheme is used

$$x_{n+1} = x_n - \frac{1}{\gamma_i}\frac{\partial U}{\partial x}(x_n, t_n)\,\delta t + \sqrt{\frac{2k_B T \delta t}{\gamma_i}}\hat{\xi}_n. \tag{5}$$

The simple Euler Scheme was used due to the stochastic variables being sensitive, and here as few function evaluations as possible is used.

### *2.1. Differential equation in Reduced Units*

Dividing the equation by L, we get

$$\hat{x}_{n+1} = \hat{x}_n - \frac{1}{\gamma_i L}\frac{\partial U}{\partial x}\delta t + \sqrt{\frac{2k_B T \delta t}{\gamma_i L^2}}\hat{\xi}_n, \tag{6}$$

where $\hat{x} = x/L$. By defining

$$\hat{U}(\hat{x}, \hat{t}) = \frac{U(x,t)}{\Delta U}, \tag{7}$$

the chain rule gives

$$\frac{\partial U}{\partial x} = \Delta U \frac{\partial \hat{U}}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial x} = \Delta U \frac{\partial \hat{U}}{\partial \hat{x}}\frac{1}{L}, \tag{8}$$

which by inserting into Equation (6), gives

$$\hat{x}_{n+1} = \hat{x}_n - \frac{\Delta U}{\gamma_i L^2}\frac{\partial \hat{U}}{\partial \hat{x}}\delta t + \sqrt{\frac{2k_B T \delta t}{\gamma_i L^2}}\hat{\xi}_n. \tag{9}$$

Defining $\hat{D} = k_B T/\Delta U$, $\omega = \Delta U/(\gamma_i L^2)$ and $\hat{t} = \omega t$, so that $\delta\hat{t} = \omega \delta t$, we get

$$\hat{x}_{n+1} = \hat{x}_n - \frac{\Delta U}{\gamma_i L^2 \omega}\frac{\partial \hat{U}}{\partial \hat{x}}\delta\hat{t} + \sqrt{\frac{2k_B T \delta\hat{t}\gamma_i L^2}{\gamma_i L^2 \Delta U}}\hat{\xi}_n, \tag{10}$$

$$\hat{x}_{n+1} = \hat{x}_n - \frac{\partial \hat{U}}{\partial \hat{x}}\delta\hat{t} + \sqrt{\frac{2k_B T \delta\hat{t}}{\Delta U}}\hat{\xi}_n. \tag{11}$$

Finally the expression becomes

$$\hat{x}_{n+1} = \hat{x}_n - \frac{\partial \hat{U}}{\partial \hat{x}}\delta\hat{t} + \sqrt{2\hat{D}\hat{t}}\hat{\xi}_n. \tag{12}$$

Equation 2.8 in the Assignment sheet [1] becomes

$$\max\left|\frac{\partial \hat{U}}{\partial \hat{x}}\right|\delta\hat{t} + 4\sqrt{2\hat{D}\delta\hat{t}} \ll \alpha \tag{13}$$

in reduced units. Equations 2.5 in Assignment [1] becomes

$$\hat{U}_{\mathrm{r}}(\hat{x}) = \begin{cases} \frac{\hat{x}}{\alpha} & \text{if } 0 \leq \hat{x} < \alpha \\ \frac{1-\hat{x}}{1-\alpha} & \text{if } \alpha \leq \hat{x} < 1 \end{cases} \tag{14}$$

and

$$\hat{f}(\hat{t}) = \begin{cases} 0 & \text{if } 0 \leq \hat{t} < \frac{3\tau\Delta U}{4\gamma_i L^2} \\ 1 & \text{if } \frac{3\tau\Delta U}{4\gamma_i L^2} \leq \hat{t} < \frac{\tau\Delta U}{\gamma_i L^2} \end{cases} \tag{15}$$

In reduced units, the time-scale is

$$\hat{t} = \omega t = \frac{\Delta U}{\gamma_i L^2} t = \frac{\Delta U t}{6\pi\eta r_2 L^2}. \tag{16}$$

In addition, there is no other dependence on $r_i$, so the entire $r_i$ dependence is contained in this term. Thus, for particles of different size, changing the radius, corresponds to changing the *time scale*.

### *2.2. Numerical Implementation*

The values input as parameters are in normal units, which is then converted to reduced units in the program for the computation. Before the data is computed stored, it is converted back to normal units.

The functions (14) and (15) need to be periodic. For the function (14), a position, $x$ which is larger than 1 should be equal to $x/\alpha$ if $x < 1+\alpha$. This is resolved in the implementation by using the absolute value of the `fmod()`function, which computes the remainder of a floating point number. This is done before evaluation of the functions, so that the value evaluated always is within the interval specified in the functions. For function (14), the value over which the value of $x$ needs to be modulated is 1, while for the function (16) this value is $\tau\omega$.

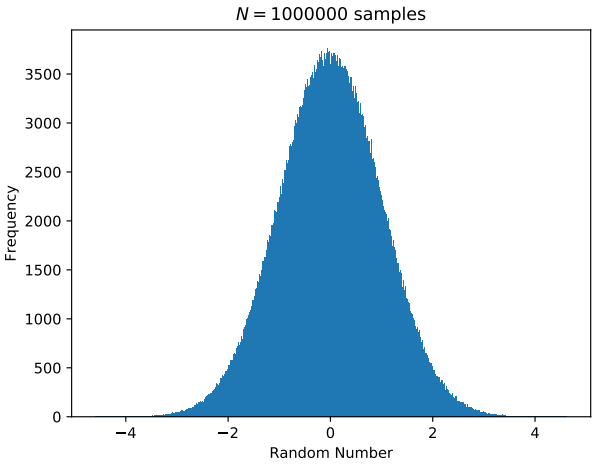


**Figure 1:** Plot of $n = 1000000$ random values generated.

The Euler Scheme is implemented by a function `eulerScheme`. The function first checks the criterion for $\delta t$, Equation (13), which is set to check if it is smaller than $0.1\alpha$.

For each time-step, as set by the user, the Forward Euler scheme is evaluated, as expressed in Equation (12). The values are stored in a vector. The previous value $x_n$ is simply obtained from the previous element in the vector. $\hat{\xi}_n$ is obtained from a random number generator `randn()`which gives Gaussian distributed random numbers.

The density of particles is computed by for each particle $N$ in the ensemble, perform computation of the Euler Method. Then in Python, a histogram for different times is plotted.

## 3. Results and Discussion

The values used for the parameters were $r = r_1 = 12\,\mathrm{nm}, L = 20\,\mu\mathrm{m}, \alpha = 0.2, \eta = 1\,\mathrm{mPa\,s}, k_B T = 26\,\mathrm{meV}$ and $\Delta U = 80\,\mathrm{eV}$, unless otherwise stated.

### *3.1. Random Value Generator*

To ensure that the random values used are in Gaussian distributed random numbers, 1000000 samples were generated using the `randn()`function in Armadillo. The values were imported into Python, and plotted in a histogram with 1000 bins.

The result is shown in Figure 1. The values were computed to have a mean of $-0.0001$ and computed in python to have a standard deviation of 0.9987. The deviations from 0 and 1 are likely due to not using an infinite amount of numbers. From this the random value generator is assumed to in fact produce Gaussian distributed numbers.

### *3.2. Without Flashing*

With the flashing turned off, the position of the particle as a function of time is plotted in Figure 2 and 3. The start

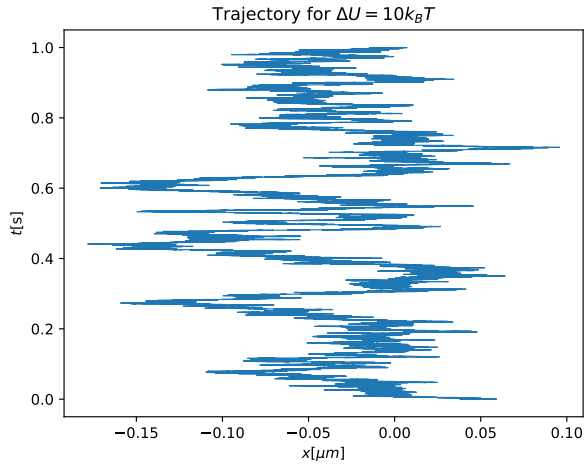


**Figure 2:** Plot of the position of one particle as a function of time, for $\Delta U = 10k_BT$.

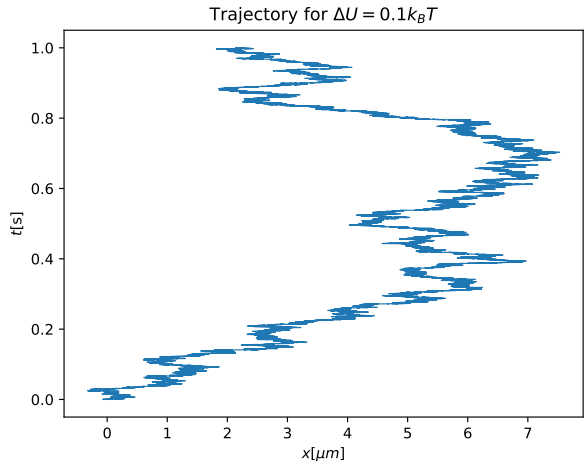


**Figure 3:** Plot of the position of one particle as a function of time, for $\Delta U = 0.1k_BT$

position was the origin, $x = 0$. Notice that the particle in the weaker potential is able to diffuse further than the one in the strong potential. This is as expected, due to the particle in the weaker potential having higher thermal energy than the potential, so that it can travel beyond the potential (which is located at $x = 0.2\mu$m). For the potential which is larger than the thermal energy, the particle is contained within the potential.

To compute the energy distribution, the position at the last time-step is stored, and the energy at that point is computed. To analyze the data, a histogram is plotted.

In Figure 4 a plot of the distribution of energy for a particle is shown, using $N = 1000$ realizations for $\Delta U = 10k_BT$ and $\Delta U = 0.1k_BT$. Observe that for small $\Delta U$ the particles are seemingly evenly distributed, as the potential is weak compared to the kinetic energy. However for larger

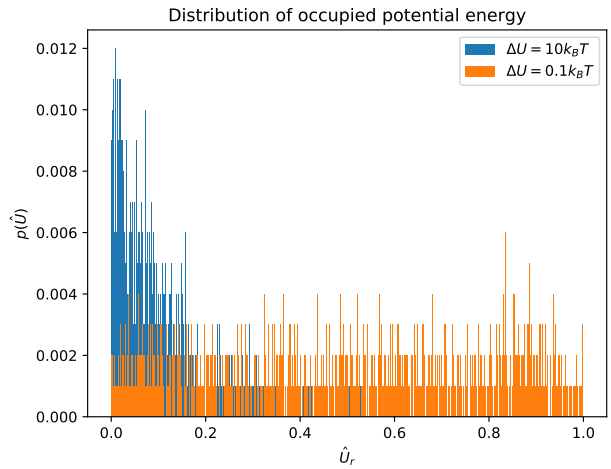


**Figure 4:** Plot of the distribution of occupied energy, for $\Delta U = 0.1k_BT$ and $\Delta U = 10k_BT$.

potentials, most particles and up in a energy minimum.

### *3.3. With Flashing*

For the particle to drift to the right, the potential must be chosen so that the probability of the particle being to the right when the potential is turned off, so that is is trapped to the right. This causes the motion to be to the right.

The particle needs to diffuse at least $\alpha L$, but not longer than $(1-\alpha)L$ while the potential is turned off, to drift to the right. As $\tau$ is the parameter which controls how long the potential is turned off, we find three different regimes for the parameter:

- Small $\tau$: The particle will not have time diffuse, and does not drift.
- Intermediate $\tau$: The particle has enough time to diffuse far enough, but not too far, so it drifts.
- Large $\tau$: The particle diffuses too far, and does not drift.

The value of $\Delta U$ needs to be chosen so that drifting is possible. As stated this happens when a particle moves a distance between $\alpha$, but not longer than $(1-\alpha)$ (reduced form). I.e. the change due to the diffusion term $\sqrt{2\hat{D}\delta\hat{t}}\hat{\xi}_n$ has to be contained within the interval

$$\alpha < \sqrt{2\hat{D}\delta\hat{t}}\hat{\xi}_n < (1-\alpha). \tag{17}$$

This may be rewritten to find a suitable interval for $\Delta U$

$$\frac{2k_BT\delta\hat{t}}{(1-\alpha)^2}\hat{\xi_n}^2 < \Delta U < \frac{2k_BT\delta\hat{t}}{\alpha^2}\hat{\xi_n}^2. \tag{18}$$

In Figure 5 the trajectory of a particle with $\tau = 0.01s$ is plotted. Observe that the particle does not have enough

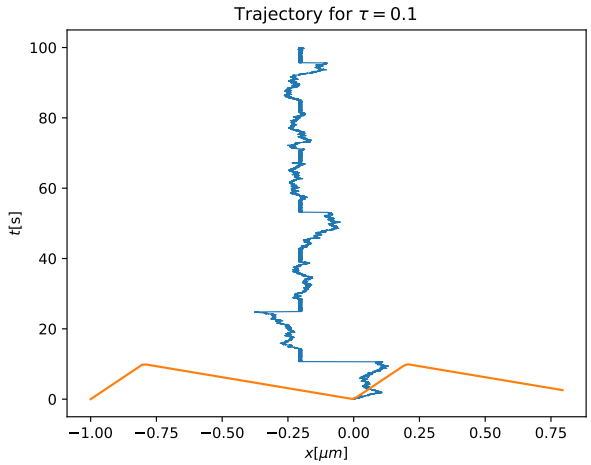


**Figure 5:** Plot of the trajectory of a particle for $\tau = 0.01s$. The line in orange represents the potential (potential not to scale on the $y$ axis).

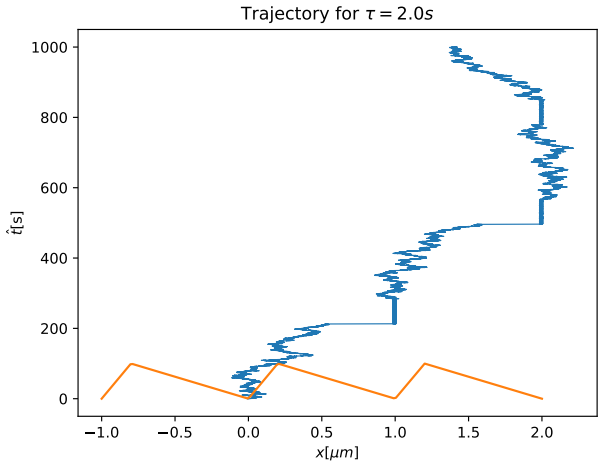


**Figure 6:** Plot of the trajectory of a particle for $\tau = 2s$. The line in orange represents the potential (potential not to scale on the $y$ axis).

time to diffuse, so that when the potential is turned on, it falls back into its original position. In this regime the particle will not drift.

In Figure 6 the trajectory of a particle with $\tau = 2s$ is plotted. Here the particle has enough time to diffuse to the right so that it is trapped in a potential to the right when the potential is turned on. This allows for the particle to drift

The statement that the particle diffuses to far for large $\tau$ can be observed in Figure 7. Here the particle diffuses to the left more than to the right, so that the particle does not drift to the right.

To find the average drift velocity as a function of $\tau$ for the $r_1$ particle, the Euler Scheme was ran 100 times for each

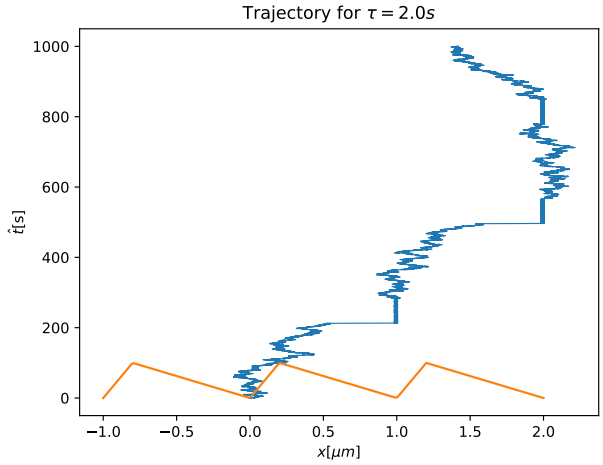


**Figure 7:** Plot of the trajectory of a particle for $\tau = 7s$. The line in orange represents the potential (potential not to scale on the $y$ axis).

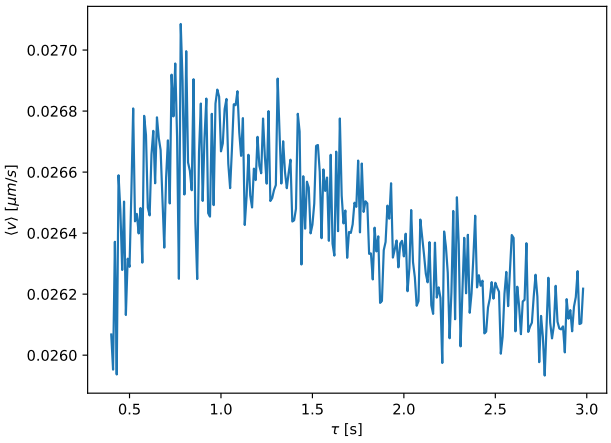


**Figure 8:** Plot of the average drift velocity, $\langle v \rangle$ of a particle with radius $r_1$.

value of $\tau$, ranging from $\tau = 0.4$ to $\tau = 3$ with steps of $\delta\tau = 0.01$. The reason for running the system multiple times, is the dependency on the stochastic variable. For each $\tau$, the average end-position was found, and divided by the (reduced) end-time.

The plot in Figure 8 shows the computed average drift velocity $\langle v_1 \rangle$. From the Figure, one can observe that the highest value of the drift velocity (optimum flashing time) happens for $\tau_{op} = 0.78s$ which gives $\langle v_1 \rangle = 0.027\,\mu$m/s. To general shape is similar to that of Figure 2 in [3]. In [4] Figure 3, an experiment done shows movement of around $4 \cdot 2\mu$m over 10 cycles (moved over two electrodes and two gaps), where the frequency is 0.7 Hz. I.e. the DNA particles has over a time $1/0.7 \cdot 10 = 14.3$s moved around $8\nu$m. This gives a drift velocity of about $v = 0.56\mu$m/s. The value deviates significantly from the value obtained numerically.

As discussed, changing the radius corresponds to changing the timescale. Thus, the plot found for the average drift velocity, Figure 8 can be used to deduce the velocity for a particle of radius $r_2 = 3r_1$. From Equation (16) we see that the time for a particle of a larger $r$ is smaller than for a small $r$. Thus, the time scales are smaller, and the time needed for the particle to diffuse is larger. We get $\delta\hat{t}_2 = 1/3\delta\hat{t}_1$, so that the value of the drift velocity for $r_1$ will be three times as large as for $r_2$. This is due to the $r_2$ particle being able to travel three times as far in the same amount of time. In addition, this leads to $\tau_2 = 3\tau_1$, so that the optimum value of $\tau$ is three times as large for particle with $r_2$.

To find the average drift velocity of a particle with $r_2 = 3r_1$, the drift velocity obtained for $r_1$ is divided by 3. The estimated average drift velocity of the particle when $\tau = \tau_{\text{op}}$ is then $\langle v_2 \rangle = 3 \cdot 0.027\mu\text{m/s} = 0.081\mu\text{m/s}$. The optimum drift velocity is estimated to be $\tau_{\text{op, 2}} = 3\tau_{\text{op}} = 3 \cdot 0.78\text{s} = 2.34\text{s}$.

The average drift velocity for $r = r_2$ was explicitly computed, which is shown in Figure 9. From the Figure the estimated optimum value of $\tau$ is $\tau_{op,2} = 1.95$, with $\langle v_2 \rangle = 0.387$. The value found for the drift velocity deviates significantly from the expected value

*3.3.1. Density of ensemble particles*

First looking at an ensemble of $N = 1000$ particles of type 1, i.e. with radius $r = r_1$. The particles are used to simulate how the particle density changes with time.

First the situation with $\Delta U = 0$, i.e. no potential, is studies. The density, $n(x)$ as a function of position,$x$, was plotted for different values of time, $t$, see Figures **??**, 10, 11 and13. The particles start off close to the origin, but as time passes, they spread out, as expected due to Brownian Motion.

For $r = r_2$ the distribution at $t = 0.84998$. Notice that although the radius of the particle, $r$, determines the trajec-

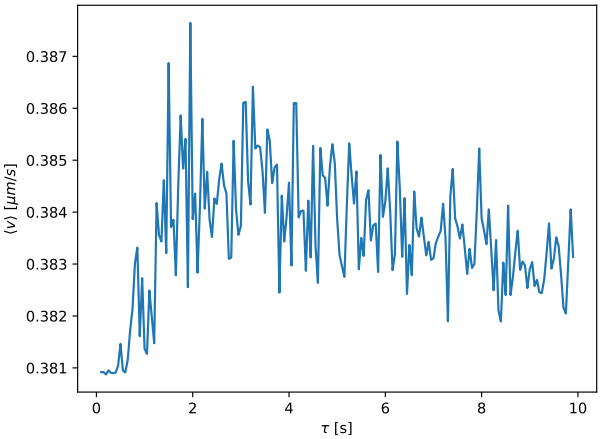


**Figure 9:** Plot of the average drift velocity, $\langle v \rangle$ of a particle with radius $r_2$.

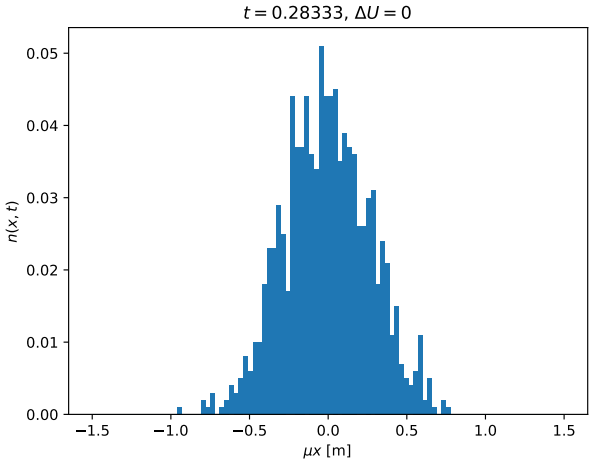


**Figure 10:** Plot of the the density of particles, $n(x,t)$ as a function of $x$, for $t = 0.28333$.

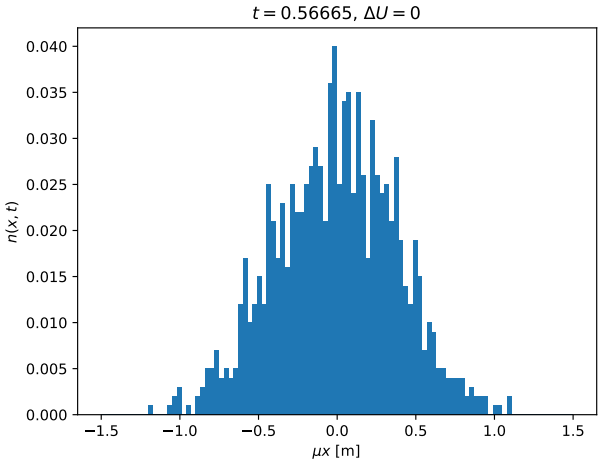


**Figure 11:** Plot of the the density of particles, $n(x,t)$ as a function of $x$, for $t = 0.56665$.

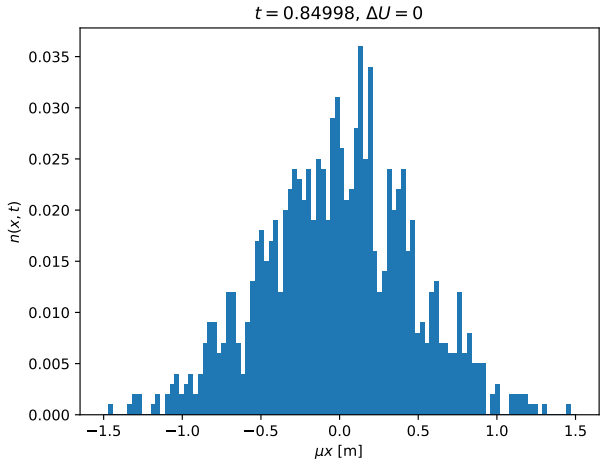


**Figure 12:** Plot of the the density of particles, $n(x,t)$ as a function of $x$, for $t = 0.84998$.

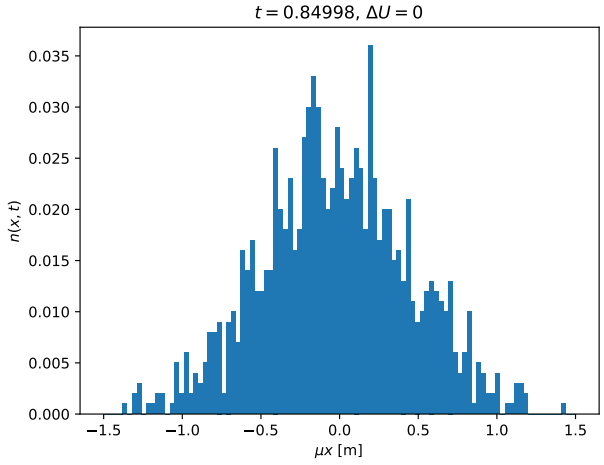


**Figure 13:** Plot of the the density of particles, $n(x,t)$ as a function of $x$, for $t = 0.84998$, for $r = r_2$.

tory, this has as seemingly negligible effect for the particle distribution.

Now the potential is turned on, and set to $\Delta U = 80$eV (as before). The value for the flashing parameter used is $\tau = \tau_{\text{op}} = 0.78s$, using $N = 200$ particles. Both particle types were included. In Figure 14 the result is plotted. There does not appear to be any particle separation, but due to time constraints this was not further investigated.

.

## 4. Conclusion

The trajectory for particles in a ratchet potential undergoing Brownian motion was studied. The Forward Euler method was applied to compute the trajectories. Three regimes for the length of time the potential is turned on

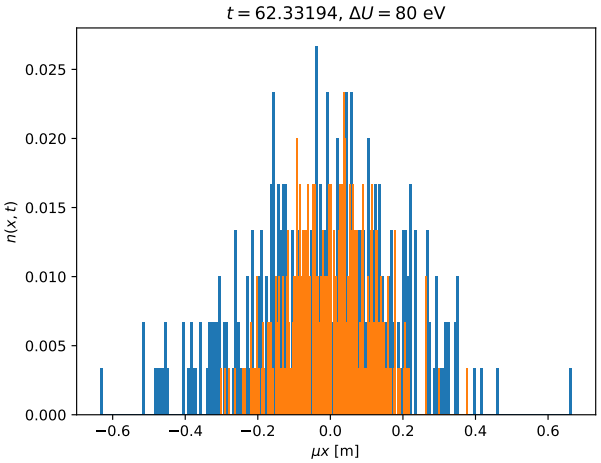


**Figure 14:** Plot of the the density of particles, $n(x,t)$ as a function of $x$, for $t = 62$, for $r = r_1$ and $r = r_2$.

was found (flashing time). The optimum flashing time, giving the largest drift velocity, for the system with $r = r_1$ was found to be $\tau_{\text{op}} = 0.78$, which gave a drift velocity of $\langle v_1 \rangle = 0.027 \mu$m/s. It was found for particles of size $r_2$ that $\tau_{\text{op, 2}} = 1.95$.

## References


[1] Jean-Philippe Banon and Ingve Simonsen (2020), *Assignment 2: Biased Brownian Motion: An Application to Particle Separation* [online, retrieved March 30th 2020]

[2] C. Sanderson and R. Curtin, *Armadillo: a template-based C++ library for linear algebra.* Journal of Open Source Software, Vol. 1, pp. 26, (2016).

[3] R. Dean Astumian, *Thermodynamics and Kinetics of Brownian Motor.* Science 9, Vol. 256 no. 5314:917–922, 1997.

[4] J. S. Bader, R. W. Hammond, S. A Henck, M. W. Deem, G. A. McDermott, J. M. Bustillo, J. W. Simpson, G. T. Mulhern, and J. M. Rothberg. DNA transport by a micromachined Brownian ratchet device. Proceedings of the National Academy of Sciences of the United States of America, 96(23):13165–13169, Nov. 9, 1999.

# Computation of Line Tension of the Two Dimensional Ising Model Using the Extended Mon-Jasnow Algorithm

TFY4235 Exam, Solfrid Hagen Johansen

*Discussed with Jeanette Bonden Isachsen and Kristine Biering Mohr.*

**Abstract**

The (original) Mon-Jasnow Algorithm and extended Mon-Jasnow Algorithm for a two dimensional cubic lattice, modeled by the Ising model was studied using the Metropolis Monte Carlo Method. It showed a phase transition in the line tension, $\tau$, at $T_c \approx 2.23K$, in agreement with Onsager's theoretical result [5]. The onset of the phase transition for small lattice sizes, $N$, was more abrupt for the extended algorithm than the original. The relationship between $\tau$ and $N$ was shown to have a slope of $-0.88$ for the extended model, and $-0.39$ for the original method. The value of $\tau_0$, found in a scaling law for $\tau$ [1], was found to be $\tau_0 = 4.252$ for the extended model. The deviation from the analytical result by Onsager, $\tau_0 = 3.99$, [5], was 20%.

## 1. Introduction

The two dimensional Ising Model is known analytically to display phase transitions [5]. Here a numerical approach is used to compute the line tension of a two dimensional cubic lattice. The computation is done using the extended Mon-Jasnow Algorithm, and the Metropolis Monte Carlo method. In addition, the original Mon-Jasnow algorithm [4] is used to compare results with published articles. The code is written in C++ using Armadillo [3], which is a library for linear algebra and scientific computing. The data computed is exported to Python, and visualized using matplotlib. Some simple regression using the SciPy library is done.

We first study the Mon-Jasnow Algorithm. First the theoretical derivations needed is done, then the numerical method used is summarized. The numerical method is then explained in more detail. Then the theoretical framework for the extended Mon-Jasnow Algorithm is presented, and the modifications of the numerical method from the Mon-Jasnow Algorithm needed is explained. Finally the numerical results are presented and compared.

## 2. Theory and Numerical Method

In the following, we set $J = 1$ and $k_B = 1$, to simplify. In addition $N_x = N_y = N$, i.e. the matrix is squared. Two different boundary conditions were used. The first type of boundary conditions gives the Mon-Jasnow Algorithm, and the second type is the *extended* Mon-Jasnow Algotithm. The algorithms are based on the description given in [1].

### *2.1. Implementation of the Mon-Jasnow Algorithm*

The procedure which needs to be implemented is the computation of $\tau$, which is derived in Reference [1], as

$$\tau = -\frac{T}{N} \ln \left\langle \exp\left[ -\frac{H_{+-} - H_{++}}{T} \right] \right\rangle_{++}. \tag{1}$$

The implementation is reliant on the fact that the expectation value is the average over the $H_{++}$ state. Thus the only difference in the $H_{+-}$ and the $H_{++}$ state is the boundary condition. As explained in Reference [4], this allows (1) to be written as

$$\tau = -\frac{T}{N} \ln \langle \exp\left[-2m_s/T\right] \rangle_{++} = \frac{2}{N} m_s \tag{2}$$

where $m_s$, is the sum over the spins is the rightmost column of either the $H_{++}$ or $H_{+-}$ state (they are equal, so it does not matter). It is therefore clear that only one lattice is needed.

The main steps used in the algorithm is summarized as follows:

First the spin-matrix is initialized to random values, and boundary conditions added. For each Monte Carlo Sweep:

1. Repeat $N \times N$ times
   (a) Select a random index of the spin-matrix.
   (b) Compute the change in the Hamiltonian, $\Delta H_{++}$ due to flipping a spin.
   (c) If $\Delta H_{++} < 0$ it is energetically favorable to flip spin, so it is flipped, and updated in the matrix.
   (d) If not, a random number between 0 and 1 is selected. if this number is smaller than the transition probability, $W$, the spin is flipped, and updated in the matrix.

2. After reaching equilibrium: Compute the magnetization, $m_s$, by summing over the rightmost (non-fixed) column of the. spin-matrix.
3. Add $m_s$ to the total sum of magnetization.

Finally, the sum of $\tau$ is divided by the number of sweeps (after equilibrium), and multiplied by $2/N$ (in accordance with Equation (2). This is done for different temperatures and lattice sizes, $N$, to obtain the result wanted. Below the steps are explained in more detail.

To implement the problem numerically, one lattice of size $N \times N$, modeled by a matrix of size $N \times (N+2)$ is implemented. The two additional columns is to allow for the boundary conditions. The spins have randomly chosen values, either $-1$ or 1, as the initial values. This is done to avoid local minimums in energy, which could make the Monte Carlo algorithm take a lot of sweeps to get to equilibrium. To account for the boundary conditions in the $x$-direction, the first and last columns were set to 1.

The numerical method for computing the needed expectation value, Equation (2), was implemented using the Metropolis Monte Carlo method (as described in Reference [2] and above).

The transition probability used is

$$W = \exp(-\Delta H_{++}/T). \tag{3}$$

Here

$$\Delta H_{++(i,j)} = 2\Delta\sigma_{i,j} \sum \sigma_{k,l}, \tag{4}$$

where $(i,j)$ denotes the current lattice site and the sum is over all nearest neighbors of the spin. $\Delta\sigma_{i,j}$ is the change of the spin, if the spin is flipped. I.e. if $\sigma_{i,j} = 1$, then the change in the spin if it is flipped is $\Delta\sigma_{i,j} = -1$. In the same way, if $\sigma_{i,j} = -1$, then $\sigma_{i,j} = 1$.

In the case at hand, with a square lattice, Equation (4) is written as

$$\Delta H_{++(i,j)} = 2\Delta\sigma_{i,j}[\sigma_{i+1,j} + \sigma_{i-1,j} + \sigma_{i,j+1} + \sigma_{i,j-1}]. \tag{5}$$

To deal with the boundaries, the algorithm uses periodic boundary conditions in the $y$-direction, as explained in Reference [1]. This is implemented in the code by first implementing a modulo function, `mod(inti,intj)` (to ensure correct behavior, and noting that the `%` operator in C++ is not defined exactly the same as in mathematics). Then the value of $\sigma_{0-1,j}$ is computed by `mod(-1,N)`,so that the index returned is $N-1$, i.e. the bottom row. The code for computing the change in $H_{++}$ is implemented in a function `computedeltaH`.

As explained, for a spin-flip to happen in a MC sweep, the computed transition probability, $W$, has to be larger than some number $r$ randomly selected from a uniform distribution between 0 and 1. Thus in the code, for each attempt of a spin-flip, $\Delta H_{++(i,j)}$ is first computed, and then $W$ from this. The reason for doing $N \times N$ attempts at flipping a spin for each sweep is so that statistically each possible spin-flip is attempted.

If the spin is to be flipped, the code simply updates the matrix by multiplying the current spin by -1, as this (obviously) gives a positive value if it was negative, and negative if it was positive.

### *2.2. Implementation of the Extended Mon-Jasnow Algorithm*

The extended Mas-Jasnow Algorithm consists of giving one lattice periodic boundary conditions (torus), and giving one periodic Klein Bottle boundary conditions, as described in Reference [1]. The reason for implementing another numerical method, is that for small lattice sizes, $N$, the ordinary Mon-Jasnow Algorithm's boundary conditions will disturb the calculation [1]. The extended method attempts to improve on this.

The value which needs to be computed is

$$\tau = -\frac{T}{N_y} \ln \left\langle \exp\left[-\frac{H_k - H_t}{T}\right] \right\rangle_t, \tag{6}$$

where $H_k$ is the Hamiltonian of the Klein Bottle lattice, and $H_t$ is the torus.

As the Hamiltonians above are computed in the system governed by $H_t$, $H_t$ and $H_k$ will be identical except for the boundary conditions. So also in this case, only one lattice is needed, which is modeled by a matrix of size $N \times N$. To find the expectation value, the Metropolis Monte Carlo Algorithm was used again.

The same procedure as for the Mas-Jasnow Algorithm above was utilized. To accommodate for the changed boundary conditions, the transition probability used is now $W = \exp(-\Delta H_t/T)$. In the code, $\Delta H_t$ is computed by summing over all nearest neighbors, as for the Mas-Jasnow Algorithm, but now there are periodic boundary conditions for the $x$-direction as well.

To compute the expectation value, the approach was to compute the total Hamiltonian of each case, and then take $H_k - H_t$. The $H_t$ case is straight forward, adding periodic boundary conditions in both directions so that it forms a torus, in the same way as described for the original algorithm.

The $H_k$ case also has periodic boundary conditions in the $x$-direction, as the structure forms a ring in the vertical direction. This was implemented in the same way as previously. In the $y$-direction, the structure forms a Möbius strip. Therefore to compute the value of the Hamiltonian for an element at the left or right of the lattice, the following was done:

- If at the leftmost column, $(i, j = 0)$, the nearest neighbor to the left is taken to be the spin at the rightmost column, $j = N - 1$, at row number $i = N - 1 - j$.
- If at the rightmost column, $(i, j = N-1)$, the nearest neighbor to the right is taken to be the spin at the leftmost column, $j = 0$, at row number $j = N-1-j$.

As explained in [1], the sign is also flipped for these boundaries.

(Note: the description stated in [1] describes the Möbius strip in the $x$-direction, but due to the periodic boundary conditions of the other lattice, this does not matter)

The actual computation of $H_k$ and $H_t$ was done by, for each lattice point, summing over its nearest neighbors, while complying with the boundary conditions described above, i.e.

$$H = -\frac{J}{2} \sum_{\langle \mathbf{m},\mathbf{n} \rangle} \sigma_{\mathbf{m}} \sigma_{\mathbf{n}}. \tag{7}$$

There are more efficient ways of doing this, by noting that the only values needed are those at the boundaries. However the implementation used is easy to explain and straight forward to implement, and therefore the method of choice.

$\exp(-(H_k - H_t)/T)$ is computed after reaching equilibrium, and measured multiple times, averaged over the number of sweeps. The obtained average then gives the line tension, $\tau$.

## 3. Results and Discussion

To determine the number of sweeps, $t$, needed to reach equilibrium for different temperatures $T$, a convergence test was performed, using the Mon-Jasnow Algorithm. The temperatures that will be dealt with are in the range $T = 1$ K to $T = T_c = 2.2691$ K, which is the critical temperature where the phase transition happens (from the analytical result) [5]. The convergence test was done by computing the magnetization for different temperatures as a function of number of sweeps. The goal is for the magnetization to become contained within a small range. The result is shown in Figure 2, computed with lattice size $N = 50$. The Figure indicates that about $t = 450000$ sweeps are needed to ensure equilibrium. However this varies strongly on temperature, and the number decreases with temperature, only $t = 30000$ sweeps were used for the calculations (due to time constraints). It is the values closer to the critical temperature which is of the main interest, where less than $t = 5000$ sweeps are needed.

The number of sweeps needed also depends on the size of the lattice, $N$. In Figure 2 the magnetization is plotted as a function of number of sweeps, $t$, at $T = 1.9K$ for

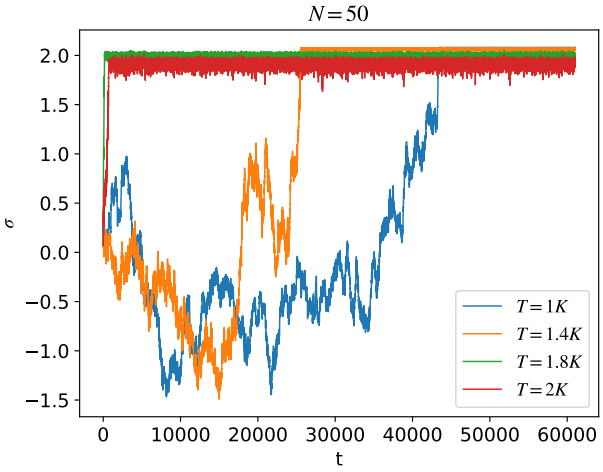


**Figure 1:** *Plot of magnetization, $\sigma$, as a function of number of sweeps $t$, for $N = 50$, for different temperatures.*

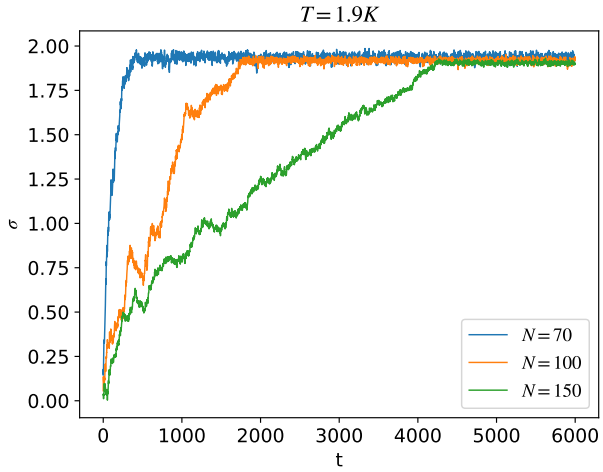


**Figure 2:** *Plot of magnetization, $\sigma$, as a function of number of sweeps $t$, for $T = 1.9K$, for different lattice sizes.*

different values of $N$. Here it becomes apparent that the number of sweeps needed depends on the size of the lattice. Due to the temperature dependence observed as well, one will expect the number of sweeps needed for low temperatures (less than around $T = 1.4K$) to increase rapidly with $N$. Therefore the results are computed mostly for lattices up to size $N = 50$, to avoid very time consuming computations, while still getting accurate results.

The state of the lattice before and after MC sweeps, for $N = 100$ and $T = 1.9K$, are shown in Figure 3 and Figure 4. It can be observed that after 11 000 MC sweeps, there appears to be clustering of the spins.

### *3.1. Results for the Mon-Jasnow Algorithm*

To find the line tension as a function of time using the Mon-Jasnow Algorithm, $t = 20000$ sweeps were used to get to equilibrium. Then the next $t = 1000$ sweeps were


**Figure 3:** *Visualization of the initial state of a lattice of size $N = 100$. Black is spin down (-1) and yellow is spin up (1).*

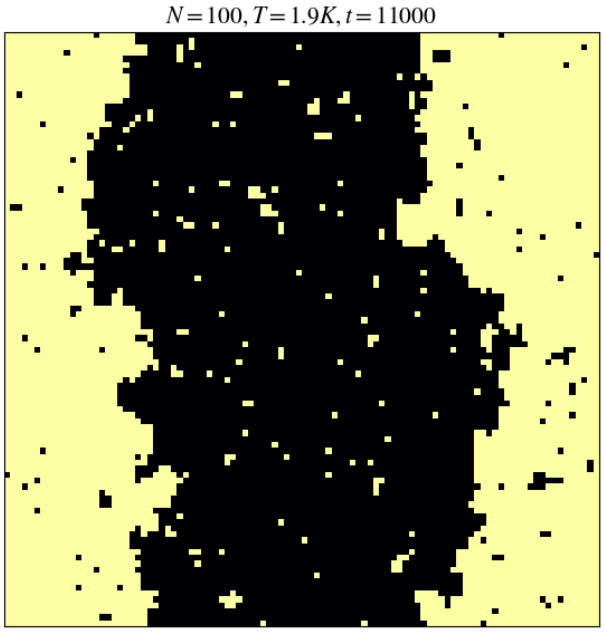


**Figure 4:** *Visualization of the state after the final sweep, $t = 11000$, of a lattice of size $N = 100$ at $T = 1.9K$. Black is spin down (-1) and yellow is spin up (1).*

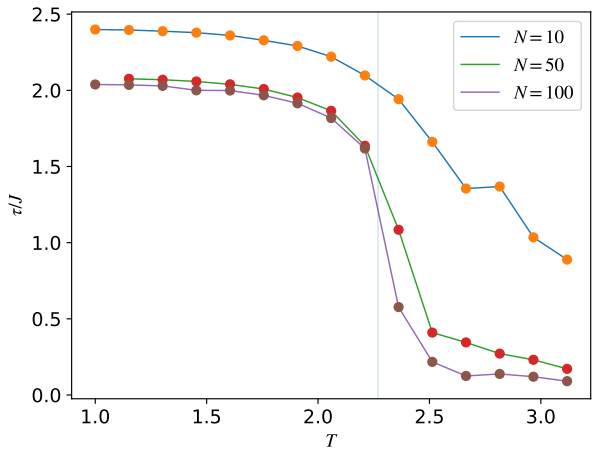


**Figure 5:** *Plot of the line tension, $\tau$, as a function of temperature, $T$, for $N = 10$,$N = 50$ and $N = 100$. The transparent line shows the value of the critical temperature, $T = 2.3$.*

used to compute the line tension. Figure 5 shows the line tension, $\tau$, as a function of temperature, $T$, for $N = 10$, $N = 50$ and $N = 100$. One can observe an abrupt decrease in $\tau$ at approximately $T = 2.3$. This is resembles as first order phase transition in the order parameter, $\tau$ [6] .

The Figure also shows the analytical solution for the magnetization, found by Onsager. This analytical solution is

$$2m = \begin{cases} 0 & , T > T_c \\ 2\left\{1 - [\sinh(2\beta J)]^{-4}\right\}^{\frac{1}{8}} & , T < T_c \end{cases} \tag{8}$$

as shown in [5]. Note that this is derived for a two dimensional lattice which is infinite in one direction. In the plot of Figure 5 the numerically obtained values are close to the analytical solution for $N = 50$ and $N = 100$, but deviates significantly for $N = 10$. Observe also that the onset of the phase transitions happens more abruptly with increasing $N$.

In Figure 6, $N\tau$ at $T = T_c = 2.2691K$ is plotted as a function of $N$. Note that the plot is on a log-log scale. The computed values were from $N = 2$ to $N = 50$. As these values are computed at $T = T_c$, only $t = 10000$ sweeps to reach equilibrium were deemed necessary. $t = 1000$ sweeps after equilibrium were used to obtain the values.

The dips observed for large $N$ could be due to not using an adequate number of sweeps, as the number of sweeps needed increases with $N$, as demonstrated for $T = 1.9$K in Figure 2. However the trend of a straight line for large values of $N$ is still apparent.

It can also be observed that the line is not linear for small $N$. As stated in Reference [1], the utilized method does not work well for small $N$ due to the boundary conditions

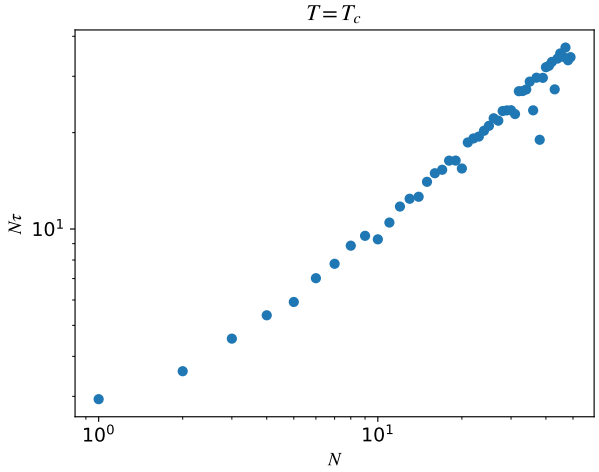


**Figure 6:** *Plot of the line tension, $\tau$, times the size of the lattice, $N$, as a function of $N$ at $T = T_c$.*

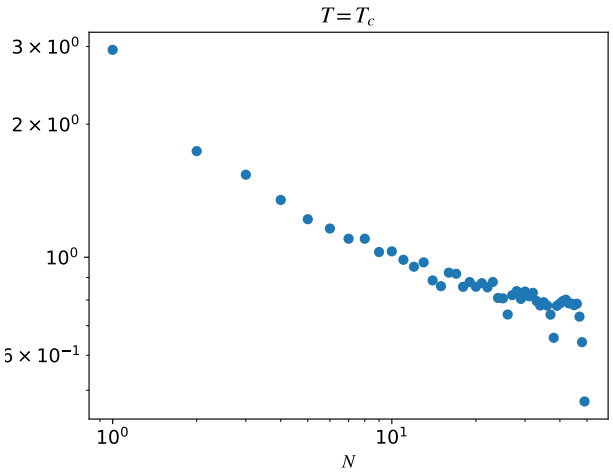


**Figure 7:** *Plot of the line tension $\tau$ as a function of $N$ on a log-log scale, at $T = T_c$, for the ordinary Man-Jasnow Algorithm.*

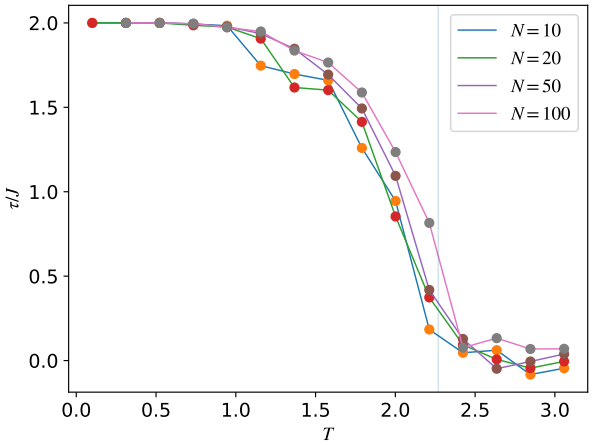


**Figure 8:** *Plot of the line tension, $\tau$, as a function of temperature, $T$, for different lattice sizes, $N$, using the extended Mas-Jasnow Algorithm. The transparent line shows the value of the critical temperature.*

used. This is also observed in Figure 5, where the phase transition happens more abruptly for larger $N$.

For comparison the line tension, $\tau$, is also plotted as a function of lattice size, $N$, for the ordinary Mon-Jasnow Algorithm in Figure 7. The slope was found to be $-0.39$.

### *3.2. Results for the extended Mon-Jasnow Algorithm*

The value of $\tau$ using the extended Mon-Jasnow Algorithm was computed for $N = 10$, $N = 40$ and $N = 100$ as a function of temperature, $T$, which is shown in Figure 8.

We can observe a abrupt decrease at around $T = 2.3K$ as expected. Again this is resembles a first order phase transition. The transition becomes more abrupt for larger values of $N$, which agrees with the observation in Figure 5.

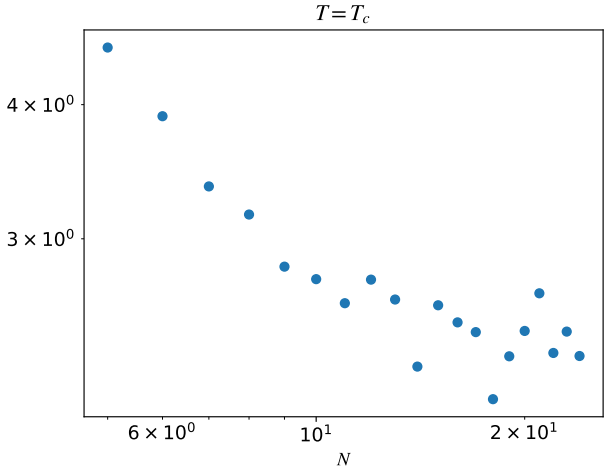


**Figure 9:** *Plot of the line tension $\tau$ as a function of $N$ on a log-log scale, at $T = T_c$, for the extended Jan-Masnow Algorithm.*

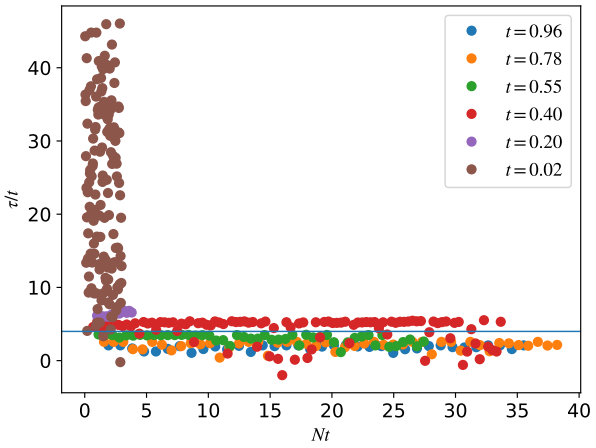


**Figure 10:** *Plot of $\tau/t$ as a function of $Nt$ for multiple values of $t$.*

In Figure 9, $\tau$ at $T = T_c = 2.2691K$ is plotted as a function of $N$ on a log-log scale. The computed values were from $N = 25$ to $N = 25$. As these values are computed at $T = T_c$, only $t = 10000$ sweeps to reach equilibrium were deemed necessary. For each $N$, the computation was done 100 times. To obtain the value of the slope, the data was imported to Python and the linear regression function `linregress`, which is part of SciPy's `stats` package, was used. The regression gave a slope of $b = -0.88$. For large $N$ the trend does appear to become less linear, which could be due to not using an adequate number of sweeps.

According to Reference [1], one can write the line tension as

$$\tau = \tau_0 t^{\mu} \Sigma \left( N^{1/\nu} t \right), \tag{9}$$

where $t = (T_c - T)/T_c$. (Note: $t$ no longer the number of sweeps). Onsager's exact solution gave the parameters $\mu = 1$, $\nu = 1$ and $\tau_0 \approx 3.99$ [1]. The sum $\sum(z)$, becomes $\sum(z) \to x^{-\mu}$ for $z \to 0^+$. For the critical temperature, $t = 0$, the scaling law, Equation (9) thus becomes

$$\tau = \tau_0 t^{\mu} (N^{1/\nu} t)^{-\mu} = \tau_0 N^{-\mu/\nu}. \tag{10}$$

Taking the logarithm of both sides, one obtains

$$\log(\tau) = \log(\tau_0 N^{-\mu/\nu}) \tag{11}$$

$$\log(\tau) = \log(\tau_0) - \frac{\mu}{\nu} \log(N), \tag{12}$$

so that the expected slope of a log-log is $-\mu/\nu = -1$. The deviation from the computed value could be due to finite-size effects, as we still use periodic boundary conditions. In addition, the points computed for large $N$ have a high variance, which could affect the result of the linear regression.

In Figure 10, $\tau/t$ is plotted as a function of $Nt$ for different values of $t$. One can observe that the plots for large values of $N$, for different $t$ follow approximately the same curve.

As previously stated, Onsager's exact solution gave $\tau_0 \approx 3.99$ in Equation (9). In the same equation, as the product $N^{1/\nu} t \to \infty$, the sum $\sum N^{1/\nu} t$ goes to 1 so that [1],

$$\tau/t = \tau_0. \tag{13}$$

Therefore, for large values of $Nt$, the value $\tau/t$ will go to the constant value $\tau_0$ (i.e. independent of $t$). To estimate $\tau_0$ from Figure 10, note that the values are constant for intermediate values of $Nt$, but somewhat differ for different $t$.

To determine the estimate, note that from Figure 1 it is observed that the number of sweeps need to reach equilibrium is smaller the closer one gets to the critical temperature. The number of sweeps used to reach equilibrium were only 2000, due to computation time. So therefore the values computed for small $t$, i.e. close to $T_c$ are assumed to be the closest to the correct values. However, as can be seen from Figure 10 the lattice size, $N$, needed for the product $Nt$ to be a large value (as the method for finding $\tau_0$ requires large $Nt$), is large for small $t$. Therefore, the value with the smallest value of $t$ which still produced a relatively large product $Nt$ is used to estimate $\tau_0$.

From Figure 10 the plot for $t = 0.4$ was therefore used to estimate $\tau_0$ to be $\tau_0 = 4.252$. The value was estimated by taking the mean over all values for that temperature. The result is 6.5% above the exact value. In literature [4] for the ordinary Mon-Jasnow Algorithm, a deviation

of 20% was found. It should be noted that the computational power utilized has significantly increased since this result, and therefore it is difficult to give a fair comparison between the two methods from this.

### *3.3. Comparison of the ordinary and extended method*

As observed in Figures 5 and 8, the line tension decreases rapidly at around $T = 2.27$ for both methods, demonstrating the onset of a first order phase transition.

The differences between the methods occur for small values of $N$. The reason that the extended Mon-Jasnow Algorithm was implemented was the failure of the Mon-Jasnow Algorithm at small $N$. This may be observed in Figure 5 and Figure 8 where the line tension for $N = 10$ is plotted. For the original algorithm the phase transition can barley be observed, while for the extended algorithm it is significantly more abrupt.

The same effect is expected seen for log-log plots $\tau$ as a function of $N$, Figures 7 and 9. Notice that the plot of the extended algorithm is linear even for small $N$. For the original algorithm at small $N$, the trend is non-linear. Analytically it is known that the trend should be linear (Equation (12)), therefore the extended algorithm is an improvement for small $N$.

Physically the differences between the results of the methods is clear. The original algorithm fixes the spins to be either 1 or $-1$ at the horizontal boundaries. For a large lattice size, these fixed spins will be a small fraction of the total system and their effect will not be so apparent. However for smaller lattice sizes, these spins will make up a larger fraction, and can significantly impact the system. In the extended algorithm the effect for small lattices is improved as no spins are forced to take a fixed value, but are computed using special periodic boundary conditions. Note that periodic boundary conditions are applied assuming a large system size, $N$, but the effects of having fixed boundary conditions is here shown to impact more than periodic ones.

From the plot of the line tension for the extended algorithm, Figure 8, it appears that $\tau$ deviates more from the analytical result than the ordinary method. So for computation of $\tau$ for large $N$, the ordinary method could be the better choice.

## 4. Conclusion

By using an intermediate number of sweeps, due to the proximity to a critical value, the line tension of a two dimensional lattice Ising model was found as a function of temperature. It was shown, in agreement with analytical results, that the line tension exhibits a phase transition at $T \approx 2.27$K. Using the extended Mon-Jasnow Algorithm, the slope of the log-log plot for $\tau$ vs. $N$ was estimated to be $-0.88$, while it was estimated to be $-0.39$ for the original algorithm. The value of $\tau_0$ was approximated to be $\tau_0 = 4.252$ using the extended algorithm. The methods used were shown to exhibit similar behavior, except for at small lattice sites $N$, where the extended algorithm clearly shows the phase transition while the ordinary does not.

## References


[1] Ingve Simonsen, *Exam in TFY4235/FY8904 Computational Physics* [online, retrieved May 6$^{\text{th}}$ 2020]

[2] I. Simonsen, *Lecture Slides TFY4235* [online, retrieved May 6$^{\text{th}}$ 2020]

[3] C. Sanderson and R. Curtin, *Armadillo: a template-based C++ library for linear algebra.* Journal of Open Source Software, Vol. 1, pp. 26, (2016).

[4] K.K. Mon and D. Jasnow, *Direct calculation of interfacial tension for lattice models by the Monte Carlo method*, Phys. Rev. A 30, 670 (1984).

[5] L. Onsager, *Crystal statistics. I. A two-dimensional model with an order-disorder transition*, Phys. Rev. 65, 117 (1944).

[6] J. O. Andersen, *Introduction to Statistical Mechanics*, Fagbokforlaget (2012).