Utrecht University

Utrecht University
Department of Mathematics

Ergodicity Properties of Opinion Models

Master's Thesis by Annika Brockhaus

Supervisor:	dr. W.M. Ruszel
Second Reader:	dr. K. Dajani
Date of Submission:	30th of August, 2024

Acknowledgements

I want to thank my supervisor Wioletta Ruszel for her support and guidance throughout this project. I could not have wished for a better supervisor!

I would also like to thank Vanessa Jacquier for her help with studying Markov processes and my second reader Karma Dajani for taking the time to read this (lengthy) thesis and come to my presentations.

Furthermore, I want to thank my fellow students and the mathematics library for providing a comfortable and supportive study environment. Particular thanks go to Fien for proofreading, Karl for enlightening discussions on programming methods, and Signe for lots of moral support during the final phase of this project.

Finally, I want to thank my family, in particular my parents, for their support throughout my studies.

Abstract

In this thesis, we study the long time behaviour of two specific models; the Compass model and the Deffuant model, which are special examples of opinion models. Opinion models are special stochastic Markov processes in which one wants to understand how global opinions can be formed given specific types of interaction.

Both models are first introduced as Markov processes and then simulated numerically and compared. The numerical simulations verify known analytical results on their respective ergodic properties.

Furthermore, the impact of two different types of noise, the uniform and the bi-modal noise, on both models is analysed using numerical simulations. The goal is to gain insight into the influence of the noise on the behaviour of the models. From the simulations we conclude that small uniform noise has little influence on the model behaviour. In contrast, even the smallest bi-modal noise changes the ergodic properties of the Compass model significantly, while not impacting the Deffuant model much.

Contents

Part I.

Introduction and Background

1. Introduction

In this thesis we study opinion models, which are mathematical models for the time evolution of opinions of a group of individuals. These models can, for example, be applied to model opinion formation leading up to elections.

In general, individuals are seen as vertices on a graph that can interact only if they are neighbours, that is, if they are connected by an edge. There are many different types of graphs used; complete finite graphs, lattices, trees, random networks, and many more.

The opinions of individuals can be modelled by any sort of set. Often either finite sets or metric spaces are used.

How and when interactions between individuals occur and how these interactions change the opinions of the individuals involved is also model specific. An individual could for example take the average of the opinions of a selection of its neighbours or a group of individuals could agree on some form of compromise whenever they interact. Many other types of interactions are possible as well.

Furthermore, one might want to model external influences, for example by adding (random) noise.

When studying these models, one mainly wants to know what long-time behaviour the group exhibits. Will all individuals eventually agree? And what opinion will they agree upon? When everyone agrees, we speak of consensus. Note that this concept is mathematically analogous to that of synchronisation used in different areas of application.

From a mathematical point of view, opinion models are often studied in terms of interacting particle systems. Some background literature can be found in [FV17], [Lig85] and [Lig10, Chapter 4]. These are often modelled using stochastic processes which, roughly speaking, model the evolution of a random variable in time. This randomness is useful, since an opinion can change and interactions in groups often have a random component. Stochastic processes in which the future of the process only depends on the present and not its past are called Markov processes. These are particularly useful, because there exist many theoretical results on their long-time behaviour. Many opinion models are therefore modelled as Markov processes. An introduction to the theory of stochastic processes and Markov processes can, for example, be found in the [Lig10] and [KS12]. An introduction to the topic using the example of Brownian motion is given in [MP10].

Opinion models can be divided into models with a finite set of opinions and models with a continuous opinion space.

One of the first models applied as an opinion model is the Ising model. It was originally proposed as a model of ferromagnetism, but has since been applied to numerous other topics, such as opinion models [HRFS$^+$22, Chapter 2.1.1]. It is for instance introduced in [FV17] and [Lig85, Chapter IV]. In the Ising model, the opinions lie in the set $\{-1, 1\}$ and neighbours tend to agree with one-another, resulting in configurations that minimise a given energy function. After originally being defined on the lattice, this model has then been studied by on various graphs. For example in [DGM02], the authors investigate the Ising model on random networks. In [BGJZ23], the Ising model with external influences is studied on specific examples of finite clustered networks. Furthermore, in [TYS21], the authors use a variant of the Ising model to investigate real-world behaviour of voters.

Another well-known model in which opinions can only take two values is the voter model. At every time step an individual is selected at random. This individual changes its opinion to be equal to that of a randomly selected neighbour. See [Lig10, Chapter 4.3], [CFL09, Chapter III.B] and [Lig85, Chapter V] for an analysis of this model. An application of the voter model to elections in the United States can be found in [FGSR$^+$14].

A popular model with a continuous opinion space, usually given by $[0, 1]$, is the DeGroot model. In this model, an individual updates its opinion to a weighted average of the opinions of its neighbours, where the weights vary per individual and are constant in time. More information on the DeGroot model, including additional references, can be found in [DDK24].

Bounded confidence models provide another group of models with continuous opinion spaces. In these models, individuals adjust their opinions during an interaction only if they are sufficiently similar, that is, if the distance between their opinions is smaller than a given confidence bound [HRFS$^+$22, Chapter 2.2.2]. Two examples of bounded confidence models, the Deffuant model (originally introduced in [DNAW00]) and the Compass model (originally introduced in [GHH20]), are studied in this thesis.

The starting point of this thesis was the paper "Strictly weak consensus in the uniform compass model on $\mathbb{Z}$" by N. Gantert, M. Heydenreich and T. Hirscher from 2020 [GHH20]. In this paper the Compass model is introduced and compared to the Deffuant model. The dynamics of the Compass model are defined analogous to those of the Deffuant model. However, the opinions in the Compass model are represented as angles in contrast to the finite intervals used for the Deffuant model. In both models, two individuals adjust their opinions to a value that is closer to their average opinion when they interact with each other, which happens at random times. If the initial opinions are chosen independently and uniformly at random for each individual, we speak of the Uniform Deffuant model or Uniform Compass model.

The Uniform Deffuant model has already been studied extensively. In [DNAW00], G. Deffuant, D. Neau, F. Amblard and G. Weisbuch proposed this model on the complete graph. They showed that consensus can only be achieved for large confidence bounds. The paper [Häg12] shows that, for the Uniform Deffuant model on $\mathbb{Z}$, the critical value of the confidence bound lies at 0.5. Convergence to consensus is shown for larger confidence bounds and absence of consensus for smaller ones. This model has also been simulated

numerically on various other graphs, for example in [DNAW00], [SSS04] and [Wei04].

Our main source for the Compass model is the paper [GHH20]. In this paper, an infinite confidence bound is assumed. Furthermore, a distinction is made between strong consensus, that is, all individuals agree on the same opinion, and weak consensus, that is, the distance in opinion between all pairs of interacting individuals approaches zero. Since the definitions of weak and strong consensus rely on convergence of sequences of random variables, it is necessary to distinguish between consensus which is almost sure, in $\mathcal{L}^1$ and in probability. The main result of the paper [GHH20] is that there is strictly weak consensus in $\mathcal{L}^1$ for the Uniform Compass model on $\mathbb{Z}$. In comparison, there is almost sure strong consensus for the Deffuant model on $\mathbb{Z}$.

The main goal of this thesis is to study consensus formation in both, the Compass and Deffuant model. In particular, we are interested in the effect of random noise on consensus building. We will answer the following research question:

> How does the long-term behaviour of the Uniform Compass model change when it is perturbed by random noise?

We answer this question numerically for two different versions of noise; the uniform noise and bi-modal noise. We started by simulating both models without noise. For this case there are theoretical results on the behaviour of both the models available. Verifying these results through our simulation provides a way of checking whether the implementation is correct. We then go on to simulate the models with different types of noise. We will see that only large uniform noise has a significant impact on the dynamics of the models. In contrast, very small bi-modal noise changes the long-time behaviour of the Uniform Compass model drastically. For the Uniform Deffuant model even large bi-modal noise has little impact.

Instead of looking at the configurations of the models directly we will calculate three parameters and study them. Two of them are the order parameters that have, for example, been used to study synchronisation in the mean-field XY-model as done in [CR16]. The third parameter is the sum of all absolute differences of opinions of neighbours. This parameter comes up in the proof of consensus for the Compass model given in Lemma 3.1 of the paper [GHH20]. When it is zero, there is strong consensus.

This thesis is divided into three parts. Part I, contains the introduction and a chapter on Markov processes. The goal of this chapter is to give the background on Markov processes needed to understand Part II.

Part II consists of four chapters that give a theoretical description of the Compass and Deffuant model. In Chapters 3 and 4, the models are introduced as Markov processes. In the special case where the models are defined on a finite path, it is verified, that they are indeed well-defined. Furthermore, the different types of noise are defined. Chapter 5 provides a definition of consensus and shows strong consensus for the Compass and Deffuant model on finite paths. Chapter 6 gives some invariant measures for the Compass and Deffuant model.

Part III presents the simulations, which form the main part of this thesis. First, the implementations are explained in Chapter 7 and then the results are given in Chapter

8. This part is particularly interesting, because some of the behaviour that it analyses numerically have not been studied so far. It therefore provides original contribution to the field of opinion models. Finally, Chapter 9 gives the conclusion and an outlook on further interesting research questions.

Note that I have generated all the graphics shown in the following text myself and that, to the best of my knowledge, I explicitly stated wherever I used work that is not my own.

2. Markov Processes

The opinion models studied in this thesis are described as Markov processes; a special kind of stochastic processes. This has the big advantage that Markov processes are stochastic processes with practical properties that are rather well studied. That means that there is a lot of literature and known results about Markov processes available. The key property of a Markov process is that the future of the process is only dependent on the immediate past and not on the entire history of the process. Thus, in the discrete time case, what happens at time $n+1$ will only depend on time n and not on the entire history of the process, that is, what happens at times $1, 2, \ldots, n$. This can make calculations and proofs significantly easier.

To motivate this further, we will start by looking at a standard example for a Markov process; the simple random walk.

Example 2.0.1 (Simple random walk on $\mathbb{Z}$)**.** *Let $(a_k)_{k\in\mathbb{N}}$ be a sequence of independent random variables that are all uniformly distributed on $\{-1, 1\}$. Then consider the stochastic process $(s_k)_{k\in\mathbb{N}}$ that is given by*

$$s_k := \sum_{i=0}^{k} a_i$$

in all $k \in \mathbb{N}$. This process is then called the simple random walk on $\mathbb{Z}$. One can visualise it the following way. Imagine a line numbered by $\mathbb{Z}$, where all integers are one step apart. Then imagine someone standing on one of the integers. After every time unit, let that person take a step to the left with probability $\frac{1}{2}$ or to the right with probability $\frac{1}{2}$. The integer on which the person stands at time $k \in \mathbb{N}$ is then given by the random variable s_k.

This process is a Markov process, because the position of the person at time $k+1$ only depends on the position of the person at time k. In order to see this, note that

$$s_{k+1} := s_k + a_{k+1} = \left(\sum_{i=1}^{k} a_i\right) + a_{k+1},$$

where a_{k+1} is independent of $\sum_{i=1}^{k} a_i$. Thus, when we know the value of s_k and a_k, we will know s_{k-1}. We do not need to know the values of $s_0, \ldots, s_{k-1}$ as well.

This turns out to be a very practical property for calculations. For example, it allows to describe these processes with only two things:

1. *the starting distribution s_0, and*

2. the transition from step $k-1$ to step k.

In this example we can describe the transition from step $k-1$ to step k using an infinite stochastic matrix $(q_{i,j})_{i,j\in\mathbb{Z}}$ given by

$$q_{i,j} := \begin{cases} \frac{1}{2}, & \text{if } j = i+1 \text{ or } j = i-1, \\ 0, & \text{else.} \end{cases}$$

Here, $q_{i,j}$ gives the probability of the person moving from integer i to integer j in one time step. The simple random walk on $\mathbb{Z}$ is a standard example of a Markov chain; a special kind of Markov process. It can be found in many books on the topic, for example in [KS12] or in [BW21].

In contrast, an example of a stochastic process that is not Markov is given in the following example.

Example 2.0.2 (Stochastic process that is not Markov)**.** *Let $(a_k)_{k\geq\mathbb{N}}$ be a sequence of random variables that are uniformly distributed on $\{-1,1\}$. Define a stochastic process $(p_k)_{k\geq 0}$ by*

$$p_k := \begin{cases} \sum_{i=0}^{\frac{k}{2}} a_{2i}, & \text{if } k \text{ is even,} \\ \sum_{i=0}^{\frac{k-1}{2}} a_{2i+1}, & \text{if } k \text{ is odd.} \end{cases}$$

This process is not a Markov process as the behaviour of the next time step is always dependent on the previous time step, and not just on the present. To see this, note that for any k, $p_{k+1} = p_{k-1} + a_{k+1}$. Thus, this is not a Markov process.

The examples shown above are using discrete time. A discrete time Markov process is often called a Markov chain. They are easier to study, because they can be described using transition matrices. We will go on to study Markov process in continuous time. Our goal is to describe them using linear operators called generators.

In the following chapter we will often be using Brownian motion as an example to illustrate various concepts. Brownian motion can, intuitively, be viewed as a continuous version of a random walk. This is for example outlined in [Gal22, Chapter 1].

Throughout this chapter, let $\mathcal{B}(S)$ denote the Borel-σ-algebra on the Polish space space S.

2.1. Markov Processes and their Generators

Before diving into the topic of Markov processes, recall the following definitions.

Definition 2.1.1 (Polish space)**.** *Let S be a topological space. Assume that there exists a complete and separable metric $d : S \times S \to \mathbb{R}_+$ generating the topology. Then S is called a **Polish space**. [Kec95, Definition 3.1]*

Many standard examples of topological spaces are also Polish spaces. For example, any separable Banach space is a Polish space. This, and many more examples of Polish spaces can be found in [Kec95].

Definition 2.1.2 (Stochastic process)**.** *Let $(\Omega, \mathcal{U}, \mathbb{P})$ be a probability space and let $(E, \mathcal{E})$ be a measurable space. Then, a **stochastic process** $X = (X_t)_{t\geq 0}$ on $(\Omega, \mathcal{U}, \mathbb{P})$ with **state space** $(E, \mathcal{E})$ is a collection of random variables $X_t : \Omega \to S$ for $t \in [0, \infty)$. Stochastic processes are sometimes also called random processes. [BW21, Definition 1.1]*

Now, let us start by defining Markov processes. The following text is in large parts inspired by the lecture notes on Markov processes by Andreas Eberle from 2023. [Ebe23]

Definition 2.1.3 (Markov process)**.** *Let $(S, \mathcal{B}(S))$ be a measurable space and let $(\Omega, \mathcal{U}, \mathbb{P})$ be a probability space. Let $X = (X_t)_{t\geq 0}$ be a stochastic process on $(\Omega, \mathcal{U}, \mathbb{P})$ with **state space** $(S, \mathcal{B}(S))$. Let μ be a probability measure on $(S, \mathcal{E})$, such that $X_0 \sim \mu$. Then we call μ the **initial distribution** of the process. Furthermore, let $\mathcal{F} = (\mathcal{F}_t)_{t\geq 0}$ be the natural filtration adapted to X. If X satisfies the **Markov property***

$$\mathbb{P}\left(X_t \in E | \mathcal{F}_s\right) = \mathbb{P}\left(X_t \in E | X_s\right) \text{ almost surely}$$

*for all $0 \leq s \leq t$ and all $E \in \mathcal{E}$, it is called a **Markov process**. [KS12, Chapter 19.2]*

*If, furthermore, $\mathbb{P}(X_{t+s} \in E | X_s) = \mathbb{P}(X_{t+h} \in E | X_h)$ almost surely for every $E \in \mathcal{E}$ and every $s, h \in [0, \infty)$, then X is called **time-homogeneous**. [MP10, Chapter 2]*

If for every $\omega \in \Omega$,

$$[0, \infty) \to S, t \mapsto X_t(\omega)$$

*is almost surely a right continuous map, then X is called **right-continuous**. [KS12, Chapter 13.6]*

We start by giving an alternative formulation of the Markov property.

Remark 2.1.4. *The following formulations of the Markov property are equivalent:*

1. $\mathbb{P}\left(X_t \in E | \mathcal{F}_s\right) = \mathbb{P}\left(X_t \in E | X_s\right)$ *for all $0 \leq s \leq t$ and all $E \in \mathcal{E}$,*
2. $\mathbb{E}\left[f(X_t) | \mathcal{F}_s\right] = \mathbb{E}\left[f(X_t) | X_s\right]$ *for all $0 \leq s \leq t$ and all measurable $f : S \to \mathbb{R}_+$.*

The equivalence, which is mentioned in [Ebe23, Chapter 1], can be proven using standard machinery. Below, we worked out the proof.

Proof. First, we show that (*2.*) implies (*1.*). We start by assuming that (*2.*) holds. Then, fix a $E \in \mathcal{E}$. The function $\mathbb{1}_{X_t \in E}$ is then a measurable function $S \to \mathbb{R}_+$. Thus,

$$\mathbb{E}\left[\mathbb{1}_{X_t \in E} | \mathcal{F}_s\right] = \mathbb{P}\left(X_t \in E | \mathcal{F}_s\right)$$

and

$$\mathbb{E}\left[\mathbb{1}_{X_t \in E} | X_s\right] = \mathbb{P}\left(X_t \in E | X_s\right).$$

Since one can do this for any $B \in \mathcal{E}$, (*1.*) follows.

For the other implication, assume that (*1.*) holds. Let $f : S \to \mathbb{R}_+$ be a measurable function. Then approximate f from below by a sequence of simple functions that are made up from indicator of the form $\mathbb{1}_{X_t \in E}$ where $E \in \mathcal{E}$. Afterwards, use the monotone convergence theorem for conditional expectations to get (*2.*). □

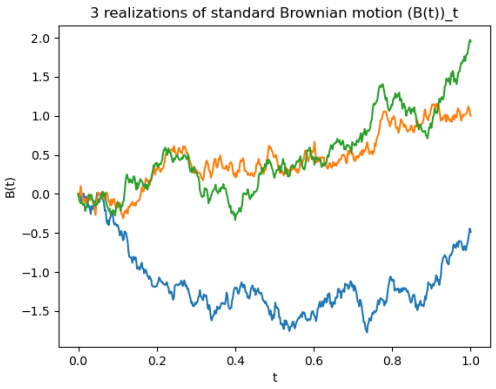

Figure 2.1.: Three realisations of Brownian motion.

From now on, we will only consider right-continuous time-homogeneous Markov processes unless explicitly stated otherwise.

Before introducing any more properties of Markov processes, let us consider Brownian motion as an example.

Example 2.1.5 (Brownian motion)**.** ***Brownian motion*** *is defined as a stochastic process* $(B_t)_{t\geq 0}$ *on a suitable probability space* $(\Omega, \mathcal{U}, \mathbb{F})$ *with state space* $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ *such that the following conditions hold:*

1. $B_0 = 0$,
2. $B_{t_1} - B_{t_0}, B_{t_2} - B_{t_1}, \ldots, B_{t_n} - B_{t_{n-1}}$ *are independent for all* $0 \leq t_1 \leq t_2 \leq \cdots \leq t_{n-1} \leq t_n$ *(we speak of* ***independent increments****),*
3. $B_t - B_s \sim \mathcal{N}(0, |t-s|)$ *for all* $t, s \geq 0$,
4. $t \mapsto B_t$ *is almost surely continuous.*

The existence of Brownian motion is a standard result that can, for example, be found in [Lig10, Chapter 1.5]. According to [SP12, Theorem 6.1], for every $s > 0$*, the process* $(B_{t+s} - B_s)_{t\geq 0}$ *is again a Brownian motion and independent of* $\mathcal{F}_s$*. According to [SP12, Theorem 6.2], this property is equivalent to the version (2.) of the Markov property given in Remark 2.1.4. That Brownian motion is time-homogeneous follows immediately. Right-continuity, follows from property (4.) in the definition above. Thus, Brownian motion is an example of a time-homogeneous right-continuous Markov process.*

An illustration of three realisations of Brownian motion can be found in Figure 2.1.

A common way to characterise a Markov process is through its transition function. We will define the transition function of a Markov process by first defining a sub-probability kernel. This set of definitions (2.1.6 and 2.1.7) is strongly inspired by the lecture notes by A. Eberle [Ebe23]. Note, however, that equivalent definitions can be found in other

literature. One example is the definition of Markov transition probabilities found in [Bas11, Definition 19.2]. Another is the Markov transition kernel in [MP10, Definition 2.30]

Definition 2.1.6 (Sub-probability kernel)**.** *A function $p : S \times \mathcal{B}(S) \to [0,1]$ is called a **sub-probability kernel**, if*

1. *$p(x,\cdot)$ defines a measure on $(S,\mathcal{B}(S))$ for each $x \in S$ and*
2. *$p(\cdot,B)$ defines a measurable function of $(S,\mathcal{B}(S))$ for all $B \in \mathcal{B}(S)$.*

*Furthermore, if $p(x,S) = 1$ for each $x \in S$, then p is called a **probability kernel**. If p and q are such sub-probability kernels, define*

$$(pq)(x,B) = \int p(x,dy)q(y,B)$$

for all $x \in S$ and all $B \in \mathcal{B}(S)$.

Using sub-probability kernels we can define transition functions.

Definition 2.1.7 (Transition function)**.** *A collection $(p_t)_{t\geq 0}$ of sub-probability kernels is called a **transition function**, if it satisfies the following properties:*

1. *$p_0(x,\cdot) = \delta_x(\cdot)$ for all x, and*
2. *$p_t p_s = p_{s+t}$ for all $0 \leq s$ and all $0 \leq t$ (**Chapman-Kolmogorov equation**).*

$(p_t)_{t\geq 0}$ is said to be the transition function of a time-homogeneous Markov process $(X_t)_{t\geq 0}$ on $(S,\mathcal{B}(S))$, if

$$p_t(x,B) = \mathbb{P}\left(X_t \in B | X_0 = x\right)$$

for all $t \geq 0$ and $B \in \mathcal{B}(S)$.

Note that the transition functions $(p_t)_{t\geq 0}$ act as an operator on measurable functions $f : S \to \mathbb{R}$ by

$$(p_t f)(x) = \int p_t(x,dy)f(y)$$

for all $x \in S$.

The existence of a transition function for every Markov process is a standard results. In the following remark, we will work out a proof.

Remark 2.1.8. *Every Markov process $(X_t)_{t\geq 0}$ has a transition function. In order to see this, define*

$$p_t : S \times \mathcal{B}(S) \to [0,1],\ (x,B) \mapsto \mathbb{P}(X_t \in B | X_0 = x)$$

and check that it is indeed a transition function. To do this, first note that it is a sub-probability kernel. Further,

$$p_0(x,B) = \mathbb{P}(X_0 \in B | X_0 = x) = \mathbb{1}_B(x).$$

Thus, $p_0(x,\cdot) = \delta_x(\cdot)$. Last but not least, we will check the Chapman-Kolmogorov equation. So, for all $x \in S$ and all $B \in \mathcal{B}(S)$, calculate:

$$\begin{aligned}
(p_t p_s)(x,B) &= \int p_t(x,dy) p_s(y,B) \\
&= \mathbb{E}[p_s(X_t,B)|X_0 = x] \\
&= \mathbb{E}[\mathbb{P}(X_s \in B|X_0 = X_t)|X_0 = x] \\
&= \mathbb{E}[\mathbb{P}(X_{s+t} \in B|X_t)|X_0 = x] \\
&= \mathbb{E}[\mathbb{E}[\mathbb{1}_{X_{s+t}\in B}|X_t]|X_0 = x] \\
&= \mathbb{E}[\mathbb{E}[\mathbb{1}_{X_{s+t}\in B}|\mathcal{F}_t]|X_0 = x] \\
&= \mathbb{E}[\mathbb{1}_{X_{s+t}\in B}|X_0 = x] \\
&= \mathbb{P}(X_{s+t} \in B|X_0 = x) \\
&= p_{s+t}(x,B),
\end{aligned}$$

where it was used that $(X_t)_{t\geq 0}$ is a Markov process and that $\{X_0 = x\} \in \mathcal{F}_0 \subset \mathcal{F}_t$.

Now that we know that every Markov process has a transition function, we calculate it for Brownian motion in the following example.

Example 2.1.9 (Transition function for Brownian motion)**.** *For standard linear Brownian motion $(B_t)_{t\geq 0}$ the transition function can be computed directly:*

$$\begin{aligned}
p_t(x,[a,b]) &= \mathbb{P}(B_{t+s} \in [a,b]|B_s = x) \\
&= \mathbb{P}(x + B_t \in [a,b]|B_0 = 0) \\
&= \mathbb{P}(B_t \in [x+a, x+b]) \\
&= \int_{x+a}^{x+b} \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{y^2}{t}\right) dy \\
&= \int_a^b \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{(x+y)^2}{2t}\right) dy
\end{aligned}$$

where $B_t \sim \mathcal{N}(0,t)$ was used. This defines the transition function fully, because the closed intervals generate $\mathcal{B}(\mathbb{R})$. This matches the result stated in [MP10, Example 2.31].

Often, instead of describing a Markov process explicitly using its transition function, one only gives its generator. This generator describes the infinitesimal properties of the process. Below, we define semigroups of operators which we can then use to define the generator.

Definition 2.1.10 (Semigroup of operators)**.** *Let $(P_t)_{t\geq 0}$ be linear operators on a Banach space $(\mathcal{A}, ||\cdot||_\infty)$. Then they define a semigroup if the* ***semigroup-property***

$$P_t P_s = P_{t+s}$$

holds for all $s,t \geq 0$. Furthermore, if in addition the ***sub-Markov-property***

$$0 \leq f \leq 1 \implies 0 \leq P_t f \leq 1$$

holds for all $t \in [0, \infty)$ and all $f \in \mathcal{A}$, they define a ***sub-Markovian semigroup****. The semigroup is called* ***strongly continuous*** *($\mathbf{C^0}$), if for all $f \in \mathcal{A}$,*

$$\lim_{t \downarrow 0} ||P_t f - f||_\infty \to 0$$

holds. If for all $t \geq 0$ and all bounded $f \in \mathcal{A}$,

$$||P_t f||_\infty \leq ||f||_\infty,$$

$(P_t)_{t \geq 0}$ is called a ***contraction semigroup****. These definitions can be found in [SP12, Chapter 7.1].*

As described in the lecture notes [Ebe23], the transition function of a Markov process induces such a semigroup. We worked out a similar argument in the following remark.

Remark 2.1.11. *Let $X = (X_t)_{t \geq 0}$ be a Markov process with state space $(S, \mathcal{B}(S))$ with transition function $(p_t)_{t \geq 0}$. Define the space*

$$E := \{f : S \to \mathbb{R} \mid f \text{ bounded and measurable, } \lim_{t \downarrow 0} ||P_t f - f||_\infty = 0\}.$$

Note that this is a Banach space.

Now, define the collection of operators $(P_t)_{t \geq 0}$ by $P_t f := p_t f$ for $f \in E$. We then have

$$(P_t f)(x) = (p_t f)(x) = \int p_t(x, dy) f(y) = \mathbb{E}\left[f(X_t) | X_0 = x\right]$$

for all $x \in S$, $t \geq 0$ and $f \in E$. Thus, $P_t f = \mathbb{E}\left[f(X_t)|X_0\right]$ for all $f \in E$. By the definition of conditional expectation, P_t is a linear operator for all $t \geq 0$. Furthermore, from the Chapman-Kolmogorov-equation, it is a semigroup. The sub-Markov property also follows directly from the properties of conditional expectations. By definition of E, the semigroup is strongly continuous on E.

Further, for every $f \in E$ and every $t \geq 0$, we calculate

$$||P_t f||_\infty = ||\mathbb{E}\left[f(X_t)|X_0\right]||_\infty \leq ||f(X_t)||_\infty \leq ||f||_\infty.$$

Here the first inequality is a well-known property of the conditional expectation. The semigroup is, thus, also a contraction.

This semigroup induced by a transition function is also called a ***transition semigroup****.*

We are now ready to define the generator of a Markov process.

Definition 2.1.12 (Generator)**.** *Let $P = (P_t)_{t \in [0, \infty)}$ be a C^0 contraction semigroup on a Banach space E. Then the linear operator $\mathcal{L}$ given by*

$$\mathcal{L} f = \lim_{t \downarrow 0} \frac{P_t f - f}{t}$$

(for all functions f such that the limit is finite and exists) is called the ***generator*** *of P. This definition is also taken from the lecture notes [Ebe23].*

We calculate such a generator for Brownian motion below.

Example 2.1.13 (Generator of Brownian motion)**.** *The generator of Brownian motion is given by*

$$(\mathcal{L}f)(x) = \frac{1}{2}f''(x)$$

for all $f \in C_b^2(\mathbb{R})$. Here, $C_b^2(\mathbb{R})$ denotes the set of all bounded and twice continuously differentiable functions on $\mathbb{R}$. This can be computed directly from the transition function:

$$\begin{aligned}
(\mathcal{L}f)(x) &= \lim_{t\downarrow 0} \frac{(p_t f)(x) - f(x)}{t} \\
&= \lim_{t\downarrow 0} \frac{\mathbb{E}[f(B_{t+s})|B_s = x] - f(x)}{t} \\
&= \lim_{t\downarrow 0} \mathbb{E}\left[\frac{f(B_{t+s}) - f(x)}{t}|B_s = x\right] \\
&= \lim_{t\downarrow 0} \mathbb{E}\left[\frac{f(B_t + x) - f(x)}{t}\right] \\
&= \lim_{t\downarrow 0} \mathbb{E}\left[\frac{f(\sqrt{t}Z + x) - f(x)}{t}\right] \\
&= \lim_{t\downarrow 0} \int_{-\infty}^{\infty} \frac{f(\sqrt{t}z + x) - f(x)}{\sqrt{2\pi t}} \exp\left(-\frac{z^2}{2}\right) dz,
\end{aligned}$$

where $Z \sim \mathcal{N}(0,1)$. A Taylor expansion gives:

$$f(x + \sqrt{t}z) = f(x) + f'(x)\sqrt{t}z + f''(x)tz^2 + t^{3/2}\mathcal{O}(z^3).$$

Thus,

$$\frac{f(\sqrt{t}z + x) - f(x)}{t} = \frac{f'(x)}{\sqrt{t}}z + f''(x)z^2 + \sqrt{t}\mathcal{O}(z^3).$$

Substituting this into the integral gives:

$$\begin{aligned}
(\mathcal{L}f)(x) &= \lim_{t\downarrow 0} \int_{-\infty}^{\infty} \frac{f(\sqrt{t}z + x) - f(x)}{\sqrt{2\pi t}} \exp\left(-\frac{z^2}{2}\right) dz \\
&= \lim_{t\downarrow 0} \int_{-\infty}^{\infty} \left(\frac{f'(x)}{\sqrt{t}}z + f''(x)z^2 + \sqrt{t}\mathcal{O}(z^3)\right) \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz \\
&= \lim_{t\downarrow 0} \left(\frac{f'(x)}{\sqrt{t}}\mathbb{E}[Z] + \frac{1}{2}f''(x)\mathbb{E}[Z^2] + \sqrt{t}\mathbb{E}[\mathcal{O}(Z^3)]\right) \\
&= \frac{1}{2}f''(x).
\end{aligned}$$

Here, the second step uses that the moments of the Gaussian random variable Z as found in [PR82]. The last step uses that

$$\mathbb{E}[\mathcal{O}(Z^3)] \leq \mathbb{E}[M|Z|^3] = M\mathbb{E}[|Z|^3] = M\sqrt{\frac{8}{\pi}} < \infty$$

for some positive M. This proof follows the argument given in [Var07, Chapter 5.6].

We have seen that for every Markov process we can define a transition semigroup and a generator. The question arises whether there is a one-to-one correspondence between Markov processes, transition semigroups and their generators. This is in general not the case. However, for a special class of Markov processes, called Feller processes, it is indeed the case. [Lig10, Chapter 3.1]

Definition 2.1.14 (Feller process)**.** *A **Feller process** is a right-continuous and time-homogeneous Markov process $(X_t)_{t\geq 0}$ with state space S such that the **Feller property***

$$x \mapsto \mathbb{E}[f(X_t)|X_0 = x] \text{ is continuous on } S \text{ for all } t \geq 0, f \in C(S),$$

holds [Lig10, Definition 3.1].

Example 2.1.15 (Brownian motion as a Feller process)**.** *Brownian motion is a Feller process as for example given in [Lig10, Chapter 3.1.3].*

In the following theorem we will give conditions under which there is a unique Feller process for a given a generator. This theorem summarises results from Chapters I.1 and I.2 of [Lig85] using the notation of introduced in this thesis.

Theorem 2.1.16 (Construction of Feller process from generator)**.** *Let S be a compact metric space and let $\mathcal{L}$ be a linear operator on $C(S)$ with domain $D(S)$ such that the following conditions hold:*

1. *The function $\mathbf{1} : S \to \mathbb{R}_+, x \mapsto 1$, is in the domain of $\mathcal{L}$.*
2. *$\mathcal{L}\mathbf{1} = \mathbf{0}$, where $\mathbf{0}$ is the function $S \to \mathbb{R}_+, x \mapsto 0$.*
3. *The domain of $\mathcal{L}$ is dense in $C(S)$.*
4. *For all $f \in D(S)$ and $\eta \in S$ such that f attains its minimum at η, we have that $(\mathcal{L}f)(\eta) = 0$.*

Furthermore, let $\mathcal{L}$ either be bounded or let the following condition hold in addition to the previous four:

5. *For all $\lambda \geq 0$, let the range of $I - \lambda\bar{\mathcal{L}}$ be equal to $C(S)$. Here, I denotes the identity operator and $\bar{\mathcal{L}}$ denotes the smallest extension of $\mathcal{L}$ such that the graph of $\mathcal{L}$ is a closed in $C(S) \times C(S)$.*

Then, the there is a unique Markov process $(X_t)_{t\geq 0}$ which has $\mathcal{L}$ as its generator in the bounded case and $\bar{\mathcal{L}}$ in the unbounded case. The transition semigroup $(P_t)_{t\geq 0}$ of the process is given by

$$(P_t f)(x) = \mathbb{E}\left[f(X_t)|X_0 = x\right]$$

for all $f \in C(S)$, $x \in S$ and $t \geq 0$.

Proof. First note that S is a Polish space by [Kec95, Proposition 4.2]. Therefore $C(\Omega)$ is a Banach space by [GvR16, Chapter 11]. Conditions (1.) through (4.) together with [Lig85, Chapter I.2, Proposition 2.2] show that $\mathcal{L}$ satisfies the definition of a Markov pregenerator as in [Lig85, Chapter I.2, Definition 2.1]. In the case that $\mathcal{L}$ is bounded, [Lig85, Chapter I.2, Proposition 2.8 (a)] implies that $\mathcal{L}$ is also a Markov generator as in [Lig85, Chapter I.2, Definition 2.7]. If instead condition (5.) holds, [Lig85, Chapter I.2, Proposition 2.8 (b)] together with [Lig85, Chapter I.2, Proposition 2.5] imply that $\mathcal{L}$ is a Markov generator.

The Hille-Yosida theorem (as given in [Lig85, Chapter I.2, Theorem 2.9 (Hille-Yosida)]) gives a one-to-one correspondence between Markov generators and Markov semigroups as defined in [Lig85, Chapter I.1, Theorem 1.4]. It furthermore states, that they fulfill our definition of a generator. The corresponding Markov semigroup $(P_t)_{t\geq 0}$ is given explicitly by

$$P_t f := \lim_{n\to\infty} \left(I - \frac{t}{n}\mathcal{L}\right)^{-n} f$$

for $f \in C(S)$ and $t \geq 0$.

According to [Lig85, Chapter I.1, Theorem 1.5], there is then a unique Markov process $(X_t)_{t\geq 0}$ with the property

$$(P_t f)(x) = \mathbb{E}\left[f(X_t)|X_0 = x\right]$$

for all $f \in C(S)$, $x \in S$ and $t \geq 0$. Thus, the semigroup is in fact the transition semigroup of the Markov process. □

2.2. Invariant Measures

In this section we will briefly introduce the concepts of invariant measures and ergodicity. We will restrict our attention to Feller processes on a compact metric space S. This makes dealing with the domain of the generator of the process easier. It is also the case that we will encounter in the opinion models studied in this project.

Definition 2.2.1 (Invariant measure)**.** *Let $(P_t)_{t\geq 0}$ be the transition semigroup of a Feller process on the compact metric space S. A probability measure μ on $(S, \mathcal{B}(S))$ is called* ***invariant*** *with respect to this transition semigroup, if*

$$\mu P_t = \mu$$

for all $t \geq 0$. [Lig85, Chapter I.1, Definition 1.7]

Let us consider some equivalent definitions of invariant measures in the following proposition.

Proposition 2.2.2. *Let μ be a probability measure on a compact metric space S endowed with the Borel-sigma algebra. Let $(P_t)_{t\geq 0}$ be a transition semigroup induced by a Feller process on S with generator L. Then the following are equivalent:*

1. *μ is invariant with respect to $(P_t)_{t\geq 0}$.*
2. *$\int P_t f d\mu = \int f d\mu$ for all $f \in C(S)$ and $t \geq 0$.*
3. *$\int \mathcal{L} f d\mu = 0$ for all functions f in the domain of $\mathcal{L}$.*

Proof. The equivalence of (1.) and (2.) is taken from [Lig85, Chapter I.1, Proposition 1.8 (a)]. In order to see this equivalence, note that due to the Feller property both μ and μP_t are probability measures defined on $C(S)$. Thus, the equivalence simply follows by re-writing

$$\mu(f) = \int f d\mu$$

and

$$(\mu P_t)(f) = \int P_t f d\mu$$

for all $f \in C(S)$ and $t \geq 0$.

The equivalence of (1.) and (3.) is proven in [Lig10, Theorem 3.37]. □

The following proposition shows that if the process converges to a probability measure, this measure needs to be invariant. This is a statement that we will use in Part II of this project.

Proposition 2.2.3. *Let $(P_t)_{t\geq 0}$ a transition semigroup induced by a Feller process on the compact metric space S endowed by the Borel-sigma algebra and let μ be a probability measure on S. If $\lim_{t\to\infty} \mu P_t$ exists and is also a probability measure on S, then it is an invariant measure.*

Proof. A detailed proof can be found in [Lig85, Chapter I.1, Proposition 1.8 (d)]. □

Part II.

Theoretical Studies

3. Deffuant Model

In this section we will formally introduce the Deffuant Model as a Markov process. As mentioned in the introduction, this model has been studied by various people. We are mainly interested in it, because the main model that we want to study, the Uniform Compass model, is a variation on it [GHH20, Chapter 1]. We start by giving a description of the Deffuant model on general graphs in Section 3.1. We then define a special case of this model, the Uniform Deffuant model on finite paths in Section 3.2. This is the version of the model that we mainly studied in this project. We will therefore also give a detailed proof that it is a Markov process and even a Feller process. In Section 3.3, we will define two variations of the Uniform Deffuant model that model noise. The types of noise introduced in this section are the same as the ones that we will study in the numerical part of the project. We will also show that the processes with noise are still Markov.

3.1. Description of the Deffuant Model

The Deffuant model is a type of bounded confidence model. It can be described by a Markov process. Intuitively, it describes how the opinions of the members of a group evolve. The individuals and their connections are modelled as an undirected graph. The individuals are represented by the vertices and individuals can only interact with each other if they are connected by an edge. Opinions are mostly considered to be elements of the interval $[0, 1]$ with the standard Euclidean metric. Each individual interacts with another individual at random times. When two individuals interact and the distance between their opinions is larger than the confidence bound given by a parameter ψ, the individuals do not change their original opinions. If the opinions are closer than ψ, both of them alter their respective opinion in order to be closer to a consensus. How much they alter their opinion is defined by a parameter μ. A formal definition is given in Definition 3.1.2. We will use (variations of) the definitions given in [GHH20, Chapter 1].

Definition 3.1.1 (Uniform initial condition)**.** *Let $(\eta_t)_{t\geq 0}$ be a Markov process with a state space denoted by $\Omega = S^V$. We speak of the process having* ***uniform initial conditions*** *when $\eta_0(v) \sim \text{Uniform}(S)$ and $\eta_0(v)$ and $\eta_0(u)$ are independent for all individuals $u, v \in V$.*

Definition 3.1.2 (Definition of the Deffuant model)**.** *Let $G = (V, E)$ be a locally finite connected undirected graph, where V denotes the vertices and E the edges. Let the* ***opinion space*** *be given by a compact and convex space S endowed with the metric*

d. Let the parameter μ *be taken from* $(0, 0.5]$ *and the* ***confidence bound*** ψ*, such that* $\psi \in [0, \infty]$*. Let the* ***state space*** *be given by* $\Omega = S^V$ *together with the product topology. Let* $(\eta_t)_{t\geq 0}$ *be a Markov process with probability generator* $\mathcal{L}$*, given by*

$$\mathcal{L}f(\eta) = \sum_{e\in E}(f(A_e\eta) - f(\eta))$$

with

$$A_e\eta(v) := \begin{cases} \eta(v), & \text{if } v \notin e \\ \eta(v) + \mu\mathbb{1}_{\{d(\eta(v),\eta(u))\leq\psi\}}(\eta(u) - \eta(v)), & \text{if } e = \langle u, v\rangle \in E \end{cases}$$

for continuous test-functions f*. We call this process the* ***Deffuant model****. When* $\psi = \infty$*, we speak of* ***unbounded confidence****. When using uniform initial conditions, we speak of the* ***Uniform Deffuant model****.*

Intuitively, one can think of this system in the following way. On each vertex, let there be an individual with an opinion in S. Two individuals can only interact with each other when their positions are connected by an edge. The interactions between different pairs of individuals are all independent. Also the waiting times between two interactions at the same edge are always distributed exponentially with parameter 1. These descriptions will allow us to simulate the process numerically in Part III.

Showing that the Deffuant model is indeed well-defined is tricky in the general case. According to [GHH20, Chapter 1] it is, however, a rather standard result. We will prove it in detail for the special case introduced in the next section.

3.2. Deffuant Model on Finite Paths

In this section we will consider the special case of the Deffuant model on finite paths with infinite confidence bound. A formal definition of a finite path is given in the following definition.

Definition 3.2.1 (Finite path)**.** *Let* $P_n := (V_n, E_n)$ *be a simple undirected graph, where the vertices are given by*

$$V_n := \{1, 2, \ldots, n\}$$

and the edges by

$$E_n := \{\langle 1, 2\rangle, \langle 2, 3\rangle, \ldots, \langle n-1, n\rangle\}.$$

We call P_n *the* ***finite path*** *of length* n*.*

We then define the Deffuant model on finite paths of unbounded confidence as follows.

Definition 3.2.2 (Deffuant model on a finite path)**.** *Let* $P_n = (V_n, E_n)$ *be the finite path of length* n *as defined in Definition 3.2.1. Define the* ***opinion space*** *by the compact and convex space* $[0, 1]$ *endowed with the standard Euclidean metric* d*. Let* $\mu \in (0, 0.5]$ *and the* ***state space*** *be given by* $\Omega = [0, 1]^V$ *together with the product topology. Denote*

the set of continuous functions $\Omega \to \mathbb{R}$ by $C(\Omega)$. Let $(\eta_t)_{t\geq 0}$ be a Feller process with probability generator $\mathcal{L}$, given by

$$\mathcal{L}f(\eta) = \sum_{i=1}^{n-1} \left(f\left(A_{\langle i,i+1\rangle}\eta\right) - f\left(\eta\right)\right)$$

with

$$A_{\langle i,i+1\rangle}\eta(m) = \begin{cases} (1-\mu)\eta(i) + \mu\eta(i+1), & \text{if } m = i \\ (1-\mu)\eta(i+1) + \mu\eta(i), & \text{if } m = i+1 \\ \eta(m), & \text{else} \end{cases}$$

*for every $i \in V_n\backslash\{n\}$, $m \in V_n$, $\eta \in \Omega$ and $f \in C(\Omega)$. We call this process the **Deffuant model on the finite path**. When the initial configuration is the uniform distribution on V_n, we speak of the **Uniform Deffuant model on the finite path**.*

In order to show that this model is well-defined, we need to show that the probability generator $\mathcal{L}$ is the generator of a well-defined Feller process. We will do this by first showing the conditions given in Theorem 2.1.16 and then showing the Feller property.

Theorem 3.2.3 ($\mathcal{L}$ defines a Feller process)**.** *There exists exactly one Feller process such that the operator $\mathcal{L}$ as defined above is its generator.*

We start by showing a couple of lemmas first and then prove the theorem.

Lemma 3.2.4. *$\mathcal{L}$ is a linear operator.*

Proof. Let $f, g \in D(\mathcal{L})$ and $\alpha \in \mathbb{R}$. For all $\eta \in \Omega$, one has:

$$\begin{aligned} \mathcal{L}(f+g)(\eta) &= \sum_{i=1}^{n-1} \left((f+g)(A_{\langle i,i+1\rangle}\eta) - (f+g)(\eta)\right) \\ &= \sum_{i=1}^{n-1} \left[\left(f(A_{\langle i,i+1\rangle}\eta) - f(\eta)\right) + \left(g(A_{\langle i,i+1\rangle}\eta) - g(\eta)\right)\right] \\ &= \sum_{i=1}^{n-1} \left(f(A_{\langle i,i+1\rangle}\eta) - f(\eta)\right) + \sum_{i=1}^{n-1} \left(g(A_{\langle i,i+1\rangle}\eta) - g(\eta)\right) \\ &= \mathcal{L}f(\eta) + \mathcal{L}g(\eta) \end{aligned}$$

and

$$\begin{aligned} \mathcal{L}(\alpha f)(\eta) &= \sum_{i=1}^{n-1} \left((\alpha f)(A_{\langle i,i+1\rangle}\eta) - (\alpha f)(\eta)\right) \\ &= \sum_{i=1}^{n-1} \left(\alpha \cdot f(A_{\langle i,i+1\rangle}\eta) - \alpha \cdot f(\eta)\right) \\ &= \alpha \sum_{i=1}^{n-1} \left(f(A_{\langle i,i+1\rangle}\eta) - f(\eta)\right) = \alpha\mathcal{L}f(\eta). \end{aligned}$$

Thus, $\mathcal{L}$ is linear. □

Lemma 3.2.5. *Let $f \in D(\mathcal{L})$ and $\eta \in \Omega$ such that $f(\eta) = \min_{\xi\in\Omega} f(\xi)$. Then $\mathcal{L}f(\eta) \geq 0$.*

Proof. Let $f \in D(\mathcal{L})$. Because f is a continuous function on a compact domain, there exists $\eta \in \Omega$ such that $f(\eta) = \min_{\xi\in\Omega} f(\xi)$. Thus, $f(\xi) - f(\eta) \geq 0$ for all $\xi \in \Omega$. Thus, $f(A_{\langle i,i+1\rangle}\eta) - f(\eta) \geq 0$ for all $i \in \{1, 2, \dots n-1\}$. Thus, $\mathcal{L}f(\eta) \geq 0$. □

Lemma 3.2.6. *Let $\mathcal{L}$ and Ω be as defined in Definition 3.2.2. Then $D(\mathcal{L}) = C(\Omega)$.*

Proof. $D(\mathcal{L}) = \{f \in C(\Omega) : ||\mathcal{L}f|| < \infty\}$. Let $f \in C(\Omega)$. Then

$$\begin{aligned}
||\mathcal{L}f|| &= \sup_{\eta\in\Omega} |\mathcal{L}f(\eta)| \\
&= \sup_{\eta\in\Omega} \left| \sum_{i=1}^{n-1} \big(f(A_{\langle i,i+1\rangle}\eta) - f(\eta)\big) \right| \\
&\leq \sup_{\eta\in\Omega} \left(\sum_{i=1}^{n-1} |f(A_{\langle i,i+1\rangle}\eta)| + (n-1)|f(\eta)| \right) \\
&= \sum_{i=1}^{n-1} \sup_{\eta\in\Omega} |f(A_{\langle i,i+1\rangle}\eta)| + (n-1) \sup_{\eta\in\Omega} |f(\eta)| \\
&\leq \sum_{i=1}^{n-1} \sup_{\eta\in\Omega} |f(\eta)| + (n-1) \sup_{\eta\in\Omega} |f(\eta)| \\
&= 2(n-1)||f||.
\end{aligned}$$

Because f is continuous and Ω is compact, $||f|| < \infty$. Thus, $D(\mathcal{L}) = C(\Omega)$. □

Lemma 3.2.7. *The generator $\mathcal{L}$ from Definition 3.2.2 is bounded.*

Proof. As seen in the proof of Lemma 3.2.6, $||\mathcal{L}f|| \leq 2(n-1)||f||$ for all $f \in C(\Omega)$. Thus, $||\mathcal{L}|| \leq 2(n-1) < \infty$, which implies that $\mathcal{L}$ is bounded. □

We are now ready to prove Theorem 3.2.3, which states that the generator $\mathcal{L}$ gives a well-defined and unique Feller process.

Proof of Theorem 3.2.3. First note that $[0, 1]$ is compact, because it is a closed and finite interval. Thus, we can apply Tychonoff's Theorem (see for example [Kec95, Proposition 4.1(vi)]) to show that Ω is also compact. Furthermore, $([0, 1], d)$ is a metric space. According to Lemma 3.2.4, $\mathcal{L}$ is a linear operator.

We first show conditions (1.) to (4.) from Theorem 2.1.16. Condition (1.) follows directly from Lemma 3.2.6. To show condition (2.), let $\mathbf{1}$ denote the function $\Omega \to \mathbb{R}$ given by $\mathbf{1}(\eta) = 1$ for all $\eta \in \Omega$. Then $(\mathcal{L}\mathbf{1})(\eta) = \sum_{i=1}^{n-1}(1-1) = 0$ for all $\eta \in \Omega$. Therefore, condition (2.) is fulfilled. According to Lemma 3.2.6, $D(\mathcal{L}) = C(\Omega)$. Thus, $D(\mathcal{L})$ is also dense in $C(\Omega)$ giving condition (3.). By Lemma 3.2.5 condition (4.) is satisfied.

By Lemma 3.2.7, we know that $\mathcal{L}$ is bounded. Therefore, Theorem 2.1.16 applies and shows that there indeed exist a unique Markov process $(X_t)_{t\geq 0}$ which has $\mathcal{L}$ as its generator. The transition semigroup $(P_t)_{t\geq 0}$ of the process is given by

$$(P_t f)(x) = \mathbb{E}\left[f(X_t)|X_0 = x\right]$$

for all $f \in C(S)$, $x \in S$ and $t \geq 0$.

According to [Lig85, Chapter I.2, Theorem 2.9 (Hille-Yosida) (c)], $P_t f \in D(\Omega)$ for all $f \in D(\Omega)$. Because of Lemma 3.2.6, this implies that $x \mapsto \mathbb{E}\left[f(X_t)|X_0 = x\right]$ is continuous on S for all $t \geq 0$ and $f \in C(S)$. Therefore, the process is also Feller. □

Note that for the case of the Deffuant model on a general graph G, the generator $\mathcal{L}$ is not necessarily bounded. This means that different arguments are needed in that case.

3.3. Deffuant Model with Noise

Now that we have properly defined the Deffuant model on finite paths, we are interested in variations of it that take into account the effects of noise. In particular, we will look at uniform noise in Section 3.3.1 and bi-modal noise in Section 3.3.2. For both versions, we will define the new model as a Markov process.

3.3.1. Uniform Noise

In this section we look at one of the most straightforward ways to define noise. At every time that individuals interact, we simply draw a number from a uniform distribution Uniform$(-\epsilon, \epsilon)$ for an $\epsilon > 0$ and add it to an individual selected uniformly at random. A precise definition follows.

Definition 3.3.1 (Deffuant model with uniform noise)**.** *Denote the finite path of length n as defined in Definition 3.2.1 by $P_n = (V_n, E_n)$. Define the* ***opinion space*** *as the compact and convex space $[0,1]$ endowed with the standard Euclidean metric d. Let $\mu \in (0, 0.5]$ and $\epsilon \geq 0$. Furthermore, define the* ***state space*** *by $\Omega = [0,1]^V$ together with the product topology. Let $C(\Omega)$ denote the set of continuous functions $\Omega \to \mathbb{R}$ and ξ be a random variable distributed as* Uniform$(-\epsilon, \epsilon)$*. Let $(\eta_t)_{t\geq 0}$ be a Feller process with probability generator $\mathcal{L}$, given by*

$$\mathcal{L}f(\eta) := \sum_{i=1}^{n-1} \left(f(A_{\langle i,i+1\rangle}\eta) - f(\eta)\right) + \sum_{i=1}^{n} \left(f(B_i\eta) - f(\eta)\right)$$

with

$$A_{\langle i,i+1\rangle}\eta(m) = \begin{cases} (1-\mu)\eta(i) + \mu\eta(i+1), & \text{if } m = i, \\ (1-\mu)\eta(i+1) + \mu\eta(i), & \text{if } m = i+1, \\ \eta(m), & \text{else} \end{cases}$$

and

$$B_i\eta(m) = \begin{cases} \eta(m), & \text{if } m \neq i, \\ \eta(m) + \xi, & \text{if } m = i \end{cases}$$

for every $i \in V_n \backslash \{n\}$, $m \in V_n$, $\eta \in \Omega$ and $f \in C(\Omega)$. We call this process the ***Deffuant model with uniform noise****. When the initial configuration is the uniform distribution on V_n, we speak of the* ***Uniform Deffuant model with uniform noise****.*

We can prove that this is in fact well-defined, by showing that $\mathcal{L}$ as defined above corresponds one-to-one to a Feller process as shown in the following corollary.

Corollary 3.3.2 (Deffuant model with uniform noise is a Feller process)**.** *There exists a well-defined and unique Feller process which has the generator from Definition 3.3.1.*

Proof. This proof is completely analogous to that of Theorem 3.2.3. □

3.3.2. Bi-Modal Noise

In this section we look at another type of noise that we call bi-modal noise. It is similar to the noise considered in the paper [CR16]. This type of noise pushes opinions of individuals towards the opinions 0 and 1. At every time when people interact, one individual x is chosen from the set of all individuals uniformly at random. Then an opinion a is chosen either from 0 or 1 uniformly at random. The opinion of the chosen individual x is then perturbed towards the chosen opinion a. The farther away the opinion of x is from a, the more it gets changed.

The noise is quantified by a parameter ϵ. The opinion of x gets changed in the same way that the dynamics of the Deffuant Model would change it when it would react with a neighbour with opinion a and if μ where ϵ. A formal definition is given below.

Definition 3.3.3 (Deffuant model with bi-modal noise)**.** *Denote the finite path of length n as defined in Definition 3.2.1 by $P_n = (V_n, E_n)$. Define the* ***opinion space*** *to be the compact and convex space $[0, 1]$ endowed with the standard Euclidean metric d. Let the parameter $\mu \in (0, 0.5]$, $\epsilon \geq 0$ and let the* ***state space*** *be given by $\Omega = [0, 1]^V$ together with the product topology. Denote the set of continuous functions $\Omega \to \mathbb{R}$ by $C(\Omega)$. Let $(\eta_t)_{t\geq 0}$ be a Feller process with probability generator $\mathcal{L}$, given by*

$$\mathcal{L}f(\eta) := \sum_{i=1}^{n-1} \big(f(A_{\langle i,i+1\rangle}\eta) - f(\eta)\big) + \sum_{i=1}^{n} \left(\frac{1}{2}\left(f(B_i\eta) + f(C_i\eta)\right) - f(\eta)\right)$$

with

$$A_{\langle i,i+1\rangle}\eta(m) = \begin{cases} (1-\mu)\eta(i) + \mu\eta(i+1), & \text{if } m = i \\ (1-\mu)\eta(i+1) + \mu\eta(i), & \text{if } m = i+1 \\ \eta(m), & \text{else} \end{cases}$$

and

$$B_i\eta(m) = \begin{cases} \eta(m), & \text{if } m \neq i, \\ (1-\epsilon)\eta(m), & \text{if } m = i \end{cases}$$

and

$$C_i\eta(m) = \begin{cases} \eta(m), & \text{if } m \neq i, \\ (1-\epsilon)\eta(m) + \epsilon, & \text{if } m = i. \end{cases}$$

for every $i \in V_n \backslash \{n\}$, $m \in V_n$, $\eta \in \Omega$ and $f \in C(\Omega)$. We call this process the ***Deffuant model with bi-modal noise****. When the initial configuration is the uniform distribution on V_n, we speak of the* ***Uniform Deffuant model with bi-modal noise****.*

Note that in the above definition, B_i stands for the case where $a = 0$ and C_i encodes the case where $a = 1$.

The proof that this generator defines a Markov process is analogous to the case without noise.

Corollary 3.3.4 (Deffuant model with bi-modal noise is a Feller process)**.** *There exists a well-defined and unique Feller process which has the generator from Definition 3.3.3.*

Proof. This proof follows fully analogously to the case of the Deffuant model without noise as shown in Theorem 3.2.3, because of the very similar generators. □

4. Compass Model

In this chapter we define the Compass model, which was first introduced by N. Gantert, M. Heydenreich and T. Hirscher in their paper "Strictly weak consensus in the uniform compass model on $\mathbb{Z}$" from 2020 [GHH20]. This model was introduced as a variation of the Deffuant model with different opinion space. In particular, the condition that the opinion space is convex will be dropped. In Section 4.1, we give a formal description of the Compass model as a Markov process as given in [GHH20, Chapter 1]. In Section 4.2, we define a special case of the model and present a proof that it is indeed a Markov process, which we have worked out for this specific model using methods outlined in [Lig85, Chapter 1]. In the last section of this chapter, we will describe versions of the model with noise.

4.1. Description of the Compass Model

The Compass model is similar to the Deffuant model, but with a few major differences. In particular, it does not assume that S is convex, only that it is path connected. We define the distance on S to be the length of the shortest path between two points. The interactions are then analogous to those of the Deffuant model.

Definition 4.1.1 (Compass model)**.** *Let $G = (V, E)$ be a locally finite connected undirected graph, where V denotes the set of its vertices, and E its edges. Let μ lie in the interval $(0, 0.5]$. Define the* ***opinion space*** *as $S := \mathbb{R}/2\mathbb{Z}$ written as $(-1, 1]$, for simplicity. The distance d is given as the length of the shortest path in S :*

$$d : S \times S \to [0, 1],\ (x, y) \mapsto \min\{|x - y|, 2 - |x - y|\}.$$

Let the ***state space*** *be $\Omega := S^{V_n}$ endowed with the product topology. Define a probability generator $\mathcal{L}$ for $\eta \in \Omega$ and a continuous test-function f by*

$$\mathcal{L}f(\eta) := \sum_{e \in E} \left(\frac{1}{2} \left[f(A_e^{(1)}\eta) + f(A_e^{(2)}\eta) \right] - f(\eta) \right)$$

with

$$A_e^{(k)}\eta(v) = \begin{cases} \eta(v), & \text{if } v \in e \\ \eta(v) + \mu(\eta(u) - \nu(v)), & \text{if } e = \langle u, v \rangle, |\eta(u) - \eta(v)| < 1 \\ \eta(v) + \mu(2 - |\eta(u) - \nu(v)|) \cdot \operatorname{sgn}(\nu(v)) \mod S, & \text{if } e = \langle u, v \rangle, |\eta(u) - \eta(v)| > 1 \\ \eta(v) + (-1)^k \cdot \operatorname{sgn}(\nu(v)) \mod S, & \text{if } e = \langle u, v \rangle, |\eta(u) - \eta(v)| = 1 \end{cases}$$

where $k \in \{1,2\}$. The Markov process $(\eta_t)_{t\geq 0}$ defined by this generator is called the ***Compass model****. If the initial configuration is the uniform initial condition, we speak of the* ***Uniform Compass model****. [GHH20, Chapter 1]*

Note that the choice of $\frac{1}{2}\left[f(A_e^{(1)}\eta) + f(A_e^{(2)}\eta)\right]$ ensures that

1. opinions always update along the shortest connecting path on the circle, and
2. when two opinions have distance 1, the path along which they are updated is chosen randomly with equal probability.

As explained for the Deffuant model, we can again describe the system by saying that at rate 1 individuals connected by an edge react and that interactions between different pairs of individuals are independent.

According to [GHH20, Chapter 1], this is not a Feller process, but still a Markov process. In the special case that we will consider in the next section, however, it turns out to be a Feller process.

4.2. Compass Model on Finite Paths

In this section we consider the Compass model on finite paths and show that this is a well-defined Feller process. The formal definition is given below.

Definition 4.2.1 (Compass model on finite path)**.** *Let $P_n = (V_n, E_n)$ be the finite path of length $n \in \mathbb{N}$ as defined in Definition 3.2.1. Define the* ***opinion space*** *as $S := \mathbb{R}/2\mathbb{Z}$ using the distance d is given by*

$$d : S \times S \to [0,1],\ (x,y) \mapsto \min\{|x-y|, 2-|x-y|\}.$$

Let the parameter μ be a value in $[0, 0.5]$. For continuous test-functions $f : \Omega \to \mathbb{R}$ and $\eta \in \Omega$, define a generator $\mathcal{L}$ by

$$\mathcal{L}f(\eta) = \sum_{i=1}^{n-1} \left(\frac{1}{2}\left[f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right) + f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right] - f(\eta)\right).$$

Here, we used

$$A^{(k)}_{\langle i,i+1\rangle}\eta(m) =$$

$$\begin{cases}
\eta(i+1) + \mu(\eta(i) - \eta(i+1)), & \text{if } m = i+1, |\eta(i) - \eta(i+1)| < 1,\\
\eta(i+1) + \mu(2 - |\eta(i) - \eta(i+1)|) \cdot \operatorname{sgn}(\eta(i+1)) \mod S, & \text{if } m = i+1, |\eta(i) - \eta(i+1)| < 1,\\
\eta(i+1) + (-1)^k \mu \cdot \operatorname{sgn}(\eta(i+1)) \mod S, & \text{if } m = i+1, |\eta(i) - \eta(i+1)| = 1,\\
\eta(i) + \mu(\eta(i+1) - \eta(i)), & \text{if } m = i, |\eta(i) - \eta(i+1)| < 1,\\
\eta(i) + \mu(2 - |\eta(i) - \eta(i+1)|) \cdot \operatorname{sgn}(\eta(i)) \mod S, & \text{if } m = i, |\eta(i) - \eta(i+1)| > 1,\\
\eta(i) + (-1)^k \mu \cdot \operatorname{sgn}(\eta(i)) \mod S, & \text{if } m = i, |\eta(i) - \eta(i+1)| = 1,\\
\eta(m), & \text{else.}
\end{cases}$$

for $i \in \{1, \dots, n-1\}$, $m \in V_n$, and $k \in \{1, 2\}$. We then call the Feller process $(\eta_t)_{t \geq 0}$ given by $\mathcal{L}$ the ***Compass model on the finite path****. When using a uniform initial condition, we speak of the* ***Uniform Compass model on the finite path****.*

For the remainder of the section, our goal is to show that Compass model on finite paths is well-defined. We will proceed analogously to Section 3.2.

Theorem 4.2.2 ($\mathcal{L}$ defines a Feller process)**.** *There exists exactly one Feller process such that the operator $\mathcal{L}$ as defined above is its generator.*

Before proving this theorem, we need to show some lemmas.

Lemma 4.2.3. *The operator $\mathcal{L}$ is linear.*

Proof. Let $f, g \in D(\mathcal{L})$ and $\alpha \in \mathbb{R}$. For all $\eta \in \Omega$, we have that

$$
\begin{aligned}
&\mathcal{L}(f+g)\eta \\
&= \sum_{i=1}^{n-1} \left(\frac{1}{2} \left[(f+g) \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + (f+g) \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - (f+g)(\eta) \right) \\
&= \sum_{i=1}^{n-1} \left(\left(\frac{1}{2} \left[f \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + f \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - f(\eta) \right) + \left(\frac{1}{2} \left[g \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + g \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - g(\eta) \right) \right) \\
&= \sum_{i=1}^{n-1} \left(\frac{1}{2} \left[f \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + f \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - f(\eta) \right) + \sum_{i=1}^{n-1} \left(\frac{1}{2} \left[g \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + g \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - g(\eta) \right) \\
&= \mathcal{L}f(\eta) + \mathcal{L}g(\eta)
\end{aligned}
$$

and

$$
\begin{aligned}
\mathcal{L}(\alpha f)(\eta) &= \sum_{i=1}^{n-1} \left(\frac{1}{2} \left[(\alpha f) \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + (\alpha f) \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - (\alpha f)(\eta) \right) \\
&= \sum_{i=1}^{n-1} \left(\frac{1}{2} \left[\alpha \cdot f \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + \alpha \cdot f \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - \alpha \cdot f(\eta) \right) \\
&= \alpha \sum_{i=1}^{n-1} \left(\frac{1}{2} \left[f \left(A^{(1)}_{\langle i,i+1 \rangle} \eta \right) + f \left(A^{(2)}_{\langle i,i+1 \rangle} \eta \right) \right] - f(\eta) \right) \\
&= \alpha \mathcal{L}f(\eta).
\end{aligned}
$$

Thus, $\mathcal{L}$ is linear. □

Lemma 4.2.4. *Let $\mathbf{1} : \Omega \to \mathbb{R}$ denote the function given by $\mathbf{1}(\eta) = 1$ for all $\eta \in \Omega$. Let $\mathbf{0} : \Omega \to \mathbb{R}$ denote the function given by $\mathbf{0}(\eta) = 1$ for all $\eta \in \Omega$. Then $\mathcal{L}\mathbf{1} = \mathbf{0}$.*

Proof. For every $\eta \in \Omega$ we have that

$$
\mathcal{L}\mathbf{1}(\eta) = \sum_{i=1}^{n-1} \left(\frac{1}{2} [1+1] - 1 \right) = 0.
$$

□

Lemma 4.2.5. *Let $f \in D(\mathcal{L})$. Then there exists $\eta \in \Omega$ such that $f(\eta) = \min_{\xi\in\Omega} f(\xi)$. For this η it holds that $\mathcal{L}f(\eta) \geq 0$.*

Proof. Let $f \in D(\mathcal{L}) \subset C(\Omega)$. Because f is a continuous function on a compact domain, it takes a minimum. Therefore, there is an $\eta \in \Omega$ such that $f(\eta) = \min_{\xi\in\Omega} f(\xi)$. By the definition of the minimum, we have that $f(\xi) \geq f(\eta)$ for all $\xi \in \Omega$. This implies that also $f(A^{(k)}_{\langle i,i+1\rangle}\eta) \geq f(\eta)$ for all $i \in \{1,\dots,n-1\}$ and $k \in \{1,2\}$. Thus,

$$0 \leq \mathcal{L}f(\eta) = \sum_{i=1}^{n-1}\left(\frac{1}{2}\left[f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right) + f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right] - f(\eta)\right) < \infty.$$

□

Lemma 4.2.6. *The domain of the operator $\mathcal{L}$ is the set of continuous functions on its state space Ω, that is $D(\mathcal{L}) = C(\Omega)$.*

Proof. Recall that $D(\mathcal{L}) = \{f \in C(\Omega) \mid ||\mathcal{L}f|| < \infty\}$, where the norm $||\cdot||$ denotes the supremum norm. Thus, by definition $D(\mathcal{L}) \subset C(\Omega)$. It therefore remains to show that $C(\Omega) \subset D(\mathcal{L})$, that is, that for every continuous function on Ω we have $||\mathcal{L}f|| < \infty$. Let therefore $f \in C(\Omega)$. Then

$$\begin{aligned}
||\mathcal{L}f|| &= \sup_{\eta\in\Omega}|\mathcal{L}f(\eta)| \\
&= \sup_{\eta\in\Omega}\left|\sum_{i=1}^{n-1}\left(\frac{1}{2}\left[f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right) + f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right] - f(\eta)\right)\right| \\
&\leq \sup_{\eta\in\Omega}\sum_{i=1}^{n-1}\left(\frac{1}{2}\left[\left|f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right)\right| + \left|f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right|\right] + |f(\eta)|\right) \\
&\leq \sup_{\eta\in\Omega}\left[\sum_{i=1}^{n-1}\frac{1}{2}\left|f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right)\right| + \sum_{i=1}^{n-1}\frac{1}{2}\left|f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right| + \sum_{i=1}^{n-1}|f(\eta)|\right] \\
&\leq \sup_{\eta\in\Omega}\sum_{i=1}^{n-1}\frac{1}{2}\left|f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right)\right| + \sup_{\eta\in\Omega}\sum_{i=1}^{n-1}\frac{1}{2}\left|f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right| + \sup_{\eta\in\Omega}\sum_{i=1}^{n-1}|f(\eta)| \\
&\leq \sum_{i=1}^{n-1}\sup_{\eta\in\Omega}\frac{1}{2}\left|f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right)\right| + \sum_{i=1}^{n-1}\sup_{\eta\in\Omega}\frac{1}{2}\left|f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right| + \sum_{i=1}^{n-1}\sup_{\eta\in\Omega}|f(\eta)| \\
&\leq 2\sum_{i=1}^{n-1}\sup_{\eta\in\Omega}|f(\eta)| \leq 2(n-1)||f||.
\end{aligned}$$

Because f is continuous and defined on a compact domain, $||f|| < \infty$.
Thus, $||\mathcal{L}f|| \leq 2(n-1)||f|| < \infty$. This concludes the proof. □

Lemma 4.2.7. *The operator $\mathcal{L}$ is bounded.*

Proof. As seen in the proof of Lemma 4.2.6, for every $f \in D(\mathcal{L})$ we have that $||\mathcal{L}f|| \leq 2(n-1)||f||$. Thus, $\mathcal{L}$ is bounded by $2(n-1)$ under the operator norm. □

Now that we have shown that $\mathcal{L}$ is indeed a Markov generator, we can go on to prove Theorem 4.2.2. This proof is completely analogous to that of Theorem 3.2.3.

Proof of Theorem 4.2.2. Note that S is homeomorphic to $S^1 = \{(x, y) \in \mathbb{R} | x^2 + y^2 = 1\}$. Thus, compactness of S follows directly from compactness of S^1. Compactness of Ω then follows from Tychonoff's theorem [Kec95, Proposition 4.1(vi)]. Furthermore, (S, d) is a metric space. According to Lemma 3.2.4, $\mathcal{L}$ is a linear operator.

We first show conditions (1.) to (4.) from Theorem 2.1.16. Condition (1.) follows directly from Lemma 4.2.6. To show condition (2.), let $\mathbf{1}$ denote the function $\Omega \to \mathbb{R}$ given by $\mathbf{1}(\eta) = 1$ for all $\eta \in \Omega$. Then $(\mathcal{L}\mathbf{1})(\eta) = \sum_{i=1}^{n-1}(1-1) = 0$ for all $\eta \in \Omega$. Therefore, condition (2.) is fulfilled. According to Lemma 4.2.6, $D(\mathcal{L}) = C(\Omega)$. Thus, $D(\mathcal{L})$ is also dense in $C(\Omega)$ giving condition (3.). By Lemma 4.2.5 condition (4.) is satisfied.

By Lemma 4.2.7, we know that $\mathcal{L}$ is bounded. Therefore, Theorem 2.1.16 applies and shows that there indeed exist a unique Markov process $(X_t)_{t\geq 0}$ which has $\mathcal{L}$ as its generator. The transition semigroup $(P_t)_{t\geq 0}$ of the process is given by

$$(P_t f)(x) = \mathbb{E}\left[f(X_t) | X_0 = x\right]$$

for all $f \in C(S)$, $x \in S$ and $t \geq 0$.

According to [Lig85, Chapter I.2, Theorem 2.9 (Hille-Yosida) (c)], $P_t f \in D(\Omega)$ for all $f \in D(\Omega)$. Because of Lemma 4.2.6, this implies that $x \mapsto \mathbb{E}\left[f(X_t) | X_0 = x\right]$ is continuous on S for all $t \geq 0$ and $f \in C(S)$. Therefore, the process is also Feller. □

Note that for the case of the Compass model on a general graph G, the generator $\mathcal{L}$ is not necessarily bounded. This means that different arguments are needed in that case.

4.3. Compass Model on Finite Paths with Noise

As before for the Deffuant model, we also want to consider the impact of noise on the Compass model. We again consider uniform and bi-modal noise that we define analogously as for the Deffuant model. The precise definitions are given in the following sections.

4.3.1. Uniform Noise

We start by looking at uniform noise. At every time that individuals interact, we simply draw a number uniformly at random form $[-\epsilon, \epsilon]$ for an $\epsilon > 0$ and add it to an individual selected uniformly at random. A precise definition follows.

Definition 4.3.1 (Compass model with uniform noise)**.** *Let $P_n = (V_n, E_n)$ be the finite path of length $n \in \mathbb{N}$ as defined in Definition 3.2.1. The **opinion space** is given by the space $S := \mathbb{R}/\mathbb{Z}$. Define the metric d on S as the metric giving the length of the shortest path between two points. Let $\mu \in (0, 0.5]$, $\epsilon \geq 0$. Furthermore, define the **state space** by $\Omega = [0,1]^V$ together with the product topology. Define a random variable ξ distributed as* Uniform$(-\epsilon, \epsilon)$*. Let $(\eta_t)_{t\geq 0}$ be a Feller process with probability generator $\mathcal{L}$, given by*

$$\mathcal{L}f(\eta) := \sum_{i=1}^{n-1} \left(\frac{1}{2} \left[f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right) + f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right) \right] - f(\eta) \right) + \sum_{i=1}^{n} \left(f(B_i\eta) - f(\eta) \right)$$

where $A^{(1)}_{\langle i,i+1\rangle}$ and $A^{(2)}_{\langle i,i+1\rangle}$ are given as before in Definition 4.1.1 and for all $i, m \in V_n$ we have

$$B_i\eta(m) = \begin{cases} \eta(m), & \text{if } m \neq i, \\ \eta(m) + \xi, & \text{if } m = i \end{cases}$$

*for all $f \in C(\Omega)$. We call this process the **Compass model with uniform noise**. When the initial configuration is the uniform distribution on V_n, we speak of the **Uniform Compass model with uniform noise**.*

To prove that this is well-defined, we need to show that $\mathcal{L}$ as defined above corresponds one-to-one to a Feller process as shown in the following corollary.

Corollary 4.3.2 (Compass model with uniform noise is a Feller process)**.** *There exists a well-defined and unique Feller process which has the generator from Definition 4.3.1.*

Proof. This proof is completely analogous to that of Theorem 4.2.2. □

4.3.2. Bi-Modal Noise

We now look at bi-modal noise. This type of noise pushes opinions of individuals towards the opinions 0 and 1. At every time when people interact, one individual x is chosen from the set of all individuals uniformly at random. Then an opinion a is chosen either from 0 or 1 uniformly at random. The opinion of the chosen individual x is then disturbed towards the chosen opinion a. The further away the opinion of x is from a, the more it gets changed.

The noise is quantified by a parameter ϵ. The opinion of x gets changed in the same way that the dynamics of the Compass Model would change it when it would react with a neighbour with opinion a and if μ where ϵ. We define the resulting process formally below.

Definition 4.3.3 (Compass model with bi-modal noise)**.** *Let $P_n = (V_n, E_n)$ be the finite path of length $n \in \mathbb{N}$. Let $\mu, \epsilon \in [0, 0.5]$. Let the **opinion space** be given by $S := \mathbb{R}/\mathbb{Z}$. Let the metric on S be the metric returning the length the shortest geodesic between two*

points in S. Define the generator $\mathcal{L}$ for continuous functions f and configurations $\eta \in \Omega$ as

$$\begin{aligned}\mathcal{L}f(\eta) := &\sum_{i=1}^{n-1}\left(\frac{1}{2}\left[f\left(A^{(1)}_{\langle i,i+1\rangle}\eta\right)+f\left(A^{(2)}_{\langle i,i+1\rangle}\eta\right)\right]-f(\eta)\right)\\ &+\sum_{i=1}^{n}\left(\frac{1}{4}\left[f\left(B^{(1)}_i\eta\right)+f\left(B^{(2)}_i\eta\right)+f\left(C^{(1)}_i\eta\right)+f\left(C^{(2)}_i\eta\right)\right]-f(\eta)\right)\end{aligned}$$

where $A^{(1)}_{\langle i,i+1\rangle}$ and $A^{(2)}_{\langle i,i+1\rangle}$ are given as before in Definition 4.1.1 and for all $i, v \in V_n$ we have

$$B^{(k)}_i\eta(v) = \begin{cases}\eta(v), & \text{if } v \neq i,\\ (1-\epsilon)\eta(v), & \text{if } v = i, \eta(v) \neq 1,\\ (-1)^k(1-\epsilon), & \text{if } v = i, \eta(v) = 1\end{cases}$$

and

$$C^{(k)}_i\eta(v) = \begin{cases}\eta(v), & \text{if } v \neq i,\\ (1-\epsilon)\eta(v)+\epsilon, & \text{if } v = i, \eta(v) \in (0,1],\\ (1-\epsilon)\eta(v)-\epsilon, & \text{if } v = i, \eta(v) \in (-1,0),\\ (-1)^k\epsilon, & \text{if } v = i, \eta(v) = 0\end{cases}$$

where $k \in \{1,2\}$.

Here $B^{(1)}_i$ and $B^{(2)}_i$ encode the case where $a = 0$. We need both versions to ensure rotational symmetry in the case where $\eta(v) = 1$. $C^{(1)}_i$ and $C^{(2)}_i$ describe the case where $a = 1$. Versions (1) and (2) ensure rotational symmetry for the case where $\eta(v) = 0$.

The proof that this generator defines a Feller process is analogous to the case without noise.

Corollary 4.3.4 (Compass model with bi-modal noise is a Feller process)**.** *There exists a well-defined and unique Feller process which has the generator from Definition 4.3.3.*

Proof. The proof that this generator defines a Feller process is analogous to the case without noise, because the generator has the same form. □

5. Consensus

The main goal of this project is to study how consensus is formed, that is, will the group eventually agree? There are different types of consensus, which are given in the following definition.

Definition 5.0.1 (Consensus)**.** *Let $(\eta_t)_{t\geq 0}$ be the Markov process describing either the Deffuant model or the Compass model on G as given in Definitions 3.1.2 and 4.1.1, respectively. We can then define:*

1. ***No consensus:*** *There is an $\epsilon > 0$ and an edge $\langle u, v\rangle \in E$ such that:*

$$\forall t_0 \geq 0 \ \exists t > t_0 : d(\eta_t(u), \eta_t(v)) \geq \epsilon.$$

2. ***Weak consensus:*** *For all edges $\langle u, v\rangle \in E$, it holds that $d(\eta_t(u), \eta_t(v)) \to 0$ for $t \to \infty$.*

3. ***Strong consensus:*** *There exists a random variable L such that $d(\eta_t(v), L) \to 0$ for $t \to \infty$ for all vertices $v \in V$.*

Because η_t is a random variable for all $t \geq 0$, the convergence to 0 needed for weak or strong consensus can either be almost sure convergence, convergence in $\mathcal{L}^1$ or convergence in probability. We then speak of a.s. (weak) consensus, (weak) consensus in mean and (weak) consensus in probability, respectively. This definition is taken from [GHH20, Chapter 1].

Intuitively, no consensus means that there are at least two neighbours that will always come to disagree again. Weak consensus means that all neighbours eventually come to agree for an indefinite time. It is the opposite of no consensus. Strong consensus is a special case of weak consensus. It means that all people will eventually agree on one common opinion. Note that strong consensus always implies weak consensus.

Also note that a.s. (weak) consensus implies (weak) consensus in probability and (weak) consensus in mean also implies (weak) consensus in probability.

Lemma 5.0.2. *For the Compass model and the Deffuant model on finite paths, weak and strong consensus are equivalent.*

Proof. The path of length 2 only contains one edge, so in that case the lemma trivially holds. Thus, we only treat paths of length $n \geq 3$ here.

As seen before, strong consensus always implies weak consensus. So it remains to show that on finite paths weak consensus implies strong consensus.

First, consider the Compass Model on a finite path (V_n, E_n) of length $n \geq 3$, that is, $V_n = \{1, \ldots, n\}$ and $E_n = \{\langle 1, 2\rangle, \ldots, \langle n-1, n\rangle\}$. Assume we have weak consensus. By the definition of weak consensus, $d(\eta_t(1), \eta_t(2)) \to 0$ and $d(\eta_t(2), \eta_t(3)) \to 0$ for $t \to \infty$. Thus, by the triangle inequality, $d(\eta_t(1), \eta_t(3)) \to 0$ for $t \to \infty$. Furthermore, by the definition of weak consensus, for all $i \in \{1, \ldots, n-1\}$, $d(\eta_t(i), \eta_t(i+1)) \to 0$ for $t \to \infty$. Thus, applying the triangle inequality repetitively gives $d(\eta_t(1), \eta_t(i)) \to 0$ for $t \to \infty$ for all $i \in V_n$. Therefore, there is a random variable L on S, such that $d(\eta_t(i), L) \to 0$ for all $i \in V_n$. That implies strong consensus. For the Deffuant model on the finite paths, the same argument holds. □

Definition 5.0.3. *Let $(\eta_t)_{t\geq 0}$ be either the Deffuant or the Compass model on the finite path P_n of length $n \geq 2$. We define the process $(c_t)_{t\geq 0}$ by $c_t = \sum_{i=1}^{n-1} d(\eta_t(i), \eta_t(i+1))$.*

Note that $(c_t)_{t\geq 0}$ is well-defined because the sum $c_t = \sum_{i=1}^{n-1} d(\eta_t(i), \eta_t(i+1))$ is always finite. Note that d is dependent on the chosen model.

5.1. Consensus in the Deffuant Model

Lemma 5.1.1. *For the Deffuant model $(\eta_t)_{t\geq 0}$ on the finite path P_n of length $n \geq 2$, we have that $(c_t)_{t\geq 0}$ is non-increasing. That is, $c_t \leq c_l$ for every $t \leq l$.*

Proof. Let $(c_t)_{t\geq 0}$ be as defined in Definition 5.0.3. In time intervals where no update of opinions occurs, c_t remains constant. Thus, we only have that c_t cannot increase when an interaction occurs. Assume an interaction occurs at time $t > 0$ at edge $e_i = \langle i, i+1\rangle$ with $i \in \{1, \ldots, n-1\}$. Because we have finitely many vertices, there is a time $s < t$ such that no interactions occur in $[s, t)$. Then, precisely the distances of pairs of neighbours where one is either i or $i+1$ change their values at time t. Let us distinguish between the following three cases:

1. $i = 1$,
2. $i \in \{2, \ldots, n-2\}$,
3. $i = n-1$.

In case (1.) only the values of $\eta_t(1)$ and $\eta_t(2)$ change. We get the following updates:

$$\begin{aligned}\eta_t(1) &= \eta_s(1) - \mu(\eta_s(1) - \eta_s(2))\\ \eta_t(2) &= \eta_s(2) + \mu(\eta_s(1) - \eta_s(2))\\ \eta_t(i) &= \eta_s(i) \text{ for all } i \geq 3.\end{aligned}$$

Thus, the distances between neighbours update in the following way:

$$\begin{aligned}
d(\eta_t(1),\eta_t(2)) &= |((\eta_s(1)-\mu(\eta_s(1)-\eta_s(2)))-(\eta_s(2)+\mu(\eta_s(1)-\eta_s(2)))| \\
&= (1-2\mu)d(\eta_s(1),\eta_s(2)) \\
d(\eta_t(2),\eta_t(3)) &= |(\eta_s(2)+\mu(\eta_s(1)-\eta_s(2)))-\eta_s(3))| \\
&= |\eta_s(2)-\eta_s(3)+\mu(\eta_s(1)-\eta_s(2))| \\
&\leq d(\eta_s(2),\eta_s(3))+\mu d(\eta_s(1)-\eta_s(2)) \\
d(\eta_t(i),\eta_t(i+1)) &= d(\eta_s(i),\eta_s(i+1)) \text{ for all } i\geq 3.
\end{aligned}$$

Thus,

$$c_t=\sum_{i=1}^{n-1} d(\eta_t(i),\eta_t(i+1)) \leq c_s-\mu d(\eta_s(1),\eta_s(2)).$$

Because all summands of this equation are positive, $c_t \leq c_s$.

An analogous statement holds for case (3.). This is the same scenario; just renumber the vertices in reversed order starting with the previously largest.

Case (2.) is slightly different, because here there are three distances between neighbours that change. In this case, η only changes at vertex i and $i+1$. We therefore get the following updates:

$$\begin{aligned}
\eta_t(i) &= \eta_s(i)-\mu(\eta_s(i)-\eta_s(i+1)) \\
\eta_t(i+1) &= \eta_s(i+1)+\mu(\eta_s(i)-\eta_s(i+1)) \\
\eta_t(j) &= \eta_s(j) \text{ for all } j\in V_n\backslash\{i,i+1\}.
\end{aligned}$$

Thus, the distances between neighbours update in the following way:

$$\begin{aligned}
d(\eta_t(i),\eta_t(i+1)) &= |((\eta_s(i)-\mu(\eta_s(i)-\eta_s(i+1)))-(\eta_s(i+1)+\mu(\eta_s(i)-\eta_s(i+1)))| \\
&= (1-2\mu)d(\eta_s(i),\eta_s(i+1)), \\
d(\eta_t(i+1),\eta_t(i+2)) &= |(\eta_s(i+1)+\mu(\eta_s(i)-\eta_s(i+1)))-\eta_s(i+2))| \\
&= |\eta_s(i+1)-\eta_s(i+2)+\mu(\eta_s(i)-\eta_s(i+1))| \\
&\leq d(\eta_s(i+1),\eta_s(i+2))+\mu d(\eta_s(i)-\eta_s(i+1)), \\
d(\eta_t(i-1),\eta_t(i)) &= |\eta_s(i-1)-(\eta_s(i)-\mu(\eta_s(i)-\eta_s(i+1)))| \\
&= |\eta_s(i-1)-\eta_s(i)+\mu(\eta_s(i)-\eta_s(i+1)))| \\
&\leq d(\eta_s(i-1),\eta_s(i))+\mu d(\eta_s(i),\eta_s(i+1)), \\
d(\eta_t(j),\eta_t(j+2)) &= d(\eta_s(j),\eta_s(j+1)) \text{ for all } j\in V_n\backslash\{i-1,i,i+1\}.
\end{aligned}$$

Thus,

$$c_t=\sum_{i=1}^{n-1} d(\eta_t(i),\eta_t(i+1)) \leq c_s.$$

Again, because all summands of this equation are positive, $c_t \leq c_s$.

The above argument shows that $c_t \leq c_s$ for all $t \geq s \geq 0$. □

Theorem 5.1.2 (Consensus of the Deffuant model on finite paths)**.** *The Deffuant model on a finite path exhibits almost sure strong consensus. Furthermore, the opinion of each individual converges to the average of the initial opinions.*

Proof. Consider the Deffuant model on the finite path $P_n = (V_n, E_n)$ as defined in Definition 3.2.2. Also consider the Compass model on the finite path P_n given by Definition 4.2.1 with initial opinions that all lie on the same half of the circle. Then by definition, both models exhibit the same behaviour and produce the same configurations. Thus, almost sure strong consensus for the Deffuant model follows from almost sure strong consensus for the Compass model as will be shown in Theorem 5.2.2.

This means, that $L = \lim_{t\to\infty} \eta_t$ is a well-defined random variable. Furthermore, notice that, whenever an interaction between two individuals occurs, the average of the opinions stays identical. The average of the opinions at the end must then be the same as the average of the initial opinions. Additionally, $\lim_{t\to\infty} c_t = 0$ because there is strong consensus. Thus, L must be given by $L = \left(\sum_{i=0}^{n} \eta_0(i)\right)^{V_n}$. □

This theorem means, that we expect c_t to converge to 0 as $t \to \infty$ in numerical simulation. That also means that if we visualise the configuration at a time t as a plot with the individuals on the x-axis and their opinions on the y-axis, we expect to see increasingly flat lines of opinions at t increases. Furthermore, we expect this line to lie at the height of approximately the average of the initial opinions. In particular, this implies that only the initial condition and not the interactions are relevant for the outcome.

5.2. Consensus in the Compass Model

First, we consider the Compass model without noise.

Lemma 5.2.1. *For the Compass model $(\eta_t)_{t\geq 0}$ on the finite path P_n of length $n \geq 2$, we have that $(c_t)_{t\geq 0}$ is non-increasing. That is, $c_t \leq c_l$ for every $t \leq l$.*

Proof. This proof is completely analogously to that of Lemma 5.1.1, which gives an analogous statement for the Deffuant model. In this proof the key point was that c_t only changes whenever an interaction occurs. Let t be the time of such an interaction. Because we have finitely many individuals in the model, there is a time $s < t$ such that no interactions occur in the interval $[s, t)$. Therefore, shortly before time t, the configuration is the same as at time s. At time t, only the opinions of two neighbours change. That means that the distance of their opinion decreases by 2μ times their previous distance. The distance of the neighbours on the edges right next to this one can increase. However, by a maximum of μ times their previous distance each. The distances of all other neighbours are unchanged.

In total, this means that c_t cannot be larger than c_s. The only difference to the case of the Deffuant model is that we look at opinions as points on the circle and their distances as the length of the shortest path on the circle between them. However, this does not change the argument. □

Theorem 5.2.2 (Consensus of the Compass model on finite paths)**.** *The Compass model on finite paths exhibits almost sure strong consensus irrespective of the chosen initial condition. [GHH20, Lemma 3.1]*

Proof. A straightforward proof of this theorem can be found in [GHH20, Lemma 3.1]. □

In the case of the Deffuant model, we saw that the configurations converged to

$$L = \left(\sum_{i=0}^{n} \eta_0(i) \right)^{V_n}.$$

Note, that this is not the case in the Compass model. The problem is that the average of the opinions can change with time.

To see this, consider the following example: Let there be an interaction between individuals i and $i+1$ at time t. Before time t, let i have opinion 0.9 and let $i+1$ have opinion -0.7. Thus, the average of their opinions is 0.1. Let $\mu = 0.5$. Then after the interaction, both i and $i+1$ have opinion -0.9. Therefore, the average of the opinions decreased by -1.0.

Again, we must ask ourselves what behaviour these results predict for the numerical simulations in the following part of the project. We expect that c_t converges to 0 as $t \to \infty$ in numerical simulation, as in the Deffuant model. This again implies increasingly flat configurations. However, we cannot predict the opinion upon which the individuals will eventually agree. According to [GHH20, Corollary 1.3], these opinions are uniformly distributed for the Uniform Compass model on $\mathbb{Z}$. Because we will simulate the Uniform Compass model on the finite path, which is a finite connected sub-graph of $\mathbb{Z}$, we expect this results hold in our simulations as well.

6. Invariant Measures

In this chapter we briefly give analytical results for the invariant measures of the Deffuant model and the Compass model.

6.1. Invariant Measures for the Deffuant Model

Let us consider the Deffuant model on the finite path $P_n = (V_n, E_n)$. For all $s \in S$, define the configuration s^{V_n} by $s^{V_n}(i) = s$ for all $i \in V_n$. Let $\delta_{s^{V_n}}$ denote the Dirac measure of the configuration s^{V_n}. These Dirac measures are invariant, which we will show in the following lemma.

Lemma 6.1.1. *$\delta_{s^{V_n}}$ is an invariant measure for every $s \in S$.*

Proof. Let the Deffuant model be as defined in Definition 3.2.2. Let $f \in C(\Omega)$ and define a function $g_s : \Omega \to \mathbb{R}, \eta \mapsto \mathcal{L}f(s^{V_n})$ for every $s \in S$. Since g_s is a constant function, it is measurable. Note that

$$\delta_{s^{V_n}}(\{\eta | g_s(\eta) \neq \mathcal{L}f(s^{V_n})\}) = \delta_{s^{V_n}}(\Omega \backslash \{s^{V_n}\}) = 0.$$

Thus, $g_s = \mathcal{L}f$ holds $\delta_{s^{V_n}}$-almost everywhere. Therefore, $\int g_s d\delta_{s^{V_n}} = \int \mathcal{L}f d\delta_{s^{V_n}}$. Because the Dirac measure is a probability measure and g_s a constant function, we have

$$\int \mathcal{L}f d\delta_{s^{V_n}} = \int g_s d\delta_{s^{V_n}} = \mathcal{L}f(s^{V_n}).$$

Now, note that for all $i \in V_n \backslash \{n\}$, we have that

$$A_{\langle i, i+1 \rangle} s^{V_n} = s^{V_n}.$$

This implies that

$$\int \mathcal{L}f d\delta_{s^{V_n}} = \mathcal{L}f(s^{V_n}) = 0$$

for all $s \in S$. By Proposition 2.2.2 that implies that $\delta_{s^{V_n}}$ is an invariant measure. □

6.2. Invariant Measures for the Compass Model

Let us consider the Compass model on the finite path $P_n = (V_n, E_n)$. Let $s \in S$. Define the configuration s^{V_n} and the Dirac measure $\delta_{s^{V_n}}$ analogously as for the Deffuant model in the previous section. According to [GHH20, Chapter 1], these Dirac measures are invariant for the Compass model on the graph $G = (\mathbb{Z}, E)$ where $E = \{\langle i, i+1 \rangle | i \in \mathbb{N}\}$. In the following lemma, we will show that this also holds for the Compass model on the finite path.

Lemma 6.2.1. *$\delta_{s^{V_n}}$ is an invariant measure for every $s \in S$.*

Proof. Let the Compass model be as defined in Definition 4.2.1. The proof that $\delta_{s^{V_n}}$ is an invariant measure for every $s \in S$ is absolutely analogous to that shown in Lemma 6.1.1. Simply note that for every $i \in V_n \backslash \{n\}$, we have that

$$A^{(1)}_{\langle i,i+1\rangle} s^{V_n} = A^{(2)}_{\langle i,i+1\rangle} s^{V_n} = s^{V_n}.$$

□

Part III.

Numerical Studies

7. Numerical Simulation of the Compass Model and the Deffuant Model

As seen in Part II, there are very few theoretical results for the Compass model. Proving anything is very difficult. Therefore, we are now implementing a computer simulation of the model with different versions of noise. For comparison, we will also implement the Deffuant model. Note that for we will always assume empty boundary conditions. Our goal is to verify the known results and to gain an insight into the effect of noise on these models. We will start by describing the models as algorithms that we can implement in Python. Some initial inspiration on how to implement the Deffuant model came from [dax].

7.1. Implementation without Noise

Let us first look at how to simulate the Deffuant model and the Compass model without noise for N individuals on the finite path given by the undirected graph $G = (V, E)$. Here the set of vertices V is the set of individuals $\{1, \dots, N\}$ and the edges are given by $E = \{< 1, 2 >, < 2, 3 >, \dots, < N-1, N >\}$. Let the parameter $\mu \in (0, \frac{1}{2}]$ be fixed. In order to simulate the models, we need to know when individuals interact and what happens when they do. The time when interactions occur and which individuals are involved in the interaction is modelled using independent Poisson clocks, as defined in Definition 7.1.2, on the edges as will be described in [GHH20, Chapter 1].

Definition 7.1.1 (Poisson process)**.** *A Poisson process with parameter $\lambda > 0$ is a stochastic process $(X_t)_{t \geq 0}$ that satisfies the following properties:*

1. *$X_0 = 0$ almost surely.*
2. *The random variables $X_{t_1}, X_{t_2} - X_{t_1}, \dots, X_{t_n} - X_{t_{n-1}}$ are independent for all finite increasing non-negative sequences of times $t_1 < t_2 < \cdots < t_n$, .*
3. *$X_t - X_s \sim Poisson(\lambda(t-s))$ for all non-negative times $t > s$.*

[KS12, Chapter 12.3]

Definition 7.1.2 (Poisson clock)**.** *A Poisson clock is a process that returns the times at which a Poisson process with parameter $\lambda = 1$ as defined in Definition 7.1.1 changes its value. [Häg12, Chapter 1] We say that a Poisson clock rings at those times.*

Note that the distances between the times at which a Poisson clock rings are exponentially distributed [KS12, Ch.12.3]. In our models, we have such a Poisson clock on every edge in E and assume that these Poisson clocks are independent. With probability one, at every point in time $t \in (0, \infty)$ there exists at most one edge such that the Poisson clock of that edge rings at time t. Two neighbours in our graph G interact at precisely the times when the Poisson clock on the edge connecting them rings. [GHH20, Chapter 1]

The pseudo-code in Algorithm 1 describes how to simulate those Poisson clocks. Let us simulate them up to the time when $m > N$ interactions between neighbours have taken place. Let t be the vector giving the interaction times. p be the vector giving the places where the interactions take place, that is, the i'th interaction takes place at edge $< j, j+1 >$ if and only if $p_i = j$.

Algorithm 1 Poisson clocks

e vector of length $m + N$ of numbers drawn from an exponential distribution with parameter 1.0
$a \leftarrow (e_1, \dots, e_N)$ vector giving the next time each Poisson clock will ring
$p \leftarrow (0, \dots, 0)$ vector of length m giving the places where interactions take place and their order
$t \leftarrow (0, \dots, 0)$ vector of length m giving the times at which interactions take place
for $i \in (1, 2, \dots, m)$ **do**
    $p_i \leftarrow \operatorname{argmin} a$,
    $t_i \leftarrow \min a$,
    $a_{p_i} \leftarrow e_{N+i} + t_i$
end for

Using the above algorithm to describe where and when interactions take place, we can go on to describe how an interaction on edge $< j, j+1 >$ changes the configuration η to a configuration $\bar{\eta}$. This is done in Algorithm 2 for the Deffuant model and in Algorithm 3 for the Compass model.

Algorithm 2 i'th step in the Deffuant model

Let p, t be as calculated by Algorithm 1
Let η be the configuration before the i'th step
$\bar{\eta} \leftarrow \eta$ configuration after i'th step
$\bar{\eta}_{p_i} \leftarrow (1-\mu)\eta_{p_i} + \mu\eta_{p_i+1}$
$\bar{\eta}_{p_i+1} \leftarrow (1-\mu)\eta_{p_i+1} + \mu\eta_{p_i}$

Algorithm 3 i'th step in the Compass model

Let p, t be as calculated by Algorithm 1
Let η be the configuration before the i'th step
$\bar{\eta} \leftarrow \eta$ configuration after i'th step
Let r be a random number drawn from a uniform distribution on $0, 1$
if $|\eta_{p_i} - \eta_{p_i+1}| = 1$ **then**
  if $r = 0$ **then**
    $\bar{\eta}_{p_i} \leftarrow \eta_{p_i} + \mu \cdot \mathrm{sgn}(\eta_{p_i})$
    $\bar{\eta}_{p_i+1} \leftarrow \eta_{p_i} + \mu \cdot \mathrm{sgn}(\eta_{p_i+1})$
  else
    $\bar{\eta}_{p_i} \leftarrow \eta_{p_i} - \mu \cdot \mathrm{sgn}(\eta_{p_i})$
    $\bar{\eta}_{p_i+1} \leftarrow \eta_{p_i+1} - \mu \cdot \mathrm{sgn}(\eta_{p_i+1})$
  end if
else
  if $|\eta_{p_i} - \eta_{p_i+1}| < 1$ **then**
    $\bar{\eta}_{p_i} \leftarrow (1-\mu)\eta_{p_i} + \mu\eta_{p_i+1}$
    $\bar{\eta}_{p_i+1} \leftarrow (1-\mu)\eta_{p_i+1} + \mu\eta_{p_i}$
  else
    $\bar{\eta}_{p_i} \leftarrow \eta_{p_i} + \mu(2 - |\eta_{p_i+1} - \eta_{p_i}|) \cdot \mathrm{sgn}(\eta_{p_i})$
    $\bar{\eta}_{p_i+1} \leftarrow \eta_{p_i+1} + \mu(2 - |\eta_{p_i+1} - \eta_{p_i}|) \cdot \mathrm{sgn}(\eta_{p_i+1})$
  end if
end if
if $\bar{\eta}_{p_i} > 1$ **then**
  $\bar{\eta}_{p_i} \leftarrow \bar{\eta}_{p_i} - 2$
end if
if $\bar{\eta}_{p_i+1} > 1$ **then**
  $\bar{\eta}_{p_i+1} \leftarrow \bar{\eta}_{p_i+1} - 2$
end if
if $\bar{\eta}_{p_i} < -1$ **then**
  $\bar{\eta}_{p_i} \leftarrow \bar{\eta}_{p_i} + 2$
end if
if $\bar{\eta}_{p_i+1} < -1$ **then**
  $\bar{\eta}_{p_i+1} \leftarrow \bar{\eta}_{p_i+1} + 2$
end if

The next step is to apply the previous algorithms repetitively to simulate the Deffuant model and the Compass model. (See Algorithm 4 for the Deffuant model and Algorithm 5 for the Compass model.) Note that the configuration only changes at the finite set of times given by the Poisson clocks. In between these time steps the configuration remains constant.

Algorithm 4 Deffuant model

p, t from Algorithm 1
add a first value of 0 at the beginning of t
m length of p
$(\eta_i)_{i \in t}$ be the configurations of the Deffuant model
η_0 be the starting configuration chosen using a suitable probability distribution
for $j \in (1, 2, \ldots, m)$ **do**
    execute j'th step of Deffuant model to compute η_{t_j} from $\eta_{t_{j-1}}$ using Algorithm 2
end for

Algorithm 5 Compass model

p, t from Algorithm 1
add a first value of 0 at the beginning of t
m length of p
$(\eta_i)_{i \in t}$ be the configurations of the Compass model
η_0 be the starting configuration chosen using a suitable probability distribution
for $j \in (1, 2, \ldots, m)$ **do**
    execute j'th step of Compass model to compute η_{t_j} from $\eta_{t_{j-1}}$ using Algorithm 3
end for

Now, that we have simulated the models, we would like to be able to check whether consensus is reached. Remember that on finite paths weak consensus is equivalent to strong consensus as shown in Lemma 5.0.2. In our computer simulations we only simulate the models on finite paths, therefore we cannot study the difference between these types of consensus. Furthermore, in every simulation we have to stop after a finite time. Thus, we will say that we have reached consensus if all individuals agree on one opinion. When that is the case, we know that we have strong consensus as defined in the theoretical part, that is, Part II. The definition of (approximate) consensus used in this part of the project will be given in Definition 7.1.3. We need the approximate version of consensus to describe the scenario in which very small noise has a negligible impact on the simulation, but because noise will always destroy consensus at arbitrarily large times $c = 0$ simply cannot be reached and is also not very important. Important is that consensus is approximated within a sufficiently small bandwidth.

Definition 7.1.3 ((Approximate) consensus)**.** *Let η_t denote the configuration at time $t \geq 0$. We say that we have consensus (at time t) if there is an opinion o such that $\eta_t(i) = o$ for all individuals $i = 1, \ldots, N$. If there exists a small $\alpha > 0$ such that there is an opinion o such that $||\eta_t(i) - o|| \leq \alpha$ for all individuals $i = 1, \ldots, N$, we speak of approximate consensus (at time t) within a bandwidth of width α.*

A necessary criterion for consensus is that the sum of the distances between neighbours at time t, c_t, as defined in Definition 5.0.3, is 0. In this chapter we will simply write c instead of c_t and assume that the time is clear from the context. This immediately implies consensus as described above. Furthermore, this sum is non-increasing in the

case without noise as shown in Lemma 5.1.1 for the Deffuant model and in Lemma 5.2.1 for the Compass model. That means that in the case without noise, once consensus is reached, the configuration cannot change anymore.

Another way to look at consensus in the models, is to view it as synchronisation of the opinions. Analogously to what is typically done for the Kuramoto model, as for example in [CR16], we can investigate the order parameters r and θ defined in Definition 7.1.4.

Definition 7.1.4 (Order parameters r and θ)**.**

1. *Let $(\eta_t)_{t\geq 0}$ denote the configuration given by the Compass model on a finite path. Let S be as defined in Definition 4.1.1. We then define $r : [0,\infty) \to [0,1]$ and $\theta : [0,\infty) \to S$ by*
$$r(t)e^{i\pi\theta(t)} := \frac{1}{N}\sum_{j=1}^{N} e^{i\pi\eta_t(j)}$$

2. *Let $(\eta_t)_{t\geq 0}$ denote the configuration given by the Deffuant model on a finite path. We then define functions $r, \theta : [0,\infty) \to [0,1]$ by*
$$r(t)e^{2i\pi\theta(t)-\pi} := \frac{1}{N}\sum_{j=1}^{N} e^{2i\pi\eta_t(j)-\pi}.$$

r then measures the degree to which the opinions have aligned and θ the average opinion. Basically we view each opinion as a point on the circle and re-scale the circle to be a unit circle. The opinions are then viewed as directions. We calculate a vector pointing into the most prominent direction θ who's length r is a measure of how prominent that directions is. If $r = 0$ all opinions are equally distributed over the circle. If $r = 1$ all opinions are equal and we have reached consensus. The closer r is to 1 the more synchronised the opinions are.

For the Compass model, $r(t)$ can be calculated as

$$r(t) = \frac{1}{N}\sqrt{\left(\sum_{j=1}^{N}\cos(\pi\eta_t(j))\right)^2 + \left(\sum_{j=1}^{N}\sin(\pi\eta_t(j))\right)^2}.$$

One can then calculate

$$\cos(\pi\theta(t)) = \frac{1}{Nr(t)}\sum_{j=1}^{N}\cos(\pi\eta_t(j))$$
$$\sin(\pi\theta(t)) = \frac{1}{Nr(t)}\sum_{j=1}^{N}\sin(\pi\eta_t(j))$$

and define

$$\phi(t) := \arcsin\left(\frac{1}{Nr(t)}\sum_{j=1}^{N}\sin(\pi\eta_t(j))\right),$$

such that

$$\theta(t) = \begin{cases} \phi(t), & \text{if } \cos(\theta(t)) \geq 0, \\ \pi - \phi(t), & \text{if } \cos(\theta(t)) < 0, \phi \geq 0, \\ -\pi - \phi(t), & \text{if } \cos(\theta(t)) < 0, \phi(t) < 0. \end{cases}$$

For the Deffuant model we calculate r and θ analogously only taking into account the different scaling of the opinions as seen in Definition 7.1.4

7.2. Implementation of Noise

Noise can, for example, be used to simulate what happens if individuals change their opinions spontaneously or due to environmental influences. There are different ways of implementing noise. We implemented uniform and bi-modal noise that were also introduced in Sections 3.2 and 4.3.

7.2.1. Uniform Noise

We start by writing an algorithm that implements the Compass model and Deffuant model with uniform noise. In both models we proceed very similarly. We start by fixing an $\epsilon > 0$ before the simulation. Then we run the simulation as described in the previous section (see Algorithm 4 for the Deffuant model and Algorithm 5 for the Compass model), but after every step of the simulation we pick one individual i uniformly at random from the population and draw a value n from Uniform($[-\epsilon, \epsilon]$). Then, we add n to the opinion of the individual i before proceeding to the next step of the simulation. For the Deffuant model, this process is described in Algorithm 6. Algorithm 7 describes the Compass model with uniform noise.

Algorithm 6 Deffuant model with uniform noise

p, t are generated by Algorithm 1
include a first value of 0 at the beginning of t
m denotes the length of p
$(\eta_i)_{i\in t}$ denotes the configurations of the Deffuant model
η_0 is chosen from a suitable initial probability distribution
ϵ is the parameter used to calculate by which amount an opinion gets changed when it is perturbed by noise
for $j \in (1, 2, \ldots, m)$ **do**
    execute j'th step of Deffuant model to compute η_{t_j} from $\eta_{t_{j-1}}$ using Algorithm 2
    pick i uniformly at random from $\{1, \ldots, N\}$
    draw n from $\mathcal{U}([-\epsilon, \epsilon])$
    add n to the opinion of individual i
end for

Algorithm 7 Compass model with uniform noise

p, t are generated by Algorithm 1
include a first value of 0 at the beginning of t
m denotes the length of p
$(\eta_i)_{i\in t}$ denotes the configurations of the Compass model
η_0 is chosen from a suitable initial probability distribution
ϵ is the parameter used to calculate by which amount an opinion gets changed when it is perturbed by noise
for $j \in (1, 2, \ldots, m)$ **do**
    execute j'th step of Compass model to compute η_{t_j} from $\eta_{t_{j-1}}$ using Algorithm 3
    pick i uniformly at random from $\{1, \ldots, N\}$
    draw n from $\mathcal{U}([-\epsilon, \epsilon])$
    add n to the opinion of individual i
end for

7.2.2. Bi-Modal Noise

In this section we will describe the bi-modal noise in terms of an algorithm that can be implemented in Python. For this type of noise, at every time step, one individual changes its opinion to be closer to an opinion chosen uniformly at random from $\{0, 1\}$. For the Compass model this is described in Algorithm 8 and for the Deffuant model in Algorithm 9.

Algorithm 8 Compass model with bi-modal noise

p, t be generated by Algorithm 1
include a first value of 0 at the beginning of t
m denotes the length of p
$(\eta_i)_{i\in t}$ denotes the configurations of the Compass model
η_0 is the starting configuration chosen using a suitable initial probability distribution
ϵ is the parameter used to calculate by which amount an opinion gets changed when it is perturbed by noise
for $j \in (1, 2, \ldots, m)$ **do**
    execute j'th step of Compass model to compute η_{t_j} from $\eta_{t_{j-1}}$ using Algorithm 3
    choose an individual I uniformly at random and denote its current opinion by I_o
    choose D uniformly at random from $\{0, 1\}$
    if $D = 0$ **then**
        if $I_o = 1$ **then**
            draw u uniformly at random from $\{0, 1\}$
            if $u = 0$ **then**
                update I_o to $\epsilon - 1$
            else
                update I_o to $1 - \epsilon$
            end if
        else
            multiply I_o by $1 - \epsilon$
        end if
    else
        if $I_o = 0$ **then**
            draw u uniformly at random from $\{0, 1\}$
            if $u = 0$ **then**
                set I_o to ϵ
            else
                set I_o to $-\epsilon$
            end if
        else
            if $I_o > 0$ **then**
                multiply I_o by $1 - \epsilon$ and add ϵ
            else
                multiply I_o by $1 - \epsilon$ and subtract ϵ
            end if
        end if
    end if
end for

Algorithm 9 Deffuant model with bi-modal noise

p, t be generated by Algorithm 1
include a first value of 0 at the beginning of t
m denotes the length of p
$(\eta_i)_{i \in t}$ denotes the configurations of the Deffuant model
η_0 is the starting configuration chosen using a suitable initial probability distribution
ϵ is the parameter used to calculate by which amount an opinion gets changed when it is perturbed by noise
for $j \in (1, 2, \ldots, m)$ **do**
    execute j'th step of Deffuant model to compute η_{t_j} from $\eta_{t_{j-1}}$ using Algorithm 2
    choose an individual I uniformly at random and denote its current opinion by I_o
    choose D uniformly at random from $\{0, 1\}$
    multiply I_o by $1 - \epsilon$ and add $\epsilon \cdot D$
end for

7.3. Programming Methods

We implemented the algorithms described in the previous sections in the programming language Python. Pythons' biggest advantage for this project is that it is a relatively common programming language that runs on every ordinary computer system. Because it is such a common language, there are many big libraries of code already available online. For example the `numpy` library was essential to this project as it provides the basic tools for array computing, which is essential in scientific computing [CKSe20]. The `scipy` library was very helpful as well, because it provides many statistical methods [VGOe20]. The figures used in this thesis where generated with the help of the `matplotlib` library [Hun07].

Furthermore, Python is available for free. This is helpful to assure that others can reproduce the results shown here.

Programming the algorithms from the previous sections and running simulations in Python caused some unexpected difficulties. The major challenge that came up was that the simulations are rather time consuming. In particular, it takes a long time for one simulation to converge. In order to assure convergence in Uniform Compass model without Noise for $N = 200$ and all values of μ, $m = 10^8$ steps where needed. Thus, we started by trying to optimise the run-time. However, that only gave somewhat satisfactory results.

Given the initial conditions are drawn from a probability distribution and the interaction times and places are calculated using Poisson clocks, it is necessary to run the simulation multiply times. In order to get somewhat interesting results, we decided that the minimum number of times that the simulation should run for is 50. Given the large number of possible parameter choices, this quickly becomes very time consuming. The solution to this challenge was to reduce the number of parameter choices analysed, keep N small, only run the simulation 50 times and run 5 simulations simultaneous using the

multiprocessing package found under [mul]. Another way to keep the time needed for the simulation down, is to calculate the parameters r, θ and c at fewer steps. In this project every 10^5th step where chosen. This has the added advantage of keeping the storage (that is memory on the computer harddrive) and memory (that is RAM) lower, which was another difficulty.

Storing one array with 10^8 floats takes approximately 762 MB of data. Saving an array with 10^8 integers takes approximately 381 MB. For every simulation we would like to store the interaction times and need to also calculate the interaction order. That takes approximately 1143 MB of storage. When we have noise, we also need to calculate two arrays of length 10^8 in order to describe this noise. Saving this as well doubles the storage needed. In total, this means we needed approximately 56 GB of storage for one set of parameters for a simulation without noise. This was challenging. It was obvious that this procedure would not work for the case with noise, because it would simply occupy too much storage. Therefore, we pre-calculated the initial condition, interaction times, interaction places, noise direction and noise location, 50 times and used these for the different parameter sets. This is possible because this part of the simulation is parameter independent. With calculating r, θ and c at only every 10^5th step of the simulation, this reduced the needed storage and memory significantly. The run-times for the different versions of the models and pre-calculated initial conditions, interaction times and interaction places are given in Table 7.1.

For the case with noise, it was not always possible to observe convergence within 10^8 steps of the simulation. Given the restrictions of the system we worked on, it turned out to be impossible to run more than 10^8 at once. For larger simulations, the data needed to be stored elsewhere before re-starting the simulation where it had left off. This exceeded the scope of the project.

Model	Run-Time
UDM without noise	2min 13s
UDM with uniform noise	4min 20s
UDM with bi-modal noise	6min 19s
UCM without noise	2min 15s
UCM with uniform noise	5min 33s
UCM with bi-modal noise	5min 21s

Table 7.1.: The table gives the run-times for one interpretation of the simulations of the Uniform Deffuant Model (UDM) and Uniform Compass Model (UCM) with different versions of noise. Each simulation was run for 10^8 with $N = 200$ and $\mu = 0.5$. When noise was added, $\epsilon = 0.5$ was used. For each model, one interpretation of one model produced a folder of size 25.5 KB. The initial configuration, interaction order, interaction times, noise location and direction where calculated before the simulation and only loaded during the simulation. This pre-calculated data produced a folder of size 2.60 GB per needed interpretation of the simulation and took 34s to produce. That is, we needed 50 versions of this folder. The parameters c, r and θ where only calculated at every 10^5-th step of the simulations.

8. Numerical Results

In this section we want to illustrate the results of the numerical simulations of the Deffuant and Compass model. Our goal is to study the long-time behaviour of the models and confirm theoretical results introduced in Part II. We begin by studying both models without noise in Sections 8.1 and 8.2 respectively. Here, we also pay attention to the effect that the parameter μ has on the results. We then go on to study the effect of noise on the Deffuant model in Section 8.3 and of noise on the Compass model in Section 8.4. In these sections we restrict our attention to the case where $\mu = 0.5$.

In Section 8.5, we will discuss the question whether it makes a significant difference if we define noise at independent times from the interaction times.

After that, we compare the previously observed behaviour of the Deffuant and the Compass models in Section 8.6.

8.1. Uniform Deffuant Model without Noise

In this section we aim to visualise the results of the numerical simulations for the Uniform Deffuant model without noise.

To simulate the Uniform Deffuant model, using Algorithm 4, a population size of $N = 200$ was chosen. The simulations lasted $m = 5 \cdot 10^7$ steps for $\mu = 0.2, 0.3, 0.4$ and 0.5. For $\mu = 0.1$, 10^8 steps where needed to ensure, that the simulations had adequate time to converge. For every value of μ, each simulation consisted of 50 interpretations. Uniform initial conditions and empty boundary conditions were used.

Figure 8.1 visualises how the individual opinions can converge by showing the configuration at selected times during the simulation. We see that the opinions converge quickly at first and then once they are closer to each other the individuals take a long time to fully agree with each other. In the last configuration shown in the figure, the opinions form a horizontal line. This means everyone has the same opinion. The individuals agree on an opinion close to 0.5, which is approximately the average of the starting opinions.

In Figure 8.2, we visualised the values of the parameter c at different times in the simulations. One can see that c first decreases rapidly and then starts to take longer to converge to 0. From the zoomed in frames, we see that the initial decrease of c is larger for larger values of μ. Also the bandwidth of the values of c over the different interpretations of the simulations seems to be largest for smaller μ-values.

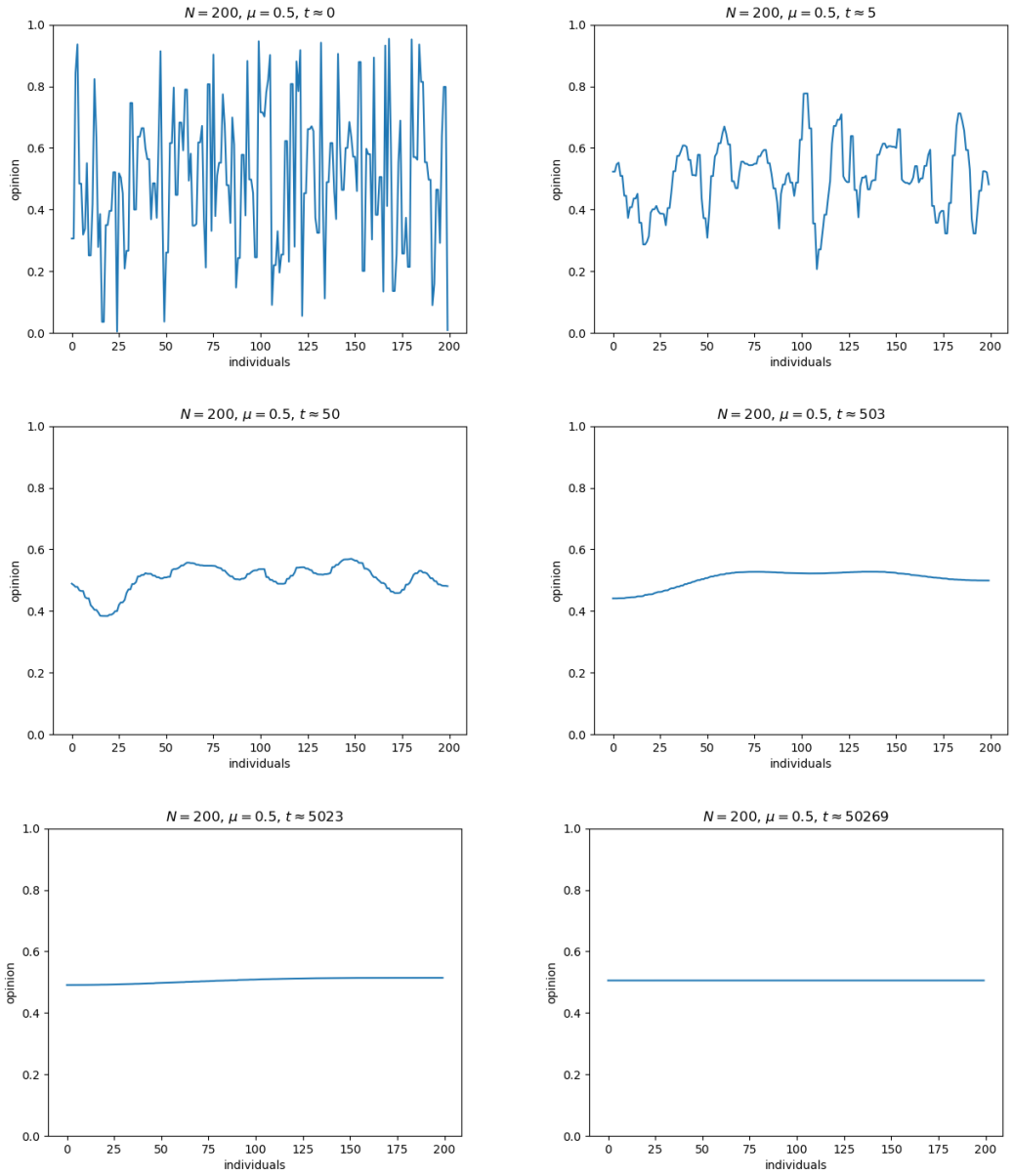

Figure 8.1.: This figure shows selected configurations that appeared in one realisation of the Uniform Deffuant Model without noise for $\mu = 0.5$. In each figure, the x-axis represents the individuals and the y-axis their opinions. The different opinions are connected by straight lines. The figure on the upper left corner gives the starting configuration. The configuration in the upper right hand corner shows the configuration at time 5. In the middle row, the configurations at time 50 and at time 503 are presented. The bottom row gives the configurations at times 5023 and 50269.

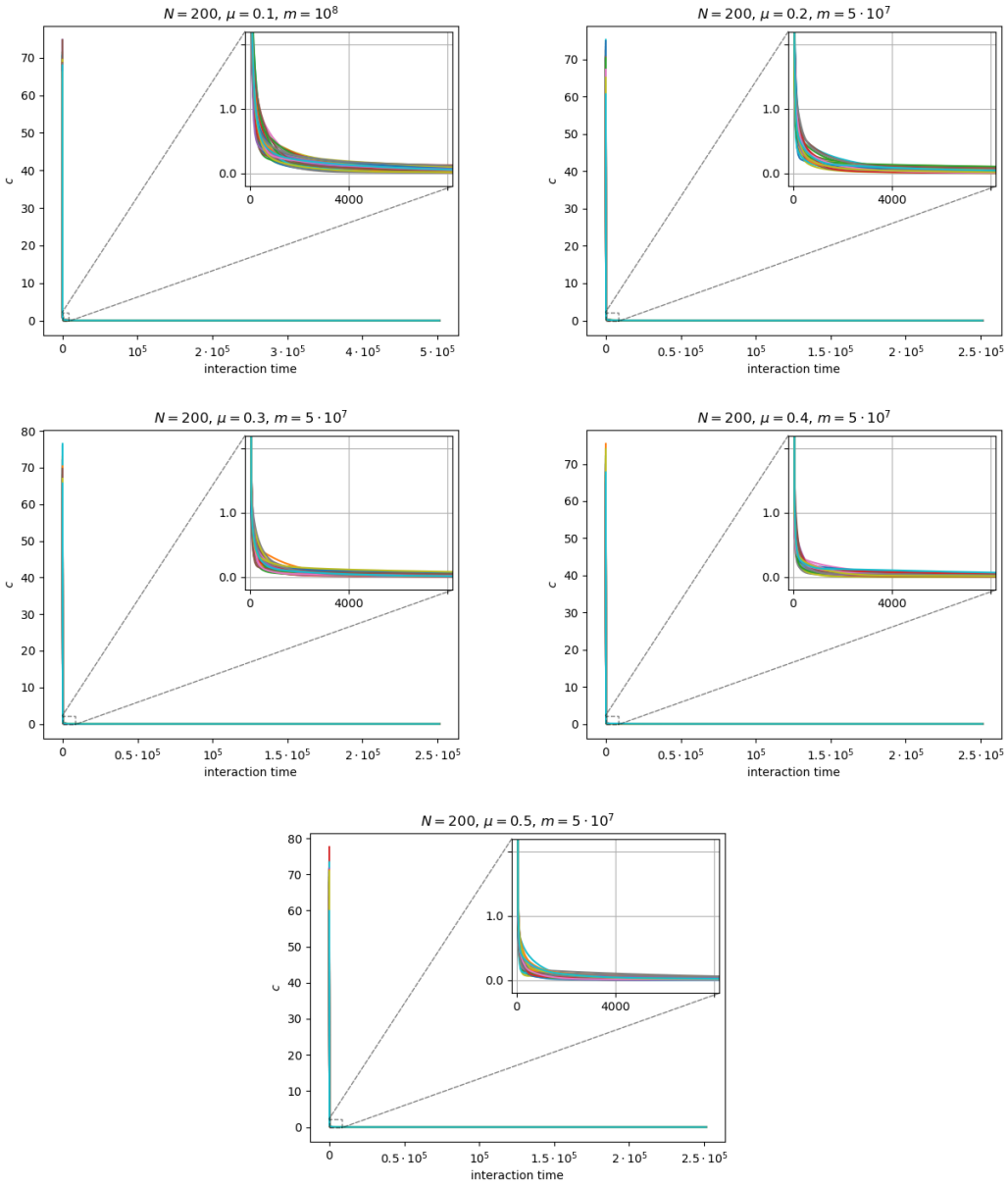

Figure 8.2.: The figure shows parameter c over time for the simulations of the Uniform Deffuant Model without noise. Each plot shows a simulation for a different value of μ, with every coloured line corresponding to a different interpretation of the simulation.

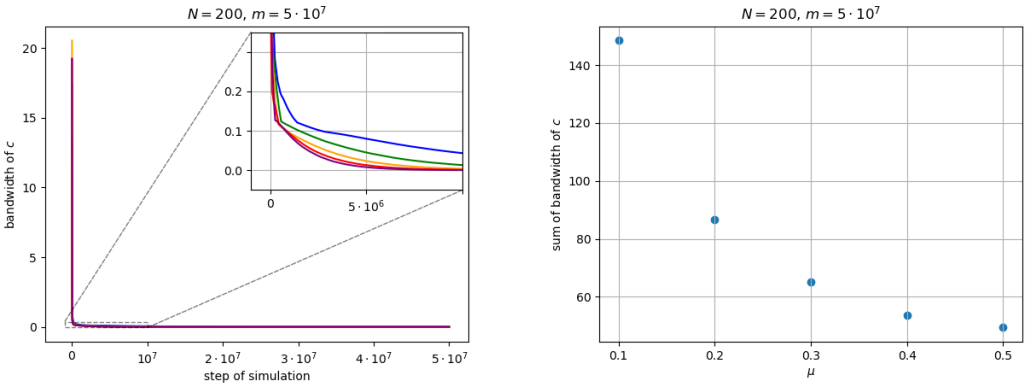

Figure 8.3.: On the left: Bandwidth of the values of c in the different simulations over the step of the simulation. The blue line corresponds to $\mu = 0.1$, green to $\mu = 0.2$, orange to $\mu = 0.3$, red to $\mu = 0.4$ and purple to $\mu = 0.5$.
On the right: the sum over the bandwidth values of c shown in the figure on the left.

As the interaction times are calculated using independent Poisson clocks on the vertices connecting neighbours as described in Algorithm 1, and because the expected waiting time between the rings of a Poisson clock is 1, the expected number of interactions per time step is the number of edges. We simulate 10^8 steps, thus whether we consider the bandwidth of c-values of time or over the interaction steps does not change the qualitative results much. We can therefore take this computational shortcut. We plotted this bandwidth in Figure 8.3. In it we see that the bandwidth of c decreases the longer the simulations run. The figure also confirms that this convergence is faster for larger values of μ.

Figure 8.4 gives visualised the order parameter r for the different simulations. For every simulation, r seems to converge quickly to 1. This convergence appears faster for larger μ-values. From checking the final values of the simulations, we know that all interpretations of all simulations eventually reach $r = 1$. This means, that in every simulation the individuals agree eventually.

The first time at which $r = 1$ for one interpretation of a simulation is again denoted by t_c. The mean and standard deviation for t_c are shown in Figure 8.5. It confirms that the time to reach consensus is on average longer for smaller values of μ. The standard deviation does not give a clear picture.

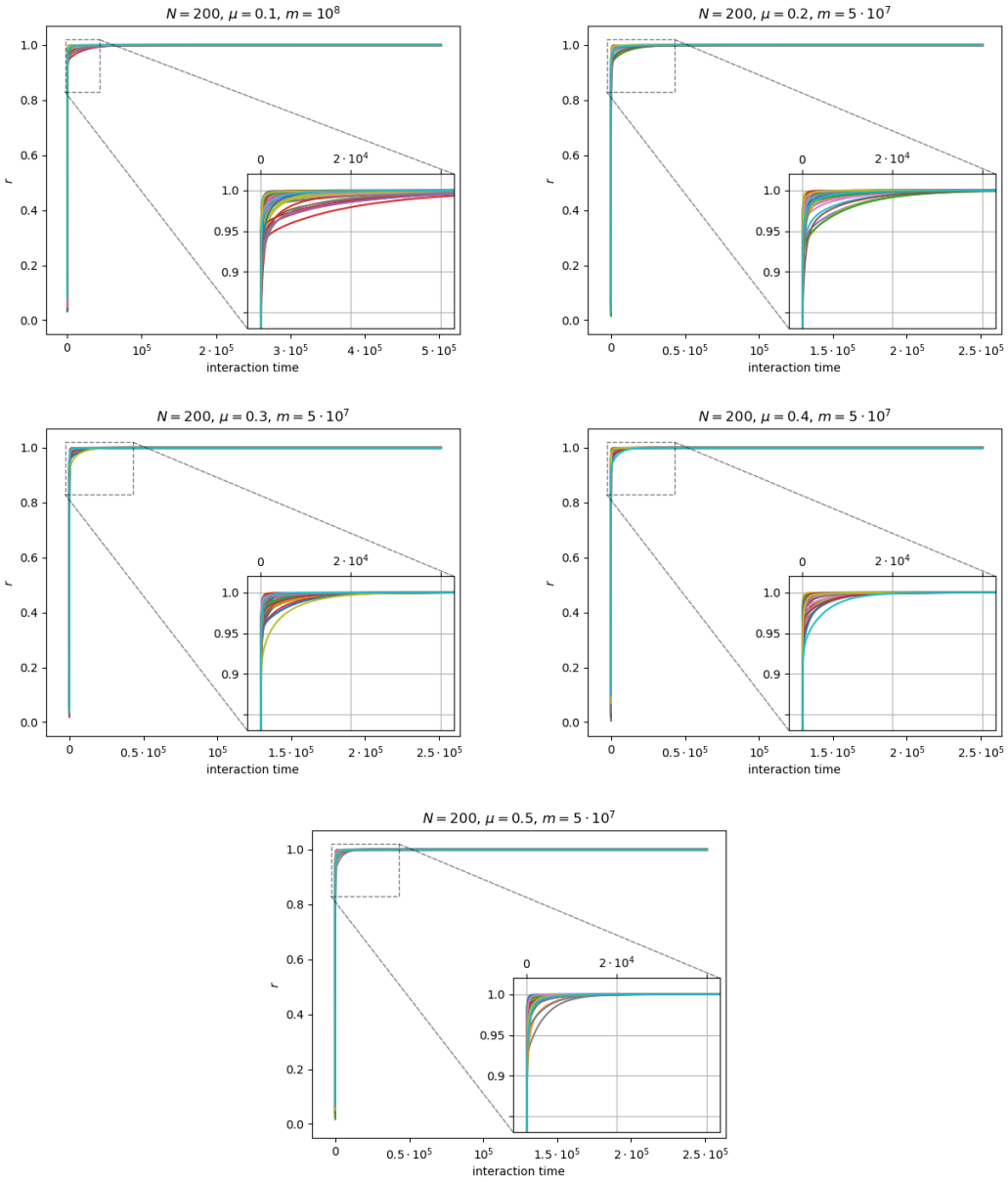

Figure 8.4.: This figure shows values of the order parameter r for simulations of the Uniform Deffuant model without noise. Each plot gives the results for a different value of μ with each coloured line corresponding to one interpretation of the simulation. The x-axis shows the time and the y-axis the value of r at that time.

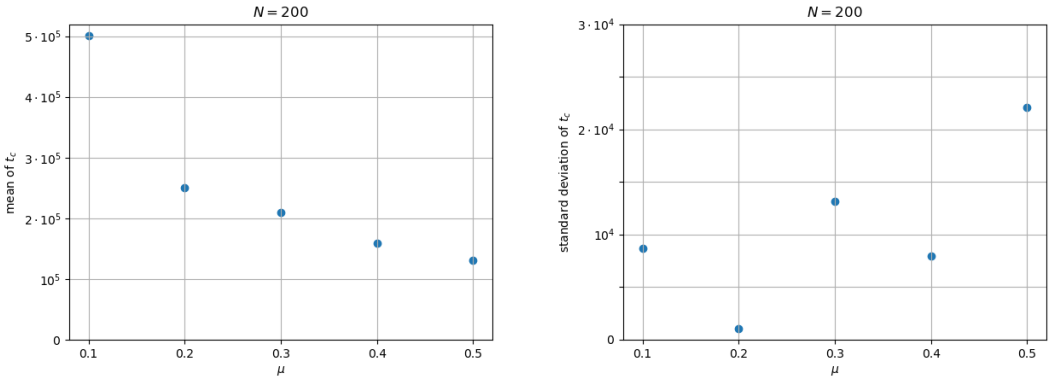

Figure 8.5.: The figures show the mean and standard deviation of t_c, that is the first time when $r = 1$ for an interpretation of a simulation of the Uniform Deffuant Model without noise, for different values of μ.

The Figure 8.6, visualises how the values of the order parameter θ evolve over time in our simulations. Because we have already seen that consensus will be reached, we know that the value shown for θ at the largest time that was plotted corresponds to the opinion that all the individuals eventually agree upon. We see can therefore see that the those final opinions are all relatively close to 0.5. This is what we expected, since we know from theory that the final opinions should be the average of the initial opinions.

In Figure 8.7, we plotted the final opinions of every simulations together with the average of the initial opinions. We can see that they match, confirming the theoretical result. This means that we have shown that the behaviour is as predicted by theory.

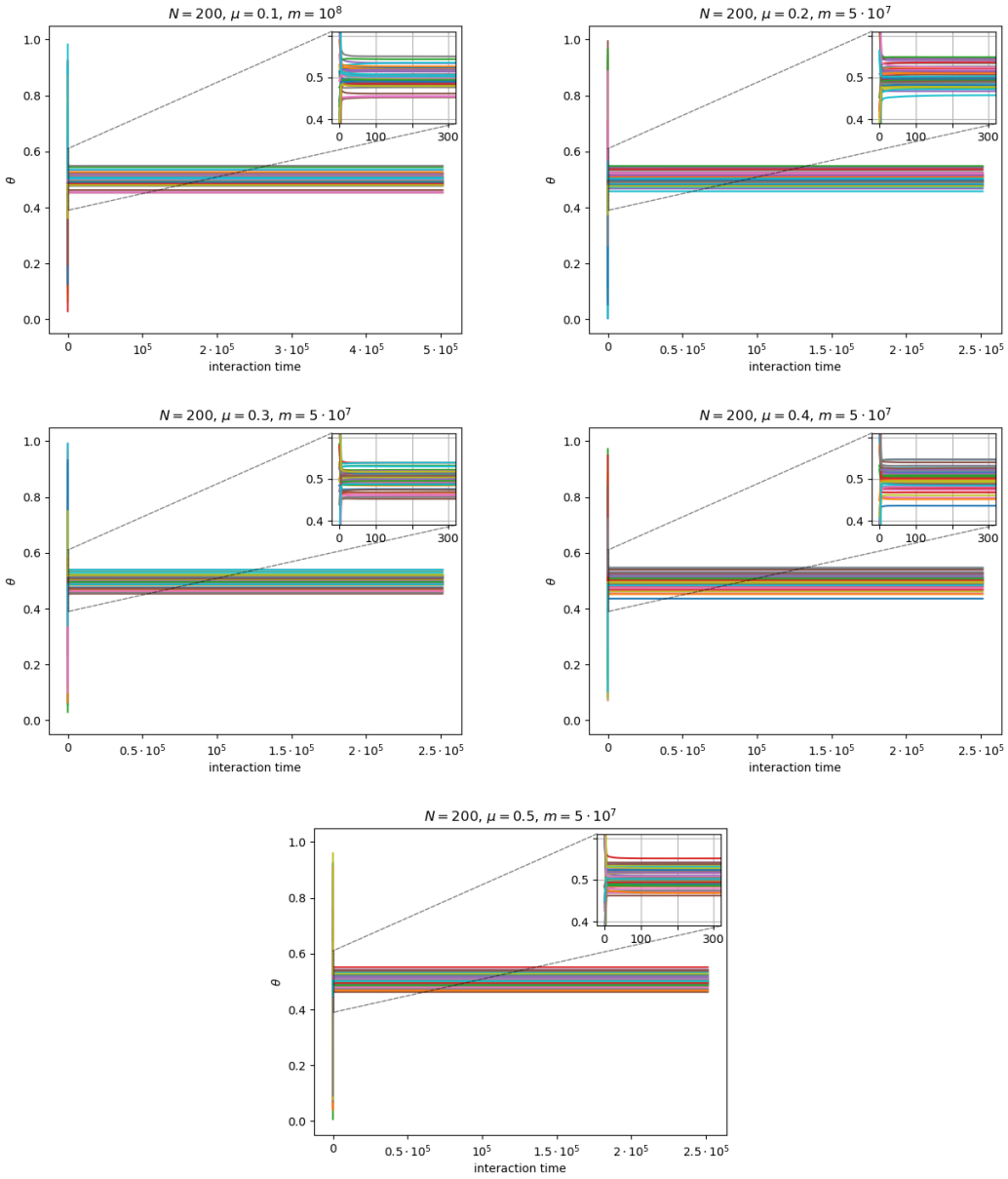

Figure 8.6.: These figures show how the values of the order parameter θ evolve over time for simulations of the Uniform Deffuant model without noise. Each one gives the results for a different value of μ with each coloured line corresponding to one interpretation of the simulation. The x-axis shows the time and the y-axis the value of θ at that time.

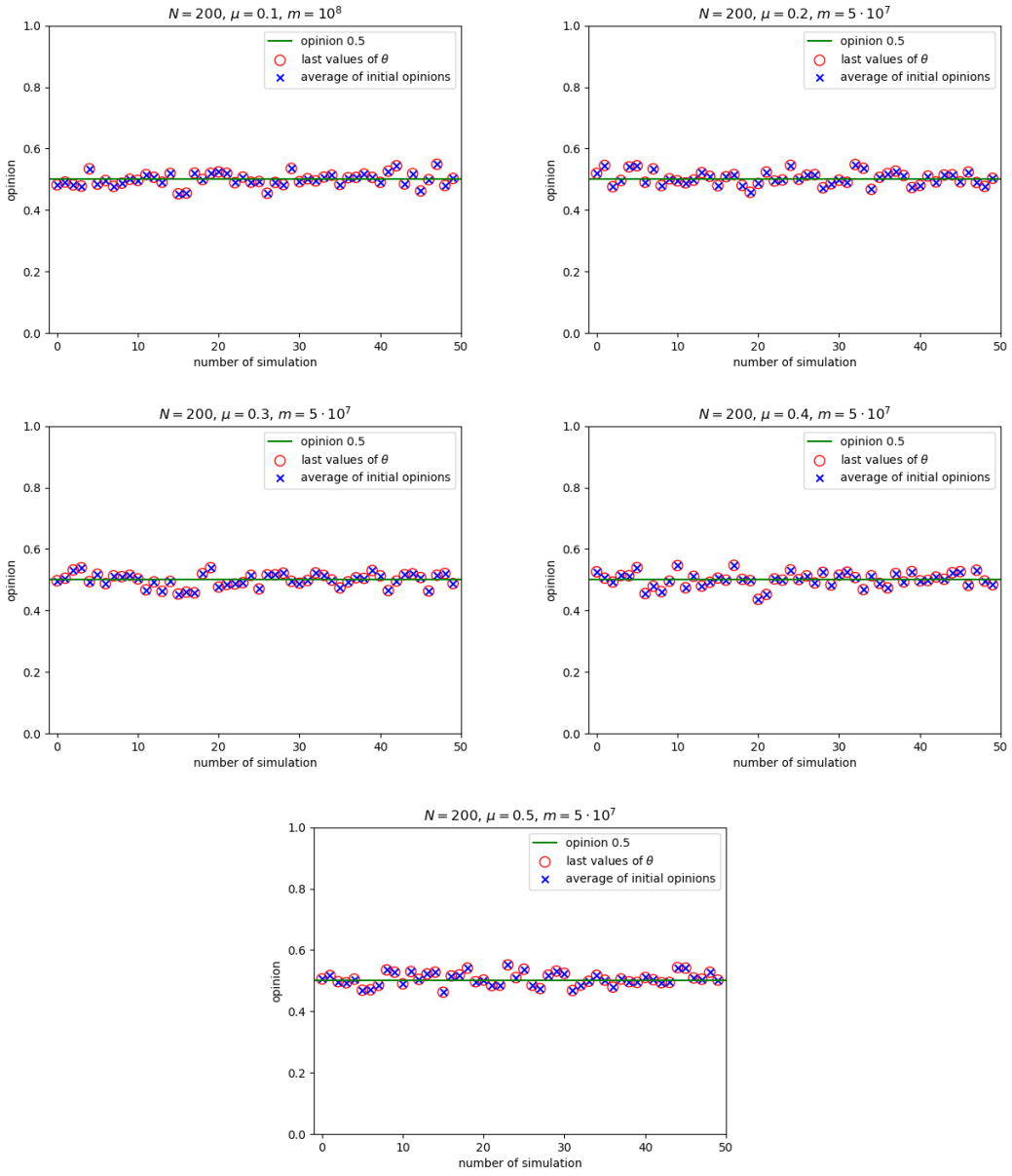

Figure 8.7.: The figures show the values that the opinions converge to in 50 runs of the simulation of the Uniform Deffuant Model for different values of μ. In each figure, the x-axis is gives the number of the interpretation of the simulation and the y-axis the value that the opinion converge to in that interpretation. All figures are for simulations with $N = 200$. The first figure is for $\mu = 0.1$, the second for $\mu = 0.2$, the third for $\mu = 0.3$, the forth for $\mu = 0.4$ and the fifth for $\mu = 0.5$.

8.2. Uniform Compass Model without Noise

The goal of this section is to illustrate the results of the numerical simulations of the Uniform Compass model (short: UCM) without noise. We do this by plotting various parameters as for example r, θ and c introduced in Chapter 7.

To simulate the Uniform Compass model, using the algorithm laid out in the previous chapter using a uniform initial distribution and a population size of $N = 200$. The simulations lasted $m = 10^8$ steps. This ensured, that the simulations had adequate time to converge. Values $0.1, 0.2, 0.3, 0.4$, and 0.5 where chosen for μ. Because of the randomness involved in drawing the initial opinions and calculating the Poisson clocks, we decided to run the model 50 times for each value of μ. This helps us to study the probabilistic properties of the model.

Figure 8.8 gives a selection of configurations at 6 different time steps in one simulation with $\mu = 0.5$. In it, we see how initially random opinions become increasingly synchronised until every individual agrees. We can see how the synchronisation of opinions is fast at first and then slows down.

In Figure 8.9, one can see the plots for parameter c, that is the sum of absolute opinion differences between neighbours. The graphics suggest that for a larger value of μ, c approaches 0 faster. When looking at the last values of the sum shown for every simulation, we see that they are all zero. This means that by the end of every interpretation of every simulation, all individual have the same opinion. Thus, consensus has been reached.

In Figure 8.9, we can see that the bandwidth of the values that c takes for different times and interpretations per simulation increases as the μ decreases. To confirm this, we calculated the bandwidth of c-values over the step of the simulations and plotted them in Figure 8.10.

The figure shows clearly that the bandwidth of c decreases the longer the simulations run. This is expected as c converges to 0 for all interpretations. The figure also show that this convergence is faster for larger values of μ as previously suspected.

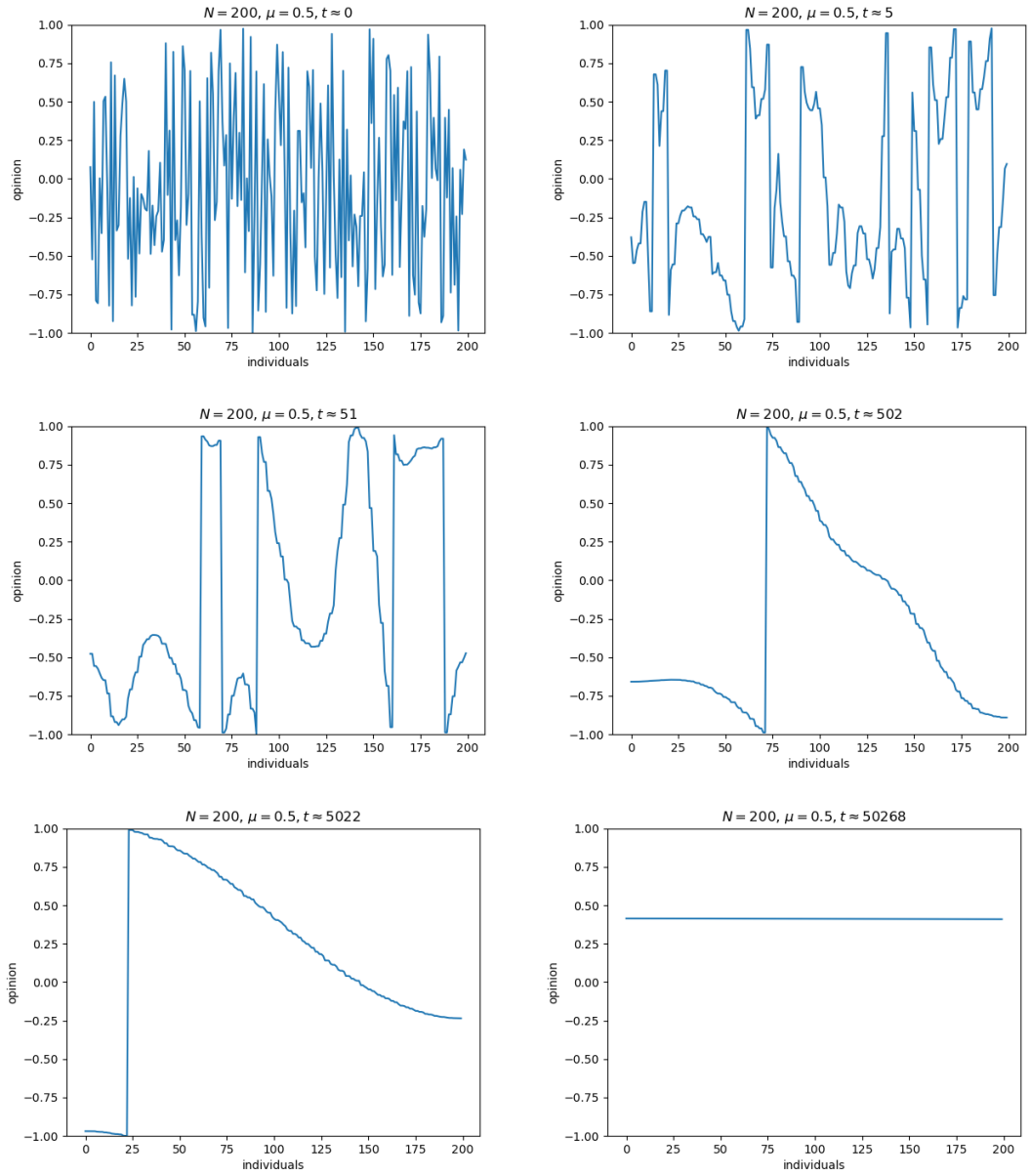

Figure 8.8.: This figure shows selected configurations that appeared in one simulation of the Uniform Compass Model without noise for $\mu = 0.5$. In each figure, the x-axis gives the number of the individual and the y-axis their opinion. The different opinions are connected by straight lines. The figure on the upper left corner shows the starting configuration. The configuration in the upper right hand corner shows the configuration at time 5. In the middle row, the configurations for time 53 and 502 are shown. The bottom row give the configuration at times 5022 and 50268.

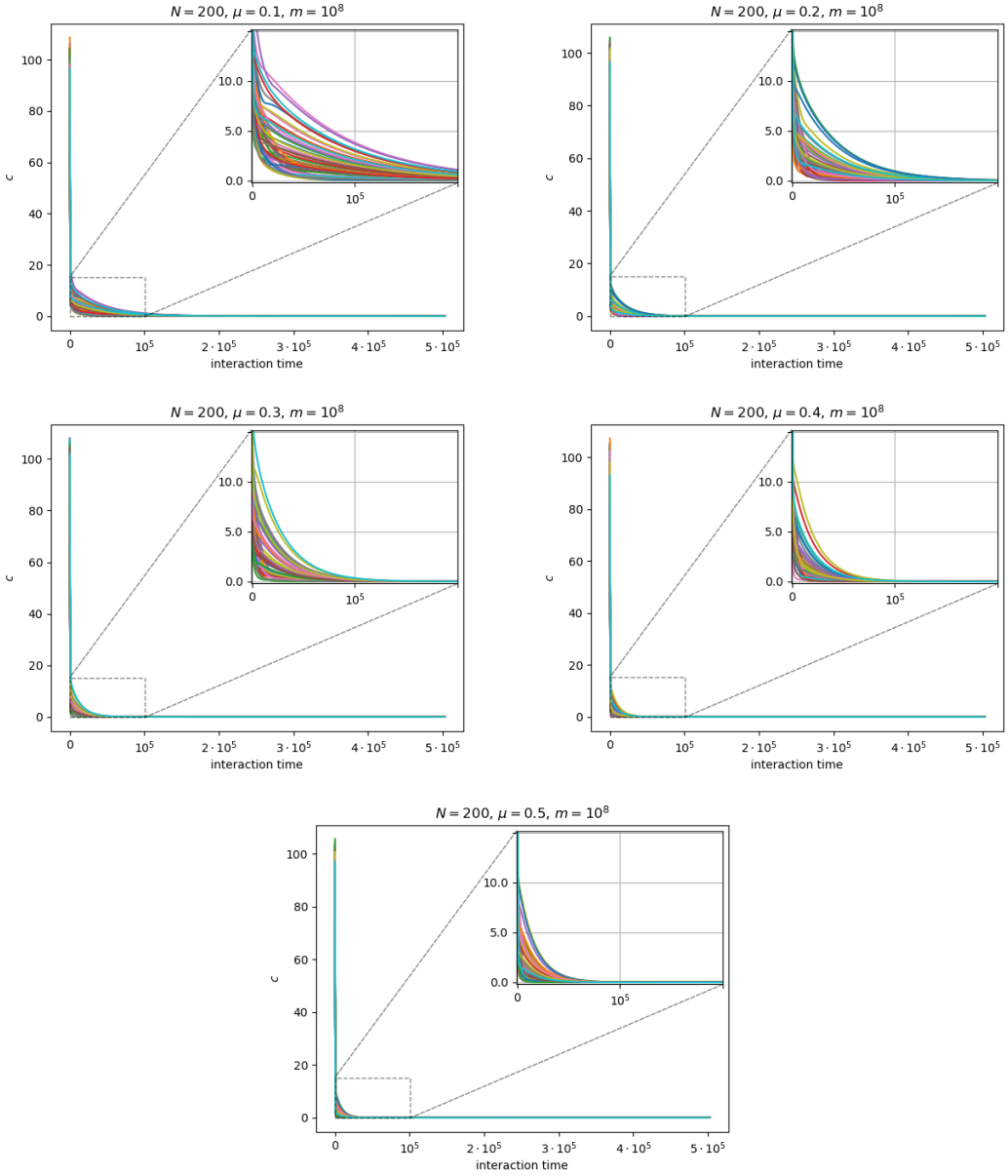

Figure 8.9.: The figures show the parameter c for the Uniform Compass model without noise over time. Every figure give the results for a different value of μ. In each figure, every coloured line represents on interpretation of the model.

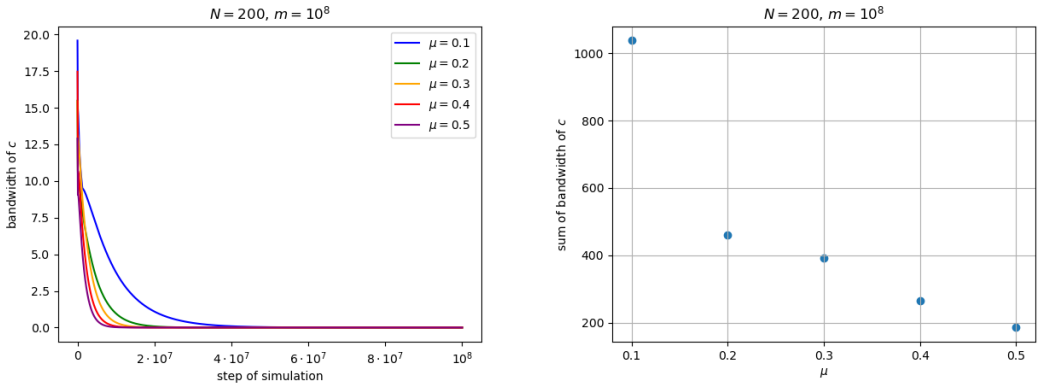

Figure 8.10.: On the left: Bandwidth of the values of c over the step of the simulation. The different colours stand for the different values of μ used in the simulation.
On the right: the sum over the bandwidth values of c shown in the figure on the left.

In the previous figures we saw that all neighbours will eventually agree. Thus, we wonder which opinions they finally agree upon. These opinions are visualised as dots on a circle in Figure 8.11. These opinions seem to be distributed uniformly. This is precisely the behaviour that we expected from Chapter 5.

To confirm that the final opinions are uniformly distributed, we perform a two-sided Kolmogorov-Smirnov test (KS-test), as introduced in Appendix A, on the data. We perform the test separately on each set of 50 final θ-values obtain in the simulations of the UCM for one specific value of μ. In this setting, we assume that the final θ-values are independent and identically distributed random variables $X_1, \ldots, X_n$ with cumulative distribution F. We call the empirical distribution function on the values we actually observe in our simulation F_n.

Our null-hypothesis H_0 is the hypothesis that the data is uniformly distributed on the circle (i.e. the interval $(-1, 1]$). The alternative hypothesis H_A is that the null-hypothesis is wrong. This means we test the following:

$$\begin{aligned} H_0 : &\quad F = F_n \\ H_A : &\quad F \neq F_n. \end{aligned}$$

We can reject the null-hypothesis at significance level $\alpha \in [0, 1]$, if the p-value obtained by the KS-test is less or equal to α.

Table 8.1 gives the test statistic and the p-value of the KS-test for every simulated value of μ. As one can see, at significance level 0.1, we accept the null-hypothesis for each value of μ. This means that we do not have any significant reason to suspect that our final θ-values are not uniform.

More information on how the KS-test works can be found in Appendix A.

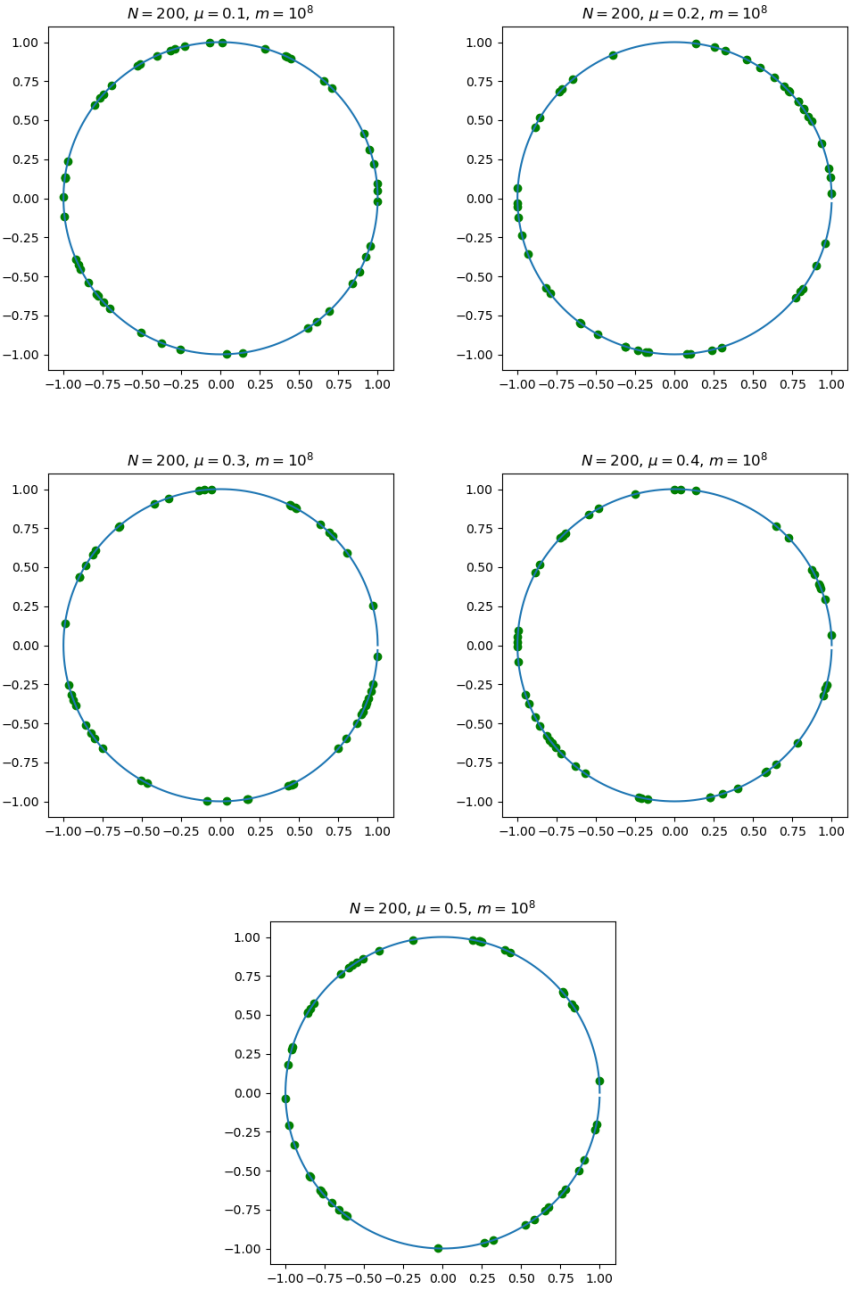

Figure 8.11.: The figures show the values that the opinions converge to in 50 interpretations of the simulations of the Uniform Compass Model for different values of μ. In each figure, every dot represents the opinion that the individuals eventually agree on for one interpretation of the simulation rescaled to lie on a circle of radius 1. The first figure is for $\mu = 0.1$, the second for $\mu = 0.2$, the third for $\mu = 0.3$, the forth for $\mu = 0.4$ and the fifth for $\mu = 0.5$.

μ-value	0.1	0.2	0.3	0.4	0.5
p-value	0.707	0.321	0.878	0.681	0.731
test statistic	0.096	0.132	0.080	0.107	0.094

Table 8.1.: Every column gives the p-value and test statistic obtained from the two-sided Kolmogorov-Smirnov test on the set of final θ-values for 50 simulations of the Uniform Compass Model without noise with the specified μ-value.

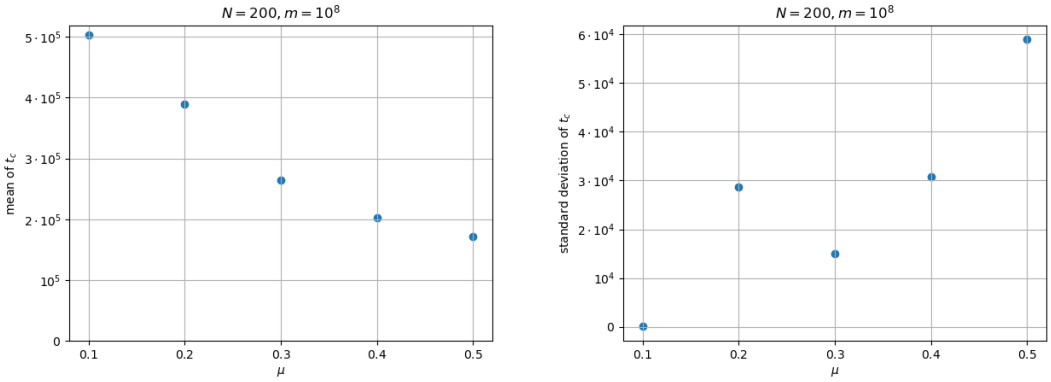

Figure 8.12.: The figure on the right shows the mean and the one on the left the standard deviation of t_c, that is, the first time at which $r = 1$, as observed for the Uniform Compass Model without noise for different values of μ.

In addition to the sum of the absolute opinion differences between neighbours, we also computed the order parameters r and θ as described in the previous chapter. First let us look at the order parameter r. Figure 8.13 shows the results for the parameter r and Figure 8.16 visualises the results for order parameter θ. We can see that for all values of μ, r converges to 1 meaning that the individuals agree. This confirms what we have seen before. As before, we see faster convergence of opinions for higher values of μ.

Figure 8.12 visualises the times when consensus is reached. As we can assume that we have consensus once $r = 1$, it gives the mean and standard deviation of the first times t_c when $r = 1$ is observed in an interpretation of a simulation for a specific value of μ. This figure supports the observation that the larger μ the faster consensus is reached. However, the standard deviation of t_c is much larger for $\mu = 0.5$ than for $\mu = 0.1$. However the relationship does not appear to be linear. In order to see why this is the case, a more thorough analysis would be needed. In particular, it would be interesting to see whether it persists if one chooses to do more than 50 samples per simulation.

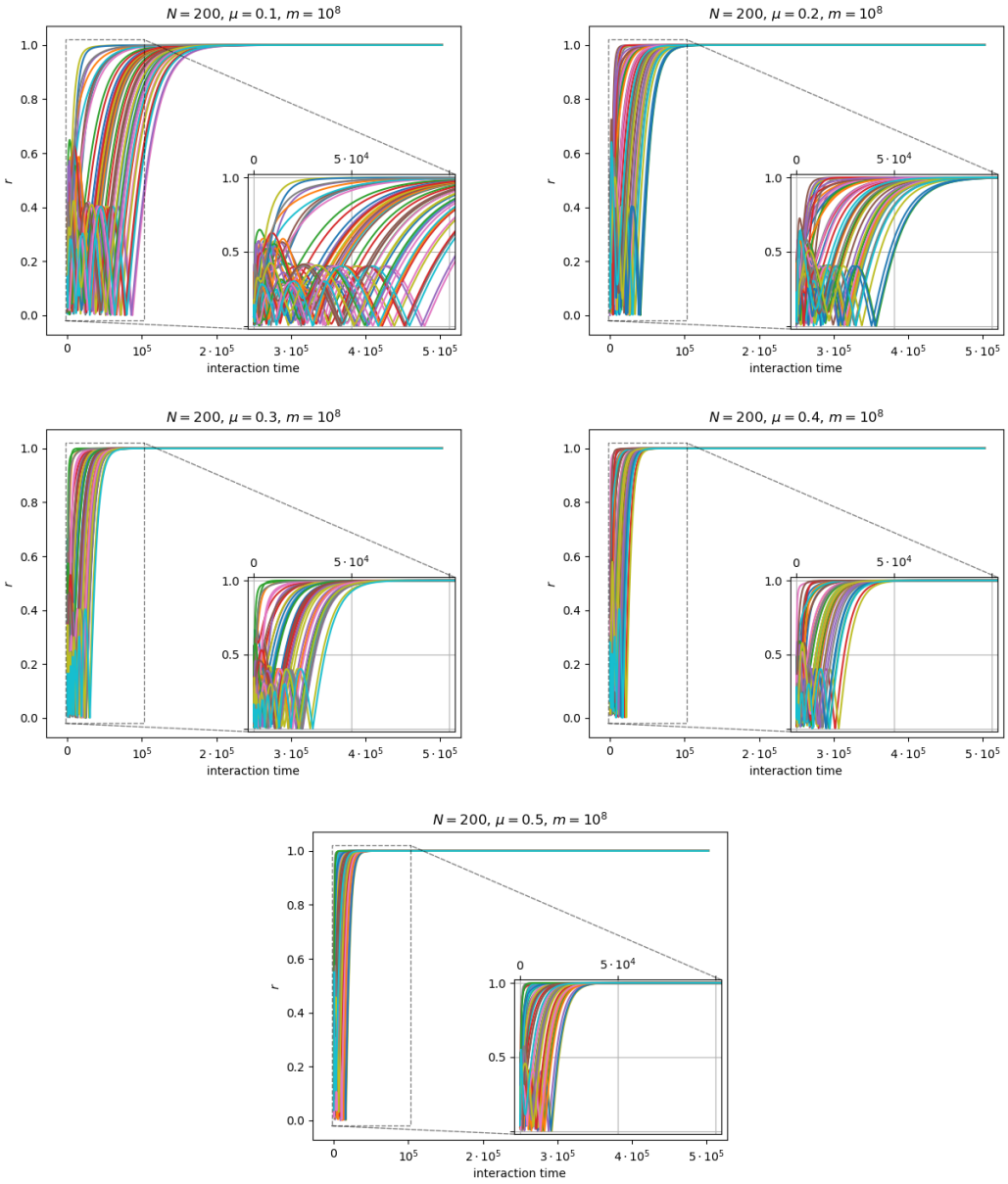

Figure 8.13.: Each figure shows the value of the order parameter r over time for the Uniform Compass Model without noise for 50 interpretations each. Every figure gives the results for a simulation with a different value of μ. In the graphics every colour corresponds to one interpretation.

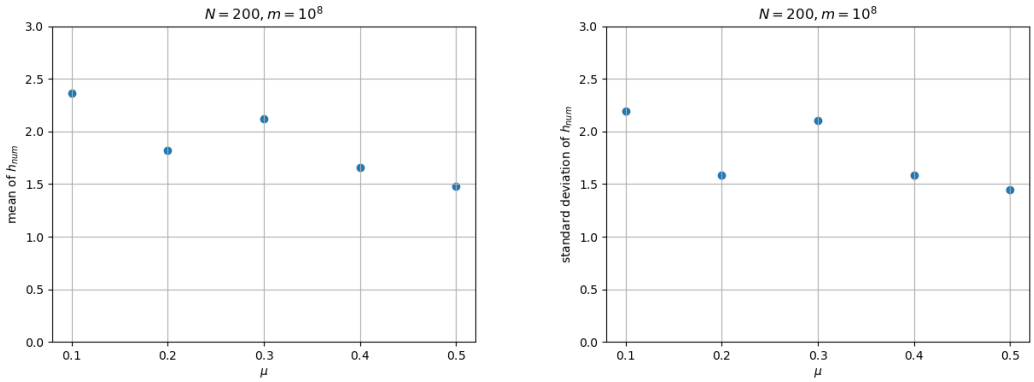

Figure 8.14.: Mean and standard deviation of h_{num}, that is the number of significant local maxima (jumps), as observed for the Uniform Compass Model without noise dependent on μ.

We observe a number of local extrema in the plots of r. It seems like the number of these extrema decreases as μ increases. However, upon calculating the number of significant local maxima h_{num}, that is those that have enough distance to their neighbouring local maxima and are significantly higher than the local minimum next to them, this suspicion is not confirmed. The mean and standard deviation of h_{num} for different values of μ are visualised in Figure 8.14. Looking at the configurations that correspond to these local maxima and minima did not give further insight into why r has these local extrema.

As we saw in Section 8.1, these jumps do not appear in the Uniform Deffuant Model (see Figure 8.4). Even when we calculate r at at all steps, we do not observe significant local maxima. The difference in these models is the way the opinion is modelled. For the Uniform Compass Model opinions lie on a circle and for the Uniform Deffuant Model they lie on a finite interval. This made us suspect that this difference might come from the fact that when we calculate with numbers on a circle we need to do modulo calculation, which is not necessary in the case of the Deffuant Model. The Kuramoto model also represents opinions as numbers on a circle and, thus, also uses modulo calculation. However, in this model the jumps are not observed as seen in the paper [CR16]. This made us suspect that the difference comes from rounding errors that are cause by the way we carry out the modulo operation.

In order to investigate this we carry out a simulation using different methods of doing the modulo calculation but using the same initial condition, interaction times and interaction places. We call the method used in our original simulation `mod1`. The same method, just for a circle parameterised as $[0, 2)$ is called `mod2`. These two methods are defined in Algorithms 10 and 11. The standard modulo operator % in Python is referred to as % and the version of the modulo operation from the numpy library, `numpy.fmod`, is simply referred to as `fmod`. `fmod_adjusted` is a customised version of `fmod` that ensures that the result always lies in $[0, 2)$. This method is shown in Algorithm 12.

Algorithm 10 `mod1`

Input a
if $a > 1.0$ **then**
  $a \leftarrow a - 2.0$
end if
if $a < -1.0$ **then**
  $a \leftarrow a + 2.0$
end if
Return a

Algorithm 11 `mod2`

Input a
if $a > 2.0$ **then**
  $a \leftarrow a - 2.0$
end if
if $a < 0.0$ **then**
  $a \leftarrow a + 2.0$
end if
Return a

Method	Final Opinion
`mod1`	0.4116734986520993
`mod2`	1.411673498652099
%	1.411673498652099
`fmod`	1.8953899075572094
`fmod_adjusted`	1.411673498652099

Table 8.2.: For every method for making modulo calculations on the left hand side, the right hand side gives the value that the opinions converge to.

Algorithm 12 `fmod_adjusted`

Input a
if $a < 0.0$ **then**
  $a \leftarrow a + 2.0$
else
  $a \leftarrow$ `fmod`(a)
end if
Return a

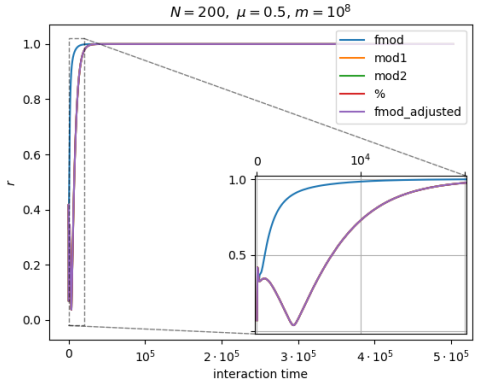

Figure 8.15.: This figure shows the plots of the order parameter r as calculated while using different methods for the modulo calculation, indicated by different colours (see legend). The lines are plotted in the order as they appear in the legend. Thus, the orange, green, red and purple line are almost identical.

As seen in Table 8.2 and Figure 8.15, the simulation using `mod1`, `mod2`, % and `fmod_adjusted` give nearly identical results. However, the simulation using `fmod` gives significantly different results. As it turns out, the result using `fmod` is wrong. The reason is that this version returns the answers as numbers in $(-2.0, 2.0]$, which leads to problems with the calculations in the following step. To visualise this, consider the following example. Note that -0.5 and 1.5 are the same numbers modulo 2.0 and they could both be opinions obtained in the previous step in the simulation using `fmod`. However, the calculations $-0.5 \cdot 1.5 + 0.3$ and $1.5 \cdot 1.5 + 0.3$ do not have the same results modulo 2. In all the other methods of modulo calculation, this problem is avoided, because the result always lies within the interval $(-1.0, 1.0]$ or $(0.0, 2.0]$.

This means that the local maxima observed in the order parameter r do not appear to be cause by the modulo calculation. Further analysis beyond the scope of this project is needed to see why these show up. We therefore end this excursion here and return to studying other properties of the Compass model.

The results for the order parameter θ are given in Figure 8.16. As we can see, θ quickly approximates the final value that it will take at the end of the simulation. The larger μ, the faster this happens. We know that if the neighbours all agree upon the same opinion, then this opinion is equal to θ. From the results for the parameters r and θ, we know that this will eventually happen. Thus, the final values of θ should be the same as the values that the individuals agree upon. We confirmed this by comparing both vectors.

To summarise, in this section we saw that the larger μ, the faster the individuals come to consensus. We also saw that with our uniform initial conditions, the opinions that the individuals end up agreeing upon are also uniformly distributed. This matches the theoretical results. Furthermore, we observed some unexplained behaviour in the parameter r for small times.

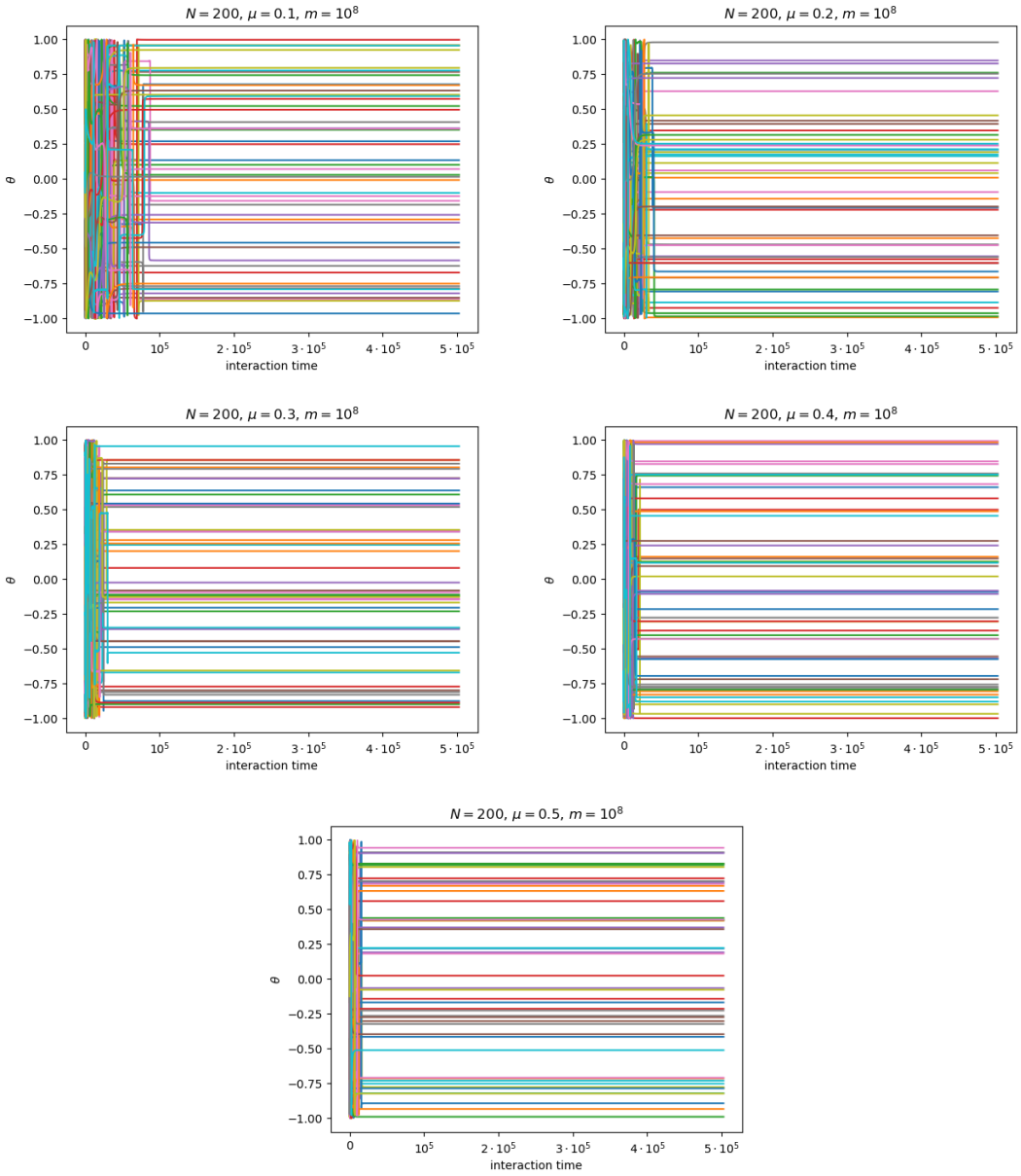

Figure 8.16.: Each figure shows the value of the order parameter θ over time for the Uniform Compass Model without noise for 50 interpretations each. Every figure gives the results for a simulation with a different value of μ. In the figures every colour corresponds to one interpretation.

8.3. Uniform Deffuant Model with Noise

The purpose of this section is to study how the Uniform Deffuant model behaves when it is subjected to uniform and bi-modal noise. Does this kind of noise change the behaviour of the model significantly? How will it compare to the Uniform Compass model with noise? How does the noise parameter ϵ influence the results?

8.3.1. Uniform Noise

The first kind of noise that we consider is uniform noise. To study its effect we simulate the Deffuant model with uniform noise as described in Algorithm 6. In all simulations we use uniform initial conditions, $N = 200$, and $\mu = 0.5$. We study the ϵ-values $0.1, 0.01, 0.001, 0.0001, 10^{-5}$, and 10^{-6}. For each simulations we again run 50 simulations in order to be able to study the randomness involved in the model.

We expect that for large ϵ the consensus cannot be reached. For those ϵ, we expect very uneven configurations, small values of r and large c. The smaller ϵ gets, the less the noise influences the opinion formation. We, thus, expect an increasing degree of synchronisation for smaller ϵ-values. Because the noise is drawn from a symmetric uniform distribution, we expect that if ϵ is sufficiently small to reach approximate consensus, it does not change the opinion that the individuals converge to. This means that we expect that the opinions of individuals still converge to approximately 0.5.

We start by looking at the values of the parameter c over time shown in Figure 8.17. As one can see, the smaller ϵ gets, the closer c gets to 0 and the less it oscillates.

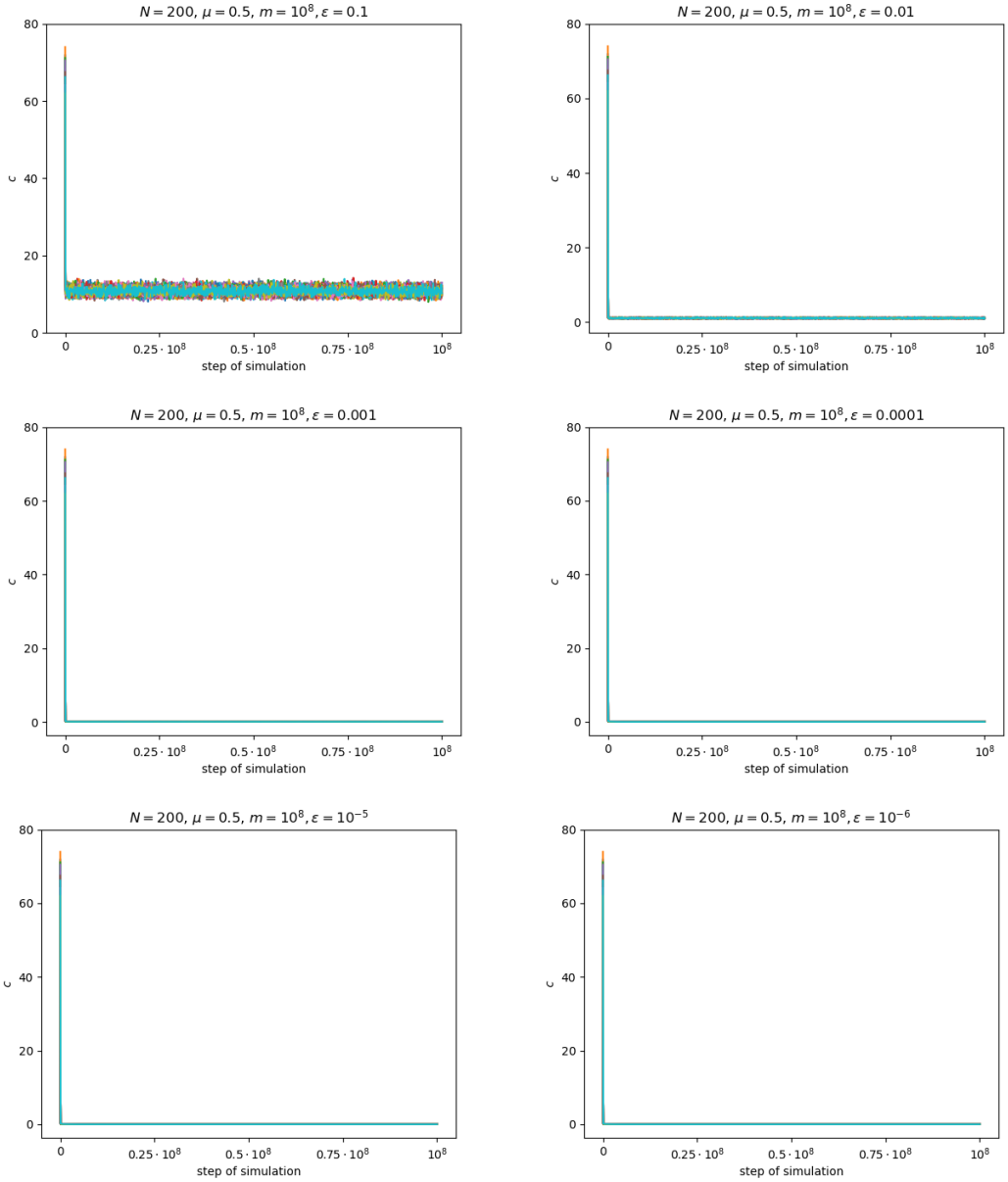

Figure 8.17.: These figures show the parameter c over the steps of the simulation for simulations of the Uniform Deffuant model with uniform noise for different values of ϵ. The different values of ϵ can be found in the headers of the respective figures. Every coloured line corresponds to one simulation.

We go on to study the order parameter r. We see that it is very noisy for large ϵ. For smaller ϵ values, the graphs get increasingly less noisy and seem to converge to 1. This suggests a high degree of synchronisation, approximate consensus even, for small ϵ.

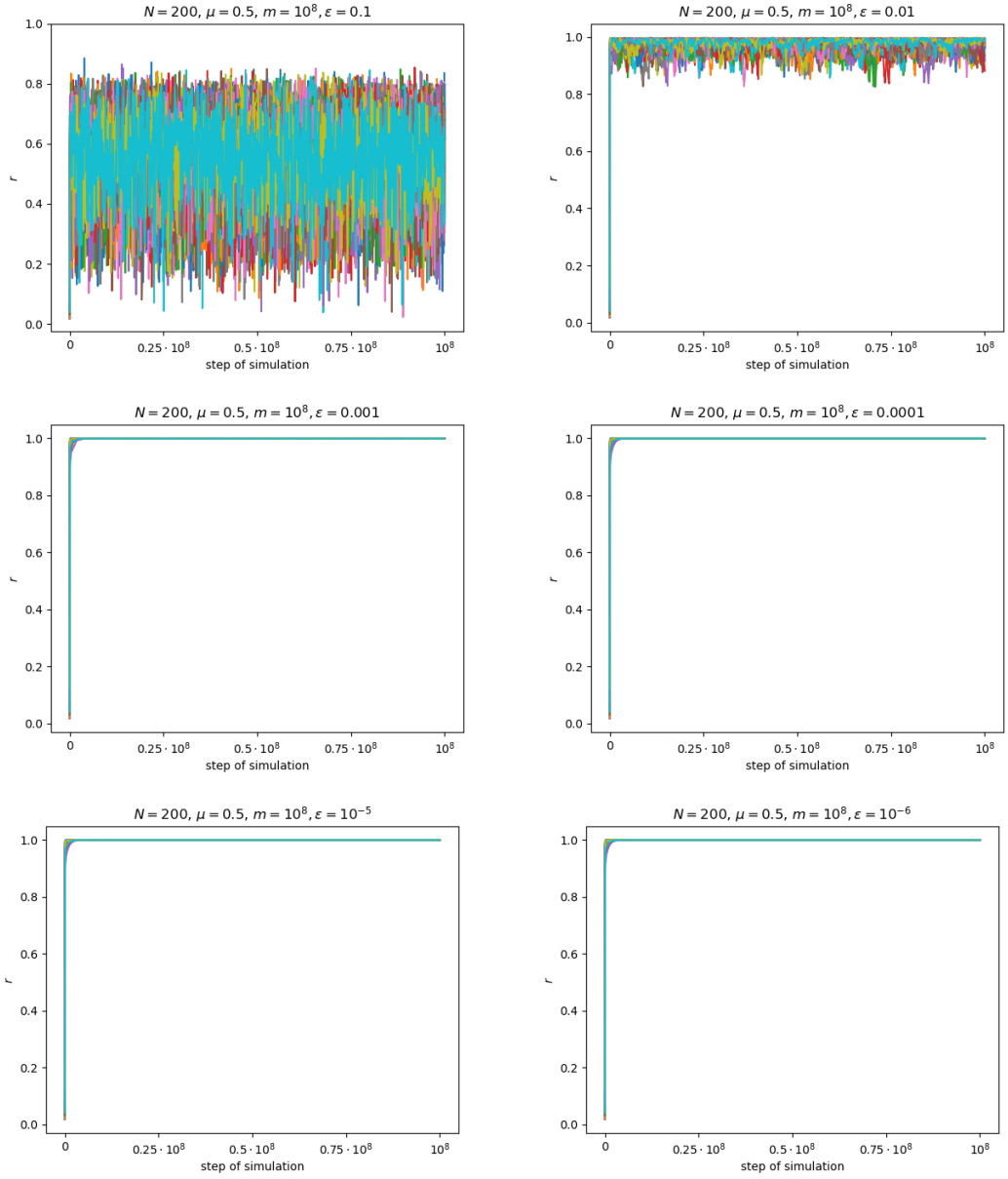

Figure 8.18.: These figures show the order parameter r over steps of the simulation for simulations of the Uniform Deffuant model with uniform noise for different values of ϵ. The different values of ϵ can be found in the headers of the respective figures. Every coloured line corresponds to one simulation.

ϵ	mean of bandwidth of last configuration	standard deviation of bandwidth of last configuration
0.1	$7.90 \cdot 10^{-1}$	$9.07 \cdot 10^{-2}$
0.01	$1.26 \cdot 10^{-1}$	$4.12 \cdot 10^{-2}$
0.001	$1.23 \cdot 10^{-2}$	$3.44 \cdot 10^{-3}$
0.0001	$1.25 \cdot 10^{-3}$	$3.87 \cdot 10^{-4}$
10^{-5}	$1.32 \cdot 10^{-4}$	$3.69 \cdot 10^{-5}$
10^{-6}	$1.25 \cdot 10^{-5}$	$3.75 \cdot 10^{-6}$

Table 8.3.: This table shows the mean and standard deviation of the bandwidth of the last configurations calculated in the simulations of the Uniform Deffuant Compass model with uniform noise for different values of ϵ.

The values of the corresponding order parameter θ are visualised in Figure 8.19. The individual lines are increasingly noisy for increasingly large ϵ indicating that the opinions change more for larger ϵ values. This confirms the results for c and r. As expected, the values are mostly clustered around 0.5. However, curiously, for $\epsilon = 0.01$ and $\epsilon = 0.001$ the values of θ seem to diverge away from $\theta = 0.5$.

The last configurations calculated in the simulations are shown in Figure 8.20. As expected the configurations look more even for smaller ϵ and are mostly clustered around the opinion 0.5.

By calculating the mean and standard deviation of the bandwidth of opinions of the last configurations simulated for different values of ϵ, we can see that the individuals indeed get closer and closer to reaching consensus when ϵ gets close to 0. The mean and standard deviation of the bandwidth for different values of ϵ are shown in Table 8.3.

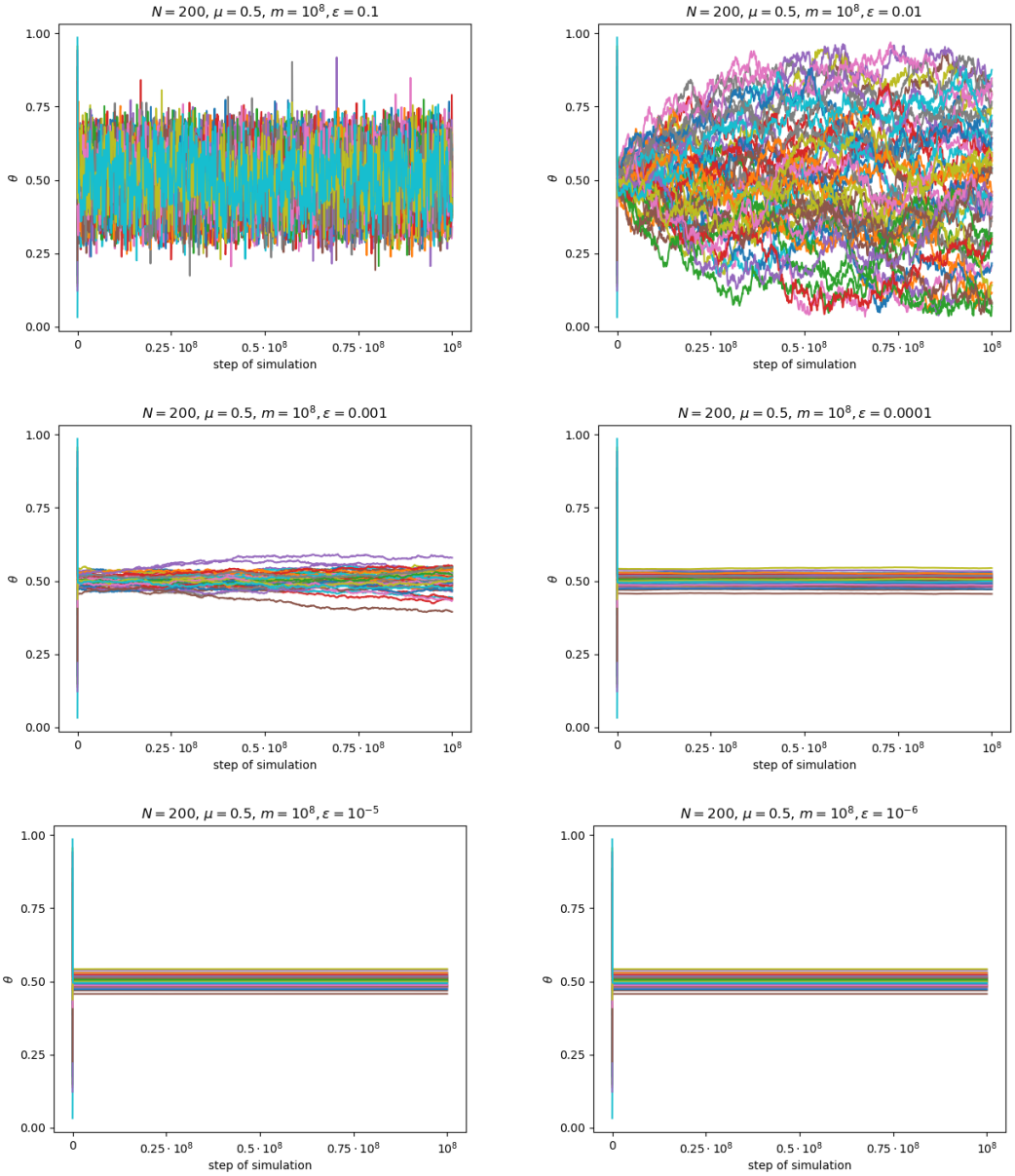

Figure 8.19.: Here we can see the order parameter r over steps of the simulation for simulations of the Uniform Deffuant model with uniform noise for different values of ϵ. Every coloured line corresponds to one simulation. In the headers of the respective figures, the different values of ϵ can be found.

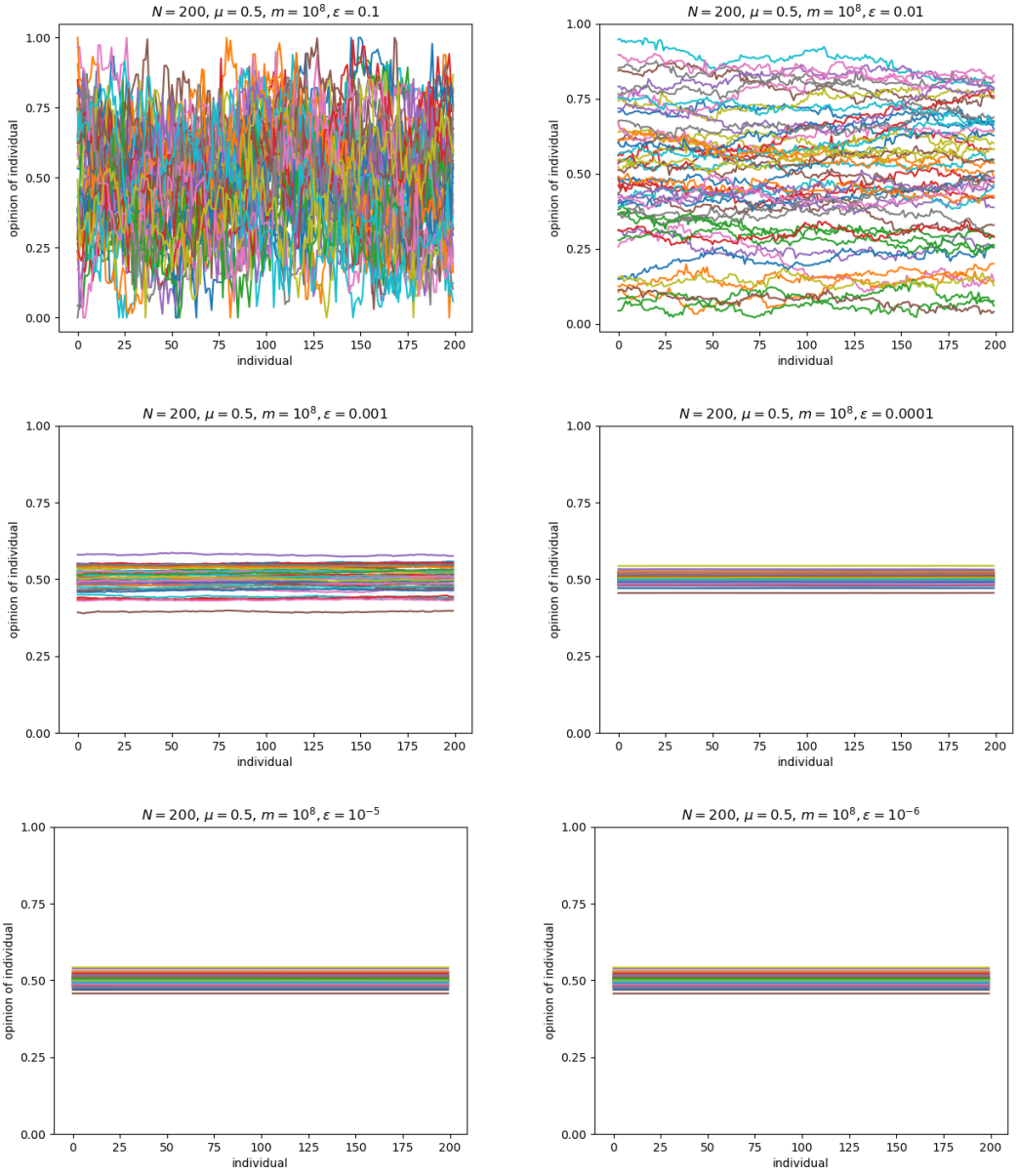

Figure 8.20.: The last configurations simulated in simulations of the Uniform Deffuant model with uniform noise for different values of ϵ are shown in these graphs. Every coloured line corresponds to one simulation. The different values of ϵ can be found in the headers of the respective figures.

8.3.2. Bi-Modal Noise

In this sections we aim to answer the question what effect bi-modal noise has on the Uniform Deffuant model. As already seen in Section 8.1, without noise, the Uniform Deffuant model tends to produce consensus on opinions close to 0.5. Furthermore, the dynamics of the bi-modal noise will work towards approximate consensus on 0.5. We therefore expect these tendencies to support one another, resulting on fast approximate consensus for smaller ϵ. For larger ϵ we expect the configurations to be in a larger interval around ϵ after a few time-steps.

In order to study the behaviour, we simulated the model as described in Algorithm 9. For all simulations we used $N = 200$, $m = 10^8$ and $\mu = 0.5$. Furthermore, we looked at $\epsilon = 0.5, 0.1, 0.05, 0.01, 0.005$ and 0.001. We ran 50 simulations per set of parameters. The results are described and visualised below.

Figure 8.21 visualises the parameter c for the simulations. We see that the smaller ϵ, the more c approaches 0 and the less it will variate.

Similarly, r gets closer and closer to 1 with decreasing variations for smaller ϵ. This is shown in Figure 8.22.

The figure giving the results for the parameter θ, Figure 8.23, shows that θ quickly ends up in an interval around 0.5. The smaller ϵ is, the smaller this interval becomes.

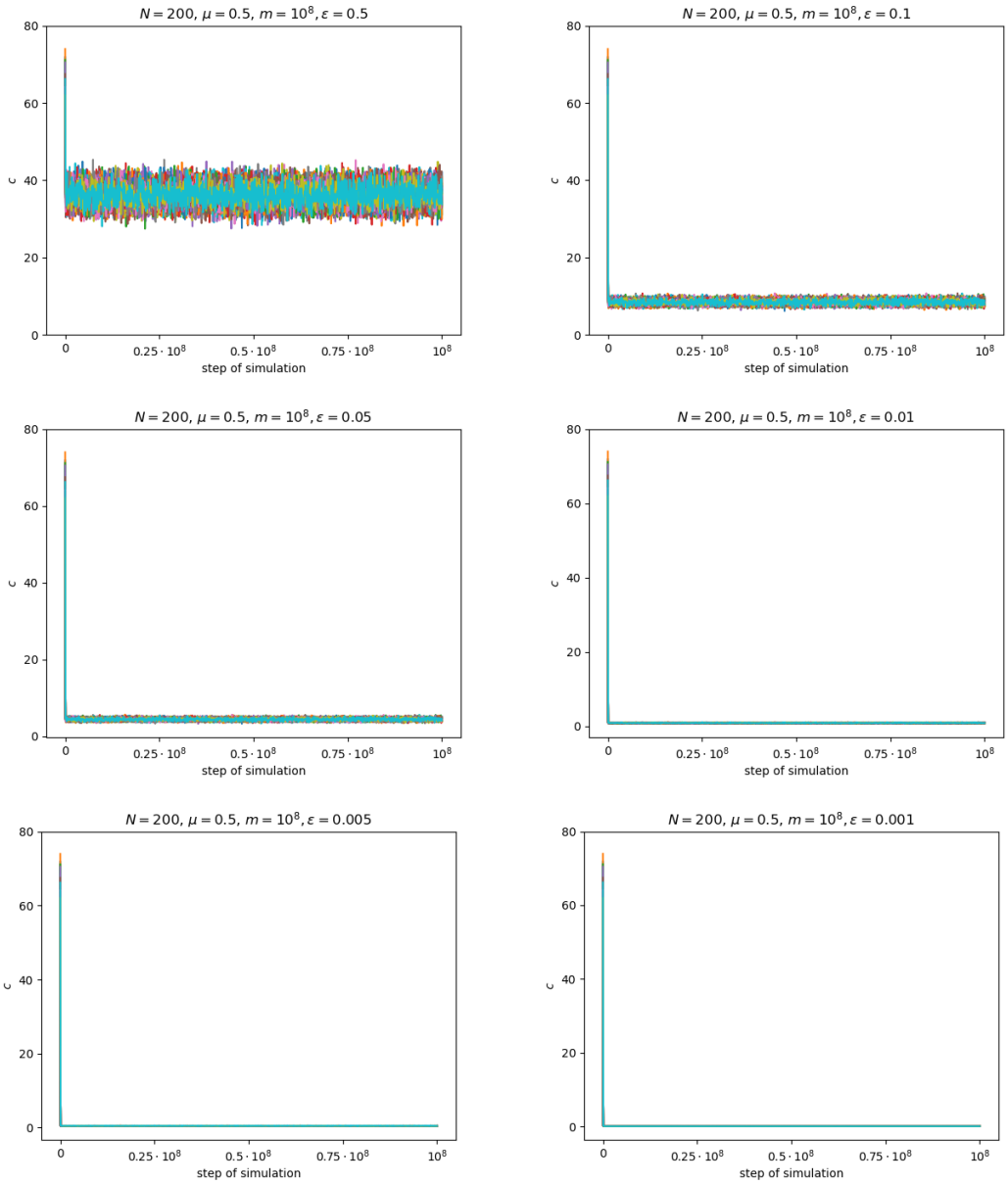

Figure 8.21.: These figures show results for the parameter c for the Uniform Deffuant model with bi-modal noise. The simulations all used the parameters indicated in the respective headers. All lines correspond to individual simulations.

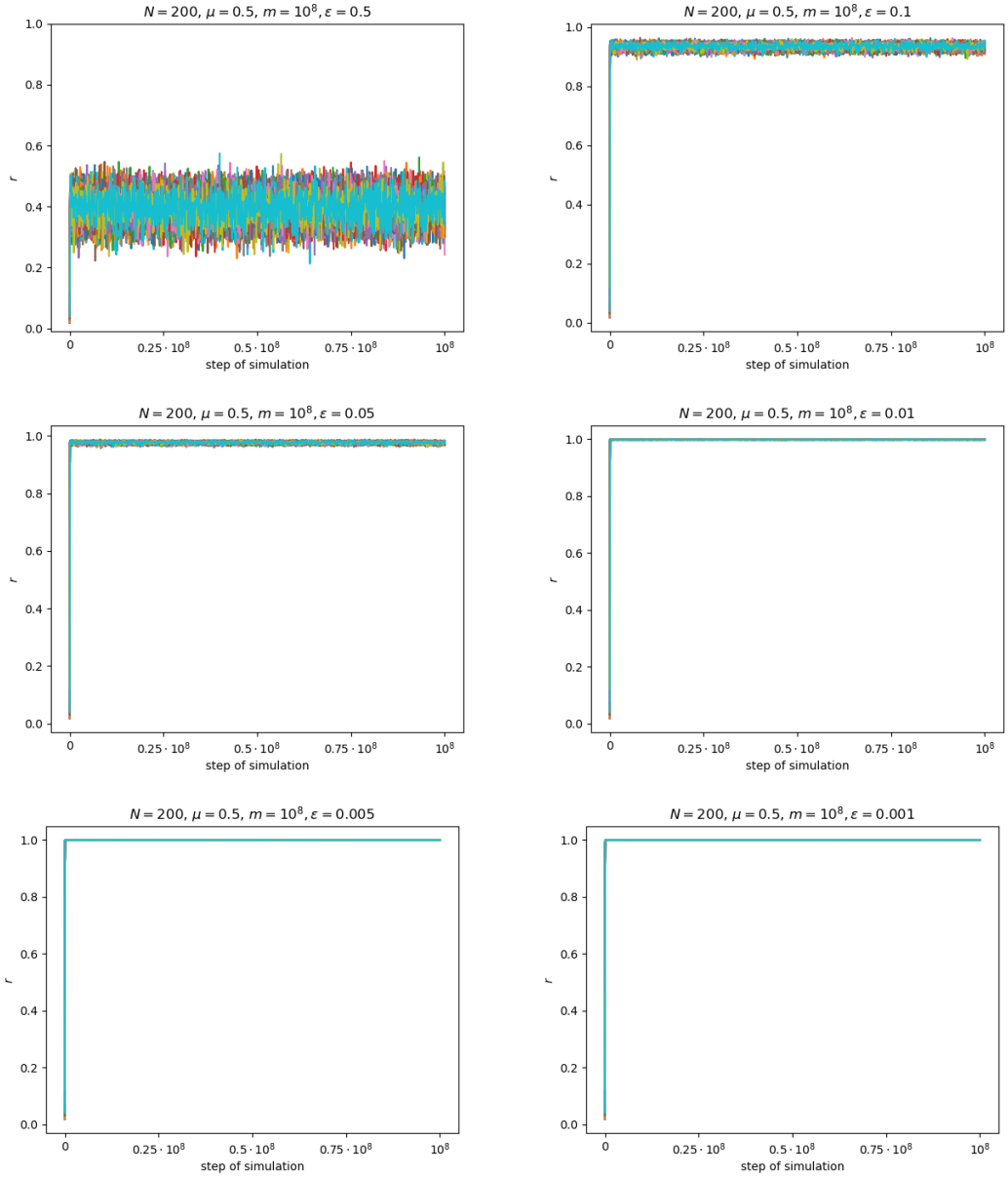

Figure 8.22.: These figures show results for the parameter r for the Uniform Deffuant model with bi-modal noise. The simulations all used the parameters indicated in the respective headers. All lines correspond to individual simulations.

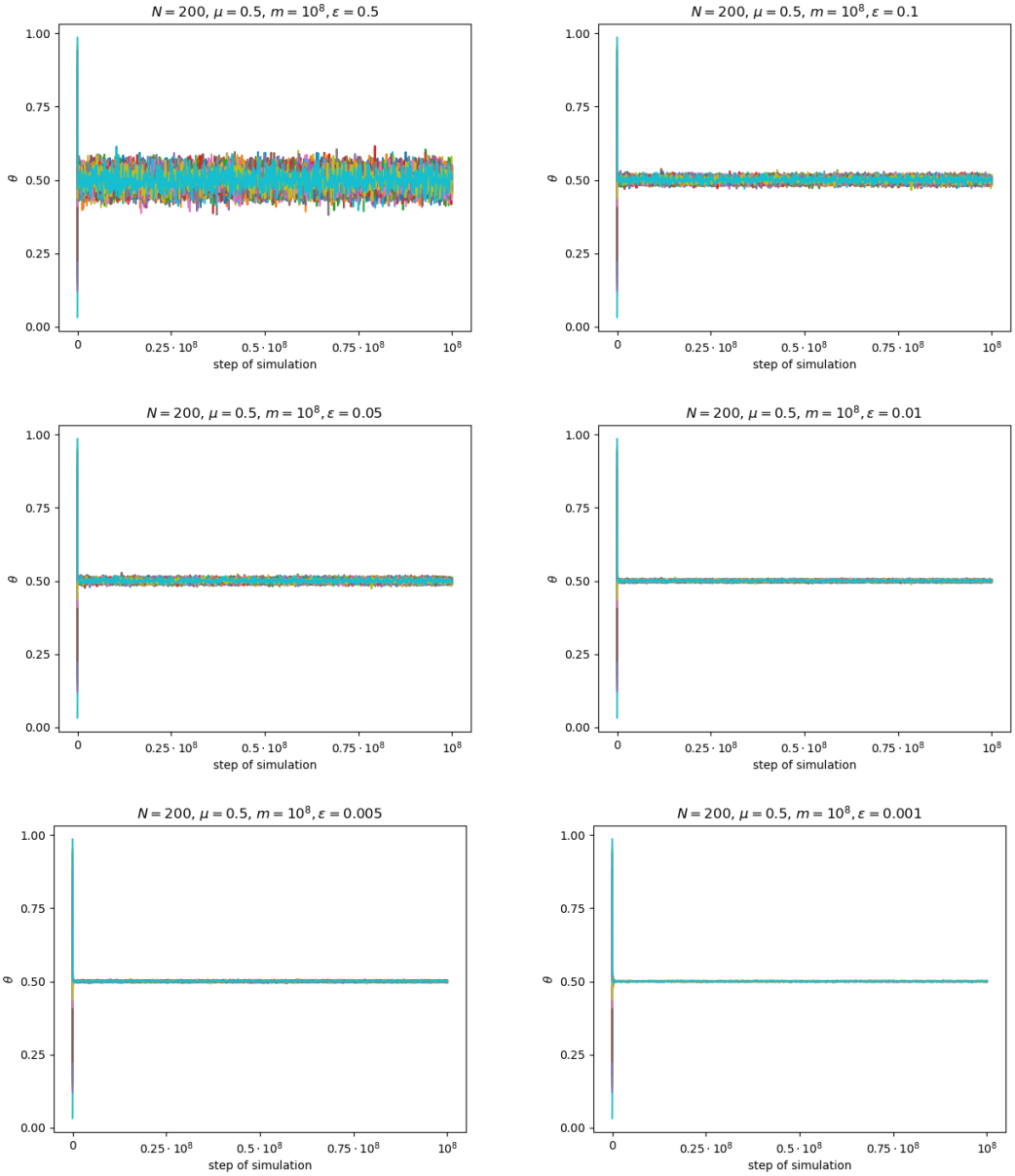

Figure 8.23.: These figures show results for the parameter θ for the Uniform Deffuant model with bi-modal noise. The simulations all used the parameters indicated in the respective headers. All lines correspond to individual simulations.

ϵ	mean of bandwidth of last configuration	standard deviation of bandwidth of last configuration
0.5	$8.80 \cdot 10^{-1}$	$3.63 \cdot 10^{-2}$
0.1	$3.26 \cdot 10^{-1}$	$3.42 \cdot 10^{-2}$
0.05	$1.92 \cdot 10^{-1}$	$2.08 \cdot 10^{-3}$
0.01	$5.22 \cdot 10^{-2}$	$7.45 \cdot 10^{-3}$
0.005	$2.96 \cdot 10^{-2}$	$4.56 \cdot 10^{-3}$
0.001	$7.67 \cdot 10^{-3}$	$1.48 \cdot 10^{-3}$

Table 8.4.: This table shows the mean and standard deviation of the bandwidth of the last configurations calculated in the simulations of the Uniform Deffuant Compass model with bi-modal noise for different values of ϵ.

In Figure 8.24, we see the configurations of the simulations after 10^8 steps. We can see that they always lie within an interval around 0.5. We see that the length of this interval decreases rapidly as ϵ decreases. For additional clarity, we calculated the lengths of these intervals, that is, the bandwidths, in Table 8.4. The results are as expected.

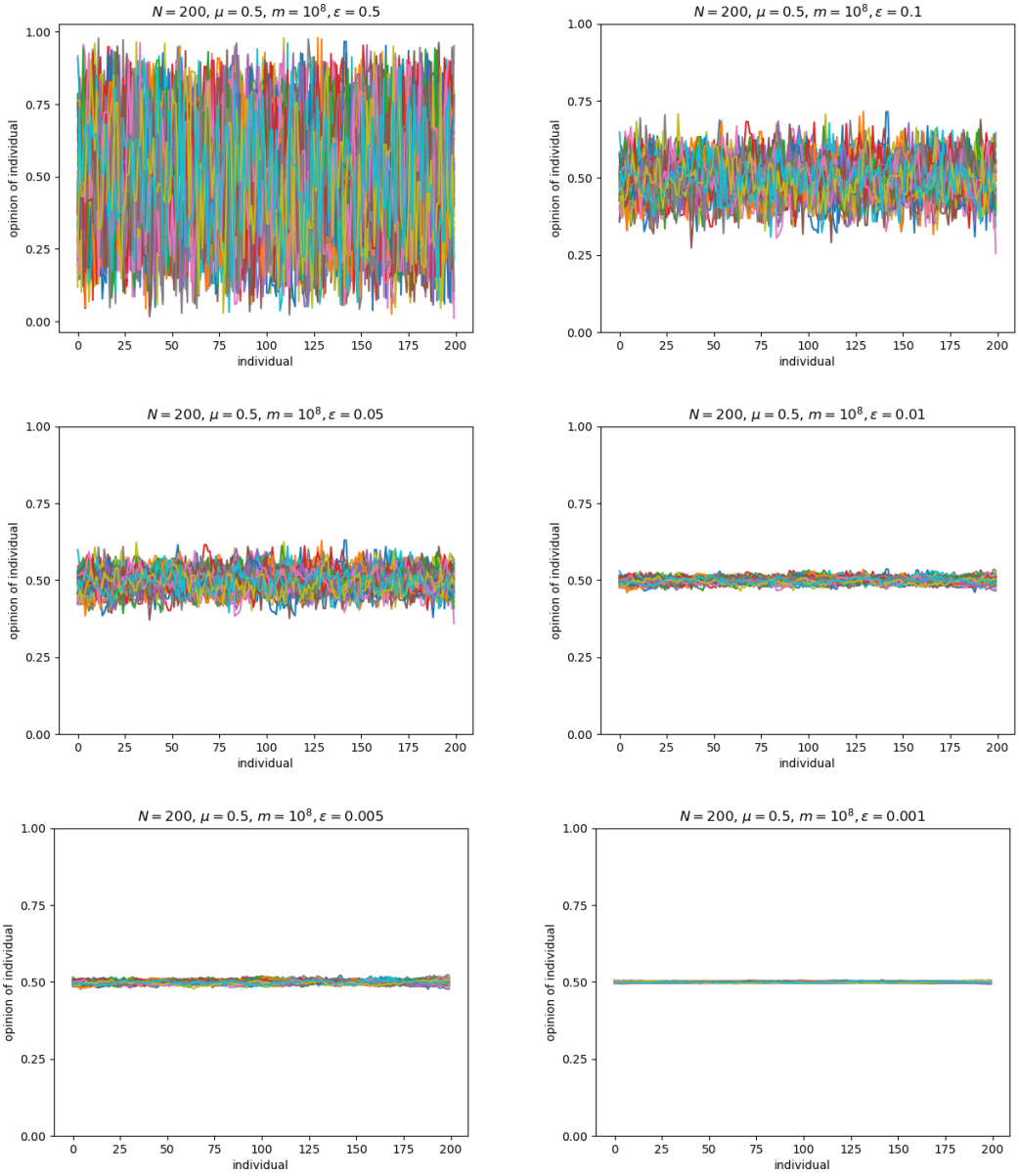

Figure 8.24.: The figures show the configurations after 10^8 steps for the Uniform Deffuant model with bi-modal noise for $N = 200$ and $\mu = 0.5$. Each coloured line corresponds to one simulation. The figures correspond to simulations for $\epsilon = 0.5, 0.1, 0.05, 0.01, 0.005$ and 0.001 as indicated in their headers.

8.4. Uniform Compass Model with Noise

After studying the behaviour of the Uniform Compass model in Section 8.2, we know go on to study the effect of both uniform and bi-modal noise. Our main question is to see whether the long term behaviour of the model changes under the influence of noise. In particular, we want to know whether this is dependent on how large the noise is. Given the length of the simulations, we restrict our attention to the parameter $\mu = 0.5$ in this section.

8.4.1. Uniform Noise

In order to study the effect of uniform noise, we simulated the Uniform Compass Model for $N = 200$, $m = 10^8$ and $\mu = 0.5$ with uniform noise as described in Algorithm 7. We simulated ϵ-values in $\{0.5, 0.1, 0.05, 0.01, 0.0001, 10^{-6}\}$. Because opinions are modelled as values on a circle of length 2, adding this type of noise with $\epsilon > 0.5$ does not allow a consensus formation. Each simulation consists of 50 interpretations with the same parameter selection.

The uniform noise is symmetric and centralised around 0. Thus, for small ϵ there are only small perturbations and on average equally often in both directions. We therefore expect the behaviour of the model to be very similar to the version without noise. The only difference that we expect is that the individuals only agree within a small interval of opinions. However, for large values of ϵ, we expect the noise to be so disruptive that no (approximate) consensus can be reached.

Figure 8.25 shows the result of the parameter c for these simulations. For every value of ϵ we can see that c falls quickly to a value significantly lower than its starting value and oscillates around it. However, the larger ϵ is, the larger this value gets and the stronger the visible oscillations.

Figure 8.26 visualises the order parameter r. For $\epsilon = 0.5$, there is absolutely no visible convergence of r to 1. For $\epsilon = 0.1$, r oscillates wildly between 0 and 1. For $\epsilon = 0.05$, r still oscillates quickly, but closer to 1. For smaller values of ϵ, r seems to converge to a value close to 1 and only oscillate slightly. This oscillation becomes less pronounced the smaller ϵ gets. As we have seen before, the closer r is to 1, the more synchronised the opinions are. Thus, we can assume that the opinions synchronise to within increasingly smaller bounds for smaller ϵ.

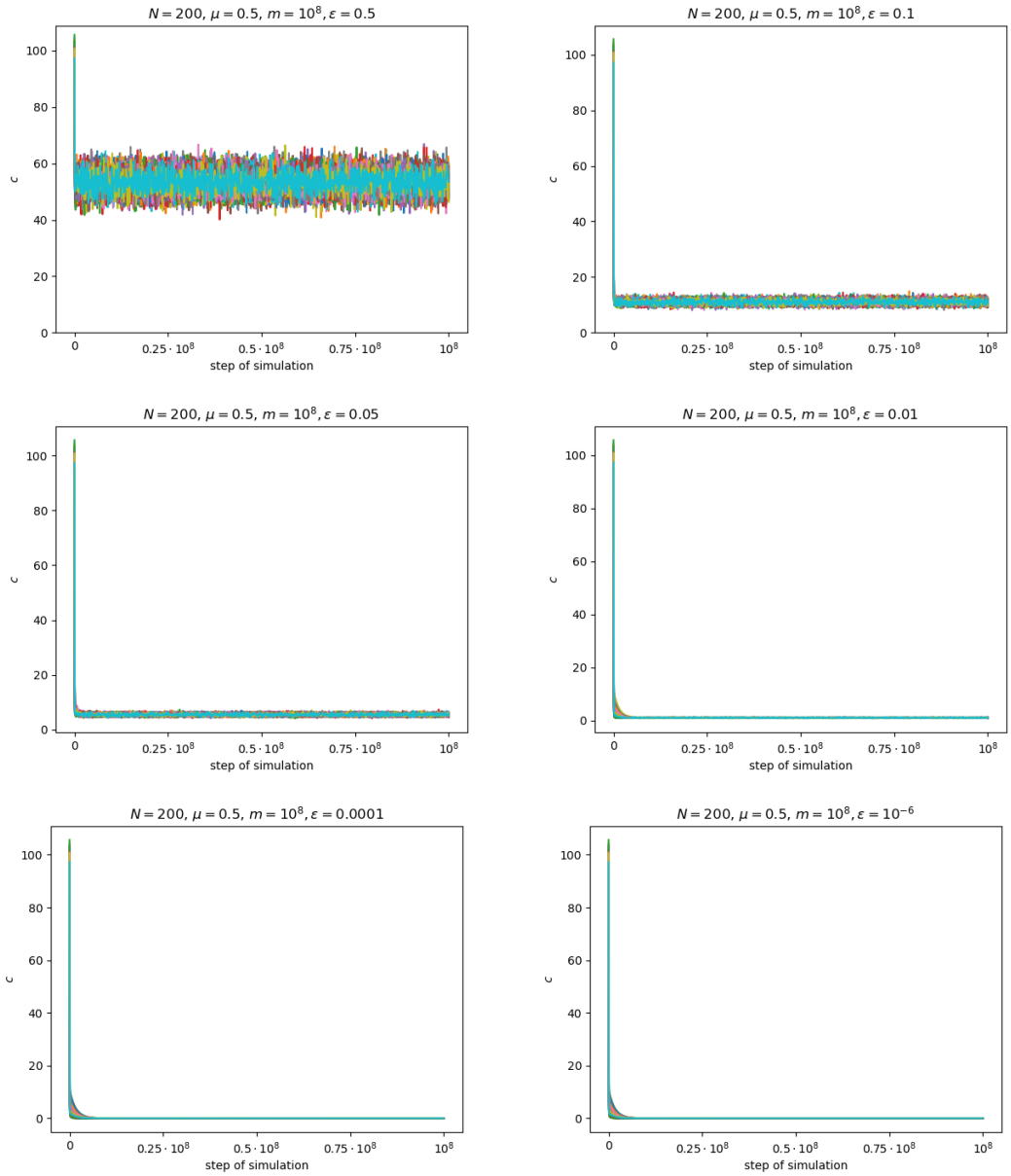

Figure 8.25.: These figures show the parameter c for the Uniform Compass Model with uniform noise for $N = 200$ and $\mu = 0.5$. Each coloured line represents one of 50 simulations with the same set of parameters. The figures on the left hand side are from top to bottom for $\epsilon = 0.5, 0.05, 0.0001$. The figures on the right hand side are from top to bottom for $\epsilon = 0.1, 0.01, 10^{-6}$.

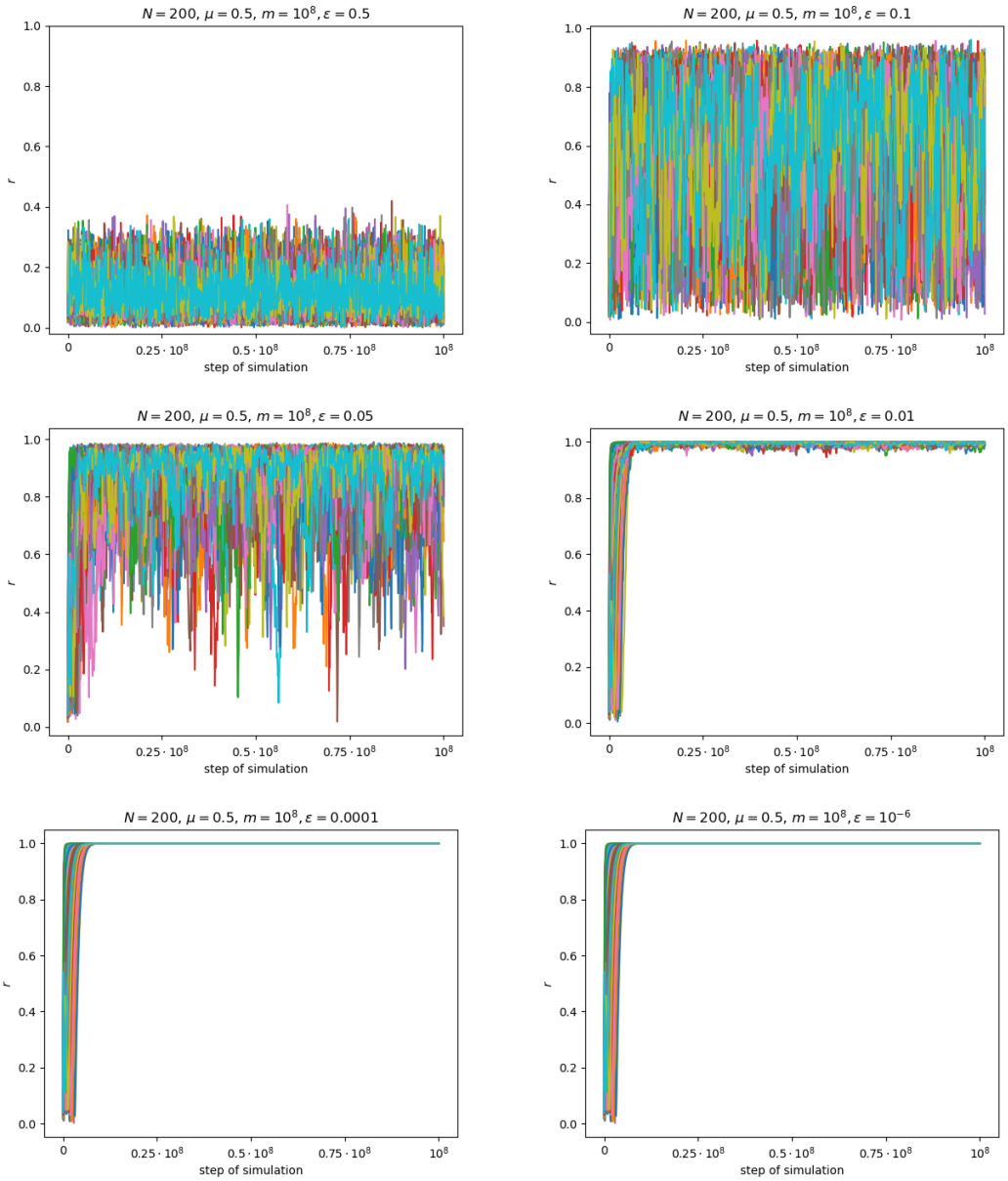

Figure 8.26.: These figures show the order parameter r for the Uniform Compass Model with uniform noise for $N = 200$ and $\mu = 0.5$. Each coloured line represents one of 50 simulations with the same set of parameters. The figures on the left hand side are from top to bottom for $\epsilon = 0.5, 0.05, 0.0001$. The figures on the right hand side are from top to bottom for $\epsilon = 0.1, 0.01, 10^{-6}$.

The results for θ are shown in Figure 8.27. For $\epsilon = 0.5, 0.1, 0.05$, the values are θ are oscillating within their full range. For lower values of ϵ, they are oscillating withing small intervals after approximately 10^7 steps. These intervals are smaller for smaller ϵ.

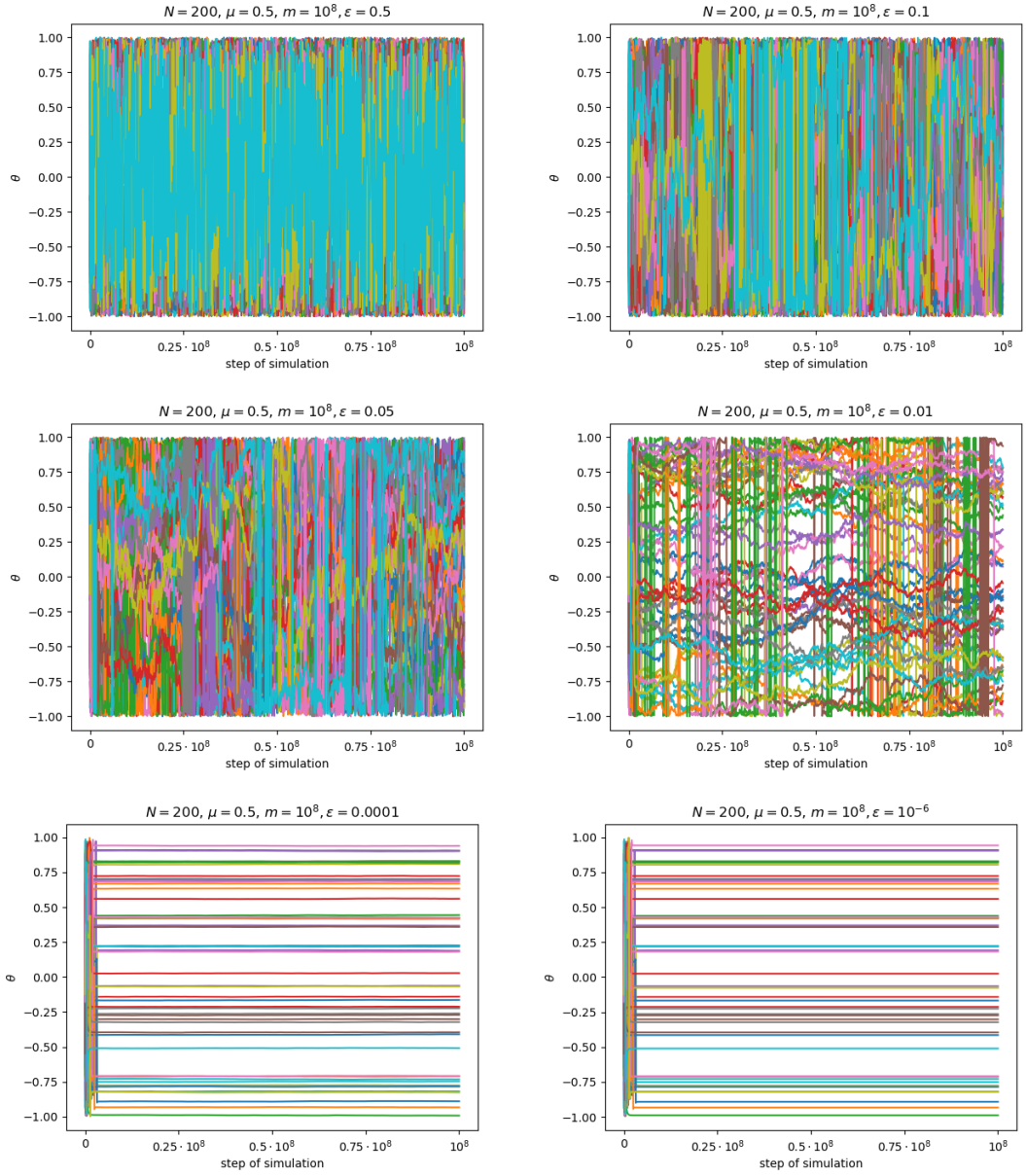

Figure 8.27.: These figures show the order parameter θ for the Uniform Compass Model with uniform noise for $N = 200$ and $\mu = 0.5$. Each coloured line represents one of 50 simulations with the same set of parameters. The figures on the left hand side are from top to bottom for $\epsilon = 0.5, 0.05, 0.0001$. The figures on the right hand side are from top to bottom for $\epsilon = 0.1, 0.01, 10^{-6}$.

ϵ	mean of bandwidth of last configuration	standard deviation of bandwidth of last configuration
0.5	1.93	$1.90 \cdot 10^{-2}$
0.1	1.26	$2.95 \cdot 10^{-1}$
0.05	$6.40 \cdot 10^{-1}$	$2.28 \cdot 10^{-1}$
0.01	$1.20 \cdot 10^{-1}$	$3.21 \cdot 10^{-2}$
0.0001	$1.18 \cdot 10^{-3}$	$3.49 \cdot 10^{-4}$
10^{-6}	$1.32 \cdot 10^{-5}$	$4.24 \cdot 10^{-6}$

Table 8.5.: This table shows the mean and standard deviation of the bandwidth of the last configurations calculated in the simulations of the Uniform Compass model with uniform noise for different values of ϵ.

The configuration that the model calculates after 10^8 steps are shown in 8.28. As expected from the results for the order parameters, there is no consensus for $\epsilon \in \{0.5, 0.1, 0.05\}$. For smaller ϵ there is agreement onto opinions in an interval that is smaller for smaller ϵ. For the smallest ϵ values there is little visible disagreement between neighbours. We also see that for every individual i, the final opinions of that individual over the 50 interpretations of the simulation for one value of ϵ appear approximately uniformly distributed over the interval.

By calculating the bandwidth of opinions of the last configurations simulated for different values of ϵ and calculating their mean, we can see that the individuals indeed get closer and closer to reaching consensus when ϵ gets close to 0. The mean and standard deviation of the bandwidth for different values of ϵ are shown in Table 8.5.

In Figure 8.29, we can see that given identical initial condition, interaction times and interaction places, uniform noise with $\epsilon = 10^{-6}$ does not change the averages of the opinions that the process converges to. This means that for sufficiently small ϵ values, we do not expect uniform noise to have a significant impact on the dynamics of the model.

To summarise, we have shown that for sufficiently small values of ϵ, the qualitative behaviour of the model is approximately as it is without noise. This is what we have suspected.

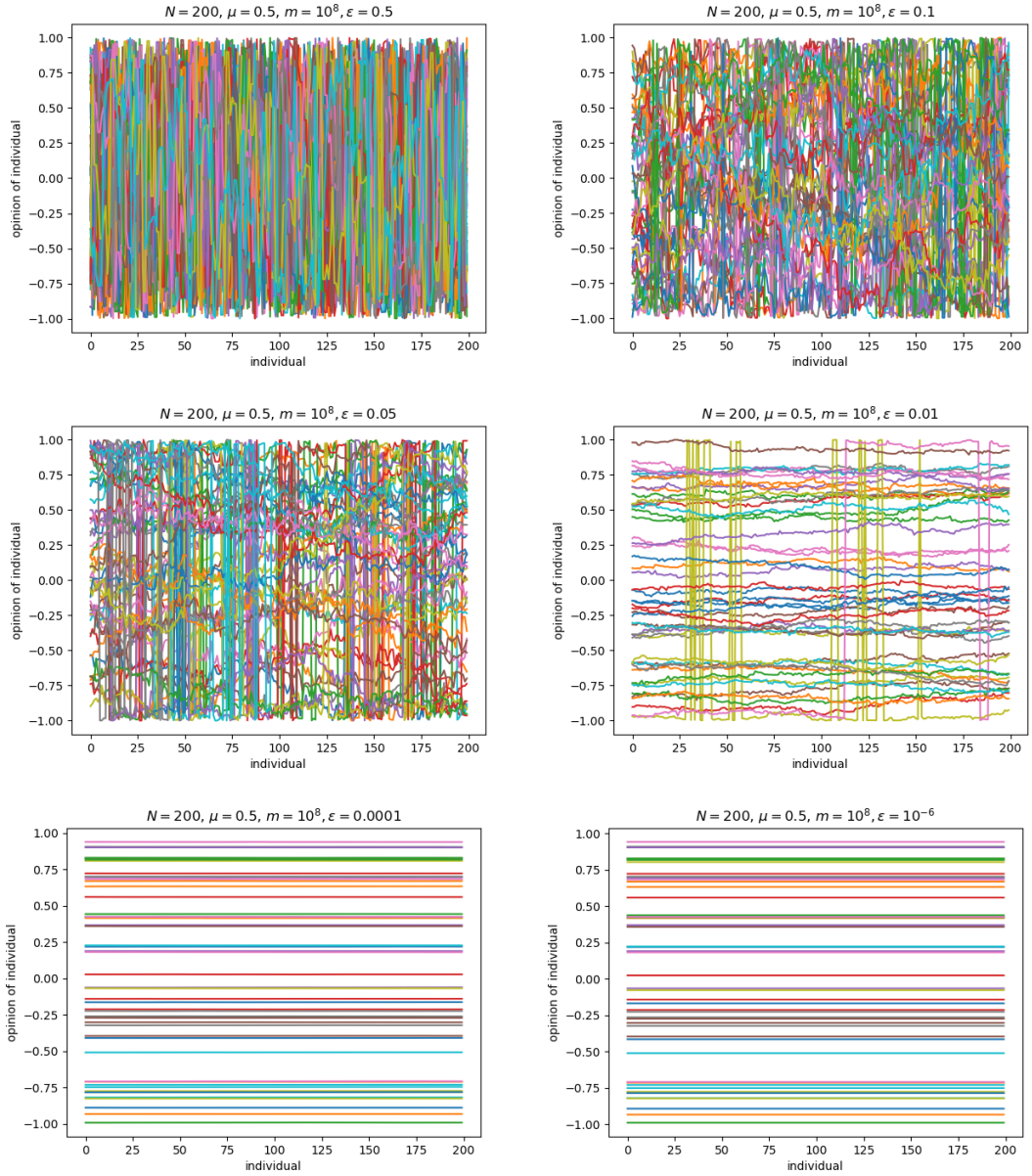

Figure 8.28.: These figures show the configurations of the Uniform Compass Model with uniform noise for $N = 200$ and $\mu = 0.5$, after 10^8 steps. Each coloured line represents one of 50 simulations with the same set of parameters. The figures on the left hand side are from top to bottom for $\epsilon = 0.5, 0.05, 0.0001$. The figures on the right hand side are from top to bottom for $\epsilon = 0.1, 0.01, 10^{-6}$.

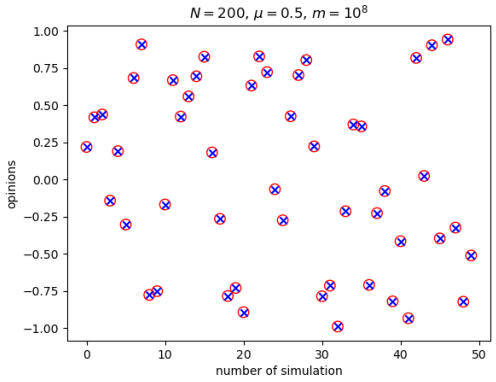

Figure 8.29.: This figure shows the averages of the last calculated configuration for simulations of the Uniform Compass model with and without uniform noise. In the simulation with noise, $\epsilon = 10^{-6}$ was used. Both simulations used identical starting conditions, interaction times, and interaction places. The x-axis of the figure shows the number of the interpretation of the simulation and the y-axis gives opinions. The blue crosses give the average of the opinions in the last simulated configuration in the case with noise. The red circles give the opinion that the individuals would agree upon in the case without noise.

8.4.2. Bi-Modal Noise

We simulated the Uniform Compass Model with bi-modal noise. In every step one opinion is perturbed towards one of the values in $\{0, 1\}$ with equal probability. How much a value gets perturbed by this noise in each step is proportional to its distance from the value it gets disturbed towards. Before each simulation we chose a value ϵ. This value quantifies the noise in the following way. In one step, let $x \in (-1, 1]$ be the opinion of the individual which we perturb by noise. Let $a \in \{0, 1\}$ be the value towards which x is pushed. Then the updated opinion is given by

$$\begin{cases} (1-\epsilon)x, & \text{if } a = 0, x \neq 0, \\ (-1)^y(1-\epsilon)x, & \text{if } a = 0, x = 0, \\ (1-\epsilon)x + \epsilon, & \text{if } a = 1, x > 0, \\ (1-\epsilon)x - \epsilon, & \text{if } a = 1, x < 0, \\ (-1)^y\epsilon, & \text{if } a = 1, x = 0, \end{cases}$$

where y is a realisation of $Y \sim \text{Uniform}(\{0, 1\})$. What results behaviour would we expect in such a simulation?

Assume a simulation where all individuals have the same opinion x at the beginning of the simulation. If $x \in \{-0.5, 0.5\}$, the noise disturbs the individual opinions equally into direction 0 and direction 1. Once disturbed, the dynamics of the compass model pull all opinions towards their average again. This means that we would expect that each opinions approximately stays at 0.5 or -0.5, respectively. Therefore, we expect $\frac{1}{2}(\delta_{0.5} + \delta_{-0.5})$ to be the invariant measure for this model.

If ϵ is small and opinions are not yet converged, we expect the dynamics of the compass model to be more relevant than the noise. This leads to individuals nearly agreeing to some value in $(-1, 1]$. This means that the dynamics of the noise start to become the most relevant factor as described in the previous chapter. If the value that the individuals agreed upon lies in $(-1, 0)$, the model will converge to -0.5. If the value lies in $(0, 1)$, it will converge to 0.5.

When ϵ is large we expect the noise to be a lot more dominant than the dynamics of the model. This means that the opinions quickly cluster around either -0.5 or 0.5. However, because the noise is large, the configurations change significantly in every step and also keep changing. Thus, making consensus formation increasingly difficult for increasing ϵ-values.

To test this intuition, we simulated the Uniform Compass Model for $N = 200$, $m = 2{\cdot}10^8$ and $\mu = 0.5$ with bi-modal noise. We simulated ϵ-values in $\{0.5, 0.1, 0.01, \ldots, 10^{-14}\}$. Larger values of ϵ are by design not plausible. For each set of parameters we ran 50 interpretations of the simulation.

As before for the Compass model with uniform noise, we begin by looking at the parameter c. It is visualised in Figures 8.30, 8.31 and 8.32. In these figures we can see that the smaller ϵ is the faster c approximates 0 and the less oscillations there are.

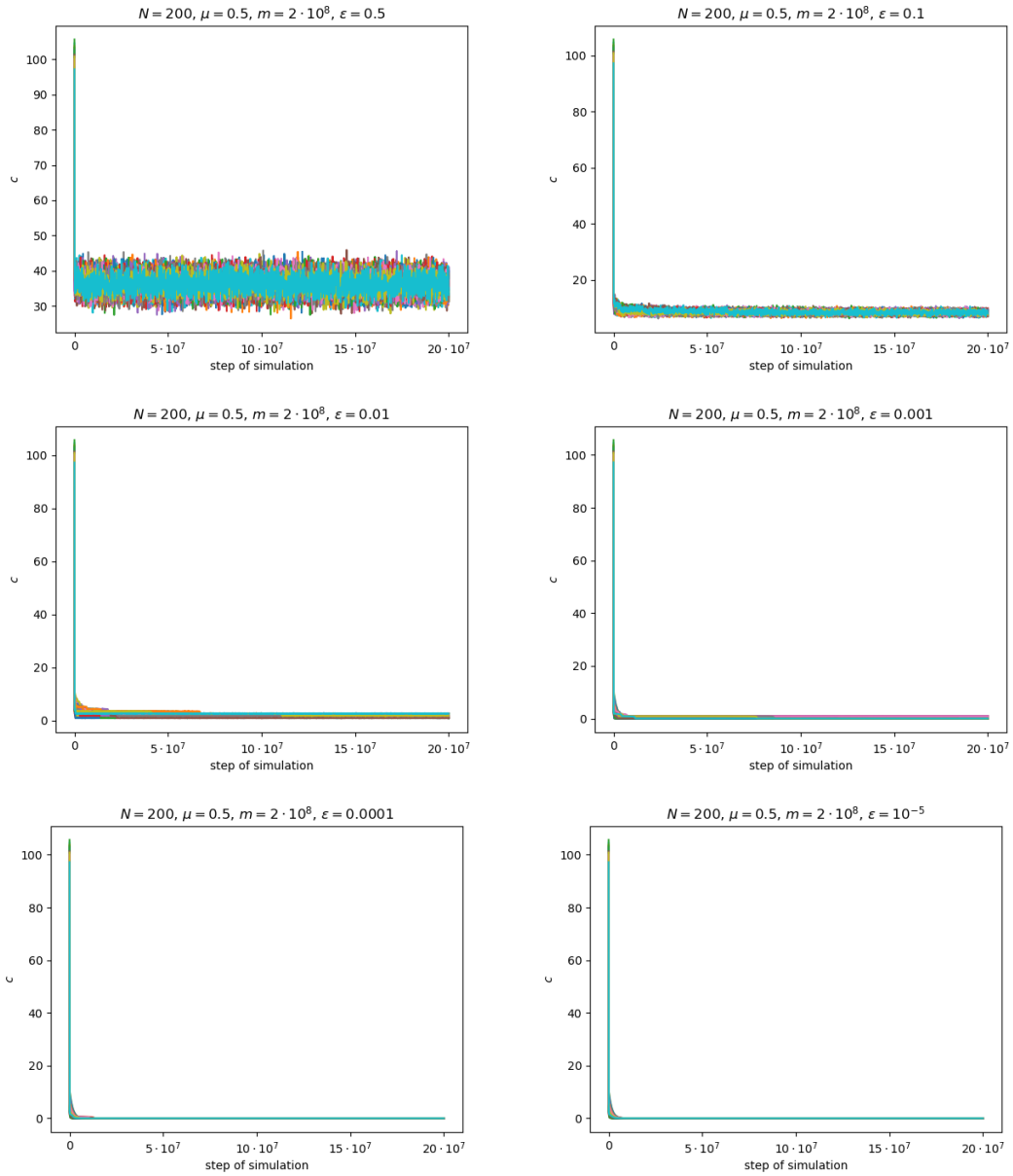

Figure 8.30.: Each figure shows the results of c for the Uniform Compass Model with bi-modal noise for different values of ϵ. Here ϵ takes the values $0.5, 0.1, 0.01, 0.001, 0.0001$ and 10^{-5}. Each coloured line represents one simulation.

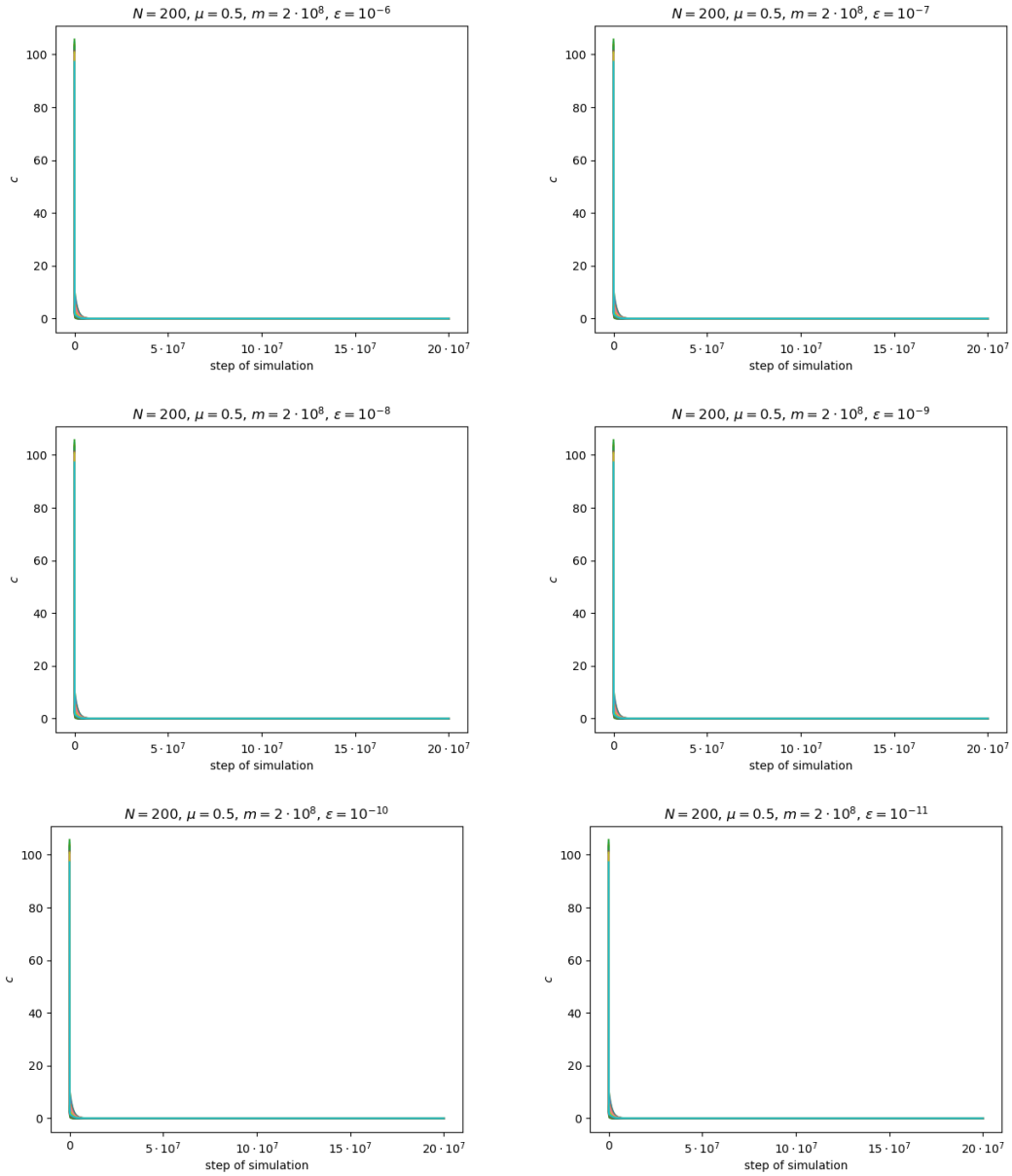

Figure 8.31.: Each figure shows the results of c for the Uniform Compass Model with bi-modal noise for different values of ϵ. Here ϵ takes the values $10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$ and 10^{-11}. Each coloured line represents one simulation.

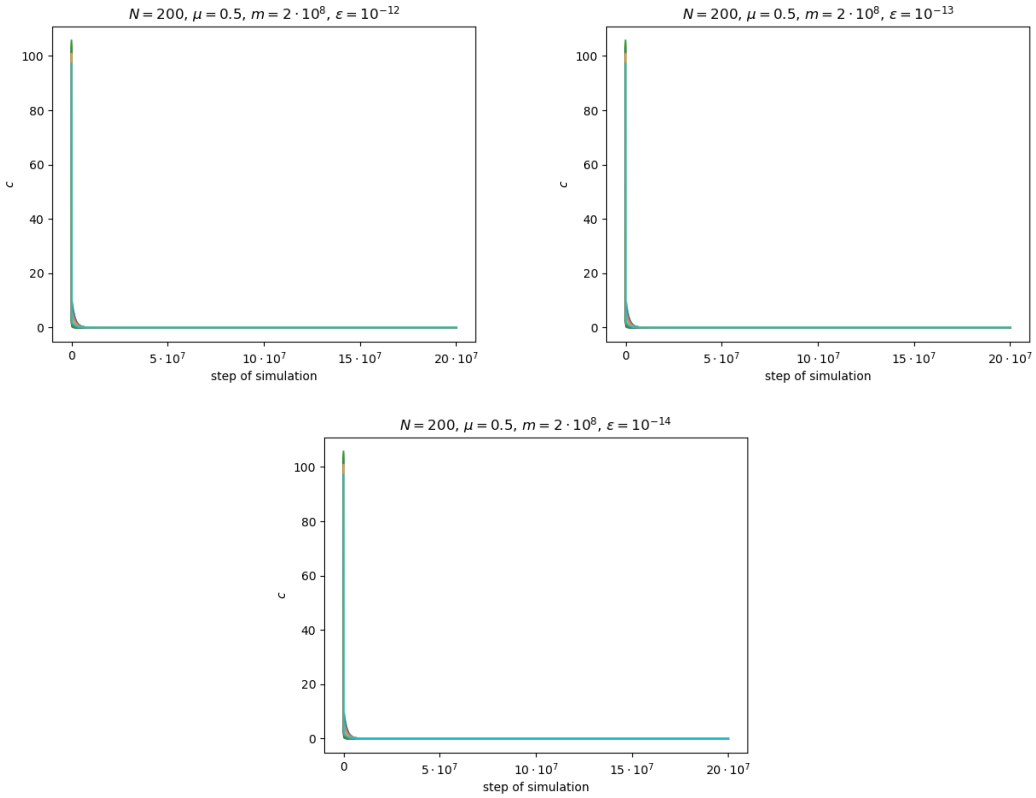

Figure 8.32.: Each figure shows the results of c for the Uniform Compass Model with bi-modal noise for different values of ϵ. Each coloured line represents one simulation. Here ϵ takes the values $10^{-12}, 10^{-13}$ and 10^{-14}.

Figures 8.33, 8.34 and 8.35 show the values of r for these simulations. Figures 8.36, 8.37 and 8.38 show the values of θ for these simulations. For large ϵ, that is $\epsilon \in (0.01, 0.5]$, the values of r seem to converge to 1 and the values of θ to -0.5 or 0.5. However, they get increasingly noisy as the value of ϵ gets closer to its maximum 0.5. For $\epsilon \in (10^{-4}, 0.1)$, r and θ only seem to converge for some simulations and not all. For small ϵ-values, that is $\epsilon \leq 10^{-4}$, r seems to go towards 1. Thus, there is approximate consensus. The values of θ converge to -0.5 and 0.5 for $\epsilon \in [10^{-3}, 10^{-5}]$. For smaller values of ϵ, convergence of θ is not achieved within the length of the simulation. However, we observe, that agreement seems to slow down for smaller ϵ. For $\theta = 10^{-5}$, the values of θ do not get close to -0.5 or 0.5, but do seem to converge towards these values.

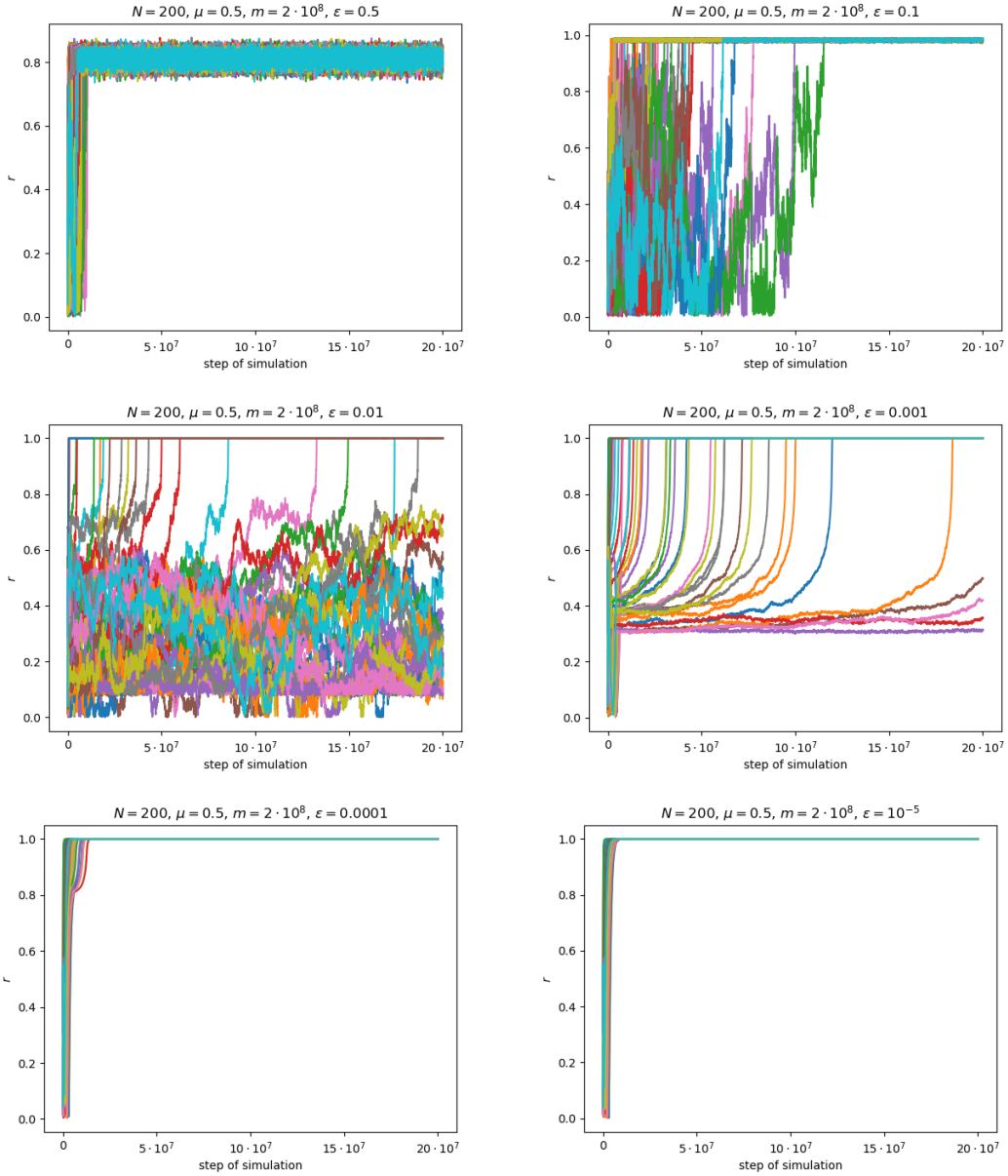

Figure 8.33.: Each figure shows the results of r for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $0.5, 0.1, 0.01, 0.001, 0.0001$ and 10^{-5}.

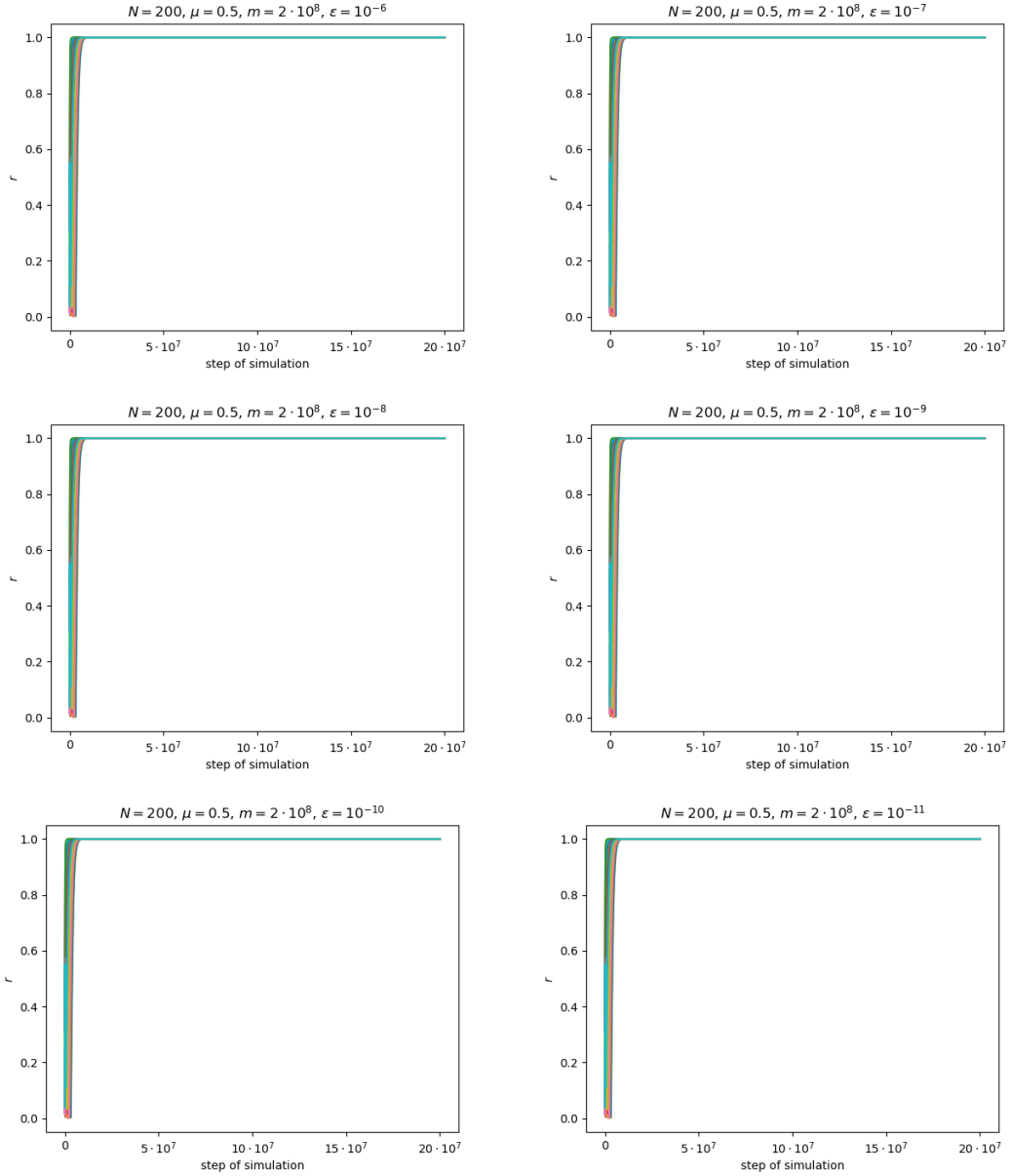

Figure 8.34.: Each figure shows the results of r for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$ and 10^{-11}.

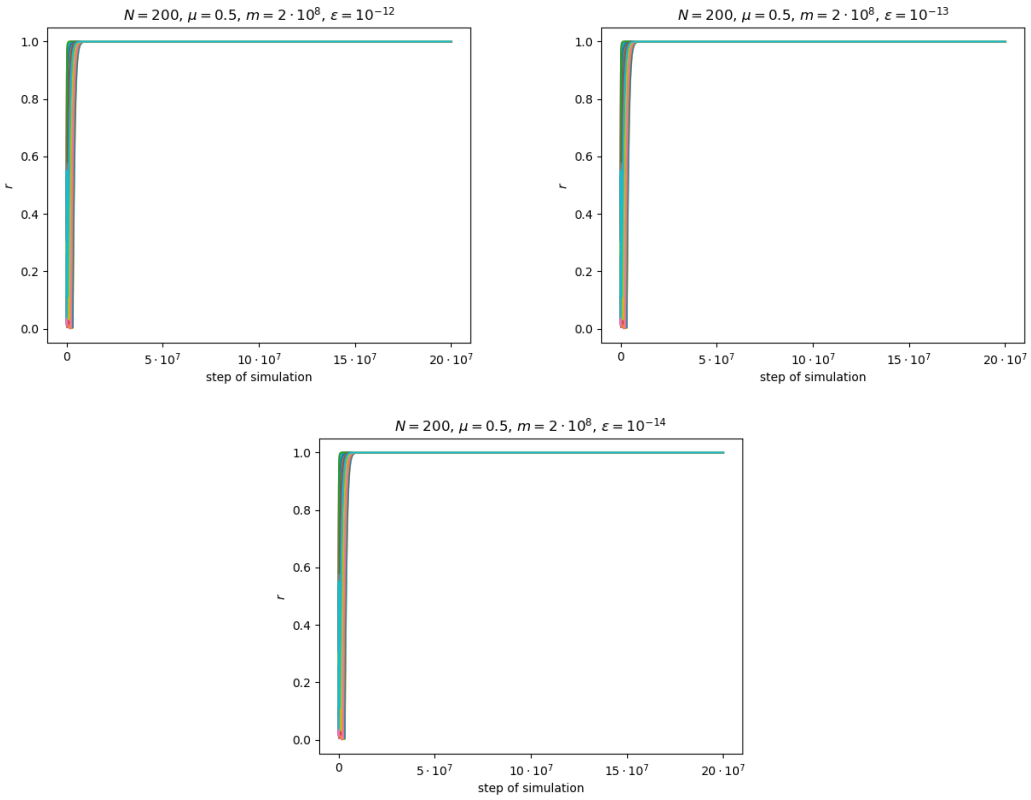

Figure 8.35.: Each figure shows the results of r for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $10^{-12}, 10^{-13}$ and 10^{-14}.

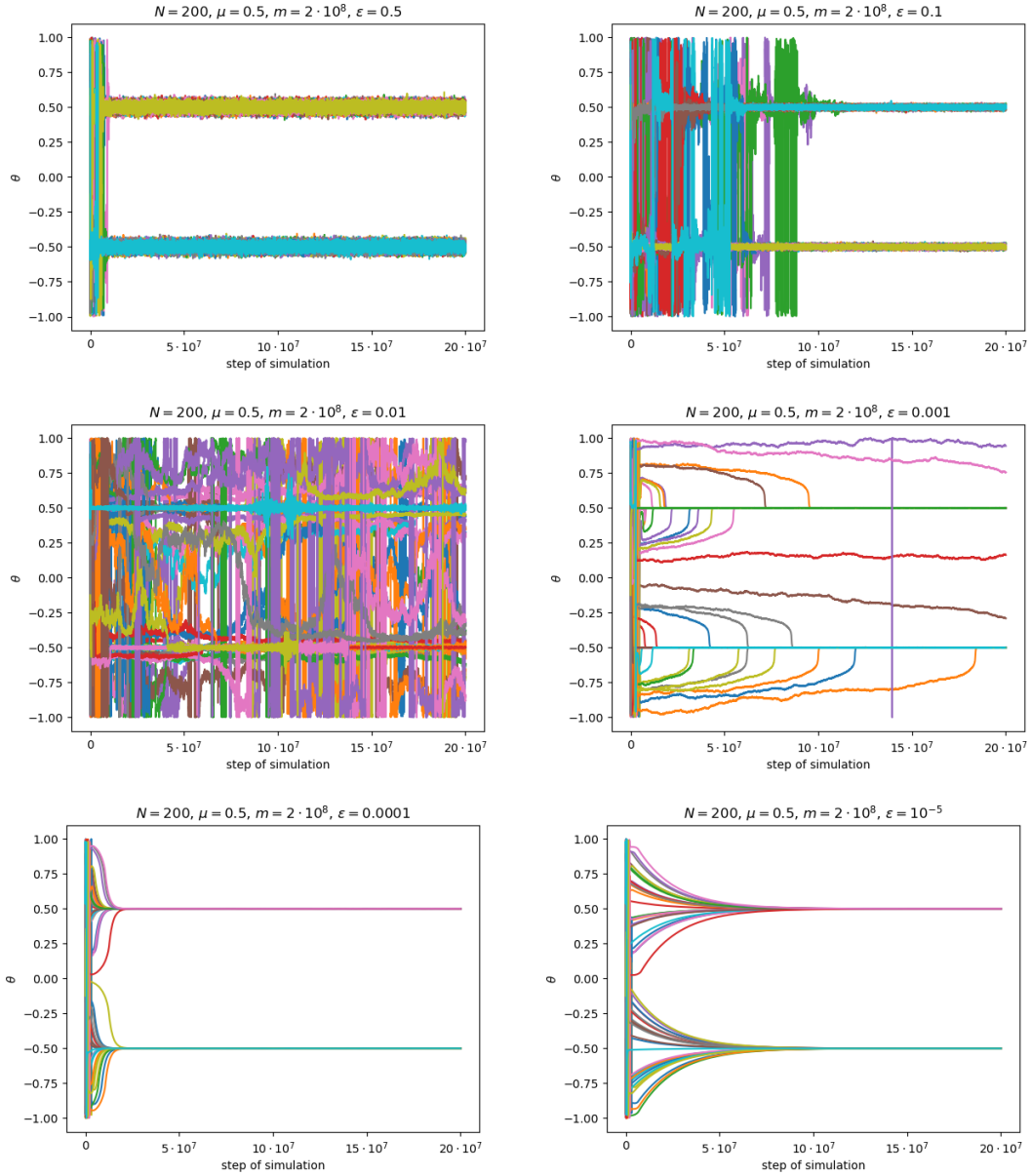

Figure 8.36.: Each figure shows the results of θ for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $0.5, 0.1, 0.01, 0.001, 0.0001$ and 10^{-5}.

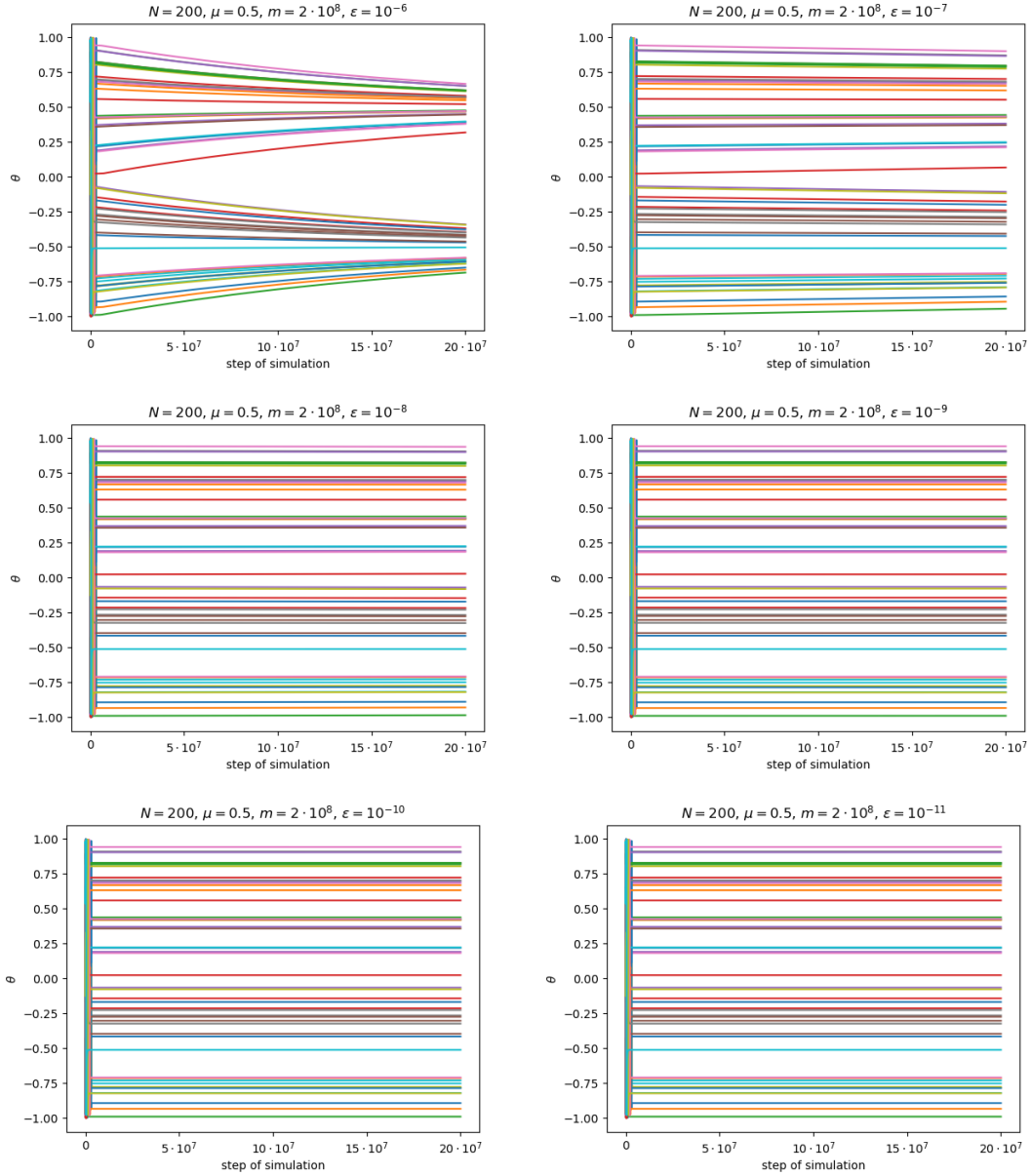

Figure 8.37.: Each figure shows the results of θ for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$ and 10^{-11}.

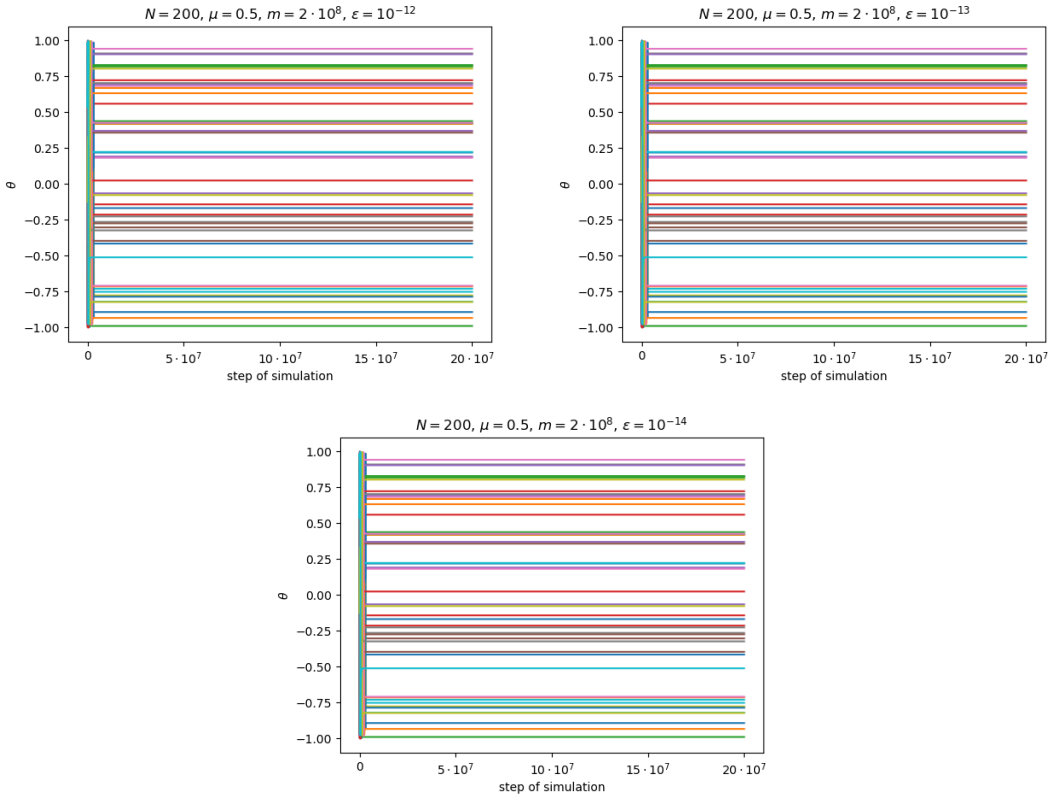

Figure 8.38.: Each figure shows the results of θ for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $10^{-12}, 10^{-13}$ and 10^{-14}.

In order to analyse the previous observation further, we consider the slope of the values of θ for very small ϵ in order to see whether there is still convergence towards -0.5 and 0.5.

Figures 8.39 and 8.40 show the difference in the value of θ between steps 10^8 and step $2 \cdot 10^8$ for $\epsilon \in \{10^{-08}, \ldots, 10^{-14}\}$. The slope is positive if the opinion is in $(-1, -0.5)$ and $(0, 0.5)$. Elsewhere the values are negative. This lets us suspect that the opinions are still converging to -0.5 and 0.5 as seen for larger ϵ, just at a much slower pace.

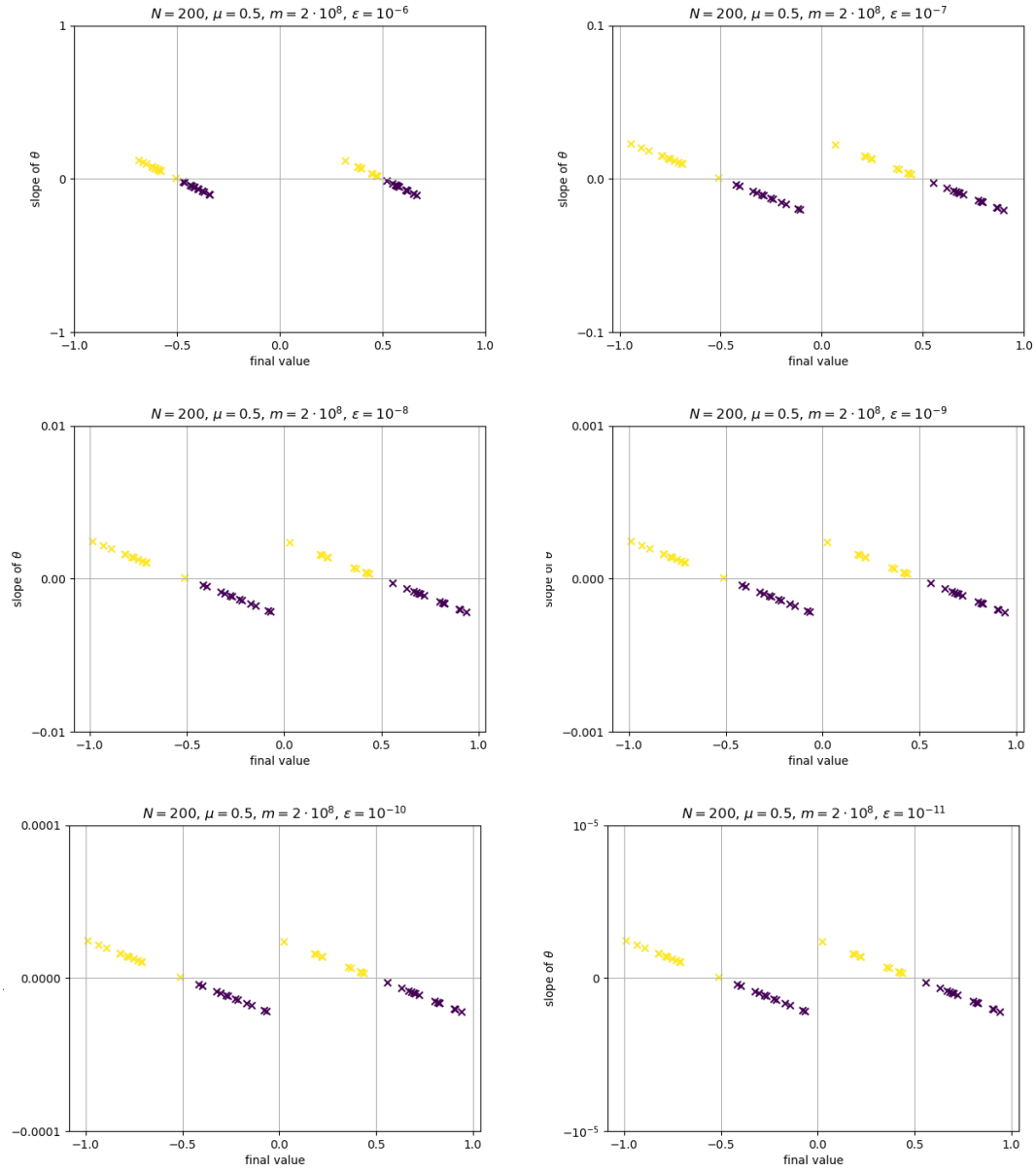

Figure 8.39.: Each figure shows the results of the difference in the value of θ between steps 10^8 and step $2 \cdot 10^8$ for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$ and 10^{-11}.

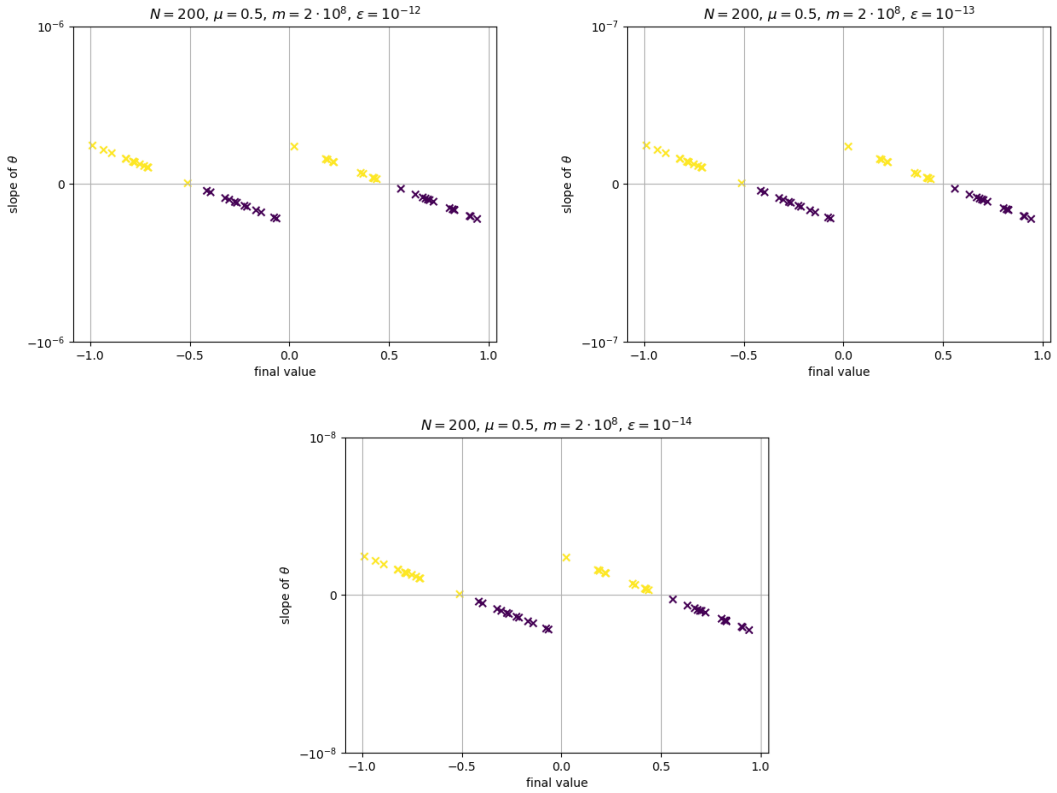

Figure 8.40.: Each figure shows the results of the difference in the value of θ between steps 10^8 and step $2 \cdot 10^8$ for 50 simulations of the Uniform Compass Model with bi-modal noise and the same ϵ. Here ϵ takes the values $10^{-12}, 10^{-13}$ and 10^{-14}.

In Figures 8.41, 8.42 and 8.43, we can see the last configurations that were simulated for the Uniform Compass model with bi-modal noise. For $\epsilon = 0.5$ and $\epsilon = 0.1$, we can see that the configurations are clustered around -0.5 and 0.5 but all still are very uneven. For $\epsilon = 0.1$ more so than for $\epsilon = 0.5$. For $\epsilon = 0.0001$ and $\epsilon = 10^{-5}$ we also get line clustered around -0.5 and 0.5, but with a lot less oscillations. For larger ϵ-values the configurations get even flatter, but spread out more and more. This is what was to be expected from the results for r, θ and c shown before. For $\epsilon = 0.01$ and $\epsilon = 0.001$, some configurations are slightly uneven lines around -0.5 and 0.5. This was also expected. However, for the other configurations, we did not get a clear picture before. It turns out that these are in parts slightly uneven lines on -0.5 and 0.5 that are connected by a rather steep line. This lets us suspect that they might still converge to values close to -0.5 and 0.5 given more simulation time.

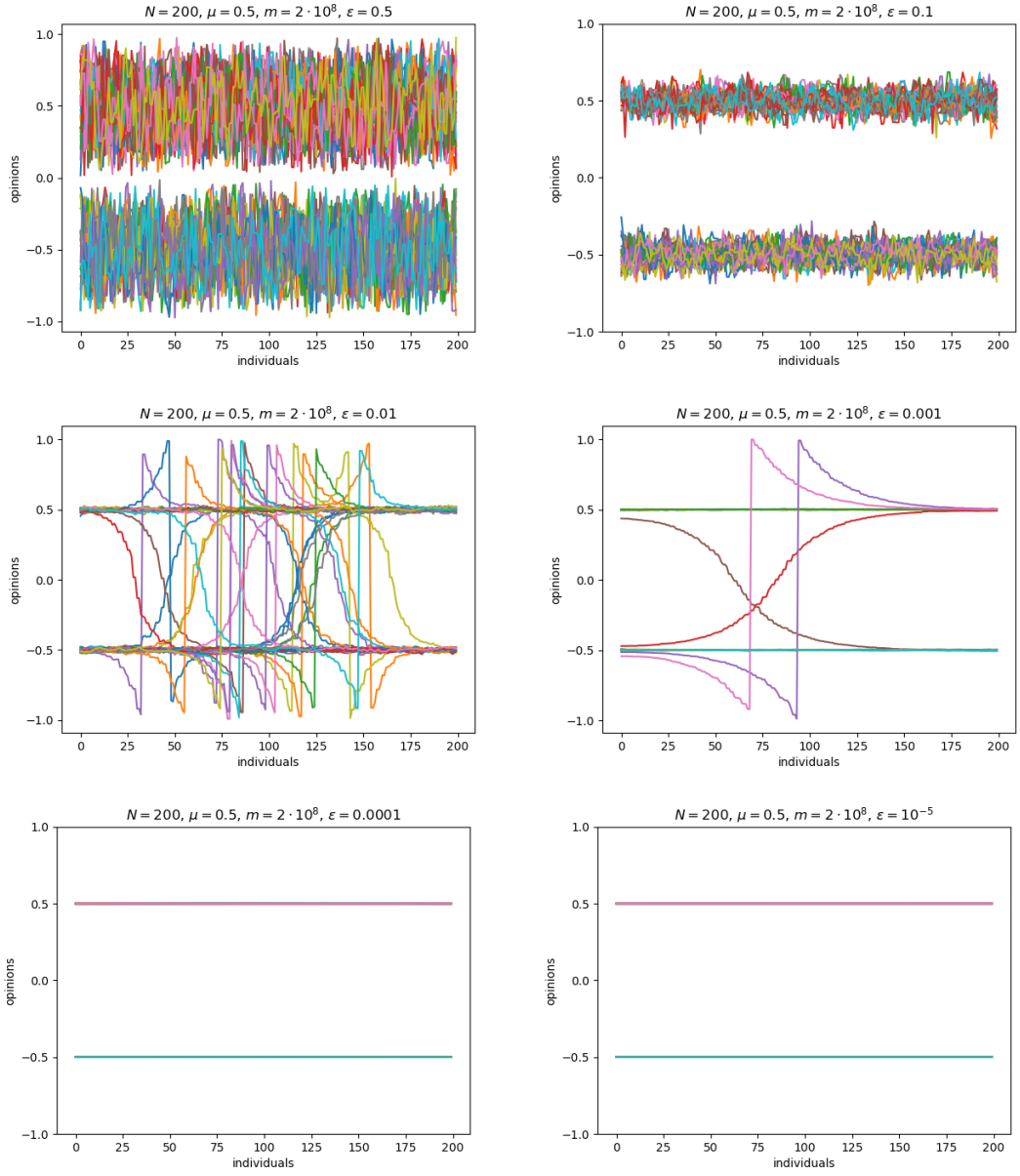

Figure 8.41.: Each figure shows the last simulated configurations for the Uniform Compass Model with bi-modal noise for different values of ϵ. Here ϵ takes the values $0.5, 0.1, 0.01, 0.001, 0.0001$ and 10^{-5}. Each coloured line represents one simulation.

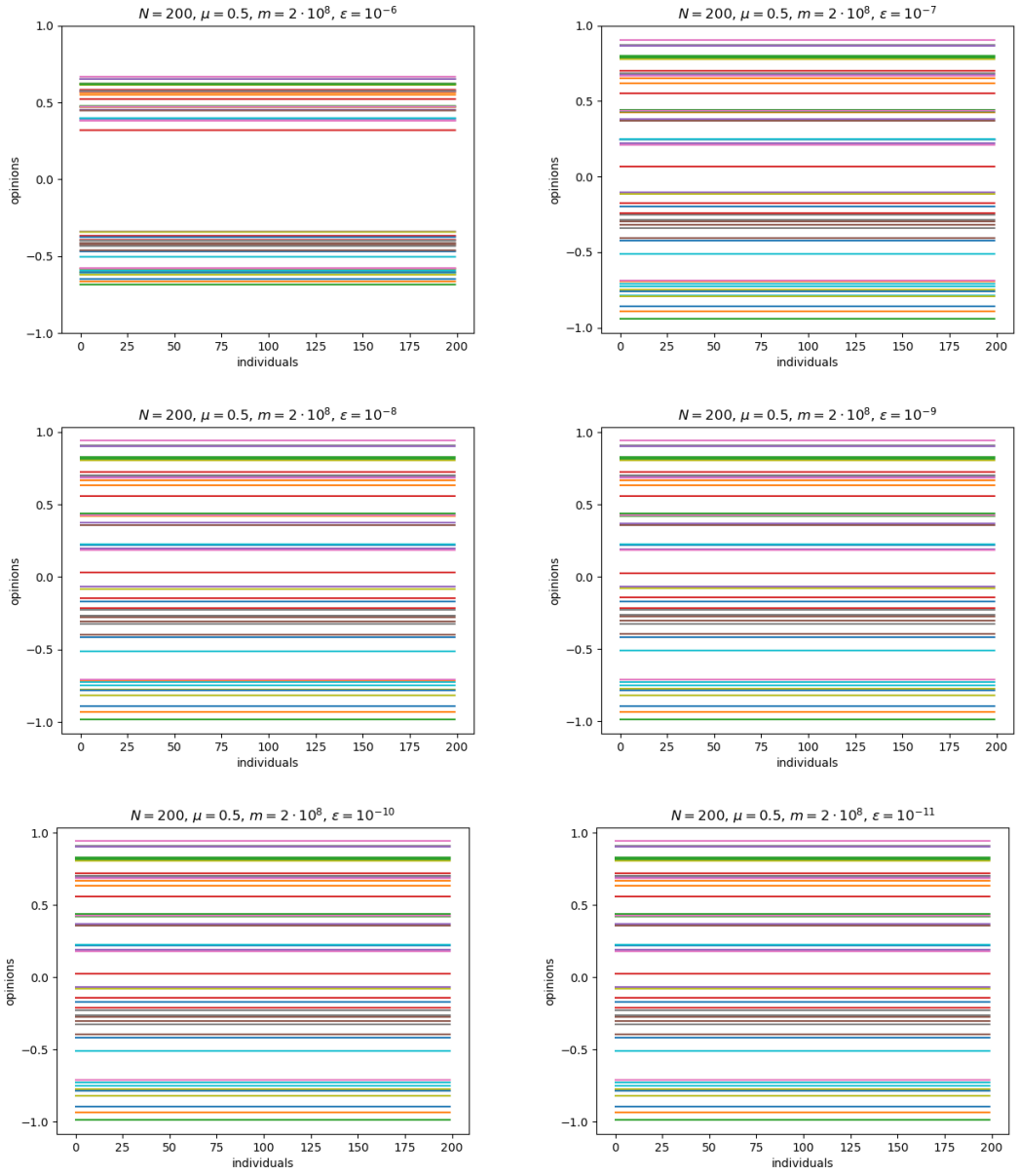

Figure 8.42.: Each figure shows the last simulated configurations for the Uniform Compass Model with bi-modal noise for different values of ϵ. Here ϵ takes the values $10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$ and 10^{-11}. Each coloured line represents one simulation.

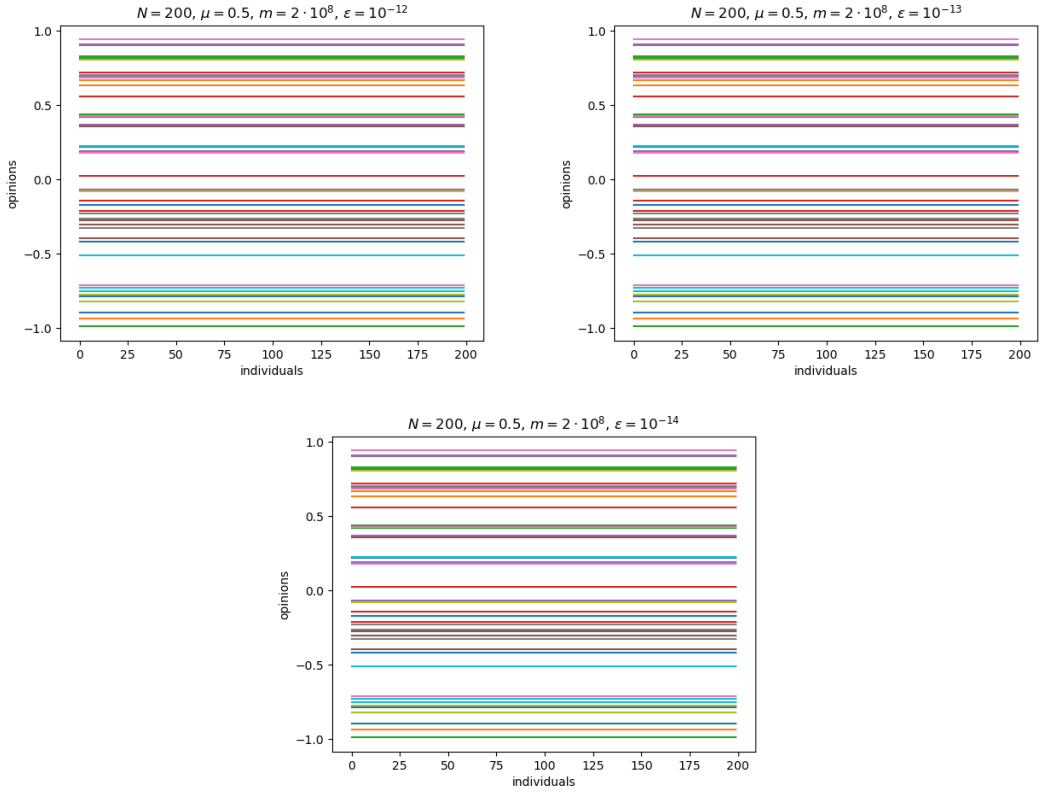

Figure 8.43.: Each figure shows last simulated configurations for the Uniform Compass Model with bi-modal noise for different values of ϵ. Each coloured line represents one simulation. Here ϵ takes the values $10^{-12}, 10^{-13}$ and 10^{-14}.

8.5. Independence of Noise

In Chapters 3 and 4 we defined two different versions of noise, uniform noise and bi-modal noise, as Markov processes. Note that the way we defined the Markov process, the noise is dependent on the process of the interactions of neighbours. In particular, noise is added at the same times as the interactions of neighbours occur. The numerical implementation of these kinds of noise was described in Chapter 7. In Chapter 8 the results of the simulations are presented.

Adding noise only when interactions between neighbours occur is less complex to implement and produces faster simulations that need less memory that using independent noise. Thus, this is a practical way to implement noise for the sake of this project. However, it is often easier to proof theoretical results for independent noise. Thus, we would like to know whether it makes a significant difference to consider independent noise in simulations. This section will investigate this question for the case of the UCM with bi-modal noise.

First let us define what we mean by independent noise. Consider the UCM. On all edges we imagine independent Poisson clocks. Whenever a Poisson clock on an edge between two individuals rings, these people interact. We now assume that we also have independent Poisson clocks on the vertices. Whenever one of those Poisson clocks rings, we add noise to the opinion of the person corresponding to the vertex it is on. As in the case without noise, with probability one no two Poisson clocks ring at the same time. This means the ringing times of the Poisson clocks gives us a well-defined order in which interaction take place and noise is added. The pseudo-code for this process is given in Algorithm 13.

We simulated the independent noise described Algorithm 13 for ϵ-values in 0.5, 0.01, 0.0001 and 10^{-6}. These values were selected because they cover all the different versions of long time behaviour shown in the simulations with dependent noise. Note that in the simulations shown before, for dependent noise, one step in the simulation meant that there was one interaction between neighbours and noise was added to one individual. Now, one step in the simulation can either be an interaction between neighbours or adding noise to one individual. Thus, what would have been denoted as one step before is now two steps.

Algorithm 13 Simulating the Uniform Compass Model with Independent Bi-Modal Noise

let N be the number of individuals
(η_i) denote the configurations of the Compass Model
η_0 is the starting configuration chosen uniformly at random
ϵ is the parameter used to calculate by which amount an opinion gets changed when it is perturbed by noise
calculate the first m interaction times between neighbours $t^{neighbours}$ and interaction places $p^{neighbours}$ using 1 for N Poisson clocks
calculate the first m times t^{noise} and places p^{noise} where noise is added to individuals using 1 for $N+1$ Poisson clocks
$i \leftarrow 0$
$j \leftarrow 0$
while $i \leq m \ \& \ j \leq m$ **do**
  if $t_i^{neighbours} < t_j^{noise}$ **then**
    execute i'th step of compass model to compute the configuration $\eta_{t_i^{neighbours}}$ from the previously calculated configuration using Algorithm 3
  else
    let I denote the individual p_j^{noise} and I_o its current opinion
    choose D uniformly at random from $\{0, 1\}$
    if $D = 0$ **then**
      if $I_o = 1$ **then**
        draw u uniformly at random from $\{0, 1\}$
        if $u = 0$ **then**
          set I_o to $\epsilon - 1$
        else
          set I_o to $1 - \epsilon$
        end if
      else
        multiply I_o by $1 - \epsilon$
      end if
    else
      if $I_o = 0$ **then**
        draw u uniformly at random from $\{0, 1\}$
        if $u = 0$ **then**
          set I_o to ϵ
        else
          set I_o to $-\epsilon$
        end if
      else
        if $I_o > 0$ **then**
          multiply I_o by $1 - \epsilon$ and add ϵ
        else
          multiply I_o by $1 - \epsilon$ and subtract ϵ
        end if
      end if
    end if
  end if
end while

In Figures 8.44 and 8.45, we can see the values of the parameter c obtained in these simulations. One can see that for $\epsilon = 0.5$ the value of c first quickly goes down to a value at around 35 but then keeps oscillating significantly around this value. For $\epsilon = 0.01$, the values of c quickly fall below 10 and then oscillate around that value. The oscillation observed here is much smaller than for $\epsilon = 0.5$. For $\epsilon = 0.0001$ and $\epsilon = 10^{-6}$, the values of c fall even faster and further to a value close to 0. They also oscillate even less.

Figures 8.46 and 8.47 visualise the order parameter r. For $\epsilon = 0.5$, we can see that after approximately $0.5 \cdot 10^8$ steps, r oscillates wildly around 0.8. This means that the opinions synchronise somewhat, but not fully. For $\epsilon = 0.01$, for a few simulations, r oscillates slightly around a value close to 1. This means that for those the simulations the opinions have nearly synchronised, that is consensus has nearly been reached. However, for most simulations, the values of r stay well below 0.6, meaning there is no consensus. For $\epsilon = 0.0001$, r approaches 1 for all synchronisation. After approximately $0.5 \cdot 10^8$ steps, every simulation has approximately reached consensus. For $\epsilon = 10^{-6}$, r approaches 1 even faster than for $\epsilon = 0.0001$.

The results for the order parameter θ are shown in Figures 8.48 and 8.49. For $\epsilon = 0.5$, we observer that after approximately $0.5 \cdot 10^8$ the values of θ cluster around -0.5 and 0.5. However, they still oscillate significantly. For $\epsilon = 0.01$, θ only oscillates around -0.5 or 0.5 for a few simulations, most simulations do not seem to converge at all. This corresponds to the behaviour observed for the order parameter r. For $\epsilon = 0.0001$, in every simulation θ converges to either -0.5 or 0.5. There is little oscillation around these values. For smaller ϵ, the same convergence takes place, just a lot slower.

In Figure 8.50 and Figure 8.51, one can see the configurations calculated after approximately 10^8 steps. As expected from the results for r, θ and c, those configurations are very uneven lines clustered around -0.5 and 0.5 for $\epsilon = 0.5$. For $\epsilon = 0.0001$ those lines are nearly even and nearly at -0.5 and 0.5. For $\epsilon = 10^6$ we saw earlier that the r was nearly at 1. Thus, we expect reasonable even lines. And because the values of θ are not quite converged, we only expect them to be clustered around -0.5 and 0.5. Furthermore we expect the lines to be at the same height as θ at the end of the simulation. This is precisely what we can observe in this figure. For $\epsilon = 0.01$, the previously considered parameters did not give a good indication what the final configurations should look like. We can see here that they are horizontal line segments at -0.5 and 0.5 that are connect at a steep slope. That means that most neighbours either have an opinion of approximately -0.5 or approximately 0.5. And the few individuals that live in between two such groups can have opinions in between.

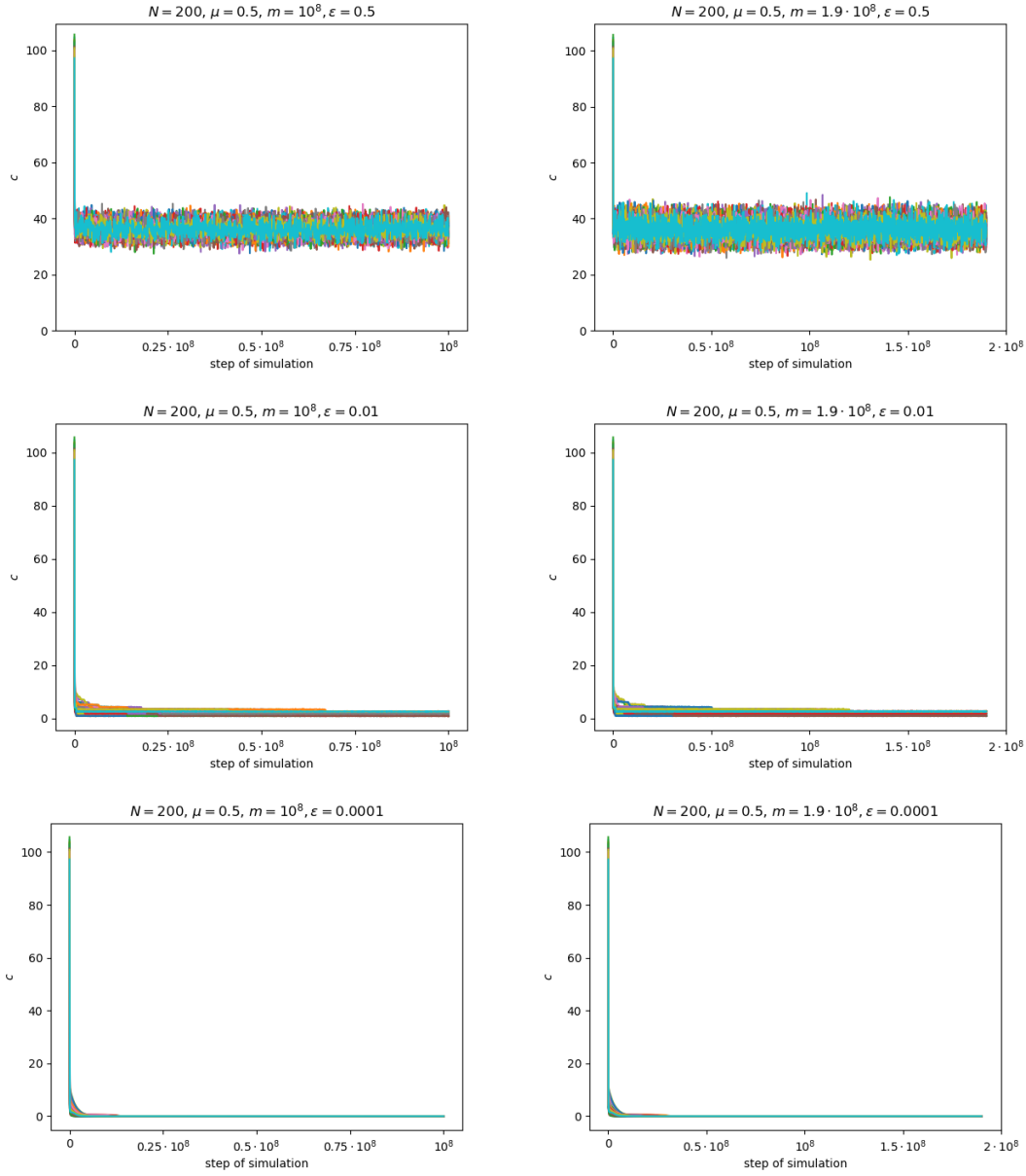

Figure 8.44.: These figures show the parameter c for the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$ and $\mu = 0.5$. Each coloured line represents one of 50 simulations with the same set of parameters. The figures in the first line are for $\epsilon = 0.5$, the second line is for $\epsilon = 0.01$ and the third line for $\epsilon = 0.0001$. Figures on the left are for the dependent case and the figures on the right for the independent case.

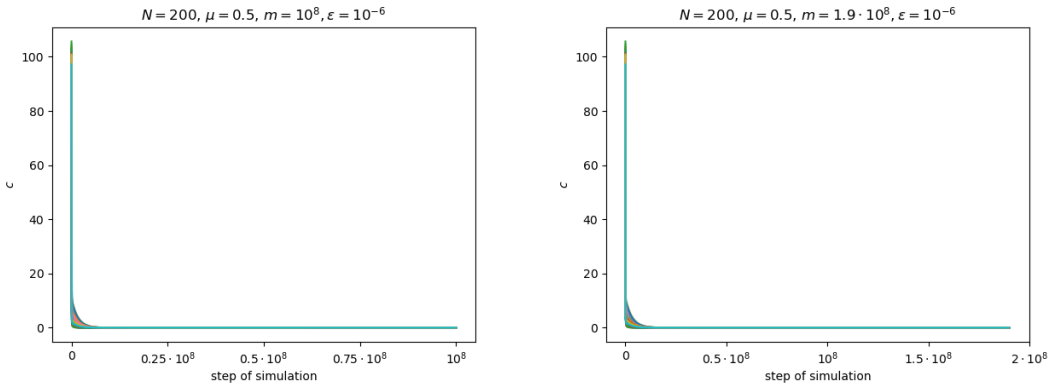

Figure 8.45.: These figures show the parameter c for the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$, $\mu = 0.5$ and $\epsilon = 10^{-6}$. Each coloured line represents one of 50 simulations with the same set of parameters. The figure on the left is for the dependent case and the figure on the right for the independent case.

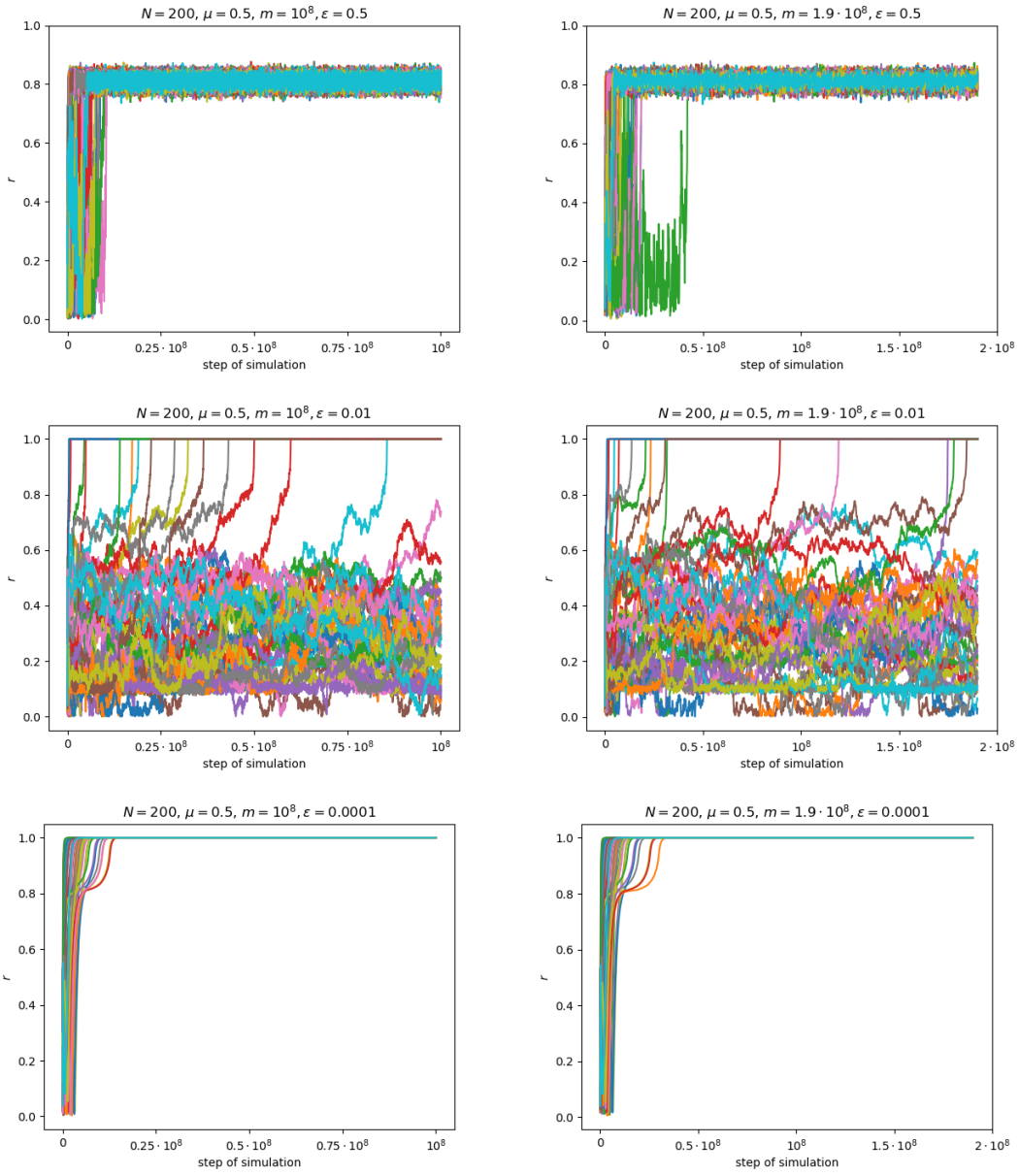

Figure 8.46.: These figures show the order parameter r for the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$ and $\mu = 0.5$. Each coloured line represents one of 50 simulations with the same set of parameters. The figures on the left hand side are for the dependent case and the figures on the right for the independent case. In the first row $\epsilon = 0.5$, in the second row $\epsilon = 0.01$, and in the third row $\epsilon = 0.0001$.

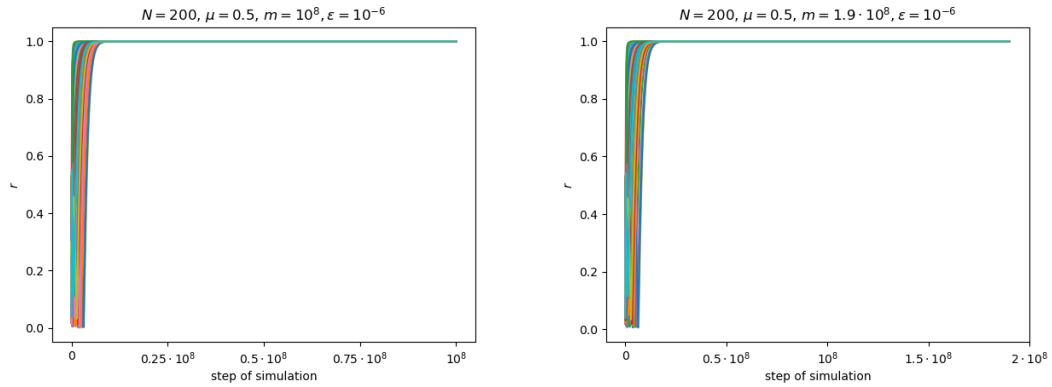

Figure 8.47.: These figures show the order parameter r for the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$ and $\mu = 0.5$ and $\epsilon = 10^{-6}$. Each coloured line represents one of 50 simulations with the same set of parameters. The figure on the left hand side is for the dependent case and the figures on the right for the independent case.

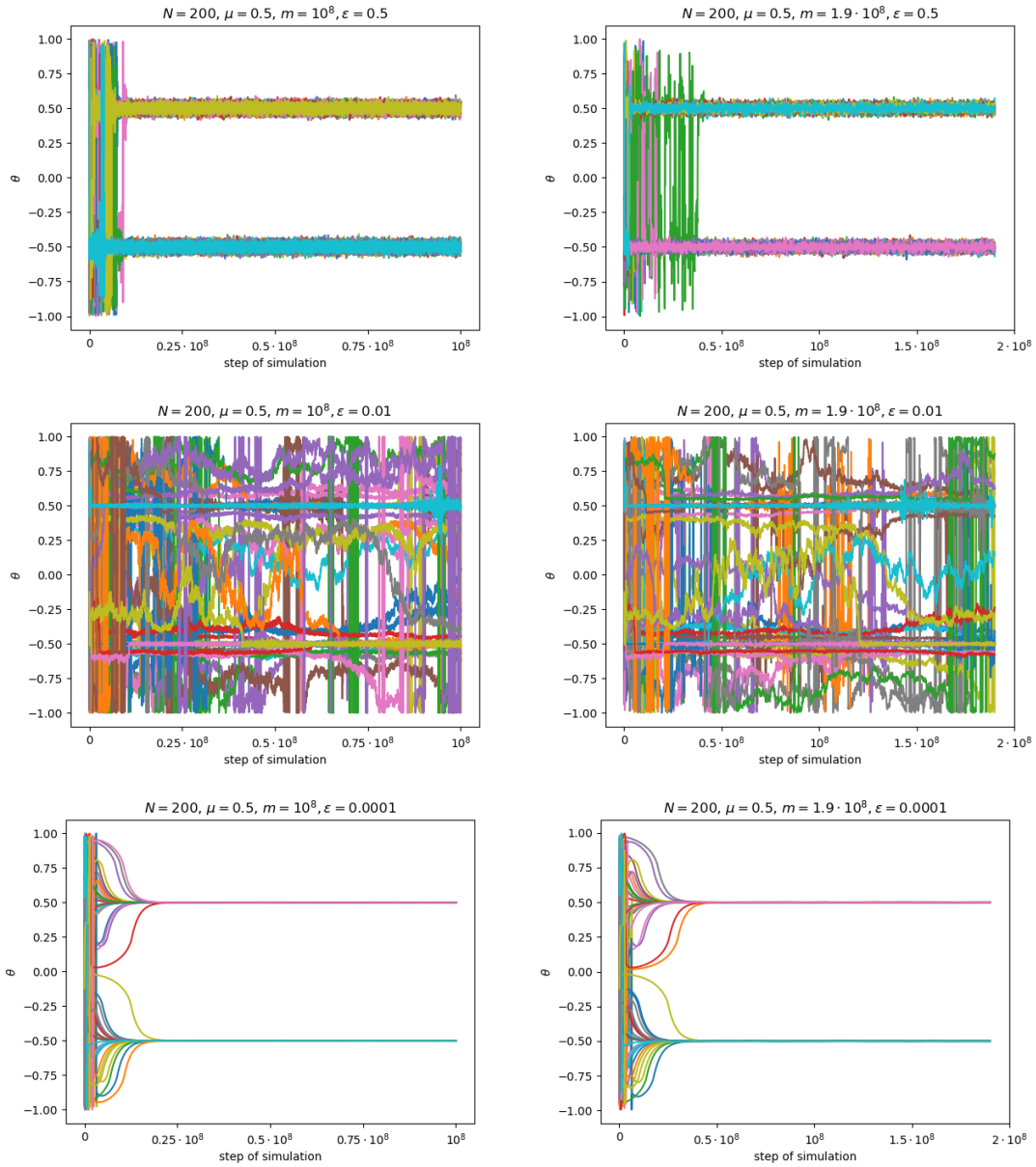

Figure 8.48.: These figures show the order parameter θ for the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$ and $\mu = 0.5$. Each coloured line represents one of 50 simulations with the same set of parameters. The figures on the left hand side are for the dependent case and the figures on the right for the independent case. In the first row $\epsilon = 0.5$, in the second row $\epsilon = 0.01$, and in the third row $\epsilon = 0.0001$.

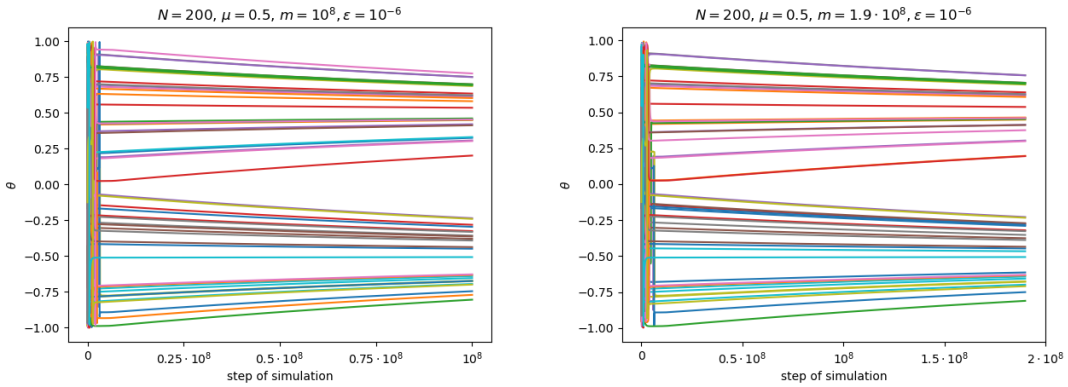

Figure 8.49.: These figures show the order parameter θ for the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$, $\mu = 0.5$ and $\epsilon = 10^{-6}$. Each coloured line represents one of 50 simulations with the same set of parameters. The figure on the left hand side is for the dependent case and the figure on the right for the independent case.

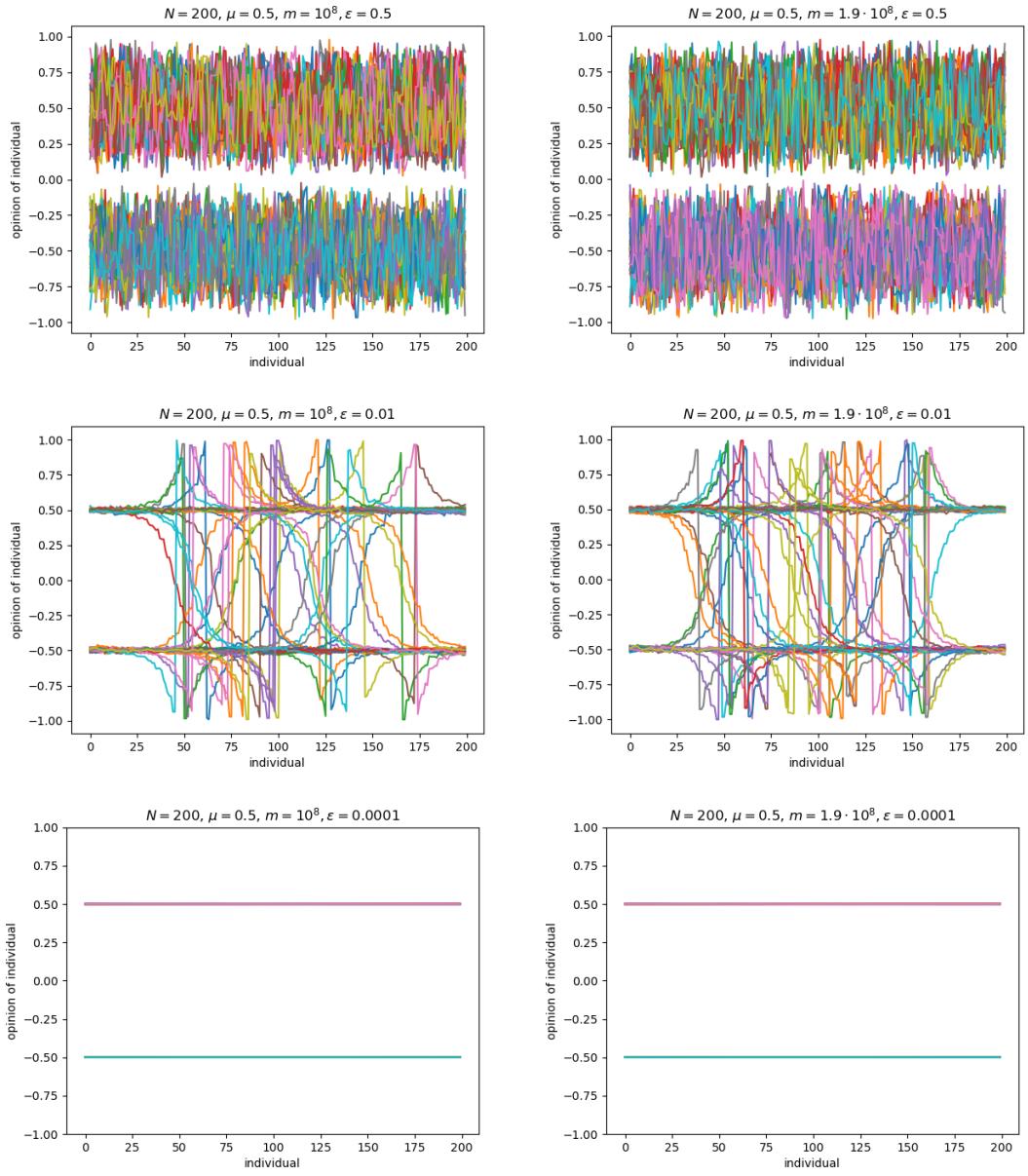

Figure 8.50.: These figures show the configuration calculated after approximately 10^8 steps in simulations of the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$ and $\mu = 0.5$. Each coloured line represents one of 50 simulations with the same set of parameters. The figures in the first line are for $\epsilon = 0.5$, in the second line for $\epsilon = 0.01$, and in the third line for $\epsilon = 0.0001$. Figures on the left represent the dependent case and the figures on the right the independent case.

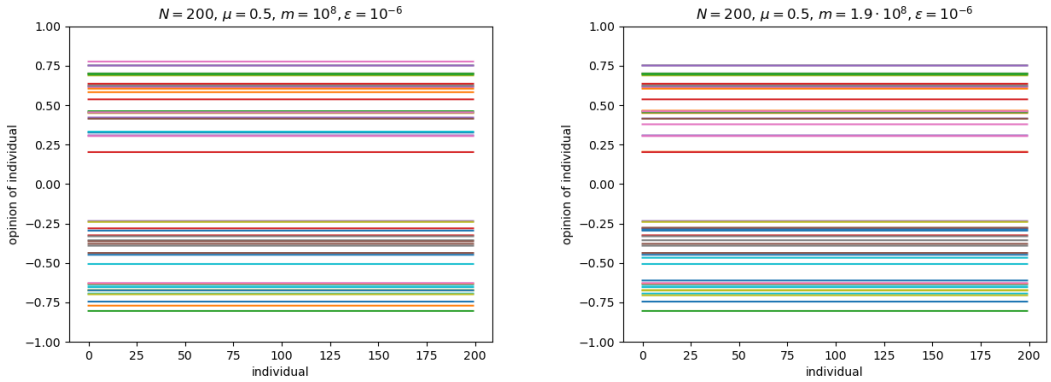

Figure 8.51.: These figures show the configuration calculated after approximately 10^8 steps in simulations of the Uniform Compass Model with (in)dependent bi-modal noise for $N = 200$, $\mu = 0.5$ and $\epsilon = 10^{-6}$. Each coloured line represents one of 50 simulations with the same set of parameters. The figure on the left represent the dependent case and the figure on the right the independent case.

As we can see in the figures, there are no significant qualitative differences. In order to find small quantitative differences, a more thorough comparison would be needed. However, this is not necessary, given our main goal is to study qualitative differences in the convergence behaviour.

We conclude that studying dependent noise as we did is not a major disadvantage over studying fully independent noise. This was not surprising. Due to the nature of the Poisson clocks, we expected that on average, with the independent noise the clocks for the noise and the ones for the interactions would take turn ringing. We would expect the clocks for the noise to ring slightly more often, that is N out $2N-1$ times. However, for large N that is not significant. Thus, implementing the noise dependently as we did in the previous sections, is a good method to safe computing time, storage and memory.

8.6. Comparison of the Deffuant and the Compass Model

In this section we will highlight some important differences between the Uniform Deffuant model and the Uniform Compass model.

One of these differences is the fact that in the Uniform Deffuant model we already know the opinion that the individuals eventually agree upon from the initial configuration, but for the Uniform Compass model we do not. In the Deffuant model, the opinion that individuals eventually agree upon is simply the average of the starting opinions as seen in Figure 8.7. However, for the Uniform Compass model this is not the case. In fact, with the same initial opinions but different interaction times and places, we can get consensus on a different opinion. This is visualised in Figure 8.52.

There is also a significant qualitative difference between the convergence behaviour of both models. In the Uniform Deffuant model, opinions converge to a value close to 0.5. However, in the Uniform Compass Model, the values, to which the opinions converge in different simulations, are uniformly distributed on the opinion space.

The uniform noise seems to have a very similar effect on both models. For small ϵ-values it has very little effect on the dynamics. In contrast, for large ϵ-values it prevents any sort of consensus and synchronisation of opinions.

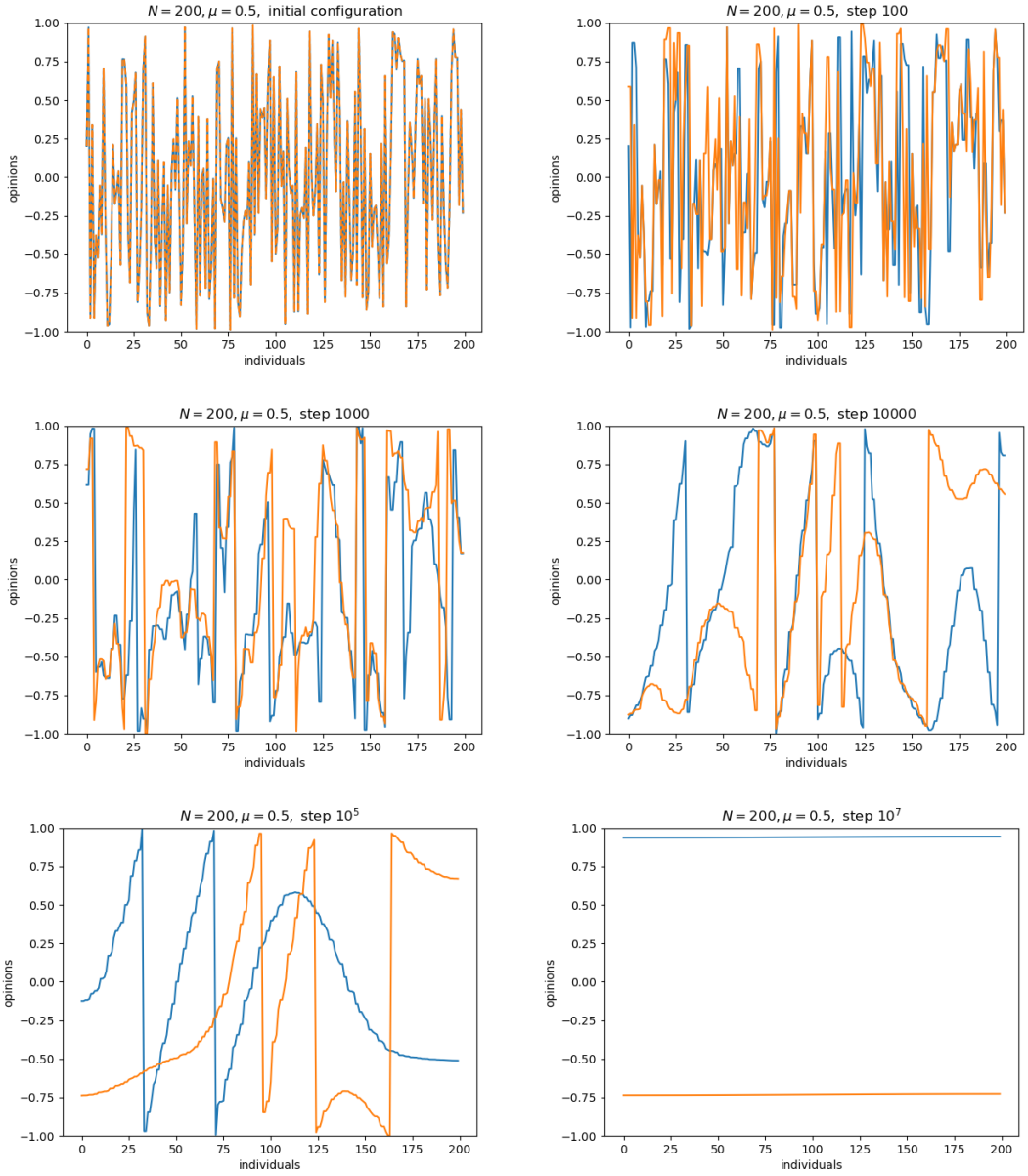

Figure 8.52.: In this figure we see configurations after a selected number of steps for two simulations of the Uniform Compass model without noise with $N = 200$ and $\mu = 0.5$. Each colour represents the results for one simulation. Both simulations use the same initial configuration.

9. Conclusion and Outlook

In this thesis we conducted numerical simulations of the Uniform Deffuant model and the Uniform Compass model on the finite path of length 200 and analysed the effect different types of noise have on the long-time behaviour of the models.

As seen in Sections 8.1 and 8.2, the simulations of both models without noise behave as expected from the theoretical results described in Part II.

In Sections 8.3.1 and 8.4.1, we analysed the effect of uniform noise on the models. We saw that both models react similarly to this kind of noise. Two regimes for ϵ can be observed. For sufficiently small values of ϵ, the noise has little impact on the behaviour of the model and approximate consensus is still reached. For large values of ϵ, consensus can no longer be reached. The critical value, for the models on the path of length 200 and with $\mu = 0.5$, is roughly $\epsilon \approx 0.01$.

The effect of bi-modal noise was studied in Sections 8.3.2 and 8.4.2. In contrast to uniform noise, bi-modal noise influences the Uniform Deffuant model and the Uniform Compass model differently. In the Uniform Deffuant model with bi-modal noise, for every ϵ, all opinions converge into a symmetric interval around 0.5. The width of this intervals decreases as ϵ decreases. There does not appear to be a phase transition. However, in the Uniform Compass model with bi-modal noise, there appears to be two distinct regimes. For large ϵ, the opinions converge to a (relatively large) interval around either -0.5 or 0.5. There does not appear to be a large degree of synchronisation. For ϵ-values under a critical value, for our simulations at around $\epsilon \approx 0.01$, the simulations converge to consensus on -0.5 or 0.5.

An open question that remains after writing this thesis is why we see local maxima in the graph of r for the Uniform Compass model without noise, see Figure 8.13. We investigated this topic, but unfortunately were unable to explain this phenomenon. Further research is needed to find a satisfactory answer.

Furthermore, additional study is necessary to analyse both models for different values of μ and larger groups of individuals. This thesis only investigated the case of unbounded confidence, uniform initial conditions and free boundary conditions. However, investigating the effect of non-trivial confidence parameters, non-uniform initial conditions and boundary conditions would also be interesting. In order to do this, it is necessary to develop more efficient algorithms, but this exceeded the scope of this thesis.

There are many interesting further research directions, both numerically and theoretically. Gaining understanding of the behaviour of the models through numerical simulations can help to prove theoretical results or provide direction for further theoretical research.

One obvious further direction is to analyse what happens with different types of noise. One could, for example, look at the effect of non-centralised uniform noise. In the Deffuant model we would expect that this type of noise, depending on the direction it favours, causes the opinions to converge to an interval around either 0 or 1. For the Compass model, this noise might be more interesting. Due to the opinions lying on a circle, it is not immediately clear what would happen and whether this noise would change the long-time behaviour significantly for all values of parameters. Another question one can ask, is, what would happen when instead of perturbing towards two opinions, that is 0 and 1 in the bi-modal noise, one would perturb towards a set of k opinions. Possibly, for small ϵ, this would simply lead to convergence towards an invariant measure of the form $\frac{1}{k}\left(\delta_{x_1} + \cdots + \delta_{x_k}\right)$ where $\delta_{x_1}, \ldots, \delta_{x_k}$ denote Dirac measures on flat configurations.

Yet another logical extension of this research project is to look at more underlying graphs. One can start with regular graphs, such as (finite subsets of) $\mathbb{Z}^d$ for $d = 1, 2, 3, \ldots$ or trees. Then one can move up to more general graphs. Furthermore, studying the models on random graphs would be particularly interesting. Random graphs are used to model and study the interactions of groups of people and would therefore help to create a more realistic opinion model. An introduction to the study of (social) networks can be found in [New18].

Another way of creating more realistic models is to look at non-Markovian dynamics. That allows the effect of memory on the opinion formation process to be taken into account. The recent papers [CP23] and [CWS$^+$20] introduce different approaches to studying a non-Markovian opinion model.

Part IV.

Appendix

A. Kolmogorov-Smirnov Test

The two-sided Kolmogorov-Smirnov-Test (short: KS-test), is a statistical goodness of fit test. It is applied in the following scenario.

Let $X_1, \ldots, X_n$ be independent and identically distributed random variables on $\mathbb{R}$ with common cumulative distribution function F. For a specific set of observations $y_1, \ldots, y_n$ let F_n be their empirical cumulative distribution function, defined as

$$F_n : \mathbb{R} \mapsto \mathbb{R}, \quad F_n(t) = \begin{cases} 0, & \text{if } t < \min\{y_1, \ldots, y_n\}, \\ \frac{|\{y_i | y_i \leq t, i=1,\ldots,n\}|}{n}, & \text{if } \min\{y_1, \ldots, y_n\} \leq t \leq \max\{y_1, \ldots, y_n\}, \\ 1, & \text{if } t > \max\{y_1, \ldots, y_n\} \end{cases} .$$

How likely is it, that $y_1, \ldots, y_n$ are sampled from $X_1, \ldots, X_n$? If n were large and $y_1, \ldots, y_n$ were indeed sampled from $X_1, \ldots, X_n$, then $F_n \approx F$. Thus, we test the following hypotheses:

$$\begin{aligned} H_0 : &\quad F = F_n \\ H_A : &\quad F \neq F_n. \end{aligned}$$

Intuitively spoken, one can be reasonably sure that $y_1, \ldots, y_n$ are sampled from $X_1, \ldots, X_n$, if F and F_n are relatively close to each other using a suitable distance measure. Such a distance measure is given by the Kolmogorov test statistic which was introduced in 1933 by Kolmogorov. It is given by

$$T_n = \sup_t |F(t) - F_n(t)|.$$

We reject the null-hypothesis H_0, if T_n is large than a critical value T. If T_n is smaller than T, we accept the null-hypothesis. The probability of a type-I-error, that is rejecting H_0 even though it is true, is called the significance level of the test. It is chosen before the test and denoted by α. The smallest level of significance at which the null-hypothesis is rejected is called the p-value of the test. We reject H_0 when the p-value is smaller than α. The p-value can be written as

$$\mathbb{P}(T_n > T | F = F_n).$$

The critical value T is random variable that returns the value of the Kolmogorov test statistic for a sample of $X_1, \ldots, X_n$. That is

$$T(x) = \sup_t |F(t) - \tilde{F}_n(t)|,$$

where $\tilde{F}_n$ is the cumulative distribution function of $X_1, \ldots, X_n$.

In order to perform this statistical test, one used to rely on tables. Nowadays, one can simply use the function `ks_test` from the Python package `scipy` to calculate the p-values. How this function calculated the p-values can be seen by studying its code on `git-hub`. First this function calculates the test statistic T_n as defined above. In order to calculate the p-value given T_n, the method outlined in [SL11] is used. [Bir52] [Ric95] [LR22] [Has22]

B. Python Code

The main parts of the code are formulated in classes. These classes then only need to be initiated and run. The classes for the Deffuant model without noise is given below as `udm`. The class for the Deffuant model with uniform noise is `udm_n1`, and with bi-modal noise it is `udm_n2`.

```
import numpy as np
import os
import math


class udm:
    def __init__(self, N, mu, steps, i, initial_configuration,
    interaction_order):
        self.i = i
        self.N = N
        self.mu = mu
        self.steps = steps
        self.config = initial_configuration
        self.interaction_order = interaction_order
        self.convergence = [0.0 for i in range(int(self.steps/100_000) +
    1)]
        self.convergence[0] = np.sum([abs(self.config[i]-self.config[i
    +1]) for i in range(self.N - 1)])
        self.r = np.zeros(int(self.steps/100_000) + 1)
        s = np.sum(np.sin(2*self.config*math.pi - math.pi))
        c = np.sum(np.cos(2*self.config*math.pi - math.pi))
        self.r[0] = np.sqrt(s**2 + c**2)/self.N
        self.theta = np.zeros(int(self.steps/100_000) + 1)
        self.theta[0] = np.arcsin(s/(self.N*self.r[0]))
        if (c < 0):
            if (self.theta[0] >= 0):
                self.theta[0] = math.pi - self.theta[0]
            else:
                self.theta[0] = - math.pi - self.theta[0]

    def run(self):
        for i in range(self.steps):
            interaction_point = self.interaction_order[i]
            old1 = self.config[interaction_point]
            old2 = self.config[interaction_point + 1]
            self.config[interaction_point] = (1-self.mu)*old1 + self.mu*
    old2
            self.config[interaction_point + 1] = (1-self.mu)*old2 + self.
    mu*old1
            if ((i+1) % 100_000 == 0):
```

```
                self.convergence[int((i+1)/100_000)] = np.sum([abs(self.
    config[j]-self.config[j+1]) for j in range(self.N - 1)])
                s = np.sum(np.sin(2*self.config*math.pi - math.pi))
                c = np.sum(np.cos(2*self.config*math.pi - math.pi))
                self.r[int((i+1)/100_000)] = np.sqrt(s**2 + c**2)/self.N
                self.theta[int((i+1)/100_000)] = np.arcsin(s/(self.N*self
    .r[int((i+1)/100_000)]))
                if (c < 0):
                    if (self.theta[int((i+1)/100_000)] >= 0):
                        self.theta[int((i+1)/100_000)] = math.pi - self.
    theta[int((i+1)/100_000)]
                    else:
                        self.theta[int((i+1)/100_000)] = - math.pi - self
    .theta[int((i+1)/100_000)]
        name = str(self.N) + '_0' + str(self.mu) + '_' + str(self.steps)
        folder_name = 'udm_version'+ str(self.i) + '_' + name + '/'
        os.mkdir(folder_name)
        np.save(folder_name + 'c_' + name, self.convergence)
        np.save(folder_name + 'r_' + name, self.r)
        np.save(folder_name + 'theta_' + name, self.theta)
        np.save(folder_name + 'final_config_' + name, self.config)

class udm_n1:
    def __init__(self, N, mu, steps, epsilon, i, initial_configuration,
                 interaction_order, noise_loc):
        self.i = i
        self.N = N
        self.mu = mu
        self.steps = steps
        self.epsilon = epsilon
        self.config = initial_configuration
        self.interaction_order = interaction_order
        self.noise = np.random.default_rng().uniform(-epsilon, epsilon,
    self.steps)
        self.noise_loc = noise_loc
        self.convergence = [0.0 for i in range(int(self.steps/100_000) +
    1)]
        self.convergence[0] = np.sum([abs(self.config[i]-self.config[i
    +1]) for i in range(self.N - 1)])
        self.r = np.zeros(int(self.steps/100_000) + 1)
        s = np.sum(np.sin(2*self.config*math.pi - math.pi))
        c = np.sum(np.cos(2*self.config*math.pi - math.pi))
        self.r[0] = np.sqrt(s**2 + c**2)/self.N
        self.theta = np.zeros(int(self.steps/100_000) + 1)
        self.theta[0] = np.arcsin(s/(self.N*self.r[0]))
        if (c < 0):
            if (self.theta[0] >= 0):
                self.theta[0] = math.pi - self.theta[0]
            else:
                self.theta[0] = - math.pi - self.theta[0]

    def run(self):
        for i in range(self.steps):
            interaction_point = self.interaction_order[i]
```

```
83              old1 = self.config[interaction_point]
84              old2 = self.config[interaction_point + 1]
85              self.config[interaction_point] = (1-self.mu)*old1 + self.mu*
    old2
86              self.config[interaction_point + 1] = (1-self.mu)*old2 + self.
    mu*old1
87              self.config[self.noise_loc[i]] = self.config[self.noise_loc[i
    ]] + self.noise[i]
88              if (self.config[self.noise_loc[i]]>1.0):
89                  self.config[self.noise_loc[i]] = 1.0
90              if (self.config[self.noise_loc[i]]< 0.0):
91                  self.config[self.noise_loc[i]] = 0.0
92              if ((i+1) % 100_000 == 0):
93                  self.convergence[int((i+1)/100_000)] = np.sum([abs(self.
    config[j]-self.config[j+1]) for j in range(self.N - 1)])
94                  s = np.sum(np.sin(2*self.config*math.pi - math.pi))
95                  c = np.sum(np.cos(2*self.config*math.pi - math.pi))
96                  self.r[int((i+1)/100_000)] = np.sqrt(s**2 + c**2)/self.N
97                  self.theta[int((i+1)/100_000)] = np.arcsin(s/(self.N*self
    .r[int((i+1)/100_000)]))
98                  if (c < 0):
99                      if (self.theta[int((i+1)/100_000)] >= 0):
100                         self.theta[int((i+1)/100_000)] = math.pi - self.
    theta[int((i+1)/100_000)]
101                     else:
102                         self.theta[int((i+1)/100_000)] = - math.pi - self
    .theta[int((i+1)/100_000)]
103         name = str(self.N) + '_0' + str(self.mu) + '_' + str(self.steps)
    + str(self.epsilon)
104         folder_name = 'udm_n1_version'+ str(self.i) + '_' + name + '/'
105         os.mkdir(folder_name)
106         np.save(folder_name + 'c_' + name, self.convergence)
107         np.save(folder_name + 'r_' + name, self.r)
108         np.save(folder_name + 'theta_' + name, self.theta)
109         np.save(folder_name + 'final_config_' + name, self.config)
110
111
112 class udm_n2:
113     def __init__(self, N, mu, steps, epsilon, i, initial_configuration,
114                  interaction_order, noise_loc, noise_dir):
115         self.i = i
116         self.N = N
117         self.mu = mu
118         self.steps = steps
119         self.epsilon = epsilon
120         self.config = initial_configuration
121         self.interaction_order = interaction_order
122         self.noise_loc = noise_loc
123         self.noise_dir = noise_dir
124         self.convergence = [0.0 for i in range(int(self.steps/100_000) +
    1)]
125         self.convergence[0] = np.sum([abs(self.config[i]-self.config[i
    +1]) for i in range(self.N - 1)])
126         self.r = np.zeros(int(self.steps/100_000) + 1)
```

```
127         s = np.sum(np.sin(2*self.config*math.pi - math.pi))
128         c = np.sum(np.cos(2*self.config*math.pi - math.pi))
129         self.r[0] = np.sqrt(s**2 + c**2)/self.N
130         self.theta = np.zeros(int(self.steps/100_000) + 1)
131         self.theta[0] = np.arcsin(s/(self.N*self.r[0]))
132         if (c < 0):
133             if (self.theta[0] >= 0):
134                 self.theta[0] = math.pi - self.theta[0]
135             else:
136                 self.theta[0] = - math.pi - self.theta[0]
137 
138     def run(self):
139         for i in range(self.steps):
140             interaction_point = self.interaction_order[i]
141             old1 = self.config[interaction_point]
142             old2 = self.config[interaction_point + 1]
143             self.config[interaction_point] = (1-self.mu)*old1 + self.mu*
    old2
144             self.config[interaction_point + 1] = (1-self.mu)*old2 + self.
    mu*old1
145             self.config[self.noise_loc[i]] = (1-self.epsilon)*self.config
    [self.noise_loc[i]] + self.epsilon*self.noise_dir[i]
146             if ((i+1) % 100_000 == 0):
147                 self.convergence[int((i+1)/100_000)] = np.sum([abs(self.
    config[j]-self.config[j+1]) for j in range(self.N - 1)])
148                 s = np.sum(np.sin(2*self.config*math.pi - math.pi))
149                 c = np.sum(np.cos(2*self.config*math.pi - math.pi))
150                 self.r[int((i+1)/100_000)] = np.sqrt(s**2 + c**2)/self.N
151                 self.theta[int((i+1)/100_000)] = np.arcsin(s/(self.N*self
    .r[int((i+1)/100_000)]))
152                 if (c < 0):
153                     if (self.theta[int((i+1)/100_000)] >= 0):
154                         self.theta[int((i+1)/100_000)] = math.pi - self.
    theta[int((i+1)/100_000)]
155                     else:
156                         self.theta[int((i+1)/100_000)] = - math.pi - self
    .theta[int((i+1)/100_000)]
157         name = str(self.N) + '_0' + str(self.mu) + '_' + str(self.steps)
    + str(self.epsilon)
158         folder_name = 'udm_n2_version'+ str(self.i) + '_' + name + '/'
159         os.mkdir(folder_name)
160         np.save(folder_name + 'c_' + name, self.convergence)
161         np.save(folder_name + 'r_' + name, self.r)
162         np.save(folder_name + 'theta_' + name, self.theta)
163         np.save(folder_name + 'final_config_' + name, self.config)
```

The classes for the Compass model without noise, with uniform noise and with bi-modal noise are given below as `ucm`, `ucm_n1` and `ucm_n2`, respectively. The class for the version of the Compass model with independent bi-modal noise discussed in Section 8.5 is called `ucm_independent_n2`.

```
import numpy as np
import os
import math

class ucm_n2:
    # Uniform compass model with noise
    # if two_sided, then there are two opinion (0.0 and 1.0) towards
    # which opinions get disturbed with equal probability.
    # if one_sided, we only disturb towards 0.0 with amount epsilon
    def __init__(self, N, mu, steps, epsilon, i, initial_configuration,
    interaction_order, noise_loc, noise_dir):
        self.i = i
        self.a = -1.0
        self.b = 1.0
        self.N = N
        self.mu = mu
        self.steps = steps
        self.epsilon = epsilon
        self.config = initial_configuration
        self.interaction_order = interaction_order
        self.convergence = np.zeros(int(self.steps/100_000) + 1)
        self.convergence[0] = np.sum([abs(self.config[i]-self.config[i
    +1]) if(abs(self.config[i]-self.config[i+1])<1.0) else 2.0-abs(self.
    config[i]-self.config[i+1]) for i in range(self.N - 1)])
        self.noise_loc = noise_loc
        self.noise_dir = noise_dir
        self.r = np.zeros(int(self.steps/10_000) + 1)
        s = np.sum(np.sin(self.config*math.pi))
        c = np.sum(np.cos(self.config*math.pi))
        self.r[0] = np.sqrt(s**2 + c**2)/self.N
        self.theta = np.zeros(int(self.steps/10_000) + 1)
        self.theta[0] = np.arcsin(s/(self.N*self.r[0]))
        if (c < 0):
            if (self.theta[0] >= 0):
                self.theta[0] = math.pi - self.theta[0]
            else:
                self.theta[0] = - math.pi - self.theta[0]

    def run(self):
        for i in range(self.steps):
            interaction_point = self.interaction_order[i]
            old1 = self.config[interaction_point]
            old2 = self.config[interaction_point + 1]
            if (abs(old1-old2) == 1.0):
                if (np.random.default_rng().integers(0,1)==0):
                    self.config[interaction_point] = old1 + self.mu * np.
    sign(old1)
                    self.config[interaction_point + 1] = old2 + self.mu *
     np.sign(old2)
                    if (self.config[interaction_point]>self.b):
                        self.config[interaction_point] = self.config[
    interaction_point] - 2.0
                    if (self.config[interaction_point + 1]>self.b):
                        self.config[interaction_point + 1] = self.config[
```

```
    interaction_point + 1] - 2.0
49                  if (self.config[interaction_point]<=self.a):
50                      self.config[interaction_point] = self.config[
    interaction_point] + 2.0
51                  if (self.config[interaction_point + 1]<=self.a):
52                      self.config[interaction_point + 1] = self.config[
    interaction_point + 1] + 2.0
53              else:
54                  self.config[interaction_point] = old1 - self.mu * np.
    sign(old1)
55                  self.config[interaction_point] = old2 - self.mu * np.
    sign(old2)
56          else:
57              if (abs(old1-old2)<1.0):
58                  self.config[interaction_point] = old1 + self.mu * (
    old2 - old1)
59                  self.config[interaction_point + 1] = old2 + self.mu *
     (old1 - old2)
60              else:
61                  self.config[interaction_point] = old1 + self.mu *
    (2.0 - abs(old2 - old1))*np.sign(old1)
62                  self.config[interaction_point + 1] = old2 + self.mu *
     (2.0 - abs(old1 - old2))*np.sign(old2)
63                  if (self.config[interaction_point]>self.b):
64                      self.config[interaction_point] = self.config[
    interaction_point] - 2.0
65                  if (self.config[interaction_point + 1]>self.b):
66                      self.config[interaction_point + 1] = self.config[
    interaction_point + 1] - 2.0
67                  if (self.config[interaction_point]<=self.a):
68                      self.config[interaction_point] = self.config[
    interaction_point] + 2.0
69                  if (self.config[interaction_point + 1]<=self.a):
70                      self.config[interaction_point + 1] = self.config[
    interaction_point + 1] + 2.0
71          for j in range(int(self.num_disturbed * i), int(self.
    num_disturbed*(i+1))):
72              nl = self.noise_loc[j]
73              nd = self.noise_dir[j]
74              if (nd == 0):
75                  if (self.config[nl] == 1.0):
76                      if (np.random.default_rng().integers(0,1) == 0):
77                          self.config[nl] = -1.0 + self.epsilon
78                      else:
79                          self.config[nl] = 1.0 - self.epsilon
80                  else:
81                      self.config[nl] = (1.0-self.epsilon)*self.config[
    nl]
82              else:
83                  if (self.config[nl] == 0.0):
84                      if (np.random.default_rng().integers(0,1) == 0):
85                          self.config[nl] = self.epsilon
86                      else:
87                          self.config[nl] = -self.epsilon
```

```
88                      else:
89                          if (self.config[nl] > 0.0):
90                              self.config[nl] = (1-self.epsilon)*self.
    config[nl] + self.epsilon
91                          else:
92                              self.config[nl] = (1-self.epsilon)*self.
    config[nl] - self.epsilon
93              if ((i+1) % 100_000 == 0):
94                  self.convergence[int((i+1)/100_000)] = np.sum([abs(self.
    config[j]-self.config[j+1]) if(abs(self.config[j]-self.config[j+1])
    <1.0) else 2.0-abs(self.config[j]-self.config[j+1]) for j in range(
    self.N - 1)])
95              if ((i+1) % 10_000 == 0):
96                  s = np.sum(np.sin(self.config*math.pi))
97                  c = np.sum(np.cos(self.config*math.pi))
98                  self.r[int((i+1)/10000)] = np.sqrt(s**2 + c**2)/self.N
99                  self.theta[int((i+1)/10000)] = np.arcsin(s/(self.N*self.r
    [int((i+1)/10000)]))
100                 if (c < 0):
101                     if (self.theta[int((i+1)/10000)] >= 0):
102                         self.theta[int((i+1)/10000)] = math.pi - self.
    theta[int((i+1)/10000)]
103                     else:
104                         self.theta[int((i+1)/10000)] = - math.pi - self.
    theta[int((i+1)/10000)]
105         name = str(self.N) + '_0' + str(self.mu) + '_' + str(self.steps)
    + '_' + str(self.epsilon)
106         folder_name = 'ucm_n2_version'+ str(self.i) + '_' + name + '/'
107         os.mkdir(folder_name)
108         np.save(folder_name + 'c_' + name, self.convergence)
109         np.save(folder_name + 'r_' + name, self.r)
110         np.save(folder_name + 'theta_' + name, self.theta)
111         np.save(folder_name + 'last_configuration_' + name, self.config)
112
113
114 class ucm_independent_n2:
115     # Uniform compass model with noise
116     # there are two opinion (0.0 and 1.0) towards
117     # which opinions get disturbed with equal probability.
118     # times for interactions and noise are independent
119     def __init__(self, N, mu, steps, epsilon, i, initial_configuration,
120                  interaction_order, interaction_time,
    interaction_order_noise,
121                  interaction_time_noise, noise_loc, noise_dir):
122         self.i = i
123         self.a = -1.0
124         self.b = 1.0
125         self.N = N
126         self.mu = mu
127         self.steps = steps
128         self.epsilon = epsilon
129         self.config = initial_configuration
130         self.interaction_order = interaction_order
131         self.interaction_time = interaction_time
```

```
132         self.interaction_order_noise = interaction_order_noise
133         self.interaction_time_noise = interaction_time_noise
134         self.convergence = np.zeros(int(self.steps/50_000) + 1)
135         self.convergence[0] = np.sum([abs(self.config[i]-self.config[i
    +1]) if(abs(self.config[i]-self.config[i+1])<1.0) else 2.0-abs(self.
    config[i]-self.config[i+1]) for i in range(self.N - 1)])
136         self.noise_loc = noise_loc
137         self.noise_dir = noise_dir
138         self.r = np.zeros(int(self.steps/50_000) + 1)
139         s = np.sum(np.sin(self.config*math.pi))
140         c = np.sum(np.cos(self.config*math.pi))
141         self.r[0] = np.sqrt(s**2 + c**2)/self.N
142         self.theta = np.zeros(int(self.steps/50_000) + 1)
143         self.theta[0] = np.arcsin(s/(self.N*self.r[0]))
144         if (c < 0):
145             if (self.theta[0] >= 0):
146                 self.theta[0] = math.pi - self.theta[0]
147             else:
148                 self.theta[0] = - math.pi - self.theta[0]
149 
150     def run(self):
151         neighbours = 1
152         noise = 1
153         i = -1
154         while ((neighbours<self.steps) & (noise<self.steps)):
155             i = i + 1
156             if (self.interaction_time[neighbours] <= self.
    interaction_time_noise[noise]):
157                 interaction_point = self.interaction_order[neighbours]
158                 old1 = self.config[interaction_point]
159                 old2 = self.config[interaction_point + 1]
160                 if (abs(old1-old2) == 1.0):
161                     if (np.random.default_rng().integers(0,1)==0):
162                         self.config[interaction_point] = old1 + self.mu *
     np.sign(old1)
163                         self.config[interaction_point + 1] = old2 + self.
    mu * np.sign(old2)
164                         if (self.config[interaction_point]>self.b):
165                             self.config[interaction_point] = self.config[
    interaction_point] - 2.0
166                         if (self.config[interaction_point + 1]>self.b):
167                             self.config[interaction_point + 1] = self.
    config[interaction_point + 1] - 2.0
168                         if (self.config[interaction_point]<=self.a):
169                             self.config[interaction_point] = self.config[
    interaction_point] + 2.0
170                         if (self.config[interaction_point + 1]<=self.a):
171                             self.config[interaction_point + 1] = self.
    config[interaction_point + 1] + 2.0
172                     else:
173                         self.config[interaction_point] = old1 - self.mu *
     np.sign(old1)
174                         self.config[interaction_point] = old2 - self.mu *
     np.sign(old2)
```

```
175                 else:
176                     if (abs(old1-old2)<1.0):
177                         self.config[interaction_point] = old1 + self.mu *
     (old2 - old1)
178                         self.config[interaction_point + 1] = old2 + self.
    mu * (old1 - old2)
179                     else:
180                         self.config[interaction_point] = old1 + self.mu *
     (2.0 - abs(old2 - old1))*np.sign(old1)
181                         self.config[interaction_point + 1] = old2 + self.
    mu * (2.0 - abs(old1 - old2))*np.sign(old2)
182                     if (self.config[interaction_point]>self.b):
183                         self.config[interaction_point] = self.config[
    interaction_point] - 2.0
184                     if (self.config[interaction_point + 1]>self.b):
185                         self.config[interaction_point + 1] = self.
    config[interaction_point + 1] - 2.0
186                     if (self.config[interaction_point]<=self.a):
187                         self.config[interaction_point] = self.config[
    interaction_point] + 2.0
188                     if (self.config[interaction_point + 1]<=self.a):
189                         self.config[interaction_point + 1] = self.
    config[interaction_point + 1] + 2.0
190                 if ((i+1) % 100_000 == 0):
191                     self.convergence[int((i+1)/100_000)] = np.sum([abs(
    self.config[j]-self.config[j+1]) if(abs(self.config[j]-self.config[j
    +1])<1.0) else 2.0-abs(self.config[j]-self.config[j+1]) for j in range
    (self.N - 1)])
192                     s = np.sum(np.sin(self.config*math.pi))
193                     c = np.sum(np.cos(self.config*math.pi))
194                     self.r[int((i+1)/100_000)] = np.sqrt(s**2 + c**2)/
    self.N
195                     self.theta[int((i+1)/100_000)] = np.arcsin(s/(self.N*
    self.r[int((i+1)/100_000)]))
196                     if (c < 0):
197                         if (self.theta[int((i+1)/100_000)] >= 0):
198                             self.theta[int((i+1)/100_000)] = math.pi -
    self.theta[int((i+1)/100_000)]
199                         else:
200                             self.theta[int((i+1)/100_000)] = - math.pi -
    self.theta[int((i+1)/100_000)]
201                 neighbours = neighbours + 1
202             else:
203                 nl = self.noise_loc[noise]
204                 nd = self.noise_dir[noise]
205                 if (nd == 0):
206                     if (self.config[nl] == 1.0):
207                         if (np.random.default_rng().integers(0,1) == 0):
208                             self.config[nl] = -1.0 + self.epsilon
209                         else:
210                             self.config[nl] = 1.0 - self.epsilon
211                     else:
212                         self.config[nl] = (1.0-self.epsilon)*self.config[
    nl]
```

```
                else:
                    if (self.config[nl] == 0.0):
                        if (np.random.default_rng().integers(0,1) == 0):
                            self.config[nl] = self.epsilon
                        else:
                            self.config[nl] = -self.epsilon
                    else:
                        if (self.config[nl] > 0.0):
                            self.config[nl] = (1-self.epsilon)*self.config[nl] + self.epsilon
                        else:
                            self.config[nl] = (1-self.epsilon)*self.config[nl] - self.epsilon
                if ((i+1) % 100_000 == 0):
                    self.convergence[int((i+1)/100_000)] = np.sum([abs(self.config[j]-self.config[j+1]) if(abs(self.config[j]-self.config[j+1])<1.0) else 2.0-abs(self.config[j]-self.config[j+1]) for j in range(self.N - 1)])
                    s = np.sum(np.sin(self.config*math.pi))
                    c = np.sum(np.cos(self.config*math.pi))
                    self.r[int((i+1)/100_000)] = np.sqrt(s**2 + c**2)/self.N
                    self.theta[int((i+1)/100_000)] = np.arcsin(s/(self.N*self.r[int((i+1)/100_000)]))
                    if (c < 0):
                        if (self.theta[int((i+1)/100_000)] >= 0):
                            self.theta[int((i+1)/100_000)] = math.pi - self.theta[int((i+1)/100_000)]
                        else:
                            self.theta[int((i+1)/100_000)] = - math.pi - self.theta[int((i+1)/100_000)]
                noise = noise + 1
        name = str(self.N) + '_0' + str(self.mu) + '_' + str(self.steps) + '_' + str(self.epsilon)
        folder_name = 'ucm_independent_n2_version'+ str(self.i) + '_' + name + '/'
        os.mkdir(folder_name)
        np.save(folder_name + 'c_' + name, self.convergence)
        np.save(folder_name + 'r_' + name, self.r)
        np.save(folder_name + 'theta_' + name, self.theta)
        np.save(folder_name + 'last_configuration_' + name, self.config)

class ucm_n1:
    def __init__(self, N, mu, steps, epsilon, i, initial_configuration,
                 interaction_order, noise_loc):
        self.i = i
        self.N = N
        self.mu = mu
        self.steps = steps
        self.epsilon = epsilon
        self.config = initial_configuration
        self.interaction_order = interaction_order
        self.convergence = np.zeros(int(self.steps/100_000) + 1)
        self.convergence[0] = np.sum([abs(self.config[i]-self.config[i
```

```
    +1]) if(abs(self.config[i]-self.config[i+1])<1.0) else 2.0-abs(self.
    config[i]-self.config[i+1]) for i in range(self.N - 1)])
256         self.noise = np.random.default_rng().uniform(-epsilon, epsilon,
    self.steps)
257         self.noise_loc = noise_loc
258         self.r = np.zeros(int(self.steps/100_000) + 1)
259         s = np.sum(np.sin(self.config*math.pi))
260         c = np.sum(np.cos(self.config*math.pi))
261         self.r[0] = np.sqrt(s**2 + c**2)/self.N
262         self.theta = np.zeros(int(self.steps/100_000) + 1)
263         self.theta[0] = np.arcsin(s/(self.N*self.r[0]))
264         if (c < 0):
265             if (self.theta[0] >= 0):
266                 self.theta[0] = math.pi - self.theta[0]
267             else:
268                 self.theta[0] = - math.pi - self.theta[0]
269 
270     def run(self):
271         for i in range(self.steps):
272             interaction_point = self.interaction_order[i]
273             old1 = self.config[interaction_point]
274             old2 = self.config[interaction_point + 1]
275             if (abs(old1-old2) == 1.0):
276                 if (np.random.default_rng().integers(0,1)==0):
277                     self.config[interaction_point] = old1 + self.mu * np.
    sign(old1)
278                     self.config[interaction_point + 1] = old2 + self.mu *
     np.sign(old2)
279                     if (self.config[interaction_point]>1.0):
280                         self.config[interaction_point] = self.config[
    interaction_point] - 2.0
281                     if (self.config[interaction_point + 1]>1.0):
282                         self.config[interaction_point + 1] = self.config[
    interaction_point + 1] - 2.0
283                     if (self.config[interaction_point]<=-1.0):
284                         self.config[interaction_point] = self.config[
    interaction_point] + 2.0
285                     if (self.config[interaction_point + 1]<=-1.0):
286                         self.config[interaction_point + 1] = self.config[
    interaction_point + 1] + 2.0
287                 else:
288                     self.config[interaction_point] = old1 - self.mu * np.
    sign(old1)
289                     self.config[interaction_point] = old2 - self.mu * np.
    sign(old2)
290             else:
291                 if (abs(old1-old2)<1.0):
292                     self.config[interaction_point] = old1 + self.mu * (
    old2 - old1)
293                     self.config[interaction_point + 1] = old2 + self.mu *
     (old1 - old2)
294                 else:
295                     self.config[interaction_point] = old1 + self.mu *
    (2.0 - abs(old2 - old1))*np.sign(old1)
```

```
296                     self.config[interaction_point + 1] = old2 + self.mu *
     (2.0 - abs(old1 - old2))*np.sign(old2)
297                     if (self.config[interaction_point]>1.0):
298                         self.config[interaction_point] = self.config[
    interaction_point] - 2.0
299                     if (self.config[interaction_point + 1]>1.0):
300                         self.config[interaction_point + 1] = self.config[
    interaction_point + 1] - 2.0
301                     if (self.config[interaction_point]<=-1.0):
302                         self.config[interaction_point] = self.config[
    interaction_point] + 2.0
303                     if (self.config[interaction_point + 1]<=-1.0):
304                         self.config[interaction_point + 1] = self.config[
    interaction_point + 1] + 2.0
305             self.config[self.noise_loc[i]] = self.config[self.noise_loc[i
    ]] + self.noise[i]
306             if (self.config[self.noise_loc[i]]>1.0):
307                 self.config[self.noise_loc[i]] = self.config[self.
    noise_loc[i]] - 2.0
308             if (self.config[self.noise_loc[i]]<=-1.0):
309                 self.config[self.noise_loc[i]] = self.config[self.
    noise_loc[i]] + 2.0
310             if ((i+1) % 100_000 == 0):
311                 self.convergence[int((i+1)/100_000)] = np.sum([abs(self.
    config[j]-self.config[j+1]) if(abs(self.config[j]-self.config[j+1])
    <1.0) else 2.0-abs(self.config[j]-self.config[j+1]) for j in range(
    self.N - 1)])
312                 s = np.sum(np.sin(self.config*math.pi))
313                 c = np.sum(np.cos(self.config*math.pi))
314                 self.r[int((i+1)/100_000)] = np.sqrt(s**2 + c**2)/self.N
315                 self.theta[int((i+1)/100_000)] = np.arcsin(s/(self.N*self
    .r[int((i+1)/100_000)]))
316                 if (c < 0):
317                     if (self.theta[int((i+1)/100_000)] >= 0):
318                         self.theta[int((i+1)/100_000)] = math.pi - self.
    theta[int((i+1)/100_000)]
319                     else:
320                         self.theta[int((i+1)/100_000)] = - math.pi - self
    .theta[int((i+1)/100_000)]
321         name = str(self.N) + '_0' + str(self.mu) + '_' + str(self.steps)
    + str(self.epsilon)
322         folder_name = 'ucm_n1_version'+ str(self.i) + '_' + name + '/'
323         os.mkdir(folder_name)
324         np.save(folder_name + 'c_' + name, self.convergence)
325         np.save(folder_name + 'r_' + name, self.r)
326         np.save(folder_name + 'theta_' + name, self.theta)
327         np.save(folder_name + 'final_config_' + name, self.config)
328
329 class ucm:
330     def __init__(self, N, mu, steps, i, initial_configuration,
    interaction_order):
331         self.i = i
332         self.N = N
333         self.mu = mu
```

```
334         self.steps = steps
335         self.config = initial_configuration
336         self.interaction_order = interaction_order
337         self.convergence = np.zeros(int(self.steps/100_000) + 1)
338         self.convergence[0] = np.sum([abs(self.config[i]-self.config[i
    +1]) if(abs(self.config[i]-self.config[i+1])<1.0) else 2.0-abs(self.
    config[i]-self.config[i+1]) for i in range(self.N - 1)])
339         self.r = np.zeros(int(self.steps/100_000) + 1)
340         s = np.sum(np.sin(self.config*math.pi))
341         c = np.sum(np.cos(self.config*math.pi))
342         self.r[0] = np.sqrt(s**2 + c**2)/self.N
343         self.theta = np.zeros(int(self.steps/100_000) + 1)
344         self.theta[0] = np.arcsin(s/(self.N*self.r[0]))
345         if (c < 0):
346             if (self.theta[0] >= 0):
347                 self.theta[0] = math.pi - self.theta[0]
348             else:
349                 self.theta[0] = - math.pi - self.theta[0]
350 
351     def run(self):
352         for i in range(self.steps):
353             interaction_point = self.interaction_order[i]
354             old1 = self.config[interaction_point]
355             old2 = self.config[interaction_point + 1]
356             if (abs(old1-old2) == 1.0):
357                 if (np.random.default_rng().integers(0,1)==0):
358                     self.config[interaction_point] = old1 + self.mu * np.
    sign(old1)
359                     self.config[interaction_point + 1] = old2 + self.mu *
     np.sign(old2)
360                     if (self.config[interaction_point]>1.0):
361                         self.config[interaction_point] = self.config[
    interaction_point] - 2.0
362                     if (self.config[interaction_point + 1]>1.0):
363                         self.config[interaction_point + 1] = self.config[
    interaction_point + 1] - 2.0
364                     if (self.config[interaction_point]<=-1.0):
365                         self.config[interaction_point] = self.config[
    interaction_point] + 2.0
366                     if (self.config[interaction_point + 1]<=-1.0):
367                         self.config[interaction_point + 1] = self.config[
    interaction_point + 1] + 2.0
368                 else:
369                     self.config[interaction_point] = old1 - self.mu * np.
    sign(old1)
370                     self.config[interaction_point] = old2 - self.mu * np.
    sign(old2)
371             else:
372                 if (abs(old1-old2)<1.0):
373                     self.config[interaction_point] = old1 + self.mu * (
    old2 - old1)
374                     self.config[interaction_point + 1] = old2 + self.mu *
     (old1 - old2)
375                 else:
```

```
                    self.config[interaction_point] = old1 + self.mu *
    (2.0 - abs(old2 - old1))*np.sign(old1)
                    self.config[interaction_point + 1] = old2 + self.mu *
     (2.0 - abs(old1 - old2))*np.sign(old2)
                    if (self.config[interaction_point]>1.0):
                        self.config[interaction_point] = self.config[
    interaction_point] - 2.0
                    if (self.config[interaction_point + 1]>1.0):
                        self.config[interaction_point + 1] = self.config[
    interaction_point + 1] - 2.0
                    if (self.config[interaction_point]<=-1.0):
                        self.config[interaction_point] = self.config[
    interaction_point] + 2.0
                    if (self.config[interaction_point + 1]<=-1.0):
                        self.config[interaction_point + 1] = self.config[
    interaction_point + 1] + 2.0
            if ((i+1) % 100_000 == 0):
                self.convergence[int((i+1)/100_000)] = np.sum([abs(self.
    config[j]-self.config[j+1]) if(abs(self.config[j]-self.config[j+1])
    <1.0) else 2.0-abs(self.config[j]-self.config[j+1]) for j in range(
    self.N - 1)])
                s = np.sum(np.sin(self.config*math.pi))
                c = np.sum(np.cos(self.config*math.pi))
                self.r[int((i+1)/100_000)] = np.sqrt(s**2 + c**2)/self.N
                self.theta[int((i+1)/100_000)] = np.arcsin(s/(self.N*self
    .r[int((i+1)/100_000)]))
                if (c < 0):
                    if (self.theta[int((i+1)/100_000)] >= 0):
                        self.theta[int((i+1)/100_000)] = math.pi - self.
    theta[int((i+1)/100_000)]
                    else:
                        self.theta[int((i+1)/100_000)] = - math.pi - self
    .theta[int((i+1)/100_000)]
        name = str(self.N) + '_0' + str(self.mu) + '_' + str(self.steps)
        folder_name = 'ucm_n1_version'+ str(self.i) + '_' + name + '/'
        os.mkdir(folder_name)
        np.save(folder_name + 'c_' + name, self.convergence)
        np.save(folder_name + 'r_' + name, self.r)
        np.save(folder_name + 'theta_' + name, self.theta)
        np.save(folder_name + 'final_config_' + name, self.config)
```

The interaction times and places were generated as shown in the following code.

```
import numpy as np


N = 200
mu = 0.5
steps = 1e8
i = 0

exp = np.random.default_rng().exponential(1.0, int(steps)+N)
```

```
folder_name = 'interactions/version' + str(i)
poisson_clocks = exp[0:N]
interaction_order = np.zeros(int(steps), dtype=np.int_)
interaction_time = np.zeros(int(steps)+1)
argmin = 0
for i in range(int(steps)):
    argmin = np.argmin(poisson_clocks)
    interaction_order[i] = argmin
    interaction_time[i+1] = poisson_clocks[argmin]
    poisson_clocks[argmin] = exp[N+i] + poisson_clocks[argmin]
np.save(folder_name + '/interaction_time', interaction_time)
np.save(folder_name + '/interaction_order', interaction_order)
```

The noise locations and, in the case of bi-modal noise, noise direction, were generated like in the following code.

```
import numpy as np

i = 0
noise_loc = np.random.default_rng().integers(0,200,100_000_000)
noise_dir = np.random.default_rng().integers(0,2, 100_000_000)
np.save('interactions/v' + str(i) + '/noise_loc', noise_loc)
np.save('interactions/v' + str(i) + '/noise_dir', noise_dir)
```

Bibliography

[Bas11] R. F. Bass. *Stochastic Processes.* Number Vol. 33 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2011.

[BGJZ23] S. Baldassarri, A. Gallo, V. Jacquier, and A. Zocca. Ising model on clustered networks: A model for opinion dynamics. *Physica A: Statistical Mechanics and its Applications*, 623:128811, 2023.

[Bir52] Z. W. Birnbaum. Numerical tabulation of the distribution of kolmogorov's statistic for finite sample size. *Journal of the American Statistical Association*, 47(259):425–441, 1952.

[BW21] R. Bhattacharya and E. C. Waymire. *Random Walk, Brownian Motion, and Martingales.* Springer International Publishing, Cham, 1 edition, 2021.

[CFL09] C. Castellano, S. Fortunato, and V. Loreto. Statistical physics of social dynamics. *Rev. Mod. Phys.*, 81:591–646, May 2009.

[CKSe20] C. R. Harris, K. J. Millman, S. J. van der Walt, and et. al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[CP23] W. Chu and M. A. Porter. Non-markovian models of opinion dynamics on temporal networks. *SIAM Journal on Applied Dynamical Systems*, 22(3):2624–2647, 2023.

[CR16] F. Collet and W. M. Ruszel. Synchronization and Spin-Flop Transitions for a Mean-Field XY Model in Random Field. *Journal of Statistical Physics*, 164:645 – 666, 2016.

[CWS+20] H. Chen, S. Wang, C. Shen, H. Zhang, and G. Bianconi. Non-markovian majority-vote model. *Phys. Rev. E*, 102:062311, Dec 2020.

[dax] Alina daxelka. Deffuant model study. `https://github.com/daxelka/Deffuant_model_study/blob/main/Readme.md`.

[DDK24] Y. Dong, Z. Ding, and G. Kou. *Social Network DeGroot Model: Supporting Consensus Reaching in Opinion Dynamics.* Springer Singapore, 1 edition, April 2024.

[DGM02] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. Ising model on networks with an arbitrary distribution of connections. *Phys. Rev. E*, 66:016104, Jul 2002.

[DNAW00] G. Deffuant, D. Neau, F. Amblard, and G. Weisbuch. Mixing beliefs among interacting agents. *Advances in Complex Systems*, 3:87–98, 01 2000.

[Ebe23] A. Eberle. Markov processes. `https://uni-bonn.sciebo.de/s/kzTUFff5FrWGAay#pdfviewer`, April 2023. lecture notes.

[FGSR+14] J. Fernández-Gracia, K. Suchecki, J. J. Ramasco, M. San Miguel, and V. M. Eguíluz. Is the voter model a model for voters? *Phys. Rev. Lett.*, 112:158701, Apr 2014.

[FV17] S. Friedli and Y. Velenik. *Statistical Mechanics of Lattice Systems: A Concrete Mathematical Introduction*. Cambridge University Press, 2017.

[Gal22] J.-F. Le Gall. *Brownian Motion*, pages 349–393. Springer International Publishing, Cham, 2022.

[GHH20] N. Gantert, M. Heydenreich, and T. Hirscher. Strictly weak consensus in the uniform compass model on $\mathbb{Z}$. *Bernoulli*, 26(2):1269 – 1293, 2020.

[GvR16] G.L.M. Groenewegen and A.C.M. van Rooij. *Spaces of Continuous Functions*. Atlantis Studies in Mathematics. Atlantis Press Paris, 1 edition, 2016.

[Häg12] O. Häggström. A pairwise averaging procedure with application to consensus formation in the deffuant model. *Acta Applicandae Mathematicae*, 2012.

[Has22] T. Haslwanter. *An Introduction to Statistics with Python, With Applications in the Life Sciences*. Springer Cham, 2 edition, 2022.

[HRFS+22] H. Hassani, R. Razavi-Far, M. Saif, F. Chiclana, O. Krejcar, and E. Herrera-Viedma. Classical dynamic consensus and opinion dynamics models: A survey of recent trends and methodologies. *Information Fusion*, 88:22–40, 2022.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[Kec95] A. S. Kechris. *Classical Descriptive Set Theory*. Springer New York, New York, NY, 1995.

[KS12] L. B. Koralov and Y. G. Sinai. *Theory of Probability and Random Processes*. Springer Berlin, Heidelberg, 2, corr., 2nd printing edition, 2012.

[Lig85] T. M. Liggett. *Interacting Particle Systems. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer New York, NY, 1985.

[Lig10] T. M. Liggett. *Continuous Time Markov Processes: An Introduction*. Graduate studies in mathematics. American Mathematical Society, 2010.

[LR22] E.L. Lehmann and J. P. Romano. *Testing Statistical Hypotheses*. Springer Cham, 4 edition, 2022.

[MP10] P. Mörters and Y. Peres. *Brownian Motion*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2010.

[mul] `https://docs.python.org/3/library/multiprocessing.html#module-multiprocessing`.

[New18] M. Newman. *Networks*. Oxford University Press, 2 edition, 2018.

[PR82] J. K. Patel and C. B. Read. *Handbook of the Normal Distribution*. Marcel Dekker, Inc., 1982.

[Ric95] J. A. Rice. *Mathematical Statistics and Data Analysis*. Wadsworth Publishing Company, 2 edition, 1995.

[SL11] R. Simard and P. L'Ecuyer. Computing the two-sided kolmogorov-smirnov distribution. *Journal of Statistical Software*, 39(11):1â18, 2011.

[SP12] R. L. Schilling and L. Partzsch. *Brownian Motion : An Introduction to Stochastic Processes*. De Gruyter Graduate. De Gruyter, 2012.

[SSS04] D. Stauffer, A. Sousa, and C. Schulze. Discretized opinion dynmaics of the deffuant model on scale-free networks. *Journal of Artificial Societies and Social Simulation*, 2004.

[TYS21] M. Tiwari, X. Yang, and S. Sen. Modeling the nonlinear effects of opinion kinematics in elections: A simple ising model with random field based study. *Physica A: Statistical Mechanics and its Applications*, 582:126287, 2021.

[Var07] S. R. S. Varadhan. *Stochastic Processes*. Number Volume 16 in Courant Lecture Notes. Courant Institute of Mathematical Sciences, New York, NY, 2007.

[VGOe20] P. Virtanen, R. Gommers, T. E. Oliphant, and et.al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[Wei04] G. Weisbuch. Bounded confidence and social networks. *The European Physical Journal B*, 2004.