Universiteit Utrecht

Exploring the link between glassy dynamics and structure using machine learning

Faculty of Science

Master Thesis

Rinske Alkemade

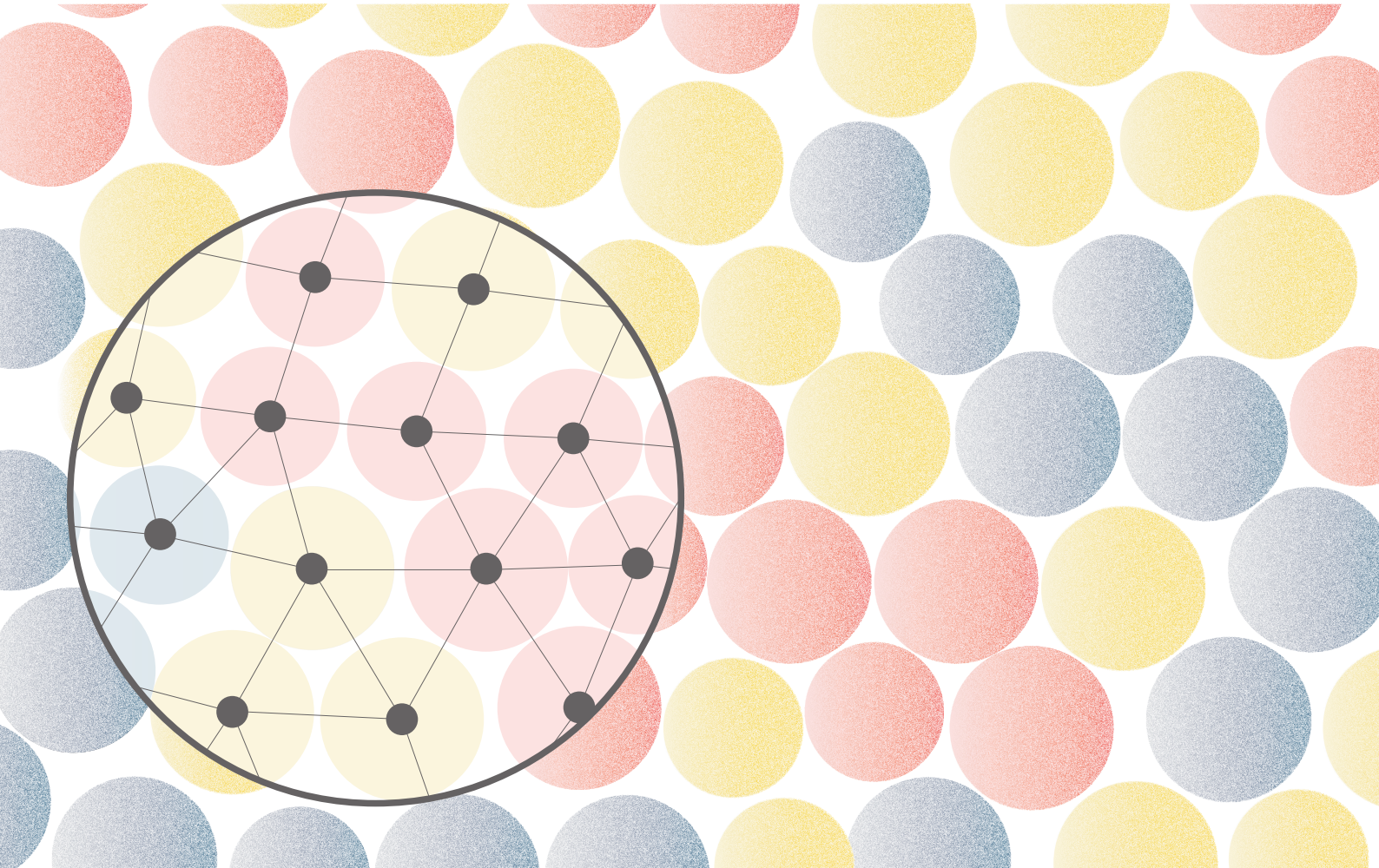
Experimental Physics

Supervisors:

Dr. Frank Smallenburg
CNRS, Université Paris-Saclay
Laboratoire de Physique des Solides

Dr. Laura Filion
Utrecht University
Debye Institute

January 14, 2022



Cover Design: Rinske Alkemade

Abstract

One of the most intriguing features of glassy fluids is their heterogeneous dynamics: the system spontaneously forms areas of fast and slow moving particles. Recent research has shown that local structure can be used as a powerful predictor for future dynamics when combined with machine learning techniques. In this thesis we examine the relation between local structure and the dynamical heterogeneity in a glassy system consisting of spheres of two different sizes. We measure the dynamics of this system using event-driven molecular dynamics simulations, and train machine learning techniques to make predictions about these dynamics.

We begin the thesis by comparing three machine learning techniques, namely linear regression, neural networks and graph neural networks, that are trained to predict the dynamic propensity based on the same set of parameters that capture the local structure of a particle's environment. We conclude that linear regression is the preferred method, since it is fast, robust and not sensitive to overfitting.

Thereafter, we examine several methods to improve the propensity prediction made by linear regression. The most significant improvement is obtained by providing the algorithm with information about the center of the cage that is formed by neighbouring particles and that acts as a temporary trap for each particle. Providing the linear regression algorithm with this cage center information, leads to impressive improvements in the accuracy of the propensity prediction at times associated with the caging regime.

Additionally, we examine the anisotropy in the movements of particles. We find that both collective local drift and preferential directions in cage escape are sources for directionality in the dynamic propensity.

Finally, we examine the dynamics of particles in the context of the slow and fast moving regions. We observe that at time scales close to the relaxation time, fast particles preferentially move parallel to the boundary between fast and slow regions. Intriguingly, at short times we find that the earliest particles to escape their cages, are located at the boundary and not, as expected, within the fast regions.

Contents

PART I

INTRODUCTION AND BACKGROUND

Chapter 1

Introduction

1.1 Introduction to glassy systems

Glassy materials are omnipresent in our daily lives: from a cup holding our tea to the windows in our houses. However, it is not only in these everyday materials that we encounter systems in a glassy state. On many different length scales and in many different materials we see glassy behaviour. Some glassy materials, like ceramics, do somewhat remind us of the brittle material that we colloquially consider to be glass. For other materials, however, it might be less obvious that they are in a glassy state. Think for example of metallic glasses, used among other things for the production of golf sticks, polymer glasses that form plastics and even systems like living cell tissue [1–3]. This brings us to the question of when we can classify a material to be in a glassy state? We will see that this question is not at all easy to answer.

Glassy substances are materials that have the disordered structure of a liquid (i.e. they are not crystals) but show dynamical behaviour similar to that of a solid. In particular, they resist deformation: under the influence of an external force, they can keep their shape for a significant amount of time, although they may flow if we wait long enough. Glassy systems are formed by quenching, i.e. rapidly cooling or compressing, a liquid. Due to this rapid cooling or compressing the system has no time to undergo a phase transition to a more stable crystal structure, and instead ends up in a meta-stable high-density disordered phase. Despite the fact that we encounter glassy materials so frequently, the physics behind glass forming is still not well understood. The main reason for this is the seeming discrepancy between structure and dynamics. While quenching the system, we observe a spectacular decrease in dynamics; the relaxation time (i.e. the time after which a system has 'forgotten' about its initial position, which is a measure for the relevant dynamics involved) can increase by twelve orders of magnitude. At the same time however, little to no structural changes seem to appear while going from a fluid to a glassy phase: at a particle level a glass still looks chaotic and disordered.

This fact that the structure of glass bears large similarities with the structure of a liquid can be demonstrated by, for example, comparing the pair correlation function of the two states. This function quantifies how the average density distribution around a particle in the system varies. In Figure 1.1 we show this correlation function for both a fluid at high temperature and a glassy system at low temperature, as well as intermediate states. As one can see, there is almost no difference in overall shape between the pair correlation function of the glassy state and the fluid.

The discrepancy between structure and dynamics in glassy systems is intriguing in itself; what causes the dynamical arrest if at first glance it cannot easily be explained from structure? Moreover, this discrepancy leads to the fact that the transition from a fluid to a glassy state cannot be described as a thermodynamic phase transition. Normally during a phase transition, dynamical changes are accompanied by structural changes. This means that we can assign a structural order parameter, which undergoes an abrupt change at the phase transition. Think for example of the phase transition in a liquid crystal from an isotropic to a nematic phase. Here the degree of

alignment that particles show with respect to a global average director could serve as an order parameter [4]. In the case of glassy materials we have not (yet[1]) been able to define such a structural order parameter. This means that besides the fact that we cannot explain dynamical arrest in glassy materials from the structure, we also cannot define an exact point at which the system goes from a liquid to a glass.

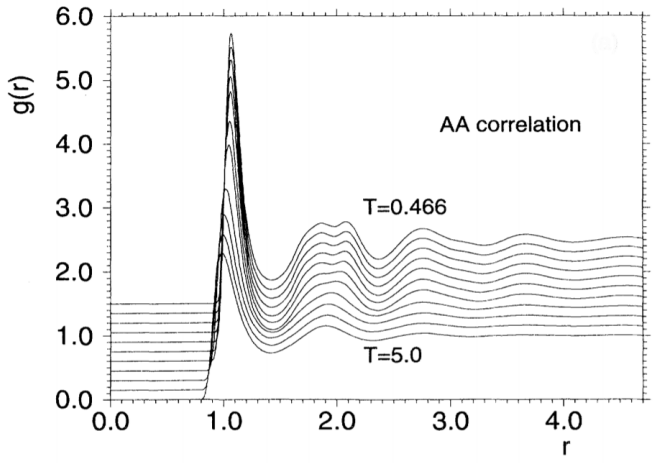


Figure 1.1: Pair correlation function $g(r)$ between large particles in a classic glass-forming mixture known as the Kob-Andersen model [8] for multiple temperatures taken from Ref. [8]. $T = 5.0$ in this system corresponds to a fluid state, while $T = 0.466$ corresponds to a glassy state.

1.1.1 Theoretical approach to the glass transition

An additional complicating factor in determining whether a a glass transition exists is the rapidly increasing relaxation time that is associated with the formation of a glass. Assuming that it takes 10 to 100 relaxation times before a liquid is in equilibrium, for systems deep in the glassy regime equilibrating the system could easily take thousands of years [5]. As a result, close to the glass transition, it is not possible to do experiments on equilibrated glassy systems. This means that systems that we do experiments on will either be glassy fluids or actual glasses that are not equilibrated.

Although doing experiments on systems close to the glass transition is thus very hard, there are multiple theories, based on extrapolating existing data, that try to explain what happens upon quenching the system further and further. One scenario proposed by Ref. [9] states that at some finite temperature, the glass would no longer be the entropically favoured state and instead[2] a crystal configuration is preferred [10]. This theory would lead to a thermodynamic phase transition from a glassy fluid to a crystal, meaning that a true glass state is never reached. An alternative scenario is that at one point the glassy system will arrest completely and we do in fact encounter a real glass. Whether or not at this point a real thermodynamic phase transition from a glassy fluid to a glass state takes place, is an open question. If at one point the relaxation time diverges, there is in fact a real distinction between the fluid and the glass. It could however also be that the relaxation time just keeps on growing, without ever reaching a divergence.

[1] The fact that we say 'yet', is because there is still research going on that tries to find 'hidden' order parameters [5–7].

[2] The precise explanation of what happens when the entropy of a glass falls below that of a crystal is quite complicated and has to do with configurational entropy. We will omit the discussion here, but for more information the reader is referred to [10].

Due to the fact that doing experiments on equilibrated systems near the glass transition is almost impossible, the discussion above is almost philosophical. As a practical definition, people therefore often consider a glassy fluid to be a glass when the relaxation time of the system exceeds some pre-defined threshold. For atomic or molecular glasses, this is typically 100 seconds. During sufficiently short experiments on these systems, the system does not lose memory of its initial state, meaning that it can be regarded as a more or less stable glass. We can however not stress enough that the glass definition above is a more or less arbitrary criterion: one could alternatively have taken a relaxation time of at least 10 or 1000 seconds [5].

1.2 Theories on glass formation

Many researchers have tried to build theories that can explain the tremendous slow down of dynamics that takes place during glass formation. As David A. Weitz, a professor at Harvard once joked: "There are more theories of the glass transition than there are theorists who propose them" [11]. The main quest is to find some theoretically plausible link between structure and dynamics. The fact that at first glance the structures of fluids and glasses look the same, does not mean that there are no structural differences. In this thesis we will not discuss all the existing theories, but note that there are many review articles that give an overview of the theoretical possibilities, see for example Refs. [2] and [5]. For now, we will limit ourselves to only briefly mentioning the most important theories.

One of the earliest attempts of constructing a glass theory was carried out by F.C. Frank in 1952. His suspicion was that the arrest of the dynamics was linked to the packing structure of particles [12]. In order to investigate this, he looked at how the nearest neighbours of a particle theoretically could be placed. In a typical mono-disperse liquid, each particle can have up to about 12 neighbours, and therefore he looked at the different arrangements in which 12 spheres can be ordered around a reference sphere while in contact with that sphere. Arrangements were considered to be different when it was impossible to go from one configuration to the other without one of the twelve spheres losing contact. It turned out that three different structures were possible: hexagonal close packed (HCP), face- centered cubic (FCC) and icosahedral. This last structure is associated with a five-fold symmetry and since this symmetry is not compatible with long-range periodicity, and thus with crystallization, Frank argued that the non-periodic glassy systems would probably contain many of these icosahedral structures. For a long time, Frank's theory took a central role in the search for understanding glassy materials. More recently however, it has become clear that although in some glassy systems icosahedral structures are indeed found, there are also glassy systems that show little to no icosahedral structures [13].

Although Frank's theory does not provide a complete theory of glasses, it opened the way to many more theories that tried to explain the relation between structure and dynamics [14]. An important class of these theories are mean-field theories. Two examples of these are mode coupling theory (MCT) [15, 16] and random first order theory [17]. Both extend liquid state theory to supercooled liquids. A limitation associated with this class of theories is that, since they are mean-field theories, they cannot incorporate local fluctuations. However, since it has been shown that at sufficiently high temperatures these local fluctuations do not play a large role, at these temperatures MCT gives qualitatively good results.

Another theory is the Adam Gibbs theory [18], which assumes that the quenched liquids form cooperatively re-arranging regions whose size increase upon quenching. These regions allow for rearrangements between different internal configurations of the enclosed particles, at the cost of crossing a certain free-energy barrier. By assuming that each of these rearrangements can switch between a small number of favoured states, the theory then provides predictions for the global relaxation time.

A different approach to the glass problem is found in energy landscape theory [9]. This theory assumes that the evolution of the system can be described completely by the associated potential

energy landscape. The movement of particles would then be associated with the system relaxing towards a local minimum combined with thermal excitation.

All these theories take a very different approach to the glass transition, however as of yet there is no single theory that satisfyingly explains all aspects of glass forming. For example, we do not have a clear method for predicting the behaviour of the relaxation time of a given system as it approaches the glass state.

1.2.1 Explaining heterogeneous dynamics

The arrested dynamics that we see in glasses is accompanied by other interesting phenomena. One of them, called dynamical heterogeneity, will be the main topic of this thesis [19, 20]. Dynamical heterogeneity describes the phenomenon that upon quenching the system, the dynamics do not decrease homogeneously for all particles. Instead growing domains of either fast or slow moving particles occur. In Figure 1.2 we show a snapshot of a typical glass configuration where particles are colored according to their averaged distance traveled with respect to some initial configuration. Red and yellow particles have moved the furthest out, while white and blue particles have moved the least. In this figure, we can clearly distinguish domains of different mobility. Note that particles with a higher mobility, do not have a higher velocity necessarily. Instead, mobility is associated with the ease with which different parts of the glass rearrange themselves, i.e. escape out of the cage formed by their nearest neighbours. What we call the fast moving areas, are areas associated with a high rearrangement probability. Dynamical heterogeneity also implies that different areas of the glass have different relaxation times [21].

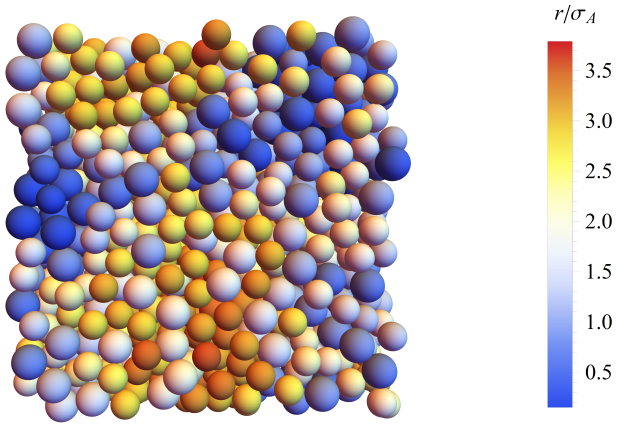


Figure 1.2: Dynamical heterogeneity of a binary hard-sphere system of 2000 particles with size ratio $\sigma_B/\sigma_A = 0.85$, composition $x_A = N_A/N = 0.3$ and packing fraction $\phi = 0.58$. The colours of the particles indicate the distance they have traveled after approximately one relaxation time. One can clearly distinguish regions of fast and slow moving particles.

Several theories have tried to find a relationship between structure and dynamical heterogeneity. One of them is dynamic facilitation [22]. As we mentioned before, the arrested dynamics that we see in glassy systems is associated with 'caging'. The more the system is quenched, the harder it is for particles to redistribute. However, if at one point a particle escapes its cage and obtains a new position, it leaves room for other particles to escape their cage too. This effect could lead to whole regions of fast and slow rearranging particles, and thus to heterogeneity.

Other approaches try to make a connection between soft modes and heterogeneity [23]. For each particle one can construct the Hessian matrix, i.e. the matrix that contains the second derivative of the potential energy with respect to the coordinates of particles. The associated eigenvalues of

this matrix are what we call the eigenfrequencies of the particles. The lower the frequency, the 'softer' the particle is. As it turns out, there is a connection between the softness of a particle and the probability of losing neighbours in the future [23]. This probability is related to an easy rearrangement of particles and thus to the mobility of the particles.

Finally, there is a subset of theories that try to capture the local environment of particles in terms of structural descriptors, and to couple these to dynamical heterogeneity [14]. This means one tries to find a link between the structure of an initial configuration and its dynamical heterogeneity over time (see e.g. Refs. [24–29]). Commonly used structural parameters are parameters based on the local density and parameters that describe the local structure in terms of spherical harmonics [30]. It has been shown that the local structure correlates with the dynamical heterogeneity in a variety of systems, including e.g. hard and soft spheres [31–33] and granular systems [34].

Recent advances in machine learning have opened the door to a new way of probing this correlation between structural parameters and heterogeneity [25, 35, 36]. It is this approach of looking at dynamical heterogeneity that we will further explore in this thesis.

1.2.2 Supervised machine learning

The underlying idea to using machine learning is that, although we as humans cannot visually or intuitively distinguish what structural changes invoke dynamical heterogeneity, we might be able to train a computer to do so. In Chapter 4 we will give an elaborate discussion of supervised machine learning, but for now it suffices to say that the goal of this approach is to build algorithms that, given certain relevant input parameters, can be trained to classify and/or predict certain interesting quantities. One of the simplest examples is the algorithm that classifies pictures of two objects, e.g. whether a picture contains the image of a cat or a dog. In that specific case the input parameters are the pixels of the images, and there is a single output parameter, which can either be *cat* or *dog*.

In the case of glassy materials one intriguing approach is to train an algorithm to predict the dynamical heterogeneity, based solely on structural order parameters. This means that one needs to define one or more parameters that reflect the dynamical heterogeneity. The first research that demonstrated the success of this approach, was performed by Ref. [37]. In the paper, the researchers show that one can use support vector machine algorithms to predict the rearrangement probability of particles.

To obtain data on glassy systems that can serve as training data for the machine learning algorithms, one common strategy is the use of computer simulations of glassy systems (see Chapter 3), in which we measure quantities that reflect the dynamic heterogeneity, such as the dynamic propensity [38]. This propensity is often defined as the average absolute distance traveled by a particle over a given time interval [14, 27]. It is measured by tracking the trajectories of particles over multiple runs, each time starting from the same initial configuration, but with different initial velocities. By averaging over all these runs, one obtains the dynamic propensity and thus a measure for the heterogeneity of the system. In Chapter 2 we will go into the propensity in more detail.

Arguably, the most accurate method to date for predicting the dynamic propensity of a glassy system, was introduced in 2020 by Bapst *et al*, who showed that it can be predicted using graph neural networks (GNNs) (see Chapter 4 for more information on these techniques). For their prediction Bapst *et al* only used minimal structural information which consisted of the distances between neighboring particles and their sizes. In 2021, Boattini *et al* showed that by choosing intelligent structural order parameters one can even train a very simple algorithm like linear regression to predict the propensity with nearly the same accuracy as the much more complex graph neural network [35].

1.2.3 Critique on Machine Learning

Although the success of machine learning indeed proves that there is an observable relationship between structure and heterogeneity, it also raises various new questions and issues. A commonly raised critique is that machine learning is like a black box: it does not tell you how it trains itself. Solely the fact that there is a relationship between structure and heterogeneity does not explain why that relationship is there, or how it could be described.

Another critique, specific to the use of machine learning to predict dynamic propensity, is that it is not certain how much of the actual dynamics actually is encoded in parameters like e.g. the dynamic propensity. Since we measure the propensity by averaging over multiple trajectories, the propensity in itself tells you nothing about the precise paths that particles have taken. As Berthier *et al* argue in Ref. [39], "dynamics of individual particles cannot be predicted on the basis of the propensity". The vast majority of fluctuations in particle paths have a statistical origin (due to the size and the direction of the initial velocity), which means that only a small percentage of the fluctuations will be captured by the dynamic propensity.

All the arguments above do not mean however that machine learning cannot be used to better understand the relationship between structure and dynamics. Although machine learning can indeed not be used to predict precise particle trajectories based on structural information only, parameters like the dynamic propensity can give an impression of how dynamic certain areas in the glass are. Furthermore, in order to understand the relationship between structure and dynamics, it is very useful, even essential, to find out what aspects of the structural information are (or are not) required to make dynamical predictions; here, machine learning can provide us with answers.

1.3 Aim of this thesis

In this thesis we will therefore try to obtain more insight into the relation between structural and dynamical heterogeneity making use of machine learning techniques. We start in Chapter 2 with an elaborate discussion on dynamic and structural heterogeneity. In this chapter we also define the parameters that can be used to capture the both the structure and the dynamical evolution of the system. The glassy system that we consider is a binary hard-sphere system at packing fraction $\eta = 0.58$. To collect data on this system we use various computer simulation techniques that are discussed in Chapter 3. Subsequently, in Chapter 4 we discuss the working of three different machine learning techniques (linear regression, neural networks and graph neural networks) that we use to predict glassy dynamics.

The results chapters in this thesis can be divided into roughly three parts. In Chapter 5 we discuss how the three machine learning methods mentioned above compare to each other in predicting the dynamic propensity in glassy systems based on local structure. Since past studies that considered these strategies were based on different sets of input parameters and on different glassy systems [35, 36], no fair comparison has been made yet. From the results, we conclude that linear regression is the preferred method to predict the propensity.

In the second part of the results, in Chapters 6 and 7 we continue with linear regression and try several methods to improve the propensity prediction. We discuss these new methods and analyze their performance.

Finally, as a follow-up to the obtained improvements, in Chapter 8 we discuss how important directional information is when one wants to capture the dynamics of the system. In Chapter 9, we share some preliminary results on the dynamic propensity of particles in relation to their position within the slower and faster rearranging regions.

In Chapter 9 we summarize all findings and give an outlook. The thesis is ended with a Layman's summary.

Chapter 2

Dynamical and structural heterogeneity

The first aim of this thesis is to determine which of the three machine learning algorithms (linear regression, neural networks and graph neural networks) can predict the dynamical heterogeneity of a glassy system based on its structural heterogeneity. In order to do so, we need local structural and dynamical parameters that capture both these aspects. In this chapter we will discuss which parameters we use, as well as give some of their physical background.

2.1 Dynamical heterogeneity

As we mentioned in Chapter 1, the relaxation of particles in a glassy system is not spatially uniform but instead leads to regions of different mobility. These regions can be observed by, for example, looking at the absolute displacement of particles over time. In Figure 2.1 we show this absolute displacement for particles in a glassy Lennard-Jones system. In the figure we see that for long periods of time particles move around some equilibrium position. In between, there are rapid jumps, meaning that the particle suddenly moves away from its initial position finding a new position somewhere else. This dynamical behaviour is associated with so called 'caged particles': at high densities, particles cannot easily advance past their first shell of neighbours. As a result, they move around in a cage formed by neighbouring particles for extended periods of time. Sometimes the fluctuations of these neighbouring particles will allow a particle to move out of its cage, resulting in a jump. After the jump, a particle will start fluctuating around a new position associated with a new cage.

The large variation in the time it takes different particles to escape their cage suggests that the dynamics is heterogeneous. This in itself is not something unforeseen: in almost all thermodynamic systems we see dynamical fluctuations around some average value. However typically such fluctuations follow a Gaussian distribution. In a glassy system the situation is different: instead of the distribution of fast and slow particles following a Gaussian distribution, one can observe a long tail of fast moving particles [40]. As a result, the escape time of particles can differ by orders of magnitude. The fact that we see such a long tail of fast moving particles means that we can make an actual distinction between collections of fast and slow moving particles[1].

The fact that the dynamics of particles in a glassy system are heterogeneous, can also be observed by looking at the relationship between the self-diffusion D_s, which is a measure for the mean displacement of particles, and the viscosity η. Normally these quantities are related by the Stokes-Einstein relation which states that $D_s\eta/T = c$, with T the temperature and c a constant that is independent of T and only depends on the size of the particles. For glassy systems it

[1] Note that although we speak about fast and slow moving particles, this could be misleading as we are not referring to instantaneous speed. If a particle is slow, we do not mean that its velocity is low, but instead that it is caged for a long time.

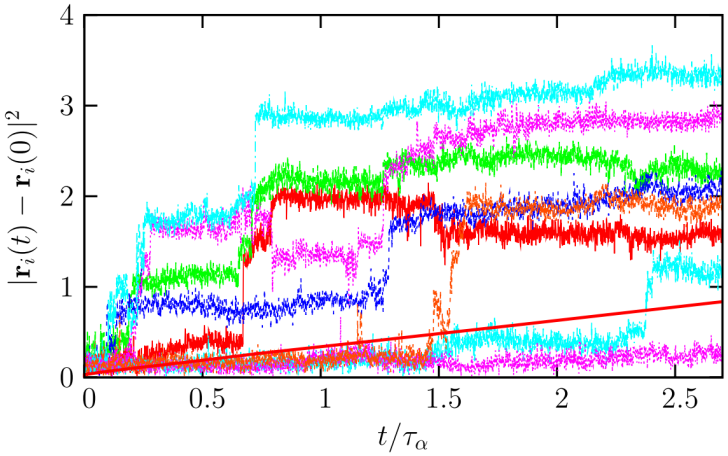


Figure 2.1: Squared displacement over time from individual particles in a system of Lennard-Jones particles in a glassy state as measured in a simulation. The red straight line represents the average squared displacement. Figure from Ref. [2].

has been shown that upon cooling down the system, the Stokes-Einstein relation breaks down: the value of $D_s\eta/T$ can increase by a factor of 2 or 3 [41]. To understand why the breakdown of the Stokes-Einstein relation leads to the conclusion that there is dynamical heterogeneity, we need to think about what aspects of the dynamical behaviour are captured by the viscosity and the self-diffusion. Both quantities are in essence measures of the relaxation time, however they probe different aspects of it. Viscosity is focused more on structural relaxation, while the diffusion coefficient concentrates on the mobility of particles. In order for the Stokes-Einstein relation to hold, an increase in viscosity must be accompanied by a decrease in self-diffusion. If a system shows a high dynamical heterogeneity, it is implied that there will be areas with particles that hardly move at all. As a result, the total structure will relax relatively slowly, resulting in a high viscosity. However, since the self-diffusion coefficient is dominated by fast moving particles, this same heterogeneous system can have a relatively high self-diffusion constant due to the more mobile areas. This means that in such a dynamically heterogeneous system a high viscosity can be accompanied by a relatively high diffusion coefficient, which in turn leads to the breakdown of the Stokes-Einstein relation.

Although the phenomena described above are a clear expression of dynamical heterogeneity, they do not tell us whether particles of high and low mobility are clustered together in the system or distributed randomly. In order to find an answer to that question Vidal *et al* probed a glassy system with an atomic force microscope [42]. This revealed that clusters of particles with similar dynamics form in the system. Moreover, they found that over time regions can change from being slow to being fast and vice versa [42]. The characterization of the sizes of these mobile clusters has been the topic of many studies that often use multi-point correlation functions [2]. One of the interesting findings is that upon cooling the system, the size of the clusters increases. This is somewhat reminiscent of the increasing correlation lengths that are often associated with critical phenomena in a variety of complex systems [2, 5].

2.1.1 Quantifying dynamical heterogeneity

In this thesis we are interested in the link between structure and dynamical heterogeneity. The question is: can we can predict whether a particle will be in a slow or fast moving cluster by only using structural information? In order to do so, we need to capture the dynamical heterogeneity in terms of a local parameter, i.e. we need a dynamic quantity that reflects the mobility of individual particles. One of the most common parameters that is used to this end is the dynamic propensity, which has already been used in machine learning studies [35, 36].

The dynamic propensity is a quantity closely related to the mean squared displacement, which captures how far particles on average move over time [43]. To obtain it, the evolution of a glassy system is measured multiple times, each time starting from the same initial configuration, while assigning each particle a random velocity drawn from a Maxwell-Boltzmann distribution at the desired temperature. This ensemble is called the isoconfigurational ensemble. To obtain the propensity $\Delta r_i(t)$ of particle i, we average the absolute distance it traveled over the time interval t over all trajectories

$$\Delta r_i(t) = \frac{1}{n_{\text{runs}}} \sum_{\text{runs}} |\mathbf{r}_i(t) - \mathbf{r}_i(0)|, \tag{2.1}$$

where n_{runs} is the number of runs. Since measuring the dynamic propensity requires multiple trajectories starting from the same initial position, it is a quantity that can only be measured in simulations. The fact that we average over multiple trajectories also implies that the dynamic propensity does not hold any information about the exact trajectories of individual particles. However, since it reflects which particles are on average fast or slow, it still gives a good impression of the dynamical heterogeneity in the system.

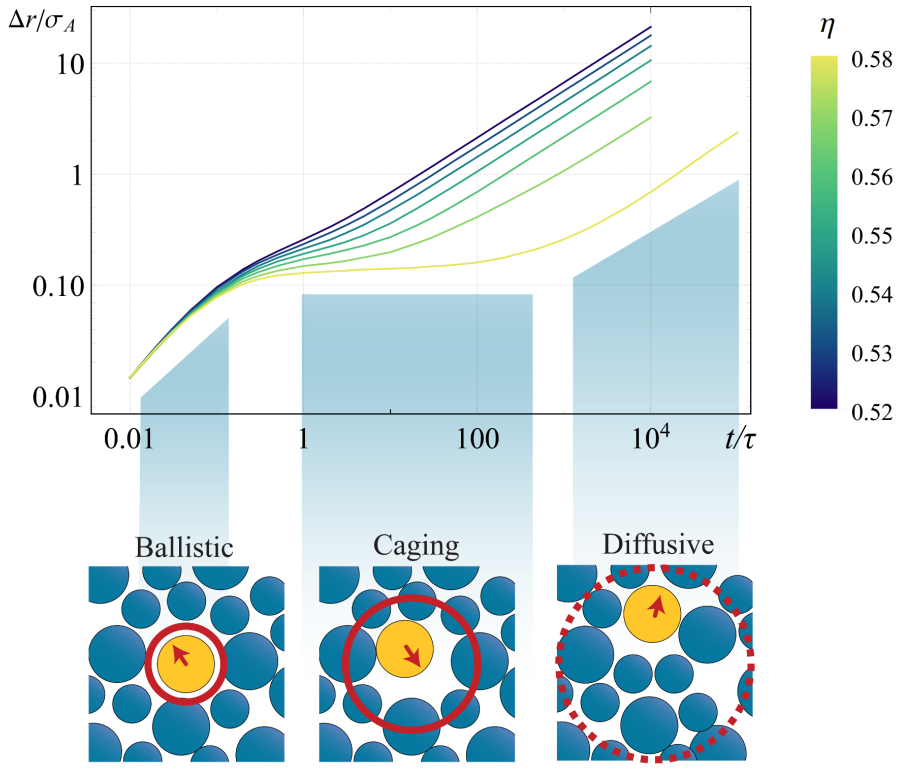


Figure 2.2: Average propensity measurement for the largest particles of a binary hard-sphere system of 2000 particles at various packing fractions η. The ratio of the diameters of the two particles types, A and B is given by $\sigma_B/\sigma_A = 0.85$ and the composition is given by $N_A/(N_A + N_B) = 0.3$. For the highest packing fraction, we also indicate the different dynamic regimes that we can observe: the ballistic, caging and diffusive regime.

To obtain some insight into what the dynamic propensity looks like over time in a glassy system, we show in Figure 2.2 the dynamic propensity averaged over all large particles in a binary

hard-sphere system for various packing fractions in the glassy regime. The dynamical behaviour that is captured in this figure is – not surprisingly – very reminiscent of the behaviour that was captured in Figure 2.1. However, since we now average over many particles, we get an impression of the general behaviour of particles. In this figure we see that the higher the packing fraction, the clearer we can distinguish three different domains. We indicate them as a 'ballistic', a 'caging' and a 'diffusive' regime. At very small times (i.e. in the ballistic regime) most particles do not yet feel any influence of their neighbours. This means that they essentially move as free particles, resulting in a propensity that scales like $\Delta r \propto t$. After some time, particles will start to collide with each other resulting in the caging behaviour we discussed before. At sufficiently high packing fraction, this caging results in an approximately constant averaged propensity, also called the glass plateau [2]. After particles start to escape their cage, we observe an average increase of propensity over time. For high densities, the average propensity in this regime scales as $\Delta r \propto \sqrt{t}$, reflecting that the system is in a diffusive state. Although Figure 2.2 shows the average dynamic propensity for a binary system of hard spheres, the behaviour is very general and is seen in all glassy systems.

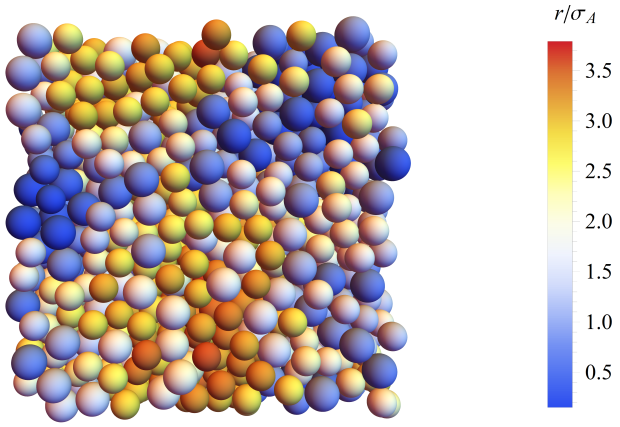


Figure 2.3: Picture of the dynamical heterogeneity of a binary hard sphere system of 2000 particles with diameters $\sigma_A = 1.0$ and $\sigma_B = 0.85$ and number ratio $N_A = 0.3N$ and packing fraction $\phi = 0.58$. The colours of the particles indicate the distance they have traveled at $t/\tau = 10^4$. One can clearly distinguish regions of fast and slow moving particles.

A proof that the dynamic propensity indeed captures the heterogeneity in dynamics can be seen in Figure 2.3, a figure that was already shown in Chapter 1. The colours of the particles reflect the dynamic propensity of individual particles for a binary hard-sphere system at a high packing fraction ($\eta = 0.58$) at a time close to the relaxation time of the system: blue particles are associated with a low propensity, while orange/red particles are associated with a high propensity. In the figure we can clearly see the clustering of fast and slow moving particles.

2.1.2 Directional propensity

In later chapters of this thesis we will not only look at the absolute distance travelled by particles, but also at the direction they travel. We are especially interested in whether in different trajectories particles move in the same direction. We obtain information about this by introducing new dynamical descriptors which we call the dynamic propensity vector and the dynamical self-alignment parameter.

Dynamic propensity vector

The dynamic propensity vector is very similar to the normal propensity; instead of measuring the absolute distance travelled, we now measure the three-dimensional displacement $\mathbf{r}_i(t)$ for each

particle. The dynamic propensity vector, $\Delta\mathbf{r}_i(t)$ is then obtained by averaging these coordinates over all the different trajectories, i.e.

$$\Delta\mathbf{r}_i(t) = \frac{1}{n_{\text{runs}}} \sum_{\text{runs}} (\mathbf{r}_i(t) - \mathbf{r}_i(0)). \tag{2.2}$$

Dynamical self-alignment parameter

The dynamical self-alignment parameter (DSP) reflects to what extent in different trajectories particles move in the same direction. To obtain it, we first calculate the normalized dynamic propensity vector in each direction, i.e. $\mathbf{n}_i(t) = \mathbf{r}_i(t)/|\mathbf{r}_i(t)|$, Afterwards we can then obtain the DSP by calculating

$$\Delta|\bar{n}_i| = \frac{1}{n_{\text{runs}}} \left| \sum_{\text{runs}} \mathbf{n}_i(t) \right| \tag{2.3}$$

where $\Delta|\bar{n}_i|$ is the dynamical self-alignment parameter of particle i. By averaging these vectors over the different trajectories, we have an order parameter that reflects how much a particles displacement aligns between different trajectories; the closer the DSP is to one, the more strongly the motion of the particle is biased towards the same direction in the isoconfigurational ensemble. If, on the other hand, the particles in different trajectories move in completely random directions, we expect the DSP value to converge to 0 when we include more and more trajectories.

2.2 Parameters to capture local structure

In the second part of this chapter we will shift our attention to quantifying structural heterogeneity, by considering order parameters that capture the local structure around each particle. There are many different parameters that one could in principle use to obtain this goal. We chose to use the parameters used in Ref. [35], which consist of a combination of radial densities and angular functions computed in different shells around a particle. Below we will discuss both sets of parameters.

Radial functions

The radial functions essentially measure the particle density inside a spherical shell of thickness 2δ at distance r with respect to a reference particle. The functions are defined as

$$G_i^{(0)}(r, \delta, s) = \sum_{j \neq i, s_j = s} \exp - \frac{(r_{ij} - r)^2}{2\delta^2}. \tag{2.4}$$

Here i is the reference particle, s is the particle type and r_{ij} is the distance between particles i and j. The summation is carried out over all particles with particle type $s_j = s$, which means that the radial density that is measured is type-specific.

Angular functions

The angular descriptors that we use are based on bond order parameters [30, 44]. These bond order parameters express the local environment in terms of spherical harmonics. The spherical harmonics can be viewed as basis functions defined on the surface of a sphere. We project the arrangement of neighbors around our particle of interest onto the set of these weight functions to obtain an expansion of the particle environment in terms of spherical harmonics. To obtain the angular descriptors we first calculate the complex coefficients

$$q_i^{(0)}(l, m, r, \delta) = \frac{1}{Z} \sum_{i \neq j} e^{-\frac{(r_{ij} - r)^2}{2\delta^2}} Y_l^m(\mathbf{r}_{ij}). \tag{2.5}$$

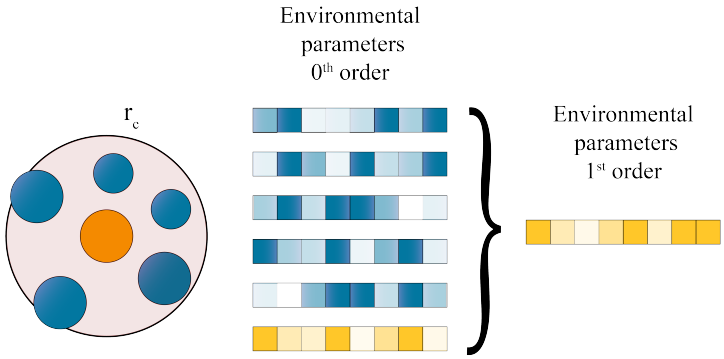


Figure 2.4: In order to obtain the higher-order structural parameters for each particle we average over the parameters of all particles within a certain cutoff distance r_{cut}. In this case the reference particle and parameters are indicated as orange, while the neighbouring particles and their parameters are indicated as blue. The output is the first generation order parameters of the orange reference particle.

Here $Y_l^m(\mathbf{r}_{ij})$ is the spherical harmonic of order l, with m an integer that runs from $-l$ to l, and Z is a normalization factor given by $Z = \sum_{i \neq j} e^{-\frac{(r_{ij}-r)^2}{2\delta^2}}$. Note that although the summation runs over all particles, again the exponent makes sure that mainly particles within a spherical shell at distance r and thickness 2δ will contribute to $q_i^{(0)}(l, m, r, \delta)$.

Finally we sum over m to obtain the rotationally invariant angular descriptors

$$q_i^{(0)}(l, r, \delta) = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{m=l} |q_i^{(0)}(l, m, r, \delta)|^2}. \tag{2.6}$$

Due to the symmetries of the spherical harmonic functions, $q^{(0)}(l, r, \delta)$ for a certain l is expected to detect l-fold symmetry in the environment at the chosen distance r. For example, a high value of $q_l^{(0)}$ at $l = 6$ would indicate a high degree of hexagonal symmetry in the environment [44].

In order to get a good set of structural descriptors, we know from Ref. [35] that we should consider shells at various distances r. In Chapter 5 we will discuss in more detail which shells we include in our description of the environment.

Boattini *et al* showed in Ref. [35], that the propensity prediction of a particle improves significantly, when the prediction is based not only on the structural parameters associated with the particle itself, but also on averaged structural information of neighbouring particles. They included this information by recursively constructing so called 'higher-order averaged bond order parameters', which are are defined as

$$x_i^{(n)} = \frac{1}{C} \sum_{j:r_{ij}<r_c} e^{-r_{ij}/r_c} x_j^{(n-1)}, \tag{2.7}$$

where $x_i^{(n)}$ represents the n^{th} generation of parameter x_i, and where the sum runs over all particles within a cutoff distance r_c. In this equation n is the order and $C = \sum_{j:r_{ij}<r_c} e^{-r_{ij}/r_c}$ is the normalisation. In Figure 2.4 we graphically show how these higher orders are calculated. The cutoff value r_c is chosen to be $r_c/\sigma_A = 2.1$. However, as shown in Ref. [35], the exact value does not have a significant influence.

With the dynamic propensity and the structural parameters, we now have parameters that can capture both the structure and the dynamical heterogeneity of the system. In the next chapter we

will discuss the various machine learning techniques that we will use to fit the dynamic propensity based on these order parameters.

PART II

METHODS

Chapter 3

Simulation methods

In this Chapter we discuss the various simulation techniques that we use in this thesis. We begin with a brief discussion of the benefits of computer simulations for studying glassy systems. Thereafter, we introduce the model that we use to simulate glass in this thesis, i.e. the binary hard-sphere model. Subsequently, in Sections 3.3, 3.4 and 3.5 we discuss three different simulation techniques that can be used to model the binary hard-sphere model, namely time driven molecular dynamics, event driven molecular dynamics, and Monte Carlo simulations. We end the chapter with a discussion of the system specifics.

3.1 Using computer simulations versus experiments

In order to gain insights into the physics behind dynamical heterogeneity, one first needs data that reflects the dynamics of glassy systems. Over the past decades, real-life experiments on both colloidal and molecular systems have been instrumental for this (see e.g. [45–49]). Although many of the early theories on glasses are based purely on experimental data, experimental setups suffer from difficulties in accurately determining the local structure of the system. Even in colloidal systems, where particles are relatively large, tracking particles, especially in 3D, comes with inherent errors including a lack of accurate knowledge of the size of individual particles, as well as noise in the particle positions. Moreover, due to the large relaxation times of glassy systems, experiments can take very long.

Over the past few decades, computer simulations have become increasingly popular for modeling glasses [50]. In the past glassy simulations were strongly limited by computer power, meaning that the deeply super cooled liquid regimes could not be explored. However, due to the rapid improvement of computers and advances in simulation techniques, we are gradually able to reach lower and lower temperatures [51–54]. Although one can argue whether the model that is used in simulations lacks some relevant aspects of reality (after all a model is always a simplification of reality), the advantage of simulations over experiments is that provide perfect access to the real-space structure of the system. Therefore, simulation data can be extremely helpful for obtaining insight into the dynamics of glasses.

3.2 Binary hard-sphere model

In order to study glassy systems, a number of simulation models have been developed. Many of these are based on binary or polydisperse mixtures of particles, that can be tuned to avoid crystallization. These include e.g. binary hard-sphere models [55], the binary Kob-Andersen Lennard-Jones mixture [8] and the binary Wahnström Lennard-Jones mixture [56]. In this thesis we will use the simplest of the three models mentioned above, namely the binary hard-sphere model. In the past, systems of binary hard spheres have extensively been used in the studies of

glassy dynamics [57–60].

The binary hard-sphere system contains particles of two difference sizes, with diameters σ_A and σ_B, that cannot overlap, but otherwise do not interact. The potential between particles i and j with diameters σ_i and σ_j is given by

$$\phi(r_{ij}) = \begin{cases} \infty & \text{if } r_{ij} \leq \sigma_{ij} \\ 0 & \text{if } r_{ij} > \sigma_{ij}, \end{cases} \tag{3.1}$$

where r_{ij} is the center-to-center distance between particles i and j and $\sigma_{ij} = (\sigma_i + \sigma_j)/2$.

The binary hard-sphere system that we consider throughout this thesis consists of two types of particles with a size ratio of $\sigma_B/\sigma_A = 0.85$. We simulate systems of 2000 particles in total, where the composition ratio is set to $N_A/N = 0.3$. Since we know that at a packing fraction of around $\eta = 0.58$ the structure of the system correlates quite strongly with the dynamics [24, 61], we focus on this packing fraction. The simulation box is set up with periodic boundaries with nearest image convention [62].

In the following sections we will discuss various simulation techniques that can be used to simulate hard-sphere mixtures. Some, such as event driven molecular dynamics are used to obtain information about the dynamics, while others, such as Monte Carlo simulations, are aimed at measuring ensemble averages.

3.3 Time-driven molecular dynamics

Molecular dynamics (MD) is a simulation technique that obtains the system's evolution over time by based on Newton's equations of motion [62]. For systems with a continuous interaction potential, this can be done via numerical integration. In particular, assuming that we know the positions and velocities at time t, and given some fixed small time step Δt, the positions and velocities at time $t + \Delta t$ can be determined using a numerical integration scheme.

Although such time-driven MD simulations are quite easy to implement, they can have disadvantages. When for example the typical interaction time between particles is short compared to the time between interactions (which would be the case for very short-ranged interaction potentials), MD simulations can be more time consuming than necessary since a large amount of the simulation time is used to simulate the rather trivial particle trajectories between interactions. The second and most important downside of time-driven MD simulations is that they do not allow for discontinuous potentials. The force that each particle experiences, i.e. the gradient of the overall potential, is not well-defined in the case of a discontinuous potential. In a system of hard spheres, time-driven simulations will unavoidably lead to overlap; since the forces at time step t are used to obtain the positions at a some later time $t + \delta t$, particles that are about to collide at time t, do not yet feel a repulsive force and will thus overlap at time $t + \delta t$.

Although this means that we cannot use time-driven MD simulations to model the normal dynamical behaviour of binary hard spheres, we will still use MD simulations in Chapter 6, to find the so-called inherent state of the system (i.e. bring the system to a local potential energy minimum). For this, we temporarily replace the hard-sphere interaction by an effective repulsive potential. In particular, it has been shown that hard spheres close to jamming can be effectively modelled using a repulsive potential given by [63]

$$V(h) = -k_B T \log(h). \tag{3.2}$$

where h is the surface-to-surface distance between particles given by $h = |\mathbf{r}_{ij}| - \sigma_{ij}$, where k_B is the Boltzmann constant and T is the temperature. Because this potential is continuous in h, we can use time-driven MD simulations to quench the system.

The MD code used in this thesis was written from scratch in C by the author. In this code we implement the time-driven MD simulations using the Verlet algorithm [64]. This algorithm assumes that, given the position $r(t)$ and force $f(t)$ at time t, the position and force at time $t+\Delta t$ are given by

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{\mathbf{f}(t)}{2m}(\Delta t)^2, \tag{3.3}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{\mathbf{f}(t+\Delta t) + \mathbf{f}(t)}{2m}\Delta t. \tag{3.4}$$

Note that for the velocity update at time $t+\Delta t$ this integration scheme requires not only the force at time t but also at time $t+\Delta t$. Therefore the velocity is updated in two steps: first we update the position and the part of the velocity that contains the force at time t. Given the new positions, we then calculate the force at time $t+\Delta t$ and finish updating the velocity.

3.4 Event-driven molecular dynamics

The technique that we use to simulate the dynamics of the binary hard-sphere system is event-driven molecular dynamics (EDMD) [65], which is closely related to time-driven molecular dynamics. Similar to time-driven MD, the goal is to solve Newton's equation of motion to extract the particle trajectories. However, in contrast to time-driven MD, the general idea of EDMD is that we only simulate the 'interesting' parts of the trajectories. For particles with stepwise constant interaction potentials, such as hard spheres these interesting parts, called events, are instantaneous, meaning that we can assign a specific time t to a specific event. For the hard-sphere system such an event could be the collision between two particles, but in principle other instantaneous events, such as a collision between a particle and a hard wall, can be implemented as well. In between events, particles follow a straight ballistic trajectory, meaning that it is very easy to extrapolate their positions from their initial position and velocity. Note that this simulation technique is only possible when the free movement of particles is simple; in other cases you are restricted to numerical integration to find the trajectories of the particles.

By using EDMD we bypass the problem with the stepwise potentials that we encountered with time-driven MD simulations. Since we know the physics behind an instantaneous collision, we can simply find the moment in time at which two particles will collide and compute their new velocities after the collision. To understand how EDMD works in practice, we will first look at the physics of an instantaneous collision and then explain how the simulation finds, stores and processes these collisions and other events.

Collisions

In order to implement the velocity events in the EDMD simulation, we need to know the velocity change of particles during a collision and we need to find the time at which the collision takes place. We assume here that the instantaneous collision between particles is an elastic collision, meaning that energy and momentum are conserved. A collision between two hard particles happens when the center-to-center distance of these particles is equal to the sum of the two radii. If at time $t=0$ we have two particles i and j (with diameters σ_i and σ_j and masses m_i and m_j) located positions $\mathbf{r}_i$ and $\mathbf{r}_j$, and with respective velocities $\mathbf{v}_i$ and $\mathbf{v}_j$, their center-to-center distance at a time t is

$$d = |\mathbf{r}_i + \mathbf{v}_i t - \mathbf{r}_j - \mathbf{v}_j t| = |\mathbf{r}_{ij} + \mathbf{v}_{ij} t|, \tag{3.5}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. By setting d equal to $\sigma_{ij} = (\sigma_i + \sigma_j)/2$ and squaring the equation, we find that a collision will take place when:

$$\sigma_{ij}^2 = v_{ij}^2 t^2 + 2bt + r_{ij}^2, \tag{3.6}$$

where $b = \mathbf{v}_{ij} \cdot \mathbf{r}_{ij}$. Solving for t yields

$$t = \frac{-b - \sqrt{b^2 - v_{ij}^2(r_{ij}^2 - \sigma_{ij}^2)}}{v_{ij}^2}, \tag{3.7}$$

where we chose the minus solution of the quadratic equation since we are interested in the first time the particles make contact. Using energy and momentum conservation, and keeping in mind that the force between two particles must act along the $\mathbf{r}_{ij}$ direction it can be shown that after the collision, both particles undergo a velocity change equal to

$$\Delta\mathbf{v}_i = -\frac{2bm_j}{(m_i+m_j)\sigma_{ij}^2}\mathbf{r}_{ij} \text{ and } \Delta\mathbf{v}_j = \frac{2bm_i}{(m_i+m_j)\sigma_{ij}^2}\mathbf{r}_{ij}. \tag{3.8}$$

Structure of an EDMD simulation

Now that we have derived the equations that allow us to predict and process a collision, we can look at the general setup of EDMD simulations. Given a system of particles with initial positions and velocities, the EDMD simulation will compute and store which collisions will take place at what time in the future. In principle the collision time could be computed for every possible particle pair. However, predicting collisions for particles that are very far away from each other is not that useful; before these particles could possibly collide, their trajectories are most likely already altered due to other collisions. Therefore, we divide the simulation box (i.e. the volume to which particles during the simulation are restricted) into cubic cells with an edge length of at least the diameter of the largest particle, such that a particle can only collide with other particles in the same or neighbouring cells. In our algorithm, each cell contains a cell list that stores all the particles that are in that cell. When computing the possible future collisions for a certain particle i, we only consider particles that are in neighbouring cells. This drastically reduces the time required to predict future collisions.

The introduction of cells also means that the simulation needs to have a *cell crossing event*. This is the event associated with a particle moving from one cell to the other during its ballistic motion. During this event the particle is added to the new cell list and the collisions with particles in the new neighbouring cells are computed. At the start of the simulation and after every collision or cell crossing we compute the time at which the next cell crossing happens for every associated particle.

While simulating these events, we also need a well-ordered data structure that allows us to save and delete events in an efficient way. The fact that we also need to delete events has to do with the fact that every time a particle collides with another particle, the simulation needs to delete all the previously planned events associated with that particle and calculate all new possible events. The main requirement of the data structure that we need is that it must be possible to quickly find the place in time where the new event must be added. If we would simply make an array with all events ordered in time, adding an event would require a lot of time; every time you want to add an event at time t_{event}, the computer needs to move all events with time $t > t_{\text{event}}$ up by one place. Instead we therefore use a binary search tree [51]. In this type of data structure each piece of data, in our case an event, is called a node. Nodes are linked to other nodes by directed edges such that a tree-like structure arises (see Figure 3.1). In the tree structure nodes are hierarchically linked to each other, in what we call parent-child relations. Every parent can have up to two node children; one on the left and one on the right. The left child is associated with an event that takes place at an earlier time than the parent event, while the right child is associated with an event that takes place at a later time than the parent event. The top of the event tree is called the root event. In Figure 3.1 a possible event tree is shown for a system containing 3 particles.

This binary tree allows you to add and remove events very efficiently. The first event that takes place is simply the utmost left event as seen from the root node. Adding an event is done by again starting at the root, but now moving right or left depending on whether the event you want to add takes place at an earlier or later time than the time associated with the current node. When this search terminates at an empty place in the event tree, the new event is added there (this could be under a parent that already has a child, e.g. if you were adding something at time 2.2 in Fig. 3.1). Deleting an event is done by removing the event from the tree and, if there are children, linking those children back into the tree to new parents [51].

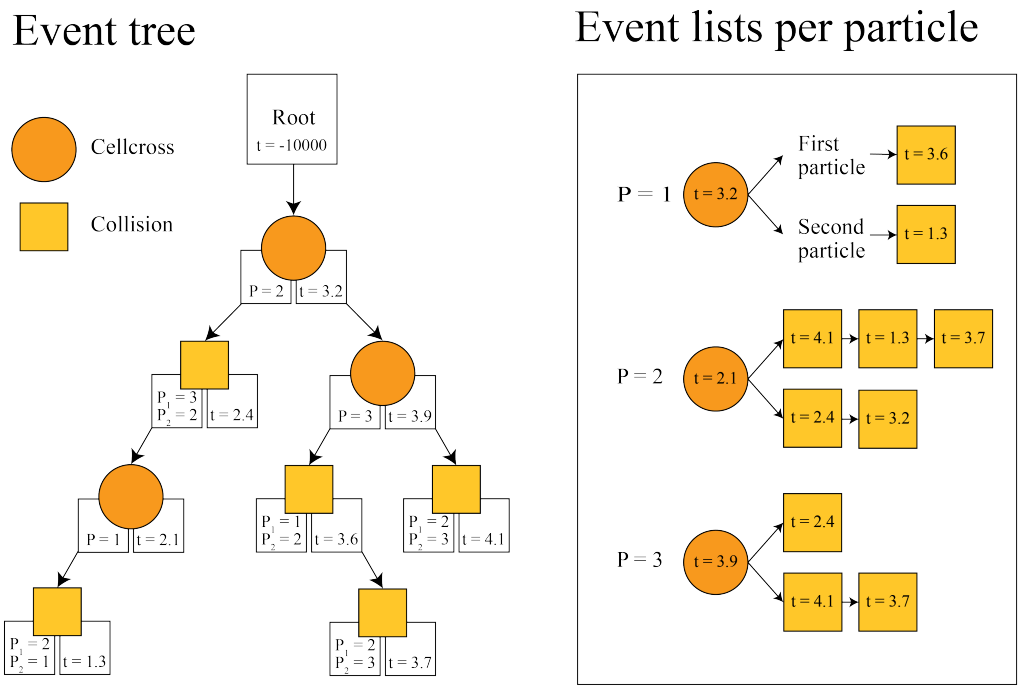


Figure 3.1: Possible event tree for a system with 3 particles. Each square/circle represent an event, either cell crossing or collision, associated with one or two particles respectively and a time t. Each arrow that starts on the bottom left of an event, points to an event that takes place earlier than the selected event, while each arrow that points to the right points to an event that takes place later. On the right the events list per particles are shown. Each event list starts with a cell crossing.

As mentioned before, we also want to able to remove all events associated with a certain particle. In order to do this, we also make connected lists of all the events that are associated with a certain particle. One list holds all the events where the particle is the first particle (i.e. cell crossings or the first particle in a collision), the other list holds all the events where the particle is the second particle, see again Figure 3.1. Whether a particle is the first or the second particle in a collision, depends on how the collision was calculated. If we try to find a collision for a certain particle, this particle will be the first particle. The second particle is then the particle that will collide with this particle. Since we know for sure that each particle has a cell crossing at one point, we use this event as the root for both lists. Since these lists are merely used to know which events must be deleted, the order of the event times in these lists does not matter.

Running the simulation

Each EDMD simulation starts with initializing the event tree by adding the predicted events, based on the initial configuration. After the tree is built, the simulation looks for the event that happens first and processes that event. Next, the simulation deletes all old events associated with the particles involved in the event, and finds and adds all new events. By repeating these steps over and over again the simulation evolves. Because the simulation leaps from event to event, time does not evolve in fixed time steps, but rather jumps from event time to event time. Note also, that there is no need to update all particle positions when we jump to a new event time. Since we know exactly how particles move between collisions, there is only a need to update the positions of particles that are involved in the current event. This means that, contrary to a standard MD simulation, each particle in the system has its own time label, reflecting the time it was last

updated.

Initializing glassy systems using EDMD simulations

In order to do measurements on the movements of particles in a glassy state, we first need to initialize a system of hard spheres in such a state. As we mentioned before, a glassy state is obtained by rapidly compressing a fluid system. This can either be done by starting the EDMD simulation in a large simulation box whose volume is decreased over time, or by starting with small particles that grow over time. In this thesis we choose the last option. Growing a particle means that in addition to their velocity in the *x,y,* and z directions, particles also have a radial growing rate, that can be expressed as a velocity v_r. As a result, we need to take into account that in the calculation of the collision time the radius of the particles at time t will be different compared the radius at time $t + \delta t$. This leads to a small alteration of Equation (3.7). Additionally, growing the particles also means that you increase the energy of the system. Using the equipartition theorem, the kinetic energy of the system is related to the temperature of the system as:

$$\sum_i \frac{1}{2} m_i v_i^2 = \frac{3}{2} N k_B T, \tag{3.9}$$

where the sum goes over all particles in the system. In a collision with two particles that are growing, energy is no longer conserved; during the collision the particles get an extra 'kick' due to their radial velocity. As a result the system heats up over time. In order to keep the temperature constant, we therefore introduce so-called thermostat events in our algorithm. During a thermostat event a random particle is selected and assigned a new velocity sampled from a Maxwell-Boltzmann distribution associated with temperature T.
The EDMD simulation code that we use is an adaptation from an existing simulation code written by Frank Smallenburg, previously used in Ref. [35].

3.5 Monte Carlo simulation

Later on in this thesis we will also use Monte Carlo (MC) simulations [62]. Contrary to molecular dynamics, Monte Carlo simulations do not realistically simulate the dynamics of the system, but instead sample configurations from the chosen thermodynamic ensemble, according to the associated probability distribution. For example, in the canonical ensemble, high-energy configurations would be sampled with a lower probability than low energy configurations. Since equilibrium systems have the property that taking the time-averaged value of a quantity is equivalent to taking the ensemble average, the configurations produced by MC simulations can be used to measure average quantities of the system, such as energy, pressure and the average position of particles. Since MC simulations are easier to adapt to various ensembles and e.g. external fields than EDMD simulations, we make use of them in this thesis when we are interested in time averaged quantities of glassy systems rather than dynamics (see Chapter 7).

In order to generate configurations according to the Boltzmann weight, a Monte Carlo simulation generates random configurations that are accepted or rejected based on the so-called detailed balance criterion. This criterion states that in equilibrium the number of moves going from state o to state n must be equal to the number of moves going from state n to state o. Since we are simulating a system that consists of spherically symmetric particles in a fixed simulation box, new configurations can only be obtained by displacing particles. For simple displacement moves, where we take a single particle and displace it by a random vector, the most common acceptance rule for a displacement move that satisfies detailed balance is given by

$$\mathrm{acc}(o \to n) = \min\left(\frac{P(\vec{r_n}^N)}{P(\vec{r_o}^N)}, 1\right), \tag{3.10}$$

where $P(\vec{r_o}^N)$ is the Boltzmann weight of the old configuration and $P(\vec{r_n}^N)$ is the Boltzmann weight of the new configuration [62]. As a result of this rule, configurations with a lower energy

than the current configuration are always accepted, while configurations with a higher energy will be accepted with a probability equal to the Boltzmann weight of the energy difference between the configurations. Since in this thesis we are looking at hard spheres, the acceptance rule can be simplified even further. Since the potential energy in a system of hard spheres is either infinite, in case of overlap, or zero, in case of no overlap, the acceptance rule takes a binary form; if the new configuration leads to overlap between particles, the probability of accepting the move is zero, while if the new configuration does not lead to overlap, the move will be accepted.

In order to make a displacement move, the most common method is to randomly select a particle and try to move it by an amount $\Delta\mathbf{r}$, where each component of this vector satisfies $\delta r_i \leq \Delta r_{\max}$ for some preset value of $\Delta r_{\max}$. This value of $\Delta r_{\max}$ is quite important since it impacts the speed of your simulation: when $\Delta r_{\max}$ is too small, almost all moves will be accepted, but since the difference between old and new configurations is fairly small, covering the entire phase space will take a long time. A too large $\Delta r_{\max}$, will lead to a low acceptance rate and thus also to a longer simulation time. As it turns out, a reasonable choice of $\Delta r_{\max}$ is typically such that the around 30% of the moves is accepted, although this depends on the system under consideration [66].

The Monte Carlo code used in this thesis was written from scratch in C by the author.

Chapter 4

Machine learning methods

In this chapter we discuss the different machine learning techniques that we use to predict the dynamic propensity based on the structural parameters introduced in Chapter 2.2. As we mentioned in Chapter 1, multiple machine learning techniques have already been found to be suitable for predicting the dynamic propensity of glassy systems [25]. In this thesis we will compare three of these, namely linear regression (LR), neural networks (NNs) and graph neural networks (GNNs). Note that, although Support Vector Machines (SVMs) have also been shown in the past to be able to predict the propensity [36], we will not include the technique in this comparison. Since linear regression and SVM are both based on linear analysis and in Ref. [35] it was shown that they perform very similarly, we choose not to include SVM here. Although standard neural networks have not been previously used to predict the propensity, we include it in this comparison since, in terms of complexity, it is a method that lies in between LR and GNNs.

We know from Ref. [35] and Ref. [36] that both linear regression and GNNs can predict the propensity approximately equally well in the case of the Kob-Andersen mixture. However, in these two papers different input parameters were used. In the case of linear regression Boattini *et al* use the recursively defined structural parameters we described in Chapter 2.2, while in Bapst *et al*, the GNN was only provided with information about the size of particles and about the vectors connecting nearest neighbours.

The fact that GNNs are able to predict the propensity based on only a very basic structural input is quite astonishing. However, the fact that the GNN is given only the 'raw' structural input, instead of the pre-selected order parameters, leads to the need for a complex GNN architecture. This has the disadvantage that it leads to a high computational cost and the requirement of a large amount of training data. To put this in perspective, the GNNs used in Ref. [36] used on the order of 10^5 fit parameters, while the LR approach in Ref. [35] only used on the order of 10^3. The fact that linear regression can perform nearly as well as the GNN by using intelligently chosen structural parameters raises the question whether the GNN could perhaps perform better if it too would be provided with more physically motivated input data.

General approach of Machine Learning

In the following sections we elaborate on how machine learning methods make predictions. The overall idea for all techniques under consideration is the same: in all cases a model is optimized by training the method on so-called training data. This data consists of both the input parameters, i.e. the structural parameters, as well as the desired output propensity. By comparing the predicted propensity with the actual propensity, the free parameters in the model are adjusted to optimize the accuracy of the prediction. After an algorithm is trained, its performance is evaluated by examining how well it performs on never seen before test data set. Since machine learning comes with a lot of terminology, in Appendix A one can find a Glossary that summarizes most of the important jargon. Before we elaborate on the details of the different methods, we first discuss how to quantify the accuracy of a trained model.

4.1 Analyzing performance of machine learning techniques

When we are examining the performance of a machine learning method, we are interested in how close – on average – the predicted propensity of each particle is to the measured propensity. If we would make a 2D plot with on one axis the predicted propensity and on the other axis the measured propensity, we are thus interested in how close the points lie to the line $y = x$. In order to evaluate this criterion, we will use two different parameters, namely the Pearson correlation coefficient ρ and the coefficient of determination r^2. The reason that we use two parameters, is that while the Pearson correlation coefficient is commonly used to evaluate the performance of ML predictions of glassy dynamics (see e.g. Refs. [35] and [36]), the coefficient of determination provides a better measure of the overall performance. Below, we will discuss both parameters and explain why we prefer r^2 over ρ.

Pearson coefficient

The Pearson coefficient is a measure for the linear correlation between two data sets [67]. Given the data sets X and Y, this coefficient is defined as

$$\rho_{XY} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}, \tag{4.1}$$

with $\text{cov}(X, Y)$ the co-variance, and σ_X and σ_Y the standard deviations of data sets X and Y respectively. Due to the normalization by the standard deviations, the value of ρ_{XY} will always lie between -1 and 1.

The Pearson coefficient reflects how deviations from the mean value in each data set correlate with each other; when in data set X a deviation in one direction corresponds to a deviation in the other direction in data set Y, we say that the data sets are negatively correlated resulting in a negative ρ_{XY} (where $\rho_{XY} = -1$ reflects perfect anti-correlation). When it is the other way around, we say that the data sets are positively correlated, meaning that ρ_{XY} is larger than zero (where $\rho_{XY} = 1$ reflects perfect correlation). Since a perfect prediction trivially results in a perfect correlation ($\rho_{XY} = 1$), the Pearson coefficient provides a measure of the accuracy of the prediction.
The problem with the Pearson coefficient is that it only measures to what extent two data sets are linearly related, and not whether the predicted and measured values lie close to each other: if the method always predicts a value that is e.g. 10 times as high as the true value, this would still be a perfectly linear relation and thus result in $\rho_{XY} = 1$.

The above observation does not mean that we cannot use the Pearson coefficient to analyze the models. As Bapst *et al* and Boatini *et al* have already shown, in practice the quality of the Machine Learning predictions is of such a level, that the majority of the points is centered around $y = x$ [35, 36]. This means that measuring ρ_{XY} in essence tells you how close the points lie to the line $y = x$ and thus how well the method performs.

Coefficient of determination

However, even though the Pearson coefficient indirectly reflects the performance of the machine learning methods, it may be better to quantify performance in a way that takes into account possible systematic deviations in the predictions. One of those is r^2, also known as the coefficient of determination, which is defined as

$$r^2 = 1 - \frac{\sum_{i=1}^{N} (y_i^{\text{predict}} - y_i^{\text{measure}})^2}{\sum_{i=1}^{N} (y_i^{\text{measure}} - \bar{y}^{\text{measure}})^2}, \tag{4.2}$$

where the summations runs over all N particles for which we are making a prediction, where y_i^{predict} and i, y_i^{measure} are respectively the predicted and measured propensity for particle i and where $\bar{y}^{\text{measure}}$ is the mean propensity of the measured values. In this expression, the numerator

contains the mean squared error (MSE) between the predicted and measured data set, while the denominator contains the variance of the measured data set.

The reason that we divide the MSE by the variance, is because how 'wrong' a prediction is, depends on the spread in the measured variable itself; if the spread is large, deviations between the measured and predicted value matter relatively less compared to the situation where the spread is small. If (and only if) the prediction is perfect, the MSE is zero, resulting in $r^2 = 1$, while if every prediction of the model would yield the average value of the measured data the MSE would be equal to the variance resulting in $r^2 = 0$. For even worse predictions r^2 becomes negative.

Since r^2 reflects the difference between the predicted and measured value, it is a better quantity for measuring the performance of the machine learning methods than the Pearson coefficient. In order to allow for a comparison to past work, in Chapter 5 we use the Pearson coefficient to analyze the machine learning methods, while in later chapters we will switch to r^2.

We now proceed to the discussion of the three machine learning techniques, LR, NN and GNN, that we use in this thesis.

4.2 Linear regression

The least complicated technique of the three machine learning techniques is linear regression. To make a prediction the method assumes that there is a linear relation between input data and output data. For particle i in our system this linear relation can be written as

$$\Delta r_i(t) = \mathbf{x}_i \cdot \mathbf{w}(t) + b(t), \tag{4.3}$$

where $\Delta r_i(t)$ is the propensity at time t, $\mathbf{x}_i$ is the parameter vector that contains the structural parameters (taken at time $t = 0$), and $\mathbf{w}(t)$ and $b(t)$ are the fit parameters associated with the linear relationship. The aim is to find the parameters $\mathbf{w}(t)$ and $b(t)$ that lead to the best correlation between the predicted and measured output.

In our case, we use linear regression to minimize the mean squared error between the predicted propensity $\Delta r(t)$ and the measured propensity $\Delta \hat{r}(t)$ with respect to $\mathbf{w}(t)$ and $b(t)$, where in this case the MSE is defined as

$$\text{MSE} = \sum_i \left(\Delta r_i(t) - \Delta \hat{r}_i(t)\right)^2 = \sum_i \left(\mathbf{x}_i \cdot \mathbf{w}(t) + b(t) - \Delta \hat{r}_i(t)\right)^2, \tag{4.4}$$

where the summation runs over all particles in the data set.

Preventing overfitting

Although minimizing the MSE will normally yield good predictions for the data that was used to train the model, the method can be sensitive to overfitting, i.e. the optimized model may not generalize well to new data. One of the ways to avoid this overfitting, is to add a penalty in the fitting algorithm for large coefficients of $\mathbf{w}$. One of the most common penalties is Ridge regression (also called L_2-regularization) [68]. With Ridge regression a term of $\alpha \mathbf{w}^T(t)\mathbf{w}(t)$ is added to the MSE, where α is a constant parameter that is chosen such that on new data, the linear regression prediction yields the highest precision. After adding the penalty associated with Ridge regression, the term that needs to be minimized with respect to the linear regression parameters is given by

$$\sum_i \left(\mathbf{x}_i \cdot \mathbf{w}(t) + b(t) - \Delta \hat{r}_i(t)\right)^2 + \alpha \mathbf{w}^T(t)\mathbf{w}(t), \tag{4.5}$$

Due to the quadratic nature of Equation (4.5), the minimization of the MSE can be written as a linear optimization problem, which can be efficiently solved with standard linear algebra methods. For this we use the linear regression model from the python package *scikit-learn* [69]. For each

time t, we carry out the minimization for multiple values of α, and finally choose the value of α such that the linear regression prediction on new data yields the highest accuracy.

4.3 Neural networks

The second technique that we use to predict the propensity are neural networks. This commonly used machine learning technique has the advantage over normal linear regression that it allows for non-linearity. Below we will discuss how neural networks work and how the non-linearity comes about.

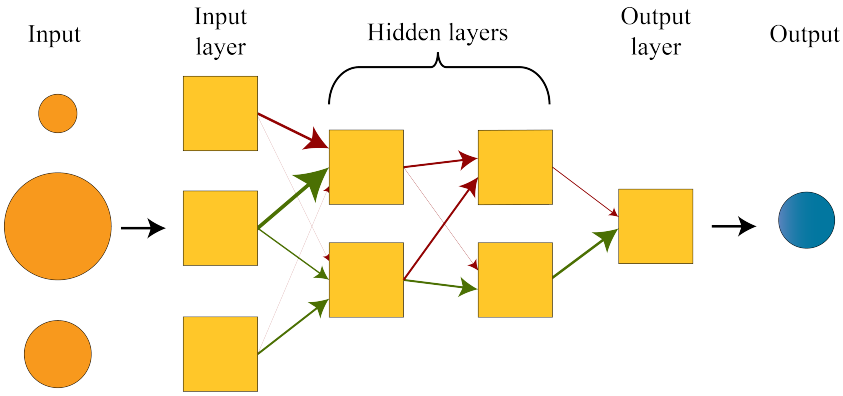


Figure 4.1: A Neural network consisting of an input layer, hidden layers and an output layer. The orange and blue circles represent input and output parameters respectively. The red and green arrows represent the weights that connect each layer in the neural network.

Neural networks are loosely based on the biological neural networks that make up our brains [70]. A neural network consists of multiple layers of connected nodes, see Figure 4.1, which mimic the neurons and synapses in the brain. The first and last layer are respectively called the input and output layer, which in our case take the structural parameters of each particle as an input, and give the predicted propensity for a certain time as the output. All the layers that lie between the input and output layer are called hidden layers. In a fully connected feed-forward neural network, as we use here, each node in a specific layer is connected to all the nodes in the following layer.

Due to the connections between nodes, information can be passed through the neural network. This is again in analogy with the brain, where signals of information travel via neurons. How the information travels through the network can be seen in Figure 4.1. Each connection between nodes is associated with a so called 'weight', which can either be positive (indicated as green) or negative (indicated as red). The values of the nodes in the hidden layer and the output layer are found by multiplying the values of the nodes in the previous layer with the associated weights. Afterwards a node-specific 'bias' is added to the total. The result is then given to a non-linear function – often a sigmoid or a step-function – to yield the value of the node, which is then used, in combination with all the other nodes in this specific layer, to calculate the values of the nodes in the next layer. Hence, the value of a node a_m in layer l is calculated as

$$a_m^{(l)} = f\left(\sum_n w_{mn}^{(l)} a_n^{(l-1)} - b_m^{(l)}\right). \tag{4.6}$$

Where f is the non-linear function, $w_{mn}^{(l)}$ is the weight associated with the connection between nodes n and m, $a_n^{(l-1)}$ is the information of node n in layer *l-1* and $b_m^{(l)}$ is the bias associated with

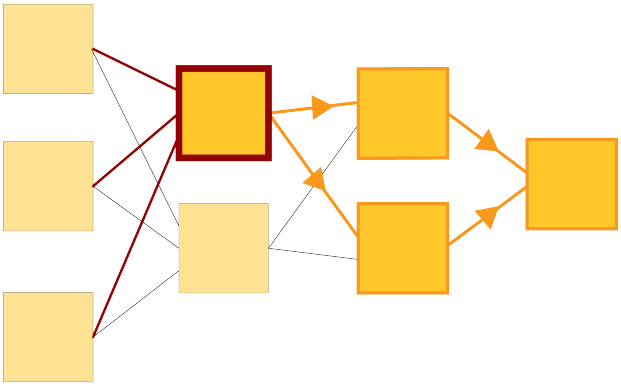


Figure 4.2: A schematic overview of Backward propagation. Consider the the top node in the first hidden layer (indicated by the box outlined in red) and the weights associated with this node (indicated by red lines). The loss function indirectly depends on these weights via later layers – indicated by orange boxes.

node $a_m^{(l)}$. The summation over n goes over all nodes in layer $l-1$. In order to make the notation more concise we can write it down in in a matrix based form:

$$\mathbf{a}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}). \tag{4.7}$$

Since the updating of a node requires knowing the value of the nodes in the previous layer, information travels from left to right through the network.

Although a neural network might seem very complicated, its structure is very closely related to linear regression: in essence linear regression is a neural network without hidden layers and without a non-linear function. The real difference between the two techniques, is the algorithm that is used to find the optimal values of the weights and biases. In the case of linear regression these parameters are obtained by minimizing the MSE error function, which has a single global minimum. However, due to the complex structure of the neural network, the corresponding MSE will have multiple minima, meaning that finding the best parameters requires another approach. Below we will discuss how a neural network finds its parameters, a process that we call 'learning'.

Learning

To evaluate its performance, a neural network compares its predicted outcome with the desired outcome via the so called 'loss function'. Just as in the case of linear regression, the mean square error defined in Equation (4.4) is often used as a loss function. In the process of training, a neural network tries to find the global minimum of this loss function by exploring its value in the multi-dimensional space made up of all weights and biases. Since walking around randomly would result in a very long training time, the search for the minimum is generally optimized by computing the gradient of the loss function with respect to all the weights and biases at discrete steps and then adjusting those weights and biases such that a step is taken along this gradient direction [71].

Although the loss function usually depends on a large number of variables, finding the gradient is simplified by the fact that the value of each node is a relatively straightforward function of the values of the nodes in the preceding layer. The technique that is used to find the gradient is called backpropagation [71], where the term stems from the fact that the components of the gradient are computed in a backwards fashion, i.e. we start with computing the dependence of the loss function on the weights and biases in the last hidden layer and then work our way back to earlier layers. The reason for this backwards direction is that weights and biases in earlier layers

influence the loss function via nodes in later layers (see Figure 4.2). The influence of the earlier layers can thus only be obtained by knowing how nodes in later layers influence the loss function, something that can only be achieved by working backwards.

To demonstrate this, we consider a fully-connected neural network where the last hidden layer has an arbitrary number of output nodes. Although using an arbitrary number of output nodes may seem less applicable to our case, where we only consider one output node, the derivation can very easily be simplified to an output layer with only one node. Assume that we have a network with L layers and a loss function C defined as

$$C = \left(\hat{\mathbf{r}}(t) - \mathbf{a}^{(L)}(t)\right)^2. \tag{4.8}$$

Here $\hat{\mathbf{r}}(t)$ is the desired output and $\mathbf{a}^{(L)}(t)$ is the output from the network. From now on we omit the t-dependence.

Our aim is to find the influence of an arbitrary weight or bias in the network on this loss function. Note that since all weights or biases influence the value of the corresponding node in the next layer via $\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$ with $\mathbf{z}^{(l)} = \mathbf{w}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$, determining the influence of $z^{(l)}$ on the loss function will immediately provide us with the influence of all the associated weights in the previous layer.

We first look at the influence of the node j in the output layer on the loss function, i.e. we want to find $\Delta C_j^{(L)} \equiv \partial C/\partial z_j^{(L)}$. Since C depends on $a_j^{(L)}$ via the loss function (Equation (4.8)) and $a_j^{(L)}$ depends on $z_j^{(L)}$ via the non-linear function (Equation (4.7)), we can write:

$$\Delta C_j^{(L)} = \frac{\partial C}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} = 2(a_j^{(L)} - \hat{r}_j) f'(z_j^{(L)}) \tag{4.9}$$

In matrix notation this becomes

$$\boldsymbol{\Delta} C^{(L)} = 2(\mathbf{a}^{(L)} - \hat{\mathbf{r}}) \circ f'(\mathbf{z}^{(L)}), \tag{4.10}$$

where the operator $\circ$ means that we multiply the vectors element wise. The influence of $\mathbf{w}^{(L)}$ and $\mathbf{b}^{(L)}$ is now found simply by multiplying $\boldsymbol{\Delta} C^{(L)}$ with respectively $\partial \mathbf{z}^{(L)}/\partial w^{(L)}$ and $\partial z^{(L)}/\partial b^{(L)}$. Using Equation (4.7) we find:

$$\frac{\partial C}{\partial \mathbf{b}^{(L)}} = \boldsymbol{\Delta} C^{(L)} \circ \frac{\partial \mathbf{z}^{(L)}}{\partial \mathbf{b}^{(L)}} = \boldsymbol{\Delta} C^{(L)} \text{ and } \frac{\partial C}{\partial \mathbf{w}^{(L)}} = \boldsymbol{\Delta} C^{(L)} \circ \frac{\partial \mathbf{z}^{(L)}}{\partial \mathbf{w}^{(L)}} = \boldsymbol{\Delta} C^{(L)} \circ \mathbf{a}^{(L-1)}, \tag{4.11}$$

since $\partial \mathbf{z}^{(L)}/\partial \mathbf{b}^{(L)} = \mathbb{1}$ and $\partial \mathbf{z}^{(L)}/\partial \mathbf{w}^{(L)} = \mathbf{a}^{(L-1)}$.

With the knowledge of the derivatives in layer L we can now find the gradient with respect to the weights and biases in layer $L-1$ and so on. Using the chain rule, we can write a general formula for the influence of $\mathbf{z}^{(l)}$ on the loss function, assuming that we know $\boldsymbol{\Delta} C^{(l+1)}$, meaning that we know the influence of the values of $z^{(l+1)}$ on the loss function. We can write $\boldsymbol{\Delta} C^{(l)}$ in terms of $\boldsymbol{\Delta} C^{(l+1)}$, via

$$\boldsymbol{\Delta} C^{(l)} = \frac{\partial C}{\partial \mathbf{z}^{(l)}} = \left(\frac{\partial C}{\partial \mathbf{z}^{(l+1)}} \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}}\right) \circ \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} = ((\mathbf{w}^{(l+1)})^T \boldsymbol{\Delta} C^{(l+1)}) \circ f'(\mathbf{z}^{(l)}), \tag{4.12}$$

where we used that $\partial C/\partial \mathbf{z}^{(l+1)} = \boldsymbol{\Delta} C^{(l+1)}$, $\partial \mathbf{z}^{(l+1)}/\partial \mathbf{a}^{(l)} = \mathbf{w}^{(l+1)}$ and that $\partial \mathbf{a}^{(l)}/\partial \mathbf{z}^{(l)} = f'(\mathbf{z}^{(l)})$. Again using that $\partial \mathbf{z}^{(l)}/\partial \mathbf{w}^{(l)} = \mathbf{a}^{(l-1)}$ and $\partial \mathbf{z}^{(l)}/\partial \mathbf{b}^{(l)} = \mathbb{1}$, we can write down the derivative of the loss function with respect to the weights and biases in layer l as:

$$\frac{\partial C}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\Delta} C^{(l)} \text{ and } \frac{\partial C}{\partial \mathbf{w}^{(l)}} = \boldsymbol{\Delta} C^{(l)} \circ \mathbf{a}^{(l-1)}, \tag{4.13}$$

with $\boldsymbol{\Delta} C^{(l)}$ as written down in Equation (4.12).

Equations (4.10), (4.12) and (4.13) give us all the information we need to compute the gradient of C with respect to the weights and biases, and thus to train the network. In order to make the training process more efficient, most training algorithms do not calculate the gradient after every individual training example, but instead look at many data points together. Not only does looking at such a 'batch' speed up the simulation, it also ensures that the network does not get influenced too much by the outliers of single points.

To what extent the system adjusts its weights and biases after each batch depends on the learning rate of the network. The value of this learning rate is very important for the training process. If it is too small the system only takes small steps in the potential landscape of the loss function, meaning that it can take a long time to reach the global minimum. Another risk of a small learning rate is that the system can get stuck in a local minimum because the step size is too small to get out again. A too large learning rate on the other hand, gives a risk of overshooting the minimum, meaning that the system never reaches the minimum. Finding a good learning rate is one of the major challenges of building a good machine learning algorithm [71].

With the discussion above the question arises when one should stop training the network. At what point can we say that the network has found the optimum value of its weights and biases? In principle one could keep training the network on the same training data over and over again (where each time the network has seen all training data is called an epoch) and improve the accuracy of this training data prediction. However, doing so will lead to a model that is very bad in generalizing to new data. Before this point is reached, the training yields a useful trade-off between accuracy and generalization. To find the optimal point we evaluate the network after every epoch with test data that it did not use for training. Whenever the accuracy of the test data prediction starts to decrease, we assume the network has found its optimal value and we stop the training.

To train the neural network we use the Pytorch Python package [72], together with an Adam optimizer [73]. This optimizer is an extension to the stochastic gradient descent procedure, and is used to find an efficient path to the optimal weights and biases using backwards propagation. During training we consider various values of the learning rate and the neural network architecture (as will be discussed in the next chapter).

Prevent overfitting

Just as with linear regression, overfitting can become a problem with neural networks. Along the same lines as in LR, we can penalize large weights to discourage this overfitting. Another solution – stop training when the network starts to perform less on new data – was already mentioned earlier.

In neural networks, we also see another form of overfitting, namely the oversensitivity of the network to collections of parameters, something that is called co-adaption [74]. It is a result of the non-linear nature and high number of fit parameters of the model: if the relation between input and output parameters is sufficiently complicated, the neural network can often find many complex patterns in the training data that yield a good outcome for this data, but that do not always generalize to new data. One way of avoiding co-adaption, is to train many different neural networks, and taking the average prediction of all these models. This however is a very time consuming task and not always feasible. Therefore we avoid co-adaption by a method called drop-out [75]. While processing the training data, a certain percentage of the nodes are randomly selected and set to zero, preventing nodes from becoming too correlated with each other. Note that the optimal percentage of nodes that should be set to zero is something that will depend on the specific application of the network.

4.4 Graph neural networks

As a third method, we also use graph neural networks (GNNs) to predict the propensity. GNNs are a relatively new class of machine learning techniques introduced in Refs. [76–78] that combine

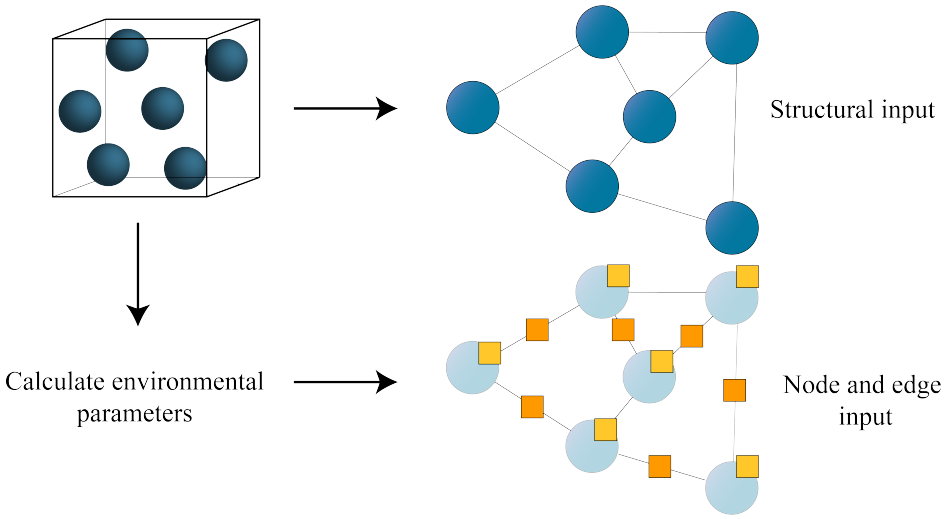


Figure 4.3: Set up of a graph structure for particles. From the system that is shown in the top left corner, we calculate structural parameters and we set up the associated graph in the right panels. Each node and edge in this graph are then coupled to the associated structural parameters.

neural networks with a graph-like data structure. As there are many variations of GNNs, our description is necessarily somewhat specific to the GNN we use in this thesis. In the context of predicting dynamic propensity, the key difference with normal neural networks is that GNNs allow predictions for nearby particles to influence each other. In the case of propensity predictions, this means that neighbouring particles affect each others predictions. In order to achieve this, GNNs take graph-like data structures as an input [79]. This means that instead of considering individual particles, we now consider the whole system of N particles at once.

A graph is a data structure that holds information about a collection of objects, which are called 'nodes', and about the relations between these nodes. Whenever two objects are related, they are connected by an 'edge'. If we would for example make a graph of the friendships in a group of people, every person would be visualized as node, while the friendship between two people would be visualized as an edge between the associated nodes. When a connection between two nodes works in a symmetrical way, we say that the edge is undirected. This could e.g. be the case for the friendship example above; if A is friends with B, B is also friends with A. If a relation only works one way, we say that the graph is directed.

The data describing our glass configuration can also be expressed like a graph: in that case each node corresponds to a particle in the system. When two particles are closer to each other than a certain distance r_c, we say that the corresponding nodes in the graph form an edge. The idea of using this graph structure for our data, is that it allows us to predict the propensity of each particle not only based on its own input parameters, but also on the parameters of neighbouring edges and nodes. This means that a GNN does not see a particle as a separate entity, but as an object embedded in a larger structure.

To understand how the graph network works and trains it is very important to make a distinction between two types of information that the GNN holds and uses, see Figure 4.3. On the one hand we have the information about how the graph is structured i.e. the number of nodes and their associated edges. On the other hand, there is the physical information about the system that the network uses to make predictions. In our case every node holds the structural parameters of the corresponding particle and each edge holds information about the distance between the two

connected particles. The information about the graph structure is only used by the network to know which particles are neighbours, while the physical information is used in neural-network-based calculations to make predictions.

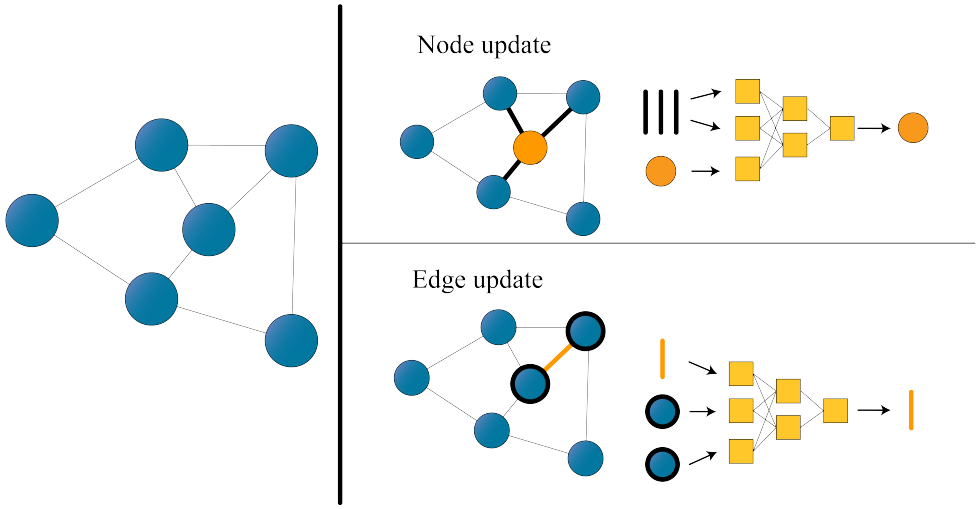


Figure 4.4: Node and edge update in a graph neural network. For the updating of the node data, the added and averaged parameters of the connected edges and given to the node neural network, together with the information of the node itself. In the case of the edge update, the information of both neighbouring nodes, together with the edge information itself is given to the edge neural network. The output of the node/edge network is an updated value of the node/edge.

Graph updating

The essence of the graph neural network is that, before using structure as an input for predicting the propensity, it will first update the values of the nodes and edges multiple times, which also takes into account the information of the neighbouring nodes and edges. How this works can be seen in Figure 4.4. Each update of the edges and nodes corresponds to a so-called graph layer inside the GNN. In every graph layer we start by updating the edges. This is done by feeding the edge parameters and the parameters of the two neighbouring nodes to a neural network. The output of the neural network then contains the updated parameters of the edge. Every edge in the graph is updated by this same neural network.

The next step is to update the nodes. For this purpose, the parameter values of that node, together with the parameter values of the neighbouring edges will be fed to a neural network. Because this same neural network is used for every node in the graph, the input dimensions for this network cannot depend on the number of edges that any specific node has. Therefore, instead of feeding the edge parameters directly to the network, in our GNN we calculate the mean value and the total value of the edge parameters. This way the edge input takes the same form for each node, independent of the number of edges. The output of the neural network now is the updated value of the node. After being processed by one graph layer, the parameters of the node and edges not only contain information about their own parameters, but also about the structural parameters of the neighbouring edges and nodes. To include also information about nodes and edges further away, we can simply include more graph layers.

In principle, the input for the second and later graph layers could simply be just the output of the nodes and edges of the previous graph layer. However, in our case we apply a slightly different procedure, also used in Ref. [36]. We chose to give a combined input of the output of the previous

graph layer, i.e. the current node and edge parameters together with the initial parameters of the same respective node or edge before they were updated by the first graph layer. The reason for this, is that the updated node and edge parameters contain averaged information about multiple neighbouring edges and nodes, meaning that information about the initial parameters of the central particle will only be present very indirectly. However, given the physical characteristics of the system we expect that these initial parameters are likely to be of large importance for the prediction; therefore we choose to provide the network with this initial information every time a node or edge is updated.

After multiple layers of graph updating, a propensity prediction is made for each particle. For this we again use a normal neural network that is provided with the current node parameters, as well as with the initial node parameters. Since the current node parameters now contain information about several shells of neighbouring particles, the network makes its prediction based not only directly on a particle's own parameters, but also indirectly on the structure of its wider environment.

Training

Note that although the structure of a GNN is somewhat different from a normal neural network, the training algorithm as well as the methods to prevent overfitting are completely equivalent: we compare the predicted propensities with the actual propensities to compute the loss function, and then adjust the parameters via backpropagation. Training is stopped when the performance of the network on new data starts to decrease. The main difference between a neural network and a graph neural network is that instead of one single neural network, a GNN contains several. Since this typically means that more parameters need to be fitted, training a GNN network usually requires more time and training data.

In order to reduce the number of weights and biases in the network it can be useful to reduce the dimensionality of the input parameters, preferably without losing important information. This can be achieved by using a dimensionality reducing neural network, also called 'encoder', at the beginning of the graph neural network. In addition, to make sure that the input of the nodes and the edges is equally important, we use another NN at the beginning of the GNN that increases the dimensionality of the edge parameters such that it matches the dimensionality of the node parameters after the encoder. Both networks are trained as an integral part of the overall training.

The network is trained using the the *Pytorch* and *Pytorch-Geometric* python packages [72, 80], together with an Adam optimizer [73]. The layer type we use is MetaLayer.

4.5 Advantages and disadvantages of GNNs, NNs, and LR

In the past three sections we have discussed the three selected methods and their features separately. Before we continue presenting the results, it is interesting to discuss how these methods and especially their specific features compare with each other: what are the advantages and disadvantages of each method? And are complex methods always better? In the following section, we will briefly address these questions.

In machine learning there is always a delicate interplay between the complexity of a model and the number of parameters that need to be fitted. If a model is too simple, it will not be able to capture more subtle effects. However, the more parameters that need to be fitted, the more sensitive the model is to overfitting and the more training data will be required.

Although it might seem that complex methods with enough training will in principle improve the prediction, this is only true if their complexity is actually necessary for an accurate prediction. An example to illustrate this, concerns the decision whether or not to allow for non linearity in a model; compared to linear regression, (graph) neural networks have the advantage that they allow for non-linear relations between input and output. However, this advantage will only lead

to a better prediction when there actually is a strong non-linear relation between the structural parameters and the propensity of a particle. In other words: NNs and GNNs will only perform better than LR if their advanced features are actually of any importance to the propensity prediction.

In principle, GNNs have an advantage over the other two methods in that they naturally use information of neighbouring particles to make a propensity prediction. This can be a huge advantage, as evidenced by the success of this method in Ref. [36]. However, as we discussed in Section 2.2, information about neighbouring particles is also provided to the neural network and linear regression via the higher generations of structural parameters [35]. The question is thus whether the exact way that information about neighbouring particles is incorporated affects the prediction: does it for example matter that in the GNN information about neighbouring nodes is passed through via neighbouring edges, while in the case of NN and LR we simply take the weighted average over all particles within a certain radius r_c?

Although complex methods can thus have advantages, these come at the cost of a more complex training process. Not only do more complex methods have more parameters that need to be fitted – meaning that more training data is necessary – they typically also have more hyperparameters. These hyperparameters are parameters that are used to control the training process, such as the batch size, the learning rate and the number and size of the hidden layers in the NNs. In the case of linear regression we only have one hyperparameter, namely α, the parameter that determines the penalization of large weights. In the case of NNs and GNNs, the number of hyperparameters is significantly larger. Especially when the training process of a method takes a long time, optimizing these hyperparameters can be a tedious and computationally expensive task.

In conclusion, GNNs, NNs, and LR all have their own advantages and disadvantages when it comes to fitting complex relationships such as the one between local structure and dynamic propensity. In the next Chapter we will look at how the three methods compare when they are trained to predict the propensity in a glassy system.

PART III

RESULTS

Chapter 5

Comparison machine learning techniques

Comparison propensity prediction machine learning techniques In this chapter we use three different machine learning algorithms to predict the dynamic propensity of a glassy binary hard-sphere mixture, and compare and contrast the results. As we show, out of the three methods, linear regression provides the best compromise between accuracy and efficiency.

5.1 Methods

As mentioned in Chapter 3.2, the glassy system that we examine is a binary hard-sphere mixture at packing fraction $\eta = 0.58$, where the two particle types have a size ratio of $\sigma_B/\sigma_A = 0.85$ and a composition of $N_A/(N_A + N_B) = 0.3$. The relaxation time of this system is approximately $t/\tau = 10^4$ with $\tau = \sqrt{m\sigma_A^2/k_BT}$ where $\beta = 1/k_BT$, and m is the particle mass. Note that this is the same glassy system as was studied in Refs. [35, 61, 81].

To measure the propensity, we use data taken from Ref. [35]. This set consists of 100 snapshots, each containing 2000 particles. The initial equilibrated glassy configurations were obtained using the method described in Chapter 3.4, after which each snapshot was equilibrated for $t/\tau = 10^5$. For each snapshot the average absolute distance traveled by each particle was measured over 50 different runs. This distance was measured at logarithmically spaced time intervals between $t/\tau = 0.01$ and $t/\tau = 10^5$.

For each initial snapshot we also measure the structural parameters as described in Section 2.2. For the radial descriptors we consider 162 different parameters; 81 different r intervals for each type. This means that we have 46 equally spaced spherical shells in the interval $r/\sigma_A = (0.85, 2.0]$ (given that $\sigma_B = 0.85\sigma_B$), 20 equally spaced spherical shells in the interval $r/\sigma_A = (2.0, 3.0]$ and 15 equally spaced spherical shells in the interval $r/\sigma_A = (2.0, 4.5]$. For the angular descriptors we use 192 descriptors; the first 12 spherical harmonics orders for 16 equally spaced spherical shells in the interval $(1, 2.5]$. Note that at distances close to half the box length, the periodic boundaries start to play a role. Therefore we made sure that the box we are using is always bigger than at least twice the maximum distance we are considering. The environment of each particle is described with 3 generations of 354 parameters each, leading to a total of 1062 parameters.

Before using the structural parameters as an input for the machine learning algorithms, they are standardized by evaluating

$$\mathbf{x}_i^{\text{norm}} = \frac{\mathbf{x}_i - \bar{\mathbf{x}}}{\sigma_{\mathbf{x}}}, \tag{5.1}$$

where $\mathbf{x}_i$ is the vector containing all parameters associated with particle i, $\mathbf{x}_i^{\text{norm}}$ is the normalized parameter vector, and where $\bar{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$ are respectively the mean and standard deviation of the parameter vector considering all particles. The need to standardize our parameters arises due to

the penalty that we add to large weights to prevent overfitting. Due to this penalty, providing the algorithm with not-standardized data would lend more weight to the parameters in the fit that have a large variance, since these parameters can have a large impact with a relatively low weight. For the same reasons that we standardize our input parameters, we also standardize the dynamic propensities for each time step separately.

To train the machine learning algorithms, half of the obtained data is used as training data, while the other half is used as test data. For each time interval over which we measured the propensity, the models are trained with several different choices for the hyperparameters. Afterwards, the hyperparameter set that yields the highest Pearson correlation coefficient is chosen. Below we will discuss the different hyperparameters settings for each of the algorithms.

Linear regression

For linear regression the only hyperparameter that can be tuned is α, the parameter that penalizes large weights. For each time step we train the algorithm 10 times, each time with different values of α in a range of multiple orders of magnitude, i.e. $\alpha \in [10^{-4}, 10^{5}]$. We furthermore provide the algorithm with different generations of parameters, i.e. (1) only the zeroth, (2) both the zeroth and the first, or (3) the zeroth, first and second generation. The regression is performed separately for small and large particles.

Neural network

For neural networks (and graph neural networks) there are many more hyperparameters that can be tuned, including the learning rate, the number of layers, the number of nodes in each layer, and the batch size. Since trying out all possible combination of settings would not be feasible, we limit ourselves to a small number of different combinations. In particular, we set the learning rate between 10^{-3} and 10^{-4}, feed the data to the network in batch sizes between 25 and 50 particles and use 1 to 3 hidden network layers, each consisting of 16 nodes. Again the network is trained on large and small particles separately in around 200 epochs. Eventually we use the network that during these epochs yielded the highest correlation for the test data. As in the case of linear regression, we also provide the neural network with different generations of structural parameters.

As the neural network turns out to be very sensitive to overfitting, we also run the neural network with ridge regression and drop-out (as mentioned in Section 4.3). We choose the parameter which penalizes large weights between 0 and 1. The percentage of nodes that is set to zero in the context of applying drop-out is varied between 0% and 25%.

Graph neural network

For the graph neural network we use a learning rate between 10^{-3} and 10^{-4}. We use either 3 or 4 graph layers, each containing their own node and edge neural networks, that consist of 2 hidden layers each. The node encoder also consists of 2 hidden layers, while the edge encoder only has one hidden layer, due to the low number of edge inputs. In all these networks, the hidden layers consist of 16 nodes, except for the first layer in the node encoder, which consists of 30 nodes.

The GNN is trained on large and small particles either simultaneously or separately[1]. The size of the particle is given as a binary input to the network. The graphs are fed to the GNN in batches of 5, and we train for between 200 and 800 epochs. In order to check whether the GNN also benefits from higher generations of structural parameters, we run the network for both the first generation and the first three generations. The input for the edge parameters consists of either the absolute distance between the corresponding particles, or the absolute distance in the x, y and z directions.

[1]Note that since the input of the GNN is a graph of all particles, 'training on large particles' means that also the small particles are used as an input. The difference with training on all particles simultaneously is that when computing the loss function we only include large particles

5.2 Predictive performance of the three methods

5.2.1 Linear regression

In Figure 5.1 we show the linear regression performance for different generations of order parameters. Note that the results for linear regression that we see here are consistent with the results from Ref. [35], see Appendix B. In this figure we clearly see that the predictions from just the zeroth generation of descriptors are significantly worse than the ones including higher-generation data, at least for longer times. In particular, we see that the information of the higher-order generations only starts to influence the performance when the system enters the caging regime. This is what we expected: before entering the caging regime not enough time has passed for particles to be influenced by particles from further away, meaning that higher-order generations will not add relevant information about the expected trajectories.

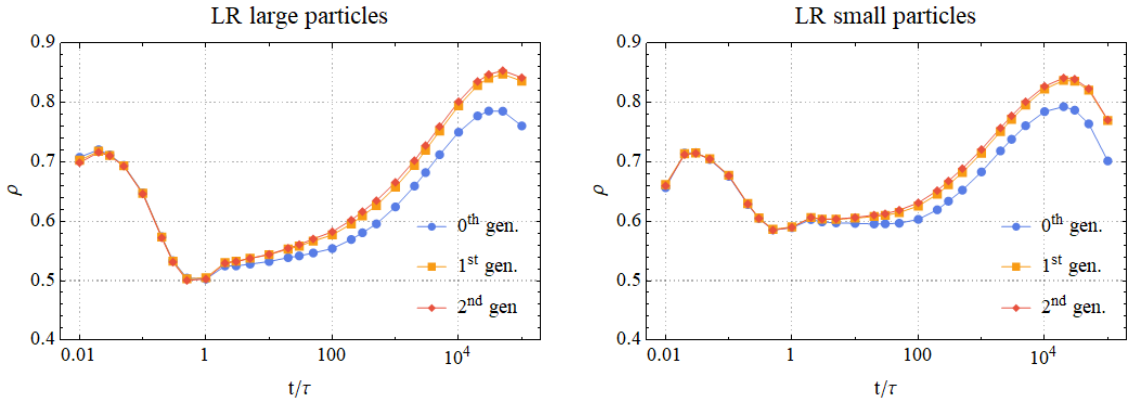


Figure 5.1: Prediction accuracy in terms of the Pearson coefficient for linear regression as a function of time, analyzed for three different generations of order parameters. Note that the caging regime occurs between approximately $t/\tau = 0.1$ and $t/\tau = 100$.

Although adding the second generation of order parameters still improves the predictions for the propensity, the effect is small in comparison to the improvement of adding the first generation. Adding even higher generations (not shown here) does not significantly improve the performance beyond this point.

5.2.2 Neural networks

For the neural networks the propensity prediction performances are shown in Figure 5.2. Here, the different labels correspond to different combinations of hyperparameters, as shown in Table 5.1. In all these five cases the set of input parameters consisted of all three generations of order parameters. Note that the different runs that we present here are only a selection of the different sets of hyperparameters that we explored. However, the other sets of parameters showed similar behaviour and are therefore omitted here.

Although the overall shape of the neural network accuracy is similar to that found for linear regression, the prediction of the NN leads to more noisy lines. The most probable explanation for this, is that the higher number of optimization parameters in the neural network inherently requires more training data. Indeed, by experimenting with different amounts of training data (not shown), we observe that more data leads to smoother predictions as a function of time. This can also be observed by comparing the performance for small and large particles in Figure 5.2: due to the abundance of small particles in the system, there is more training data for small particles, which results in smoother graphs.

The sensitivity of the network to both the input parameters as well as the hyperparameters is a very clear expression of the sensitivity to overfitting. This statement is supported by the observa-

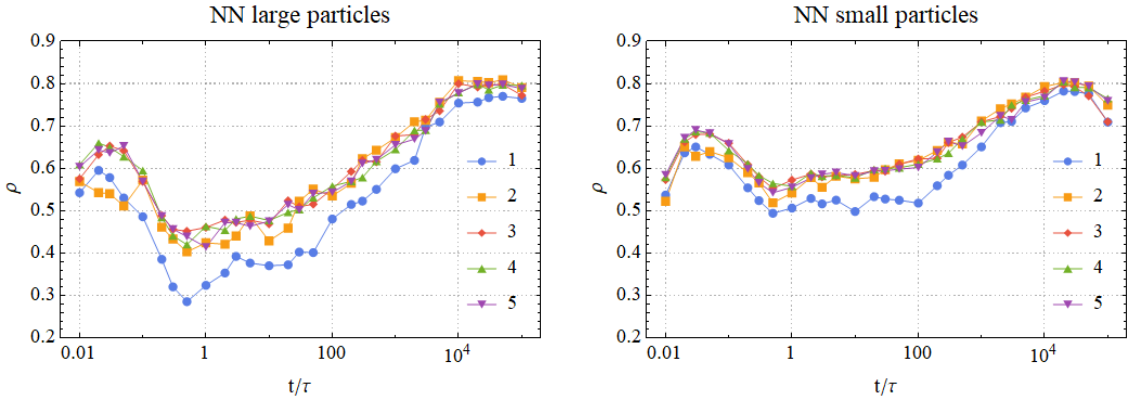


Figure 5.2: Prediction accuracy in terms of the Pearson coefficient of several neural networks as a function of time, analyzed for different hyperparameters as specified in Table 5.1.

Number	Batch size	Learning rate	Runs	Hidden layers	Drop-out	Weight
1	50	10^{-4}	100	1 (16)	0	0
2	50	10^{-4}	100	3 (16,16,16)	0	0
3	50	10^{-4}	100	2 (16,16)	0	1.0
4	50	10^{-4}	100	3 (16,16)	0.25	0
5	50	10^{-4}	100	3 (16,16,16)	0.25	0.01

Table 5.1: Hyperparameters for different neural networks used to predict the propensity. Each row corresponds to a line in Figure 5.2. The *Hidden Layers* entry contains both information about the number of hidden layers in the network, as well as the number of nodes in each hidden layer (shown in brackets). *Drop-out* shows the fraction of nodes that is set to zero during drop-out and *Weight* represent the value of the parameter associated with $L2-$regularization.

tion that the performance of the NN is mostly consistent between hyperparameter sets 3, 4, and 5. In the cases of all these performances we used some protection to overfitting, either in the form of regularization or drop-out.

In order to examine the influence of different generations of order parameters on the performance, in Figure 5.3 we show for hyper parameter set 4, how the algorithm performs when it is provided with up to three different generations of order parameters. Contrary to what we saw in the case of linear regression, providing the NN with more generations of order parameters does not always improve the performance, something that is especially clear for short times. Knowing that the NN is sensitive to overfitting, this result is something we expect. At short times, higher order generations will not yet contain useful information for the propensity prediction. This means that for these times, providing the NN with higher order generations does not provide extra information, while at the same time it makes the network structure more complex. Since a more complex network structure is harder to train, it is not surprising that for lower times, the 0^{th} generation of structural parameters often performs the best.

One could argue that, since the NN is very sensitive to the hyperparameters, it would perhaps perform significantly better if we could find an optimal set of hyperparameters. However, our results suggest that this is not likely the case. For all the runs with different hyperparameters, we did not once observe the NN performing significantly better than the linear regression. Although from the results we can conclude that the NN can get stuck in a local minimum quite easily, it will thus be very unlikely that a new set of hyperparameters would lead to a significantly better performance.

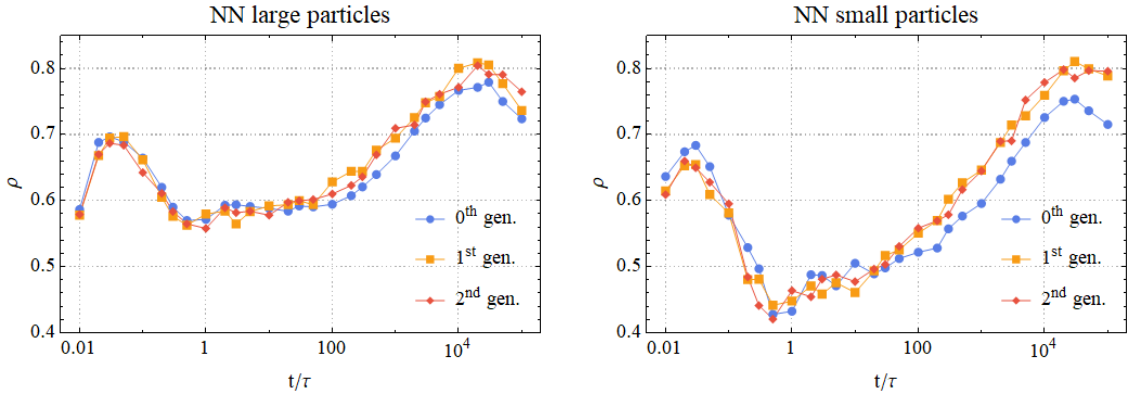


Figure 5.3: Prediction accuracy in terms of the Pearson coefficient of neural networks as a function of time analyzed for three different generations of order parameters. The hyperparameters of the networks correspond to row 4 in Table 5.1.

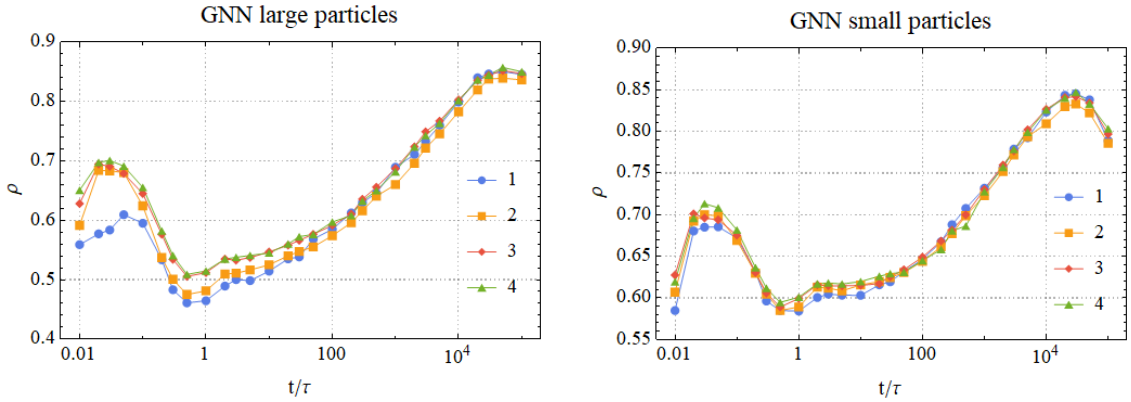


Figure 5.4: Performance in terms of the Pearson coefficient of a graph neural network as a function of time for different hyperparameter settings, which can be found in Table 5.2.

5.2.3 Graph neural networks

In Figure 5.4, we show the results for the graph neural networks. The hyperparameters of the different training runs can be found in Table 5.2. Note that again the figure only shows a selection of the different hyperparameter sets we used. In the figure we can observe that compared to the NN, the GNN is less sensitive to overfitting, resulting in relatively smooth lines. This is actually quite remarkable, since the number of weights and biases that need to be optimized in a GNN is significantly larger than in the case of a NN. Moreover, none of the training runs in Figure 5.4 used any regularization methods. The fact that GNNs, compared to NNs, have less trouble converging and are less sensitive to overfitting, might be due to the fact that they are trained and evaluated on entire snapshots at once. Realistically, in a snapshot we know that there are strong correlations in mobility between neighboring particles. The fact that the GNN can take into account the mobility of neighbouring particles, will thus probably lead to smoother variation of the predicted propensity in space than the NN/LR, which results in fewer outliers.

From the fact that the run that uses three generations of order parameters only shows a slight improvement over the runs that uses only the zeroth generation, we can conclude that the higher-order averaging method, proposed by Boattini *et al* [35] effectively mimics the averaging behaviour that takes place in a GNN. In particular, adding information from higher-order generations to the GNN does not significantly improve the prediction, because the averaging that takes place in the GNN already provides the information that is encoded in the higher generations.

Nr.	Batch size	LR	Runs	GL	Node enc H.L.	Edge enc H.L.	Node H.L	Edge H.L.	Node gen	Edge par	Tog or Sep
1	5	10^{-3}	50	3	2 (50, 50)	1 (5)	2 (30, 16) 2 (16, 16)	2 (16, 16) 2 (16, 16)	1	r	Tog
2	3	10^{-4}	100	3	2 (50, 50)	1 (5)	2 (30, 16) 2 (16, 16)	2 (16, 16) 2 (16, 16)	1	r	Tog
3	5	10^{-4}	100	4	2 (50, 50)	1 (5)	2 (30, 16) 2 (16, 16)	2 (16, 16) 2 (16, 16)	3	r	Sep
4	5	10^{-3}	100	4	2 (50, 50)	1 (5)	2 (30, 16) 2 (16, 16)	2 (16, 16) 2 (16, 16)	1	x, y, z	Sep

Table 5.2: Hyperparameters for different graph neural networks used to predict the propensity. Each row corresponds to a line in Figure 5.4. In the table the following abbreviations are used: *LR* stands for learning rate, *GL* is number of graph layers, *Node enc H.L.* and *Edge enc H.L.* represent respectively the number of hidden layers in the node and edge encoder (for both, the number of input parameters is equal to the number of parameters associated with a node or an edge, while the number of output parameters is equal to 10. *Node H.L* and *Edge H.L.* show the number of hidden layers for the respectively the node and edge hidden layers together with the number of nodes in each layers (the number of output nodes for each of these networks is equal to 10), *Node gen* represents up to how many generations of structural order parameters we provide the GNN with, *Edge par* indicates whether the edge input is given by the absolute distance between particles (r), or the absolute vector distance (x, y, z). Finally *Tog or Sep* represents whether we train the network together (Tog) for small and large particles, or train two separate networks (Sep).

Finally, we observe that the difference in performance for different hyperparameters is minimal. This suggests that apparently GNNs are not that sensitive to hyperparameters, something that was also concluded[2] by Bapst *et al* [36]. As a result, other choices of hyperparameters will likely not significantly improve the result shown in Figure 5.4. Note, furthermore, that the GNN we use is computationally significantly less heavy than the the GNN used by Bapst *et al*. The reason that we could use a simpler GNN, is because we use inputs that are already pre-treated to capture (what we think is) relevant structural input.

5.3 Comparing the three methods

To compare the methods, several aspects play a role. The primary criterion is of course how well each method performs. Secondly, it is important how easy it is to train a method, i.e. how sensitive is the algorithm to its hyperparameters, how much data is required to train the model and how long does it take to train the model. And finally, the ease with which results can be interpreted also plays a role. After all, our final goal is not only to make the best propensity prediction, but also to understand which aspects of the structure influence the dynamical heterogeneity.

In order to compare the accuracy of the three methods, we show for each method and for each moment in time the highest correlation obtained. In Figure 5.5 the results can be found for both small and large particles. The first conclusion that we can draw from this figure is that the three methods overall perform very similarly. This means that the comparison between the methods will mainly be based on how easy is to train the algorithms and to interpret the results, and less on how well they perform.

[2]Note that we cannot compare our exact results with the results of Bapst *et al* [36], since they examined the Kob-Andersen mixture instead of a binary hard-sphere mixture.

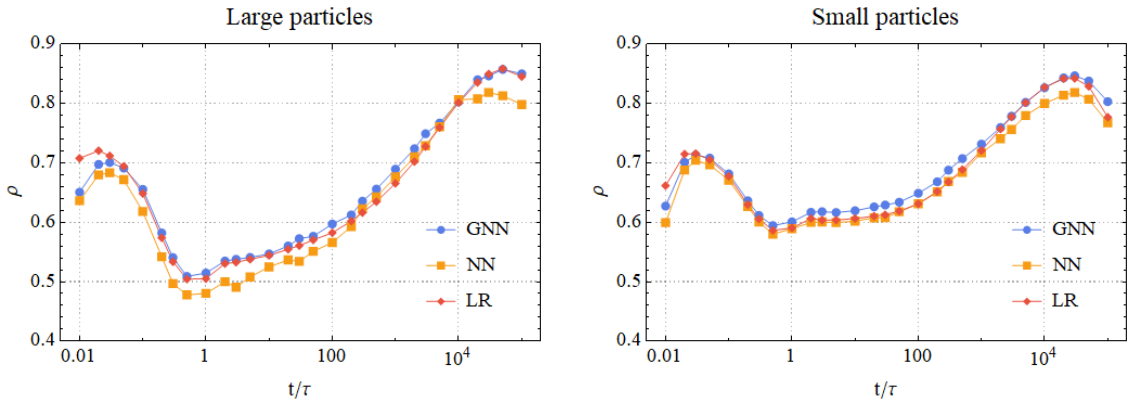


Figure 5.5: Comparison of the accuracy of the dynamic propensity prediction over time as obtained with LR, NN and GNN. For each point in time we use for each method the set of hyperparameters that resulted in the best performance.

If we look at how easy it is to train the algorithm, neural networks score less than the other two methods. As we saw before, the method is very sensitive to its hyperparameters and needs more data than the other two methods to obtain robust predictions. This difficulty in training is likely also responsible for the fact that NNs show the worst accuracy for almost every point in time in Figure 5.5.

As we saw earlier, GNNs are less sensitive to their hyperparameters than NNs, and converge more easily. Moreover, during intermediate times in the caging regime and the beginning of the diffusive regime GNNs slightly outperform LR. This implies that the averaging that takes place in a GNN provides the network with slightly different information than the averaged parameters of the first and second generation. However, the fact that the difference is only slight, underlines the earlier conclusion that the higher generations of order parameters are an excellent approximation of the averaging behaviour that takes place in the GNN. We already knew from Ref. [35] that linear regression with higher order generations performed almost equally well as a GNN when tested on a Lennard-Jones mixture. However, this was in comparison with the GNN from Ref. [36] that only used distances between particles as an input. Now we can conclude that even if the GNN is provided with more structural information, the difference between linear regression and GNNs is minimal.

Although in terms of accuracy the GNN is thus slightly better than linear regression, if we take into account how long it takes to train the model, linear regression is the favoured method. The total training time required to obtain Figure 5.5 for the GNN was multiple weeks: even training the model for just one set of hyperparameters could take from several days up to a week on a high performance work station, making use of several NVIDIA GPUs (GeForce RTX 2080Ti). In contrast, LR only required a total training time of about one hour on a simple laptop.

Moreover, due to its simple structure, linear regression is easier to interpret compared to the GNN. Where the GNN consists of multiple NNs, the relation between input and output for LR consists of just one vector of weights and biases. Therefore it is significantly easier to interpret which parts of the structure are important for the dynamics when LR is used, compared to GNN. We conclude that, given the discussion above, linear regression is the preferred method; it is fast, robust, easy to interpret and, most importantly, provides accurate predictions.

5.4 Further research in this thesis

In this chapter, following up on the research of Ref. [35], we showed that the set of order parameters defined in Chapter 2.2 is a very good predictor for the dynamical heterogeneity at times close to the relaxation time.

Although the predictive power of the order parameters indicates that there are qualitative differences between the structure of fast and slow moving areas, in this thesis we do not explore the nature of these differences. The main reason for this, is that we first want to improve the propensity prediction as much as possible, before analyzing what the prediction can teach us about the structural aspects underlying dynamical heterogeneity.

Although for future research it would be interesting to look at the qualitative differences between the order parameters of fast and slow moving areas, it is good to keep in mind that solely the fact that we can distinguish areas based on the structural parameters, does not necessarily mean that these order parameters provide a good representation of the essential differences in structure. The structural parameters that we use are based on spherical harmonics, meaning that they are focused on capturing the symmetry instead of asymmetry or anisotropy of the structure. As we will see in Chapter 8 the anisotropy of the structure is important for at least part of the systems evolution. Therefore it could be that, even though the order parameters are be able to distinguish fast and slow moving areas, they are not actually capturing the aspects of the structure that form the root cause of the dynamical heterogeneity [3]. Perhaps we are still only looking at, what you could call, indirect 'proxies' for some unknown structural features that are not explicitly captured by our order parameters, but that provide a more direct route towards understanding dynamical heterogeneity.

In the remaining chapters we examine various methods that potentially can improve the propensity prediction even further and discuss the implications of these improvements.

[3] To explain what we mean with this, we turn to a machine learning example, where an algorithm was trained to determine whether a photograph contained an animal or not [82]. After training algorithm reached a very accuracy, however, it turned out that it had on whether or not the background of the picture was blurry, instead of whether the picture contained an animal (in many animal pictures the camera is focused on an animal nearby, meaning that the background becomes blurry). This means that although the algorithm was able to make good predictions, it did not capture the essence of the difference between a photo with and without an animal. Although the above story is an extreme case, it could be that also our parameters do not capture the aspects of the structure that we are interested in.

Chapter 6

Improving propensity prediction

In the previous chapter, we concluded that linear regression is the preferred machine learning algorithm to make propensity predictions for a glassy system. Our next aim is to improve the LR performance. In this chapter we try to do so by providing the algorithm with new structural information in various ways. The reason that we choose to add new information, rather than try to extract more dynamical information out of the current order parameters is as follows: in the previous chapter we saw that three significantly different methods perform about equally well, something that strongly suggests that almost all useful dynamical information that is encoded in the current order parameters is already successfully extracted by the algorithms. As a result, to improve the prediction even further, presumably new structural information must be added.
Before we examine he different approaches to improve the predictions, we first briefly discuss what the current prediction looks like and the times for which it could be improved the most.

6.1 An upper bound to the prediction accuracy

To see where the current propensity predictions could be improved the most, we compare them to a theoretical upper bound. The fact that there is a theoretical limit to the accuracy of the propensity prediction, comes from the fact that the propensity is measured by averaging over a finite number of trajectories. This means that the propensity measurement can only be an approximation of the true propensity, which would be the propensity obtained by averaging over an infinite number of trajectories. As a result, even a perfectly accurate model would not be result in a Pearson's correlation of 1, because we compare this prediction to test data that only contains an approximation of the true propensity.

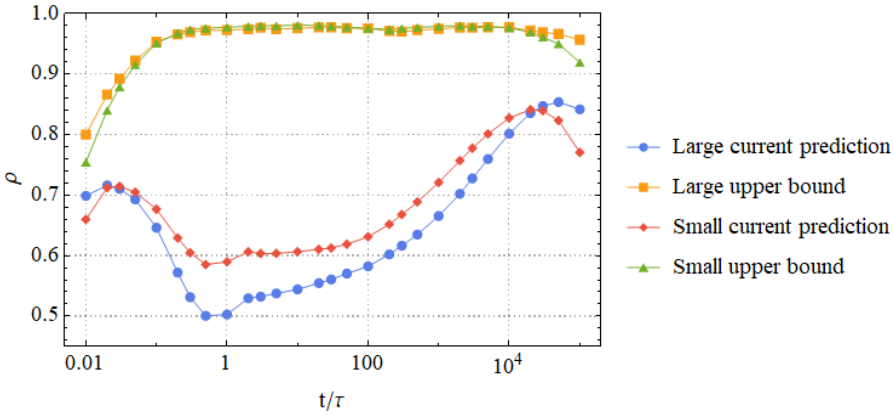


Figure 6.1: The current state-of-the art performance of the propensity prediction performed by linear regression in terms of the Pearson coefficient, together with the theoretical upper bound to the propensity prediction.

To find the upper bound of the correlation one needs to find the correlation between the measured propensity based on 50 trajectories and the true propensity based on an infinite number of trajectories. Since measuring an infinite number of trajectories is not possible, we instead measure the correlation between the propensity based on 50 snapshots and the propensity based on a separate set of n trajectories, a method proposed by Ref. [36]. We consider various values of n in the range $60 \leq n \leq 170$. By extrapolating the graph to $n \to \infty$ we can then estimate the maximal correlation that we theoretically could obtain. In Figure 6.1 we show the best performance fit for both small and large particles together with the current performance as found with linear regression. From this figure we conclude that trying to further improve the predictions for times close to zero is not very useful; the optimal Pearson coefficient is already quite close to the Pearson coefficient of our model. The fact that the upper bound of the Pearson correlation is quite low for these times is expected: since at very short timescales the particles move in random directions, the variation in the true propensity (where we average over an infinite amount of trajectories) will be very low, since the particle environments do not significantly impact their mobility. As the measured propensity only takes into account a finite number of trajectories, statistical errors will start to dominate the measurement.

As the system approaches the caging regime (at $t/\tau \approx 0.1$) the upper bound of the prediction increases sharply. At this point, the specific cage structures start to have an influence on the propensity, and hence the variation in the true propensity becomes large enough to dominate the statistical error. The fact that the upper bound increases, while the linear regression performance decreases, means that our current order parameters are lacking sufficient information to capture the dynamical behaviour during these times. Finally we see that for large times our prediction again comes closer to the upper bound. This means that for times after the caging regime, when the system starts to enter the diffusive regime, linear regression is able to extract a substantial amount of information about the dynamics from the current order parameters.

We can thus conclude that the dynamical information that lies embedded in the current structural parameters is mostly associated with the average mobility of particles in the diffusive state, rather than with the mobility of particles at earlier times such as the caging regime. It would thus be interesting to explore whether adding structural information can improve the predictions during the caging regime.

6.2 Improving propensity predictions

Here we present three different methods that we developed to improve the linear regression prediction. In all three cases we modify or expand the vector of order parameters per particle that we use as input for LR. In Appendix C.1 we discuss one extra method for improving the predictions, by fitting the measured propensity data as a polynomial function of the linear regression prediction. However, although this method slightly improves the predictions, it does not add any physical understanding to dynamical heterogeneity, and therefore we omit it from this Chapter.

6.2.1 Local variation in the structural order parameters

The first piece of additional structural information that we will consider is the spread in the structural parameters of neighbouring particles. Currently the higher generations of order parameters only hold information about the average value of the order parameters of neighbouring particles. However, one could imagine that the variation in the order parameters also influences the propensity prediction; for example, one could imagine that a particle is more likely to escape its cage if it has neighbors in both a low-density and a high-density environment, instead of a set of neighbors that all have an environment of approximately the same average density.

To quantify this variation in the structural parameters of neighbouring particles, we calculate the associated standard deviation. Since the average of the higher-order generations is a weighted average, see Equation (2.7), we also need a weighted standard deviation. For particle i and

structure parameter generation n we define this weighted standard deviation as

$$\mathrm{SD}_i^{(n)} = \sqrt{\frac{\sum_{j:r_{ij}<r_c} e^{-r_{ij}/r_c}\left(x_j^{(n-1)} - \bar{x}_i^{(n)}\right)^2}{\frac{(M-1)}{M}\sum_{j:r_{ij}<r_c} e^{-r_{ij}/r_c}}}, \tag{6.1}$$

where M is the number of particles with $r_{ij} < r_c$ and $\bar{x}_i^{(n)}$ is the weighted average given by Equation (2.7). In Section 6.3 we will present the effect of adding this spread to the LR algorithm.

6.2.2 Quenching the initial configuration

The second strategy we explore is giving the regression algorithm information about the inherent structure of the snapshot. This is the the structure associated with the local potential energy minimum [83–85]. Normally, the snapshots that we use as input are instantaneous configurations taken from a dynamical system. Hence, the particle positions and thus the structural parameters are influenced by thermal fluctuations. In some systems, such as patchy particles, it has been shown that eliminating this thermal noise by quenching the system improves machine learning predictions at longer time scales [86]. Note that any information about the inherent state of a system is still regarded as structural information; after all, an inherent state can be obtained from a snapshot without knowing anything about the dynamics of the system.

Normally one obtains the inherent state of a system simply by quenching it to a local potential energy minimum. In the case of hard spheres however, this not directly possible due to the discontinuous potential; as long as particles do not overlap the system will always have zero potential energy. In order to quench a hard-sphere system, one therefore has to use an effective potential. Arceri and Corwin showed in Ref. [63] that thermal hard spheres close to jamming can be effectively modelled using the following effective potential

$$V(h) = -k_B T \log(h), \tag{6.2}$$

where h is the surface-to-surface distance given by $h = |\mathbf{r}_{ij}| - \sigma_{ij}$.

Quenching simulation

To quench the system we perform a molecular dynamics (MD) simulation (see Section 3.3) on a system of particles interacting via the potential described in Equation 6.2. Efficient quenching is obtained by using the FIRE algorithm described in Ref. [87]. This is an optimization algorithm that is used for structural relaxation. The algorithm is explained in the paper using the analogy of a blind skier trying to ski down a hill as fast as possible. The easiest algorithm, namely letting the skier always go down into the direction of steepest descent, leads to a method that is very sensitive to becoming stuck in local minima. The FIRE algorithm uses the fact that information about the direction in the past can be used as an average predictor for a future descent direction. It therefore suggests that the skier only introduces a small acceleration in the direction of steepest descent, and mostly keeps moving in the direction it came from. Only when this leads to the skier going uphill again, should the skier stop and start descending in the direction of steepest descent again. In the paper the authors show that following this algorithm leads to a surprisingly fast descent.

During the quench, we carry out the following procedure for each time step Δt. In the procedure several parameters are used, for which Ref. [87] proposes the following system-independent values: $N_{\min} = 5$, $f_{\mathrm{inc}} = 1.1$, $f_{\mathrm{dec}} = 0.5$, $\alpha_{\mathrm{start}} = 0.1$ and $f_\alpha = 0.99$.

1. Use MD[1] to calculate the positions $\mathbf{x}$, the force $\mathbf{F}$ and the velocity $\mathbf{v}$.

[1] Note that in the update scheme of MD dynamics, we set the mass m of all particles equal. We follow Ref. [87] here. In this paper the authors recommend equal masses, since the algorithm requires the velocities to be more or less on the same scale.

2. Calculate $P = \mathbf{F} \cdot \mathbf{v}$.

3. A small acceleration in the direction of the force, proportional to the current velocity is introduced. The new velocity becomes $\mathbf{v} \to (1-\alpha)\mathbf{v} + \alpha\hat{\mathbf{F}}|\mathbf{v}|$, where . The value of the parameter α will be discussed below.

4. If $P > 0$, i.e. the system still goes 'downhill' and P has not been negative for $N_{\min}$ steps, we can assume that the system is in an area where the the potential is relatively smooth. This means that we can increase $\Delta t \to \min(\Delta t f_{\text{inc}}, \Delta t_{\max})$, such that the simulation does not take too long. At the same time we decrease $\alpha \to \alpha f_\alpha$ such that small bumps influence the simulation less.

5. If $P \leq 0$ the particles are moving perpendicular or even opposite to the direction of the force. In that case we set the velocity to zero, put α back to its initial value $\alpha \to \alpha_{\text{start}}$, such that we start moving along the gradient of the force again and decrease $\Delta t \to \Delta t f_{\text{dec}}$.

6. Go back to step 1.

The only system-dependent parameters are therefore Δt and $\Delta t_{\max}$. Since in our system initially particles can be close to each other, choosing a Δt that is too large can lead to overlaps. We therefore start at $\Delta t/\tau = 0.001$ (with τ an arbitrary time scaling unit) and lower it by a factor 10 each time we encounter overlap, after which we restart the simulation. The value of $\Delta t_{\max}$ is then set to $\Delta t_{\max}/\tau = 0.01$.

Convergence

It can be a challenge to find an exact criterion for when the system is quenched. A simple solution is therefore to let the system run for an extended period of time, in our case $t/\tau = 10$, and then analyze whether the potential energy as a function of time has reached a constant value. In Figure 6.2 we show the potential energy as a function of time for 10 different runs. As one can see at $t/\tau = 10$ the system is converged sufficiently.

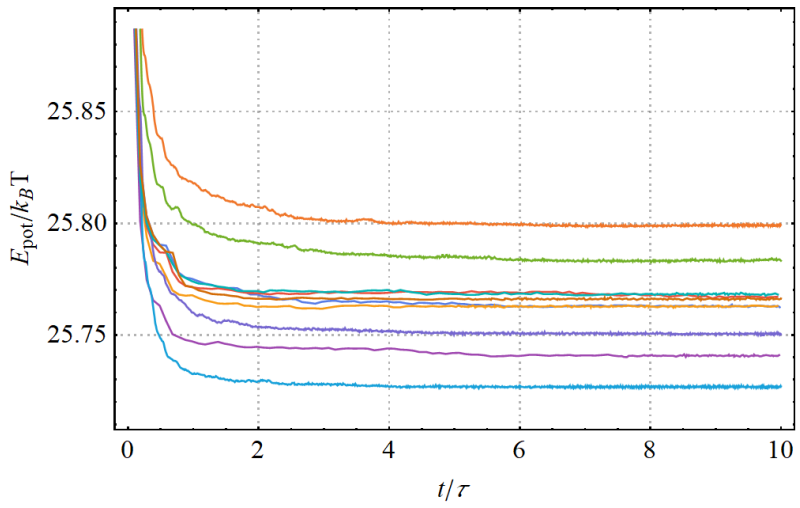


Figure 6.2: Potential energy over time for ten different binary hard-sphere systems at $\eta = 0.58$ with an extra potential as given by Equation (6.2). Time is measured in arbitrary units τ

Structural parameters obtained from quenching

From the quenched structure we obtain several parameters that will serve as an input for the linear regression. First of all we use the quenched coordinates to calculate a new set of structural parameters analogous to our normal ones. Using these parameters will tell us whether or not

removing noise from the system structure improves the propensity prediction. In addition to this, for each particle we define a new structural parameter d_i^{quench} as the absolute distance the particle has moved during the quench.

6.2.3 Using predictions from earlier times

The final method that we use to try to improve the propensity is providing linear regression with information about predictions made at earlier times. This means that for time t our propensity prediction is not only based on the structural parameters, but also on the propensity predictions at the time t_{prev} of the previous propensity prediction. Since the movement of neighbours will influence the movement of a particle, one can imagine that providing this information will improve the prediction. Note that providing linear regression with this information is somewhat reminiscent of the GNN, where neighbouring particles could influence each others predictions. The difference is that the influence is no longer simultaneous.

As with the other recursive structural parameters, when including higher generations of input parameters, we average the propensity predictions at t_{prev} over neighboring shells of particles. Specifically, the zeroth generation contains the propensity predictions at t_{prev} of the particle itself. Higher order generations are then obtained by averaging over neighbouring particles via the following equation, where n reflects the generation we are considering

$$\Delta r_i^n(t_{\text{prev}}) = \frac{\sum_{j:r_{ij}<r_{\text{cut}}} \exp(-r_{ij}/r_{\text{cut}})\Delta r_j^{n-1}(_{\text{prev}})}{\sum_{j:r_{ij}<r_{\text{cut}}} \exp(-r_{ij}/r_{\text{cut}})}. \tag{6.3}$$

In this equation $\Delta r_i^n(t-1)$ contains the n^{th} generation of average propensity. We again set r_{cut} to 2.1.

6.3 Results improving propensity prediction linear regression

In Figure 6.3 we show how the different methods discussed in the previous sections influence the propensity prediction made by linear regression. As mentioned in Section 4.1, we will use r^2 instead of the Pearson coefficient as the parameter that analyzes the accuracy of the prediction. Note furthermore that for the structural parameters, as well as for the previous propensity parameters, we always provide linear regression with up to the second generation of parameters, since this results in the best predictions. In these figures, the dark blue line represents the baseline plot, i.e. it reflects the accuracy of a model based on only the normal order parameters. In the following, we discuss for each method separately, how its performance relates to this baseline plot.

Role of the local variation in structural parameters

As we can see from the orange line in Figure 6.3, adding information about the standard deviation of the structural parameters does not improve the prediction. Moreover, for most times it even slightly worsens the prediction. This can be explained from the fact that adding more parameters makes linear regression more sensitive to overfitting. Although it is tempting to conclude, based on these results, that the spread in structure does not hold information about the propensity, this is not a conclusion we can draw. It could for example be that the variation in the structural parameters holds information about the propensity, but that these aspects are not captured in our structure parameters; or it could be that essential information about the spread is lost while calculating the standard deviation. We can thus only conclude that providing LR with the standard deviation of our structural parameters does not improve the propensity prediction.

Role of the inherent state

As can be clearly seen in Figure 6.3, adding information about the inherent state improves the propensity prediction substantially; the green lines show that adding information about the abso-

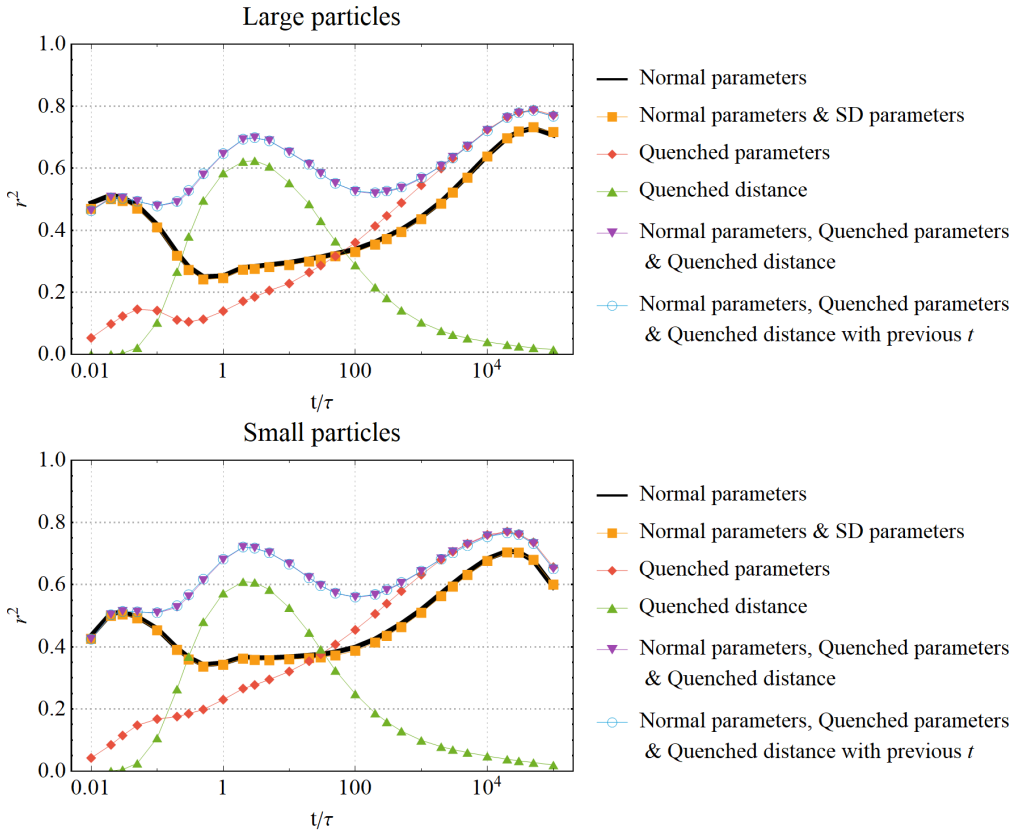


Figure 6.3: Comparison of the various methods to improve propensity predictions described in this chapter. Note that the abbreviation SD stands for standard deviation.

lute distance between particles in both the normal system and the quenched system improves the prediction for times in the caging regime and the red lines show that adding the structural parameters based on the quenched system improves the prediction for larger times. Note that the fact that for smaller times quenched parameters do not improve the prediction is not surprising. For these times the movement of particles will be highly influenced by their initial position, and it is exactly the information about this position that we lose by quenching the system. The most significant improvement is found when we provide linear regression with a combination of normal structural parameters, quenched structural parameters and quenched distance (see purple line in the figures).

The most remarkable finding here is presumably the improvement during the caging regime. From the fact that adding information about the quenched positions improves the prediction, we can conclude that particles in a glassy system must have a well defined caging center. The reasoning is as follows: We already knew that glassy particles move around an equilibrium position in the cage, see Chapter 2, but we now have learned that this equilibrium position is a structural characteristic of the system, rather than something determined by dynamics. Moreover, we now know that this equilibrium position is closely related to the structure of the system in its nearest potential energy minimum.

Role of propensity predictions at previous times

Finally, the light blue lines in Figure 6.3 show that adding information about the predictions for neighbouring particles at previous times, does not improve the propensity prediction; in the figure, the purple and the blue line almost perfectly overlap. Clearly, the average predicted movement of neighbouring particles at a slightly earlier time, does not carry any information that was not already included in the set of structural parameters. This may be partially due to the possibility

that essential information about the movement of individual particles is lost by considering the movement of all neighbouring particles as a whole. We will discuss this subject in somewhat more detail in Chapter 8

6.4 Conclusion

With the improvement of the propensity prediction, new questions and possibilities to look at the dynamics of glassy systems arise. From the strong correlation between the quench distance and the propensity we can conclude that, at least up to a certain degree, cages are associated with structurally determined cage centers. Currently the correlation is not high enough to decisively conclude whether we can speak about a single, well-defined cage center, or whether in different trajectories, particles have different but closely related cage centers.

Chapter 7

Finding the cage location from structure

In the previous chapter, we used a quench to find the inherent state of our system and showed that the positions of particles in this state correlate strongly with the dynamic propensity during the caging regime. Since during the caging regime, particles move on average towards their cage center, the correlation between quenched coordinates and propensity suggests that this cage center is a structural quantity, i.e. that in different trajectories particles move towards the same center. We say 'suggest', because the correlation was not high enough to make definite conclusions: One possibility is that, although from the relatively high correlation between quenched coordinates and propensity we can deduce that the caging centers in different trajectories all lay in a restricted cage *region*, we can not actually speak of a single well-defined structural cage *center*. The other option, is that there is in fact a structurally defined cage center, however its position does not perfectly correlate with the quenched coordinates. This last option raises the question whether there is a better method to find cage centers from the structure of the system than quenching. It is this question that we will address in this chapter.

We start by discussing the shortcomings of quenching, and use that analysis to adopt a new method that can predict the cage centers with more precision. We then use this new approach to examine to what extent the caging behaviour is collective, i.e. at which distance particles still influence each other in the caging regime.

7.1 Improving cage center detection

7.1.1 Shortcomings associated with quenching a system

Assuming that each cage indeed has a well-defined center – independent of the specific trajectory – the most plausible reason that we cannot find this center by quenching the system, is because quenching algorithms are vulnerable to getting stuck in local minima [1]. As a result, it could be that a quenched system does not end up in the true minimum associated with the caging regime, but instead gets stuck in a slightly higher minimum that is separated from the actual minimum by an energy barrier, see Figure 7.1. Under this hypothesis, the cage prediction could be improved by using a method that avoids the use of an energy landscape, and rather searches for the cage centers directly.

In the following section we introduce such a method. Note that this method still uses only structural information to find the cage centers. However, in order to evaluate its performance we

[1] Note that in principle our algorithm can overcome escape from sufficiently shallow local minima. Since we use discrete time intervals Δt to quench the system, it is possible to overcome energy barriers, as long as the width of the barrier is small compared to the distance that can be overcome in Δt. However since Δt is very small, in practice, substantial energy barriers cannot be overcome.

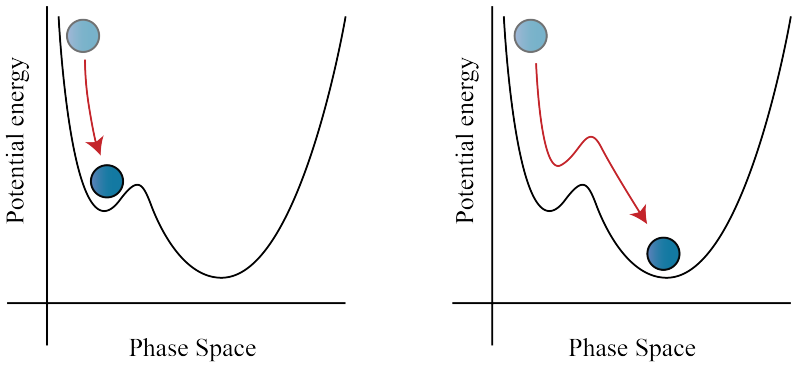


Figure 7.1: Cartoon of a one dimensional particle system getting stuck in a local minimum. On the x-axis we plot the configurational space and on the y-axis the associated potential energy. We assume that the system starts in the upper left corner, represented in the figure as the light blue sphere. Quenching the system from this position would lead to the particle getting stuck in a local minimum, as shown in the left figure. On the right we show the system going to the global minimum.

will no longer use linear regression. Since we discovered that the propensity during the caging regime is linked to the caging center, it is more natural to directly look at the correlation between the propensity and the cage center instead of training a model to do so.

7.1.2 Adopting a new method to find the caging center

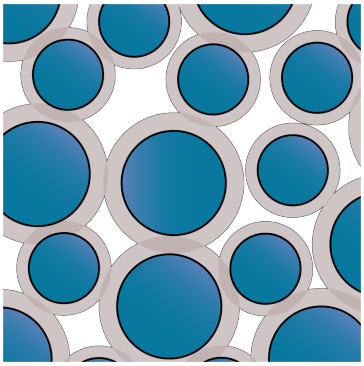


Figure 7.2: Cartoon of the cage restriction in our Monte Carlo simulations. The movements of particles is restricted to a radius r_c (indicated by the grey circle around each particle).

In order to find the cage centers, we would like to construct a simulation which explores the phase space around the initial configuration, while simultaneously ensuring that particles do not escape their cages. This means that we need a method that only allows particles to explore their local cage. Constructing a molecular dynamics simulation to do so is quite challenging. Instead, a more elegant and simple solution is to use a Monte Carlo (MC) simulation (see Chapter 3.5). Assuming that caged particles fluctuate around their cage center, we can identify the cage centers ($\mathbf{r}_i^{\mathrm{MC}}$) by using an MC simulation that measures the ensemble average positions of each particle $\langle \mathbf{r}_i \rangle$, while restricting the movements of particles to their associated cages. Note that by using an MC simulation to measure the caging position we are still only using structural information: doing an MC simulations requires no dynamic information.

In order to prevent particles from leaving their cage, we do not allow particles to move further out than a certain radius r_c from their initial position. Figure 7.2 graphically shows this. During the simulation the center of the particles cannot move further outwards than the grey circle surrounding the particle. In practice this means that every particle experiences an external field on top of its interaction with other particles, which takes the form of

$$\phi(r_0) = \begin{cases} 0 & \text{if } r_0 \leq r_c \\ \infty & \text{if } r_0 > r_c, \end{cases} \tag{7.1}$$

where r_0 is the absolute distance between a particle's current position. To find the value of r_c that leads to the highest correlation with the propensity, we will perform several simulations with

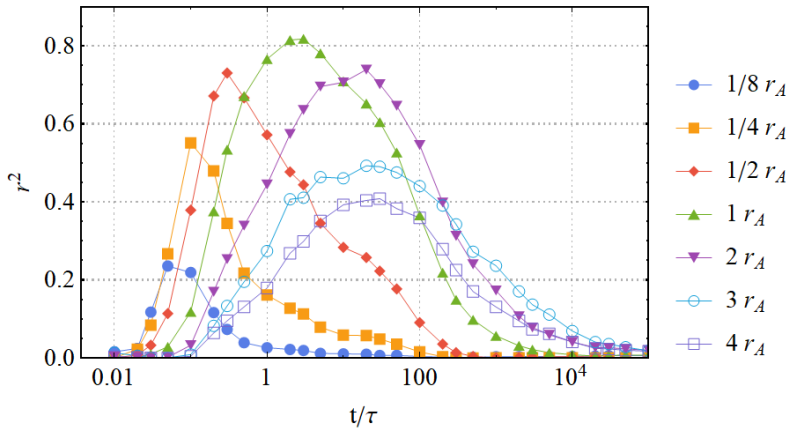


Figure 7.3: Correlation between the propensity and the absolute distance between initial coordinates and ensemble averaged coordinates, $\Delta\mathbf{r}_i^{\mathrm{MC}}$, in a Monte Carlo simulation for 600 large particles. The movement of particles is restricted to different radii ranging from $r_c \in [1/8r_i, 4r_i]$.

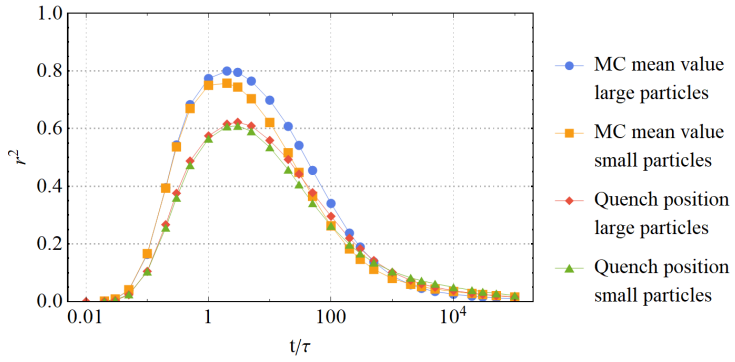
different values for r_c.

In order to measure the average positions of particles in their cage we run a MC simulation of $5{\cdot}10^4$ initial steps, where we adjust the maximum displacement for particles in such a way that the fraction of accepted moves is around 0.3. After these initialisation cycles, we perform a simulation with 10^5 steps, where after every 100 steps the position of each particle is measured. Afterwards, the averaged position of each particle is computed. For the initial configurations we use the same snapshots that we used earlier. We perform several simulations with different values of r_c in the interval $r_c \in [r_i/8, 4r_i]$, where r_i is the radius of the particle under consideration. The reason that we choose to let the value of r_c depend on the particle size, is to make sure that large particles have a relatively larger space to move around than small particles.

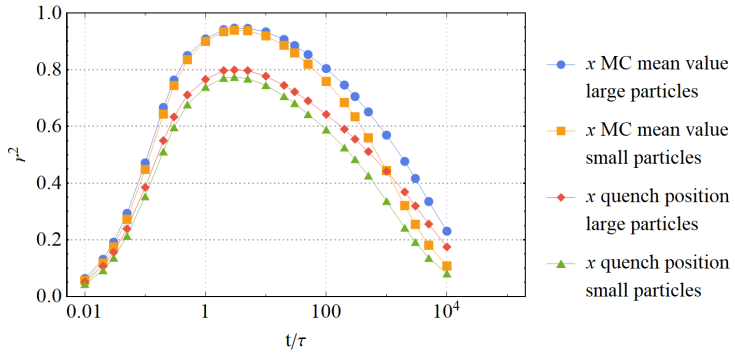
7.1.3 Improved cage center measurement

Before evaluating the correlation between the measured cage centers and the propensity, we first focus on finding the optimal value of r_c. In Figure 7.3 we show the correlations between the dynamic propensity and the absolute distance between the initial coordinates and the ensemble averaged coordinates ($\Delta\mathbf{r}_i^{\mathrm{MC}}$) for different choices of r_c. As we can see, the value of r_c that yields the highest correlation, is when $r_c = r_i$, i.e. when the center of a particle is restricted to its own radius. Additionally, we see that the peak in the correlation moves monotonically to later times as we increase r_c. This result is not very surprising; for low values of r_c particles that start very far from their equilibrium position will not be able to reach their cage position, while for large r_c particles will sometimes be able to move past each other and thus escape their cage. Based on this, we choose to continue with a restriction radii of $r_c = r_i$.

To obtain a clear picture of how well the obtained cage centers correspond to the propensity during caging, in Figure 7.4a we plot the correlations between propensity and the cage centers for both large and small particles, and compare these to the results obtained from quenching. Additionally, in Figure 7.4b, we show the correlation between the x-coordinate of the dynamic propensity vector and the x-coordinate of the found cage centers, i.e. $\Delta\mathbf{r}_i^{\mathrm{MC}} \cdot \hat{\mathbf{x}}$. Since the equivalent correlations between the y- and z-coordinates are completely similar by symmetry, we omit them here.
Note that the fact that the correlation between absolute distances and propensity is lower than the correlation between the dynamic propensity vector and the separate cage center coordinates, probably has to do with the fact that the variance in the values of individual coordinates is higher than the variance in the absolute distance. In the definition of r^2, we divide the MSE by the variance of the data set and then subtract that value from 1 (see Chapter 4). As a result, due to the



(a)



(b)

Figure 7.5: Correlation between the propensity and the absolute distance between the initial positions and the caging centers as found with two different methods (quenching the system and measuring the ensemble average of a MC simulation, in which the movements particles is restricted to a distance equal to their own radius). The top figure contains the correlation between the propensity and the absolute distance between caging coordinates and initial position. The lower figure contains the correlation between the x coordinate of the dynamic propensity vector (as discussed in Section 2.1.2) and the x-coordinate of the caging centers with respect to the initial position. The data sets contained $14 \cdot 10^4$ small particles and $6 \cdot 10^4$ large particles.

larger variance in individual coordinates, we expect the correlation between vector propensity and cage coordinates to be higher than the correlation between the absolute propensity and $|\Delta \mathbf{r}_i^{\mathrm{MC}}|$, even when the MSE is about the same the two groups.

From the results presented in Figure 7.5 we can first of all conclude that the ensemble average coordinates have a significantly higher correlation with the propensity than the quenched coordinates. This strongly suggests that our hypothesis that quenching the system sometimes leads to (part of) the system getting stuck in a local minimum was true. We furthermore see that the correlation between the ensemble averaged coordinates and the dynamic propensity vector reaches a value very close to 1 during the caging regime. This clearly shows that there is a well-defined cage center for each particle. Moreover we have now developed a method that can find this cage center with high accuracy.

7.1.4 Propensity prediction using new cage centers

In the previous Chapter, we saw that using quenched coordinates for our propensity predictions led to an improved accuracy. It is then natural to ask whether our new cage center measurements

lead to a further improvement in our propensity predictions. To test this, we use the new cage coordinates, found by the Monte Carlo simulation, to evaluate the structural parameters described in Chapter 2.2 and then train a linear regression model to predict the propensity based on a combination of the cage center parameters, the initial position parameters and the values of $|\Delta \mathbf{r}_i^{\mathrm{MC}}|$. We show the results in Figure 7.6. As a comparison, we also plot the prediction based on the cage center as found by the quench, and the prediction based on solely initial position parameters. From this figure we see that the predictions based on Monte Carlo coordinates result in a significant better prediction, not only compared to the previous-state-of-the-art prediction, but also compared to the prediction based on the quenched system. The improvement of Monte Carlo coordinates with respect to quenched coordinates is primarily visible for times during the caging regime. At times around the relaxation time, we see that the two predictions perform about equally well. This is not surprising: the difference between the cage coordinates found by the two methods will have the largest influence at times where we actually want to predict the center. At later times, when the propensity prediction is mainly based on the structural parameters, the influence of the relatively small difference in cage coordinates will decrease.

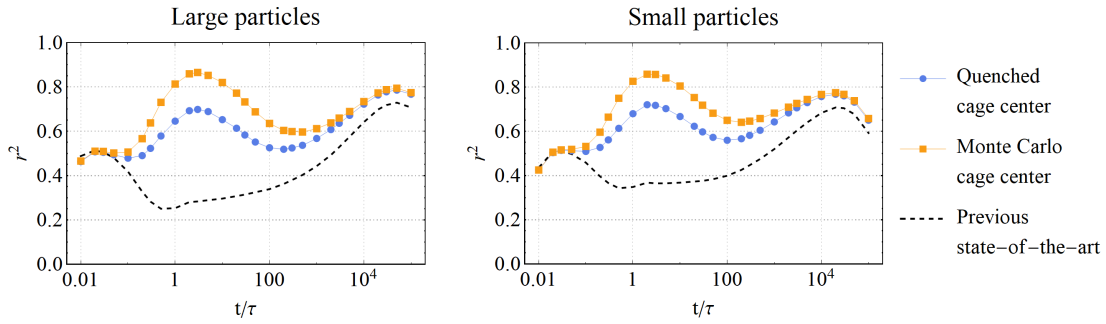


Figure 7.6: Propensity prediction made with a trained linear regression model, based on three different sets of input parameters. The black lines uses the structural parameters discussed in Chapter 2.2 based on initial coordinates. The blue lines use the structural parameters based on both the normal coordinates and the quenched coordinates, as well as the absolute distance between quenched and initial coordinates $|\Delta \mathbf{r}^{\mathrm{quench}}|$. The orange lines use the structural parameters based on both the normal coordinates and the ensemble average coordinates, as well as the absolute distance between ensemble averaged and initial coordinates $|\Delta \mathbf{r}^{\mathrm{MC}}|$. In all cases, we use structural parameters up to the third generation.

7.2 Collective effects in determining the cage position

These MC simulation also allow us to examine more closely the extent to which particles influence each other during the caging regime, i.e. up to what distance particles affect the location of each others caging center. We already know that caging must be a collective phenomenon: after all, the movement of neighbouring particles influences the shape of a particles cage and thus the position of the cage center. To examine the extent of this collective behaviour, we perform a simulation where we measure a particle's caging position while part of the system remains stationary, or 'frozen'; only particles within a certain radius of the reference particle are allowed to move. This situation is schematically shown in Figure 7.7. In this setup we measure the average position of the yellow particle, while only allowing the dark blue particles, that lie within a radius r_{freeze}, to move. By varying r_{freeze}, and comparing the found cage center with the actual cage center, we can determine the distance over which particles still influence each other's cage.

In Figure 7.8 we show the correlations between the propensity and the distance between the initial position of particles and the ensemble average position for different choices of r_{freeze}, as well as the correlation for the unfrozen system as a reference. When we compare the partially frozen system with the unfrozen system, we see that the correlations occurring in the partially frozen system

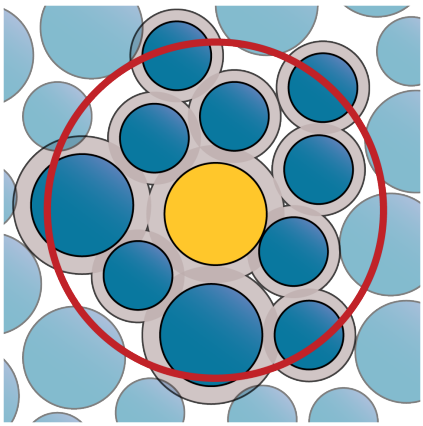


Figure 7.7: Cartoon of the Monte Carlo simulation of a partially frozen system. Only particles that lie within the red circle as seen from the yellow reference particle can move. The movement of the non-frozen particles is still restricted to r_c

are significantly lower than in the non-frozen system. Even in the system where particles up to a distance of $4\sigma_A$ from the reference particle were allowed to move, we still see a significant drop in correlation. From this we can conclude that caging is a highly collective process that spans over many neighbor shells. In other words, predicting the exact caging position of a particle requires information from particles at very large distances. Although we expected caging to be a somewhat collective phenomenon, the fact that the influence of particles on the caging characteristics reaches this far is quite remarkable.

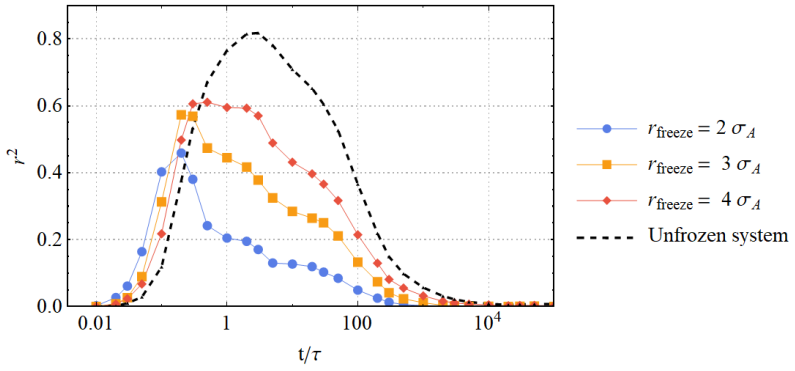


Figure 7.8: Correlation between the propensity and the absolute distance between ensemble averaged coordinates in a Monte Carlo simulation for 600 large particles. The movement of particles is restricted to their own radius. During each simulation, only particles within a certain radius r_{freeze} of the considered particle were allowed to move.

Although it falls outside of the scope of this chapter, we also examined whether information about the spread in the fluctuations of particles during the MC simulation was of influence on the propensity predictions. It turned out that this information did not significantly improve the predictions, however the analysis and the results are still included in Appendix C.2

7.3 Conclusion

In this chapter it was shown that the cage centers that particles move on average to during cage, are a quantity that is mainly determined by structure. Moreover, we showed that we can predict the cage center of a particle, by measuring its ensemble averaged coordinates in an MC simulation

where the movement of a particle's center is restricted to remain within a radius r_c from its initial position. We found that the optimal choice of r_c is close to the actual radius of the particle. Furthermore, we also learned that the position of the cage center is determined by a highly collective process, i.e. the cage position of a particle is influenced by particles over a large volume of space.

The fact that we found a very strong correlation between the individual spatial components of the propensity and those of the displacement from the cage center, suggests that directionality plays an important role in fully understanding the dynamics. In the next chapter, we will explore this in more detail.

Chapter 8

Importance of direction in dynamics

In the previous chapter we developed a method that can find a particle's cage center with high accuracy. The 3D position of this cage center turned out to be an excellent predictor for the average dynamic behavior during the caging regime. This result strongly suggests that at least during the caging regime, directionality is an important factor for the dynamics of the system. In this chapter we take a closer look at directionality for both the structure as well as the dynamics of our glassy system.

We start by discussing the role of structural anisotropy in the current order parameters. Thereafter, by making use of the in Chapter 2 introduced dynamical self-alignment parameter (DSP), we examine whether information about structural anisotropy is also important for times after the caging regime.

8.1 Absence of directional information in our structural order parameters

When considering the importance of directionality, it is good to first consider why directional effects are not (fully) captured in our basic selection of the structural order parameters described in Chapter 2.2. The main reason behind this, is that ignoring information about the anisotropy of the system significantly simplifies the training of machine learning algorithms; if two configurations are the same apart from their orientation, we want the machine learning algorithm to make the same propensity prediction. However, in the process of making parameters and algorithms rotationally invariant, information will inevitably be lost, and sometimes, like in the case of predicting the cage center, it turns out that this information is essential for the prediction.

The first instance where information about the anisotropy is lost, is in the formulation of the structural parameters, i.e. the radial density and the bond order parameters. Clearly, the radial density averages out all directionality by design. In the case of bond order parameters, we express the structure of different shells in terms of rotationally invariant quantities associated with spherical harmonics. Although these do capture some information about the anisotropy of a the particle's environment, their rotational invariance implies that we lose information about how the particles in different shells are located with respect to each other, as well as with respect to the reference particle.

Information is also lost in the various averaging processes that we encounter before and while applying the machine learning algorithm. One example is the construction of the higher-order generations of structural parameters. These parameters are averaged over all nearest neighbours within a certain radial distance. As a result we lose information about where different structures are located. Analogously, information is also lost in the averaging processes that take place in the

edge and node updates of the GNN.

8.2 Anisotropy of the propensity beyond the caging regime

As we have already seen, during caging the movement of particles is highly anisotropic: we found that particles preferentially move towards their cage center. However, we cannot conclude a priori that also for later times particles show anisotropic movements. To examine this, we examine to what extent particles in different trajectories move in the same direction using the 'dynamical self-alignment parameter' (DSP) that was introduced in Chapter 2.1.2. The DSP is an order parameter that reflects how much the average displacement of a particle is aligned when comparing different trajectories at a given time interval; if all trajectories move in exactly the same direction the DSP has a value of 1, while lower values indicate more variation in the direction of movement. Since thermal fluctuations are averaged out by considering multiple trajectories, all directional preference that we observe in the DSP must have a structural origin.

Instead of measuring the DSP with respect to the initial positions, we measure it with respect to the cage position as found with the Monte Carlo simulations described in Chapter 7. Not only does this remove the influence of thermal fluctuations in the initial configuration on the analysis, it also allows us to look at the escape direction of particles with respect to the cage centers.

In Figure 8.1 we plot the DSP of both large and small particles as a function of time. As a baseline, we also plot the DSP results that one would find if particles moved in a completely random direction (using the same number of trajectories as we use for our propensity data). As can be seen in the graph, this baseline has a non-zero value. Since the baseline DSP is obtained by averaging over a finite number of trajectories, even if the vectors point in random directions the sum of all these vectors will never be perfectly zero.

In Figure 8.1 we see that for short times the DSP of both large and small particles is very close to one. This is expected since at $t/\tau = 0$ particles always start at the same initial positions. As a result, for short times (where the dynamical fluctuations are still small) the vector between a particle's position and its cage center is essentially given by the fixed vector between the initial position and this cage center for every trajectory (hence the DSP value of one). Once the particle has reached its cage center, the DSP reflects motions that are nearly purely random, reaching values close to the baseline. Again this is something we expect: during caging particles randomly fluctuate around their cage centers.

The interesting part of the graph is found at times $t \gtrsim 5$, where cage escape starts to become likely. Here we see that the DSP increases again, resulting in a significant deviation from the baseline. From this we can conclude that even after the caging regime the movement of particles has a directional preference that originates from structure. From this figure it is not yet possible to extract the processes that underlie this directional preference; it could either be the escaping of particles from their cage and/or collective motion (drift) that shifts the cage positions. In the next section we will examine the influence of both of these processes on the anisotropy of the propensity.

Origin of the directional preference

To examine whether drift and/or cage escape lead to an anisotropy of the propensity, we consider the dynamics of two different groups of particles: one group consists of particles that escape their cages at a very early point in time and one group consists of particles that stay caged for a significant period of time. For this last group, we know that possible directional preference at intermediate times must primarily be caused by drift. For the group that consists of early escaped particles however, the cage escape process may play a more important role in the development of directional preference at intermediate times (although we will also investigate the influence of drift for this group). We set up the two groups by selecting the 25 small and large particles in a system that at $t/\tau = 500$ have moved respectively the furthest and the least with respect to their

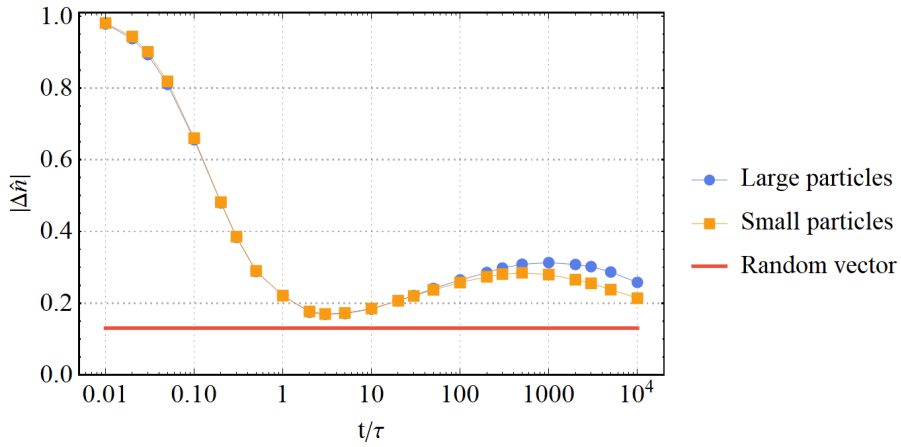


Figure 8.1: Dynamical self-alignment parameter $\Delta|\bar{n}|$ over time, for both large and small particles averaged from 50 trajectories. The red line indicates the baseline, obtained by evaluating the DSP for 50 random vectors.

caging center (where we look at 50 systems in total). Note that although we will call the particles that have travelled the most 'fast' and the particles that have stayed close to their starting point 'slow', this is not be interpreted as a reference to their velocity but rather based on their average displacement at $t/\tau = 500$.

To confirm that at times following the caging regime, the two particle groups indeed consist of either escaped or not escaped particles, we compare their associated average propensities at $t/\tau = 500$ with the average cage size. We can make a rough estimation of this average cage size by assuming that cages are spherical and have a radius r_{cage} (which on average will be a good assumption). Assuming that particles move freely inside their spherical cage for an extended period of time, the average propensity $\langle \Delta r_{\text{cage}} \rangle$ in the cage will be given by

$$\langle \Delta r_{\text{cage}} \rangle = \frac{\int_0^{r_{\text{cage}}} \int_0^{2\pi} \int_0^{\pi} r\, r^2 \sin\theta dr d\phi d\theta}{\int_0^{r_{\text{cage}}} \int_0^{2\pi} \int_0^{\pi} r^2 \sin\theta dr d\phi d\theta} = \frac{3}{4} r_{\text{cage}}. \tag{8.1}$$

Since we can measure the average propensity of particles in their cage, the expression above thus gives us an estimate of the value of r_{cage}. In this case we obtain the averaged propensity in the cage by looking at the distance between initial and cage coordinates[1], which turns out to lie around $0.1\sigma_A$ for both small and large particles. As a result, the average cage size is given by $r_{\text{cage}} = 4/3 \langle \Delta r_{\text{cage}} \rangle \approx 0.14\sigma_A$.

We compare the average propensity of our two particle groups with the average cage size. At $t/\tau = 500$, both the large and small slow particles have travelled around 0.08 σ_A with respect to their initial position. Given that the average cage size lies around 0.14 σ_A, we can safely assume that most of these particles have not escaped yet (and in fact most of the particle will still be caged at $t/\tau = 10^4$). The fast moving particles, on the other hand have on average moved around 0.5 σ_A (large particles) and 0.8 σ_A (small particles) at $t/\tau = 500$. Since these propensities are significantly higher than the average cage size, we are allowed to assume that most of these particles have already escaped their cage at $t/\tau = 500$.

In Figure 8.2a we plot the average DSP for the 25 fastest and slowest particles at $t/\tau = 500$ for a total of 50 different systems. As we can see, both the slow and fast particle groups show directional preference in their movements. For the slowest moving particles we see that after $t/\tau = 10$ there is a small but steady increase in the average DSP. The fact that the movement of particles that

[1]Since the initial snapshot is taken from a dynamical system, the initial coordinates give a measure of the typical fluctuations of particles in their cage.

have not yet escaped their cage shows a directional preference, strongly suggests that at least some parts of the system show collective drifting motion of particles[2]. In fact this drifting behaviour is something that has already been observed in the measurement of single trajectories of particles in a glassy system. In Figure 8.3 we show a plot of the single-particle displacements of particles in a 2 dimensional binary mixture of Lennard-Jones particles taken from Ref. [2]. The arrows represent the displacement of particles at times comparable to the relaxation time of the system. If we take a close look at this figure we can see that some areas with less mobile particles clearly show collective drifting behaviour. The fact that we also see this drift in averaged dynamical behaviour, implies that at least part of the observed drifting is caused by structural influences and not purely by thermal fluctuations.

Although the above results show that in the case of slow particles the directional preference is most likely caused by drifting, we have not yet examined whether this drifting can also explain the directional preference in the case of fast moving particles. Looking at Figure 8.2a we see that the movement of the fast moving particles results in a peak in the DSP at times around $t/\tau = 100$ to $t/\tau = 1000$. Afterwards the DSP starts to decrease again. The question is whether this peak in the DSP of fast particles is caused by a faster drift, or by cage escape. To answer this question we examine how the drift of fast and slow moving particles compares.

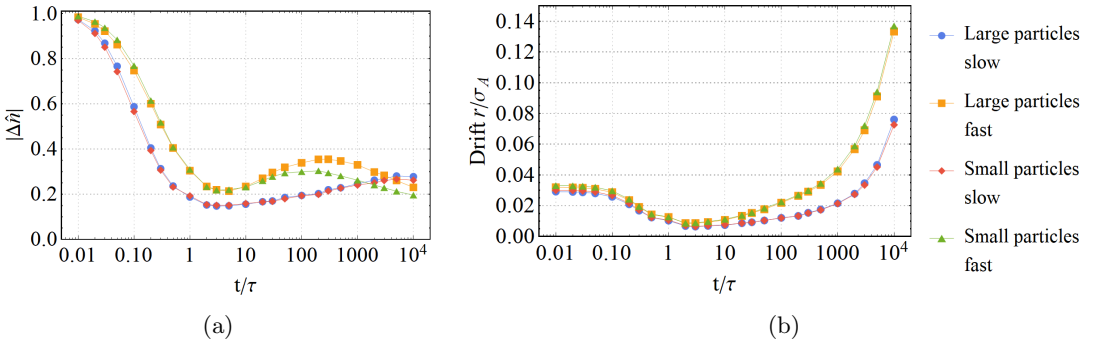


Figure 8.2: Figure (a) contains the dynamical self-alignment parameter over time, averaged over the fastest and slowest moving large and small particles of 50 runs. Figure (b) contains the average drift for the same collection of particles.

To obtain an rough estimate of the drift, we average for each particle the dynamic propensity vectors of all neighbouring particles within a radius of $r = 2.1/\sigma_A$ and then compute the length of this average vector. The more collective motion there is present in the vicinity of a particle, the larger this average vector will be. In Figure 8.2b we show the average drift of the fastest and slowest 25 particles of 50 systems in total. Comparing the drift of the two groups, we see that the fast moving particles are on average located in areas that show stronger collective movement. This holds for both large and small particles. However, the absolute difference in drift between fast and slow moving particles is small and moreover their graphs follow the same trend. This implies that, if the directional preference of fast moving particles would only be caused by collective motion and nothing else, we would expect that the graphs of fast and slow moving particles in Figure 8.2a would follow roughly the same trend too; something that clearly is not the case. Furthermore, we also see that the average drift of fast moving particles at $t/\tau = 500$ lies around $0.04\ \sigma_A$. As we mentioned earlier, most fast moving particles will at that time already have moved around $0.5\ \sigma_A$ (large particles) or $0.8\ \sigma_A$ (small particles). Clearly, the main movement of the fast particles in this time regime cannot be due to collective drift. Hence, the observed directional preference must be caused by particles having a preferred direction for escaping their cage. Note that with this statement we do not imply that particles always escape in the same direction; we only conclude that there exist escape routes that are on average more favourable than others.

[2]Note that the system as a whole cannot drift, since we set the the initial center of mass velocity to zero. It is however possible that smaller areas of particles move collectively.

Moreover, we expect that this average directional preference is the most prominent for particles that escape their cage relatively early: at later times, the environment of particles will start to differ progressively between trajectories. As a result, particles are associated with different cage structures in difference trajectories and thus with different preferential escape routes. Therefore, we expect the anisotropy of the propensity to decrease with increasing time.

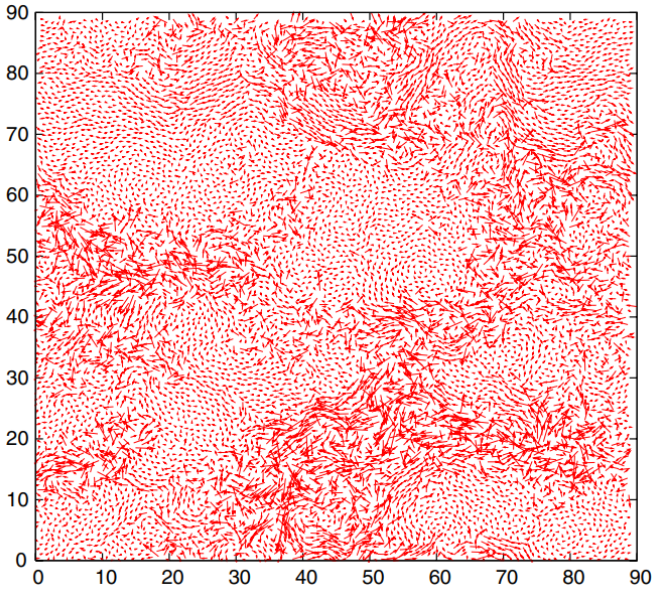


Figure 8.3: Single-particle displacements of particles in a 2 dimensional binary mixture of Lennard-Jones particles taken from Ref. [2]. The arrows represent the displacement of particles at times comparable to the relation of the system.

8.3 Conclusion

From the above we can conclude that particles have at least some tendency to move in a preferred direction, not only during the caging regime, but also at later times. We showed that this directional preference is associated with two processes. Firstly, areas in the system show collective drift, and secondly, the escape of particles from their cage is associated with a directional preference. This naturally leads to the question of whether it would be possible to predict the full dynamic propensity vector using methods similar to the ones we use to predict the magnitude of the propensity. Unfortunately, the answer is that predicting the propensity vector will be very complicated due to multiple reasons.

The first problem one encounters, is that predicting a vector would require a new set of structural descriptors that carry less symmetric information about the particle environments. Ideally we would like to have parameters that capture both the degree of asymmetry of a particle's environment, as well as information about the orientation of this asymmetry. Although we have made some attempts to explore this during this project, we have not yet succeeded in constructing a practically applicable set of such parameters. Moreover, although changing the machine learning algorithms such that they are able to predict vectors instead of dimensionless quantities is possible[3], it makes the algorithm substantially more complex and it requires significantly more training data. Implementing such an algorithm lies outside the scope of this thesis.

[3]See for example Ref. [88], where the authors consider a PointRNN to model the movement of a point cloud.

Finally, the question is whether being able to predict the vector will teach us anything about the physics underlying dynamical heterogeneity. We know that for times around the relaxation time, the rotationally invariant structural parameters that we currently use are very well able to predict whether a particle is located in a fast or slow moving area. At the same time, we see that at times around the relaxation time, the DSP of most fast moving particles goes down again. These two observations lead to the conclusion that directional information is not very important to distinguish whether a particle is located in a fast or slow moving area.

Chapter 9

An outlook on dynamic heterogeneity

In the previous chapter we saw that structure plays a role in determining the preferred direction of movement for particles – even at time scales beyond the caging regime. In this chapter, we take a closer look at the dynamic propensity of particles in relation to their position within the slower and faster rearranging regions in the system. In particular, at times close to the relaxation time, we find a clear division of the system into fast and slow moving regions, which can be predicted from structural observations. It is then interesting to examine how the presence of these regions affect the dynamics of individual particles, both around the relaxation time and at shorter time scales.

The aim of this chapter is not to make quantitative claims, but instead to raise new questions and indicate new directions that can be explored and looked at in future research.

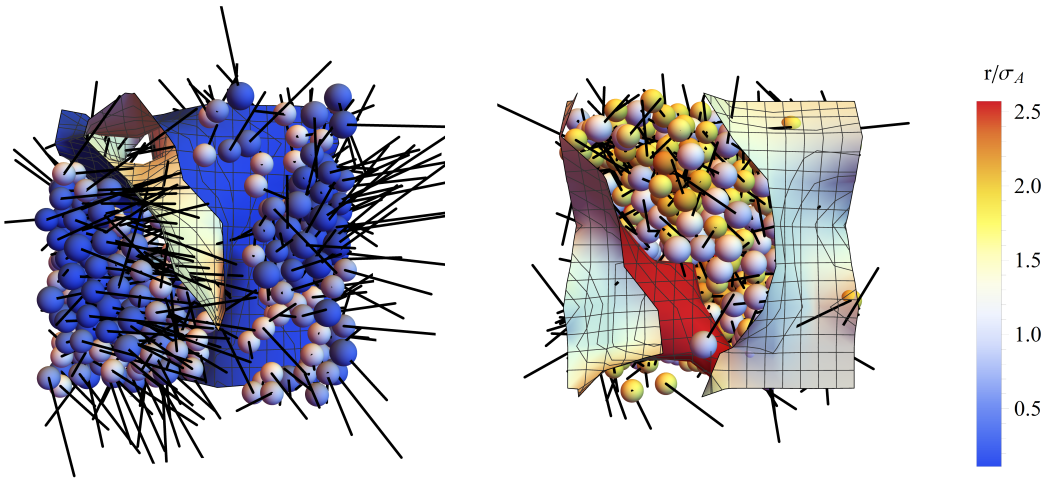
9.1 Particle motion at times close to the relaxation time

In the first part of this chapter we examine the movement of particles that are located at the boundary of fast and slow moving areas. In Figure 8.3, we showed a figure from Ref. [2], which displayed the single trajectory displacements of particles in a 2-dimensional binary Lennard-Jones mixture at times corresponding to approximately the relaxation time of the system. One observation from this figure is that in areas where there is a sharp boundary between fast and slow moving particles, the fast moving particles have a tendency to move parallel to this boundary. In a single trajectory this is something we can understand: because the slow moving areas act roughly as a wall, faster particles are forced to move parallel to it. It is interesting to examine however, whether we see the same behaviour when we consider not a single run, but look at averaged dynamical behaviour.

To examine the directional preference of particles at the boundary, we again consider the dynamical self-alignment parameter (DSP) and dynamic propensity vector. For visualization purposes, we define a vector $\Delta\mathbf{a}_i$, which points in the direction of the dynamical vector propensity, but that is also scaled with the value of the DSP, i.e.

$$\Delta\mathbf{a}_i = \Delta|\bar{n}_i|\Delta\mathbf{r}_i(t), \tag{9.1}$$

where both $\Delta|\bar{n}_i|$ and $\Delta\mathbf{r}_i(t)$ are measured with respect to the cage center in order to reduce the influence of thermal noise. The reason that we scale the length of the vector $\Delta\mathbf{a}_i$ with the value of the DSP is to ensure that drawing these vectors in 3D will clearly show regions of particles that have a preferential direction of motion, even if the distance they move (i.e. their propensity) is very low.



(a) Slow particles (b) Fast particles

Figure 9.1: Two representations of the directional preference of the slowest and fastest third of the particles at $t/\tau = 10^4$. The colour of the particles represents the propensity, while the line segments represent the value of the DSP scaled by a factor of 10. The continuous surfaces are only plotted for visual support.

For visibility purposes, we also draw these boundaries of both the fast and slow moving areas. To define the boundaries, we apply an artificial Gaussian potential on every particle in one the groups, given by:

$$\phi(r) = \begin{cases} 1 & \text{if } r \leq \frac{\sigma}{2} \\ \exp(-2(r/\sigma - \frac{1}{2})) & \text{if } r > \frac{\sigma}{2}, \end{cases} \tag{9.2}$$

with σ the diameter of the particle. The surface that we consider is defined as the equipotential surface

$$\Phi(\mathbf{r}) = \sum_{i=1}^{M} \phi(\mathbf{r}_i - \mathbf{r}) \equiv 0.95, \tag{9.3}$$

where the summation runs over all particles that are part of either the fast or the slow group. For emphasis, this approach gives us two distinct boundaries: one which denotes the edge of the fast region (and hence encloses most of the fast particles), and one which encloses the slow region. Note that the surfaces are purely intended as a rough estimate of the boundary of the fast or slow region that can be used for visualization purposes; the potential and its parameters do not hold any physical meaning.

In Figures 9.1a and 9.1b we show respectively the slow and fast moving particles, their associated $\Delta\mathbf{a}_i$ vectors at $t/\tau = 10^4$ scaled by a factor of 10 to increase visibility (black lines) and the boundary surfaces defined above. In the figure the colours of the particles represent the propensity with respect to the caging position of each particle at $t/\tau = 10^4$. In Figure 9.1a we clearly see the collective motion of the slow particles discussed in Chapter 8: the vectors of neighbouring particles often point in roughly the same direction. Moreover, we see that this collective behaviour is often in a direction perpendicular to the surface, either moving towards or away from it. This is in strong contrast to the behaviour of the fast particles. Although in the case of fast moving particles the DSPs are smaller as indicated by the shorter black lines, we clearly see that at the boundaries, particles move parallel to the drawn surface. Out of all the fast particles near the surface, only very few have a dynamic propensity vector that points through the surface. Although we only show the results for one system and only seen from one viewpoint, this kind of behaviour was seen

in all the runs that we checked. This means that although particles at the boundary of a fast moving area might still move in several different directions, the direction will often be in the plane parallel to the local boundary surface. This is in agreement with the single-trajectory behavior from Ref. [2] (Figure 8.3).

9.2 Evolution of fast and slow moving areas

In the second part of this chapter, instead of focusing on the dynamics at the relaxation time, we look at the dynamics observed at shorter time scales. Specifically, we are interested in whether particles that escape their cages early, are located within the fast moving regions of the system (as defined via their mobility at the relaxation time). In order to examine this, we want to map the location of the particles that on average escape their cages early. We start by defining an escape criterion for such particles.

9.2.1 Definition of an escape criterion

In order to estimate whether a particle has escaped its cage, the most straightforward approach is to simply define an escape distance, such that any particle that has moved more than this distance with respect to its cage center, is considered to have escaped. For this, we make use of the estimate from Chapter 8 for the cage size, i.e. $r_{\text{cage}} = 0.14\sigma_A$. It is reasonable to say that a particle that has moved significantly more than $2r_{\text{cage}}$ away from its initial position has escaped. As a result, we say that a particle has escaped 'early', when its dynamic propensity exceeds $2r_{\text{cage}}$ at a time $t/\tau = 50$. Note that this choice of time is somewhat arbitrary. However, as shown in Figure D.1 in Appendix D, our results are not strongly sensitive to this choice.

We are aware that this escape definition is an approximation, which may not easily extend to long times where the cages of particles in different trajectories start to significantly differ. Moreover, this criterion does not account for drift. As a result, for the times where drift starts to play a large role, the criterion will probably fail. Despite all these flaws, for the small times we consider here, where drift does not play a large role yet (see Chapter 8) and where cages in different trajectories will still be sufficiently similar, the criterion will give a useful indication of the particles that escape the earliest.

9.2.2 Definition of the surface between fast and slow moving areas

We are interested in the location of the early escaped particles with respect to the areas that will later become the fast and slow moving areas. Rather than using two individual boundaries for the two regions, we now define a single boundary between the fast and slow moving areas. To obtain this surface, we again select the fastest and slowest third of the both large and small particles at $t/\tau = 10^4$ and apply on each particle the following potential

$$\phi(r) = \begin{cases} 1 & \text{if } r \leq \frac{\sigma}{2} \text{ and particle is fast} \\ -1 & \text{if } r \leq \frac{\sigma}{2} \text{ and particle is slow} \\ \exp(-2(r - \frac{\sigma}{2})) & \text{if } r > \frac{\sigma}{2} \text{ and particle is fast} \\ -\exp(-2(r - \frac{\sigma}{2})) & \text{if } r > \frac{\sigma}{2} \text{ and particle is slow,} \end{cases} \tag{9.4}$$

with σ the diameter of the particle. The surface that we will plot is then the equipotential surface defined by

$$\Phi(\mathbf{r}) = \sum_{i=1}^{M} \phi(\mathbf{r}_i - \mathbf{r}) \equiv 0, \tag{9.5}$$

where the summation runs over the fastest and slowest third of the particles. Note that this surface will lie in between the surfaces shown in Figure 9.1.

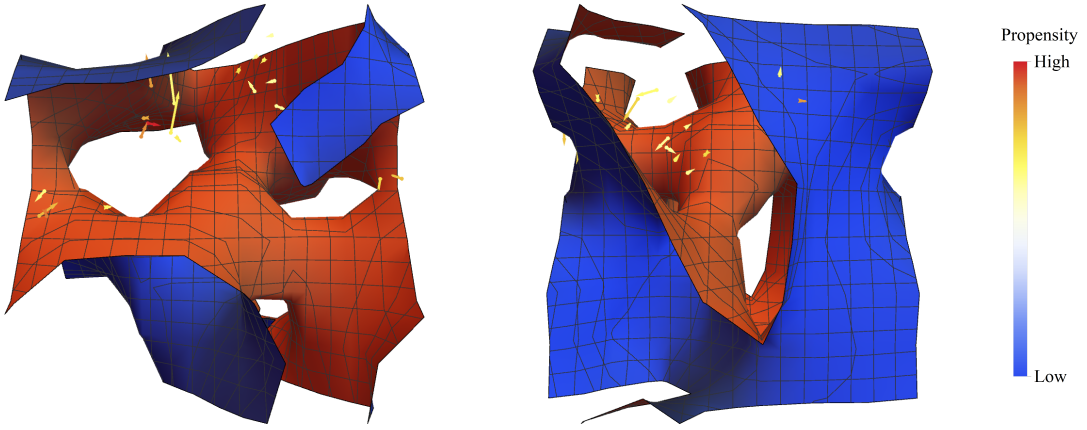


Figure 9.2: In this figure the DSP vectors associated with escaped particles $t/\tau = 50$, as well as the surface defined by Equation (9.4). The DSP vectors are scaled by a factor of two to increase visibility and the associated colors hold information about the relative propensity (see bar). On the red side of the surface, particles associated with the fast moving area are located, while on the blue side slow moving particles are situated.

9.2.3 Spatial escape evolution

In Figure 9.2 we plot both the boundary between the fast and slow moving areas, as well as the vector $\Delta\mathbf{a}_i$ for the particles that, following the above defined escape criterion, have escaped at $t/\tau = 50$. The fast-moving area is located on the red side of the surface, while the slow-moving area is on the blue side. In these figures (and in the figures in Appendix D) we see that the particles that escape earliest, are often particles that are located relatively close to the surface. This is strongly in conflict with what we would have expected; naively, one would think that the early mobile particles are located deep inside the fast-moving regimes.

The high mobility of the interface region is somewhat reminiscent of a grain boundary in polycrystals or crystal-crystal coexistences, where the most mobile particles are at the interface [89]. However, in this case, the surface is associated with the boundary between a fast and slow moving area, not two crystal domains. Having such an interface in a glassy system, could suggest that the structures associated with fast and slow moving areas are in fact reminiscent of two different phases in coexistence. We already know from earlier research that fast and slow moving areas indeed have different structures – otherwise the structural parameters would not be able to identify whether a particle is located in a fast or slow moving area. The question is however, whether the structures of fast and slow moving areas are merely two ends of a continuum of slowly varying structure, or that they represent genuinely different states of matter, separated by some sort of interface. Although the results we show here are certainly not strong enough evidence to conclude that we are dealing with such an interface, the possibility of its existence is very intriguing and is somewhat reminiscent of the two-state picture of the glassy behavior of liquid water [90, 91].

Since glass is a disordered material, distinguishing whether the structures of fast and slow moving ares are two ends of a continuum or two truly distinctive glassy states separated by an interface will be difficult: as we have already seen in the past, it is hard to find qualitative differences between two disordered structures. Moreover, the order parameters that we use in this thesis are not suited for exclusively distinguishing between the two scenarios: since the parameters are averaged over multiple shells, parameters associated with possible interface particles will always contain information about both fast and slow moving areas. Even if there is an interface, the order parameters will thus show a continuous transition of structure from fast to slow moving areas.

9.2.4 Conclusions and outlook

While highly qualitative, in this chapter we showed that, consistent with what was seen in single trajectories in 2D, at time scales near the relaxation time, there is a distinct connection between the preferential motion of particles and their proximity to the interface between fast and slow moving regions. In particular, slow particles near such an interface preferentially move perpendicular to the interface, while fast ones move parallel.

Looking at the particles that tend to escape their cages the earliest reveals a surprising phenomenon: these particles are largely located close to the interface between the fast and slow moving regions and not deeply in the fast moving regime, as we would have expected. This observation suggests that the interface between these two regions has distinct properties from the two regions themselves, and hence might be interpretable as an interface between two distinct structural phases. However, it will be difficult to prove whether or not a two-phase picture is indeed applicable to our system. We do however have some suggestions for future research that might help to solve this problem.

Firstly, we should consider larger systems. Currently each system contains roughly one fast and one slow moving area. It would be interesting to see if we see the same behavior of early escaped particles in larger systems that contain multiple non-connected areas of different mobility.

Secondly, it would be beneficial to develop an accurate order parameter or parameters that can distinguish the fast and slow regions too, but that also focuses more on structural asymmetry. Maybe from this order parameter, we could learn something about qualitative differences between interface particles and particles that are part of the fast or slow moving area. An alternative is to look at the current structural parameters, and see which parameters are most important for distinguishing between fast and slow moving areas. This can be done for example by looking at the associated weights in the linear regression algorithm. Although, from the discussion above, it is not certain whether this will teach us something new about the qualitative differences of the structure of interface particles, it is still something that should be examined. Note that we already briefly analyzed which order parameters are the most important. If we look at the weights that are given to the total set of order parameters by linear regression, the results are seemingly random. However if we look at individual parameters, it turns out that there are several order parameters that are are quite discriminative for whether a particle is located in a fast or slow moving area.

Finally it would be interesting to see how the boundary between fast and slow moving areas changes over time. In order to examine this, one could run a single trajectory simulation of a glassy system for a duration of multiple times the relaxation time. By making snapshots at discrete time intervals Δt, we can use linear regression to predict the fast and slow moving areas of these snapshots. By using the technique described in Section 9.2.2, one can then also find the (approximate) boundary surface for these snapshots. This analysis would give us an impression of how the boundary moves over time, which could then be compared with the movement of interfaces in other systems.

Conclusions and outlook

In this thesis we examined the relation between structural and dynamical heterogeneity in a glassy system. To this end, we explored in detail the structure and dynamics of a binary hard-sphere system deep in the glassy regime.

In Chapter 5, we made a comparison between three machine learning techniques (linear regression (LR), neural networks (NN) and graph neural networks (GNN)) that we trained to predict the dynamic propensity of particles, based on structural parameters. Earlier research already showed that these three methods could be used to predict the propensity based on structure, however, no comparison had been made yet [35, 36]. When we compared the three methods, we found that, although they all predict the propensity of particles with approximately the same accuracy, the NN and the GNN took longer to train compared to LR. Additionally, the GNN and especially the NN were found to be more sensitive to hyperparameters than LR. Therefore we concluded that linear regression is the preferred method to predict the propensity based on structure.

During the comparison of the three methods, we found that the structural parameters contained a substantial amount of information about the propensity for times close to the relaxation time, but not for the shorter time scales associated with the caging regime. To address this, in Chapter 6 we examined whether we could improve the propensity prediction by providing the algorithm with more structural information. The three methods that we tried, were giving the LR algorithm information about i) the standard deviation of the structural parameters, ii) the predictions of neighbouring particles at earlier times and iii) the inherent state of the system. From our analysis, it became clear that although the first two methods did not influence the accuracy of the predictions, adding information about the quenched system did lead to a significant improvement. In particular, adding information about the absolute distance between quenched and initial coordinates improved the prediction for the caging regime, while providing linear regression with information about the structural parameters of the quenched system improved the prediction for times close to the relaxation time.

The improved predictions during the caging regime, led to the conclusion that caged particles on average move around a well-defined cage center. The position of this center is primarily determined by structure and closely related to the inherent state positions of the particles. We already knew from single trajectory measurements that particles move around an equilibrium point in the cage, but it was not yet known that the position of this point was influenced by structure rather than dynamics [2]. Inspired by these results, in Chapter 7 we designed an improved method for determining the cage center from the structure of the system. In particular, we measured the ensemble averaged coordinates of particles in a Monte Carlo simulation where particles are restricted to their respective cages. The cage position found by the MC method turns out to be an extremely strong indicator for the dynamic propensity of the particles during the caging regime. Moreover, the correlation of the vector propensity with the vector connecting the cage center to the initial position of the particle is nearly perfect during the caging regime. This indicates that in this regime, the average movement of particles is dominated by the transport to their cage center. We furthermore showed, by freezing part of the system, that finding the position of the caging point is a highly collective phenomenon in which particles at a large distance still influence each other.

Combining our improvements to the prediction from Chapter 7 and the improved cage determination method, we were finally able to significantly enhance our predictions of the dynamic propensity based on structural parameters. In Figure 9.3 we show how the new developed methods improve the propensity prediction in terms of the coefficient of determination r^2. In the figure, the blue line represents the state-of-the-art before this thesis [35], while the orange line represents our best predictions.

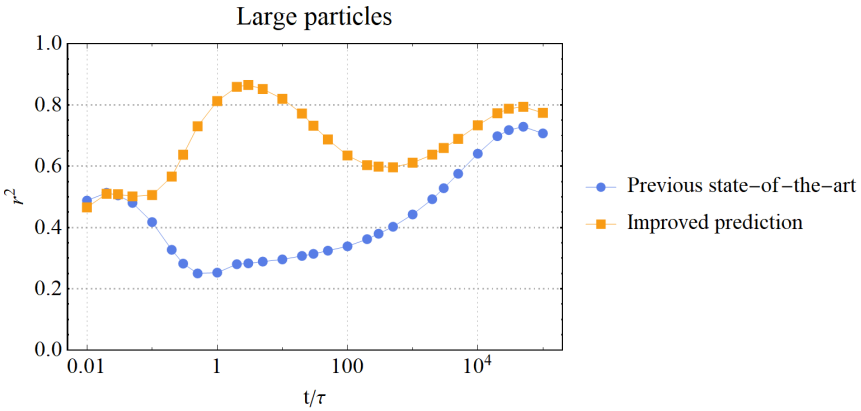


Figure 9.3: Propensity prediction using linear regression, based on two different sets of input parameters. The blue line uses only the structural parameters discussed in Chapter 2.2 up the third generation. The orange line includes the improvements made in this thesis, by using the structural parameters based on both the normal coordinates and the cage center coordinates, as well as the absolute distance between the cage center and the initial coordinates.

In Chapter 8, we explored the role of directionality in the glassy dynamics of our system. In particular, we examined whether particles have a preference for moving in a given direction at times beyond the caging regime. As a measure for the anisotropy of the dynamics, we considered the dynamical self-alignment parameter, which is an order parameter that reflects how much the direction of movement of particles between different trajectories is aligned. We concluded that, also after caging, the movement of particles is anisotropic, something that can be explained by at least two processes. Firstly, different parts of the system show collective motion in the form of drifting. Secondly, we found that when particles escape their cage, there are certain directions that are on average more favourable to move to. We expect that this effect will be the most prominent for particles that escape their cage early, since for those particles the cage structure in different trajectories will still be the same as in the initial configuration. This statement is supported by the result that for times close to the relaxation time the propensity starts to become more isotropic.

In Chapter 9, we explored the dynamic propensity of particles in relation to their position within the slower and faster rearranging regions in the system, both at short times as well as near the relaxation time. In particular, we divided the system into slow and fast moving regions based on their dynamics at the relaxation time, and drew interfaces between the two regions. We found that near the relaxation time, consistent with previous literature [2], the interface affects the preferred direction of motion of nearby particles. In particular, fast moving particles prefer to move parallel to this interface, while slow moving particles move perpendicular to the interface. Additionally we looked at the location of particles which tend to escape their cage early. The most interesting finding was that the positions of these early escaping particles lie close to the boundary between (what will later be) the fast and slow moving area.

The existence of an interface might imply that fast and slow moving areas are two genuinely different structural states of glass instead of two extremes of one structural continuum. Currently our results are not sufficient to decisively conclude whether or not this interface truly separates two different structural states, and therefore more research will need to be conducted in the future.

To investigate the possible existence of the interface we propose several future avenues of research in Chapter 9:

- Consider larger systems (e.g. $\sim$ 16000 particles) to see the spatial distribution and size variation of fast and slow moving areas form.
- Make a quantitative comparison between the structure of fast and slow moving areas, possibly by designing new order parameters that also capture the asymmetry of the local environments.
- Obtain a picture of how the boundary between fast and slow moving areas moves over time. This could be done by running a single-trajectory measurement of a glassy system for an extended period of time, making snapshots at discrete time intervals. With a trained linear regression model and the methods discussed in Chapter 9), the location of the boundary can then be predicted for the different snapshots.

The possible existence of an interface between the regions that move quickly or slowly over long time scales is one of the more intriguing findings of this thesis. If such an interface indeed exists, it could drastically change our view on dynamical heterogeneity and therefore on the glass transition in general. It will be exciting to see what future research can teach us.

Layman's Summary

In this thesis we looked at the movement of particles in a glassy material. Although most people will immediately picture windows and tea glasses when they think of glass, from a scientific point of view there are many more materials in a glassy state. Think for example of ceramic objects, the material that golf sticks are made of and even the scab that closes a wound. The reason that these seemingly different materials can all be classified as glass, is because a glass is a system state (like a gas or a liquid), rather than a material. To understand when we can say that a material is in a glassy state, we will first discuss what a liquid and crystal state look like, since a glassy state bears resemblances to both of these states.

In general a material is in a crystal state when the positions of the particles in the system follow an ordered pattern, see Figure 4. As a result, the movement of particles is restricted, meaning that particles only move in a small volume around their position in the lattice. On the other hand, a material is classified as a liquid when the positions of particles are disordered and particles can move more or less freely through the system.

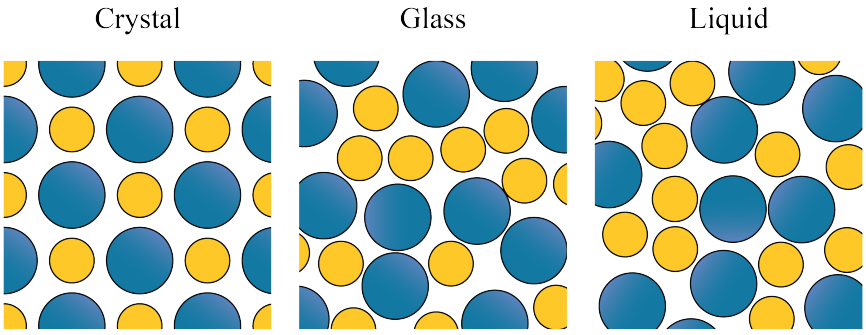


Figure 4: Cartoon of the free different material states.

Although the story above is very simplified, it gives a general picture of the two-fold difference between a liquid and a crystal state; in terms of structure the liquid state is disordered and the crystal state is ordered. Dynamically, the liquid state contains particles that can move through the system, while the crystal state contains particles with restricted movement. To go from a liquid to a crystal state, one typically has to decrease the temperature or increase the density.

The glassy state that we mentioned above lies somewhere in between the crystal and the liquid state. One can reach it by very rapidly decreasing the temperature or increasing the density of a system. Doing so, the system does not have time to form an ordered lattice and instead all particles will be crammed up in a disordered structure (see Figure 4). One can understand this by imagining a box full of marbles. In principle more marbles fit in the box when they are stacked nicely, but if you throw them in the box all at once, there is no time for nice stacking, and instead a disordered structure will arise. As a result the structure of a glass bears the most resemblance to that of a liquid.

Dynamically however, a glass state is very different from a liquid. Due to the high density of a glass, particles almost do not move. The easiest way to understand this is by imagining a room full of people that walk in random directions. If we decrease the volume of the room, people are pushed together more and more and at one point they are so close to each other that they cannot move past their neighbours. In a glass the same thing happens; particles are restricted to a so called 'cage' made up by the particles around them, see the left figure of Figure 5. The difference with crystals, where we also see that the movement of particles is restricted, is that particles at one point will leave their cage and move past their first shell of neighbours (see the right figure of Figure 5). The reason for this is that in a disordered structure the random movement of particles will at some point create enough space for a particle to move out. As a result, glass does flow like a liquid. However, because the escapes are quite rare and particles are captured again in a new cage after escaping, the flowing of glass is very slow; it can take thousands of years to really see a significant movement in glass. That is also why in normal life we often often regard glassy materials as crystals; the movement is just too slow for us to observe.

The open question that has puzzled researchers over the past few decades is why in a glass the slowdown in the mobility of particles is so enormous. From the story above one can understand that the higher the density, the longer particles will be caged, and thus the slower the material will move. However it does not explain why the increase in time that it takes a particle to escape its cage is so substantial: looking at the structure of a glass and a liquid, it is sometimes almost impossible to distinguish the two, however the mobility of the particles in the two states can differ by orders of magnitude.

Aim of this thesis

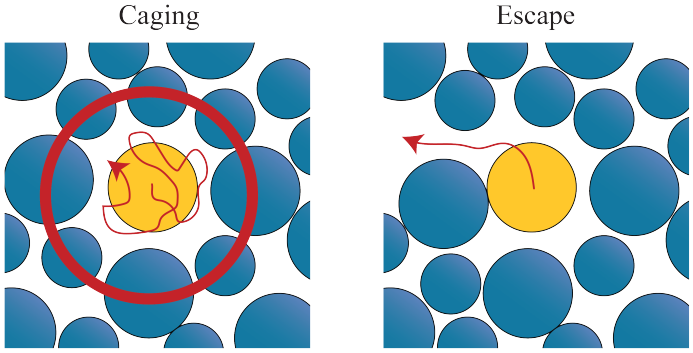


Figure 5: Cartoon of the caging and escaping of particles. In the left figure a particle is caged by its neighbouring particles. In the right figure, fluctuations in the movement of neighbouring particles allow a particle to escape.

In this thesis we looked at a phenomenon associated with the slowdown of particles; when going from a liquid to a glass, not all particles slow down equally. Instead regions of very mobile and very non-mobile particles arise. This means that in some regions particles are more likely to escape their cage than in others. This phenomenon is what we call 'dynamical heterogeneity'. In this thesis we looked at whether this heterogeneity could be explained from a structural point of view, i.e. can we, from the way particles are arranged in the glass, explain why some particles are more likely to escape than others?

The approach for answering this question was using machine learning. Machine learning is a collective name that includes a lot of different techniques. The basic idea of all these techniques is to give a computer information (data) about a system and then let an algorithm learn to draw

conclusions or make predictions based on that data. In our case we used machine learning to predict the movement of particles (which tells you something about the mobility of particles and thus the dynamical heterogeneity), based on the local structure of these particles.

The thesis consisted of three main parts. In the first part we compared three different machine learning techniques to see which method predicted the dynamical heterogeneity the best. In the past people have already shown that various machine learning techniques were able to predict the movement of particles and we wanted to quantify how these techniques compared [35, 36]. For the second part we used the best performing machine learning technique to examine if we were able to improve the prediction. In the final part we took a closer look at the results we obtained from these predictions and reflected on what they could tell us about the physics behind dynamical heterogeneity. In the next sections, we will discuss the most important findings of this thesis. We will however start with explaining how we collected data on glassy systems.

Characterizing the dynamics of particles in a glassy system

To collect data on glassy systems we used computer simulations that simulated a system that consisted of two different sizes of so called 'hard spheres'. These spheres are marble-like, meaning that they cannot overlap, but otherwise do not interact. It has already been shown in the past that the dynamical behaviour of such a system mimics the behaviour of real glassy materials.

The computer simulations allowed us to measure the 'dynamic propensity'. This propensity is a measure for the average mobility of a particle in the system, i.e. how much we can expect it to move on average. We took a box that contained a particle system in a glassy configuration, gave each particle a random velocity and then measured how far the particles moved over time as seen from their initial position. By measuring this many times, each time letting the particle start from the same position but with a slightly different velocity, we obtained a picture of how mobile the particles were on average over time.

The initial positions of particles were also the basis for the structural analysis. To capture the local environment of each particle, we looked at spherical shells at different distances as seen from that particle. For each shell we constructed parameters that described the local environment by measuring the density and the way particles in that shell were structured.

Thesis part one: Comparing machine learning techniques

The first goal was to compare three machine learning algorithms that could predict the dynamic propensity based on these structural parameters. Although this might sound very futuristic, it is something that we humans unconsciously do every day. Think for example of reading. When we were young we trained on recognizing letters, such that after a while we were able to read. We learned to recognize that a line with a dot above it can be classified as an 'i' or a 'j' and that when something consists of a single circle it must be an 'o'. Machine learning does essentially the same thing, it uses input data (in the case of reading for example a picture of the letter) to make a prediction (in this case the letter type). The way an algorithm learns is by comparing its predictions with the actual (measured) outcome, and then changing the algorithm accordingly.

To measure the performance of an algorithm, we look at how the algorithm performs on new data that it has never seen before. This is because we want the algorithm to be able to make predictions not only on data it has seen before, but also on new data; just like we humans do not have to learn how to read again, each time we see a new handwriting.

For this thesis we compared three different machine learning algorithms, that we will all briefly discuss below. The first one, called linear regression, gives the best prediction possible by multiplying every input parameter by a number (or weight) and afterwards adding everything together.

The second method that we used, neural networks, also multiplies the different input parameters with weights. However, instead of just adding everything like in the case of linear regression, it applies a more complicated (non-linear) function. As a result, neural networks are better in capturing a more complex relation between input and the output parameters. Finally, we used graph neural networks to predict the propensity. This method is quite similar to neural networks. The main difference is that it also allows the predictions of neighbouring particles to influence each other. This means that the network first makes a prediction for a particle based solely on its own parameters and then uses that prediction, as well as the predictions of nearby particles, to make a new and hopefully improved prediction. As a result, graph neural networks can use more structural information for their prediction than neural networks and linear regression. In order to make a fair comparison between the three techniques, we therefore allowed linear regression and neural networks to also use input parameters of particles nearby.

Note that although the three methods may seem (very) different, the general idea behind them is the same: the algorithms all fit a function that takes the structural parameter as input and gives a prediction of the propensity as an output. It is the parameters of this function that are adjusted while training the algorithms, to make the prediction as good as possible.

The conclusion of the comparison was that all three methods worked about equally well, but since linear regression was significantly faster than the other two methods, this method was the preferred method. The fact that the simplest model works the best might be surprising, but it is something we see quite often in the field of machine learning. A complicated method can use more advanced tools to make predictions, but is also harder to train. For the dynamic propensity it turned out that the advantages of the more complicated methods did not outweigh the disadvantage of having a model that is harder to train.

To give some impression of how well the method worked we plot the result in Figure 6. For multiple points in time we trained a model to predict the propensity, and then compared how close this predicted propensity was to the measured propensity. On the x-axis in the figure we plot the time and on the y-axis we plot the r^2 parameter that evaluates how accurate the prediction is. The closer the value of r^2 is to one, the better the prediction is for that time. As we can see, the model does quite well for short and large times, but has a hard time predicting where the particle will be at intermediate times. These are the times when the particle is still caged in its initial cage. Overall we can however conclude that there is a strong link between structure and dynamics. Especially for large times, the local structure of particles is a really good predictor of whether a particle will move a lot or not.

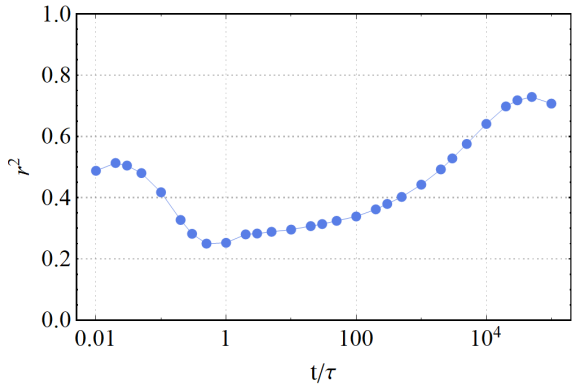


Figure 6: Performance of linear regression over time. On the x-axis we show the time (where the time unit is not important for now), while on the y-axis we show the coefficient of determination r^2, which is a measure of the accuracy of the prediction (the closer to one r^2 is, the better the prediction).

Thesis part two: Improving linear regression

After the conclusion that linear regression was the best method to work with, we wanted to see whether we could improve the prediction by providing extra structural information. We tried several approaches that mostly did not significantly improve the prediction and that we will therefore omit here. There was, however, one approach that did work very well, and moreover, that improved the prediction for intermediate times, i.e. during the times where our earlier prediction was not very good. Below we will discuss what information we added and why this increased the prediction.

As we mentioned earlier, before they escape, particles in a glass will move around some equilibrium position. In Figure 7 we schematically show what we mean by this. The movement of particles in a cage can be compared with the movement of balls moving around in a well. If you let the particles go, they will wobble around some equilibrium position, also called caging center, associated with the lowest part of the well (i.e. the yellow points in the figure). This means that if you measure the position of a particle in a cage many times, like we do to obtain the propensity, on average that particle will be at its equilibrium position. Making a good prediction of the propensity during the times that a particle is caged, thus essentially means predicting where the equilibrium position of that cage is.

There are several reasons why linear regression has a hard time predicting the equilibrium position of particles based on only the structural parameters. The first reason is that just from structural parameters, or even from the initial positions of particles, it is very hard to tell how far away a particle is from the center of its cage. The main reason for this, is that the cages of particles are not static. Since they are formed by the surrounding particles, when those particles move, the cage structure and thus the equilibrium position change. This means that in order to find the equilibrium position of a particle, you cannot regard particles separately. Another, somewhat more subtle reason why it is hard to find the cage positions is because all the structural parameters we use are rotationally invariant. This means that if you would rotate the surrounding of a certain particle, the structural parameters associated with that particles would not change. The reason that we use these rotational invariant parameters is because it simplifies the training of the linear regression algorithm. However since our parameters are rotationally invariant you can not use them to make a prediction about a direction, i.e. you cannot use them to tell whether your particle goes left or right, up or down, etc. Knowing in which direction a particle moves to its equilibrium position, is however essential to find the equilibrium position of neighbouring particles: whether a neighbouring particle goes left or right influences the cage of a particle and thus the equilibrium position.

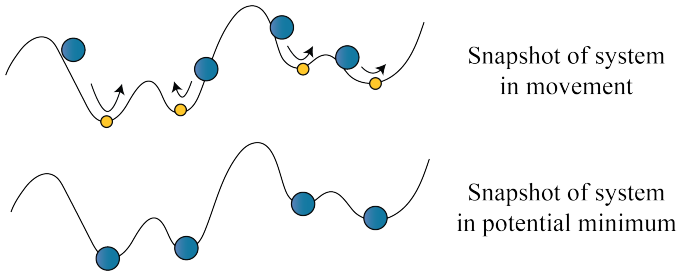


Figure 7: Cartoon of a system in movement with particles that around their equilibrium position (top figure) and a system where are particles are situated in their equilibrium position (lower figure).

The story above explains why it is very hard to predict the cage center based on the structural parameters. Fortunately we can use other methods to provide information about the equilibrium position. In this case we decided to 'quench' the system. What this technique does, is only allowing particles to move towards a direction that brings them to a lower energy. To understand why this

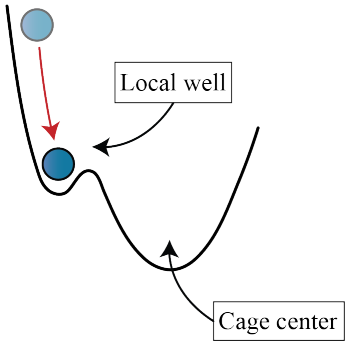


Figure 8: Cartoon of particles getting stuck in an energy well using the quench method.

works we can again look at Figure 7. Just as the equilibrium position in this figure is associated with the lowest point in the well, i.e. the point with the lowest potential energy, the equilibrium position in a cage is associated with the point in the cage that has the lowest potential energy. For the system in Figure 7 quenching the system would mean that particles can only move down. By doing so we end up in the situation sketched in the lower figure. By giving the linear regression information about the distance between the particles in the initial snapshot and the snapshot in the potential minimum, it turns out that we improve the propensity prediction enormously.

Moreover, using the success of quenching, we developed an even better method to find the center of the cage. Discussing this method lies outside the scope of this Layman summary, however we can explain why quenching the system might not be the best method to find the cage center: during quenching, particles are only allowed to move down in energy. As a result, they can sometimes get stuck in a small energy well (see Figure 8). The new method uses another technique that turns out to be better in finding the equilibrium point. As a result, the propensity prediction improved even more.

In Figure 9 we show the overall improvement we made to the propensity prediction in this thesis: the blue line represents the best prediction as was available before this thesis, while the orange line represents the best prediction that we achieved. As one can see, the improvement is very substantial.

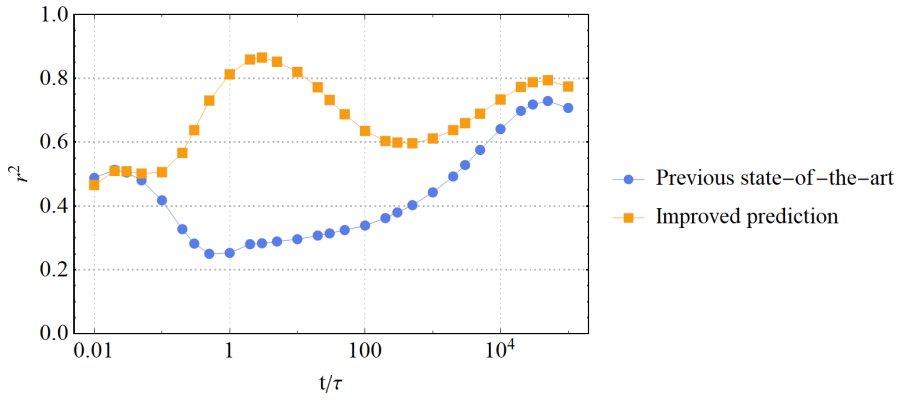


Figure 9: Improvement of the propensity prediction that was achieved in this thesis. Again, on the x-axis we show the time, and on the y-axis we show the goodness-of-fit parameter r^2. The higher the value of r^2, the better the prediction. The blue line represents the best existing prediction before this thesis, while the orange line represents the improved prediction we made.

Thesis part three: a closer look at the dynamics

In the last part of the thesis we discussed what we can learn from the improvements. The fact that on average particles move towards a well defined caging center, led to the conclusion that the direction in which particles travel is important when one tries to understand the dynamics of caged particles. Moreover, we were curious whether directional information would also be important at times during and after particles have escaped. The reason that we want to examine this, is because it tells us something about what aspects of the structure influence the dynamics. Currently, as mentioned earlier, our order parameters are rotationally invariant (i.e. they do not hold information about direction) and as a direct result, they will never be able to predict the direction a particle will move in. If it turns out that directional information is important, we know that the current parameters are probably missing some of the important aspects of the structure.

In order to investigate the importance of direction, we looked at how much particles in different trajectories travel in the same direction (remember that the propensity is obtained by measuring the trajectories of particles multiple times). If particles in different trajectories always move in the same direction, we can conclude that there is something in the initial structure that leads to this directional preference. By looking at the trajectories, we found that indeed particles have the tendency to move in certain directions. We coupled this directional preference to two interesting phenomena. The first one is that there is a preference of direction when particles escape their cage. This means that some caging structures allow particles to escape more easily in one direction than the other. Secondly, we found that parts of the system show collective motion, also called drift.

Finally, we briefly looked at where in the system the particles are located that start to escape their cage the earliest. As we mentioned before, when looking at the dynamics we see areas of fast and slow moving particles arise. If we examine the location of the early escaped particles, we often see that these particles lie at boundary between what later will be the fast and slow moving areas. If one thinks about it, this is quite an interesting finding. Intuitively, one would expect particles to start moving the earliest in areas that become fast moving areas.

Outlook

Our findings lead to several options for follow-up research. The first direction that we would like to look at is what structural parameters hold important information. Right now the prediction is based on at least 1062 parameters per particle. The fact that in Figure 6 for the largest times, the prediction does such a good job, means that there is a lot of information about the dynamics hidden in the structure. It would be interesting to examine which of these parameters and thus which part of the structure are most important for the propensity prediction.

The second and maybe most interesting follow-up research is associated with the evolution of fast and slow moving areas. The fact that the first particles escape in the area between what will later be fast and slow moving areas, is somewhat reminiscent of an interface. An interface is a surface or thin layer of particles that acts as a boundary between two different states of matter, for example a liquid and a crystal. If a glassy system would include such an interface, it would lead to the conclusion that fast and slow moving areas are associated with different glassy states. Although right now we have definitely not enough evidence to conclude whether our glassy system contains such an interface, it is something that will be very interesting to look at in future research. Can we for example observe two different states of matter if we look at the structural parameters of fast and slow moving areas? And how do the fast and slow moving areas and the boundary between them move over time? Looking at these questions will hopefully bring us closer to understanding dynamical heterogeneity.

Acknowledgements

During this thesis a lot of people helped and supported me and I would like to thank them. First and foremost I would like to thank my supervisor Dr. Frank Smallenburg. Your supervision was a perfect balance between allowing me to try out my own ideas and giving the right amount of direction to the project (such that in the there were actually results to present). Moreover, you are just a genuinely nice and pleasant person and I am very glad I got to work with you. Next, I would also like to thank my other supervisor Dr. Laura Filion. Over the past years I have had the opportunity to work with you several times, and each time I really enjoyed it and learned a lot. Also a thanks to Joost de Graaf for being my second supervisor.

Furthermore I would like to thank Marjolein de Jager for all the discussions we had about physics, academic life and other random topics (often related to cats). Moreover, thank you for helping me with some of the analysis. Also a thanks to all the master and bachelors students in this group. In the short period of time we were allowed to be at the university, I really enjoyed getting to know you during one of the many coffee breaks (and of course the city walk).

Finally I would like to thank my friends and family. A big thanks to my parents for always being very interested to hear more about the weird dynamical behaviour of glass (or at least being really good at pretending). I cannot express enough how important your interest and support is to me. Moreover, thank you for proofreading several chapters of this thesis. I would also like to thank my sister: although you were less involved in my thesis, I really liked the weekends in which we made music together. Also a big thanks to my roommates, especially Jeshua, Marjolein, Sieb, Sophie, Martijn en Jesse. Because of corona, most of my thesis was written from home. As roommates we have spend many hours studying together in our so called 'home office'. I really enjoyed those hours, and moreover I really enjoyed the many walks and well deserved coffee breaks. A special thanks to Marjolein for proofreading my Layman's summary.

Bibliography

[1] S. Patel, B. Swain, A. Behera, and S. Mohapatra, *Metallic Glasses: A Revolution in Material Science* (IntechOpen, 2020).

[2] L. Berthier and G. Biroli, *Theoretical perspective on the glass transition and amorphous materials*, Reviews of Modern Physics **83**, 587 (2011).

[3] L. M. C. Janssen, *Active glasses*, Journal of Physics: Condensed Matter **31**, 503002 (2019).

[4] G. R. Luckhurst, *Orientational order in liquid crystals: Experiment and theory*, Berichte der Bunsengesellschaft für physikalische Chemie **97**, 1169 (1993).

[5] C. Royall and S. R. Williams, *The role of local structure in dynamical arrest*, Physics Reports **560**, 1 (2015).

[6] S. Karmakar, C. Dasgupta, and S. Sastry, *Length scales in glass-forming liquids and related systems: a review.*, Reports on Progress in Physics **79 1**, 016601 (2016).

[7] S. Albert, T. Bauer, M. Michl, G. Biroli, J.-P. Bouchaud, A. Loidl, P. Lunkenheimer, R. Tourbot, C. Wiertel-Gasquet, and F. Ladieu, *Fifth-order susceptibility unveils growth of thermodynamic amorphous order in glass-formers*, Science **352**, 1308 (2016).

[8] W. W. Kob and H. C. Andersen, *Testing mode-coupling theory for a supercooled binary Lennard-Jones mixture i: The van hove correlation function*, Physical Review E **51**, 4626 (1995).

[9] M. N. Goldstein, *Viscous liquids and the glass transition: A potential energy barrier picture*, Journal of Chemical Physics **51**, 3728 (1969).

[10] W. Kauzmann, *The nature of the glassy state and the behavior of liquids at low temperatures.*, Chemical Reviews **43**, 219 (1948).

[11] K. Chang, *Anything but clear*, New York Times (2008).
https://www.nytimes.com/2008/07/29/health/29iht-29glass.14846468.html.

[12] F. C. Frank and N. F. Mott, *Supercooling of liquids*, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences **215**, 43 (1952).

[13] M. Leocmach and H. Tanaka, *Roles of icosahedral and crystal-like order in the hard spheres glass transition*, Nature Communications **3**, (2012).

[14] H. Tong and H. Tanaka, *Revealing hidden structural order controlling both fast and slow glassy dynamics in supercooled liquids*, Physical Review X **8**, 011041 (2018).

[15] L. M. C. Janssen, *Mode-coupling theory of the glass transition: A primer*, Frontiers in Physics **6**, 97 (2018).

[16] W. van Megen and S. M. Underwood, *Glass transition in colloidal hard spheres: Measurement and mode-coupling-theory analysis of the coherent intermediate scattering function*, Physical Review E **49**, 4206 (1994).

[17] V. Lubchenko and P. G. Wolynes, *Theory of structural glasses and supercooled liquids*, Annual Review of Physical Chemistry **58**, 235 (2007).

[18] J. C. Mauro, Y. Yue, A. J. Ellison, P. K. Gupta, and D. C. Allan, *Viscosity of glass-forming liquids*, Proceedings of the National Academy of Sciences **106**, 19780 (2009).

[19] M. D. Ediger, *Spatially heterogeneous dynamics in supercooled liquids*, Annual Review of Physical Chemistry **51**, 99 (2000).

[20] L. Berthier, G. Biroli, J.-P. Bouchaud, L. Cipelletti, and W. van Saarloos, *Dynamical Heterogeneities in Glasses, Colloids and Granular Media* (Oxford Science Publications, 2013).

[21] H. Sillescu, *Heterogeneity at the glass transition: a review*, Journal of Non-Crystalline Solids **243**, 81 (1999).

[22] G. Biroli and J. P. Garrahan, *Perspective: The glass transition*, The Journal of Chemical Physics **138**, 12A301 (2013).

[23] A. Widmer-Cooper, H. Perry, P. Harrowell, and D. R. Reichman, *Localized soft modes and the supercooled liquid's irreversible passage through its configuration space*, The Journal of Chemical Physics **131**, 194508 (2009).

[24] S. Marín-Aguilar, H. H. Wensink, F. Foffi, and F. Smallenburg, *Tetrahedrality dictates dynamics in hard sphere mixtures*, Physical Review Letters **124**, 208005 (2020).

[25] E. D. Cubuk, S.S.Schoenholz, J. M. Rieser, B. D. Malone, J. Rottler, D. J. Durian, E. Kaxiras, and A. J. Liu, *Identifying structural flow defects in disordered solids using machine-learning methods*, Physical Review Letters **114**, 108001 (2015).

[26] A. Malins, J. Eggers, C. P. Royall, S. R. Williams, and H. Tanaka, *Identification of long-lived clusters and their link to slow dynamics in a model glass former*, The Journal of chemical physics **138**, 12A535 (2013).

[27] R. Shi and H. Tanaka, *Distinct signature of local tetrahedral ordering in the scattering function of covalent liquids and glasses*, Science Advances **5**, eaav3194 (2019).

[28] H. Tong and H. Tanaka, *Structural order as a genuine control parameter of dynamics in simple glass formers*, Nature Communications **10**, (2019).

[29] J. Paret, R. L. Jack, and D. Coslovich, *Assessing the structural heterogeneity of supercooled liquids through community inference*, The Journal of Chemical Physics **152**, 144502 (2020).

[30] W. Lechner and C. Dellago, *Accurate determination of crystal structures based on averaged local bond order parameters*, The Journal of Chemical Physics **129**, 114707 (2008).

[31] X. Jia, W. Lin, H. Zhao, H. Qian, and Z. Lu, *Supercooled melt structure and dynamics of single-chain nanoparticles: A computer simulation study*, The Journal of Chemical Physics **155**, 054901 (2021).

[32] A. V. Anikeenko and N. N. Medvedev, *Polytetrahedral nature of the dense disordered packings of hard spheres*, Physical Review Letters **98**, 235504 (2007).

[33] B. Charbonneau, P. Charbonneau, and G. Tarjus, *Geometrical frustration and static correlations in a simple glass former*, Physical Review Letters **108**, 035701 (2012).

[34] C. Xia, J. Li, Y. Cao, B. Kou, X. Xiao, K. Fezzaa, T. Xiao, and Y. Wang, *The structural origin of the hard-sphere glass transition in granular packing*, Nature Communications **6**, 8409 (2015).

[35] E. Boattini, F. Smallenburg, and L. Filion, *Averaging local structure to predict the dynamic propensity in supercooled liquids*, Physical Review Letters **127**, 088007 (2021).

[36] V. Bapst, T. Keck, A. Grabska-Barwińska, C. Donner, E. D. Cubuk, S. S. Schoenholz, A. Obika, A. W. R. Nelson, T. Back, D. Hassabis, and P. Kohli, *Unveiling the predictive power of static structure in glassy systems*, Nature Physics **16**, 448 (2020).

[37] S. S. Schoenholz, E. D. Cubuk, D. M. Sussman, E. Kaxiras, and A. J. Liu, *A structural approach to relaxation in glassy liquids*, Nature Physics **12**, 469 (2016).

[38] A. Widmer-Cooper, P. Harrowell, and H. Fynewever, *How reproducible are dynamic heterogeneities in a supercooled liquid?*, Physical Review Letters **93**, 135701 (2004).

[39] L. Berthier and R. L. Jack, *Structure and dynamics of glass formers: Predictability at large length scales*, Physical Review E **76**, 041509 (2007).

[40] P. Chaudhuri, L. L. Berthier, and W. Kob, *Universal nature of particle displacements close to glass and jamming transitions*, Physical Review Letters **99**, 060604 (2007).

[41] J. Hansen and I. R. McDonald, in *Theory of Simple Liquids (Fourth Edition)* (Academic Press, 2013).

[42] M. Ediger, *Can density or entropy fluctuations explain enhanced translational diffusion in glass-forming liquids?*, Journal of Non-Crystalline Solids **235**, 10 (1998).

[43] A. Widmer-Cooper and P. Harrowell, *On the study of collective dynamics in supercooled liquids through the statistics of the isoconfigurational ensemble*, The Journal of Chemical Physics **126**, 154503 (2007).

[44] P. .J.Steinhardt, D. R. Nelson, and M. Ronchetti, *Bond-orientational order in liquids and glasses*, Physical Review B **28**, 784 (1983).

[45] P. Pusey and W. Megen, *Phase behavior of concentrated suspensions of nearly hard colloidal spheres*, Nature **320**, 340 (1986).

[46] C. A. Angell, *Formation of glasses from liquids and biopolymers*, Science **267**, 1924 (1995).

[47] Y. Elmatad, D. Chandler, and J. Garrahan, *Corresponding states of structural glass formers*, The Journal of Physical Chemistry. B **113**, 5563 (2009).

[48] L. M. Martinez and C. Angell, *A thermodynamic connection to the fragility of glass-forming liquids*, Nature **410**, 663 (2001).

[49] W. Kegel and A. van Blaaderen, *Direct observation of dynamical heterogeneities in colloidal hard-sphere suspensions*, Science (New York, N.Y.) **287**, 290 (2000).

[50] H. C. Andersen, *Molecular dynamics studies of heterogeneous dynamics and dynamic crossover in supercooled atomic liquids*, Proceedings of the National Academy of Sciences **102**, 6686 (2005).

[51] D. C. Rapaport, *The Event-Driven Approach to N-Body Simulation*, Progress of Theoretical Physics Supplement **178**, 5 (2009).

[52] E. P. Bernard, W. Krauth, and D. B. Wilson, *Event-chain Monte Carlo algorithms for hard-sphere systems*, Physical Review E **80**, 056704 (2009).

[53] L. Berthier, E. Flenner, C. J. Fullerton, C. Scalliet, and M. Singh, *Efficient swap algorithms for molecular dynamics simulations of equilibrium supercooled liquids*, Journal of Statistical Mechanics: Theory and Experiment **2019**, 064004 (2019).

[54] L. Berthier, D. Coslovich, A. Ninarello, and M. Ozawa, *Equilibrium sampling of hard spheres up to the jamming density and beyond*, Physical Review Letters **116**, 238002 (2016).

[55] A. B. Doliwa and Heuer, *What does the potential energy landscape tell us about the dynamics of supercooled liquids and glasses?*, Physical Review Letters **91**, 235501 (2003).

[56] G. Wahnström, *Molecular-dynamics study of a supercooled two-component lennard-jones system*, Physical Review A **44**, 3752 (1991).

[57] W. van Megen and P. N. Pusey, *Dynamic light-scattering study of the glass transition in a colloidal suspension*, Physical Review A **43**, 5429 (1991).

[58] W. van Megen and S. M. Underwood, *Glass transition in colloidal hard spheres: Mode-coupling theory analysis*, Physical Review Letters **70**, 2766 (1993).

[59] F. Sciortino and P. Tartaglia, *Glassy colloidal systems*, Advances in Physics **54**, 471 (2005).

[60] G. L. Hunter and E. R. Weeks, *The physics of the colloidal glass transition*, Reports on Progress in Physics **75**, 066501 (2012).

[61] E. Boattini, S. Marín-Aguilar, S. Mitra, F. Foffi, F. Smallenburg, and L. Filion, *Autonomously revealing hidden local structures in supercooled liquids*, Nature Communications **11**, 5479 (2020).

[62] D. Frenkel and B. Smit, *Understanding molecular simulation* (Academic Press, Inc., 2001).

[63] F. Arceri and E. I. Corwin, *Vibrational properties of hard and soft spheres are unified at jamming*, Physical Review Letters **124**, 238002 (2020).

[64] L. Verlet, *Computer "experiments" on classical fluids. i. thermodynamical properties of Lennard-Jones molecules*, Physical Review E **159**, 98 (1967).

[65] B. J. Alder, T. E., and Wainwright, *Studies in molecular dynamics. i. general method*, The Journal of Chemical Physics **31**, 459.

[66] D. Frenkel, *Simulations: The dark side*, The European Physical Journal Plus **128**, 10 (2012).

[67] D. Freedman, R. Pisani, and R. Purves, *Statistics: Fourth International Student Edition* (W.W. Norton & Company, 2007).

[68] D. E. Hilt, D. W. Seegrist, United States Forest Service, and Northeastern Forest Experiment Station (Radnor, Pa.), *Ridge, a computer program for calculating ridge regression estimates* (Upper Darby, Pa : Dept. of Agriculture, Forest Service, Northeastern Forest Experiment Station, 1977).

[69] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12**, 2825 (2011).

[70] W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics **5**, 115 (1943).

[71] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer-Verlag, 2006).

[72] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., *Pytorch: An imperative style, high-performance deep learning library*, Advances in neural information processing systems **32**, 8026 (2019).

[73] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2014).

[74] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, Computing Research Repository **1207.0580**, (2012).

[75] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15**, 1929 (2014).

[76] P. W. Battaglia, J. B. C. Hamrick, V. B., A. S., V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, *Relational inductive biases, deep learning, and graph networks*, (2018).

[77] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al., *Interaction networks for learning about objects, relations and physics*, Advances in Neural Information Processing Systems **29**, 4502 (2016).

[78] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, *The graph neural network model*, IEEE Transactions on Neural Networks **20**, 61 (2009).

[79] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, *Graph neural networks: A review of methods and applications*, AI Open **1**, 57 (2020).

[80] M. Fey and J. E. Lenssen. *Fast graph representation learning with PyTorch Geometric.* in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, (2019).

[81] S. Marïn-Aguilar, H. H. Wensink, G. Foffi, and F. Smallenburg, *Tetrahedrality dictates dynamics in hard sphere mixtures*, Physical Review Letters **124**, 208005 (2020).

[82] W. Landecker, M. D. Thomure, L. M. Bettencourt, M. Mitchell, G. T. Kenyon, and S. P. Brumby. *Interpreting individual classifications of hierarchical networks.* in *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, page 32. IEEE, (2013).

[83] U. Buchenau, *Thermodynamics and dynamics of the inherent states at the glass transition*, Journal of Non-Crystalline Solids **407**, 179 (2015).

[84] D. A. Stariolo, J. J. Arenzon, and G. Fabricius, *Inherent structures dynamics in glasses: a comparative study*, Physica A: Statistical Mechanics and its Applications **340**, 316 (2004).

[85] F. H. Stillinger and T. A. Weber, *Hidden structure in liquids*, Physical Review A **25**, 978 (1982).

[86] D. de Vos, *Using Predictive Graph Neural Networks to Simulate Patchy Particle Systems [Unpublished master's thesis]*, (2021).

[87] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, and P. Gumbsch, *Structural relaxation made simple*, Physical Review Letters **97**, 170201 (2006).

[88] H. Fan and Y. Yang, *Pointrnn: Point recurrent neural network for moving point cloud processing*, arXiv preprint arXiv:1910.08287 (2019).

[89] T. Hamanaka and A. Onuki, *Heterogeneous dynamics in polycrystal and glass in a binary mixture with changing size dispersity and composition*, Physical Review E **75**, 041503 (2007).

[90] R. Shi, J. Russo, and H. Tanaka, *Origin of the emergent fragile-to-strong transition in supercooled water*, Proceedings of the National Academy of Sciences **115**, 9444 (2018).

[91] F. Caupin and M. A. Anisimov, *Thermodynamics of supercooled and stretched water: Unifying two-structure description and liquid-vapor spinodal*, The Journal of Chemical Physics **151**, 034503 (2019).

APPENDICES

Appendix A

Glossary machine learning

In this Appendix some of the most important terms associated with machine learning will be briefly introduced. For more detailed explanations, we refer the reader to e.g. the textbook in Ref. [71].

- **Activation function**
  The activation function defines how the input of a node in a neural network, consisting of the weighted sum over all the node values in the previous layer together with a bias, is transformed into an output. Its aim is to introduce non-linearity in a neural network. There are different activation functions possible, for example a sigmoid, a hyperbolic tangent, or a Heaviside step function. In this thesis we use the Rectified Linear Unit (ReLU). This function gives zero as an output when the input is negative or zero, while when the input is positive, the output is equal to the input.

- **Backpropagation**
  In order to find the gradient of the loss function, the derivative of this function is calculated with respect to all the parameters of the network. Since the loss function only depends indirectly on the parameters of earlier layers, the partial derivatives are calculated in a backwards manner, i.e. first with respect to the output layer, then the last hidden layer, etc. Due to the fact that this happens in a backwards manner we call it backpropagation.

- **Batch size**
  The batch size is the number of training examples the network analyzes before adjusting the weights and biases during training. In order to increase the learning speed and reduce memory requirements, the batch size is often much smaller than the total amount of training data.

- **Drop-out**
  Drop-out is one strategy to prevent overfitting. In this technique, randomly selected nodes in the network are set to zero during training. This prevents the network from excessively adapting collections of neurons to fit specific variations in the training set.

- **Early stopping**
  In order to prevent overfitting on the training data set, we use early stopping. Every time the machine learning model is updated, we use test data to evaluate how well the model performs. In the end we use the model that yields the highest correlation on the test data. Since this model often is not the last model that is evaluated, we call this early stopping.

- **Epochs**
  When training a neural network, the number of epochs is equal to the number of times the training data set as a whole is passed through the network.

- **Gradient descent**
  Gradient descent is an optimization algorithm that is commonly used in machine learning. The loss function of a neural network depends on all the parameters in the network. In order

to find the minimum of the loss function, the gradient with respect to all these parameters is computed. The parameters are then adjusted in such a way that a step along the gradient is taken, an approach that is called gradient descent.

- **Hidden Layers**
  A neural network has multiple layers of neurons. The hidden layers are the layers that lie between the input and the output layer. In the case that there are no hidden layers, a neural network essentially reduces to linear regression (except for the possible non-linear activation function).

- **Hyperparameters**
  The hyperparameters of a machine learning approach are the parameters that you can tune manually in order to control e.g. the network architecture and the learning process. They include parameters like batch size, learning rate and number of hidden layers.

- **Learning rate**
  The learning rate is a hyperparameter used in training algorithms such as gradient descent. In order to train a neural network, the weights and biases of the network are adjusted in the direction of the gradient of the loss function. The learning rate determines how big the steps are that are taken along this gradient.

- **Loss function (also referred to as cost function)** In order to train a machine learning model, we need a method that evaluates how far the predictions lie from the measured data. The function that we use to measure this is what we call the loss function. There are multiple options for loss functions; one that is used very often, also in this thesis, is the the mean squared error (MSE), but other options are for example the mean absolute error and Bayesian minimum mean-square error.

- **Parameter**
  When training a machine learning model, the parameters are all the variables in the model that are adjusted while learning, i.e. the weights and biases. The parameters are adjusted by the network, meaning that contrary to the hyperparameters, we do not have a direct influence on the their values.

- **Regularization**
  Regularization is a broad term for methods that are used to avoid overfitting. Options include dropout and early stopping, but also adding a penalty in the loss function to penalize large weights. Two common options to add a penalty are Ridge and Lasso regression. In the first case the penalty is proportional to the square of the weights, and in the second case the penalty is equal to the absolute value of the weights.

- **Supervised machine learning**
  Supervised machine learning is a class of machine learning, where the machine learning algorithm is provided with both input parameters as well as the desired output parameters. This means that while training, the model knows what the predicted output should look like. In contrast, in the case of unsupervised machine learning, the model is only provided with the input parameters, meaning that the algorithm itself must try to find patterns in the input data. In this thesis we only use supervised machine learning.

- **Overfitting** The goal of training e.g. a neural network, is to have a model that performs as well as possible for data that it has not yet seen, i.e. we want the model to generalize to new data easily. Whenever the model performs very well on training data, but not on new data, we say that the model overfits. In order to avoid overfitting, we use regularization.

Appendix B

Validating structural parameters

In order to make sure that the structural parameters discussed in Section 2.2 and the linear regression algorithm were implemented correctly, we checked the results from our linear regression against the results obtained in the paper of Boattini *et al* [35], which examined the same system as the one studied here. The only difference between the performance shown in Figure B.1 and the performance shown in Figure 5.1, is that the dynamic propensity of the particles for which the predictions were made in the latter case, was obtained by averaging over 100 trajectories, instead of 50. The results for both the small and large particles can be found in Figure B.1. As one can see, they agree very well.

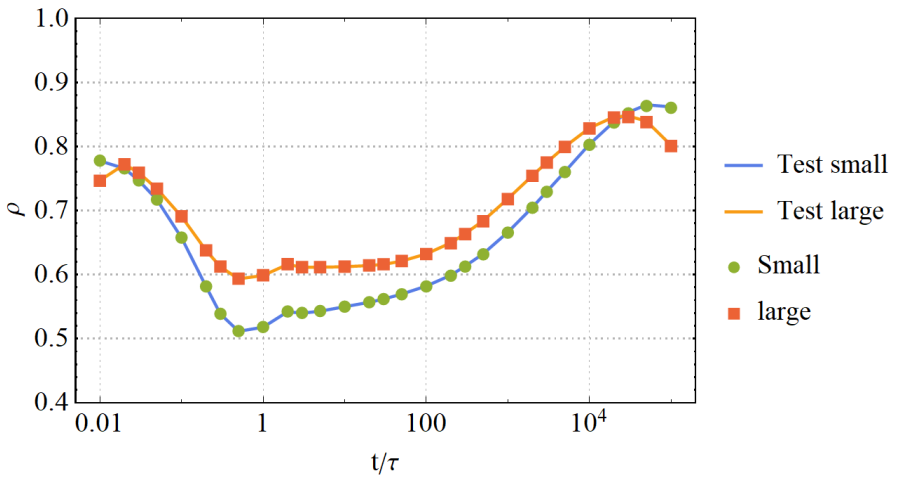


Figure B.1: Pearson correlation coefficient between measured and predicted dynamic propensity values over time as analyzed with a trained linear regression model. Blue and orange lines denote the Pearson coefficient as found by Boattini *et al* [35] for respectively large and small particles. Green and red points indicate results from this thesis.

In Chapter 6.2.2 we stated that after quenching the system for $t/\tau = 10$, with τ an arbitrary unit of time, we reached equilibrium. In Figure 6.2 we show that this is indeed true. In the figure we plot the potential energy as a function of time for ten different systems up to $t/\tau = 10$. As we can see the potential energy eventually becomes a constant as a function of time for all systems, meaning that quenching for $t/\tau = 10$ is sufficient.

Appendix C

Further improvement attempts

Further attempts to improve propensity prediction using LR

C.1 Fitting a polynomial through the predicted propensities

In Chapter 6 we discussed various techniques to improve the propensity predicted using linear regression. In this appendix we discuss one extra method that we tried. Unlike the other improvement techniques described, this method does not add extra structural information, but instead post-processes the data after linear regression is applied. Since this is something rather different from the techniques used in Chapter 6 we decided not to include it in that chapter.

To illustrate this approach, we turn to a scatter plot of the predicted and measured propensity at $t = 10^4$, which can be found in Figure C.1. In this figure one can see that for particles that have low measured propensity, the model tends to overestimate the prediction, while for particles that have a large measured propensity, linear regression tends to underestimate.
In order to remove the curvature we tried to fit a fourth order polynomial through the data points in the scatter plot of the training data. As an input for this polynomial we used the predicted propensity, while the output was the measured propensity. Afterwards we then applied this fitted polynomial on the test data. In Figure C.2 we show the performance of linear regression based with and without polynomial fit. As one can see the improvement is minimal but consistent over time.

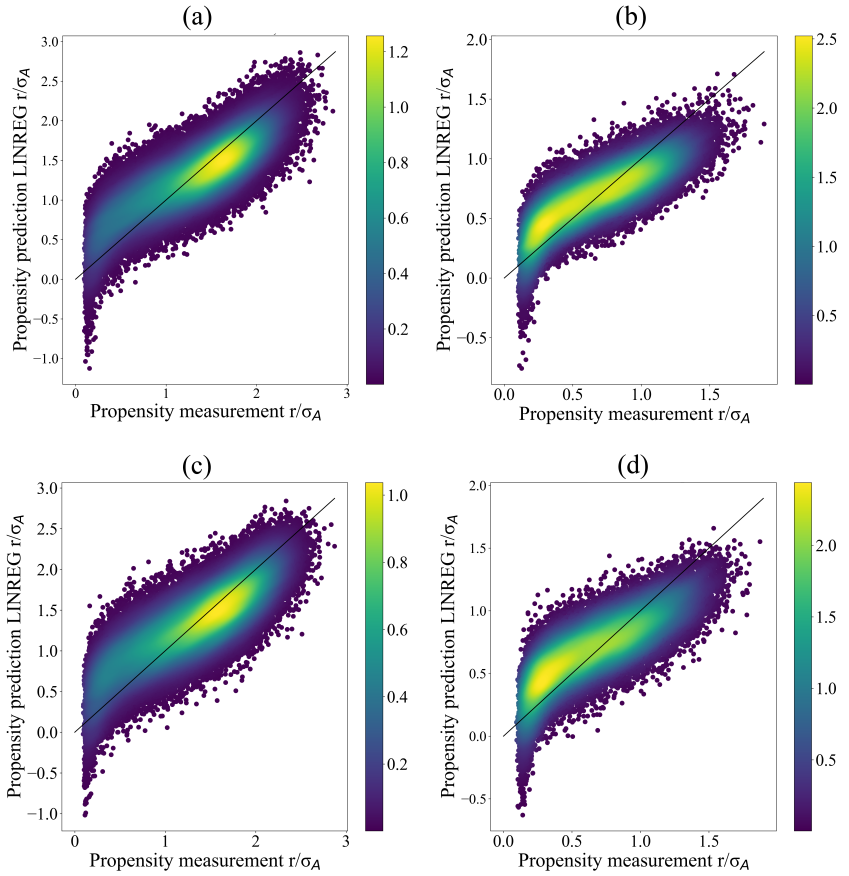


Figure C.1: Linear regression performance trained at 25 snapshots at time $t/\tau = 10^4$ for packing fraction $\eta = 0.58$ and 3 generations of input parameters. We show the models performance for training data on small particles (panel (a)), training data on large particles (panel (b)), test data on small particles (panel (c)), test data on large particles (panel (d)). The colors of the points in the scatter plot represent the local number density of points. The point clouds are associated with the following correlations: $r^2_{(a)} = 0.70$, $r^2_{(b)} = 0.67$, $r^2_{(c)} = 0.68$, $r^2_{(d)} = 0.64$.

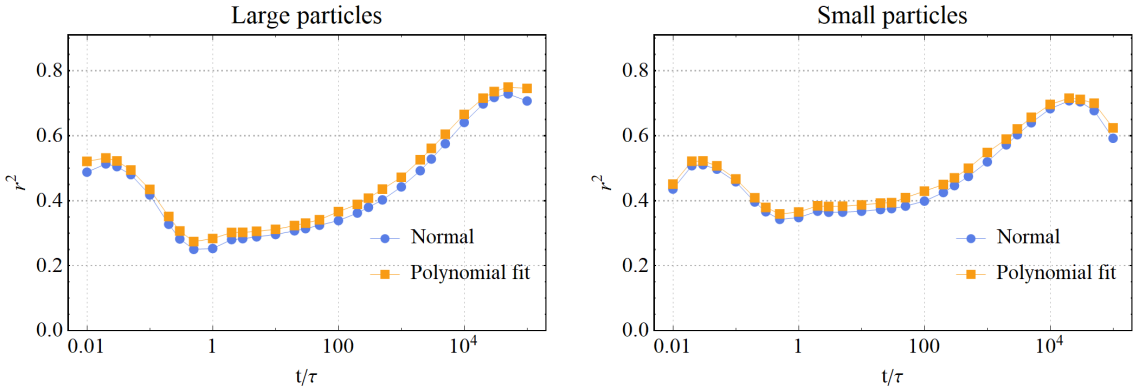


Figure C.2: Linear regression performance in terms of the r^2-coefficient, based on three generations of structural order parameters, with and without fourth-order polynomial fit.

C.2 Measure inertia tensor of caged particle movements

In Chapter 7 we showed that we can predict the cage center of a particle by measuring the average position of restricted particles in an MC simulation. In addition we also examined whether the spread of a particle's position during this simulation holds information about the propensity.

In order to capture the spread in the position of a particle in terms of rotationally invariant parameters, we measured the point cloud associated with the positions of a particle at discrete time intervals with respect to the initial position and then looked at the associated inertia tensor. This inertia tensor is defined as

$$\mathbf{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz}, \end{pmatrix} \tag{C.1}$$

with $I_{ii} \equiv \sum(r_j^2 + r_k^2)$ and $I_{ij} \equiv -\sum r_i r_j$. Here, $\mathbf{r} = (r_x, r_y, r_z)$ is the position of a particle and the summation runs over all the different positions measured for a specific particle. Although the inertia tensor itself is not rotationally invariant, the associated eigenvalues are. These eigenvalues capture the magnitude of the fluctuations of a particles position along the three principal axes of the point cloud and are therefore a measure for the fluctuations in a particles position. Similar to how we constructed the higher generations of order parameters, we also constructed higher-order generations of eigenvalues by averaging the inertia tensor over neighbouring particles, after which we computed the eigenvalues. The way that we averaged the inertia tensor is exactly equivalent to the method described in Chapter 2.2.

To examine how much information the inertia eigenvalues hold about the propensity, we train two linear regression models. The first regression model is provided with 3 generations of inertia tensor eigenvalues. In order to make sure that the model is rationally invariant, for each particle and each generation the eigenvalues are always sorted from large to small. The second linear regression model is trained on both 3 generations of inertia tensor eigenvalues, as well as the normal structural parameters. The training data and the test data both consist of 50 snapshots with each 2000 particles. The inertia tensor is evaluated using an MC simulation of $5 \cdot 10^4$ initial steps and $1 \cdot 10^5$ measuring steps where every 100 steps the position of each particle is recorded. The positions of particles are again restricted to a spherical volume equal to their own particle size.

In Figure C.3 we show the results, where we include the prediction based on only the normal coordinates (blue dashed line), the prediction based on solely the eigenvalues (blue line), the prediction based on eigenvalues and normal parameters (orange line) and as a comparison the prediction based on Monte Carlo ensemble coordinates as shown in Chapter 7 (red line). In this figure we see that the eigenvalues of the inertia tensor hold almost the same information about the propensity during the caging regime as prediction based on the cage center coordinates. This is not very surprising: Since we measure the point cloud with respect to the initial position, the main axis of the cloud will likely be parallel to the vector connecting the initial position and the the middle of the point cloud (i.e. the cage center). Therefore the first eigenvalue will indirectly reflect the distance between initial position and cage center. The fact that for later times, the the prediction based on Monte Carlo coordinates surpasses the other two, is because for its prediction it also uses the parameters based on caged coordinates; Information that the other predictions do not use.

From the blue line, we furthermore see that the eigenvalues also hold some information about the propensity for times after the caging regime. Closer examination shows that this information is mostly found in the higher order generations of eigenvalues. However, we also see that this information is already captured also by our normal order parameters; if we combine the inertia tensor eigenvalues with the normal order parameters (orange line) we see that for large times the prediction does not exceed the prediction that is made with normal parameters only.

For future research it could be interesting to measure the inertia tensor with respect to the cage

center instead of the initial positions. That way we can deduce whether there is useful information in the cage-fluctuations for prediction motion at times after the caging regime.

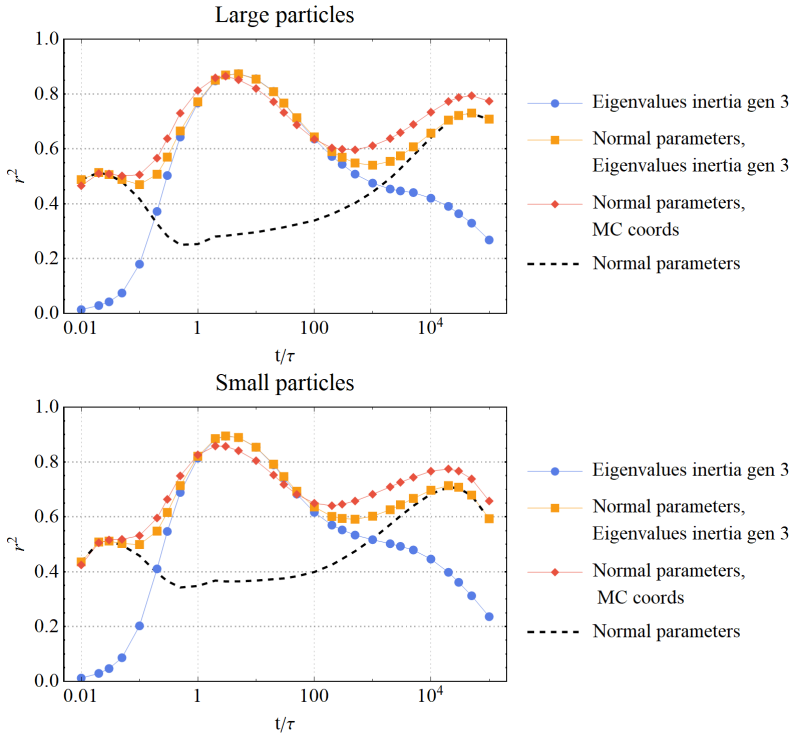
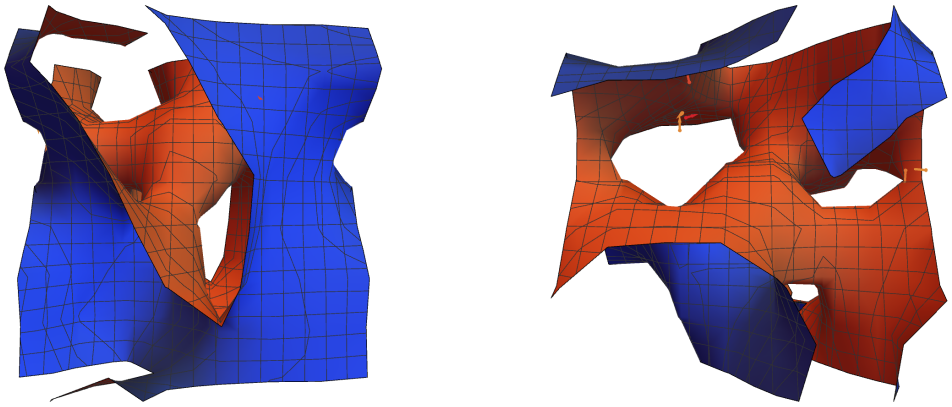


Figure C.3: Linear regression performance in terms of the r^2-coefficient for large and small particles. The three different lines represent the performance based on 3 generations of structural order parameters, 3 generations of inertia tensor eigenvalues and 3 generations of both structural parameters and inertia tensor eigenvalues.

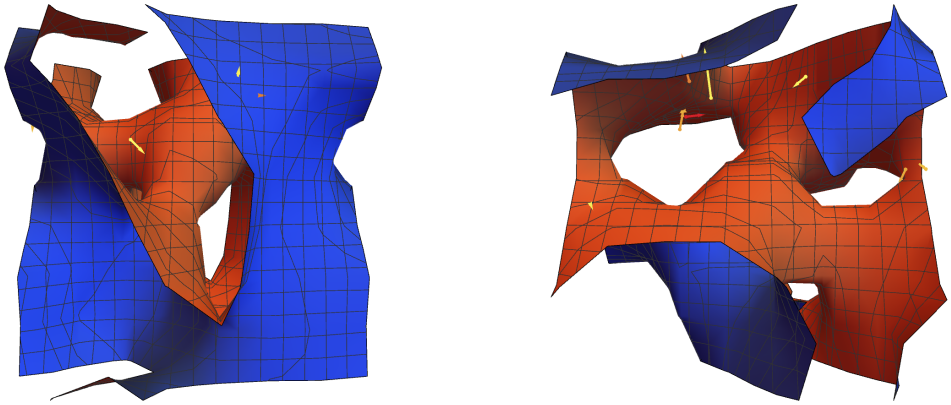
Appendix D

Location of early escaped particles

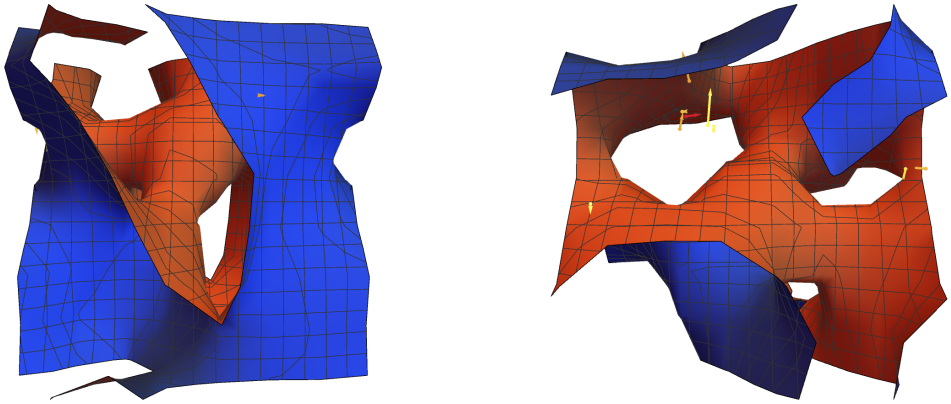
In Chapter 9 we showed that particles that on average have escaped their cage after a simulation time of $t/\tau = 50$ are preferentially located near the interface between the fast and slow moving regions. Here, we show that this statement is not strongly sensitive to this choice of time. In Figure D.1 we show for one system, the DSP (see Equation (2.3)) of particles that escape the earliest as seen from two different points of view (following the definition that particles have escaped when the have on average moved further out than $2r_{\text{cage}}$, with $r_{\text{cage}} \approx 0.14$). In the figure we also plot the boundary surface as defined in Equation (9.4). We see that roughly between $t/\tau = 20$ tot $t/\tau = 200$ the statement that early escaped particles lie in the vicinity of the boundary surface holds.



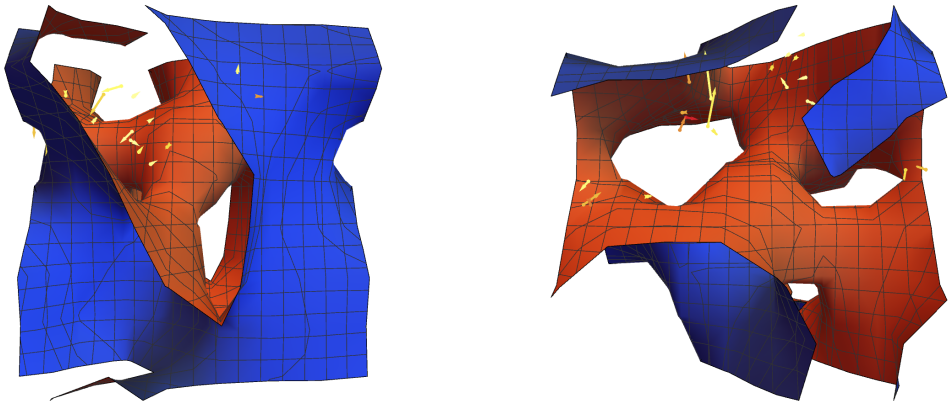
(a) We plot the DSP vectors associated with escaped particles at $t/\tau = 10$ as seen from two different points of view. A particle is said to have escaped, when on average it has moved further than 2 times the average cage size $r_{\text{cage}} \approx 0.14$. The DSP vectors are scaled by a factor of two to increase visibility and the associated colors hold information about the relative propensity (red vectors are associated with a high propensity, while blue vectors are associated with a low propensity). The displayed surface is defined by Equation (9.4). On the red side of the surface, particles associated with the fast moving area are located, while on the blue side slow moving particles are situated.



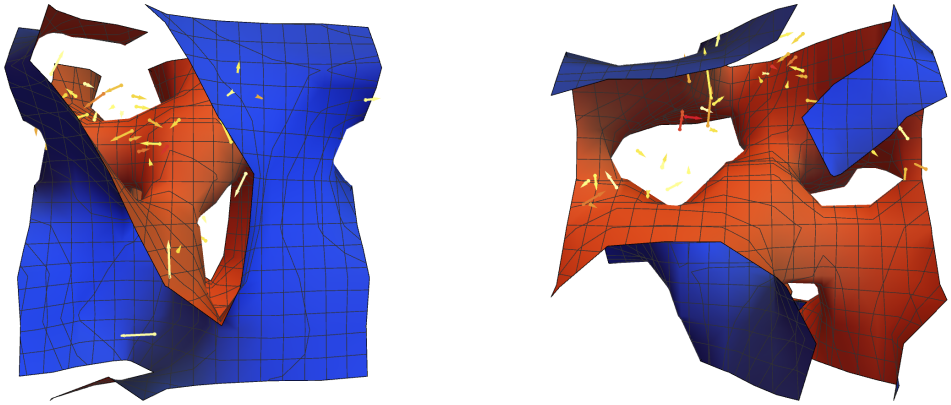
(b) Same system as in Figure D.1a, displayed at $t/\tau = 20$.



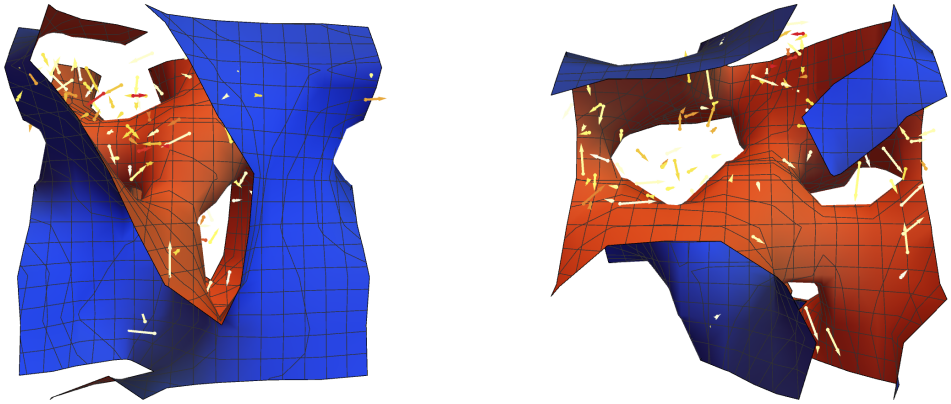
(c) Same system as in Figure D.1a, displayed at $t/\tau = 30$.



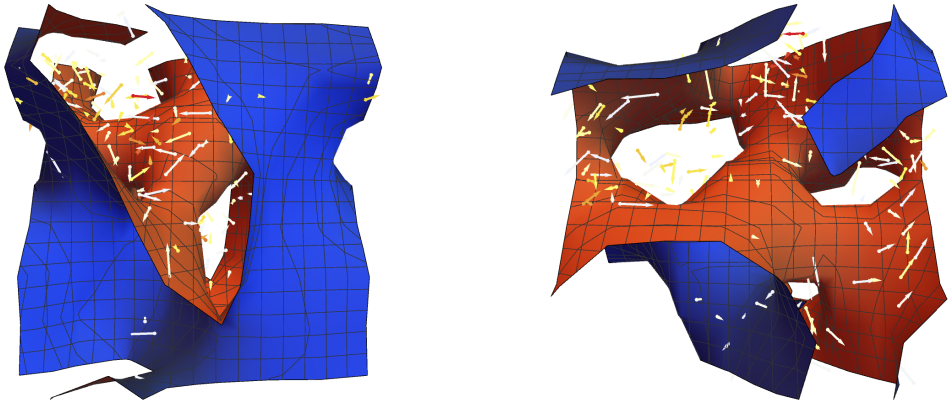
(d) Same as Figure D.1a, however now at $t/\tau = 50$.



(e) Same system as in Figure D.1a, displayed at $t/\tau = 100$.



(f) Same system as in Figure D.1a, displayed at $t/\tau = 200$.



(g) Same system as in Figure D.1a, displayed at $t/\tau = 300$.

Figure D.1: Spatial location of the early escaped particles with respect to the boundary surface between fast and slow moving regions, displayed for several times between $t/\tau = 10$ to $t/\tau = 300$.