University of Southern Denmark

A bachelor project in

Modelling and simulation of pedestrian behaviour

Author:
Nicky C. Mattsson
Supervisors:
Kristian Debrabant
Carsten Svaneborg

Web pages:
imada.sdu.dk/~nmatt11
imada.sdu.dk/~debrabant/index_en.php
http://www.zqex.dk

January 2, 2015

Contents

1 Introduction

Some of the precautions we encounter in our everyday life are results of the officials trying to control how crowds of people move and behave, and by that try to prevent panic induced casualties. For instance, one might consider the queuing system at Theme parks: If the officials did not guide the crowd and control the queue, everyone would try to come first. Another characteristic, which describes the difficulty of simulating pedestrian behaviour, that is there is typically nothing to guide the pedestrians. If we consider cars and roads, then we have signs and lines on the road to tell people where to drive. This is not the case with pedestrians there is nothing to tell them, where and how they should move, we try using fences, however whenever panic starts, it often breaks down those fences.

To this end, the research has been two fold as you are interested in two basic properties of the model: efficiency and precision. This has made the research go in two directions: A microscopic direction, where each pedestrian is simulated how to move, and a macroscopic, where densities of people are of interest and not the placement of a single pedestrian. The problem is, however, that the microscopic models are in most cases the most precise models, but you have to calculate the force on every pedestrian, making it ineffective.

The macroscopic model, which is resulting in a partial differential equation (PDE) model is efficient as the time complexity solving the PDE, with V mesh points, to an acceptable tolerance is about $O(V)$[1] by using the method of generalised minimal residual, but all this is at the expense of precision. Another aspect is that macroscopic methods only consider this problem on large scales - typically the interesting length scale is the size of the pedestrians. Using microscopic models we can consider what happens at doors and other bottlenecks which is at the same length scale as the pedestrians, which is of significantly more interest, than how pedestrians can flee without serious constraints.

In this project I will focus on the microscopic approach and therefore extend the existing models, in an attempt to implement group behaviour. Further, instead of creating new techniques and methods for pedestrian behaviour, I would like to use already existing methods used in statistical mechanics to simulate pedestrian behaviour.

Another important aspect of this thesis is to address the problem of runtime or complexity of microscopic methods. I will therefore investigate the actual runtime of a standard problem, using different methods to solve the system of differential equations.

In this thesis we will first consider the basic microscopic model, which is used to model how pedestrians flee. This model is a second order differential equation and depending on how you choose the forces it will become a stochastic differential equation. This then requires us to be able to integrate such systems both with and without stochastic behaviour properly. Therefore we have first included a small section on existence and uniqueness of solutions of ordinary differential equations, how one can reduce the error made by numerical methods for ordinary differential equation and how to treat "stiff" differential equations. Secondly we have an entire section containing an introduction to stochastic differential equations, existence and uniqueness, and a derivation of a new algorithm proposed by N. Grønbech Jensen and O. Farago, which we will use for properly integration of the resulting system of stochastic differential equations. Now knowing how to treat stochastic differential equations we will first redo simulations from two articles made by D. Helbing, I. Farkas and T. Vicsek. After doing this we will make our attempt to extend the basic model using group behaviour. After this thorough discussion of the models and how to extend the models, we will take another view of the problem and consider the problem of the runtime or time complexity, to see if we can discover what destroys it.

[1] This is under the assumption that the matrix that we should invert is sparse, which is a reasonable assumption when we are working with partial differential equations

2 The Basic Model

The primary idea of the microscopic method, is that we will consider the force acting upon each pedestrian. We will in the standard approach assume that we are dealing with deterministic trajectories. This can be described by Newton's second law of motion:

$$f_i = m_i \cdot \frac{d^2 s_i}{dt^2} \quad i = 1, 2, \ldots, n \tag{2.1}$$

where $s_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$, m_i is the mass of the pedestrians i, n is the number of pedestrians. Here i describes the i's pedestrian.

So even though we are working in 2 dimensions, we can easily reduce it to a system of $2n$ second order ordinary differential equations or a system of $4n$ first order ordinary differential equations by using standard methods. The quantity of interest is the position of the pedestrian, s_i, at any time, t.

We will try to describe which forces each pedestrian is acted upon. Different models use different forces, but all models are based on the same fundamental physical principles, which are described in [1] which we will use as the essential baseline. Other models considered later in this thesis also use this base, see for instance the articles [3] and [2]. These basic forces that we are going to use can be broken down into:

- A driving force
- A repulsive force from the wall
- A repulsive force from other pedestrians

Let us consider each of the above three terms independently.

2.1 A Driving Force

The driving force is the force which models the desire of a pedestrian to move towards the exit, or what they believe to be the exit. This force depends on the position of the pedestrian, as they want to adjust their movement such that they try to move towards the location of the door. We desire that if a pedestrian can move freely, then the pedestrians will not keep on accelerating, but will reach some speed[2] and continue with this speed toward the location of the door. This makes the driving force dependant on velocity too, in physics this can be interpreted as a drag. All this typically results in the following force:

$$f_i^{driving} = m_i \cdot \frac{v_0 e_i - v_i}{\tau}, \tag{2.2}$$

where m_i is the mass, v_i is the current velocity of the i'th particle, τ is a characteristic time and e_i and v_0 is the direction and speed in which the pedestrian moves under no constraints[3].

The complexity of a system evolving according to this force is $O(n)$, as every pedestrian has a driving force, and the driving force does not depend on any other pedestrian.

[2]Note the difference between the use of speed and velocity. Velocity is a vector and speed is the length of it.

[3]That they are moving without constraints is also called free flow

2.2 A Repulsive Force to the Wall

Panicking situations often happen as a result of having too many people stuffed in a too small place, each pedestrian will tend to move away from the walls or will at least try not to run into them. This results in a repulsive force from all present walls, giving the following force:

$$f_i^{wall} = \sum_{w \in W} f_{iw}(s_i), \tag{2.3}$$

where W is the set of all present walls and $f_{iw}(s_i)$ is the repulsive force from the wall w, which depends only on the position of the pedestrian relative to the wall. This could for instance be the exponential function taken to minus the distance or simply the reciprocal of the distance. We just require that if the distance gets smaller, then the force increases and vice versa. Often the pedestrian is assumed to have a size, so the distance used is typically the distance from the center of the pedestrian to the wall minus the radius. For a more efficient computation a cut-off distance is often set, such that if the distance between the pedestrian and the wall is more than this distance, then the force is assumed to be 0.

The complexity of this force is $O(n \cdot |W|)$, assuming that the number of walls closer to the pedestrians than the cut-off is much smaller than the number of pedestrians[4], the complexity reduces to $O(n)$.

2.3 A Repulsive Force to All Other Pedestrians

As a pedestrian is neither interested in running into another they would preferably like to keep some distance between them, which is the reason why this is often referred to as the *social force*. The social force is of the form:

$$f_i^{social} = \sum_{i \neq j} f_{ij}(s_i), \tag{2.4}$$

where f_{ij} is a function depending at least on the positions of the pedestrian i. Often this force is of the same form as f_{iw}, with just the distance to pedestrians j instead of the distance to the wall w. As before, it is typically assumed that the pedestrians have a radius and the distance used is the distance from center of pedestrian i to center of pedestrian j minus the sum of their radii. As the form of this force is typically the same as the one of the force to the wall, the same cut-off is typically used.

The complexity here is $O(n^2)$ as for every pedestrian we have to calculate the force between every other $n-1$ pedestrian. By using symmetry, one may reduce the constant as the force between particle $i-j$ is the same as between $j-i$ with a change of sign, another way to reduce the constant is to use a Verlet list [17] to keep track of which pedestrians is within a certain cutoff distance, if the distance is larger then the force is negligible

2.4 In Total

The total force is:

$$f_i = f_i^{driving} + f_i^{wall} + f_i^{social},$$

which becomes:

$$m_i \cdot \frac{d^2 s_i}{dt^2} = m_i \cdot \frac{v_0 e_i - v_i}{\tau} + \sum_{w \in W} f_{iw}(s_i) + \sum_{i \neq j} f_{ij}(s_i). \tag{2.5}$$

The total complexity is now $O(n + n + n^2)$ which is the same as $O(n^2)$.

[4]Often the number of walls within the cut-off is 2 (the pedestrian is in the corner) or less, so this is a valid assumption

3 Ordinary Differential Equations

To be able to solve and work with the resulting system once we insert specific forces in Equation (2.5), we want an existence and uniqueness theorem, which is provided by C. Picard and E. Lindelöf, also as we are dealing with quite a complex system of differential equations, we would like to solve it numerically. We will therefore discuss what order of convergence means in the sense of how fast a method converges and different methods to lower the error that we are making.

If nothing else is stated we will consider the following (system of) initial value problem(s). Note that we will later refer to this simply as an initial value problem or IVP without declaring whether it is a system of equations or just one equation:

$$y' = f(t, y(t)), \quad y(\xi) = \eta \tag{3.1}$$

here y' is shorthand notation for time derivative of the variable y, t is the time and $f(t, y)$ is a function $f : [a, b] \times \mathbb{R}^d \to \mathbb{R}^d$

Note that the theory that we develop for the above IVP, also covers higher order differential equations as we can reduce the order by increasing the system. For instance for a second order differential equation we add the equations $y' = v$, and make a change of variables, in the original equation such that $y \to y$, $y' \to v$ and $y'' \to v'$. Then we have a system of first order differential equations in y and v.

3.1 Existence and Uniqueness

The existence and uniqueness of solutions of differential equations can be found in most textbooks on the area, one place to find it is in the book by W. Boyce and J. Brennan [20].

Consider the IVP defined by Equation (3.1), and the existence and uniqueness theorem by C. Picard and E. Lindelöf

Theorem 3.1. *Let $f(t, y)$ be as in Equation (3.1) and in addition let it fulfil a Lipschitz condition of the form:*

$$\|f(t, y) - f(t, \hat{y})\| \leq L\|y - \hat{y}\| \tag{3.2}$$

with a constant $L > 0$ for all $(t, y), (t, \hat{y}) \in [a, b] \times \mathbb{R}^d$. Then there exists on $[a, b]$ for each initial value $(\xi, \eta) \in [a, b] \times \mathbb{R}^d$ a unique solution of the IVP, Equation (3.1).

Proof. Assuming that $y \in C^1([a, b])$, then the IVP is equivalent to solve the integral equation given by:

$$y(\tau) = \eta + \int_\xi^\tau f(t, y(t))\mathrm{d}t =: (Ay)(x) \quad \forall \tau \in [a, b], \tag{3.3}$$

thus we want to solve the problem given by:

$$y = (Ay)(\tau), \quad y \in [a, b] \tag{3.4}$$

As we know that the space of continuous functions equipped with the supremum norm is a complete metric space we only need that A being a contraction. Because, then we can use Banach's fixpoint theorem to give us the existence and uniqueness of the above fix point iteration. That A is a contraction we see by considering:

$$\|(Ay)(\tau) - (A\hat{y})(\tau)\| \leq \| \int_\xi^\tau [f(t, y(t)) - f(t, \hat{y}(t))]\, \mathrm{d}t\|,$$

by using the Lipschitz condition specified in the theorem we get

$$\| \int_{\xi}^{\tau} [f(t, y(t)) - f(t, \hat{y}(t))] \, \mathrm{d}t \| \leq L \| \int_{\xi}^{\tau} y(t) - \hat{y}(t) \mathrm{d}t \| \leq L(b-a) \| y(t) - \hat{y}(t) \|_{\infty}. \tag{3.5}$$

Where the $\| \cdot \|_{\infty}$ is the usual infinity norm, which takes the supremum of the argument. So A is a contraction if $L(b-a) < 1$. However by choosing the following norm:

$$\| y(t) \|_{L} = \sup_{t \in [a,b]} e^{-L|x-\xi|} \| y(t) \| \tag{3.6}$$

which is Lipschitz equivalent to the standard supremum norm we used before, one can easily show by doing the same calculations again that the constraint $L(b-a) < 1$ can be dropped as the new constant is guaranteed to be less than one. Which finishes the proof. □

From topology we know that if such a fixed point iteration converges for initial value $t = \xi$, then if $f(t, y(t))$ is continuous it also converges for $t \in [\xi - \varepsilon, \xi + \varepsilon]$ for some $\varepsilon > 0$. So we can loosen up the conditions a bit, making us able to put pieces together on the timeline where the FPI converges. So if the solution for instance does not exist at $t = 0$[5] then we can still say something about the existence and uniqueness on $\mathbb{R} \backslash \{0\}$.

3.2 Numerics

Let us first consider the initial value problem given in Equation (3.1) in a slightly changed version:

$$y' = f(t, y(t)), \quad y(0) = \eta, \tag{3.7}$$

this as we saw in the proof of the existence and uniqueness theorem is equivalent to the said integral equation.

$$y(\tau) = \eta + \int_{0}^{\tau} f(t, y(t)) dt \tag{3.8}$$

We are now interested in approximating the integral. This can be done in a wide range of ways, one way which builds on top of quadrature formulas is called collocation methods, here one applies a quadrature formulas directly to the IVP and obtains an equation one needs to solve[6]. An alternative to this is either using Taylor methods, where one just creates a Taylor expansion or one of the most typically used methods is the class of Runge - Kutta methods, which we will consider a bit later. Note that these methods are no way exhaustive and that methods can be classified in multiple classes, the classification is just different ways of considering the methods.

Common for all of the numerical methods is that we are interested in the order of convergence, or in other words, how fast does the method converge? For this we have the following definition:

Definition 3.1. *On the discretized interval with step-size dt, let e_j denote the error between the discretized solution y_j and the exact solution $y(x_j)$ of the IVP Equation (3.1), further let E denote the maximum error on that interval, such that:*

$$E = \max_{j} \| e_j \| \tag{3.9}$$

Then the method is called convergent if and only if $\lim_{dt \to 0} E = 0$ and called convergent of order p if the method in addition fulfils that:

$$\exists C > 0 : \forall dt > 0 : E \leq C(dt)^p. \tag{3.10}$$

[5] Could be that the solution was $y(t) = \frac{1}{t}$

[6] This equation is typically non linear but its solution can typically be approximated using Newton-Raphsons method.

We can now see that we can reduce the error that we make by either increasing the order[7], reducing the time stepping or reducing the constant.

To construct higher order methods one uses a very important result which says that under some Lipschitz conditions the error in Equation (3.10) corresponds to the error term that remains in the difference between the Taylor expansion of the exact solution and the numerical approximation[8][9]. Using this we can also see that it makes sense to use Taylor methods as these are guaranteed to converge (under the Lipschitz constraint) of the same order as the Taylor expansion of the problem. Whenever one makes a Taylor expansion, one has to find the derivatives of the problem which can seem kind of strange and waste of computing power as the ultimate goal is to approximate the integral in Equation (3.1). To circumvent this one uses the ideas of C. Runge and M. W. Kutta, they showed that using multiple function evaluations of the integrand in various points one can make arbitrarily high order methods without having to differentiate. This was later refined by first J. C. Butcher and later on by E. Hairer and G. Wanner who introduced a general scheme for creating higher order methods using Runge-Kutta methods.

Another way to reduce the error is to reduce the step-size, however reducing the step-size will also increase the runtime of the problem. So what is typically done is to use a so called adaptive step-size. Here one calculates an error estimate and if this estimate is too large then the step-size is lowered, alternatively if the error is "too" small then we increase the step-size. There are different methods to estimate the error and the most efficient method is of course a method that reuses what we already have calculated such that the computing time needed in addition is as low as possible. One way to do it is to use two methods of different orders for instance 4 and 5 and then use the difference between these two methods as the error estimate. J. R. Dormand and P. J. Prince [23] created such a method where the function evaluations of the fourth order method are a part of the fifth order method, making it almost free to estimate the error. This is currently the standard method in Matlab implemented as the ode45 method.

A final option is to reduce the constant. In contrast to the other two ideas this is problem specific and you have to figure out what type of problem you are dealing with in order to choose an appropriate method, which has such properties. Also note that the constant is not just problem dependant but it does also depend on the norm one uses to calculate the maximum error. What we will consider here are the so called stiff equations and how one can solve them appropriately.

Stiff equations

The problem of stiffness was first described by C. F. Curtiss and J. O. Hirschfelder where they called a specific class of chemical reactions for stiff. What they observed was that explicit Runge-Kutta methods would not give them an appropriate solution for the normal choice of step-size and using adaptive step-sizes would make the solver run considerably longer (small step-sizes) to make it up for the supposedly large error constant. Now we do however use the term stiff for a much wider

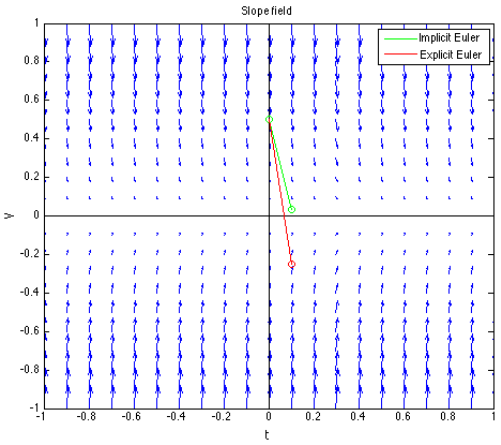


Figure 1: Vector field of $y' = -15y$

[7] Assumed that $dt < 1$, which is nearly always the case.

[8] This last part about the difference between the exact solution and the numerical approximation is also called consistency.

[9] This result is called "Central convergence theorem for one-step methods" and is used in many textbooks in the area of numerical analysis of Ordinary differential equations

class of equations where explicit methods do not
work properly.

As in the problem C. F. Curtiss and J. O. Hirschfelder encountered, the problem of stiff equations typically arises for equations with fast changing values around some equilibrium. To understand why, consider the Figure 1 where we have plotted the vectorfield corresponding to the differential equation $y' = -15y$. We here see that the differential equation has an equilibrium at $y = 0$ and that the slope of the field is fast changing around this line. If you now consider one arrow on the line above the equilibrium and make one step with some explicit method this means that we are to use this slope in this point (the arrow) and use this to move forward. As this slope is large we are very likely to overshoot the equilibrium, then we are to do it again but from below, but also here we will overshoot the equilibrium so the approximative solution will oscillate around the equilibrium. If we instead consider implicit methods (for instance the implicit Euler method), then we are to evaluate the method in the other end point, i.e. answer the question of where will we end? As this point will be closer to the equilibrium and on the same side, repeating this we will get closer and closer to the equilibrium, without the method is oscillating. The method is said to have a dampening effect. Note that not all implicit methods have this good behaviour, but none of the explicit methods have it.

To investigate how stable the method is, G. Dahlquist argued that using a specific test function, which was chosen to be $y' = \lambda y$ with $Re(\lambda) < 0$, then the qualitative behaviour of the exact solution $y(t) = y_0 e^{\lambda t}$ should reasonably coincide with the numerical solution. In this case two important properties were observed and if the numerical method would have the same properties, then it would be called either A-stable if it fulfils the first or L-stable if it fulfils both. To see the two properties consider one step of the exact solution:

$$y(t_m + dt) = y(t_m)e^{\lambda dt}.$$

Now as $Re(\lambda) < 0$ the exponential part is less than 1, and we can therefore make the following bound:

$$|y(t_m + dt)| \leq |y(t_m)| \tag{3.11}$$

We can also see that by letting $Re(\lambda dt) \to -\infty$ then we get:

$$\lim_{Re(\lambda dt)\to-\infty} y(dt + t_m) = 0. \tag{3.12}$$

Now if we consider a general one step method, then we know that it can be rewritten into the form:

$$y_{m+1} = R(z)y_m, \quad z = \lambda dt. \tag{3.13}$$

We say that the numerical solver is A-stable if Equation (3.11) is mimicked i.e if the below is true:

$$R : \{\forall \lambda \in \mathbb{C}_-\} \times \{\forall dt \in \mathbb{R}_+\} \to (-\infty, 1).$$

If it in addition mimics Equation (3.12) the method is called L-stable, i.e. if:

$$\lim_{Re(z)\to-\infty} R(z) = 0$$

4 Stochastic Numerical Integration

It will also be necessary to be able to integrate a stochastic system. For these systems one cannot just use basic Runge-Kutta methods (see for instance Chapter 2 in [19]). Therefore this section is dedicated to explain how to integrate stochastic systems properly. As an extension, a new [8] method has been proposed by N. Grønbech Jense and O. Farago, which will also be discussed in the end, this is also the method that I will use for the simulations performed in Large Atomic Molecular Massively Parallel Simulator (LAMMPS) [21], Section 6. I will limit the discussion to methods which can be used on stochastic differential equations (SDEs) with Brownian motion. I will for the matter of convenience change notation for the Brownian motion to be B_t or even $B_t(\omega)$ to emphasize that it is a realization of a Brownian motion at time t. Further, if not anything else is stated, then we are working on a fixed probability space, $(\Omega, \mathcal{F}, P)$, where Ω is the probability space, $\mathcal{F}$ is the σ-algebra and P is the probability measure.

4.1 Construction of the Brownian Motion

This section is based on the corresponding section in the book by I. Karatzas and S.E. Shreve [13].

First, we need a general definition of the white noise that we are using. The Brownian motion or the Wiener process are both names for the same stochastic process, which we will now define:

Definition 4.1 (The normalized Brownian motion)**.** *A real stochastic process $B_{t\geq 0}$ is called a normalized Brownian motion (Wiener process) starting in* 0 *if the following is satisfied:*

- $P(B_0 = 0) = 1$
- $B_t - B_s \sim N(0, t-s) : \forall t, s$ *given that* $t > s$ *(Stationary increments)*
- *All increments are stochastically independent, given that the time intervals are disjoint*

We are now interested in showing that such a stochastic process exists, also please note that since we are dealing with increments that are normally distributed, one can easily transform a normalized Brownian motion into a general one, which is why we will only consider the normalized one. The next theorem does state the existence of such stochastic processes

Theorem 4.1 (Existence of a continuous Brownian motion)**.** *The Brownian motion defined in Definition 4.1 exists both in the discrete case and the continuous case.*

An outline of the general procedure is the following:

1. Prove that by setting $\mathcal{H}$ equal to the span of independent standard Gaussian variables, and having T, an arbitrary isometry from $L_2(0,\infty) \to \mathcal{H}$ then, $T(1_{[0,t]})$ is a discrete Brownian motion.
2. Having the above, $f_n(s)$ an orthonormal basis and g_n as Gaussian normal variables show that $B_t = \sum_{n=1}^{\infty} g_n \int_0^t f_n(s)\mathrm{d}s$ converges in $L_2(0,\infty)$ and almost surely.
3. Showing that by choosing a specific orthonormal basis, we can construct the continuous Brownian motion.

However before we start proving the result, we will need the following result from Hilbert space theory:

Theorem 4.2. *Let H_1 be a Hilbert space with an orthonormal basis e_n and let f_n be an orthonormal basis for H_2, then the map $T : H_1 \to H_2$ defined by:*

$$Tx = \sum_{n=1}^{\infty} (x, e_n) f_n \quad \textit{for all } x \in H_1 \tag{4.1}$$

is an isometry of H_1 into H_2

Proof of Theorem 4.1. We will use the above outlined procedure.

1.

First let $\mathcal{H} = \overline{span}(g_n)$, where g_n are normal Gaussian variables, we would now like to show that $B_t = T(1_{[0,t]})$ is a Brownian motion, which we do by checking the three requirements in Definition 4.1.

First we need $B_0 = 0$; this is clear as $B_0 = T(1_{[0,0]}) = T(0) = 0$. Then we need the increments to be normally distributed with mean 0 and variance $(t-s)$. We first note $B_t - B_s \in \mathcal{H}$ for all s, t. So it is normally distributed with mean 0. We can now find the variance using $var(x) = E(x^2)$ as $E(x) = 0$. So we have:

$$\int_{\Omega} (B_t - B_s)^2 \mathrm{d}P = \|B_t - B_s\|^2 = \|T(1_{[0,t]}) - T(1_{[0,s]})\| \tag{4.2}$$

As T is an arbitrary isometry, we can now join the two expressions and use $\|T(x)\| = \|x\|$. Thus we get:

$$\|1_{(s,t]}\| = (s - t) \tag{4.3}$$

Finally, as the sets $\{1_{[0,t_1]}, 1_{(t_1,t_2]}, ..., 1_{(t_{n-1},t_n]}\}$ are orthogonal, the isometry of them is still orthogonal. We also know that all linear combinations of them are normally distributed, implying that they are independent.

2.

We can now define the isometry $T : L_2(0, \infty) \to L_2(P)$ by $Tf_n = g_n$ (from Equation (4.1)), then by using 4.2, we get that:

$$Tf = \sum_{n=1}^{\infty} \int_0^{\infty} f(s) f_n(s) \mathrm{d}s \cdot g_n$$

or more importantly if we set $f = 1_{[0,t]}$ we get by the previous:

$$B_t = \sum_{n=1}^{\infty} \int_0^{t} f_n(s) \mathrm{d}s \cdot g_n \tag{4.4}$$

We know it converges in $L_2(P)$ and as all the terms are independent and have mean 0, we know that:

$$\sum_{n=1}^{\infty} E(\int_0^{t} f_n(s) \mathrm{d}s \cdot g_n)^2 = \|B_t\|^2 = t < \infty \tag{4.5}$$

Then it follows that Equation (4.5) converges almost surely as well.

3.

To finish the proof we have to choose a specific orthonormal basis $f_n(s)$, this basis we choose to be the Haar system, defined by:

$$\tilde{h}_1(t) = 1 \text{ for all } t \in [0,1]$$

and for all $k = 0, 1, 2, ...$ and $l = 1, 2, 3, ..., 2^k$

$$\tilde{h}_{2^k+l}(t) = \begin{cases} 1 & \text{if } t \in [(2l-2)2^{-k-1}, (2l-1)2^{-k-1}[\\ -1 & \text{if } t \in [(2l-1)2^{-k-1}, 2l \cdot 2^{-k-1}[\\ 0 & \text{else} \end{cases} \tag{4.6}$$

To get an orthonormal basis for $L_2(0,1)$, we normalise it by setting $h_{2^k+l} = 2^{\frac{k}{2}} \tilde{h}_{2^k+l}$

It is easy to show that this is an orthonormal basis by using the fact that the indicator function of a dyadic interval is dense in $L_2(0,1)$ and using the property that this indicator lies in the span of the Haar system.

Using what we previously found, we now have:

$$B_t = g_1 \int_0^t h_1(s)\mathrm{d}s + \sum_{k=0}^{\infty} \sum_{m=2^k+1}^{2^{k+1}} g_m \int_0^t h_m(s)\mathrm{d}s \tag{4.7}$$

It is now easy to see that for all $2^k < m \leq 2^{k+1}$ we can bound $\sum_{m=2^k+1}^{2^{k+1}} \int_0^t h_m(s)\mathrm{d}s$ from above by $2^{-k/2-1}$. It is also easy to show that Gaussian random variables can be bounded on a set with probability 1.

Then by using the Lebesgue dominated convergence theorem with the counting measure, we know that it converges. So there exists a continuous version of the Brownian motion on the interval $[0,1]$ but we can easily shift everything to account for $[n-1, n]$ for all $n \in \mathbb{N}$ and by putting everything together we have a continuous Brownian motion on $L_2(0,\infty)$ □

We do now know that the Brownian motion exists, the second thing that we are interested in is that we can actually integrate with respect to the Brownian motion, for this we are going to use the Itô integral.

4.2 The Itô Integral

This section is based on the corresponding section in the book by B. Øksendal [14].

In the following, I would like to show how the Itô integral is defined and sketch how to show that if the integrand is a progressively measurable function then there is a continuous version of the Itô integral.

Definition 4.2 (The Itô integral)**.** *For every $n \in \mathbb{N}$ we define the sequence (t_k^n) by*

$$t_k^n = \begin{cases} k2^{-n} & \text{if } 0 \leq k2^{-n} \leq T \\ T & \text{if } k2^{-n} > T \end{cases}$$

We do now define $\varepsilon \subset L_2([0,T] \times \Omega, m \otimes P)$, where m is the Lebesgue measure, as the set consisting of all functions ϕ on the form:

$$\phi(t,\omega) = \sum_k e_k(\omega) \mathbf{1}_{[t_k^n, t_{k+1}^n]}(t)$$

*where $e_k \in L_2(P)$ and is $\mathcal{F}_{t_k^n}$ measurable. Now for all $\phi \in \varepsilon$ we define the **Itô integral** as:*

$$\int_0^T \phi dB = \sum_k e_k (B_{t_{k+1}^n} - B_{t_k^n}) \tag{4.8}$$

Note that e_k is evaluated at the left endpoint, which is what characterizes the Itô integral, for instance for the Stratanovich integral which is more commonly used in physics, uses the midpoint instead.

Now to be able to prove that there exists a continuous version the strategy is to first show that the Itô integral is a linear isometry, because then it can be extended to the closure of ε. Then we are to prove that it is a martingale. Combining those two results and using Doob's martingale inequality and the Borel-Cantelli lemma one has shown that the Itô integral is continuous almost everywhere. So let us first prove the Itô isometry:

Theorem 4.3 (Itô isometry)**.** *If $\phi \in \varepsilon$ then:*

$$E\left(\int_0^T \phi dB\right)^2 = E\left(\int_0^T \phi^2 dm\right) \tag{4.9}$$

Proof. We consider the left-hand side, taking the square of a sum we get some mixed terms and some identical terms. We will consider those two cases individually:

Consider the mixed terms, note that $e_j e_k(B_{t_{j+1}^n} - B_{t_j^n})$ is $\mathcal{F}_{t_k^n}$-measurable, and therefore independent of $B_{t_{k+1}^n} - B_{t_k^n}$ and remember that an increment in the Brownian motion is normally distributed with mean 0 and variance $t - s$

$$E(e_j e_k(B_{t_{j+1}^n} - B_{t_j^n})(B_{t_{k+1}^n} - B_{t_k^n})) = E(e_j e_k(B_{t_{j+1}^n} - B_{t_j^n}))E(B_{t_{k+1}^n} - B_{t_k^n}) = 0$$

Now consider the identical terms, again note that e_k is independent of the increment in the Brownian motion.

$$E(e_j^2(B_{t_{j+1}^n} - B_{t_j^n})^2) = E(e_j^2)E((B_{t_{j+1}^n} - B_{t_j^n})^2) = E(e_j^2)(t_{k+1}^n - t_k^n)$$

Adding everything together finishes the proof. □

As mentioned before this means that it is a linear isometry and it can therefore be extended to the closure of the space. What is left to prove is that every Itô integral are a Martingale:

Theorem 4.4. *If $f \in \overline{\varepsilon}$, then the Itô integral is a Martingale.*[10]

Proof. Consider the definition:

$$\int_0^T \phi \mathrm{d}B = \sum_k e_k(B_{t_{k+1}^n} - B_{t_k^n})$$

Now let $0 \leq t < T$, say $t_m^n \leq t < t_{m+1}^n$, then for $k > m$ we get:

$$E(e_k(B_{t_{k+1}^n} - B_{t_k^n})|\mathcal{F}_{t_k^n}) = e_k E(B_{t_{k+1}^n} - B_{t_k^n}|\mathcal{F}_{t_k^n}) = 0$$

Then we also know that by using the rule of repeated expectations:

$$E(e_k(B_{t_{k+1}} - B_{t_k})|\mathcal{F}_t) = E(E(e_k(B_{t_{k+1}} - B_{t_k})|\mathcal{F}_{t_k})|\mathcal{F}_t) = 0 \tag{4.10}$$

On the contrary, if $m > k$ we know that it is $\mathcal{F}_t$ -measurable, and that it does not change anything to take the mean. One may say that we have all the information or we know the outcome so the expectation of a constant is that constant. Adding everything together finishes the proof. □

Now by using Doob's martingale inequality and Borel-Cantelli one may show that **there exists a continuous version of the Itô integral almost everywhere.** To conclude, we know that the noise we are using exists, and we can integrate with respect to it, even in a continuous way. It is therefore time to define the stochastic differential equation and investigate how one solves it numerically.

[10] It is actually also possible to show it the other way around, namely that every martingale is an Itô integral. It is known as Martingale's representation theorem

4.3 Stochastic Differential Equations

We will start of by defining an SDE:

Definition 4.3 (Stochastic integral equation)**.** *Let X_t be an $(\mathcal{F}_t)$-adapted process, then we say that (X_t) satisfies the stochastic integral equation:*

$$X_t = X_0 + \int_0^t b(s, X_s)\,ds + \int_0^t \sigma(s, X_s)\,dB_s \tag{4.11}$$

one immediately notices the Itô integral-term at the end. Here we have b which is the drift and σ which is the noise.
Another point is that the above is not really a differential equation but as the definition states an integral equation. One might rewrite it into differential form:

$$dX_t = b(t, X_s)\mathrm{d}t + \sigma(t, X_s)\mathrm{d}B_t \tag{4.12}$$

However this is only notation, as it can be proven that the Brownian motion is nowhere differentiable. This does not cause any problems for us, because whenever we are confronted with a differential equation, we want to rewrite it into a well defined integral equation[11].

As always we are interested in existence and uniqueness, so let us consider that:

Theorem 4.5. *Let $Z \in L_2(P)$ such that Z is independent of $\{\mathcal{F}_t | 0 \leq t \leq T\}$. Additionally we demand that:*

$$|b(t,x) - b(t,y)| + |\sigma(t,x) - \sigma(t,y)| \leq D|x-y|; \quad \forall x, y \in \mathbb{R}, t \in [0,T]$$

called the Lipschitz condition. Then the equation:

$$X_t = X_0 + \int_0^t b(s, X_s)\,ds + \int_0^t \sigma(s, X_s)\,dB_s$$

has a unique solution with $X_t \in L_2(m \otimes P)$ such that X_t is adapted to $\mathcal{F}_t^Z$ generated by Z and $\mathcal{F}_t$.

Idea of proof. We use the typical procedure, namely we use a Picard iteration. Set $Y_t^{(0)} = Z$, and define:

$$Y_t^{(k+1)} = Z + \int_0^t b(s, Y_s^{(k)})\mathrm{d}s + \int_0^t \sigma(s, Y_s^{(k)})\mathrm{d}B_s \tag{4.13}$$

Then we use our assumptions to show that $Y_t^{(k)}$ has a limit in $L_2(m \otimes P)$. This, limit is our solution. In the ODE case, this would have been enough as a result of Banachs fixed point theorem, however in the SDE case, we are not finished and we have to show uniqueness too. To do this one uses the Itô isometry and the Lipschitz condition. □

Numerics

We do now have the integral equation that we want to solve. We can consider the basic method to derive numerical methods; here we assume the integrand to be constant over the interval that we integrate, giving us the following method:

$$X_t = X_0 + b(0, X_0)t + \sigma(t, X_t)B_t$$

[11] As we for instance saw in Section 3.1.

As this is a rather bad estimation one typically uses the additivity of domains property, so that we get:

$$X_{n+1} = X_n + b(t_n, X_{t_n})\Delta_{t_n} + \sigma(t_n, X_{t_n})\Delta B_{t_n} \tag{4.14}$$

this is called the Euler-Maruyama method or simply the stochastic Euler method. However, as in the ODE case is this the simplest method, we would therefore like to consider a new method which should produces considerably better results.

N. Grønbech Jensen and O. Farago [8] have recently developed a new algorithm efficient at integrating models based on Langevin dynamics. The idea is that whenever these types of systems are simulated one is typically not interested in the position or trajectory of a single particle, but general statistical properties expected from equilibrium statistical mechanics. If we translate this into our problem, we are more interested in knowing; whether the pedestrians clog in general and not which pedestrian that get stuck or possibly trampled on. The algorithm is derived at follows

Derivation

First thing is to integrate the Langevin equation over a small time interval $\tau \in [t_n, t_{n+1}]$, with length dt. As we are solving vector equations the same method can be applied to all entries in the vector, so we will only consider the one dimensional case.

$$\int_{t_n}^{t_{n+1}} m\frac{\mathrm{d}v(\tau)}{\mathrm{d}\tau}\mathrm{d}\tau = \int_{t_n}^{t_{n+1}} f(s(\tau),\tau)\mathrm{d}\tau - \int_{t_n}^{t_{n+1}} \lambda v(\tau)\mathrm{d}\tau + \int_{t_n}^{t_{n+1}} \sigma \mathrm{d}B_\tau$$

where the first term which the conservative force and the second term which is the drag is two ordinary integrals. The last term is the noise, which is an Itô integral. We can without approximations rewrite it into:

$$m(v^{n+1} - v^n) = \int_{t_n}^{t_{n+1}} f(s(\tau),\tau)\mathrm{d}\tau - \lambda(s^{n+1} - s^n) + \beta^{n+1} \tag{4.15}$$

where v^n and s^n are shorthand notations for $v(t_n)$ and $s(t_n)$ respectively and $\beta^{n+1} = \int_{t_n}^{t_{n+1}} \sigma \mathrm{d}B_\tau$ which is a new Gaussian number characterized by $E(\beta^n) = 0$ and $E(\beta^n\beta^l) = 2\lambda k_B T dt \delta_{n,j}$.

As we are interested in the position and not just the velocity, we do have an additional equation that we need to solve:

$$\frac{\mathrm{d}s(\tau)}{\mathrm{d}\tau} = v(\tau) \iff \int_{t_n}^{t_{n+1}} \frac{\mathrm{d}s(\tau)}{\mathrm{d}\tau}\mathrm{d}\tau = s^{n+1} - s^n = \int_{t_n}^{t_{n+1}} v(\tau)\mathrm{d}\tau$$

Using the trapezoidal rule, one can approximate it with:

$$s^{n+1} - s^n = \frac{dt}{2}(v^{n+1} - v^n) + O(dt^3)$$

Now using equation 4.15, we know what $v^{n+1} - v^n$ is and we can insert it into the above equation and get a pair of equations:

$$s^{n+1} - s^n \approx bdtv^n + \frac{bdt}{2m}\int_{t_n}^{t_{n+1}} f(s(t')t')dt' + \frac{bdt}{2m}\beta^{n+1}$$

$$v^{n+1} - v^n = \frac{1}{m}\int_{t_n}^{t_{n+1}} f(s(t),t)\mathrm{d}\tau - \frac{\lambda}{m}(s^{n+1} - s^n) + \frac{1}{m}\beta^{n+1}$$

where $b = \frac{1}{1+\frac{\lambda dt}{2m}}$. If we ignore the stochastic term, then the equations are respectively of second order and exact. So we approximate the integral, $\int_{t_n}^{t_{n+1}} f(s(\tau), \tau)\mathrm{d}\tau$, such that both methods are of order two, again note that this is in the deterministic case where the noise is ignored.:

$$s^{n+1} \approx s^n + bdtv^n + \frac{bdt^2}{2m}f^n + \frac{bdt}{2m}\beta^{n+1} \tag{4.16}$$

$$v^{n+1} \approx v^n + \frac{dt}{2m}(f^{n+1} - f^n) - \frac{\lambda}{m}(s^{n+1} - s^n) + \frac{1}{m}\beta^{n+1} \tag{4.17}$$

Note that in the case where we solve Newton's second law of motion the pair of equations reduce to the classical Stormer -Verlet method. We could have mad higher order approximations in both cases, however this will require more function evaluations per step, without any guarantee that it would converge in the case of we are solving a stochastic differential equation.

5 Pedestrian Dynamics Revisited

The model in Section 2 is just the basic model, but as we will now see, it is the core of the models that are being used. Also as earlier mentioned will we now apply the methods developed in the above section to these extensions.

In this thesis we investigate these models in terms of how the pedestrians react to panic. Therefore, we would need some measurements that are applicable to both models, where we can measure the discrepancy of the escape.

To test the two models we use the same set-up. The set-up that we are going to use is a 2 dimensional box with length $L_x = 20$ and width $L_y = 5$, this models a corridor that the pedestrians have to pass. The box has periodic boundary conditions at the ends, even though this does not seem like a reasonable thing to have as if you run into a wall you do not come out on the other side of the room. However this models the fact that the people keep on rushing through, so once a person leaves the room and comes back in, it should be interpreted as a new person entering the scene. This makes the computation a lot faster, as the simulation box only has to represent the key areas e.g. exits, and not necessarily the entire area, e.g. the stadium. The largest benefit is however that the number of people can be lower, and as we have seen the time complexity of the basic model is $O(n^2)$ so if we can lower the number of people in the simulation box, this has huge efficiency benefits. At the top and bottom we have walls with forces described in the models.

Then inside the simulation box, we have two evenly divided groups of pedestrians. One on the left and one on the right. The idea now is to send all the people from the right to the left and vice versa. This might not be a situation, but it serves perfectly as a test example, as here we have interaction with the walls, interaction to the people who have the same interests and people who have different interests, modelling that different groups of pedestrians clash together, so it takes almost everything of interest into account.

Once the model has run, we would like to investigate the outcome, especially the discrepancy of the escape is of interest and how panic is affecting it. Earlier work has made different suggestions to how one can measure the efficiency or corresponding discrepancy (See for instance [3]). As we want to tie things together, we would like to use a well known quantity, the **kinetic energy** as such measure. However, to be able to do that we consider the particle to be in a co-moving rest-frame. For each particle this rest-frame is moving in the direction of e_0 with a speed of v_0. In other words the discrepancy over time is given by:

$$\mathrm{DCP}_i = m_i \cdot \|v_i(t) - e_0 v_0\|^2, \tag{5.1}$$

where m_i is the mass and v_i is the velocity, both of pedestrian i.

So the smaller value for the discrepancy the better. We will later use the mean of the quantity both in respect to all of the pedestrians and with respect to time. Such that we get:

$$\overline{\mathrm{DCP}} = \int_{t_0}^{t_{max}} \frac{1}{n} \sum_{i=1}^{n} m_i \cdot \|v_i(t) - e_0 v_0\|^2 dt \tag{5.2}$$

The two models that we are to investigate have both been proposed by D. Helbing, I. Farkas and T. Vicsek articles [2] and [3] and are discussed separately under the following sections. Please note that the following sections are attempts to replicate the results that they got in the respective articles, and to make connections between them. Which is also the reason for the change in the social force.

5.1 Tangential Velocity Force

In this model, we want to investigate the basic behaviour of pedestrians trying to follow in each other's footsteps. So if one pedestrian succeeds to break through, we model that nearby pedes-

trians want to move in the same direction. This creates fluid-like motion and has some relation to a deterministic DPD (**D**issipative **P**article **D**ynamics [18]) model known from physics. So we have chosen the repulsive forces in the following way. Please note the last terms where the tangential vector operates - this is the term that gives us that pedestrians tend to follow one another.

$$f_{iw}(s_i, v_i) = \left\{ A \exp\left[\frac{(r_i - d_{iw}(s_i))}{B} \right] + kg(r_i - d_{iw}(s_i)) \right\} n_{iw} - \kappa g(r_i - d_{iw}(s_i))(v_i \cdot t_{iw}) t_{iw}$$

Where $d_{iw}(s_i)$ is the closest distance between the wall and the pedestrian, r_i is the size of the pedestrian, $g(x)$ is a function which is zero if the argument is negative and equal to the argument if it is positive. A, B, k and κ are constants, n_{iw} is the normal vector between the pedestrian and the wall, t_{iw} is the tangential vector between the pedestrian and the wall and v_i is the velocity of the pedestrian. If we consider the terms, then the first term is the typical force to the wall, then next term is a friction term given that the pedestrian touches the wall and the last term is the tangential velocity term also given that the pedestrian touches the wall.

Similarly the repulsive force between pedestrians are defined as:

$$f_{ij}(s_i) = \left\{ A \exp\left[\frac{(r_{ij} - d_{ij}(s_i))}{B} \right] + kg(r_{ij} - d_{ij}(s_i)) \right\} n_{ij} - \kappa g(r_{ij} - d_{ij}(s_i))(v_j - v_i(s_i)) t_{ij}$$

where $r_{ij} = (r_i + r_j)$, the last term is the friction due to the relative velocity difference as we also saw to the wall. It is this term that gives the fluid like behaviour which is also observed in DPD models. The rest is defined equivalent to the above equation.

We choose the constants to represent football fans, meaning that we chose $m = 80kg$ and a size r_i uniformly distributed in the interval $[0.25, 0.35]$. It has been investigated [10], that the free flow speed is approximately:

- $v_0 \approx 0.6m/s$ under relaxed conditions
- $v_0 \approx 1.0m/s$ under normal conditions
- $v_0 \approx 1.4m/s$ under nervous conditions

In other words, if we are to adjust the level of panic in this model we have to change the free flow velocity. So if we turn up the free flow velocity, then we model an increased level of panic. In the article [2] D. Helbing et al. do also note that an increased degree of panic causes the pedestrian to be more individualized, and not behave like everyone else, however they do not include that factor in this model.

Reasonable values for constants under this model have been chosen as in [11]: $\tau_i = 0.5s$, $A = 2000N$, $B = 0.08m$, $k = 1.2e5kgs^{-2}$ and $\kappa = 2.4e5kgm^{-1}$.

So as we are to investigate the behaviour as the panic increases we have to run simulations with increasing free flow velocities.

Results

Simulations under the above conditions have been run and we have observed the same behaviour as we expected for a system that experiences an increased panic. Namely that the discrepancy, from equation (5.2) of increased. This is shown in Figure 2.

In this figure we see the discrepancy per mass of the pedestrians as a function of the free flow velocity, so the expected behaviour is that as the pedestrians want to move faster, then the

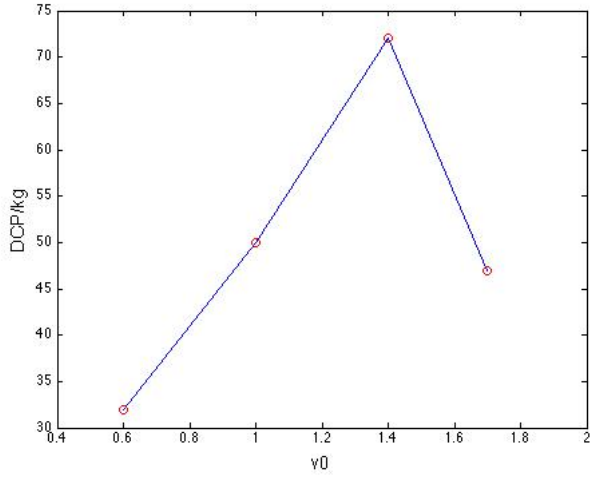


Figure 2: Plot of the discrepancy per mass of the pedestrians as a function of the free flow velocity

discrepancy became worse. However here the last data point is especially of interest as this goes against what we have expected. What this data point tells us is that the pedestrian is more likely to move with the free flow velocity as we increase the free flow speed. We can also see this from the simulations. In Figure 3 we can see that with a low free flow velocity the pedestrians try to move past each other nice and easy, which takes a bit longer time, however, the force that the pedestrians collide with is significantly smaller at the lower speed compared to the larger speed. If we consider the right group of people in the model where they move the fastest, one can see that they, in matter of a very short time, just push violently through the other group. This creates two lanes in a matter of very short time, however the number of casualties are larger and the damage is more severe [2]. Also if there were more people and they were not able to create lanes because of the limited space, then the discrepancy would probably not be better, this is also supported by the simulations in [2], where they investigate the movement through a narrow door, where lanes are not possible. What is also important to remember is that panic is also associated with individualistic movement, which is not included in this model. So when we increase the speed the pedestrians in one group can aid each other in breaking through the other group, if they were moving more individually this would not be the case. This we will investigate in the next model.

5.2 Stochastic Behaviour

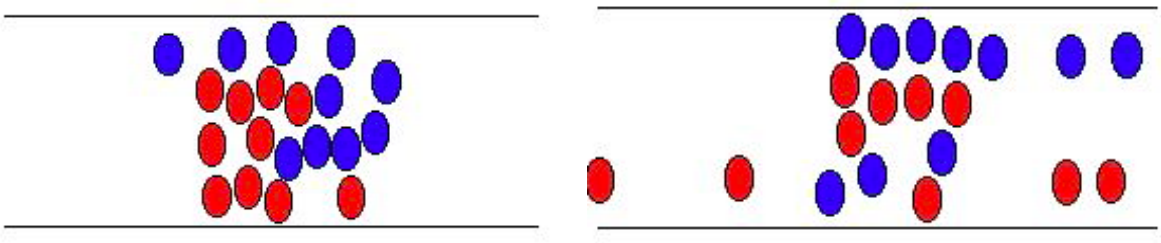
In this model we would like to investigate the importance of a stochastic noise. Using this model the Newtonian motion generalizes to the so called Langevin equation which is of the following form:

$$f_i = m_i \cdot \frac{d^2 s_i}{dt^2} = f_{ij}^c(s_i) - m\lambda \frac{ds_i}{dt} + \beta_i(t) \tag{5.3}$$

Here the first term is the conservative force typically describing the force between the molecules or in this case the pedestrians in the simulation. The second term is the friction term, which pulls energy out of the system and the last term is the noise which pushes energy into the system.

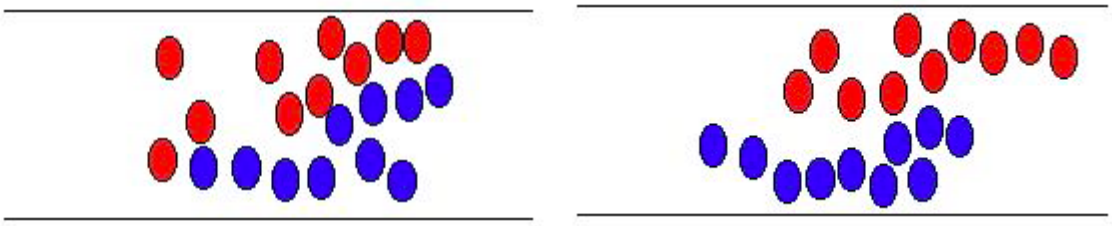
To this Langevin equation one will in our case add a driving force, which will make the pedestrians move from one side to another. Which will make it equal to:

$$f_i = m_i \cdot \frac{d^2 s_i}{dt^2} = m\frac{e_0 \cdot v_0}{\tau} + f_{ij}^c(s_i) - m\lambda \frac{ds_i}{dt} + \beta_i(t) \tag{5.4}$$



(a) $v_0 = 1$, here we see that the pedestrians have clogged and try to sort it out

(b) $v_0 = 1$, after some time they have sorted it out



(c) $v_0 = 1.7$, They never actually clog, but smash through

(d) $v_0 = 1.7$, and after a short amount of time they have created lanes

Figure 3: Snapshots of simulations with 20 pedestrians moving inside a box of size $L_x = 20$ and $L_y = 5$, the blue group moves from right to left and the red group moves left to right.

From equation (2.5), we can easily see that by choosing $\lambda = \frac{1}{\tau}$, and the conservative force $f_{ij}^c(s_i) = \sum_{w \in W} f_{iw}(s_i) + \sum_{i \neq j} f_{ij}(s_i)$ the above equation has the same structure, the only addition is the stochastic term. This stochastic noise is assumed to be the previously described Brownian motion with $E(\beta(t)_i) = 0$ and $var(\beta(t)_i) = \theta$.

The entire idea behind doing so is that we can now relate this type of problem to existing problems in statistical mechanics, where much work already has been made in order to solve such systems efficiently as we saw in Section 4.

Again as we will reproduce the results in the article [3], we will chose the forces as they do. The entire exercise is to investigate the influence of the size (variance) of the noise, which force us to choose a rather strong repulsive force such that the pedestrians do not overlap, corresponding to the fact that every person does not fall but can keep pushing other persons away from him or her. To meet this, the forces has been chosen in the following way:

$$f_{iw}(s_i) = -\nabla A(d_{iw}(s_i) - r)^{-B}, \tag{5.5}$$

and

$$f_{ij}(s_i) = -\nabla A(d_{ij}(s_i) - r)^{-B}. \tag{5.6}$$

Here, A and B are constants, d_{iw} and d_{ij} is the distance from the i'th pedestrian to respectively the wall and pedestrian j, r is the size of the pedestrian and ∇ is the gradient.

As mentioned the panic is represented as stochastic noise, so the larger the noise (variance) the more panic, or in terms of physics, then the noise is proportional to the equilibrium temperature of the system, so we can relate the panic to the temperature of the system. To investigate what panic does in this case, we run simulations with increased stochastic noise or temperature.

Results

Here we have run simulations using the stochastic Euler method described in Section 4.3, with noise variances ranging from $1e0$ to $1e3$, each time increasing with a factor of $1e0.5$. Now,

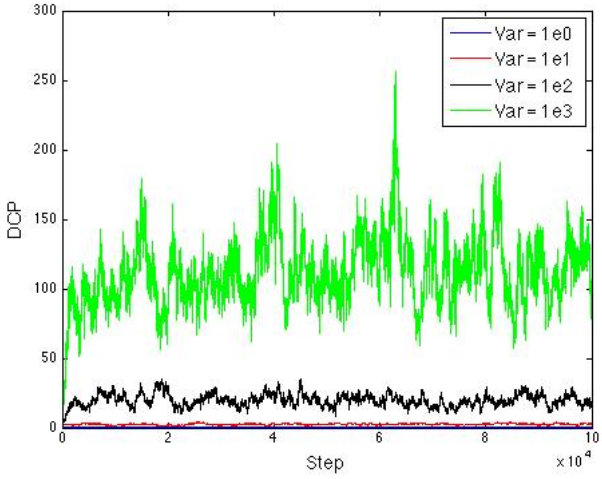


Figure 4: The time dependant discrepancy at different variances.

plotting the discrepancy against time we produce Figure 4. What we see is that as we increase the variance, the discrepancy becomes worse, i.e. the value increases. What we also see is that all discrepancy plots fluctuate around some average. We would have expected to be able to see when the two groups clash together, but what we actually observe here is that under relaxed conditions, $var = 1e0$ and $var = 1e1$, the discrepancy is nearly zero and the pedestrians move in the desired direction almost all the time, apart from the random kick, which causes some disturbance. But when we consider the discrepancy at higher variances, we see that they are no longer moving in the desired direction, but have frozen into a structure and stay in that formation the entire time, or in other words, they are no longer moving in a certain direction. What is important here is that whatever there is happening it is almost the same over the entire time.

Another important aspect is how the discrepancy is directly affected by the variance. Figure 4 gives a pretty good idea but in Figure 5, this is explicitly plotted.

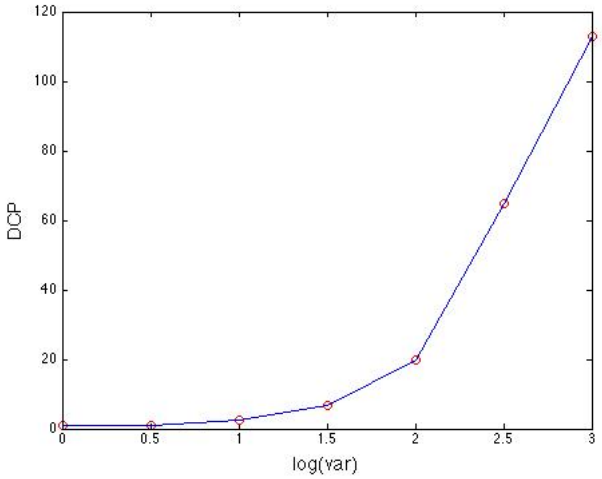


Figure 5: The average discrepancy as a function of the variance. For a larger number of pedestrians see the article [3]

Here we see the same behaviour as expected, namely that as we increase the panic i.e. the variance, then they move less coordinated and get stuck between each other. This is the same behaviour as seen in the original paper [3]. Note that they in the article [3] are using a different

measurement, where lower value corresponds to higher discrepancy.

5.3 General Results

As seen in the previous two models, if we increase the degree of panic, then the efficiency of the escape decreases, the discrepancy gets larger. In [3], they call it freezing by heating, and for good reason. If we are to take a physical perspective on this and use ideas and concepts from molecular dynamics (MD), then one can see the driving force as energy being pumped into the system, or as the temperature being increased. In the same manner you can see the temperature in the stochastic case as the cause of the noise. In MD one writes the variance of the noise as:

$$var(\beta(t)_i) = \sqrt{\left(\frac{k_B T m}{dt\lambda}\right)},$$

where it is clear that the stochastic noise is proportional to the equilibrium temperature T, which means that increasing the variance, increases the temperature.

The conclusion is that in both cases we can see the panic in the system as the temperature of the system if we also keep in mind that panic should make the pedestrians move more independently, which is the case if we increase the temperature using a stochastic term. Knowing this, it is obvious why freezing by heating makes sense.

6 Group Behaviour

This new attempt to improve existing models with respect to accuracy focuses mainly on the implementation of group behaviour and the consequences of the improved model. We will for instance investigate the importance of the random kick in the same manner as the authors of [3]. We will write the model in terms of Langevin dynamics, which in general is given by [6]

$$m_i \cdot \frac{dv_i}{dt} = f_i(r,t) - \lambda v_i + \beta_i(t) \tag{6.1}$$

Where $f_i(r,t)$ is the resulting force from the other pedestrians, the wall, the group behaviour and the determination to reach the door. The λv_i term is a damping factor ensuring that each pedestrian does not just continue to accelerate, as noted just before equation (2.2). To extend the existing models we choose $\lambda = \frac{m}{\tau}$ where τ is the characteristic time used earlier in this thesis and in this paper [2]. Finally the $\beta_i(t)$ is a random force kick which statistical properties are $E(\beta(t)_i) = 0$ and $var(\beta(t)_i) = \sqrt{\left(\frac{K_b T m}{dt * \lambda}\right)}$, which as investigated in [3], showed to have some importance.

6.1 Extension

The interesting part is the introduction of the strong group behaviour which is defined in two different ways. We assume that this strong group behaviour, denoted by $f_j^G(r)$ for the ith pedestrian, is different for friends going to, for instance, a concert and a family at the same event. The difference is that when disaster strikes the order in which the family sits or stands freezes, in the sense that the two parents which are most likely to be at the two ends literally grabs the children who sits in between them and by that, most likely sacrifice speed and efficiency to make sure that their children are unharmed. Friends on the other hand do not grab each other, and therefore their order is not frozen. The friends have a cohesive force as they want to stay together, but to all persons in the group, so they are more adaptive to their surroundings than the family. We will in this thesis only consider families. However, our thoughts of how this can be expanded to friends can be found in the appendix, section 10.1. This strong group behaviour is only active between two pedestrians if they are in the same group denoted by $G2_i$ for the ith pedestrian.

We can now divide the force $f(r,t)_i$ into a force which is more descriptive, so:

$$f_i(r,t) = m_i \frac{e_i^0 v_0}{\tau} + \sum_{j \neq i} f_{ij}^S(r) + \sum_{w \in W} f_w^W(r) + \sum_{\substack{j \in G2_i \\ i \neq j}} f_j^G(r) \tag{6.2}$$

The first term is the usual term defining the free flow velocity, the direction of the exit, and the characteristic time, describing how fast one adapts to the wanted direction.

The next term defines the social force between each pedestrian and this should now also depend on which group the two pedestrians in question are in. We choose the force between two pedestrians as:

$$f_{ij}^S(r) = (A\mathbf{1}_{\{j \in G1_i\}} + B\mathbf{1}_{\{j \notin G1_i\}}) \cdot \exp\left(\frac{\sigma_p - d_{ij}(s_i)}{\rho}\right) \tag{6.3}$$

Here $\mathbf{1}_B = \begin{cases} 1 & \text{if } B \\ 0 & \text{if } \neg B \end{cases}$.

We here define the large groups, $G1_{\{i \in \{1,2,\ldots,n\}}$ for pedestrian i, one group with pedestrians who move in one direction and another group with pedestrians who move in the opposite

direction, by direction we mean the intended direction, so it does not matter if the force on a pedestrian forces him to move in another direction, this does not change the group. We further set two constants with $A \leq B$, so if pedestrians i and j are in the same group, then the repulsive force between them is smaller than if they were not in the same group. A natural special case of this is if $A = B$ then there is no group behaviour. Further σ_p is the two radii added together, such that each pedestrian has a radius and is not just a point. ρ is a scaling factor of the effective distance between the two pedestrians.

The force against the wall is assumed to be the same as the force between two pedestrians, however, here the wall is a point and the wall has no group behaviour so it reduces to:

$$f_i^w(r) = B \cdot \exp\left(\frac{\sigma_w - d_{iw}(s_i)}{\rho}\right) \tag{6.4}$$

Here we have that the repulsive force to the wall is identical to the force from a pedestrian, who you are not in a group with. Again $\sigma_w - d_{iw}(s_i)$ is the effective distance between the pedestrian and the wall, but this time σ_w is equal to the radius of one pedestrian, which by other means is the same as the range of the person plus the range of the wall, but the range of the wall is zero, so it reduces to just the range of the pedestrian. Further a reflective property of the wall is assumed, such that the if the force to the wall is to large and the pedestrian is pushed outside the simulation box, then the pedestrian is mirrored back in.

The strong group behaviour is assumed to be a Morse potential, which is given by:

$$E_{ij}^G(r) = \begin{cases} D(1 - \exp(-\alpha(d_{ij}(s_i) - r_0)))^2 - D & \text{if } \text{on}_{ij} = 1 \\ 0 & \text{if } \text{on}_{ij} = 0 \end{cases} \tag{6.5}$$

The variable *on* is a boolean variable which determines whether the bond is still active. This potential fulfils the following properties:

1. A potential which grows large as $r \to 0$, which mimics that friends and family do not want to step on each other. Note that this is in addition to the existing potential between all pedestrians

2. A potential which can be smooth around the minimum allowing friends to stay together, but not restricted to a certain distance

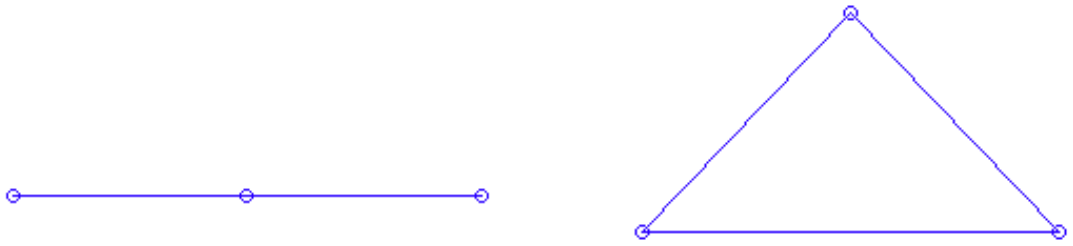
3. A potential which can be sharp, such that family members who are locked at a distance to each other also can be mimicked.

4. A potential where the desired distance between pedestrians can be set directly, r_0.

The second and third properties of the potential are controlled by the scaling factor α, which gives the length, and D, which gives the depth. We will in the following describe how the family members are assumed to be interconnected. The following observations are based on personal experience, and how we think we would react in these type of situations

As we want to specify the forces we have to change the energy into the corresponding force, which is done by the relation $f(x(t)) = -\frac{d}{dx}E(x)$, we then get:

$$F_{ij}^G(r) = \begin{cases} -2\alpha D \exp(\alpha(d_{ij}(s_i) - r_0))(\exp(\alpha(d_{ij}(s_i) - r_0) - 1)) & \text{if } \text{on}_{ij} = 1 \\ 0 & \text{if } \text{on}_{ij} = 0 \end{cases} \tag{6.6}$$



(a) Sketch of the bond between **family** members

(b) Sketch of the bond between **friends**

Figure 6: Two different bonds

6.2 Family

As earlier described we are assuming families to be of a fixed order, in other words, the order cannot be permuted, which is sketched in Figure 6a.

So we define a group $G2_{\{i\in\{1,2,\ldots,n\}}$ which is the strong group, which pedestrian i is in. For the above case we further set the variable

$$on_{ij} = \begin{cases} 0 & \text{if } |i-j| > 1 \\ 1 & \text{if } |i-j| \leq 1 \text{ and } j \in G2_i \end{cases} \tag{6.7}$$

So if two pedestrians inside the same group (family) are sitting next to each other then they will grab each other and stick to that formation.

But as humans we have limited strength and our arms have a limited length, so if the distance between two persons inside the same group gets larger than R_c then the variable on is set to 0, indicating that the bond is lost and the group will have to continue with a person less. It is reasonable to assume that the bond cannot be made again as it is either the person in front or back that goes missing, which here is one of the parents. If one of the parents are lost, the other parent will typically continue with the children, as you believe the other parent might cope, and the children are more important.

We want to set the parameters to mimic how strongly an adult and a child can hold on to each other, further we are interested in that the force field mimics the length of such a connection. With this in mind we have chosen the constants:

Parameter	Value
r_0	0.5
D	2000
α	3
R_c	1.5

(6.8)

One thing that is special to families is that the bonds are actual bonds, in the sense that the parents hold on to the children. We will therefore have to consider the case where the one in the middle runs backwards and the two other forward or vice versa. To cope with this problem we put an angular force on the middle person. This force is given by:

$$f_A(\theta) = k \cdot (\theta - \pi), \tag{6.9}$$

which is the spring equation, where θ is the angle between the two outer pedestrians and we choose $k = 10$.

6.3 Complexity

As we saw in the original problem the complexity was $O(n^2)$. By adding the additional contracting force to the groups of friends and family members, we add an additional complexity of $O(\sum_{i \in N} |G2_i| + \sum_{i \in N} |G1_i|)$, now letting $G_{max} = \max_{i \in N}\{|G2_i|, |G1_i|\}$ we get that we can bound the complexity added by $O(2nG_{max})$. We can now ignore the constant and as the groups consist at most of all people ($G_{max} \leq n$), we know that we in total do not alter the complexity. So the complexity of the total problem is $O(n^2)$ just as before.

6.4 Results

As done in [3], we have run simulations with variances ranging from $1e0$ to $1e4$, with a factor of $1e0.5$ in increments. This corresponds to a steady increase in temperature, which as earlier discussed corresponds to an increase in panic and a higher tendency to move in other directions than to the nearest exit. The size of the time-steps are in all cases, $dt = 0.001$.

As discussed already in Section 5 we are interested in the discrepancy of the escape. The first thing we are interested in is the discrepancy over time, which is plotted in Figure 7.

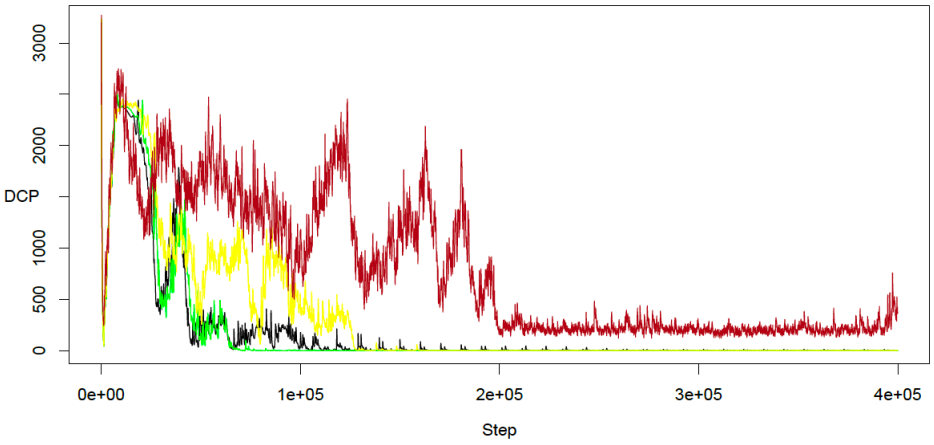


Figure 7: Time dependant discrepancy of the escape, where black corresponds to a variance of 0, green corresponds to $1e1$, yellow $1e2$ and red $1e3$. The first 1e4 steps are excluded as all pedestrians stand still initially.

What we see here is that without any stochastic behaviour the escape is actually not the most efficient. If they move in a greedy way, meaning the shortest way and solve the problems as they arise, they might get stuck, which is depicted by the small spikes we see on the black line. However, if there is a small amount of panic or tendency to move independently, then if they are close to getting stuck the stochastic behaviour makes them try a new way, making the escape a bit better. As we increase the panic further we see that the time it takes for the pedestrians to get aligned i.e. run in lanes, takes longer and longer.

Now we consider the cases with even more panic, which can be seen in Figure 8.

Then we see that the panic has taken over the system and the pedestrians are jammed and can no longer move in the desired direction.

We can now consider the mean of the discrepancy as a function of the variance. This is plotted in Figure 9. Here we see the same as depicted by Figure 7. Namely that in the beginning the discrepancy decreases as we increase the stochastic behaviour, but at some stage the panic takes over and the system has frozen.

To conclude we here saw that in contrast to what was observed in the article [3] did the discrepancy not immediately increase, but actually drop. As we further increased the panic, we did observe the same behaviour, where the panic took over the system making the pedestrian

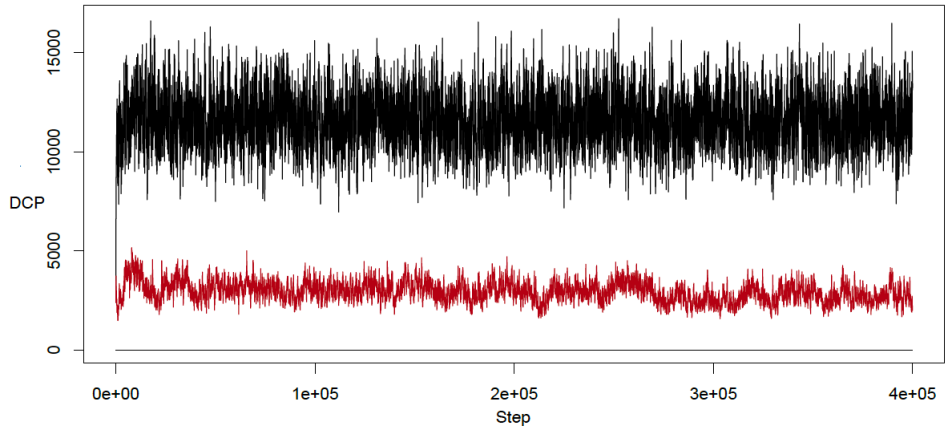


Figure 8: Time dependant discrepancy of the escape, where black corresponds to a variance of $1e4$, and red $1e3.5$. The first 1e4 steps are again excluded

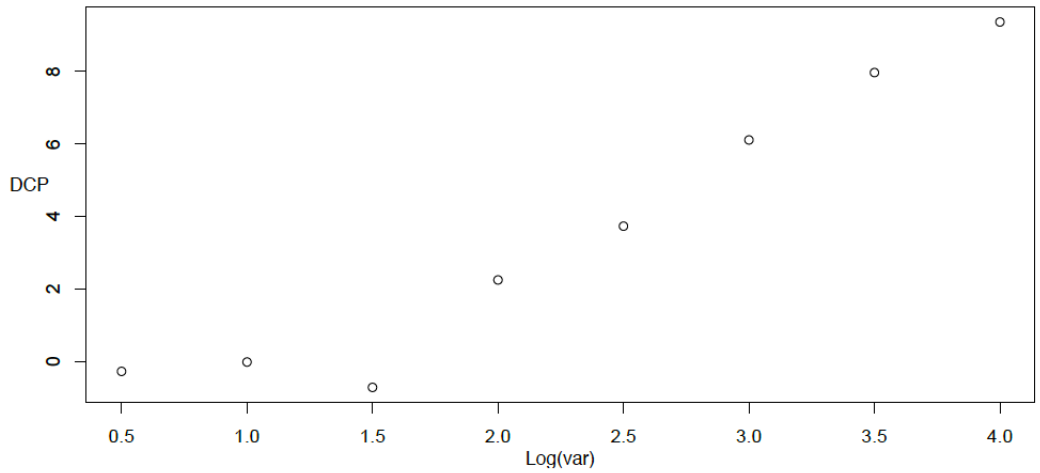


Figure 9: Figure of the mean of the discrepancy as a function of the variance. Note that the deterministic case is not included as $log(0)$ is not defined

move uncontrollably and eventually freeze the system. Another important thing we observed was that the discrepancy was able to decrease over time, this was not the case, when we redid the experiments in Section 5.2, here the system kept the discrepancy during the entire simulation. This information actually tells us that it is better to stand back a while and let the lines form before you start to move, given that you have the time to wait and that the degree of panic is not so large that they will never form. Where if we do not assume group behaviour, then it is better to be the first one, to see if you can get out before everyone clog.

7 Efficiency

In all models described in Sections 5.1, 5.2 and 6, we saw that the complexity of the problem was $O(n^2)$ under the assumption that the number of walls is smaller than the number of pedestrians. But the question is now; is the runtime of the solver also of the same complexity? This will be investigated in this section.

7.1 Complexity and Actual Runtime

To investigate the runtime I have chosen to use the model from Section 5.2, however without the stochastic noise. The code was implemented into Matlab and the built-in solver ode23s was used to solve the problem. The ode23s is an L-stable solver as the potentials in the model were indicating that the problem would be stiff, which can easily be confirmed when trying to solve it using ode45 [12]. ode23s is an adaptive method, and it is used with standard settings, meaning that the relative tolerance was set to: $'RelTol' = 1e(-3)$. To measure the time, I have used Matlab's functions "tic - toc" which measures the CPU time used. The times measured are the actual run-times of the solver. So all initialization is not included in the measurements as this is not the quantity of interest. However, for future work it would be interesting to investigate efficient methods to create a feasible initial position.
After running the simulation with various numbers of pedestrians, I got the following data:

$$\begin{pmatrix} \text{Pedestrians} & 2 & 4 & 6 & 8 & 10 & 12 & 14 \\ \text{Times} & 2.9741 & 302.4589 & 482.6357 & 1216.9395 & 4029.4324 & 14999.4265 & 45339.3822 \end{pmatrix} \tag{7.1}$$

If we first plot the data we get the scatter plot in figure 10

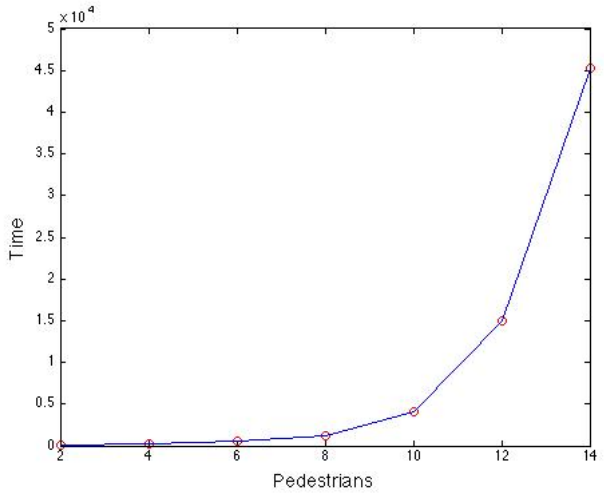


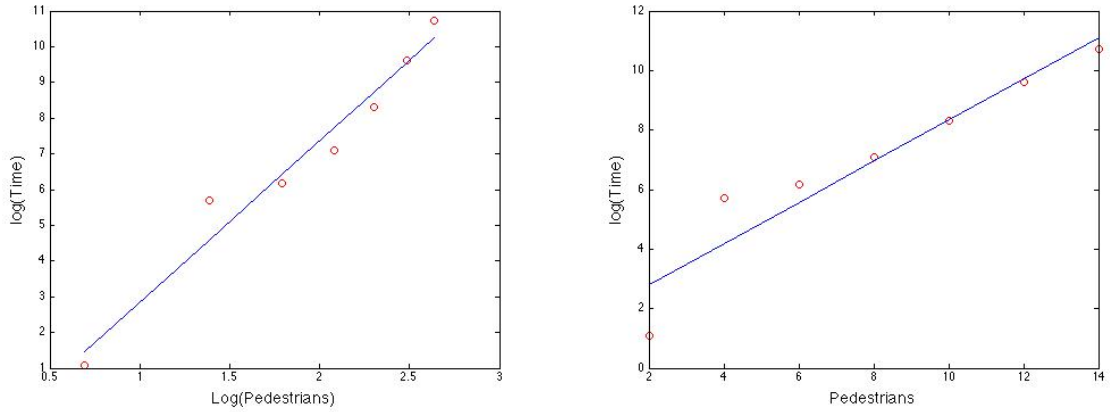
Figure 10: Scatter plot of adaptive time stepping times

We can now try to fit the data with a polynomial and exponential, this is done by using a double logarithmic or single logarithmic plot respectively, the best linear fit then decides which models are the best. Further if the solution is the polynomial fit, then the slope decides the order.

As mentioned earlier; if the solver runs with the same complexity as the problem, then we should be able to fit it with slope of 2. I have here used Matlab's polyfit command to fit the

[12] The standard explicit Runge-Kutta method of Dormand and Prince, adaptive 4th order with extrapolation

data, the result can be seen in Figure 11.



(a) Linear regression of a loglog plot. It has a slope of 4.5167 and $R^2 = 0.9624$.

(b) Linear regression of a single logarithmic plot. I has a $R^2 = 0.9025$

Figure 11: Regression of runtimes for $n \in \{2, 4, 6, 8, 10, 12, 14\}$

From Figure 11 we can see a regression line for both a single logarithmic and double logarithmic scale. It is clear from the correlations coefficient, that the polynomial model fits considerably better than the exponential, indicating that if we use adaptive step-size then the solver runs as $(n^{4.5})$ which is considerably higher than the problem itself. Though we have to keep in mind, that because of the small dataset, our estimate here is an upper bound, because smaller terms still might have an increasing effect making the order seem larger than it should. The first idea for where this order difference could come from is that we are using adaptive step-sizes. So we would now like to see the actual run-times if we are using fixed time-steps, after running simulations with fixed step-size of $dt = 0.001$ and various number of pedestrians we get:

$$\begin{pmatrix} \text{Pedestrians} & 2 & 4 & 6 & 8 & 10 & 12 & 14 \\ \text{Times} & 126.0735 & 236.5285 & 443.7769 & 933.3079 & 1789.4506 & 2757.2031 & 3943.1458 \end{pmatrix} \tag{7.2}$$

We make a scatter plot of the data and get Figure 12:

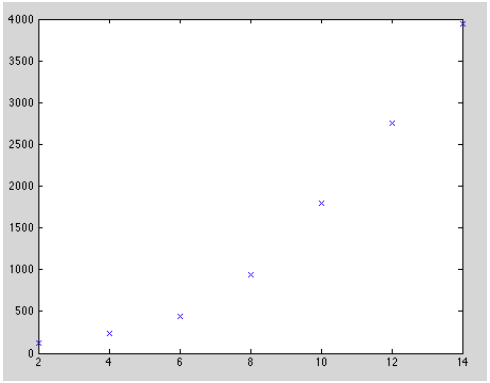


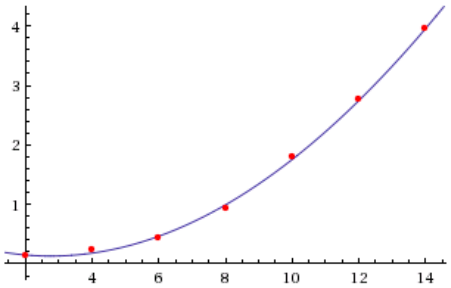
Figure 12: Scatter-plot of fixed time-stepping times

If we analyse these data in the same manner as before we get Figure 13.

Fit diagnostics:

AIC	BIC	R^2	adjusted R^2
−16.7729	−17.0434	0.999287	0.998574

Plot of the least-squares fit:

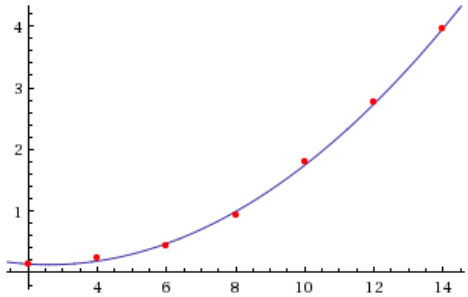


(a) Cubic regression: $R^2 = 0.9992$

Fit diagnostics:

AIC	BIC	R^2	adjusted R^2
−18.4617	−18.678	0.999254	0.998882

Plot of the least-squares fit:



(b) Quadratic regression: $R^2 = 0.9992$

Figure 13: Cubic and quadratic regression of runtimes. As the fit is so good, there is no need to check for eksponential fit

Both regressions are nearly identical, and as you can always use a higher order polynomial to fit a lower order polynomial, this tells us that when we are using fixed time steps, then the solver runs as $O(n^2)$, so the increase comes from the adaptive time-stepping. The adaptive method reacts to approximated errors, and if this estimate gets too large, then it lowers the step-size, and on the other hand if the error is too small then it increases the step-size to make the algorithm go faster. If we compare the times with two pedestrians, then we can see that the adaptive method is faster and has therefore made the time-stepping larger than what I obtained with fixed time steps, however if we look at the times for fourteen pedestrians then the adaptive time-stepping is considerably slower than the fixed time-stepping. This indicates that the error might rise along with the increase in pedestrians. This is what I will investigate in the next section.

7.2 Precision

We have now seen that it is the step-size control which destroys the runtime, and using fixed step-sizes, we can reduce the runtime. But why is it that the adaptive step-size control destroys it? The obvious reason is that the error increases with the number of pedestrians, but how much does the error increase and how does the method react to it? These two questions can be answered by looking at the error with a varying number of pedestrians and varying step-sizes.

It is obvious that an analytical solution does not exist or it is nevertheless very difficult to find. So we can not just compare the numerical solution to the exact solution, what we can do instead is to use that for the step-size going to zero, our numerical solution should converge to the correct one. So if we compare two solutions to the same problem but with different step-sizes, we will get an idea of how big the error is given that it converges. In the chart below, we see simulations with step-sizes $dt_n = \frac{1}{2^n}$ for $n = \{1, 2, 3, 4, 5, 6, 7, 8\}$, and used the lowest as comparison. This I have done for two to fourteen pedestrians, so we can see how the error behaves. As the error control is the reason for the higher order in time complexity, we would expect some increase in error as we add more pedestrians.

The simulations have been run on the same problem as in Section 7.1, however with the random kick set to 0 to avoid wrong measurements. I have used the three typical norms for the error term from Equation (3.10): the 1-norm, the 2-norm and the infinity-norm, the results

can be seen in the appendix, Section 10.2.

We will now first investigate the convergence rate. For this I will use the case of $n = 14$. As we can see in the tables, the system is unstable at $dt_1 = 0.5$ so for this reason I will leave out this data point for the coming analysis.

To find the order of convergence we use the definition of order of convergence, Equation (3.10) and take the logarithm such that we get:

$$\log(E) = \log(C) + p\log(dt).$$

If we now plot the errors in a double logarithm diagram and find the linear regression, then the slope corresponds to the order of convergence. This we have plotted in Figure 14.

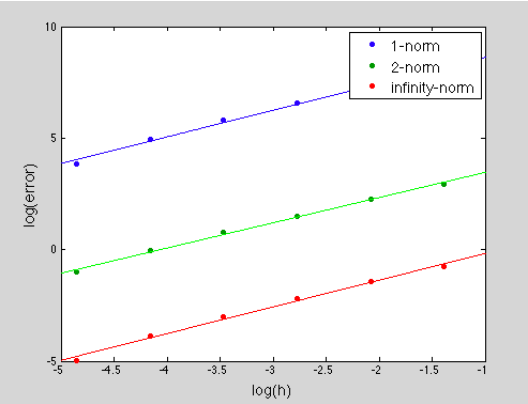


Figure 14: Here we see the difference between the best approximation with a step-size, $dt = \left(\frac{1}{2}\right)^8$ and to all other approximations for $n = 14$ pedestrians.

If we use linear regression to find the slope we get the following values for convergence rates:

Norm	order
1-norm	1.2092
2-norm	1.1457
infinity-norm	1.2110

(7.3)

As we are using ode23s, we have an expected order of 2, which we do not reach. However this can be due to the problem that we are comparing with another approximative solution. So what one does to make sure that we observe the correct order, is to use a three point approximation scheme. Consider the three numerical approximations, v_{dt}, $v_{\frac{dt}{2}}$ and $v_{\frac{dt}{4}}$:

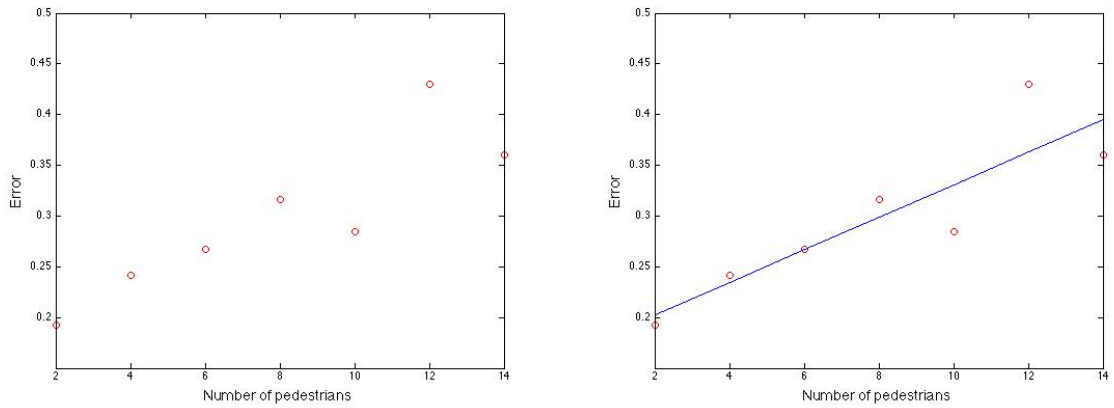
$$\frac{v_{dt} - v_{\frac{dt}{2}}}{v_{\frac{dt}{2}} - v_{\frac{dt}{4}}} \overset{Def.of order}{=} \frac{Cdt^p - C(\frac{dt}{2})^p + O(dt^{p+1})}{C(\frac{dt}{2})^p - C(\frac{dt}{4})^p + O(dt^{p+1})} = \frac{1 - 2^{-p} + O(dt)}{(1 - 2^{-p})2^{-p} + O(dt)} = 2^p + O(dt) \quad (7.4)$$

Using this formula for $dt = 2^{-6}$ and the Euclidean norm on the last step, we get:

$$\log_2\left(\frac{v_{dt} - v_{\frac{dt}{2}}}{v_{\frac{dt}{2}} - v_{\frac{dt}{4}}}\right) \approx p \iff p \approx \log_2\left(\frac{0.00098508}{0.00019992}\right) = 2.3008 \quad (7.5)$$

which looks more like what we expected. Actually it is slightly better than second order. What this tells us, is that the ode23s method converges towards the correct solution as it should, implying that also the error control functions properly. The last matter to consider is that the number of pedestrians has some influence on the error.

We want to investigate the increase in error with an increasing number of pedestrians. Here I will look at the lowest step-size. If we plot it we get Figure 15. It is clear that as we increase



(a) Scatter plot of the error as a function of pedestrians

(b) Linear regression of the corresponding scatter plot

Figure 15: Increase in error as n increases

the number of pedestrians we also increase the error statistically, which can be seen from the linear regression. Note that there is an outlier, this is probably due to the fact that we are using random initial positions, which can have some effect on the error during the simulation.

This increase in error as we increase the number of pedestrians helps explain why the actual runtime of the adaptive solver is of $O(n^{4.5})$ and not $O(n^2)$ as in the original problem. So if we want to keep the same precision we have to lower the step-size of the traditional methods, and lowering the step-size will of course cause the algorithm to run for a longer time. For future investigations it would therefore be of interest to work with numerical methods where the error does not increase along with the number of pedestrians if such methods exist or otherwise try to construct such methods.

Again one idea is that we can use methods from statistical physics, where they use structure preserving methods such as the Störmer-Verlet scheme to integrate the dynamics. A general class of such methods is called symplectic methods, and it is possible to create variable step size methods with such structure preserving properties, though this will most likely destroy the time reversibility. For more on this see [7], as this is outside the scope of this project.

8 Conclusion

We have now seen the huge strength of the microscopic models. Using these type of models we can, by using standard methods from physics, model the force on each pedestrian and therefore more easily describe the dynamics of each pedestrian. We have used this to investigate how the amount of panic influences the system. Here we saw that panic is equivalent to temperature in the system seen from a physical point of view. Increasing this temperature using a random kick caused, what D. Helbing et al. in [3] called freezing by heating. We saw that whenever the temperature increased then the discrepancy also increased causing the system to eventually freeze, such that the pedestrians jam and are prevented from escaping.

We also saw that if we increased the temperature, only by letting the pedestrians move faster, then we at first saw the same behaviour as in the stochastic behaviour case, but when the temperature became too large the pedestrians did not care about the other pedestrians any more and simply bashed through without clogging. However, we predict that if there was less space in the corridor then this would not have happened, this is also what is simulated in [2], where the pedestrians have to move through a door, which is the same as making the corridor smaller. Also, increasing the panic this way does not model a more independent behaviour, which is modelled in the stochastic case.

For our extension we therefore chose to use the stochastic method to increase the panic. This however called for efficient methods to integrate the stochastic dynamical system. We therefore derived the method of N. Grønbech-Jensen and O. Farago [8], and used this for efficiently integrating this system.

Most importantly, we have developed a new extension of the known models. This new extension includes two new types of group behaviour, one that makes people with the same intentions stay closer to each other, this is modelling the same as the tangential velocity force, where pedestrians tend to follow each other. However, we think that it is more intuitively to implement it as a group behaviour. The second type of group behaviour has to the best of our knowledge not been described before. Here we bind family members together in a fixed order, which makes the efficiency or discrepancy differently influenced by the panic. This we saw as it took time to create lanes, whereas in the case where the pedestrians were not tied together they either created lanes almost immediately or not at all.

After having considered the strength of the microscopic model, we also had to consider the weaknesses of this approach. An additional problem is that as we saw is the actual time complexity $O(n^{4.5})$ if we use an adaptive method to solve the resulting system. This really emphasises the fact that microscopic models have a runtime issue. We did, however, see that when using fixed time steps, the problem was integrated with the expected time complexity of $O(n^2)$.

We therefore investigated what phenomena affects the runtime, we could here reject the hypothesis of the adaptive method not being convergent of the expected second order, as we saw an order of convergence of 2.3, which is about the second order, if not only a bit better. Our last idea from where this increased order in time complexity came, was that it rose because the number of pedestrians increased the error of the numerical method. Here we did not have enough data to tell which order this was of, but we did indeed confirm that the error was increasing as we increased the number of pedestrians in the simulation.

9 Future perspectives

From what we have found it would be of interest to consider an investigation of the strong group behaviour applied to friends which is described in the appendix, Section 10.1. Also it could be of interest to consider the nematic order parameter known from liquid crystal physics (see e.g.

[22]) of the family as time lapse, this could help us understand how groups of family members react to panic and other panicking families. If we for instance saw that they were no longer moving after each other, but next to each other, then it would be of interest to investigate whether it would be optimal to put up fences which made this behaviour impossible, forcing them to move after each other.

It would also be of interest to use or create numerical integration schemes, where the error does not increase along with the number of pedestrians. This could for instance as mentioned be symplectic method, which can be implemented with adaptive time stepping though most likely at the cost of time reversibility. By using this we could maybe reduce the time complexity back to $O(n^2)$ for adaptive methods too, and by this increase the step size accordingly, such that for a smaller number of pedestrians the microscopic model would be, if not faster, at least as fast as the macroscopic model and by that merge the two wishes of having both an efficient and a precise model for pedestrian behaviour.

References

[1] D. Hartmann and I. Von Sivers, *Structures first order models for pedestrian dynamics*, American Institute of Mathematical Sciences, Volume **8**, Number **4**, December **2013**, pp. **985-1007**

[2] D. Helbing, I. Farkas and T. Vicsek, *Simulating dynamical features of escape panic*, Nature, Volume **407**, 28 September **2000**, pp. **487-490**

[3] D. Helbing, I. Farkas and T. Vicsek, *Freeing by heating in a driven mesoscopic system*, The American Physical Society, Volume **84**, Number **6**, 7 February **2000**

[4] **L**arge-scale **A**tomic/**M**olecular **M**assively **P**arallel **S**imulator, http://lammps.sandia.gov

[5] D. Helbing and P. Molnár, *Social force model for pedestrian dynamics*, The American Physical Society, Volume **51**, Number **5**, 1 May **1995**

[6] T. Schneider and E. Stoll, *Molecular-dynamics study of a three-dimensional one-component model for distortive phase transitions*, Physical review letter, Volume **17**, Number **3**, 1 February **1978**

[7] E. Hairer, C. Lubich, G. Wanner, *Geometric Numerical Integration*, second edition, ISBN: **978-3-540-81832-8**.

[8] Niels Grønbech-Jensen and Oded Farago (2013), *A simple and effective Verlet-type algorithm for simulating Langevin dynamics*, Molecular physics: An international Journal at the Interface Between Chemistry and Physics, 111 : 8, 983 − 991, DOI: 10.1080/00268976.2012.760055.

[9] Current documentation for the solver: `http://www.mathworks.se/help/matlab/ref/ode23s.html`, January 2, 2015

[10] Predtetschenki, W.M. and Milinski, A. I. Personenströme in Gebauden - Berechnungsmethoden für die Projektierung (Müller, Köln-Braunsfeld, 1971).

[11] Weidmann, U. Transporttehnik der Fussgänger (Institute für Verkehrplanung, Transporttechnik, Strassen- und Eisenbahnbau (IVT), ETH Zürich, 1993).

[12] P. E. Kloeden, E. Platen *Numerical Solution of Stochastic Differential Equations*, ISBN: **978-3-642-08107-1**.

[13] I. Karatzas and S.E. Shreve, Brownian motion and stochastic calculus, Springer Verlag 1999.

[14] B. Øksendal, Stochastic differential equations, Universitext, 6.edition, Springer Verlag 2005.

[15] Current documentation for the Morse implementation in LAMMPS `http://lammps.sandia.gov/doc/bond_morse.html`, January 2, 2015

[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Second edition, ISBN: 978-0-898715-34-7

[17] Verlet, L. (1967). "Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules". Phys. Rev. 159: 98–103. doi:10.1103/physrev.159.98

[18] P. Español and P. B. Warren. Statistical-mechanics of dissipative particle dynamics. Europhysics Letters, 30(4):191–196, MAY 1 1995

[19] E. Hairer, C. Lubich, G. Wanner, *Solving Ordinary Differential equations 1 - Nonstiff problems*, second revised edition, ISBN: **978-3-642-05233-0**.

[20] M. Tenenbaum, H. Pollard *Ordinary Differential Equations*, ISBN: 0486649407

[21] S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J Comp Phys, 117, 1-19 (1995)

[22] P. G. de Gennes and J. Prost, *The Physics of Liquid Crystals*, Second Edition, ISBN: 978-0-19-851785-6

[23] Dormand, J. R.; Prince, P. J. (1980), "A family of embedded Runge-Kutta formulae", Journal of Computational and Applied Mathematics 6 (1): 19–26, doi:10.1016/0771-050X(80)90013-3.

10 Appendix

10.1 Bond Between Friends

Friends on the other hand are fixed in neither distance or permutation, as their group force is not characterised by them holding each others hands, but more like they just really want to stay together. This is sketched as in Figure 6b. As before, we use, the definition of group, $G2_i$, as the group which pedestrian i is in. But in contrast to before the friends do not have to be right next to each other, so here we set:

$$on_{ij} = 1 \text{ if } j \in G2_i \tag{10.1}$$

Here the limit is not our strength and arm length, but how far one can see in a crowded room. Again the bond can break, but it is due to the fact that one friend no longer can see the group with the rest of the friends. Again is it assumed that the bond cannot be re-established. This is due to the fact that if friends go to a concert they have a certain age, and due to this age, the rest of the group will try to get out without the friend, assuming that he is old enough to cope. In the family case the last parent thought mainly of the children, here they do not have as strong a relationship as a family does, so they think more of themselves than keeping the group together.
In this case nobody is actually attached to each other so the length of the bond can be considerably longer, though in case of panic friends do tend to stay close to each other, but do get away from each other more easily. With this in mind we have set the parameters as:

Parameter	Value
r_0	0.75
D	200
α	0.5
R_c	5

(10.2)

10.2 Resuls for convergence test

2 pedestrians

Stepsize	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^4$	$\left(\frac{1}{2}\right)^5$	$\left(\frac{1}{2}\right)^6$	$\left(\frac{1}{2}\right)^7$
1-norm	1403.6131	709.4639	334.4322	158.4052	73.3034	31.2785	10.4164
2-norm	19.72	10.4275	5.0257	2.4209	1.1448	0.51054	0.19276
infinity-norm	0.499	0.26382	0.12688	0.060655	0.028155	0.012032	0.004007

4 Pedestrians

Stepsize	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^4$	$\left(\frac{1}{2}\right)^5$	$\left(\frac{1}{2}\right)^6$	$\left(\frac{1}{2}\right)^7$
1-norm	4896.44	1136.7189	553.9319	269.576	126.0105	54.0907	18.1444
2-norm	51.159	12.3968	6.0896	2.9869	1.4217	0.63664	0.24164
infinity-norm	1.303	0.27136	0.13192	0.064501	0.030175	0.012963	0.0043597

6 Pedestrians

Stepsize	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^4$	$\left(\frac{1}{2}\right)^5$	$\left(\frac{1}{2}\right)^6$	$\left(\frac{1}{2}\right)^7$
1-norm	2791.20	1456.3835	722.7602	350.5079	163.6674	70.0888	23.3998
2-norm	26.17	13.62	6.788	3.3208	1.5797	0.70614	0.26712
infinity-norm	0.5502	0.29666	0.14996	0.072901	0.034109	0.014608	0.0048738

8 Pedestrians

Stepsize	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^4$	$\left(\frac{1}{2}\right)^5$	$\left(\frac{1}{2}\right)^6$	$\left(\frac{1}{2}\right)^7$
1-norm	7116.11	2296.183	1017.6947	498.0556	235.8343	101.9032	33.78
2-norm	53.706	17.0921	7.9532	3.9176	1.8843	0.84666	0.31617
infinity-norm	0.999	0.34376	0.16384	0.077077	0.035537	0.015432	0.005031

10 Pedestrians

Stepsize	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^4$	$\left(\frac{1}{2}\right)^5$	$\left(\frac{1}{2}\right)^6$	$\left(\frac{1}{2}\right)^7$
1-norm	$42028e6$	2283.0215	1056.2669	496.4253	230.4963	98.766	32.8287
2-norm	$4517e5$	15.2262	7.4904	3.5701	1.6908	0.75578	0.28487
infinity-norm	$602e4$	0.33838	0.15383	0.072514	0.033606	0.014414	0.0047802

12 Pedestrians

Stepsize	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^4$	$\left(\frac{1}{2}\right)^5$	$\left(\frac{1}{2}\right)^6$	$\left(\frac{1}{2}\right)^7$
1-norm	4468464	4786.6215	2027.7873	896.3542	410.9417	176.3612	59.1105
2-norm	59767	29.3898	12.2532	5.3982	2.5299	1.1333	0.42986
infinity-norm	1179	0.63209	0.26652	0.098179	0.041718	0.017247	0.0056649

14 Pedestrians

Stepsize	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^4$	$\left(\frac{1}{2}\right)^5$	$\left(\frac{1}{2}\right)^6$	$\left(\frac{1}{2}\right)^7$
1-norm	1114168	3160.4224	1524.4388	712.8302	331.5445	142.2567	47.5373
2-norm	12014	19.1427	9.3651	4.4998	2.1346	0.95388	0.36035
infinity-norm	269.29	0.45852	0.2362	0.11003	0.049693	0.020889	0.0068946

10.3 Freezing by heating model [3] - Matlab implementation

```
%[t,time,res,fence,radius,axess,energy] = freezing3(20,100)
function [t,time,res,fence,radius,axess,energy] = freezing3(n,sigma,pos0)

%%Basic simulation stuff
global obstacle N e0 r
r = 0.5;
endT = 10;
dt = 0.00005;
time = 0:dt:endT;

%%Set up inital positions
N = n;

e0 = ones(2,N);
e0(2,:) = e0(2,:)*0;
e0(1,N/2+1:end) = e0(1,N/2+1:end)*-1;

pos = zeros(2,N,length(time));

if nargin >2
    pos(:,:,1) = pos0;
else
    %Pedestrian position
    pos(:,1:N/2,1) = [8*rand(1,N/2)+1; 3*rand(1,N/2)+1];
    pos(:,N/2+1:end,1) = [8*rand(1,N/2) + 11; 3*rand(1,N/2)+1];
end

%Obstacle placement
obstacle = [[0;20;5;5],[0;20;0;0]];
axess = [-5 25 -5 10]; % axis

%Make the initial position feasable
done = false;
k=0;
while ~done
    %Make sure that they arent on top of each other
    k = k+1;
    for i = 1:N
        for j = i+1:N
            while norm(pos(:,i,1)-pos(:,j,1))<1.2*2*r
                pos(1,j,1) = pos(1,j,1)+r;
            end
        end
    end
    %if the pedestrian is pushed outside the wall, put him inside again
    for i = 1:N
        if pos(1,i,1) > obstacle(2,2)
            pos(1,i,1) = mod(pos(1,i,1),obstacle(2,2));
        elseif pos(1,i,1) < obstacle(1,1)
            while pos(1,i,1) < obstacle(1,1)
```

```
                pos(1,i,1) = pos(1,i,1) + obstacle(2,2) - obstacle(1,1);
            end
        end
    end
    done = true;
    %Are we finished
    for i = 1:N
        for j = i+1:N
            if norm(pos(:,i,1)-pos(:,j,1))<1.2*2*r
                done = false;
            end
        end
    end
    if k ==1000;
        pos(:,1:N/2,1) = [8*rand(1,N/2)+1; 3*rand(1,N/2)+1];
        pos(:,N/2+1:end,1) = [8*rand(1,N/2) + 11; 3*rand(1,N/2)+1];
        display('Could not make inital position, tries again');
        k = 0;
    end
end


posx = pos(1,:,1);
posy = pos(2,:,1);
vel = zeros(1,2*N);

init = [posx' ; posy' ; vel'];

display('Initial position set')
display('Begins to solve IVP')

tic
if sigma == 0
%Use built in method
    options =
        odeset('InitialStep',dt,'RelTol',1e6,'AbsTol',1e6,'MaxStep',dt,'Stats','on');
    [~,y] = ode23s(@funode,[time(1),time(end)],init,options);
else
%Use Stochastic Euler
    y = zeros(length(time),length(init));
    y(1,:) = init;
    for i = 2:1:time(end)/dt
        w = [zeros(1,length(init)/2) ,
            sqrt(sigma)*sqrt(dt)*randn(1,length(init)/2)];
        ode = funode(time(i),y(i-1,:)');
        y(i,:) = y(i-1,:) + dt*ode' + w;
    end
end
t = toc;
vel = zeros(size(pos));
for i = 1:size(y,1)
    pos(1,:,i) = y(i,1:N);
    pos(2,:,i) = y(i,N+1:2*N);
```

```
    vel(1,:,i) = y(i,2*N+1:3*N);
    vel(2,:,i) = y(i,3*N+1:end);
%    for j = 1:N
%        if pos(1,j,i) > obstacle(2,2)
%            pos(1,j,i) = mod(pos(1,j,i),obstacle(2,2));
%        elseif pos(1,j,i) < obstacle(1,1)
%            while pos(1,j,i) < obstacle(1,1)
%                pos(1,j,i) = pos(1,j,i) + obstacle(2,2) - obstacle(1,1);
%            end
%        end
%    end
end

energy = calcenergy(endT,dt,vel,1);


res = pos;
radius = r;
fence = obstacle;

end



function [f] = force(y)
global obstacle N e0 r
tempF = zeros(2,N);

pos = [y(1:N)' ; y(N+1:2*N)'];
for j = 1:N
    if pos(1,j) > obstacle(2,2)
        pos(1,j) = mod(pos(1,j),obstacle(2,2)) + obstacle(1,1);
    end
    if pos(1,j) < obstacle(1,1)
        while pos(1,j) < obstacle(1,1)
            pos(1,j) = pos(1,j) + obstacle(2,2) - obstacle(1,1);
        end
    end
end
vel = [y(2*N+1:3*N)' ; y(3*N+1:end)'];

%%Different parameters
r = 0.5;
A = 0.2;
B = 2;
v0 = 1;
tau = 0.2;
m = 1;

%%Looping over all pedestrians
%s = rng;
walltmp = [obstacle(2,2)-obstacle(1,1); 0];
for i = 1:N
```

```
    %%Force against the wall
    EwallU = @(x1,x2) A/((abs(x2-obstacle(3,1))-r)^B);
    EwallD = @(x1,x2) A/((abs(x2-obstacle(3,2))-r)^B);
    Ewall = @(x1,x2) EwallU(x1,x2) + EwallD(x1,x2);
    Fwall = -grad(Ewall,pos(:,i));


    %%force against other pedestrians
    Fsoz = 0;
    for j = 1:N
        if i~=j
            if abs(pos(1,j)-pos(1,i))<(obstacle(2,2)-obstacle(1,1))/4
                Esoz = @(x1,x2) A*(norm([x1;x2]-pos(:,j))-2*r)^(-B);
                Fsoz = Fsoz -grad(Esoz,pos(:,i));
            elseif abs(pos(1,j)-pos(1,i))>(obstacle(2,2)-obstacle(1,1))*3/4 &&
                pos(1,i)>(obstacle(2,2)-obstacle(1,1))/2
                Esoz = @(x1,x2) A*(norm([x1;x2]-walltmp-pos(:,j))-2*r)^(-B);
                Fsoz = Fsoz -grad(Esoz,pos(:,i));
            elseif abs(pos(1,j)-pos(1,i))>(obstacle(2,2)-obstacle(1,1))*3/4 &&
                pos(1,i)<(obstacle(2,2)-obstacle(1,1))/2
                Esoz = @(x1,x2) A*(norm([x1;x2]+walltmp-pos(:,j))-2*r)^(-B);
                Fsoz = Fsoz -grad(Esoz,pos(:,i));
            end
        end
    end

    %Total force
    dir = v0*(e0(:,i))/tau;
    drag = - vel(:,i)/ tau;
%    randomF = randn(2,1)*sqrt(sigma);
    randomF = 0;
    tempF(:,i) = dir + drag + (randomF + Fsoz + Fwall)/m;

end
%rng(s);
f = tempF(1,:)';
f = [f;tempF(2,:)'];

end

function [energy] = calcenergy(endT,dt,vel,v0)
global N e0
energy = zeros(endT/dt,1);
for t = 1:endT/dt
    sumterm = 0;
    for i=1:N
        sumterm = sumterm + norm(vel(:,i,t)-v0*e0(:,i))^2/N;
    end
    energy(t) = sumterm;
end
end
```

```
function dydx = funode(~,y)
global N
dydx = zeros(4*N,1);
f = force(y);

for i = 1:4*N
   if i <=2*N
       dydx(i) = y(2*N+i);
   else
       dydx(i) = f(i-2*N);
   end
end

end


function [dydx] = grad(f,pos)
dydx = [(f(pos(1)+sqrt(eps),pos(2))-f(pos(1),pos(2)))/sqrt(eps);
        (f(pos(1),pos(2)+sqrt(eps))-f(pos(1),pos(2)))/sqrt(eps)];

end
```

10.4 Model from Nature [2] - Matlab implementation

```
%[t,time,res,fence,radius,axess,energy] = freezing3(20,100)
function [t,time,res,fence,radius,axess,energy] = nature3(n,pos0)

%%Basic simulation stuff
global obstacle N e0 r
r = 0.5;
endT = 20;
dt = 0.01;
time = 0:dt:endT;

%%Set up inital positions
N = n;

e0 = ones(2,N);
e0(2,:) = e0(2,:)*0;
e0(1,N/2+1:end) = e0(1,N/2+1:end)*-1;

pos = zeros(2,N,endT/dt);

if nargin >2
    pos(:,:,1) = pos0;
else
    %Pedestrian position
    pos(:,1:N/2,1) = [8*rand(1,N/2)+1; 3*rand(1,N/2)+1];
    pos(:,N/2+1:end,1) = [8*rand(1,N/2) + 11; 3*rand(1,N/2)+1];
end
```

```
%Obstacle placement
obstacle = [[0;20;5;5],[0;20;0;0]];
axess = [-5 25 -5 10]; % axis

%Make the initial position feasable
done = false;
k=0;
while ~done
    %Make sure that they arent on top of each other
    k = k+1;
    for i = 1:N
        for j = i+1:N
            while norm(pos(:,i,1)-pos(:,j,1))<1.2*2*r
                pos(1,j,1) = pos(1,j,1)+r;
            end
        end
    end
    %if the pedestrian is pushed outside the wall, put him inside again
    for i = 1:N
        if pos(1,i,1) > obstacle(2,2)
            pos(1,i,1) = mod(pos(1,i,1),obstacle(2,2));
        elseif pos(1,i,1) < obstacle(1,1)
            while pos(1,i,1) < obstacle(1,1)
                pos(1,i,1) = pos(1,i,1) + obstacle(2,2) - obstacle(1,1);
            end
        end
    end
    done = true;
    %Are we finished
    for i = 1:N
        for j = i+1:N
            if norm(pos(:,i,1)-pos(:,j,1))<1.2*2*r
                done = false;
            end
        end
    end
    if k ==1000;
        k=0;
        pos(:,1:N/2,1) = [8*rand(1,N/2)+1; 3*rand(1,N/2)+1];
        pos(:,N/2+1:end,1) = [8*rand(1,N/2) + 11; 3*rand(1,N/2)+1];
        display('Could not set inital position, try over')
    end
end

posx = pos(1,:,1);
posy = pos(2,:,1);
vel = zeros(1,2*N);

init = [posx' ; posy' ; vel'];
display('Inital position set, Solving IVP')
tic
%Use built in method
```

```
    options =
        odeset('InitialStep',dt,'RelTol',1e6,'AbsTol',1e6,'MaxStep',dt,'Stats','on');
    [~,y] = ode45(@funode,[time(1):dt:time(end)],init,options);
t = toc;
vel = zeros(size(pos));
for i = 1:size(y,1)
    pos(1,:,i) = y(i,1:N);
    pos(2,:,i) = y(i,N+1:2*N);
    vel(1,:,i) = y(i,2*N+1:3*N);
    vel(2,:,i) = y(i,3*N+1:end);
end

energy = calcenergy(endT,dt,vel,1);


res = pos;
radius = r;
fence = obstacle;

end



function [f] = force(y)
global obstacle N e0 r
k = 1;

x = [y(1:N)' ; y(N+1:2*N)'];
for j = 1:N
    if x(1,j) > obstacle(2,2)
        x(1,j) = mod(x(1,j),obstacle(2,2)) + obstacle(1,1);
    end
    if x(1,j) < obstacle(1,1)
        while x(1,j) < obstacle(1,1)
            x(1,j) = x(1,j) + obstacle(2,2) - obstacle(1,1);
        end
    end
end
v = [y(2*N+1:3*N)' ; y(3*N+1:end)'];

%%Different parameters
r = 0.5;
A = 2000;
B = 0.08;
v0 = 1;
tau = 0.5;
walltmp = [obstacle(2,2)-obstacle(1,1); 0];
m = 80;
kappa_1 = 1.2e5;
kappa_2 = 2.4e5;
lambda = 0.5*ones(N,1);
f_soz = zeros(2,N,N); % territorial effect
f = zeros(2,N);
```

```
f_ph = f_soz;
const_n = [1;0];

%%Force against the wall
    for i = 1:N
        tmp_wall = 0;
            for l = 1:size(obstacle,2)
                normal = [-obstacle(4,l)+obstacle(3,l);obstacle(2,l)-obstacle(1,l)];
                normal = normal/norm(normal);
                d = (x(:,i) - [obstacle(1,l);obstacle(3,l)])'*normal;
                normal = sign(d)*normal;
                d = abs(d);
                tmp = A*exp((r - d)/B);
                omega = r - d;
                tmp_1 = [obstacle(2,l);obstacle(4,l)] -
                    [obstacle(1,l);obstacle(3,l)];
                if tmp_1'*(x(:,i,k) - [obstacle(1,l);obstacle(3,l)]) >= 0 &&
                    -tmp_1'*(x(:,i,k) - [obstacle(2,l);obstacle(4,l)]) >= 0
                    if omega < 0
                        f_wall = tmp*normal;
                    else
                        f_wall = (tmp + kappa_1*omega)*normal -
                            kappa_2*omega*v(:,i)'*...
                        [-normal(2);normal(1)]*[normal(2);normal(1)];
                    end
                else
                    f_wall = 0;
                end
                tmp_wall = tmp_wall + f_wall;
                if x(2,i)<8-r && x(2,i)>7+r
                    tmp_wall = 0;
                end
             end

    %%force against other pedestrians

        tmp_3 = 0;
        for j = 1:N
            if i ~= j
                if abs(x(1,j)-x(1,i))<(obstacle(2,2)-obstacle(1,1))/4 %If they are
                    close enough calculate the social force
                    if (x(:,j) - [77;0])'*const_n < 0
                        tmp_1 = ((r + r) - norm(x(:,i)-x(:,j)))/B;
                        n(:,i,j) = (x(:,i)-x(:,j))/norm(x(:,i)-x(:,j));
                        if v(1,i) == 0 && v(2,i) == 0
                            phi(i,j) = acos(0);
                        else
                            phi(i,j) = acos(-n(:,i,j)'*v(:,i)/norm(v(:,i,k)));
                        end
                        tmp_2 = lambda(i)+(1-lambda(i))*(1+cos(phi(i,j)))/2;
                        f_soz(:,i,j) = A*exp(tmp_1)*n(:,i,j)*tmp_2;
                        omega = (r+r) - norm(x(:,i)-x(:,j));
                        if (omega < 0)
```

```
            f_ph(:,i,j) = 0;
        else
            f_ph(:,i,j) = kappa_1*omega*n(:,i,j) +
                kappa_2*omega*(v(:,i)-v(:,j))' ...
                *[-n(2,i,j);n(1,i,j)]*[-n(2,i,j);n(1,i,j)];
        end
    tmp_3 = tmp_3 + f_soz(:,i,j) + f_ph(:,i,j);
    end
    %Make the boundary continuous
elseif abs(x(1,j)-x(1,i))>(obstacle(2,2)-obstacle(1,1))*3/4 &&
    x(1,i)>(obstacle(2,2)-obstacle(1,1))/2
    if (x(:,j)-walltmp - [77;0])'*const_n < 0
        tmp_1 = ((r + r) - norm(x(:,i)-walltmp-x(:,j)))/B;
        n(:,i,j) =
            (x(:,i)-walltmp-x(:,j))/norm(x(:,i)-walltmp-x(:,j));
        if v(1,i) == 0 && v(2,i) == 0
            phi(i,j) = acos(0);
        else
            phi(i,j) = acos(-n(:,i,j)'*v(:,i,k)/norm(v(:,i,k)));
        end
        tmp_2 = lambda(i)+(1-lambda(i))*(1+cos(phi(i,j)))/2;
        f_soz(:,i,j) = A*exp(tmp_1)*n(:,i,j)*tmp_2;
        omega = (r+r) - norm(x(:,i)-walltmp-x(:,j));
        if (omega < 0)
            f_ph(:,i,j) = 0;
        else
            f_ph(:,i,j) = kappa_1*omega*n(:,i,j) + kappa_2*omega ...
          *(v(:,i)-v(:,j))'*[-n(2,i,j);n(1,i,j)]*[-n(2,i,j);n(1,i,j)];
        end
    tmp_3 = tmp_3 + f_soz(:,i,j) + f_ph(:,i,j);
    end
elseif abs(x(1,j)-x(1,i))>(obstacle(2,2)-obstacle(1,1))*3/4 &&
    x(1,i)<(obstacle(2,2)-obstacle(1,1))/2
    if (x(:,j)+walltmp - [77;0])'*const_n < 0
        tmp_1 = ((r + r) - norm(x(:,i)+walltmp-x(:,j)))/B;
        n(:,i,j) =
            (x(:,i)+walltmp-x(:,j))/norm(x(:,i)+walltmp-x(:,j));
        if v(1,i) == 0 && v(2,i) == 0
            phi(i,j) = acos(0);
        else
            phi(i,j) = acos(-n(:,i,j)'*v(:,i,k)/norm(v(:,i,k)));
        end
        tmp_2 = lambda(i)+(1-lambda(i))*(1+cos(phi(i,j)))/2;
        f_soz(:,i,j) = A*exp(tmp_1)*n(:,i,j)*tmp_2;
        omega = (r+r) - norm(x(:,i)+walltmp-x(:,j));
        if (omega < 0)
            f_ph(:,i,j) = 0;
        else
            f_ph(:,i,j) = kappa_1*omega*n(:,i,j) + kappa_2*omega ...
          *(v(:,i)-v(:,j))'*[-n(2,i,j);n(1,i,j)]*[-n(2,i,j);n(1,i,j)];
        end
    tmp_3 = tmp_3 + f_soz(:,i,j) + f_ph(:,i,j);
    end
```

```
                end
            end

        end
%         size(e0)
%         size(v)
%         size(tau)
%         size(v0)
        tempf(:,i) = (v0*e0(:,i) - v(:,i))/ tau + (tmp_3 + tmp_wall)/m;
        f = tempf';
    end
end


function [energy] = calcenergy(endT,dt,vel,v0)
global N e0
energy = zeros(endT/dt,1);
for t = 1:endT/dt
    sumterm = 0;
    for i=1:N
        sumterm = sumterm + norm(vel(:,i,t)-v0*e0(:,i))^2/N;
    end
    energy(t) = sumterm;
end
end

function dydx = funode(~,y)
global N
dydx = zeros(4*N,1);
f = force(y);
f2 = f;
for i = 1:4*N
   if i <=2*N
       dydx(i) = y(2*N+i);
   else
       dydx(i) = f(i-2*N);
   end
end
end
```

10.5 Model including group behavior, Section 6 -LAMMPS implementation

```
#
#  This is a file for defining how LAMMPS should run a simulation
#  for a explanation of the commands see lammps.sandia.gov/doc/Section_commands.html
#

#set up basic simulation stuff
units          si
atom_style     molecular
```

```
boundary      p  p  p

dimension 2

newton on off

neighbor 3.0 bin
#create output catalogs, and clean them if old files exists
shell mkdir conf
shell mkdir img
shell "rm conf/*"
shell "rm img/*"

#read input configuraiton
#read_restart finalt3000000.restart
read_data config.input

change_box all boundary p p p

variable tau equal 0.5
variable v0 equal 1
variable size equal 0.6
variable b equal 0.08
variable A equal 2000
variable mass equal 80

#Make walls repulsive
variable kwall equal 1.1
variable dummy equal 0.25

#define groups of beads based on their types
group save    type   2 3
group rod     type   2 3

#Define the two groups
group kick2   type   2
group kick3   type   3


#define masses of all beads as M
mass          * ${mass}

pair_style none

pair_style    born 20
#DPD pair interaction parameters between bead pairs
#pair_coeff   * *    0   0.08   ${size}    0  0  #Group 1 - Group 1


#           type  type  Coeff  B_i   diameter delete terms
pair_coeff    2     2    100   0.08   ${size}    0  0  #Group 1 - Group 1
pair_coeff    2     3    160   0.08   ${size}    0  0  #Group 1 - Group 2
pair_coeff    3     3    100   0.08   ${size}    0  0  #Group 2 - Group 2
```

```
pair_coeff     1     3    0       0.08    0      0  0  #To make things work
pair_coeff     1     2    0       0.08    0      0  0  #To make things work
pair_coeff     1     1    0      0.08    0      0  0  #To make things work

#define bonded interactions
bond_style bondnicky
#       type    D      alpha     r0
bond_coeff 1    2000     3       1     #Family
bond_coeff 2    100      0.5     1.5   #Friends

special_bonds lj 0 1 1

compute rdist all pair/local dist
compute mdist all reduce min c_rdist
compute bdist all bond/local dist
compute bmax all reduce max c_bdist
compute bave all reduce ave c_bdist


#define angular interaction
angle_style harmonic
angle_coeff * 10.0 180.0

#angle_style none

# specify timestep
# ZQEX: high spring contants => slower integration steps.
timestep       0.01



#define how images of the configuration are saved (note only rods PT)
dump dump2 rod image 1 img/eq*.jpg  type type adiam ${size} bond type 0.2 zoom 4
    size 1250 450 view 0.0 90.0
dump_modify dump2 backcolor white boxcolor black bcolor * darkmagenta
dump_modify dump2 pad 8



compute ytemp all temp/partial 0 1 0

#define how often, and what termodynamic information is printed
thermo         1
thermo_style custom step temp c_ytemp ke pe epair ebond eangle c_mdist fmax c_bmax
    c_bave

# thermodynamic information:
# step   timestep
# temp   temperature
# press  pressure
# ke     kinetic energy
# pe     total potential energy
# epair       pair-energy contribution
```

```
# ebond       bond energy contribution
# eangle      angular energy contribution


minimize 1e-8 1e-8 1000 1000
undump dump2

compute rke all kez restframe 1 0 0 0 restframe 2 1 0 0 restframe 3 -1 0 0

compute collision kick2 group/group kick3 pair yes


compute rvel kick2 reduce ave vx vy
compute lvel kick3 reduce ave vx vy

dump dump2 rod image 100 img/t*.jpg  type type adiam ${size} bond type 0.2 zoom 4
    size 1250 450 view 0.0 90.0
dump_modify dump2 backcolor white boxcolor black bcolor * darkmagenta
dump_modify dump2 pad 8

timestep      0.001

change_box all boundary p f p

fix walls all wall/reflect ylo EDGE yhi EDGE

# Top and Bottom wall
variable e equal 2.71828182846
variable RtoWT atom v_A/v_b/v_mass*v_e^((0.3-y)/v_b)
variable RtoWB atom v_A/v_b/v_mass*v_e^((0.3-(5-y))/v_b)
variable forceY atom v_RtoWT-v_RtoWB

fix wallsf all addforce 0 v_forceY 0


#Make one type of the people move in one direction

variable extfl  equal -${mass}*${v0}/${tau}
variable extfr  equal  ${mass}*${v0}/${tau}

fix fixkick2 kick2 addforce v_extfr 0 0
fix fixkick1 kick3 addforce v_extfl 0 0

#Make the other type of people move in the opposite direction

#Construct the drag
variable dragx atom -v_mass/v_tau*vx
variable dragy atom -v_mass/v_tau*vy
fix drag all addforce v_dragx v_dragy 0

#construct random kick
variable sigma equal 10^(0.0)
variable randfx atom normal(0,v_sigma,3453453)
```

```
variable randfy atom normal(0,v_sigma,3453453)

#fix rnd all addforce v_randfx v_randfy 0


fix fixbondkill all bondkill 1 killbond 1 2 killbond 2 6 seed 23581 logbonds
    bonddynamic.txt
fix syncronize all sync 1

# integrator of the chosen dynamics
fix integrator all nve/limit 0.0034
#fix integrators all langevin 270.0 270 2 81075
#fix temp all temp/rescale 1 0 0 0 1

#Make 2D
fix beads all enforce2d

#define how often configurations are saved
dump dump1 save custom 1000 conf/t*   id mol type x y vx vy fx fy
dump_modify dump1 pad 8


thermo 10
thermo_style custom step ke c_rke pe  epair ebond eangle c_collision &
c_rvel[1] c_rvel[2] c_lvel[1] c_lvel[2] c_bmax c_bave &
#f_walls   f_fixkick1 f_fixkick2 f_drag f_rnd etotal



#number of steps to run
run            400000

#save a binary restart file, allowing us to continue simulation
write_restart finalt*.restart
```

10.6 Creating initial positions and bonds, Section 6) - Perl implementation

```
#!/usr/bin/perl

#
# Makes an initial configuration with a mixture of rod like molecules
# and solvent beads in a box of fixed size.
#
#  gen_rod.pl <volume faction of rods>
#
# solvent = bead type 1
# rod made of bead type 2.

my $solventtype = 1;
my @rodtype11   = (2,2,2);
```

```
my @rodtype12   = (2,2,2);
my @rodtype21   = (3,3,3);
my @rodtype22   = (3,3,3);

# Specity volume fraction of families compared to friends on the commandline, if
    not specified then
# a 50/50 system is produced.

my $rodproc=shift @ARGV;
$rodproc=0.5 unless (defined $rodproc);

#define simulation box and density line 22
my $xaxis=20;
my $yaxis=5;
my $zaxis=1;


use lib $ENV{HOME};
use LammpsConfig;
my $config=LammpsConfig->new();

#define simulation box
$config->SetBoxX(0,$xaxis);
$config->SetBoxY(0,$yaxis);
$config->SetBoxZ(-$zaxis/2,$zaxis/2);


#define number of interactions
$config->SetBondTypes(2);
$config->SetAngleTypes(2);
$config->SetAtomTypes(3);

#total number of beads defined by box volume and density line 40
my $nbead=60;
my $length11=scalar(@rodtype11);
my $length12=scalar(@rodtype12);
my $length21=scalar(@rodtype21);
my $length22=scalar(@rodtype22);
my $proc11=$rodproc*$length11/($length11+$length12);
my $proc12=(1-$rodproc)*$length12/($length11+$length12);
my $proc21=$rodproc*$length21/($length21+$length22);
my $proc22=(1-$rodproc)*$length22/($length21+$length22);

#Number of rod molecules and solvent beads
my $Nrodbeads  = $nbead;
my $Nrod11     = $Nrodbeads*$proc11/$length11;
my $Nrod12     = $Nrodbeads*$proc12/$length12;
my $Nrod21     = $Nrodbeads*$proc21/$length21;
my $Nrod22     = $Nrodbeads*$proc22/$length22;


printf "Rods11: $Nrod11 Rods12: $Nrod12 Rods21: $Nrod21 Rods22: $Nrod22 \n";
```

```
#add Rods11
for (my $i=0;$i<$Nrod11;$i++)
   {
        $tx = rand(3)+5;
        $ty = rand(4) + 1;
       AddFamily(\@rodtype12,$tx,$ty,0,0,-1,0);
   }

#add Rods21
for (my $i=0;$i<$Nrod21;$i++)
{
        $tx = rand(3)+$xaxis-5;
        $ty = rand(4) + 1;
    AddFamily(\@rodtype21,$tx,$ty,0,0,1,0);
}

#add Rods12
for (my $i=0;$i<$Nrod12;$i++)
   {
        $tx = rand(3)+5;
        $ty = rand(4) + 1;
       AddFriends(\@rodtype12,$tx,$ty,0,0,-1,0);
   }

#add Rods22
for (my $i=0;$i<$Nrod22;$i++)
{
        $tx = rand(3)+$xaxis-5;
        $ty = rand(4) + 1;
    AddFriends(\@rodtype22,$tx,$ty,0,0,1,0);
}

#save configuration
$config->SaveInput("config.input");


sub AddFamily
#auxillary function that adds a molecule defined by an array to a simulation box
{
   my $aref=shift;
   my $x=shift;
   my $y=shift;
   my $z=shift;
   my $dx=shift;
   my $dy=shift;
   my $dz=shift;
   my @mol=@{ $aref };

   my $m=$config->AddMolecule();
   $aold=undef; $aoldold=undef;
   for ($i=0;$i<scalar(@mol);$i++)
     {
       $a=$config->AddMoleculeAtomType($m, $x,$y,$z, $mol[$i] ); $nbead++;
```

```
      $x+=$dx; $y+=$dy; $z+=$dz;

      $config->AddBond($aold,$a,1) if (defined $aold and defined $a);
      $config->AddAngle($aoldold,$aold,$a,1) if (defined $aoldold and defined
          $aold and defined $a);
      $aoldold=$aold; $aold=$a;
    }
}

sub AddFriends
#auxillary function that adds a molecule defined by an array to a simulation box
{
   my $aref=shift;
   my $x=shift;
   my $y=shift;
   my $z=shift;
   my $dx=shift;
   my $dy=shift;
   my $dz=shift;
   my @mol=@{ $aref };

   my $m=$config->AddMolecule();
   $aold=undef; $aoldold=undef;
   for ($i=0;$i<scalar(@mol);$i++)
     {
       $a=$config->AddMoleculeAtomType($m, $x,$y,$z, $mol[$i] ); $nbead++;
       $x+=$dx; $y+=$dy; $z+=$dz;

       $config->AddBond($aold,$a,2) if (defined $aold and defined $a);
       $config->AddAngle($aoldold,$aold,$a,2) if (defined $aoldold and defined
           $aold and defined $a);
       $aoldold=$aold; $aold=$a;
     }
}
```

10.7 Bond type implementation, Section 6 - C++ implementation

Only minor changes from standard implementation of the Morse potential has been made [15]

```
/* ----------------------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Copyright (2003) Sandia Corporation. Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software. This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */
```

```
/* ----------------------------------------------------------------------
   Contributing author: Jeff Greathouse (SNL)
------------------------------------------------------------------------- */

#include "math.h"
#include "stdlib.h"
#include "bond_nicky.h"
#include "atom.h"
#include "neighbor.h"
#include "domain.h"
#include "comm.h"
#include "force.h"
#include "memory.h"
#include "error.h"

using namespace LAMMPS_NS;

/* ---------------------------------------------------------------------- */

BondNicky::BondNicky(LAMMPS *lmp) : Bond(lmp) {}

/* ---------------------------------------------------------------------- */

BondNicky::~BondNicky()
{
  if (allocated) {
    memory->destroy(setflag);
    memory->destroy(d0);
    memory->destroy(alpha);
    memory->destroy(r0);
  }
}

/* ---------------------------------------------------------------------- */

void BondNicky::compute(int eflag, int vflag)
{
  int i1,i2,n,type;
  double delx,dely,delz,ebond,fbond;
  double rsq,r,dr,ralpha;

  ebond = 0.0;
  if (eflag || vflag) ev_setup(eflag,vflag);
  else evflag = 0;

  double **x = atom->x;
  double **f = atom->f;
  int **bondlist = neighbor->bondlist;
  int nbondlist = neighbor->nbondlist;
  int nlocal = atom->nlocal;
  int newton_bond = force->newton_bond;
```

```
  for (n = 0; n < nbondlist; n++) {
    i1 = bondlist[n][0];
    i2 = bondlist[n][1];
    type = bondlist[n][2];

    delx = x[i1][0] - x[i2][0];
    dely = x[i1][1] - x[i2][1];
    delz = x[i1][2] - x[i2][2];

    rsq = delx*delx + dely*dely + delz*delz;
    r = sqrt(rsq);
    dr = r - r0[type];
    ralpha = exp(-alpha[type]*dr);

    // force & energy

    if (r > 0.0) fbond = -2.0*d0[type]*alpha[type]*(1-ralpha)*ralpha/r;
    else fbond = 0.0;

//    if (eflag) ebond = d0[type]*(1-ralpha)*(1-ralpha);
// Zqex:
    if (eflag) ebond = d0[type]*(1-ralpha)*(1-ralpha)-d0[type];

    // apply force to each of 2 atoms

    if (newton_bond || i1 < nlocal) {
      f[i1][0] += delx*fbond;
      f[i1][1] += dely*fbond;
      f[i1][2] += delz*fbond;
    }

    if (newton_bond || i2 < nlocal) {
      f[i2][0] -= delx*fbond;
      f[i2][1] -= dely*fbond;
      f[i2][2] -= delz*fbond;
    }

    if (evflag) ev_tally(i1,i2,nlocal,newton_bond,ebond,fbond,delx,dely,delz);
  }
}

/* ---------------------------------------------------------------------- */

void BondNicky::allocate()
{
  allocated = 1;
  int n = atom->nbondtypes;

  memory->create(d0,n+1,"bond:d0");
  memory->create(alpha,n+1,"bond:alpha");
  memory->create(r0,n+1,"bond:r0");
  memory->create(setflag,n+1,"bond:setflag");
  for (int i = 1; i <= n; i++) setflag[i] = 0;
```

```
}

/* ----------------------------------------------------------------------
   set coeffs for one type
------------------------------------------------------------------------- */

void BondNicky::coeff(int narg, char **arg)
{
  if (narg != 4) error->all(FLERR,"Incorrect args for bond coefficients");
  if (!allocated) allocate();

  int ilo,ihi;
  force->bounds(arg[0],atom->nbondtypes,ilo,ihi);

  double d0_one = force->numeric(FLERR,arg[1]);
  double alpha_one = force->numeric(FLERR,arg[2]);
  double r0_one = force->numeric(FLERR,arg[3]);

  int count = 0;
  for (int i = ilo; i <= ihi; i++) {
    d0[i] = d0_one;
    alpha[i] = alpha_one;
    r0[i] = r0_one;
    setflag[i] = 1;
    count++;
  }

  if (count == 0) error->all(FLERR,"Incorrect args for bond coefficients");
}

/* ----------------------------------------------------------------------
   return an equilbrium bond length
------------------------------------------------------------------------- */

double BondNicky::equilibrium_distance(int i)
{
  return r0[i];
}

/* ----------------------------------------------------------------------
   proc 0 writes to restart file
------------------------------------------------------------------------- */

void BondNicky::write_restart(FILE *fp)
{
  fwrite(&d0[1],sizeof(double),atom->nbondtypes,fp);
  fwrite(&alpha[1],sizeof(double),atom->nbondtypes,fp);
  fwrite(&r0[1],sizeof(double),atom->nbondtypes,fp);
}

/* ----------------------------------------------------------------------
   proc 0 reads from restart file, bcasts
------------------------------------------------------------------------- */
```

```
void BondNicky::read_restart(FILE *fp)
{
  allocate();

  if (comm->me == 0) {
    fread(&d0[1],sizeof(double),atom->nbondtypes,fp);
    fread(&alpha[1],sizeof(double),atom->nbondtypes,fp);
    fread(&r0[1],sizeof(double),atom->nbondtypes,fp);
  }
  MPI_Bcast(&d0[1],atom->nbondtypes,MPI_DOUBLE,0,world);
  MPI_Bcast(&alpha[1],atom->nbondtypes,MPI_DOUBLE,0,world);
  MPI_Bcast(&r0[1],atom->nbondtypes,MPI_DOUBLE,0,world);

  for (int i = 1; i <= atom->nbondtypes; i++) setflag[i] = 1;
}

/* ----------------------------------------------------------------------
   proc 0 writes to data file
------------------------------------------------------------------------- */

void BondNicky::write_data(FILE *fp)
{
  for (int i = 1; i <= atom->nbondtypes; i++)
    fprintf(fp,"%d %g %g %g\n",i,d0[i],alpha[i],r0[i]);
}

/* ---------------------------------------------------------------------- */

double BondNicky::single(int type, double rsq, int i, int j,
                         double &fforce)
{
  double r = sqrt(rsq);
  double dr = r - r0[type];
  double ralpha = exp(-alpha[type]*dr);
  fforce = 0;
  if (r > 0.0) fforce = -2.0*d0[type]*alpha[type]*(1-ralpha)*ralpha/r;
//  return d0[type]*(1-ralpha)*(1-ralpha);
// zqex:
  return d0[type]*(1-ralpha)*(1-ralpha)-d0[type];
}
```