Benjamin Franz
University of Oxford
College: Mansfield College
E-Mail: benjamin.franz@mansfield.ox.ac.uk
Reading for: MSc in Mathematical Modelling and Scientific Computing

Synchronisation Properties of an Agent-Based Animal Behaviour Model

Master's Dissertation

Supervisor: Dr Mason Alexander Porter
Prof Marian Stamp Dawkins
Handed in: September 3, 2009

Abstract

We developed an agent-based model of animal behaviour inspired by the model presented in [6]. The original model was expanded to allow the agents to eat and rest. The decisions for state changes are governed by a deterministic finite-state machine. In order to evaluate the validity of this model, we compared simulation results with real data obtained from video footage of different farms. During the validation process, we used a simulated annealing algorithm to find parameter sets that allow for a best match with the data. Because the agents are situated in a mutual environment, they can influence each others' behaviour. This effect, combined with the constant change in the agents' state between resting and standing, motivated the interpretation of the multi-agent system as a combination of weakly coupled oscillators. Using this interpretation, we investigated the possibility of synchronisation between the agents and we developed different order parameters to measure the amount of emerging synchrony. Different biological studies like [9] and [19] show that both growth rate and welfare of beef cows increase when the herd acts in unison. This renders synchronisation as an effect that is highly intended by farmers. We hence developed an algorithm that tries to optimise the layout of feeding and bedding areas in order to find a barn design that favours synchronisation. We successfully fit our model to the data, but incompleteness in the data set did not allow for a conclusive model evaluation. During the simulations, we detected emerging synchrony and increased it using the optimisation algorithm. We concluded that an agent-based model with a small set of rules can describe certain aspects of cattle behaviour and can yield to complex behaviour like synchrony.

Acknowledgements

I take pleasure in expressing my gratitude to those who have helped me with this project. Without the help of my supervisors Marian Dawkins and Mason Porter this project would have been almost impossible to master. Our weekly talks helped me developing most of the ideas in this dissertations. During the final write-up they both happily offered to read through my drafts and made very useful comments, indeed. Additionally, I am thankful for the feedback I received of Mark Richardson, Radek Erban, Iain Couzin, Steven Strogatz and Peter Mucha towards the project in general and the dissertation in particular. The project was based on prior work done by Nicole Milligan and Tom Shaw. I was delighted that I had the chance of meeting Tom and that he helped me understand the different data sets. In a more general note, I would like to thank our course organiser Kathryn Gillow, who never hesitated to reply to urgent requests regarding the progress of the course or other matters.

I am also very thankful to the people who gave me the opportunity of doing this course in the first place. First, my scholarship "Studienstiftung des Deutschen Volkes" for their generous funding and second – but more importantly – my parents for providing four helping hands whenever I needed them. During the final work on this dissertation I was especially grateful to my dear friends Chloé Joyeux and Ricardo Engel for providing me with their company and their joy. Additionally, I very much appreciate the help I received on my former essays from Erin Null, Benjamin Clark and John Platt.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Animal behaviour research was initially driven mainly by observations. One of the first written records of the systematic study of animal behaviour was Charles Darwin's "On the Origin of Species by Means of Natural Selection" [7], which includes a full chapter about what he called "Instinct". With the development of computers during the 20th century, modelling and simulations started to play an important role in animal behaviour research. Nowadays, models for different kinds of animals exist and are used in combination with simulations and experiments to gain deeper insight into animal behaviour and to explore possible ways for human use.

In [6], Couzin et al. presented a model for the movement of fish schools and bird flocks that was able to produce various kinds of swarming behaviour observed in nature. The model was capable of creating a chaotic swarm as well as a nicely organised group movement or movement in a torus depending on the different parameters chosen. This model might be adequate for swarm movement, but it does not include effects like eating or resting. These effects play an important role in the behaviour of big farm animals like cows. This dissertation will extend Couzin et al.'s model by incorporating eating and resting periods in order to resemble the behaviour of big and slow moving animals – in particular, cattle.

Using this model, we investigate the possibility of emerging synchrony in a cattle herd by interpreting the agents as a set of weakly coupled oscillators. Different kinds of order parameters will be used to measure the amount of synchrony. This investigation has a special importance for beef cows because of their unique double-digestive system [16]. The first digestion takes place during the actual eating process when the cow usually stands upright. Thereafter it moves to the bedding area to lie down. This is when the second digestion, often referred to as 'chewing the cud', happens. Fisher et al. showed in [9] that the growth rate of beef cows is higher when the resting period is not disturbed and the cow can stay lying as long as it wants. A study by Nielsen et al. [19] demonstrated that the occurrence of undisturbed cycles coincides with the synchrony in the herd, implying

that most of the animals rest at the same time. Hence, a higher synchrony of the cows leads indirectly to a better growth rate, which itself leads to a better outcome for the farmer. Thus, the farmer's goal coincides with the cows' wish for an undisturbed resting period, which improves animal welfare.

We will then use these synchrony measures to help optimise the pen design. We composed an algorithm that optimises the positions and sizes of the bedding and feeding areas in order to find a configuration that favours synchrony in the multi-agent system. We implemented this algorithm and incorporated it into a graphical user interface (GUI). This GUI should give farmers the chance to analyse the effects of changes in barn layout based on simulation results. They can then use the algorithm to find a configuration with an optimal predicted outcome.

In the first chapter of this dissertation the agent-based model for cattle behaviour is presented. This model, though inspired by Couzin et al.'s model, is original in the way it introduces resting and eating periods into the agents' behavioural rules. The second chapter contains the evaluation of the model using comparison to data from farms. The third chapter begins with a short review of weakly coupled oscillators. We then present different ways to measure synchrony in the model and the results of a series of simulations. Both the second and the third chapters contain reviews of known concepts and their special adaptation for this model. The fourth chapter will then present an original algorithm that optimises the configuration of the pens with respect to the synchrony measures. Additionally, we will demonstrate the work of the algorithm on an example. The main results of this dissertation are summed up in the conclusion along with suggestions for future research. We will provide additional details on the GUI, on some known concepts that we used during the dissertation and on the data in the various appendices.

2 Model of cow behaviour

The modelling of discrete entities arises in many problems, such as traffic flow analysis and the modelling of disease spread [3]. In general, there are two possible ways to model these systems. If the number and density of the elements is reasonably high, one can concentrate on analysing the density flow in the considered area. Using this modelling approach, one typically obtains a set of partial differential equations. However, in a usual barn the number of cows and their density is too small to use such a continuum model. In such cases, one can instead apply a modelling framework based on agents. The development of such a model is presented in this chapter.

In the first section of this chapter we give a review of agent-based modelling. After a short introduction and an informal description, agents and their characteristics are formulated more rigorously. In the second section, we present the basic rules used for the behavioural cow model. This model generalises Couzin et al.'s model for animal movement in [6], which we will also present in this section. We conclude the chapter with a description of the implementation of the model along with the discussion of specific problems that had to be addressed when implementing it.

2.1 Agent-based modelling

During the first half of the 20th century, the Hungarian-American mathematician John von Neumann developed the first ideas of an intelligent agent, which eventually found its climax in the famous von-Neumann-machine, a theoretical automaton that was capable of self-reproduction. After von Neumann passed away in 1959, his work was published by Arthur W. Burks in 1966 in [18]. In the 1970s John Horton Conway developed the famous "Game of Life" [12]. This game showed that complex behaviour can emerge by combining a number of agents with very simple rules and restricted information. Since then the concept of individual agents with their own rules and a restricted communication between each other has been used in many scientific disciplines and gained considerable

importance with computational advances that have allowed increasingly large systems to be analysed. See references [23] and [29] for more details.

2.1.1 An informal definition

There is no standard definition of the term agent. Explanations are often based on the properties of an agent. The way we use the term throughout this dissertation is loosely based on the definition given in [29].

Definition 2.1. *An agent is a system that uses a fixed set of rules based on communication with other agents and information about the environment in order to change its internal state and fulfil its design objectives.*

All agents of the system are situated inside a finite environment and exchange information about their respective statuses. Theoretically it is also possible that an agent changes the environment in which it lives, but we will neglect this possibility here. We will also assume that an agent is time-discrete, which means it only changes its state at certain times $t_0 < t_1 < \ldots$. One can distinguish between *adaptive* and *non-adaptive* agents. *Adaptive* agents can change their rules throughout their life process and therefore learn certain behaviour. We will concentrate on *non-adaptive* agents, which follow a constant set of rules and thus cannot adapt their behaviour to environmental changes. We will denote the combination of a number of agents and the common environment as a *multi-agent system*.

2.1.2 A more rigorous definition

We now want to give a more rigorous explanation of an agent. We assume that our system consists of N agents that all follow the same basic rules with only slight differences in parameter values. The agents are numbered $1, \ldots, N$ and each of them has a vector of internal states $\mathbf{x}_i(t),\ i = 1, \ldots, N$. The internal states are assumed to be invisible to

other agents in the system, so we introduce a vector of external states $\mathbf{y}_i(t),\ i = 1, \ldots, N$. These external states depend solely on the internal states of the agent and are given by

$$\mathbf{y}_i(t_l) = g_i(\mathbf{x}_i(t_l)), \quad i = 1, \ldots, N, \quad l \geq 0. \tag{2.1}$$

The agents live in a mutual environment E, which is modelled as a set of boundary conditions for the internal states of an agent. We allow E to change with time, but as mentioned in Section 2.1.1 we assume that it cannot be changed by the agents. Because we assume the system to be time-discrete and the rules not to change with time, the general update function for the internal states is

$$\mathbf{x}_i(t_{l+1}) = f_i\left(\mathbf{x}_i(t_l), \mathbf{y}_1(t_l), \ldots, \mathbf{y}_{i-1}(t_l), \mathbf{y}_{i+1}(t_l), \ldots, \mathbf{y}_N(t_l); E\right),\ i = 1, \ldots, N,\ l \geq 0. \tag{2.2}$$

In equation (2.2), one can see that the environmental information E can also have an influence on the internal states of an agent and therefore on its behaviour. The system is started at time t_0 with the internal states $\mathbf{x}_1(t_0), \ldots, \mathbf{x}_N(t_0)$ and the external states $\mathbf{y}_i(t_0) = g_i(\mathbf{x}(t_0))$ for $i = 1, \ldots, N$. Without loss of generality, we can choose $t_0 = 0$, because f and g are both independent of time. In the general model, no restrictions need to be made for the functions f_i and g_i. A sketch of the behaviour of a single agent can be seen in Figure 1.

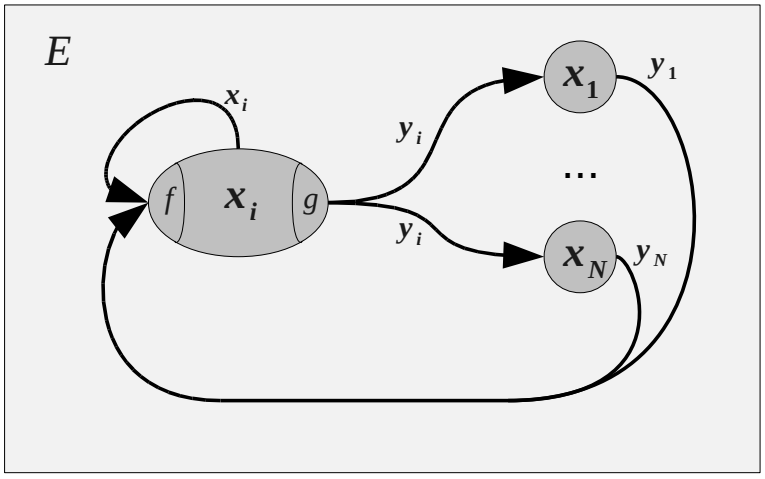


Figure 1: Behaviour of a single agent

2.2 Rules for the behavioural cow model

The next step is to develop a set of rules that govern the behaviour of cattle and model the most important aspects of a beef cow's life in the barn. The model should include resting and eating periods as well as movement of the cows. This led to the chosen set of internal states $\mathbf{x}_i$, which we show in Table 1. Throughout this dissertation we will use the words resting and lying interchangeably.

$\mathbf{r}_i \in B \subseteq \mathbb{R}^2$	position of the cow in the barn B
$d_i \in [0, 2\pi]$	angle between the x-axis and the facing direction of the cow
$s_i \in S$	status of the cow chosen from a set of possible statuses S, with $S = \{\textit{Standing, Walking, Resting, Eating}\}$
$a_i \in [0, 1]$	state of hunger ($a_i = 0$ - very hungry)
$b_i \in [0, 1]$	desire to lie down for a rest ($b_i = 1$ - no desire)

Table 1: Internal states for the agent-based cow model

In this model, the environmental information is given by the set $B \subseteq \mathbb{R}^2$, which represents the shape of the barn. Walls and barriers are implicitly described by the boundary ∂B. Additionally the sets $F \subseteq B$, representing the feeding, and $R \subseteq B$, representing the bedding area, are part of the environment and will have an influence on the agent's behaviour (see the discussion below). Both these sets can theoretically be changed in time, as the farmer could decide to adjust their positions or sizes. Therefore, we obtain for the environmental information $E = \{B, R, F\}$.

Out of the internal states shown in Table 1, only $\mathbf{r}_i, d_i$, and s_i can potentially be observed by other cows in the barn. These therefore constitute the set of external states $\mathbf{y}_i$. Hence, the output function g takes the following form

$$g : \mathbf{x}_i = (\mathbf{r}_i, d_i, s_i, a_i, b_i) \mapsto (\mathbf{r}_i, d_i, s_i) = \mathbf{y}_i. \tag{2.3}$$

In fact, we will see that the actual model only uses a subset of these states. The definitions of these internal and external states allows us to describe basic behavioural rules and formulate them in mathematical terms.

2.2.1 Avoidance

We will start this part with a short review of Couzin et al.'s model for animal movement [6]. Because this model was originally used to model fish schools and bird flocks, some adaptations are needed in order to make it suitable for cattle. A detailed review of the model and further simulations can also be found in [17].

In the original model, each agent has three different ways to react to other agents depending on the distance between them. Hence, Couzin et al. differentiated between 3 different zones around each agent, as can be seen in Figure 2.

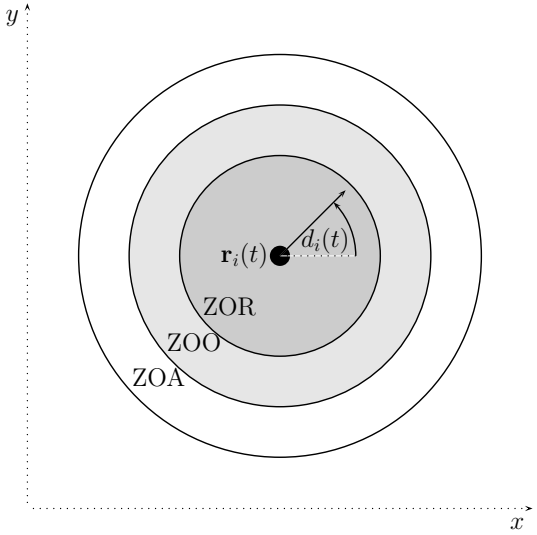


Figure 2: 3 Zones around an agent in Couzin et al.'s model

The innermost zone is called the "Zone of Repulsion" (ZOR), which the agent tries to keep as its private space. We define the number of other agents in this zone to be n_r, and these agents to be numbered $j_1, \ldots, j_{n_r}$. In order to keep this zone for his own, the agent will try to walk away from every intruder entering it. Therefore, if $n_r > 0$ the movement is governed by an avoidance rule, with the desired direction defined through

$$\mathbf{r}_{r,i} := -\sum_{k=1}^{n_r} \frac{\mathbf{r}_{j_k}(t_l) - \mathbf{r}_i(t_l)}{\|\mathbf{r}_{j_k}(t_l) - \mathbf{r}_i(t_l)\|}. \tag{2.4}$$

The direction $\mathbf{r}_{r,i}$ is the sum of the unit vectors pointing away from the intruders. The second zone is called "Zone of Orientation" (ZOO). The agent tries to adjust its current facing direction in order to align with other agents in its ZOO. The desired direction of

movement for the orientation zone is thus defined to be

$$\mathbf{r}_{o,i} := \sum_{k=1}^{n_o} \begin{pmatrix} \cos d_{o_k}(t_l) \\ \sin d_{o_k}(t_l) \end{pmatrix}, \tag{2.5}$$

where the n_o other agents in the ZOO are numbered $o_1, \ldots, o_{n_o}$. The agent is attracted to all agents in the third zone, the so called "Zone of Attraction" (ZOA). Again, we define the number of other agents in the ZOA to be n_a and assume these are numbered $c_1, \ldots, c_{n_a}$. The desired direction defined through agents in the ZOA is

$$\mathbf{r}_{a,i} := \sum_{k=1}^{n_a} \frac{\mathbf{r}_{c_k}(t_l) - \mathbf{r}_i(t_l)}{\|\mathbf{r}_{c_k}(t_l) - \mathbf{r}_i(t_l)\|}. \tag{2.6}$$

We can see that equation (2.6) is similar to equation (2.4) except for the different sign that makes the agent move towards other agents rather than away from them. As mentioned earlier, the total desired direction $\mathbf{r}_{d,i}$ is defined to be $\mathbf{r}_{r,i}$ if $n_r > 0$, which means that the avoidance rule has priority over the other two. If n_r vanishes and both n_o and n_a are non-zero, the desired direction is defined to be

$$\mathbf{r}_{d,i} := \frac{1}{2} \left(\mathbf{r}_{o,i} + \mathbf{r}_{a,i} \right). \tag{2.7}$$

If $n_r = 0$ and only one of n_a or n_o is non-zero, the rule belonging to the non-zero value governs the movement fully. In the case where $n_o = n_a = n_r = 0$, Couzin et al. define the animal to keep moving in its current direction.

In order to address the restricted physical properties of cows compared to fish and birds, we only keep the avoidance rule defined in the ZOR for the cow model. If there are no other agents in the ZOR, the agent does not have the desire to avoid anything and stops walking. Exceptions for this are hunger or the desire to rest, which will be discussed in Section 2.2.2.

If $n_r > 0$, we can calculate the desired direction of movement using equation (2.4) and define the angle between $r_{d,i}$ and the x-axis to be $d_{d,i}$. Another feature inherent in Couzin et al.'s model is the finite turning speed of the agents, because animals cannot turn infinitely fast. Hence, we define the turning speed γ_i measured in rad/s. In order to calculate the facing direction at the next time step, the angle $\Delta d \in [-\pi, \pi]$ is defined

to be the difference between $d_{d,i}$ and $d_i(t_l)$. The new facing direction can be calculated using

$$d_i(t_{l+1}) = \begin{cases} d_{des,i}, & \Delta d \leq \gamma_i(t_{l+1} - t_l), \\ d_i + \Delta d\gamma_i(t_{l+1} - t_l), & \Delta d > \gamma_i(t_{l+1} - t_l). \end{cases} \tag{2.8}$$

The actual movement of the agents occurs before the turning and proceeds in the current facing direction. Therefore, the new position is

$$\mathbf{r}_i(t_{l+1}) = \mathbf{r}_i(t_l) + \begin{pmatrix} \cos d_i(t_l) \\ \sin d_i(t_l) \end{pmatrix} v_i(t_{l+1} - t_l), \tag{2.9}$$

where v_i is the speed of the agent. In the implementation, we will need to make some restrictions to this movement in order to ensure that animals cannot run through each other or through walls (see Section 2.3 for details). Also, we will see in Section 2.2.3 that moving and turning only occurs if the agent is in a certain state $s_i(t_l)$.

2.2.2 Hunger and Resting

As mentioned in Chapter 1, cows have a unique double-digestive system, which requires them to eat and rest alternately [16]. In order to incorporate this feature into the model, we introduced the internal states $a_i(t)$ and $b_i(t)$. For both of these states, the value of 0 means the desire to eat or rest is the highest and 1 means no desire at all. For simplicity, we assume these variables to change linearly in time. This assumption is reasonable, because, as we will see in Section 2.2.3, the agent only changes its behaviour when one of these variables becomes 0 or 1. The progress between these values only has a minor effect. We also assume the change of a_i and b_i to depend on the state the agent is currently in, because for example the consumption of energy could differ between a resting and a standing cow. This yields

$$a_i(t_{l+1}) = \begin{cases} \min\left(1, a_i(t_l) + \frac{t_{l+1}-t_l}{T_{eat}}\right), & s_i(t_l) = \textit{Eating}, \\ \max\left(0, a_i(t_l) - \frac{t_{l+1}-t_l}{T_{h,s}}\right), & s_i(t_l) = \textit{Standing, Walking}, \\ \max\left(0, a_i(t_l) - \frac{t_{l+1}-t_l}{T_{h,r}}\right), & s_i(t_l) = \textit{Resting}, \end{cases} \tag{2.10}$$

$$b_i(t_{l+1}) = \begin{cases} \min\left(1, b_i(t_l) + \frac{t_{l+1}-t_l}{T_{rest}}\right), & s_i(t_l) = \textit{Resting}, \\ \max\left(0, b_i(t_l) - \frac{t_{l+1}-t_l}{T_{stand}}\right), & s_i(t_l) = \textit{Standing, Walking, Eating}. \end{cases} \tag{2.11}$$

We will investigate the time constants $T_{eat}, T_{h,s}, T_{h,r}, T_{rest}$, and T_{stand} in more detail during the process of model validation in Chapter 3.

2.2.3 Finite-state machine for s_i

Switching between the 4 possible states *Standing*, *Walking*, *Resting*, and *Eating*, and the reaction to hunger and the desire to rest are controlled by a *finite-state machine*. A *finite-state machine* is a model consisting of a finite number of states and the conditions for state changes, often referred to as *transitions*. A detailed introduction can be found in [13]. The transitions for our model can be seen in Figure 3.

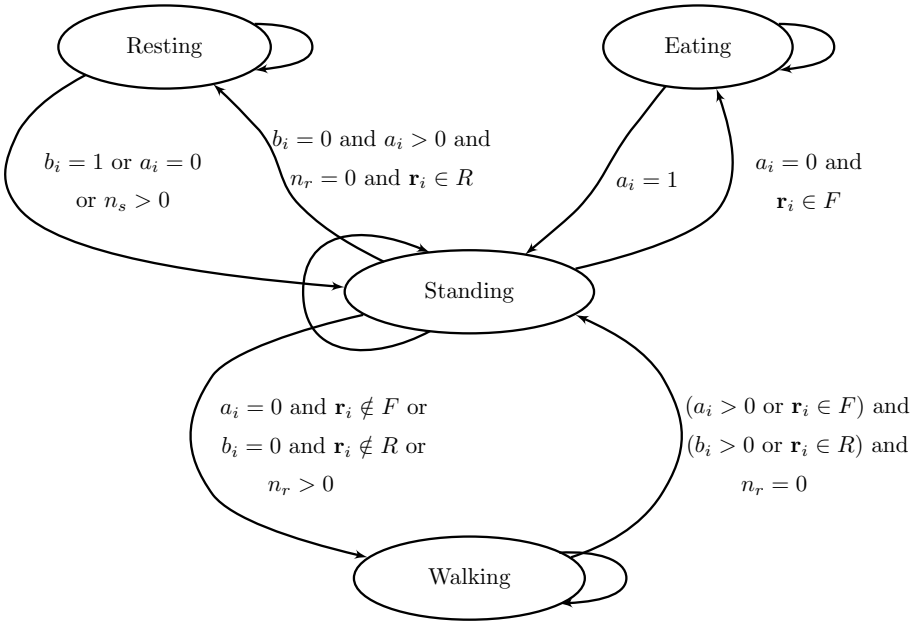


Figure 3: Finite state machine

The starting point in this state machine is $s_i = \textit{Standing}$. If the agent gets hungry or wants to rest, it will change to $s_i = \textit{Moving}$ in order to walk to the feeding or bedding area. If it is hungry and enters the feeding area, it will change to $s_i = \textit{Eating}$ and stay in this state until it is full and the hunger variable a_i reaches the value 1. This is due to the assumption that an eating agent does not give way to other agents. These kind of social interactions in cows are driven by dominance relationships between the individuals, which are explained in detail in [2]. For this model we simplify these relationships by assuming that all cows are equal.

For resting, the matter is a bit different. In order to handle the resting case, we introduce another zone around the agent, the so called "Sleeping Zone" (SZ). As soon as the agent is disturbed by another agent in its SZ, it gets up, which is another simplified assumption of the complex social interactions between cows. Therefore we introduce n_s, the number of other agents in the sleeping zone, similarly to n_r used earlier. Also, the nature of the double-digestion required that eating occurs before resting [16]. This leads to the assumption that an agent gets up from resting when it is hungry even if $b_i < 1$.

The second mechanism for an agent to enter the state $s_i = \mathit{Moving}$ is if another agent enters its ZOR. In this case, the movement is determined by the avoidance rule and the agent tries to move away from its contender. Some special cases of conflicting interests will be considered in Section 2.3 in order to ensure that the actual simulation does not get caught in an infinite loop.

2.3 Implementation details

For the implementation of the present model, we used the C++ programming language. For reasons of simplicity, the time stepping was chosen to be equidistant with a step size of $\tau > 0$. The parameter τ should be chosen in a way that the distance covered in a single simulation step is significantly smaller than the size of an agent's ZOR. Hence, we get for the time-steps

$$t_l = l \cdot \tau, \quad \text{and} \quad t_{l+1} - t_l = \tau, \quad \forall\, l = 0, 1, 2, \ldots, L, \tag{2.12}$$

where L is the number simulation steps. In the implementation of the present model, one has to consider some special situations that can arise during a simulation. First, one has to ensure that the agents cannot walk through walls or into each other. Therefore, we have to install an additional test before the actual movement occurs. In this test, the movement step determined by (2.9) is precalculated and the new position is tested for collisions with walls or other agents. If no collision occurred, the new position is confirmed. Otherwise, the step does not take place and the agent is set back to its original position. In order to test the overlap with other obstacles, a shape has to be assigned to the model cows.

For simplicity, we used disks with a constant radius r_{ag}. The initialisation at time $t_0 = 0$ is done by an assignment of random values to the internal states of every agent. It is possible that agents overlap initially. In this case, the avoidance rule makes them move away from each other immediately, so that the collision is eliminated.

A second effect we need to address is the possibility of conflicting interests in an agent. As mentioned earlier, we resolved the conflict between the desire for food and for a rest in favour of eating. One must also consider how to resolve the case when an agent is on its way to move towards the feeding or bedding area and is disturbed by another agent in its ZOR. In this case, the new desired direction is chosen to be the vectorial average direction between the two original desired directions. This allows the agent to simultaneously move towards its target and avoid other agents.

Putting all the rules and special considerations together, we can now obtain a pseudo-code version of the complete cow model, which we show in Algorithm 1. Here, the locations of feeding areas F, bedding areas R and walls defined through B are all part of the environmental information E as mentioned in Section 2.2. In the implementation we assumed these sets to be general polygons stored in their respective pointwise representation. When agents get hungry or tired, a random empty spot in the respective area F or R is calculated. Unless the agent is disturbed by other agents, it moves towards this point in a straight line. In order to get visual results and to give farmers a chance to experiment with the model, we implemented a graphical user interface that is described in Appendix A.

Algorithm 1: One update step of an agent

Input: $\mathbf{x}_i(t_l), \quad \mathbf{y}_j(t_l), j \in \{1, \ldots, N\} \backslash \{i\}, \quad E = \{B, F, R\}$

Output: $\mathbf{x}_i(t_{l+1})$

Calculate n_r and n_s;

Calculate $a_i(t_{l+1})$ and $b_i(t_{l+1})$ using equations (2.10) and (2.11) respectively;

$\mathbf{r}_i(t_{l+1}) := \mathbf{r}_i(t_l); \quad d_i(t_{l+1}) := d_i(t_l); \quad s_i(t_{l+1}) = s_i(t_l);$

if $s_i(t_l) ==$ *Resting and* $(n_s > 0 \quad$ *or* $a_i(t_{l+1}) = 0$ *or* $b_i(t_{l+1}) = 1)$ **then**

  $s_i(t_{l+1}) :=$ *Standing*;

end

if $s_i(t_l) ==$ *Eating and* $a_i(t_{l+1}) == 1$ **then** $s_i(t_{l+1}) :=$ *Standing*;

if $s_i(t_l) ==$ *Standing* **then**

  if $a_i(t_{l+1}) = 0$ *and* $\boldsymbol{r}_i(t_{l+1}) \in F$ **then** $s_i(t_{l+1}) :=$ *Eating*;

  else if *(*$a_i(t_{l+1}) = 0$ *and* $\boldsymbol{r}_i(t_{l+1}) \notin F$*) or (*$b_i(t_{l+1}) = 0$ *and* $\boldsymbol{r}_i(t_{l+1}) \notin R$*) or* $n_r > 0$ **then**

    $s_i(t_{l+1}) :=$ *Walking*;

  else if $b_i(t_{l+1}) = 0$ *and* $\boldsymbol{r}_i(t_{l+1}) \in R$ **then** $s_i(t_{l+1}) :=$ *Resting*;

else if $s_i(t_l) ==$ *Walking* **then**

  Calculate $\mathbf{r}_i(t_{l+1})$ using equation (2.9) and test for overlap with other agents or walls;

  Calculate $\mathbf{r}_{d,i}$ using equation (2.4) and merge with directions to feeding / bedding areas;

  Calculate $d_i(t_{l+1})$ using equation (2.8) with $d_{d,i}$ as described above;

  if $n_r == 0$ *and* $(a_i(t_{l+1}) > 0$ *or* $\boldsymbol{r}_i(t_{l+1}) \in F)$ *and* $(b_i(t_{l+1}) > 0$ *or* $\boldsymbol{r}_i(t_{l+1}) \in R)$ **then**

    $s_i(t_{l+1}) :=$ *Standing*;

  end

end

3 Model validation

Modelling is often used to gain insight into a process that is difficult or expensive to track. Simulation of the model is then used to make predictions and draw conclusions about the real process. In order to allow the last step, one has to ensure that the model resembles the behaviour of the original system to the extent possible. This step of the modelling process is called model evaluation and is split into the two parts validation and verification. In the latter, one checks if the model is implemented correctly. The process of model validation is the actual check if the modelling assumptions allow for a model behaviour that is sufficiently close to the analysed system. Additionally, this step can be used to help determine the values of modelling parameters. For the model of cow behaviour, we used empirical data to fit the parameters. The data was gathered from video footage and is explained in more detail in the first section of this chapter. The second section gives a summary of the parameters used in the model and the variables that will be fit. In the third section, we will give a short presentation of the simulated annealing optimisation concept along with the adaptations we used for the cow model. The results of the validation process will be explained in the fourth section.

3.1 Validation data

The data we used for the model evaluation is originated from video data that was recorded between 2004 and 2006 by Thomas Alan Shaw. He assembled video footage from 9 different farms including 60 separate groups of cattle. The visited farms are typical examples for rearing beef cattle during the winter months, found across England. In order to obtain time series data from these videos, image processing was done in combination with marking of events and positions. Most of this marking was done by Nicole Milligan in 2006. Three different streams of data were obtained from the videos: Lying data, Housing data, and Standing data. All of them are explained in more detail below. During the analysis of the video data, we spotted some problems in the prior work. We will also present these problems in detail and discuss possible consequences for the validation

process. More details and sample data can be found in Appendix E.

3.1.1 Lying data

The first data stream contains the so-called "Lying data" that consists of a set of lying events (see Appendix E.1 for sample data). A lying event is the action of a single cow lying down at a time t_1 and getting up at a later time t_2. For each of these events, the times t_1 and t_2 are recorded together with the position of the cow performing the event. Additionally, the positions of all visible standing cows at the times t_1 and t_2 are tracked. For all of these cows, two positional values – one for the head and one for the tail – are stored, which gives the additional information of the facing direction for each cow.

One possible problem for this data stream is that cows could be lying down at either the beginning or the end of the video. In these cases, the time and the positions of the standing cows can only be recorded for the visible part of the lying event. The other time value was set to a special error code to mark this lying event as incomplete.

3.1.2 Housing data

For most of the pens, the cameras could not capture the whole pen – a problem that we will address later. In order to store information about the visible parts of the pens, a second stream of data was generated – "Housing data". We show sample data and further details in Appendix E.2.

The first data stored in these data files is the boundary of the visible area of the pen, which is followed by the positions of bedded areas. After that, all visible barriers on the boundary of the pen or sometimes even extending into the interior are saved. The last two sets represent the positions of food and water troughs respectively. All those areas are represented by a set of points, which set up the polygon approximating the areas. Two-point-polygons are used for simple one dimensional barriers. Except for the

representation of the visible area itself, all other parts can theoretically be missing because of the restricted view.

3.1.3 Standing data

In order to allow an analysis of the movement of each of the cows, a third stream of Standing data was recorded – "Standing data" (see Appendix E.3 for more details). In these files, the positions of all visible standing cows was recorded every 30 seconds. Again, both the head and the tail coordinates are stored. As mentioned earlier, the view of the camera is usually restricted, which makes it possible that cows enter and leave the visible area. This combined with the nearly indistinguishable appearances of the cows makes it impossible to keep track of the position of a single cow throughout the whole video, which is another problem to be discussed later.

3.1.4 Units

Because the data was collected from different barns and shows cattle herds of different breeds and ages, comparability of the data is a major issue. The first problem following from this is the need for a consistent set of units for all of the videos. The initial lengthscale in the barns was provided by a tape measure attached to one of the visible corners. Having a known length, these measures can be tracked on the videos and used to map the positions on the video into two dimensional coordinates in the barns.

This solution still does not ensure the comparability of the different data sets. This problem was addressed by measuring the average length of a standing cow for each of the videos. This average length was defined to be the lengthscale and all coordinates were scaled accordingly. We will denote this lengthscale as 1 *cl* (cowlength) throughout the rest of the dissertation. This process implies that every video has an average length of standing cows of 1, no matter how old the filmed cows were. In order to ensure for a comparable time scale the time unit was chosen to be one second on the video. Because every video

with the length of 180 minutes contains 24 hours of footage, a unit time represents 8 s of real time and is equal for all of the different barns. In all the following explanations the unit 1 s stands for a second on the videos, unless explicitly stated otherwise.

3.1.5 Problems with the data

As mentioned earlier, some problems lead to the data being incomplete in certain aspects. One problem is that during darkness it was impossible to mark the cows. Hence a reasonable amount of every video is missing due to nighttime, which makes an investigation of lying behaviour during the night impossible. Thus, we had to concentrate on daytime resting patterns.

A second incompleteness occurs due to the restricted view of the cameras. Cows can leave and enter the visible area and it becomes impossible to keep track of the movement of individual cows. Because we are more interested in the behaviour of the whole herd of cattle rather than individual cows, this is not too big a problem as long as the visible section is representative for the whole pen. However, this effect can become a major problem for pens where only small parts of bedding or feeding areas are visible or when they are even missing completely.

3.2 Parameter fitting

During the explanation of the model in Section 2.2, we introduced a set of parameters. These values have a significant impact on the behaviour of the system. Hence, we want to adjust these parameters so they fit best with the data described above. First we will describe the variables used for the fitting of the parameters. We will then give a short review and further discussion of the parameters themselves.

3.2.1 Variables compared for fitting

Two different measures will be used to adjust the parameters. The first one concerns the distribution of lying time lengths. Using the Lying data discussed earlier, one can obtain the length of every complete lying event recorded and construct a distribution of the lying times. In Figure 4, the distribution obtained from all the video data and used for the parameter fitting can be seen. In order to adjust the parameters, we will later use the least square distance between the simulated value and the curve shown in Figure 4 as an objective function that needs to be minimised. Therefore we use the distributional data points from Figure 4 to calculate the error value using

$$e_1 = \sum_{i=1}^{m} (s_{i,data} - s_{i,sim})^2, \tag{3.1}$$

where m is the number of data points, $s_{i,data}$ are the individual data points, and $s_{i,sim}$ are the respective data points obtained from the simulation.

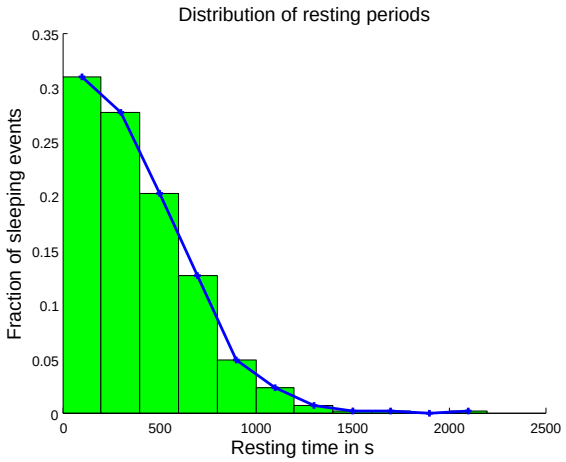


Figure 4: Distribution of lying time lengths

Because the distribution of the lying times does not fully represent the lying behaviour, we need a variable that represents the percentage of time that a cow spends resting. The incompleteness of the data discussed above does not allow us to extract this information directly from the videos. Because this value can vary for cows of different breeds and age, we have to make a unifying choice. Based on the information gathered in [8], we assume the amount of time a cow spends resting to be 50 %, which will be the value we use during the parameter fitting. Hence, the second error component satisfies the equation

$$e_2 = (p_{rest,sim} - 0.5)^2, \tag{3.2}$$

where $p_{rest,sim}$ is the probability that an agent is resting in the simulation results. We can now add these two error values together to acquire the total error value e that is equivalent to the objective function we seek to minimise. Hence, e is given by

$$e = e_1 + e_2. \tag{3.3}$$

3.2.2 Adjustable parameters

A summary of the parameters that appear in the model can be seen in Table 2.

T_{eat}	eating time	$\sim 100\ s$
T_{rest}	maximum resting time	$\sim 2000\ s$
$T_{h,s}$	time until agent gets hungry when standing	$\sim 2000\ s$
$T_{h,r}$	time until agent gets hungry when resting	$\sim 2500\ s$
T_{stand}	time between two consecutive rests	$\sim 2000\ s$
r_{zor}	size of retention zone	$\sim 1.5\ cl$
r_{sz}	size of sleeping zone	$\sim 1.2\ cl$
r_{ag}	size of an agent	$= 0.5\ cl$
v	movement speed	$\sim 1\ cl/s$
γ	turning speed	$\sim 2\pi\ rad/s$

Table 2: Parameters of agent-based model

Out of these parameters, the last 3 are physical attributes of the cow. These can be directly incorporated from general information known about cows and are therefore not used for the parameter fitting. In particular the size of an agent r_{ag} can be estimated to be 0.5 cl, because we defined the lengthscale 1 cl to be the size (diameter) of a cow. The parameters v and γ only have an impact on the moving behaviour of the agents, but neither on eating nor resting patterns. Hence, we will use values for these two parameters that represent the moving and turning speed of the cows we saw on the videos – we assume $v = 1\ cl/s$ and $\gamma = 2\pi\ rad/s$.

We can obtain an initial guess for the maximum resting time T_{rest} from the maximum lying time measured in the data. Using the distribution of resting times described earlier,

we get $T_{rest} = 2000\ s$ as an initial guess. Additionally, the ratio $T_{rest}/(T_{rest} + T_{stand})$ ideally represents the fraction of the day a cow spends resting, which we defined to be 0.5. This gives rise to an initial guess of $T_{stand} = 2000\ s$.

Because of the problems in the data described in Section 3.1.5, no consistent information about the length of eating periods is available. We can, however, assume the corresponding parameter T_{eat} to be much smaller than all the other time constants. For the validation process, an arbitrary value of $T_{eat} = 100\ s$ was chosen, because eating is usually a shorter process than resting. Higher values of T_{eat} can lead to additional queueing at food troughs, but should not influence the resting behaviour.

The values for r_{zor} and r_{sz} can be obtained from the minimum distances of a resting cow to a standing cow at the times t_1 (lying down) or t_2 (standing up), respectively. Because the value of r_{zor} does not directly influence the lying pattern of the agents, we assume it to be 1.5 cl, which is consistent with the data. However, the value of r_{sz} has an impact on the resting behaviour, as it predicts how easily a lying agent gets disturbed. To get initial guesses for the missing two parameters, we assume a resting agent to use up less energy than a standing one (hence, we assume $T_{h,r} > T_{h,s}$).

3.2.3 Problems of parameter fitting

Our goal in the parameter fitting process is to minimise an objective function that is a combination of the error variables described in Section 3.2.1. The optimisation is performed with respect to the parameter space shown in Section 3.2.2. This inherits a set of problems. First, the number of adjustable parameters is relatively big compared to the small set of available data, which could cause overfitting in the parameters – the system represents the data rather than the underlying behaviour. Additionally, the search space is in fact infinitely big, which makes an exhaustive search impossible.

Another problem is that each evaluation of the objective function requires a simulation of a large number of steps ($L \geq 100000$) of the whole system in order to gain representative

distributions. For a barn with 10 cows the simulation of 100000 steps takes about 5 s on a computer with a 4 GHz processor and 1GB RAM. Hence, we desire an optimisation algorithm that minimises the number of function evaluations. Additionally, there is no chance to obtain gradient information for the objective function in this system.

The third major problem is that the behaviour of the system can depend on different initial settings of the agents, which implies that different random seeds can produce different values of the objective function. A process with this property is generally called "noisy". This problem can be addressed by the use of ensemble averages with the disadvantage of even longer computation times. Hence, the optimisation algorithm needed for this parameter fitting problem should be direct (no derivatives), stable with respect to noise, and minimal in the number of function evaluations. One technique satisfying these requirements is simulated annealing.

3.3 Simulated annealing

The method of simulated annealing was first formulated by Kirkpatrick et al. in [15] and later independently redeveloped and used as an approximation algorithm for the travelling salesman problem by Černý in [4]. This direct optimisation algorithm is inspired by the annealing process used in metallurgy. In order to achieve a crystalline structure with a minimum number of defects, the metal is first heated over its melting point and later cooled very slowly to give the atoms time to settle into the minimum energy state, which is the crystal lattice. If one considers all possible states of the metal's atoms as the search space and the energy as the objective function, this annealing process can be interpreted as the search for a global minimum. This gave rise to the adaptation of a simulated version for general objective functions with large search spaces.

Let $\mathcal{G}$ be the search space, which is assumed to be too large for an exhaustive search, and $h : \mathcal{G} \to \mathbb{R}$ the objective function we seek to minimise. We assign a *transition probability*

$P(\xi \to \eta) \in [0,1]$ to every pair of states $\xi, \eta \in \mathcal{G}$, which satisfies the property

$$\sum_{\eta \in \mathcal{G}} P(\xi \to \eta) = 1 \quad \forall \xi \in \mathcal{G}. \tag{3.4}$$

If the search space $\mathcal{G}$ is continuous as it is for our model, the function P shall be interpreted as a probability density function and the sum becomes the integral

$$\int_{\mathcal{G}} P(\xi \to \eta) d\eta = 1 \quad \forall \xi \in \mathcal{G}. \tag{3.5}$$

We can also define the set of possible successors of a state $\xi \in \mathcal{G}$, often referred to as its *neighbourhood* $\mathcal{N}(\xi)$, as the subset of $\mathcal{G}$ with nonzero transition probability $P(\xi \to \eta)$. The simulated annealing algorithm starts with an initial state $\xi_0 \in \mathcal{G}$. In every step the successor state η is chosen randomly from the probability distribution defined through $P(\xi \to \eta)$. The objective function value is then calculated at the potential new state and $h(\eta)$ is compared to $h(\xi)$. If $h(\eta)$ is smaller than $h(\xi)$, η becomes the new state, which is equivalent to the classic random search algorithms. However, unlike the standard algorithms, simulated annealing still allows η to be accepted even if $h(\eta)$ is bigger than $h(\xi)$. The latter happens with the so-called "acceptance probability" $Q(\xi, \eta, T)$, which is governed by an artificial temperature value T described later and defined similarly to the Boltzmann probability as

$$Q(\xi, \eta, T) := \exp\left(-\frac{h(\eta) - h(\xi)}{kT}\right), \tag{3.6}$$

where $k \in \mathbb{R}$ is the counterpart of the Boltzmann constant k_B in the physical system and is chosen to be proportional to the dimension of h. The possibility of jumping to a state with a bigger objective function value allows the algorithm to leave local minima in order to find the global minimum. The question arising from this is how to change the temperature T during the optimisation process. This choice is called the *cooling schedule* and some possible approaches are shown in [27]. In one of those cooling schedules, the temperature is kept constant for a specified number of optimisation steps (denoted L_T), before it is updated using the formula

$$T_{k+1} = \lambda T_k, \quad \alpha \in [0,1]. \tag{3.7}$$

The constant λ is usually chosen to be between 0.6 and 1 in order to keep the cooling process slow and allow the states to settle to the global minimum. The process of cooling

is repeated until either $h(x)$ reaches a satisfactorily small value h_{min}, the state does not change anymore, or T becomes smaller than a predefined value of T_{min}. All these aspects are summarised in the pseudo-code seen in Algorithm 2.

Algorithm 2: Simulated annealing

Input: Initial state $\xi_0 \in \mathcal{G}, \quad \lambda, k \in \mathbb{R}$

Output: Final state $\xi_{fin} \in \mathcal{G}$ with objective function value $h(\xi_{fin})$

$\xi_{curr} := \xi_0$;
$T := 1$;
while $h(x) > h_{min}$ *and* $T > T_{min}$ *and* ξ_{curr} *changes* **do**
    for $i := 1$ *to* L_T **do**
        Choose ξ_{pot} from $\mathcal{N}(\xi_{curr})$ with respect to probability density function $P(\xi_{curr} \to \xi_{pot})$;
        if $f(\xi_{pot}) < f(\xi_{curr})$ *or* rand $< \exp\left(-\frac{f(\xi_{pot})-f(\xi_{curr})}{kT}\right)$ **then**
            $\xi_{curr} := \xi_{pot}$ `/* State` ξ_{pot} `is accepted */`
        end
    end
    $T := \lambda T$
end
$\xi_{fin} := \xi_{curr}$;

Simulated annealing is mainly used in combinatorical optimisation problems, where the search space $\mathcal{G}$ is usually finite but big and the neighbourhoods $\mathcal{N}(\xi)$ are comparably small. One example is the travelling salesman problem and the use of simulated annealing for this is shown in [4]. For these special problems, it can be shown that the algorithm converges to the global minimum for an adequate cooling schedule and a proof along with further discussion of different cooling schedules can be found in [1]. However, the problem we are facing here does not provide a finite set $\mathcal{G}$ and hence special adaptations for a continuous search space have to be made. In this case, we have $\mathcal{G} = \mathbb{R}^n$, where n is the number of parameters used for the fitting. The objective function for this problem can be calculated according to the curve fitting described earlier. For the transition probability

we use a multidimensional normal distribution in the form

$$P(\xi \to \eta) := \prod_{i=1}^{n} \exp\left(-\frac{(\xi_i - \eta_i)^2}{2\sigma_i^2}\right), \quad \forall\, \xi, \eta \in \mathcal{G} = \mathbb{R}^n. \tag{3.8}$$

This means that parameter values are changed independently using a normal distribution with standard deviation σ_i and mean 0. The choice for $P(\xi \to \eta)$ can be interpreted as a special case of the more general idea presented in [28].

3.4 Simulation and results

As mentioned earlier, the validation data was obtained from a number of different barns. All pens had a comparable but different layout with one food trough and a relatively large bedding area. For the validation process, we use a simple barn configuration that featured these essential characteristics. A picture of the barn used in all following simulations, together with the position of the agents, can be seen in Figure 5. We can see that the barn has a big bedding area (grey) and a relatively small feeding trough (green). We discuss the graphical user interface (GUI) we developed in Appendix A.

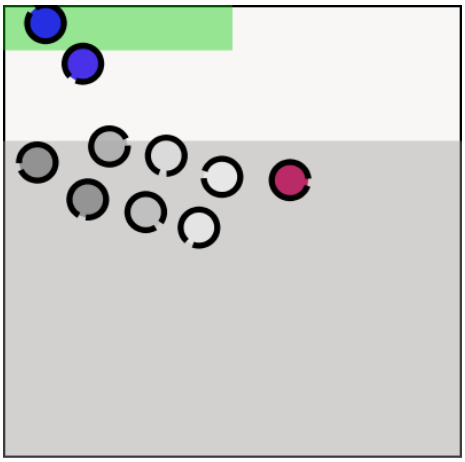


Figure 5: Barn used for validation (grey area - bedding, green area - feeding, grey agents - resting, hunger represented by colour gradient from red - hungry to blue - not hungry)

Following the discussion about the parameters in Section 3.2.2, we only used 5 of them for the actual simulated annealing process. The other 5 were given the values in Table 3. The parameters used for the curve fitting, together with their initial values and standard

deviations can be seen in Table 4. A justification for the fixed values and the initial guesses was given in Section 3.2.2.

Name	T_{eat}	r_{zor}	r_{ag}	v	γ
Value	100 s	1.5 cl	0.5 cl	1 cl/s	5 rad/s

Table 3: Fixed validation parameters

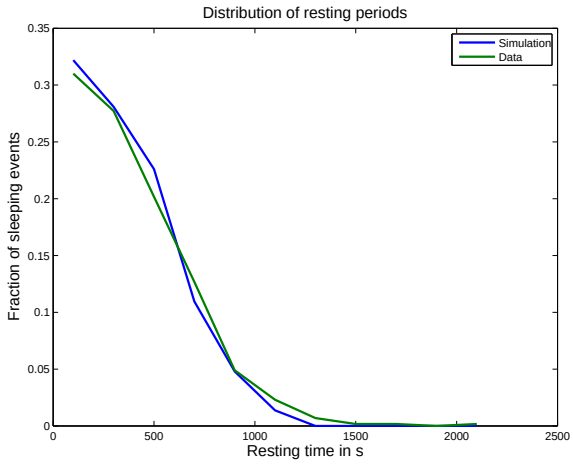
Name	T_{rest}	T_{stand}	$T_{h,s}$	$T_{h,r}$	r_{zs}
Initial value	2000 s	2000 s	2000 s	2500 s	1.25 cl
Stand. dev	200 s	200 s	200 s	200 s	0.0125 cl

Table 4: Variable validation parameters

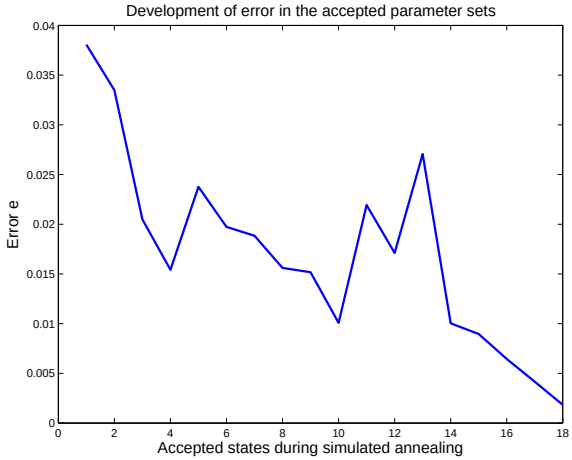
In all simulations, we chose $\tau = 0.1$ in order to satisfy the condition $\tau v \ll r_{zor}$, which we mentioned in Section 2.3. We used $100,000$ simulation steps so that every simulation is worth $10,000$ s of video time, which is equivalent to about 22 hours – nearly a full day. The number of cows in the unit barn was chosen to be 10, which is a representative value of the numbers of cows in the video pens.

3.4.1 Results of a single curve fitting

The best curve fitting obtained during the validation process can be seen in Figure 6(a). The error value belonging to this fit – calculated by equations (3.1) - (3.3) – is $e = 0.0018$. The parameter values for this special curve can be found in Table 5. Figure 6(b) shows the development of the fitting error during the simulated annealing process. One can see that the error generally tends to decrease, but the character of simulated annealing also allows for an increase of the error. The average percentage of a cow resting in the simulation with the parameter values given in Table 5 is 52.6%. Hence, we see that the parameter values found by the simulated annealing allow the model to resemble the lying behaviour represented in the data.



(a) Curve fit



(b) Error development during simulated annealing

Figure 6: Results of a single parameter fitting process using simulated annealing

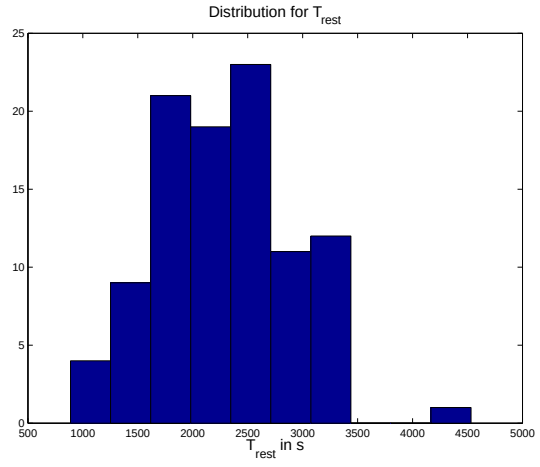
Name	T_{rest}	T_{stand}	$T_{h,s}$	$T_{h,r}$	r_{sz}
Resulting value	2824.5 s	1885.1 s	2309.9 s	1168 s	1.178 cl

Table 5: Validation parameters for best curve fitting

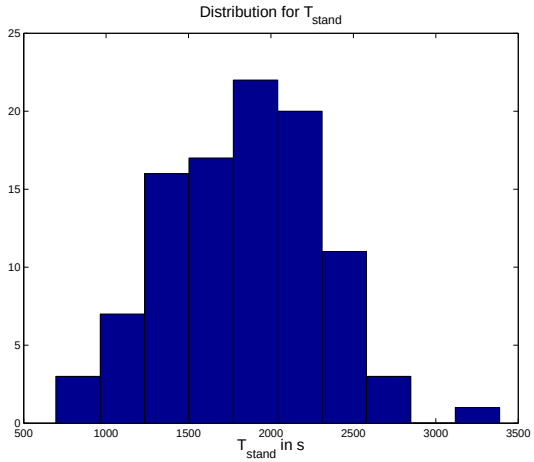
3.4.2 Collective results of a number of curve fittings

In order to show the consistency of the parameter fitting process, we repeated it 100 times, which took a total calculation time of about 8 days. The distributions of the resulting parameter values along with the errors can be seen in Figure 7. Most of these distributions feature peaks that represent a high number of annealing processes converging to similar values. These peaks indicate regions where the model behaviour resembles the data reasonably well. However, all of the distributions also have a relatively wide range of possible values. This effect is due to the noise in the minimisation process, which makes the convergence to a singular global minimum impossible. The error obtained in all cases is small (see Figure 7(f)), which indicates that very different parameter sets can lead to good curve fits. This suggests that the data used for the parameter fit is not sufficient to ensure that the model behaviour resembles the behaviour of the real system. For a further investigation of this model, additional data needs to be gathered in order to perform a cross-validation process. In a process like that, one would use the data obtained from a subset of barns for the parameter fitting. Thereafter, the model would be used to predict the outcome for another set of barns. These predictions should then be compared to the data belonging to these barns to evaluate the validity of the model. One prerequisite for conducting this type of validation is the need for exact information about barn layouts and cow numbers – data that is currently not available. Another possibility is to use a set of different objective functions. One would then fit the parameters to minimise one of these objective functions and could analyse the behaviour of the other objective functions.

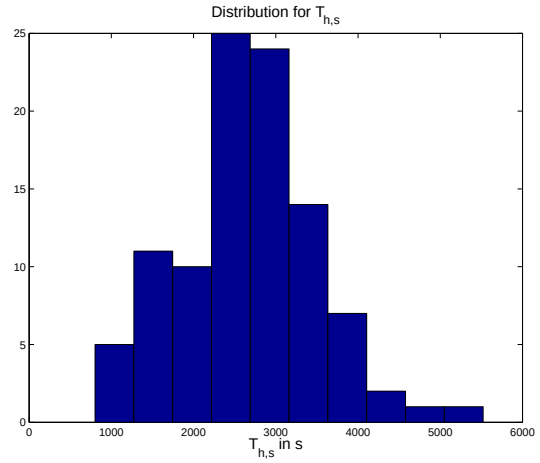
In Appendix D we show additional plots obtained from these set of validation processes. In Figure 16(a) we can see that T_{rest}/T_{stand} has a peak value at about 1.25, which is slightly higher than the value of 1 we expected. In Section 3.2.2, we anticipated the value for $T_{h,s}$ to be smaller than the value of $T_{h,r}$. However, Figure 16(b) shows that in most cases the opposite is true ($T_{h,s} > T_{h,r}$). Figures 16(c) and 16(d) show the distributions of the error components e_1 and e_2, respectively. From these plots, we can see that in most cases e_1 is the dominant error component.



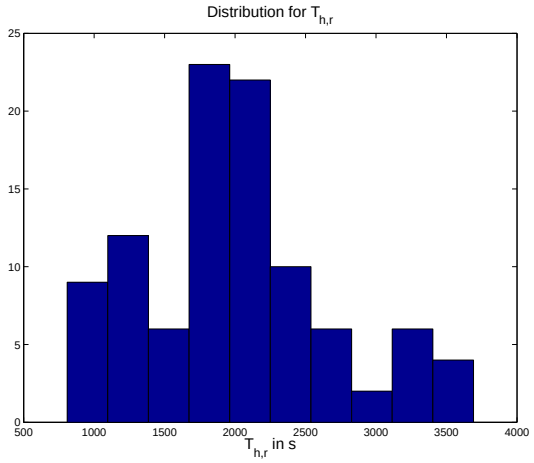
(a) T_{rest}



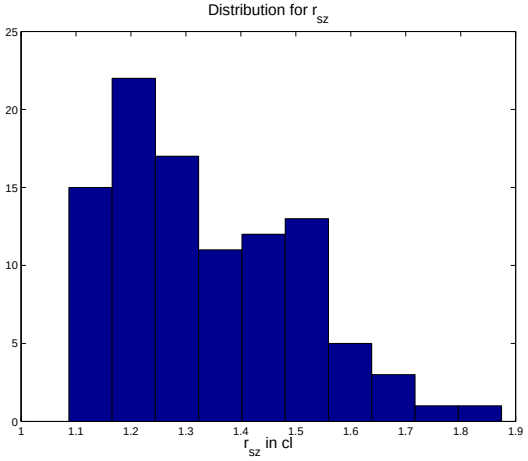
(b) T_{stand}



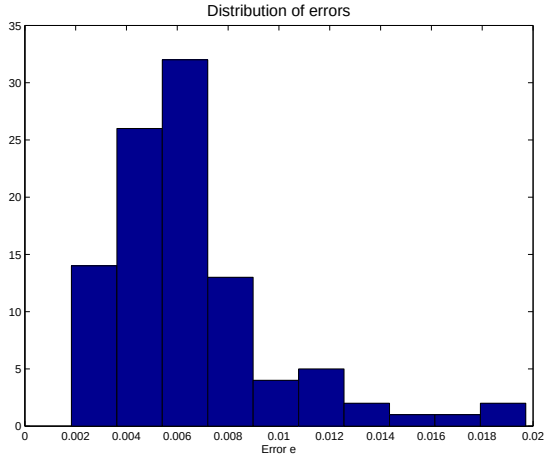
(c) $T_{h,s}$



(d) $T_{h,r}$



(e) r_{sz}



(f) Error value

Figure 7: Distributions of parameters and error found by simulated annealing process

3.4.3 Summary of results

In Section 3.4.1, we saw that the choice of certain parameter values can lead to good agreement between the model behaviour and the data gathered from the real world, which indicates that the model represents the resting behaviour observed in the videos reasonably well for this particular parameter set. However, in Section 3.4.2, we showed that a wide range of different parameter values can induce a good match to the data. This implies that the data used for the validation process is not sufficient to support the validity of the model. In order to obtain better results, the problems in the data discussed in Section 3.1.5 have to be addressed and the gaps have to be filled. As discussed in Section 3.4.2, it is especially important to store information about the pens and the surroundings.

Another problem in the data set is that only resting behaviour can be studied. This implies that the model cannot be used to predict movement or eating patters, as we do not have the data to validate these aspects. Therefore, the synchronisation considerations in Chapter 4 will mainly concentrate on resting effects.

From the plots shown in Figure 7, we will use the peak values for all further investigations. These values can be seen in Table 6.

Name	T_{rest}	T_{stand}	$T_{h,s}$	$T_{h,r}$	r_{sz}
Resulting value	2500 s	2000 s	2600 s	1900 s	1.2 cl

Table 6: Validation parameters obtained from ensemble of annealing processes

4 Synchronisation

The phenomenon of spontaneous synchronisation is widespread throughout a number of different scientific disciplines. In the popular science book "SYNC", Steven Strogatz explains the synchronisation of two pendulum clocks hung on a mutual support, the synchronisation of fireflies flashing in unison and more technical phenomena such as the synchronisation of Josephson-junctions in quantum physics [25]. In all these cases, a set of oscillators that would otherwise fluctuate in their own uncoordinated way are synchronised through weak coupling.

In the first section of this chapter, we want to review the classic Kuramoto model of weakly coupled phase oscillators. The next section then explains how the presented agent-based model can be interpreted as a set of weakly coupled oscillators and how synchrony can be measured for our model. In the last section, we perform some simulations in order to understand how the different order parameters are related and how a combination of them can be used to explore the synchronisation properties of cattle in a given barn. Here, we will experiment numerically with different sized bedding areas in order to compare the simulations to the observations by Færevik et al. in [8].

4.1 Weakly coupled oscillators

The following review of the Kuramoto model is paraphrased from its explanations in [24, Section 3]. The Kuramoto model consists of M nearly identical oscillators with natural frequencies $\omega_i,\ i = 1, \dots, M$. At every time $t \geq 0$, each of the M oscillators has phase $\phi_i(t)$, and the change in phase is governed by the equation

$$\dot{\phi}_i(t) = \omega_i + \sum_{j=1}^{M} \Gamma_{i,j}(\phi_j(t) - \phi_i(t)), \quad i = 1, \dots, M. \tag{4.1}$$

In this equation, the functions $\Gamma_{i,j}$ represent the *interaction functions* between the different oscillators. Kuramoto investigated the simplest possible case of equally weighted

coupling with the interaction functions defined to be

$$\Gamma_{i,j}(\phi_j(t) - \phi_i(t)) = \frac{K}{M}\sin(\phi_j(t) - \phi_i(t)), \tag{4.2}$$

with the coupling strength K. Hence, the governing equation becomes

$$\dot{\phi}_i(t) = \omega_i + \frac{K}{M}\sum_{j=1}^{M}\sin(\phi_j(t) - \phi_i(t)), \quad i = 1, \ldots, M. \tag{4.3}$$

Kuramoto defined the natural frequencies ω_i to be randomly distributed with a probability density function $p(\omega)$, which he assumed to be unimodal and symmetric with a mean frequency Ω. Without loss of generality, one can assume $\Omega = 0$ because of the system's global rotational symmetry. This is equivalent to considering the oscillators as rotating points on a unit circle, which itself rotates with angular velocity Ω contrary to the oscillators. Kuramoto introduced an *order parameter* to measure the degree of synchrony in the system by

$$re^{i\psi} = \sum_{i=1}^{M} e^{i\phi_i}. \tag{4.4}$$

The variable ψ is the average phase of the system, while r contains information of how synchronised the oscillators are. A value of r near 1 indicates a high level of synchrony – all the oscillators move in unison. If r is close to 0, the oscillators are spread all over the circle, as the different phases tend to cancel each other out.

When simulating this model with a fixed probability distribution $p(\omega)$ and different coupling strengths K, one can see an interesting phase transition. If K is smaller than a critical value K_C, no synchronisation is obtained and $r \to 0$ as $t \to \infty$. However, with a coupling above the critical value, synchronisation is obtained and $r \to r_\infty > 0$ as $t \to \infty$, where r_∞ grows with K.

4.2 Agents as oscillators

The system of coupled oscillators we consider here is the multi-agent system described in Chapter 2. In what sense can the agents be interpreted as oscillators? We saw that an agent can take 1 of 4 different states (*Standing, Walking, Resting, Eating*) and that

the transition between states is driven mainly by the variables a_i and b_i that represent, respectively, the desire to eat and rest. These variables both oscillate between the values 0 and 1. As we mentioned in Section 3.4.3, the model can only be used for predictions regarding the resting pattern, which is why we will concentrate on the oscillation of the variable b_i.

The system of oscillating agents is coupled through the avoidance rule explained in Section 2.2. Whenever one agent enters the zone of repulsion of another agent, it influences its movement and therefore its behaviour. A special case of this influence occurs when one agent forces another one to get up from resting by entering its sleeping zone. We consider the coupling between the agents to be weak, because one agent cannot actively enforce its resting cycle on another one.

The problem of measuring synchrony in this system is more difficult to address than in the abstract oscillator models. We will now introduce three different order parameters. The first one provides an *ad-hoc* approach to measure synchrony in a real barn, while the second one is inspired by the order parameter used in the Kuramoto model. The third possibility is adapted from the statistical analysis of multiple choice surveys.

4.2.1 *Ad-hoc* approach to measure synchrony

The simplest idea on how to measure synchrony in a barn is to measure the amount of time when all cows are lying down together [8, 19]. The higher the amount of mutual lying time, the higher is the synchrony between the different animals. This approach can be adapted and generalised for the agent-based model: we define the order parameter $\alpha_p,\ p \geq 0$ to be the fraction of the total simulated time T during which the fraction of cows currently lying is at least p. If we define the time T_p to be the amount of time satisfying this condition, then α_p can be calculated using the equation

$$\alpha_p = \frac{T_p}{T}. \tag{4.5}$$

The advantage of this order parameter is that it can be directly compared to data gathered from video footage. A disadvantage is that only lying cows are considered and that the

measure does not consider the difference between cows that just started resting and those that have already been resting for a while.

4.2.2 An abstract measure of synchrony

A second way to measure synchrony is adapted from the order parameter used by Kuramoto in his analysis of weakly coupled oscillators. In order to use this approach, we first need to define the phase of an agent. We want to understand the phase $\phi_i \in [0, 2\pi]$ of an agent as the position on its standing-resting cycle. This phase depends on the current state $s_i(t)$ and the current desire to rest $b_i(t)$ in the following manner

$$\phi_i(t) := \begin{cases} 2\pi \frac{b_i(t) T_{rest}}{T_{stand}+T_{rest}}, & s_i(t) = \textit{Resting}, \\ 2\pi \left(1 - \frac{b_i(t) T_{stand}}{T_{rest}+T_{stand}}\right), & s_i(t) = \textit{Standing}. \end{cases} \tag{4.6}$$

Because we chose $b_i(t)$ to evolve linearly in time, ϕ_i also passes through all values in $[0, 2\pi]$ linearly if the standing-resting cycle of the agent is undisturbed. However, if the cycle is disturbed by another agent or the agent becoming hungry while resting, the phase ϕ_i contains a discontinuity at the change between standing and resting, which will also cause a discontinuous behaviour of the order parameter defined by formula (4.4). If we want to obtain a single number as a synchronisation measure, we can average $r(t)$ over the time period $[T_{settle}, T]$, where T_{settle} is the time that we allow the system to settle down after the initialisation. For a continuous $r(t)$, this averaging can be done as follows:

$$\overline{r} = \frac{1}{T - T_{settle}} \int_{T_{settle}}^{T} r(t) dt. \tag{4.7}$$

Because the simulation of the agents is done in a time-discrete manner, the averaging has to be done using a summation rather than an integration. Hence, let L be the total number of simulation steps and $L_s = LT_{settle}/T$ the number of steps we allow the system to settle. We now define the order parameter β by

$$\beta := \frac{1}{L - L_s} \sum_{l=L_s+1}^{L} r(t_l). \tag{4.8}$$

One problem with this order parameter was already mentioned earlier: it can incorporate rapid changes from the possible jumps in the phases ϕ_i. Another disadvantage is that it

is a theoretical expression and cannot be measured on real cattle, as there is no way to measure the current desire to rest. A third problem can occur when all agents constantly disturb each others' lying cycle. In this case, $b_i(t)$ takes a value near 0 at all times and for all cows and $r(t)$ takes a value near 1, which suggests a high level of synchrony. We can, however, detect these events using the average phase $\psi(t)$ of the system. If $\psi_i(t)$ stays nearly constant around 0 for a long time, we expect the agents to constantly disturb each other. In order to identify these events, we define the *average phase displacement* $\overline{\psi}$ by

$$\overline{\psi} := \frac{1}{L - L_s} \sum_{l=L_s+1}^{L} |\psi(t_l)|, \tag{4.9}$$

where $\psi(t_l) \in [-\pi, \pi]$. A very small value of $\overline{\psi}$, which we choose to be $\overline{\psi} < 0.1$, indicates a constant disturbance in the multi-agent system. This case is most likely to happen if the lying area fails to provide enough space for all agents to rest at the same time.

4.2.3 Fleiss' Kappa statistics

Kappa statistics were first described by Cohen in [5] as a way to measure the agreement between the categorical ratings done by a set of raters. In 1971, Fleiss generalised that idea to obtain a measure for the extent of agreement between a number of N subjects rated into k categories by n raters [10]. The idea of such statistics is to count the number of agreeing ratings and compare them with the agreement expected by chance. The general formula takes the form

$$\kappa := \frac{\overline{P} - \overline{P}_e}{1 - \overline{P}_e}, \tag{4.10}$$

where $\overline{P}$ is the observed probability of agreement and $\overline{P}_e$ is the probability of agreement expected by chance. A positive value of κ indicates synchrony, with perfect unison for $\kappa = 1$. If we now understand every rater as an individual time t_l and classify the agents in the classes "Resting" and "not Resting", we can use the value κ to measure the synchrony between the agents' behaviour. Let us therefore denote by p_{rest} the probability that an individual agent is resting and by $p_{stand} := 1 - p_{rest}$ the probability that it is standing. At every time t_l, we can observe a number $N_{l,stand}$ of standing agents and a number $N_{l,rest} := N - N_{l,stand}$ of resting agents. The agreement measure at every time step is then

defined by

$$P_l := \frac{1}{N(N-1)} \left[N_{l,rest}(N_{l,rest} - 1) + N_{l,stand}(N_{l,stand} - 1) \right]. \tag{4.11}$$

These probabilities can now be averaged to calculate $\overline{P}$. We again allow a number of L_s steps for the system to settle down and obtain

$$\overline{P} := \frac{1}{L - L_s} \sum_{l=L_s+1}^{L} P_l. \tag{4.12}$$

The probability of agreement by chance can be calculated using the *a priori* probabilities p_{stand} and p_{rest} by

$$\overline{P}_e := p_{stand}^2 + p_{rest}^2. \tag{4.13}$$

4.3 Simulations and results

In the next step, we want to investigate how the different synchrony measures interact and how their combination can be used to detect synchrony. Additionally, we will try to obtain in our simulations the results observed by Færevik et al. in [8].

4.3.1 Færevik et al.'s experiments

Færevik et al. performed a set of experiments with groups of calves housed in pens with different sized bedding areas. They used a relatively simple barn layout (see Figure 8). Again, the green area indicates the food trough and the grey area indicates the bedding. The values for the length x of the bedding area can be found in Table 7. In all cases, the pen housed 5 animals of the same age.

Configuration	*Small*	*Medium*	*Big*
x	1.25 m	2.08 m	2.92 m

Table 7: Length of the bedding area used by Færevik et al.

In the experiments, Færevik et al. showed that a bigger bedding area leads to a higher synchrony, which they measured by the amount of time all cows rest simultaneously.

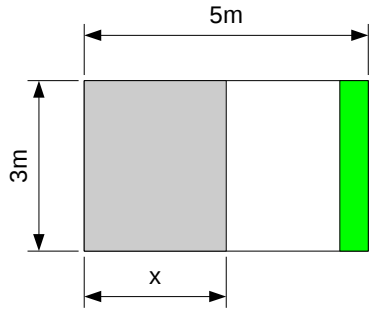


Figure 8: General barn layout used by Færevik et al.

Additionally, they concluded that the difference in synchrony between the configurations *Small* and *Medium* is higher than that between *Medium* and *Big*.

4.3.2 Simulation results

In order to investigate the synchronisation properties, we conducted simulations that mimicked the experiments done by Færevik et al. For similar reasons as during the validation process, we used $\tau = 0.1$. Because every one of Færevik et al.'s experiments only needs a single simulation, we can allow more simulation steps than during the validation process and therefore used $L = 200,000$ steps for each of the three configurations. In order to ensure that results are not biased by random behaviour, we performed 50 repetitions for each of the configurations and calculated ensemble averages of the order parameters. These averaged order parameters can be seen in Table 8. We show distribution plots of the different order parameters for each of the three configurations in Figure 17 and 18 in Appendix D.

Configuration	*Small*	*Medium*	*Big*
$\alpha_{0.7}$	0.29	0.28	0.25
β	0.89	0.81	0.77
$\overline{\psi}$	0.31	0.36	0.40
κ	-0.01	-0.11	-0.13

Table 8: Order parameters for the simulations of Færevik et al.'s experiments

Generally, we can see that synchrony can occur, which is indicated by $\alpha_{0.7} > 0$ and the high values of β. However, we see that for all three of the order parameters the course

is opposite to the one found by Færevik et al. during their experiments - the highest values occur for *Small*, the second highest for *Medium* and the lowest for *Big*. For $\alpha_{0.7}$ we also have to take into account that the difference between the values for the three configurations is negligible.

Another problem that appears in these experiments is that κ is negative in all cases, which generally indicates no synchrony. In the simple configuration used here with only 5 cows, the number of cows doing the same thing at a time needs to be 4 or 5 in order to lift κ to a positive value. However, the nature of the model makes it unlikely that 4 or even all 5 agents rest at the same time in the relatively small bedding area. Hence, a negative value of κ does not necessarily mean that no synchronisation occurs at all, but again a combination of the 3 measures has to be used to judge the synchrony of the system. This negative value of κ could also indicate that κ is not an appropriate way of measuring synchrony, at least not for such a small number of agents.

In order to evaluate the relations between the different order parameters, we calculated the correlation between them for all three configurations. A short introduction to the calculation of correlation coefficients can be found in Appendix C. The correlation coefficients are shown in Table 9 - 11.

	$\alpha_{0.7}$	β	$\overline{\psi}$	κ
$\alpha_{0.7}$	–	0.14	0.50	0.66
β	0.14	–	−0.56	0.09
$\overline{\psi}$	0.50	−0.56	–	0.59
κ	0.66	0.09	−0.59	–

Table 9: Linear correlation coefficients for *Small*

	$\alpha_{0.7}$	β	$\overline{\psi}$	κ
$\alpha_{0.7}$	–	0.12	0.67	0.73
β	0.12	–	−0.18	0.22
$\overline{\psi}$	0.67	−0.18	–	0.80
κ	0.73	0.22	0.80	–

Table 10: Linear correlation coefficients for *Medium*

	$\alpha_{0.7}$	β	$\overline{\psi}$	κ
$\alpha_{0.7}$	–	0.14	0.68	0.77
β	0.14	–	0.27	0.53
$\overline{\psi}$	0.68	0.27	–	0.83
κ	0.77	0.53	0.83	–

Table 11: Linear correlation coefficients for *Big*

From these tables we see that the three parameters $\alpha_{0.7}$, $\overline{\psi}$ and κ have a strong positive correlation between them in all three configurations. This comes as a surprise if one considers the fact that $\overline{\psi}$ is not directly an order parameter but was introduced to detect queueing at the food troughs. However, the synchrony measure β that we adapted from the Kuramoto model does not correlate with the other two order parameters $\alpha_{0.7}$ and κ, which suggests that β is not an appropriate order parameter.

5 Barn optimisation

After investigating different ways of measuring synchrony in the multi-agent system, the next step is to develop an algorithm that finds barn configurations that favour the occurrence of synchronisation. This optimisation shall be done with respect to a maximum budget b available for the design of the barn. We assume that a unit square of feeding area costs c_f and a unit square of bedding area c_r. Hence, the condition that has to be satisfied by the barn configuration is

$$c_r A_r + c_f A_f \leq b, \tag{5.1}$$

where A_b and A_f are the size of the bedding and feeding areas, respectively. These sizes are measured in unit squares. Additionally, we assume that the general shape of the barn is fixed and given by the set $B \in \mathbb{R}^2$ similarly to all earlier considerations. Other inputs are the possible feeding and bedding areas $F_{poss}, R_{poss} \subseteq B$, which we assume to be disjoint. The algorithm shall calculate sets R and F that represent the bedding and the feeding areas, respectively, and which satisfy the inequality (5.1). In this algorithm, we do not allow changes in the pen shape, as the farmers do not intend to rebuild their barns.

In the first section of this chapter, we describe the algorithm we developed to solve this optimisation problem. The second section shows the result of the algorithm for an example.

5.1 Algorithm

A pseudo-code of the developed algorithm can be seen in Algorithm 3. Every step of this pseudo-code will be described individually below.

Algorithm 3: Optimisation algorithm

Input: $c_f, c_r, b, B, F_{poss}, R_{poss}$

Output: B, F

```
Tiling of the barn;
Find initial solution;
for i = 1, ..., N_optsteps do
    Make changes to the barn;
    if new barn is better than current best barn then
        Overwrite current optimal barn with new barn;
    end
end
```

5.1.1 Tiling of the barn

In the first step of the algorithm, we tile the barn with $n_x \times n_y$ rectangular tiles (see Figure 9). During the whole algorithm we will use these tiles as units that build up the feeding and bedding areas. In Figure 9, we can also see some problems this sort of tiling implies. Some of the tiles are only partly inside the barn and some are even completely outside the barn. Because we later want to measure the size of feeding and bedding areas exactly, we need to calculate the area of each tile that overlaps with the barn. Because the barn can have a general polygonal shape, we have to perform a polygon clipping algorithm first. We chose to use a specialised version of the Sutherland-Hodgman clipping algorithm [26], which is explained in Appendix B.2. The area of the clipped polygon can then be calculated by the formula explained in Appendix B.3.

For each of the tiles $T_{i,j}$, we now calculate if it could be a bedding tile, a feeding tile, or none of these. If the tile $T_{i,j}$ lies completely inside the possible bedding area B_{poss}, it is considered a possible bedding tile. Similarly, it is a potential feeding tile if it lies inside F_{poss}. Because we required B_{poss} and F_{poss} to be disjoint, a tile cannot be both. In order to test whether a tile lies entirely inside one of the polygonal regions B_{poss} or F_{poss}, we

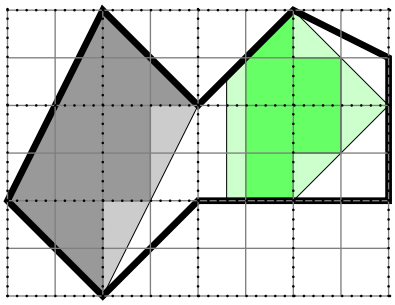


Figure 9: Tiling of the barn (light green - F_{poss}, dark green - possible feeding tiles, light grey - R_{poss}, dark grey - possible bedding tiles)

can again use the clipping algorithm explained in Appendix B.2. If the clipped polygon differs from the tile, the tile does not lie inside the polygonal region.

In Section 2.3, we mentioned that the simulation needs all sets in a polygonal form. Hence, this form needs to be obtained from the chosen tiles for both the feeding and bedding areas. The algorithm we used to attack this problem is explained in Appendix B.4.

5.1.2 Initial solution

After the tiling is performed and we know for every one of the rectangular tiles whether it can be used for a feeding area or a bedding area, we want to find an initial barn configuration to start the optimisation process. A pseudo-code version of this can be seen in Algorithm 4. In this algorithm, the random choices of tiles are always done uniformly.

Algorithm 4: Calculate initial solution for optimisation

Input: $F_{t,p} = \{T_{i,j} | T_{i,j} \text{ is possible feeding tile }\}$,
$B_{t,p} = \{T_{i,j} | T_{i,j} \text{ is possible bedding tile }\}, c_f, c_r, b$

Output: Initial solution

```
Choose T_{i_0,j_0} from F_{t,p} randomly as initial feeding tile;
Choose T_{k_0,l_0} from B_{t,p} as tile with maximum Manhattan-distance
|i_0 − k_0| + |j_0 − l_0| from initial feeding tile as initial bedding tile;
while Inequality (5.1) holds do
    if random > 0.5 then
        Add T_{i_m,j_m} from F_{t,p} randomly to feeding area, if possible choose tile
        neighbouring to current feeding tiles;
    else
        Add T_{k_m,l_m} from B_{t,p} randomly to bedding area, if possible choose tile
        neighbouring to current bedding tiles;
    end
end
Delete last added tile to ensure that (5.1) holds;
```

To make this and further parts of the algorithm more effective, the possible feeding and bedding tiles area stored in a list, with the actually used tiles first, the tiles neighbouring at least one used tile second, and the other tiles after that. This allows every update step to happen in constant time. When a tile is chosen as a feeding or bedding tile, it is moved in the list to its newly appropriate position. Also, the 4 neighbouring tiles have to be updated and possibly moved as well.

5.1.3 Test if new barn is better

In order to evaluate if the new barn configuration has a higher tendency to favour synchronisation than the currently best one, a simulation of the agents in the new barn is performed and the results are compared. The goal of the optimisation algorithm is to find a barn configuration with the best possible synchronisation properties. As mentioned in

Chapter 1, higher synchrony between the cows indirectly leads to a higher growth rate and welfare of the cows, which is in the mutual interest of farmers and cows. Hence, we will use a combination of the order parameters developed in Section 4.2 to compare the different barns.

First, the algorithm checks if constant disturbance occurs. Because this is an unwanted effect, barns that lead to the agents disturbing each others' resting cycles are dismissed immediately. For this test, the average phase displacement $\overline{\psi}$ defined in Section 4.3 is used. If this value is very small (again, if $\overline{\psi} < 0.1$), we assume that the agents constantly disturb each others' lying periods and that the barn is insufficient for further considerations.

If the new barn is not dismissed by the disturbance criterion, the 3 synchronisation measures are compared directly to the ones of the current configuration. If at least 2 of the 3 measures indicate a higher synchronisation during the simulation, the new barn is accepted.

5.1.4 Perform changes in the barn

During the optimisation process, new tiles are chosen to be part of the feeding or bedding area, while other tiles are deleted from those. In every one of those steps, we will delete one bedding and feeding tile and add different tiles as long as the budget is not exceeded. In order to identify whether the feeding or bedding area should be bigger than before, a set of heuristics can be used. These heuristics evaluate data gathered from the simulation.

The first main idea is to investigate whether the bedding area is to small to allow for regular resting periods. One way to determine this is to use the average phase displacement defined in Section 4.3. If the value $\overline{\psi}$ is small, we assume that the agents constantly disturb each others' lying periods and that a larger bedding area is required.

Similarly, if queueing at the feeding area turns out to be a major issue during the simulation, we decide to enlarge the feeding area. As a measure for the amount of queueing, we use the average time an agent stays in the state $b_i(t_l) = 0$, which represents hunger.

Large values of this average length can indicate queueing for food.

To decide which of the old tiles is deleted from the feeding and bedding areas, one can measure the average time agents spend in each tile. The tiles with the lowest average time should be most likely to be removed.

5.2 Simulation

We now illustrate the algorithm using an example. In Figure 10, we can see the general shape of the barn along with the possible feeding and bedding areas in green and grey respectively. We used 10 agents for this barn.

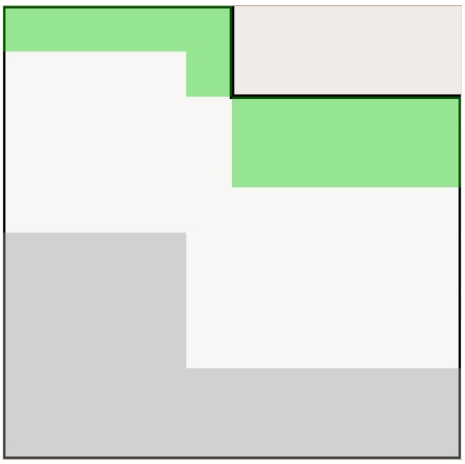


Figure 10: Possible feeding and bedding areas

The tiling divides this barn into 10×10 rectangular tiles. The costs were chosen in a way that every full tile of feeding costs 0.8 units and every full tile of bedding costs 0.5 units. Hence, all possible areas together would require an amount of about 30 units. In order to allow about two thirds of the tiles to be used, we chose the budget b to be 20 units. The parameters of the agents were chosen as calculated in the validation process (see Table 6). For similar reasons as during the validation process, we choose the number of steps to be $100,000$ and $\tau = 0.1$.

In Figure 11, we show the initial solution of the algorithm. Here, we can clearly see that both the feeding and the bedding area feature pathologies like holes or not connected

tiles. The synchronisation measures for this barn are $\alpha_{0.7} = 0.23$, $\beta = 0.81$, $\overline{\psi} = 0.21$ and $\kappa = 0.06$.

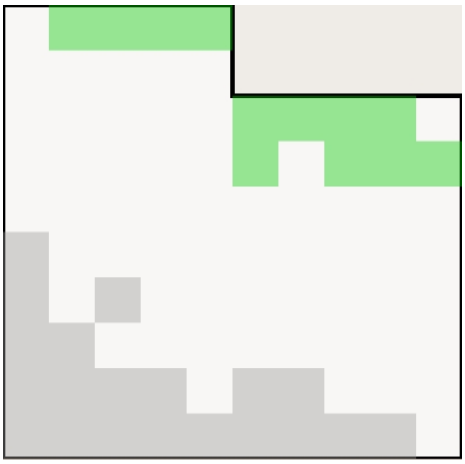


Figure 11: Initial solution of optimisation algorithm

The final solution of the optimisation process can be seen in Figure 12. Here, the synchronisation measures are $\alpha_{0.7} = 0.31$, $\beta = 0.9$, $\overline{\psi} = 0.37$ and $\kappa = 0.23$. All of the order parameters improved significantly from the initial solution. For the final solution, we can also see that both the feeding and the bedding area are much more compact than for the initial solution.

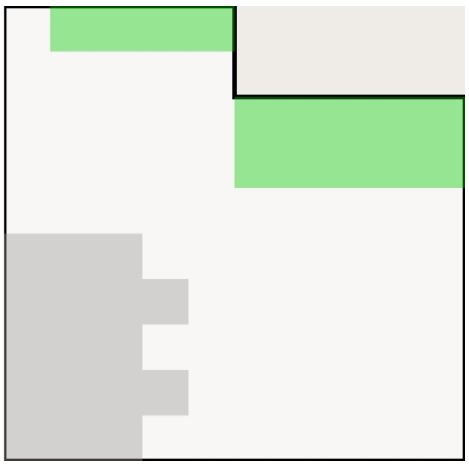


Figure 12: Final solution of optimisation algorithm

6 Conclusions and future work

In this dissertation, we developed an agent-based animal behaviour model that we used to examine the behaviour of beef cattle in a barn. The model generalised the work of Couzin et al. [6] by forcing the agents to eat and rest regularly. The changes between these different states were governed by a finite state machine based on the agents' desire to eat and rest. In order to determine the parameters used in the model, we implemented an optimisation algorithm based on simulated annealing. The data for this was gathered from video footage of different barns. We were able to find parameter sets that allowed a good match with the data. However, when repeating the validation process, we saw that the outcome was somewhat inconsistent, as very different parameter sets were found to match the data reasonably well. This effect arose mainly from the incompleteness of the data. For further investigation, we therefore suggest that more data is gathered with special care of complete day and night data. Additionally, the cameras should be installed in a way so they cover the whole barn and all animals. The shapes of the pens should be stored exactly. With such data, one could perform a more thorough and accurate model evaluation by using cross-validation techniques for the different barns. Here one would fit the model parameters to the data obtained from of a subset of the barns. Afterwards one could test the validity of the model using the rest of the data.

We then investigated the effect of spontaneous synchronisation emerging from the rules that the agents follow. Inspired from research on weakly coupled oscillator systems, we developed 3 different ways of measuring synchrony between the agents. Using these order parameters, we performed numerical experiments similar to the experiments with real barns done by Færevik et al. in [8]. We were able to show that a small amount of synchrony occurs in these experiments. However, opposing to the results in [8], the simulations showed that a larger bedding area leads to lower synchronisation in the multi-agent system. For further research, we suggest to carry out more simulations to investigate the ability of the different order parameters to indicate synchrony in more detail. One could also perform experiments with actual herds by changing the sizes of bedding and feeding areas. This data could be compared to simulation results to examine the validity

of the model regarding synchronisation effects.

The last step was the development of an algorithm to optimise the design of pens. We assumed that the general shape of the pens was fixed and that only a certain amount of money was available to build the feeding and bedding areas. We then illustrated the work of the algorithm using an example barn. It might be beneficial to again combine this algorithm with simulated annealing to potentially find a configuration closer to the global optimum. Additionally, one could perform a cost-benefit analysis with different budgets to weigh the amount of money it is worth spending or allow changes in dimension and shape of the pen. Again, long-term experiments could be conducted in a special barn where the growth rate of the cattle can be measured on a regular basis. These experiments could then show whether the change in the pen configuration actually influenced the outcome.

A completely different direction one could take with this model or a simpler version of it is mathematical analysis rather than simulations. In a more abstract setting the emergence of synchrony could be investigated in mathematically exact terms and one could try to prove that a certain amount of synchrony has to occur. However, the mathematical investigation of spontaneous synchronisation requires techniques that are presently only suited to analyse relatively simplistic models. This implies that the model presumably had to be simplified in order to allow this type of analysis.

More interesting for farmers would be to refine the model to improve the resemblance of the agent dynamics to actual cattle behaviour. Possibilities in this direction include an additional state of drinking or the consideration of more complex social interactions. The latter could be used to analyse the coupling strength between agents in order to examine whether the assumption of weak coupling is reasonable. One could also add external effects to the model, such as different types of food, fixed feeding times, and a day-night cycle. Probability models based on Markov-chains could perhaps be used instead of the finite state machine. In these models, the state of the agent changes with certain probabilities depending on external facts, whereas in the existing model such state changes are entirely deterministic.

A Graphical user interface

One long-term goal of the investigations in this project is to provide advice to farmers on how to design their barns to optimise cattle welfare and growth rate. In order to give them the opportunity to easily experiment with different configurations on the computer, we designed a graphical user interface (GUI). This is also helpful to visualise the simulation results and get an overview of the agents' behaviour during the process of this project.

A screenshot of this interface can be seen in Figure 13. A special colour-coding was used to symbolise the current status of each of the agents. An agent coloured in grey means it is currently resting. A shading from red ($a_i = 0$) to blue ($a_i = 1$) shows the state of hunger. The grey area in the background represents the bedding and the green area the feeding. The configuration of the barn and the number of agents can be read through an XML-file. The shape of the barn, feeding areas and bedding areas can be simple polygons or circles. For the barn, it is also possible to use a rectangle with periodic boundary conditions. Here, agents that move out on one end of the barn return on the opposite end. This possibility can be useful during a more abstract inspection of the system, as the absence of walls simplifies the analysis substantially.

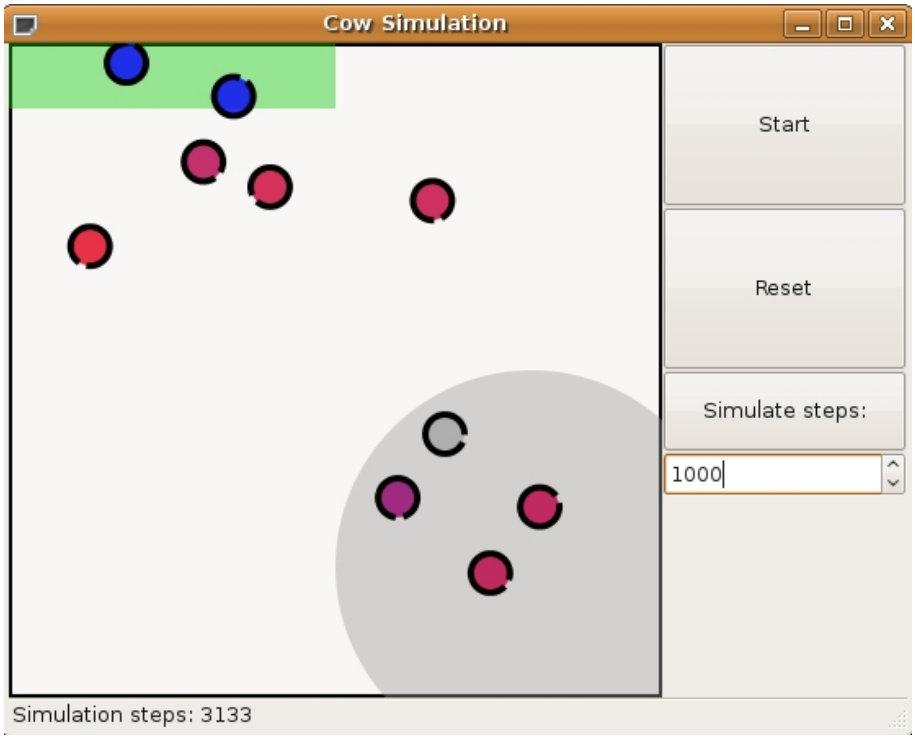


Figure 13: Screenshot of graphical user interface

B Polygon algorithms

In the present section, we explain and give additional references to the polygon algorithms that we used. In all of these algorithms, we understand a polygon as an ordered set of $n \geq 3$ points $P_0, P_1, \ldots, P_n \in \mathbb{R}^2$ with $P_0 = P_n$. The points $P_i,\ i = 0, \ldots, n$ are called the nodes of the polygon. The x- and y-coordinates of the points $P_i,\ i = 0, \ldots, n$ are denoted by x_i and y_i respectively.

B.1 Check if point is inside polygon

The standard algorithm for checking whether a point lies inside a polygon or outside is the so-called *Ray-casting-algorithm* [21]. In this algorithm, a ray is shot starting from the test point in a fixed direction (usually the positive x-direction). One counts the number of intersections between this ray and the boundary lines of the polygon. A point is found to lie inside a polygon if and only if its ray has an odd number of intersections with the polygon boundary. A pseudo-code implementation of this algorithm can be found in [21,

Section 2.2]. The algorithm takes $\mathcal{O}(n)$ computational time, because it iterates through every edge of the polygon.

Some special cases of this algorithm have to be considered separately. If the ray goes directly through one of the nodes, P_i say, this has to be seen as a single intersection, even though the ray shares common points with the two edges leaving P_i. Another special case occurs, if one of the polygon lines is parallel to the ray, and they lie on top of each other for a finite distance. This is only counted as a single intersection, even though the ray passes through two nodes. An additional test can be added to determine whether the tested point lies on the boundary of the polygon. This can also be done in $\mathcal{O}(n)$ time by simply iterating through every line of the polygon and testing whether the point lies on it.

In Figure 14, we show a polygon with two test points A and B. Point A lies inside the polygon and its ray has 3 intersections with the the edges of the polygon. The ray starting from point B has 4 intersections with the polygon and hence B is rightly ruled not to be in the interior.

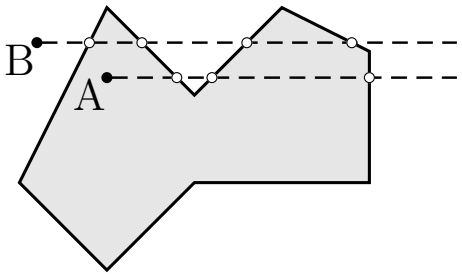


Figure 14: Point inside polygon

B.2 Polygon clipping

Polygon clipping is the process of clipping one polygon to the area defined by another one. A standard algorithm for this problem is the Sutherland-Hodgman algorithm [26]. During the barn optimisation, we need a special case of polygon clipping, where the clipping polygon is a rectangle. In Figure 15, we show the result of the clipping process.

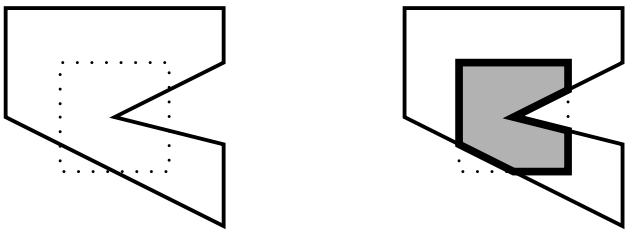


Figure 15: Polygon clipping

The main idea of the algorithm is to iterate through all edges of the clipping polygon and to clip the original polygon with respect to the current edge. During each of those line clippings, all points of the original polygon that lie on the side of the line facing away from the clipping polygon are deleted. All intersection points between the original polygon and the clipping edge are added as new points to the polygon. One weakness inherent in this algorithm is that the clipping polygon has to be convex to ensure the correct results. Because we only use rectangles as clipping polygons, this problem does not diminish the suitability of the algorithm in this case. This algorithm takes $\mathcal{O}(n)$ time if the clipping polygon is a rectangle, because it iterates through all points of the current polygon for each of the 4 edges of the rectangle. The number of nodes in the clipped rectangle cannot be more than $3n$, because every edge of the original polygon can only produce 2 intersection points with the rectangle.

B.3 Area of a polygon

During the optimisation algorithm, we need to calculate the fraction of each of the tiles that lies inside the barn polygon. Therefore, we need a way of calculating the area of a regular but not necessarily convex polygon. A formula is given by the following theorem.

Theorem B.1. *The area of a polygon given by the points $P_0, \ldots, P_n \in \mathbb{R}^2$ with $P_0 = P_n$ and the coordinates $P_i(x_i, y_i)$, $i = 0, \ldots, n$ can be calculated using the so-called surveyor's formula given by*

$$A = \frac{1}{2} \sum_{i=0}^{n-1} (x_i y_{i+1} - x_{i+1} y_i). \tag{B.1}$$

Proof. To prove this result, we use Green's Theorem for planar areas. In [14, Ch. 20.3,

Theorem 1], Green's Theorem is given in the following form

$$\iint_\Omega \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dxdy = \oint_{\partial\Omega} Pdx + Qdy, \tag{B.2}$$

where P and Q are continuous with continuous first partial derivatives in an open disk containing the simply connected region $\Omega \subseteq \mathbb{R}^2$. We define the region Ω to be the interior of the polygon given by $P_0, \ldots, P_n$. The area of this polygon can be calculated by

$$A = \iint_\Omega 1 dxdy.$$

Hence, if we define the functions P and Q so that

$$1 = \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}.$$

Then we can use Green's Theorem to get a formula for the area A of the polygon. We choose $P = -\frac{1}{2}y$ and $Q = \frac{1}{2}x$ and get

$$A = \frac{1}{2}\oint_{\partial\Omega} xdy - ydx.$$

Here, the boundary $\partial\Omega$ is represented by the path $P_0P_1 \ldots P_n$ around the polygon. Thus, we can split the integral into integrals along the lines $P_iP_{i+1},\ i = 0, \ldots, n-1$ to get

$$A = \frac{1}{2}\sum_{i=0}^{n-1} \int_{P_iP_{i+1}} xdy - ydx. \tag{B.3}$$

Let us now define the curve $\gamma_i(t),\ i = 0, \ldots, n-1$ by

$$\gamma_i(t) := (1-t)\begin{pmatrix} x_i \\ y_i \end{pmatrix} + t\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}, \quad i = 0, \ldots, n-1.$$

The integrals in equation (B.3) can now be calculated as follows:

$$\begin{aligned}
A &= \frac{1}{2}\sum_{i=0}^{n-1} \int_{P_iP_{i+1}} \begin{pmatrix} -y \\ x \end{pmatrix} d\gamma_i \\
&= \frac{1}{2}\sum_{i=0}^{n-1} \int_0^1 \begin{pmatrix} -y_i - t(y_{i+1} - y_i) \\ x_i + t(x_{i+1} - x_i) \end{pmatrix} \cdot \frac{d\gamma_i}{dt} dt \\
&= \frac{1}{2}\sum_{i=0}^{n-1} \int_0^1 \begin{pmatrix} -y_i - t(y_{i+1} - y_i) \\ x_i + t(x_{i+1} - x_i) \end{pmatrix} \cdot \begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{pmatrix} \\
&= \frac{1}{2}\sum_{i=0}^{n-1} (x_iy_{i+1} - x_{i+1}y_i).
\end{aligned}$$

□

This result can also be proved using simpler geometric considerations, shown in [20, Theorem 1.4.3].

B.4 Polygon gathering

In the last step of the optimisation algorithm we need to obtain a polygon representation of the feeding and bedding area from their representation as used tiles. This algorithm starts with a finite set $\mathcal{M} \subseteq \{T_{i,j}\}$ of tiles that define the simply connected set $M := \bigcup_{T_{i,j} \in \mathcal{M}} T_{i,j}$. Since M was created from rectangular tiles, its boundary can be described as a polygon. The algorithm first finds a tile $T_{i,j} \in \mathcal{M}$ on the boundary of M and uses an edge of $T_{i,j}$ that points to the outside of M as the starting point. From this edge, the algorithm traverses the boundary of M by always taking the leftmost way at every lattice point of the tiling. Because the set $\mathcal{M}$ is finite, this algorithm is guaranteed to terminate.

C Correlation

The idea to calculate correlations between two random variables X and Y was first introduced by Sir Francis Galton in 1877 [11]. For a set of sample values $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ of the random distributions X and Y, he defined the correlation coefficient r_{XY} as follows: (see also [22, Section 7.5.4])

$$r_{XY} = \frac{1}{(n-1)s_X s_Y} \sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y}), \tag{C.1}$$

where s_X and s_Y are the standard deviations of the sets $\{X_1, \ldots, X_n\}$ and $\{Y_1, \ldots, Y_n\}$, respectively. These standard deviations can be calculated using

$$s_X = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (X_i - \overline{X})^2}. \tag{C.2}$$

The value $\overline{X}$ is the mean of the sample set $\{X_1, \ldots, X_n\}$, which is defined by

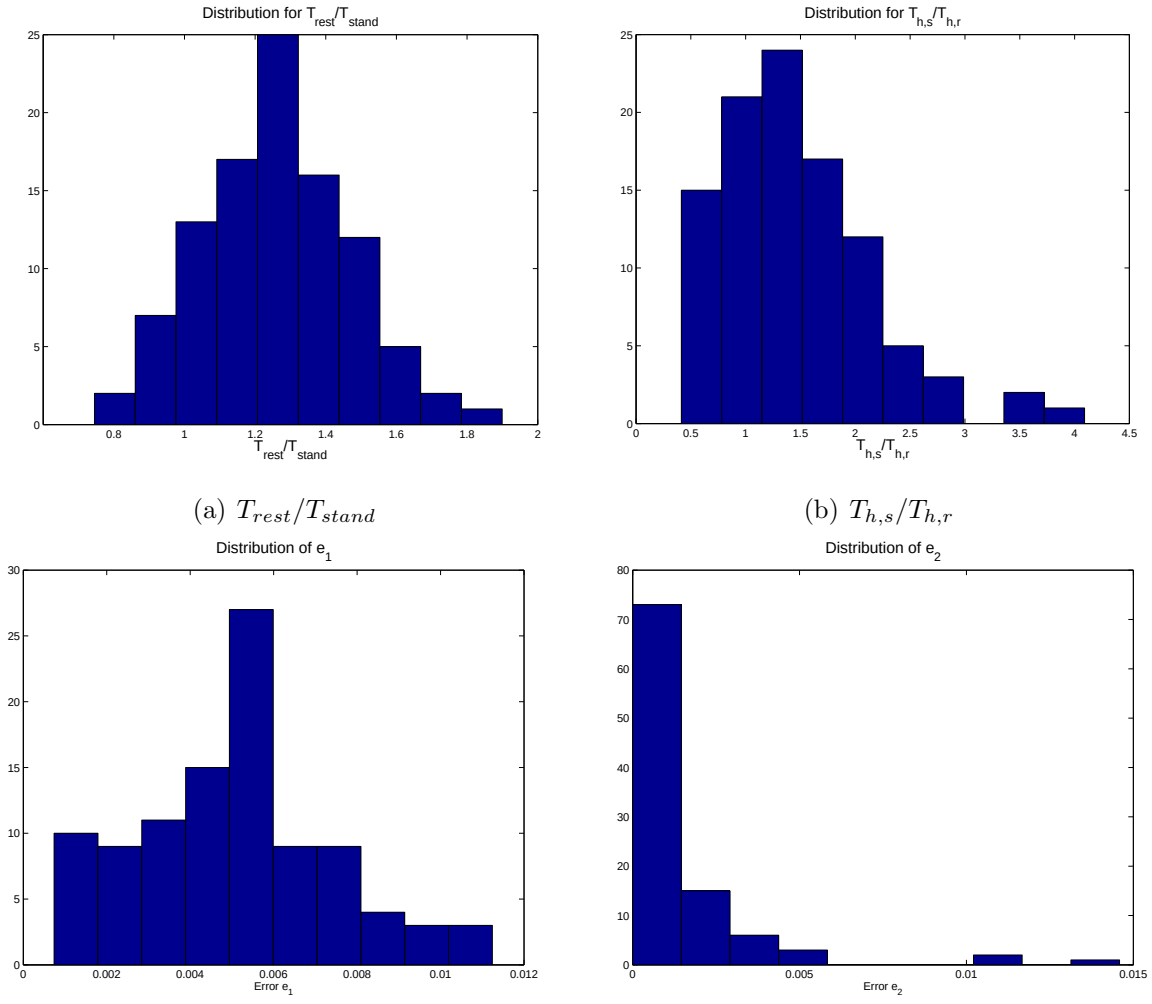
$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i. \tag{C.3}$$

Equations (C.2) and (C.3) are explained in more detail in [22, Section 1.5.2].

D Additional plots

Additional plots for the validation process explained in Chapter 3 are shown in Figure 16.

In Figure 17 and 18, we can see the distributions of the order parameters obtained from the simulations explained in Section 4.3.



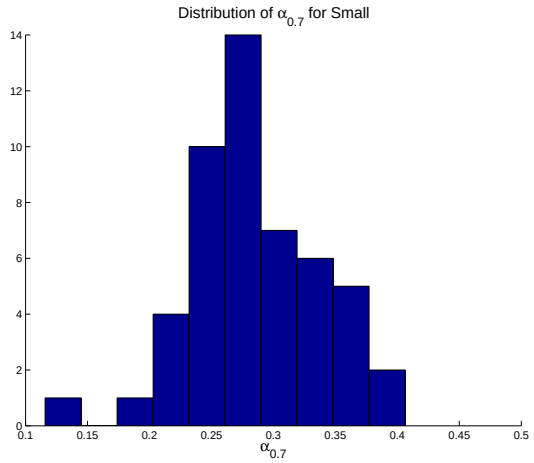
(a) T_{rest}/T_{stand}

(b) $T_{h,s}/T_{h,r}$

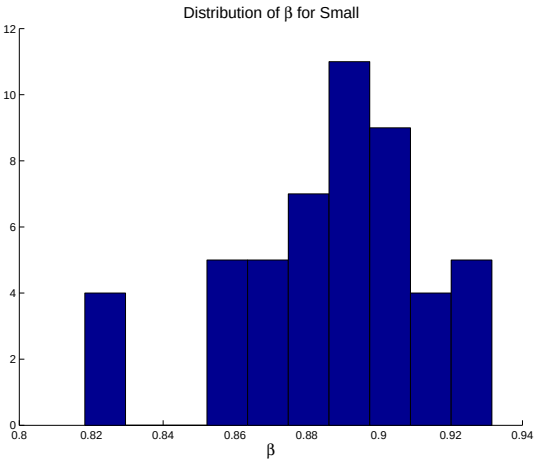
(c) e_1

(d) e_2

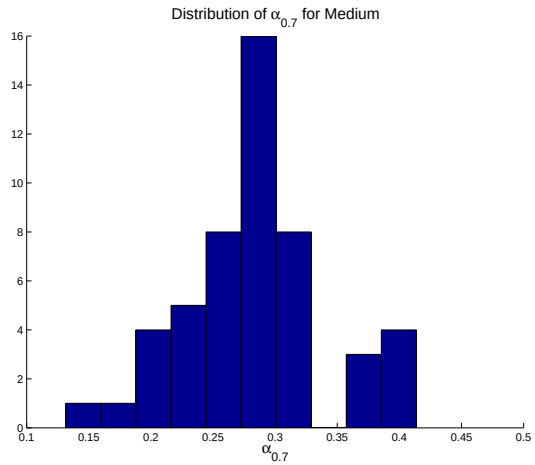
Figure 16: Additional distribution plots for the validation process



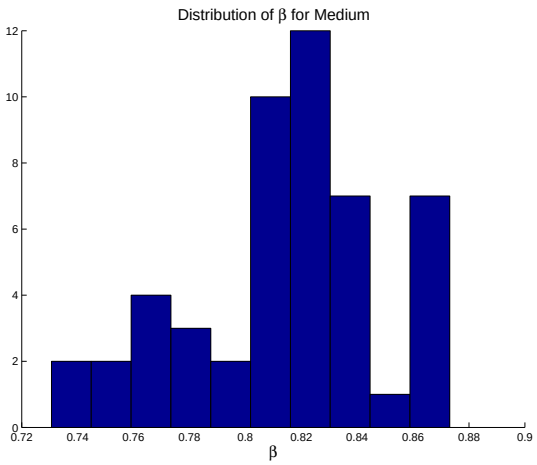
(a) $\alpha_{0.7}$ for *Small*



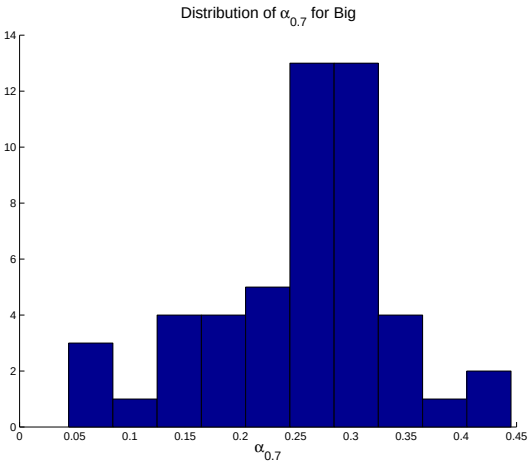
(b) β for *Small*



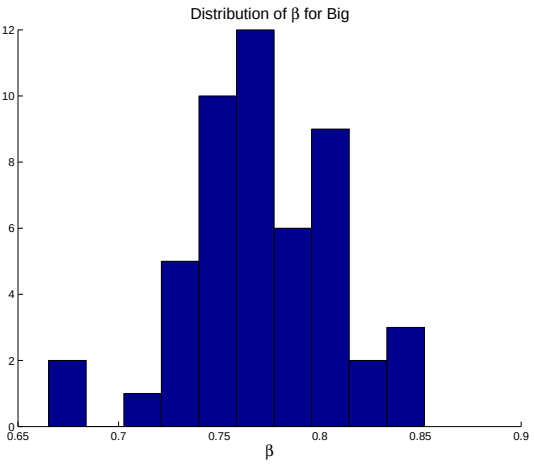
(c) $\alpha_{0.7}$ for *Medium*



(d) β for *Medium*

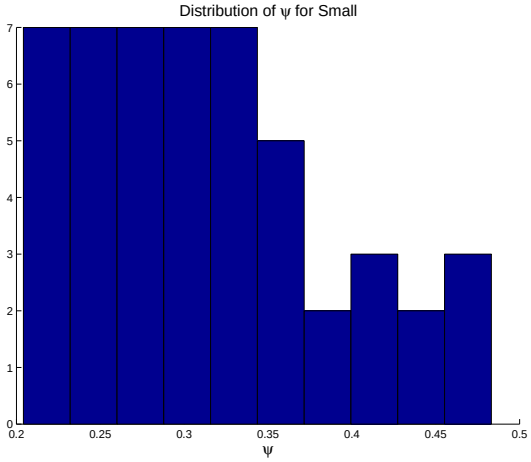


(e) $\alpha_{0.7}$ for *Big*

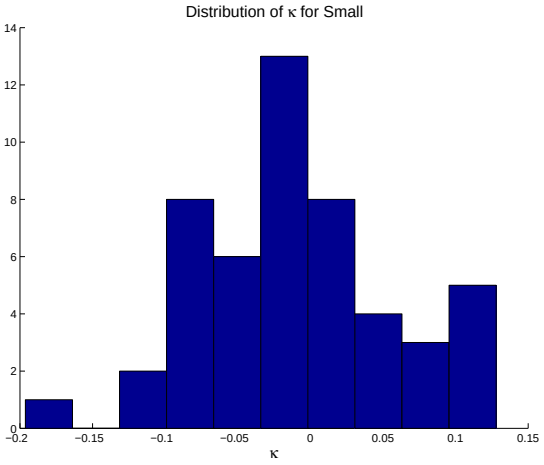


(f) β for *Big*

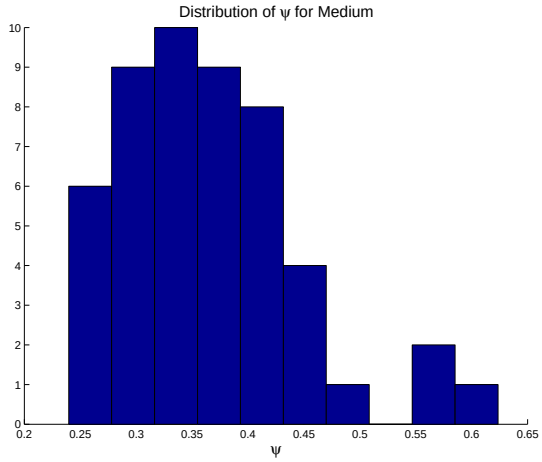
Figure 17: Distributions of order parameters $\alpha_{0.7}$ and β in synchronisation simulations



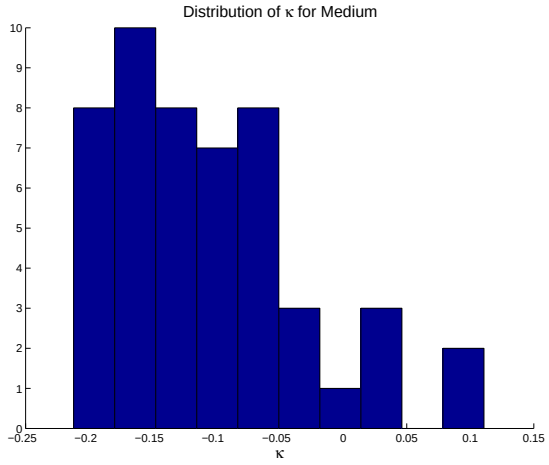
(a) $\overline{\psi}$ for *Small*



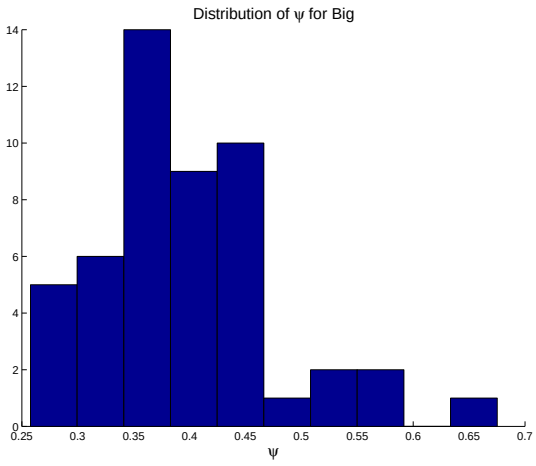
(b) κ for *Small*



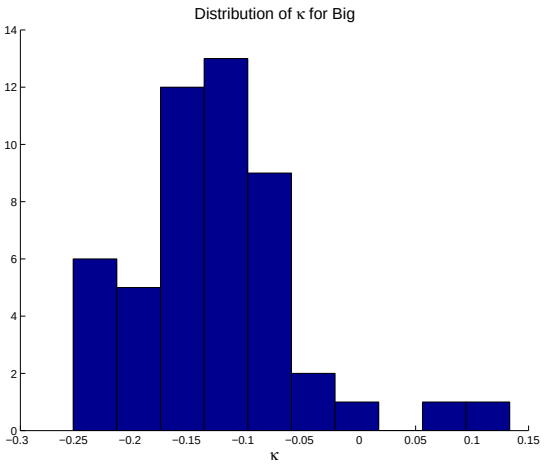
(c) $\overline{\psi}$ for *Medium*



(d) κ for *Medium*



(e) $\overline{\psi}$ for *Big*



(f) κ for *Big*

Figure 18: Distributions of order parameters $\overline{\psi}$ and κ in synchronisation simulations

E Sample data

In this section we explain the data used during the model evaluation in more detail and show samples of it.

E.1 Lying data

Below, we can see the first entry in one of the Lying data files. This entry represents exactly one of the lying events explained in 3.1.1. It starts with an integer (0 in this case), which represents a numbering of the entries in the file. The two numbers following that are the time when lying starts t_1 and the time of getting up t_2 respectively. Thereafter, 4 real numbers give the position of the resting cow. The first two are the x- and y-coordinate of the head and the next two are the x- and y-coordinate of the tail. After that, an integer indicates the number of visible standing cows at the time t_1. The positions of these cows follow in a similar format to the position of the lying cow. The same is repeated for the time t_2.

```
0 7285 8197.74
   0.114041 1.39079  0.069155 1.79707
5  0.337741 0.415522 0.623464 0.410706
   0.289616 0.206006 0.564482 0.0623421
   0.90663  0.454256 0.722778 0.143268
   0.299985 0.57927  0.64284  0.569142
   1.15566  0.497808 0.960707 0.664036
5  0.523665 0.163361 0.765882 0.065669
   1.2181 1.24214  1.36934  1.56161
   0.99963  1.2119   1.21644  1.45404
   0.885585 1.37389  1.12734  1.64357
   0.679697 1.41685  0.927193 1.68001
```

In some of the Lying data files, the time t_1 was found to be 0, which is a signal for the fact that the cow has already been resting at the start of the video. In this case, no

information about standing cows is available. Similarly, if the cow kept lying through the end of the video, the time t_2 is set to be -1. As mentioned in 3.1.5, during the night time, no marking of lying events was possible. To cover this case, t_2 is also set to -1 if a cow is resting at the start of the night. If a cow is found to be lying after the darkness is over, a new lying event is started at this time.

E.2 Housing data

One of the Housing data files generated during the image processing is shown below. It consists of 5 different segments. Every such segment starts with a type number $2, \ldots, 6$ that identifies the kind of information in this segment. The meaning of these numbers can be seen in Table 12. After that, the number of polygons for this object type is printed, as is the number of points per polygon. Every segment ends with the pointwise representation of each of the polygons. The nodes are stored using their x- and y-coordinates.

2	Visible part of the pen
3	Bedding area
4	Barriers and walls
5	Food trough
6	Water trough

Table 12: Meaning of the respective signalling values in housing files

```
2 1 8
  -0.204943     2.75468
  -0.179362     2.20451
  -0.0624029   -0.0493521
   0.00884045  -0.0290205
   0.73337      0.00840731
   1.13937     -0.0204204
   1.41418      0.00387694
   1.99681      1.81331
3 1 7
  -0.192388     2.72375
```

```
  -0.169466      2.18299
  -0.0583764    -0.0424812
   0.00925619   -0.0257272
   0.736722      0.00769238
   1.14432      -0.018075
   1.55466      -0.0056717
4 1 2
   0.189696      0.959446
   0.240176      0.0722235
5 0 0
6 1 8
   0.243845      0.0751758
   0.74122       0.0109375
   1.2322        0.0537562
   1.78703       1.67549
  -0.14707       2.17651
   5.30215e-005 1.02862
   0.192761      1.00004
   0.217316      0.404681
```

E.3 Standing data

In these data files, the positions of all visible standing cows were recorded. An entry was made every 30 seconds. Below, we can see two lines of one of the Standing files. The first two numbers in every line represent a counter and the time in seconds. After that, an integer indicating the number of visible standing cows is printed. Thereafter the positions of the standing cows are shown in the same format as for the Lying data.

```
0  0
  4  0.277938  0.303527 0.532531 0.282211
     0.256219  0.545228 0.454572 0.392473
     0.433045  0.624252 0.596778 0.48531
     0.0750349 1.27841  0.271132 1.49574
1 30
  5  0.301417  0.270732 0.570441 0.257352
```

```
0.486766  0.706365 0.553359 0.444493
0.261975  0.681475 0.500011 0.505321
0.273419  0.519185 0.490327 0.385477
0.0605419 1.78491  0.226548 1.49743
```

References

[1] Emile Aarts and Jan Korst. *Simulated Annealing and Boltzmann Machines*. Wiley, 1989.

[2] J. L. Albright and C. W. Arave. *The Behaviour of Cattle*. CAB International, Wallingford, 1997.

[3] Nino Boccara. *Modeling Complex Systems*. Springer, 2004.

[4] Černý, V. A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41 – 51, 1985.

[5] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37 – 46, 1960.

[6] Iain D. Couzin, Jens Krause, Richard James, Graeme D. Ruxton, and Nigel R. Franks. Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218:1 – 11, 2002.

[7] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, 1860.

[8] Gry Færevik, Kari Tjentland, Stine Løvik, Inger Lidse Anderson, and Knut Egil Bøe. Resting pattern and social behaviour of dairy calves housed in pens with different sized laying areas. *Applied Animal Behaviour Science*, 114:54 – 64, 2008.

[9] A. D. Fisher, G. A. Verkerk, C. J. Morrow, and L. R. Matthews. The effects of feed restrictions and lying deprivation on pituitary adrenal axis regulation in lactating cows. *Livestock Production Science*, 73:255 – 263, 2002.

[10] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulleting*, 76:378 – 382, 1971.

[11] Francis Galton. Typical laws of heredity. *Proceedings of the Royal Institute of Great Britain*, 8:282 – 301, 1877.

[12] Martin Gardner. The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, 223:120 – 123, October 1970.

[13] Arthur Gill. *Introduction to the Theory of Finite-State Machines*. McGraw-Hill, 1962.

[14] Stanley I. Grossmann. *Calculus*. Academic Press, Inc., third edition, 1984.

[15] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671 – 680, 1983.

[16] Primrose McConnell and Richard J. Soffe (ed.). *Primrose McConnel's The Agricultural Notebook*. Blackwell, 20th edition, 2003.

[17] Alistair Merrifield. *Models for Animal Group Movement, Using Classical and Statistical Approaches*. PhD thesis, University of Sydney, 2006. http://ses.library.usyd.edu.au/handle/2123/1132.

[18] John Von Neumann and Arthur W. Burks (ed.). *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[19] Lone Harder Nielsen, Lisbeth Mogensen, Christian Krohn, Jens Hindhede, and Jan Tind Sørensen. Resting and social behaviour of dairy heifers housed in slatted floor pens with different sized bedded lying areas. *Applied Animal Behaviour Science*, 54:307 – 316, 1997.

[20] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1993.

[21] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer, 1985.

[22] Fred L. Ramsey and Daniel W. Schafer. *The Statistical Sleuth: A Course in Methods of Data Analysis*. Duxbury, second edition, 2002.

[23] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[24] Steven H. Strogatz. From Kuramoto to Crawford: Exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143:1 – 20, 2000.

[25] Steven H. Strogatz. *SYNC: The Emerging Science of Spontaneous Order*. Penguin, 2004.

[26] Ivan E. Sutherland and Gary W. Hodgman. Reentrant polygon clipping. *Communications of the ACM*, 17:32 – 42, 1974.

[27] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, 1988.

[28] David Vanderbilt and Steven G. Louie. A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56:259 – 271, 1984.

[29] Michael Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002.