ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Implementing two mathematical swarm models in MATLAB and determining response times of different swarm configurations

Naterop Luca & Petrides Ioannis & Prasad Nishant

Zurich
December 2013

Contents

1 Abstract

In the following paper we present two different methods of simulating fish swarms. The first being a unit-vector based model which updates the velocity vector of each individual depending on the positions of its neighbours. The second model assumes mechanical forces acting on individuals depending on the positions of λ closest neighbours. The acceleration, velocity and position at each time step is calculated using Newton's Second Law of Motion.

In addition, a comparison of object-oriented and matrix-based implementation of one of the two models is done. The latter yielded faster processing speed although the model failed to accurately simulate the elaborate structure of a swarm.

2 Individual Contributions

The MATLAB coding and implementation was done by Nishant Prasad and Luca Naterop. The mathematical description was done by Ioannis Petrides. The work was evenly distributed among us.

3 Introduction

The primary motivation to investigate swarm behaviour arose from one of the author's diving experiences in south-east Asia, where he was mesmerized by the extreme smoothness of the fish swarm coordination mechanism. On delving a little further into the matter, it was observed that swarm behaviour is a widely researched phenomenon, with a gamut of potential real-world applications that are governed by decentralized organisation - cooperative robotics, vehicle navigation, optimization techniques, and telecommunication network design.

Most simple swarm models are based on the following three rules: (i) The highest priority is to avoid collisions with neighbours. (ii) Align direction of movement as the neighbour's direction. (iii) Remain close to neighbours. A literature survey on mathematical swarm models reveals two broad classifications:

1. **Force-based model**: This model allocates forces on every individual swarm in accordance with its position and distances from other neighbours for every time-step. This force then determines the acceleration, which is then twice integrated to arrive at the positional shift for a single time-step.

2. **Unit-vector-based model**: Instead of forces, in this model the positional shift of an individual is determined by taking into account the relative positions and direction vectors of the neighbours, such that the individual tends to align its own directional vector with that of its neighbours.

In this paper, particular cases of both models are picked out from existing literature, and simulated on MATLAB with some modifications to simplify the models. These are

then studied to understand whether the swarms conform to the afore-mentioned rules. Certain parameters are also studied and compared with the results presented by the original authors.

Additionally, for the position-based direction correcting model, the original authors describe different kinds of swarm formations and the possible varied responses each of those would have towards an external stimulus. In this paper, we have attempted to quantify that by placing an external stimulus (food) and comparing times that the different formations take to reach that stimulus (which bears a direct correlation to response time).

4 Description of Models

4.1 Unit-vector-based model

This model follows closely reference [2]. Here, point-like individuals are described by a 2-dimensional vector, denoted by $\vec{r}$, on a real Euclidean space $\mathbb{R}^2$. The velocity is given by the first derivative of the position and has a constant magnitude.

$$\vec{r_i} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{1}$$

$$\vec{v_i}(t) =\mid \vec{v} \mid \cdot\hat{\eta_i}(t) \tag{2}$$

At each time step, a member will change its unit vector according to the positions of the λ closest neighbours. The space around an individual is divided into three zones with each modifying the unit vector of the velocity in a different way (as described below).

4.1.1 Behavioural rules

The first region, called repulsion zone, corresponds to the "personal" space of the fish. Individuals within this region will try to avoid each other by swimming in opposite directions. The second is called the orientation zone, where members try to align their velocities and collectively move in the same direction. Next, is the attractive zone in which members swim towards each other and tend to cluster, while anyone beyond that point has no influence.

Let $\lambda_r, \lambda_o \& \lambda_r$ be the number of closest neighbours in zone of repulsion, orientation and attraction respectively. For $\lambda_r \neq 0$ the unit vector of an individual, at each time step τ, is given by:

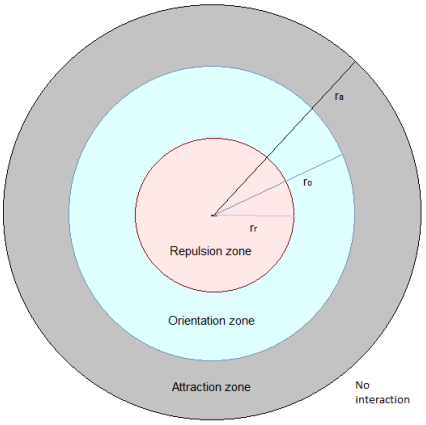

Figure 1: The four zones as described in text. (i)For $r < r_r$ zone of repulsion (highest priority). (ii) For $r_r < r < r_o$ zone of orientation. (iii) For $r_o < r < r_a$ attraction zone. (iv) For $r > r_o$ no interaction.

$$\hat{\eta}_i(t+\tau, \lambda_r \neq 0) = -\sum_{j=1}^{\lambda_r} \frac{\vec{r}_{ij}(t)}{\mid \vec{r}_{ij}(t) \mid} \tag{3}$$
$$where \quad \mid \vec{r}_{ij} \mid = \mid \vec{r}_j - \vec{r}_i \mid$$

The velocity vector points away of neighbours within this zone, thus collisions are avoided. This zone is given the highest priority; if and only if $\lambda_r = 0$, the remaining zones are enabled. The unit vector in this case is given by:

$$\hat{\eta}_i(t+\tau, \lambda_r = 0) = \frac{1}{2}\left(\sum_{j\neq i}^{\lambda_o} \hat{\eta}_{\mathbf{j}}(t) + \sum_{k\neq i}^{\lambda_a} \frac{\vec{r}_{ik}(t)}{\mid \vec{r}_{ik}(t) \mid}\right) \tag{4}$$

The first term corresponds to the orientation zone where the desired direction of two members is made parallel while the second term refers to the attraction zone where members move towards each other. The above equation contains a 1/2 factor which normalises the unit vector in the case of both zones having non zero neighbours. If no members are found in the vicinity of any zone, then the individual maintains constant velocity at each time step.

4.1.2 Further Criteria

In addition to the above, we constrain the angle by which a member can change its unit vector at each time step to a maximum of $\bar{\varphi}$ degrees. This condition was imposed in order to facilitate rigid body dynamics. Because we assumed point-like members, all information on the physical dimensions of the actual fish is lost; therefore, the unit vector is unconstrained to rotate at any angle. But in reality, conservation of angular momentum will limit the ability of the fish to turn.

$$\hat{\eta}_i(t+\tau) \cdot \hat{\eta}_i(t) = cos(\theta) \tag{5}$$
$$If \quad \theta > \bar{\varphi} \rightarrow \hat{\eta}_i(t+\tau) \cdot \hat{\eta}_i(t) = cos(\bar{\varphi})$$

If the above is not met, the angle of the desired direction at the next time step is rescaled to $\theta = \bar{\varphi}$. In this way, any un-physical behaviour, such as having a π-rotation of the velocity in a time step, is prevented.

Furthermore, a food source is implemented through a weighted vector along the relative positions of the food ($\vec{r}_f$) and an informed member ($\vec{r}_k$) which rotates the desired direction of the fish towards the source. A fraction of 1 : 5 individuals are randomly selected from the ensemble to play the role of informed member:

$$\hat{\eta}_k(t) = \frac{\hat{\eta}_k(t) + \omega\hat{\vec{r}}_{kf}(t)}{\mid \hat{\eta}_k(t) + \omega\hat{\vec{r}}_{kf}(t) \mid} \tag{6}$$
$$where: \quad \hat{\vec{r}}_{kf}(t) = \frac{\vec{r}_f(t) - \vec{r}_k(t)}{\mid \vec{r}_f(t) - \vec{r}_k(t) \mid}$$
$$k = \{1, 2, .., \gamma\} \quad with \quad \gamma : N = 1 : 5$$

The rest of the swarm is completely ignorant of the presence of food. They follow the behavioural rules described above, making no discrimination between informed and uninformed members.

4.1.3 Parametrisation

Initially we fix each parameter $r_r, r_o, r_a, \bar{\varphi}$ & $\mid \vec{v} \mid$ for the swarm. In order to generate dynamical configurations [2],each individual is assigned a value of the respective parameter but with a deviation from the actual value of the swarm. The spread of values is described by a Gaussian probability $\mathcal{N}(\mu, \sigma^2)$ with standard deviation σ and mean μ.

$$\mu_i \sim \mathcal{N}(\mu_0, \sigma^2) \tag{7}$$
$$\mu_0 = \{r_r, r_o, r_a, \varphi \ \& \mid \vec{v} \mid\}$$

In other words, members will have different values of a particular parameter (μ_i) , but the swarm as a whole will have mean value at the set parameter μ_0.

4.1.4 Analysis

To do an analysis on the collective behaviour of the swarm we first have to define expressions which quantitatively describe the system. Each parameter is chosen as shown in table below and the system is let to evolve. The collective behaviour of the swarm is measured by the following quantities.

Polarisation Polarisation is a measure of the degree of alignment between swarm members. Is defined as the weighted sum of the unit vectors.

$$P(t) = \frac{1}{N} \sum_{i=1}^{N} \hat{\eta}_i(t) \tag{8}$$

Note that if all members travel in parallel directions, polarisation approaches 1. In cases where the swarm as a whole stays stationary, the sum will average to zero.

Parameter		Units	Values
Population size	N	-	2^n where $n = \{4, 5, 6, 7, 8\}$
Number of influential neighbours	λ	-	up to N
Repulsion radius	r_r	m	1
Orientation radius	r_o	m	2-13
Attraction radius	r_a	m	15
Maximum turning angle	$\bar{\varphi}$	degrees	50
Speed	$\mid \vec{v_i} \mid$	ms^{-1}	2
Standard deviation	σ	-	0.05

Table 1: Values chosen for each parameter.

Angular momentum Angular momentum is a measure of the degree of rotation about the centre of mass. For a group of N members, the geometric centre can be found by:

$$\vec{r}_c(t) = \frac{1}{N} \sum_{i=1}^{N} \vec{r}_i(t) \tag{9}$$

The angular momentum $\vec{L}_i$ is found by the cross product of the position of the individual with respect to the geometric centre $\vec{r}_{ic}$ and the velocity vector. The swarm's angular momentum is simply the weighted sum of all individual angular momenta.

$$\vec{L}_i = \vec{r}_{ic} \times \vec{v}_i \quad \rightarrow \quad \vec{L} = \frac{1}{N} \sum_{i=1}^{N} \vec{r}_{ic}(t) \times \vec{v}_i(t) \tag{10}$$
$$where \quad \vec{r}_{ic}(t) = \vec{r}_c(t) - \vec{r}_i(t)$$

Expanse and pack density Expanse is a measure of how the swarm spreads in space with respect to the centre of the group. Is defined as the average distance from the centre of mass of a set of N independent positions.

$$\varepsilon(t) = \frac{1}{N} \sum_{i=1}^{N} \mid \vec{r}_{ic} \mid \tag{11}$$

Consider a large number of N. If we introduce one more member, it will only be able to stay in the outer regions of the swarm; since positions close to the centre are already occupied by others. Therefore, for increasing population size, expanse will be increased as a consequence of more and more terms being added to the above sum with each successive term being at least equal or greater than the previous. The scaling due to the population size can be eliminated by defining a new parameter, packing density which is the weighted value of expanse for a particular population number.

$$\rho(t) = \frac{1}{N} \cdot \varepsilon(t) \tag{12}$$

Response time When food is introduced we need to have a measurement which characterises how quickly the swarm responds to the external stimulus. This can simply be the time duration needed for the first member to reach the food source.

4.2 Force-based Model

The first model is based on the paper done by Vincida [3]. We assume simple mechanical forces acting on point-like members which are described by a 2-dimensional vector, denoted by bold $\vec{r}$, on a real Euclidean space $\mathbb{R}^2$. The velocity and acceleration are given by the first and second derivative respectively.

$$\vec{r}_i = \begin{pmatrix} y_i \\ x_i \end{pmatrix} \tag{13}$$

$$\vec{v}_i = \dot{\vec{r}}_i$$

$$\vec{a}_i = \ddot{\vec{r}}_i \tag{14}$$

An individual member will interact with the world and change its position vector with time according to Newtons second law of motion. To simulate a fish swarm in the sea bottom, we assume 2 types of forces; a member-member interaction force ($F^{Int.}$) which corresponds to the influence of the swarm on the individual and a random force ($F^{rand.}$). For the i'th individual the resultant force and thus the equation motion is given by:

$$\Sigma\vec{F}_i = \vec{F}_i^{Int.} + \vec{F}_i^{rand.} = m\ddot{\vec{r}}_i \tag{15}$$

4.2.1 Member-member Interaction Force

The interaction force must simulate the desire of long separated individuals to come together while maintaining their personal space by repulsing anyone close by. Similarly to the previous model, the force takes values depending on how far the neighbour is located. The force on the i-th individual (located at $\vec{r}_i$) due to the j-th neighbour (at position $\vec{r}_j$) is given by

$$\vec{F}_{ij}^{Int.} = \begin{cases} (g_r r_{ij} - F_{max})\hat{n}_{ij} & \text{if } r_{ij} < r_r \\ 0 & \text{if } r_r < r_{ij} < r_0 \\ g_a(r_{ij} - r_a)\hat{n}_{ij} & \text{if } r_0 < r_{ij} < r_a \\ F_{max} & \text{if } r_a < r_{ij} < R \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$where \quad g_r = \frac{F_{max}}{r_a} \quad r_{ij} = \mid \vec{r}_j - \vec{r}_i \mid$$
$$g_a = \frac{F_{max}}{r_a - r_0} \quad \hat{n}_{ij} = \frac{\vec{r}_j - \vec{r}_i}{|\vec{r}_j - \vec{r}_i|}$$

Where r_r, r_0, r_a are the repulsion, equilibrium and attraction distance and R is the maximum distance of influence.

At zero separation the force has the minimum value $-F_{max}$ and increases linearly with rate g_r until it reaches zero. For $r_r < r_{ij} < r_0$, the force is zero, corresponding to the equilibrium region required to produce stable groups [1][4]. After that, the force is positive and increases linearly with rate g_a until it reaches the maximum value of $+F_{max}$ at $r_{ij} = r_a$. For $r_a < r_{ij} < R$ the force is constant while anything beyond that has no influence.

The total force acting on the i-th individual can be calculated from the vector sum of the closest λ neighbours.

$$\vec{F}_i^{Int.} = \sum_{j \neq i}^{\lambda} \vec{F}_{ij}^{Int.} \tag{17}$$

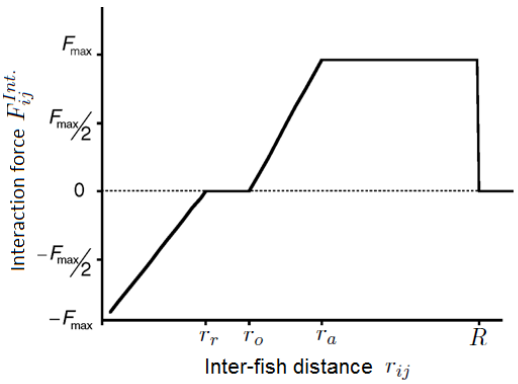

Figure 2: A graphical representation of the interaction force as a function of the distance between two members. The force ranges between $\pm F_{max}$ depending on the distance between the individuals.

4.2.2 Random Force

The random force provides a "freedom" to the individual to swim in any direction, provided that the influence of others is not strong. It reflects our ignorance on the intricate processes (i.e. decision making depending on a particular sensory input or other cognitive features) that a fish will go through in order to figure out its direction. It is defined as a random normal variable with zero mean and standard deviation σ.

$$\vec{F}_i^{Rand.} \sim \mathcal{N}(\vec{0}, (F_{max}/6)^2) \tag{18}$$

For small or zero values of the interaction force $F^{Int.}$, the random force $F^{Rand.}$ dominates; effectively simulating a free roaming fish. In contrast, when $F^{Int.}$ is large then $F^{Rand.}$will be of less importance although some minor random effect will still present in the total force $\Sigma\vec{F}_i$.

The values used for the different variables are shown it Table2.

4.3 MATLAB Implementation

Agent-based modelling was used in all cases of implementation, wherein each individual's position is updated with every time-step. Individuals were treated as point masses and graphically represented by asterisks ("*"). The initial distribution was always random

Parameter		Units	Values
Population size	N	-	100
Number of influential neighbours	λ	-	8
Maximum Force	F_{max}	ms^{-2}	100
Repulsion distance	r_r	m	1
Equilibrium distance	r_0	m	1-15
Attraction distance	r_a	m	20
Maximum distance of influence	R	m	30
Repulsion gradient	g_r	s^{-2}	1
Attraction gradient	g_a	s^{-2}	5

Table 2: Showing the chosen value of each parameter.

within an area of 3N, where N is the population size; defining the initial positional distribution in this way ensures that the initial packing density of the individuals remains more or less constant across all populations. To ensure reproducibility of our computation, the MATLAB internal random number stream was set to a seed of 10 for every simulation.

Unit-vector based model This model was implemented using matrices. The matrix-based operation takes advantage of the various matrix operations in MATLAB, to enable just one line of code for performing operations like computation of distance of one individual from all others, and updating all individual's position and direction at once.

Forces-based model Two different, completely independent implementations of this model were done:

- Object-based: Each fish was treated as an object with properties such as mass, colour, position, velocity and acceleration. For each time-step, a loop goes through every individual in the swarm after which its distances from a specified number of closest neighbours is computed. Based on these distances, the vector sum of forces acting on an individual is computed, from where the acceleration is derived. The acceleration is then integrated twice to arrive at the new position of the individual.
- Matrix-based: The matrix-based implementation is similar to that in the earlier model, but accounting for a different way of re-positioning the individuals (using forces).

5 Results and discussion

5.1 Force-based Model

To gauge whether this model is an accurate one for swarms, the main criteria is that the values of expansion should stabilize after a point of time, which should be expected in

the absence of external forces. Hence, values of expansion were plotted against time for two cases:

1. Varying population N (16,32,64,128), keeping number of influential neighbours constant, $\lambda(=8)$
2. Varying number of influential neighbours, λ (4,8,16,32) for constant population of N (=35)

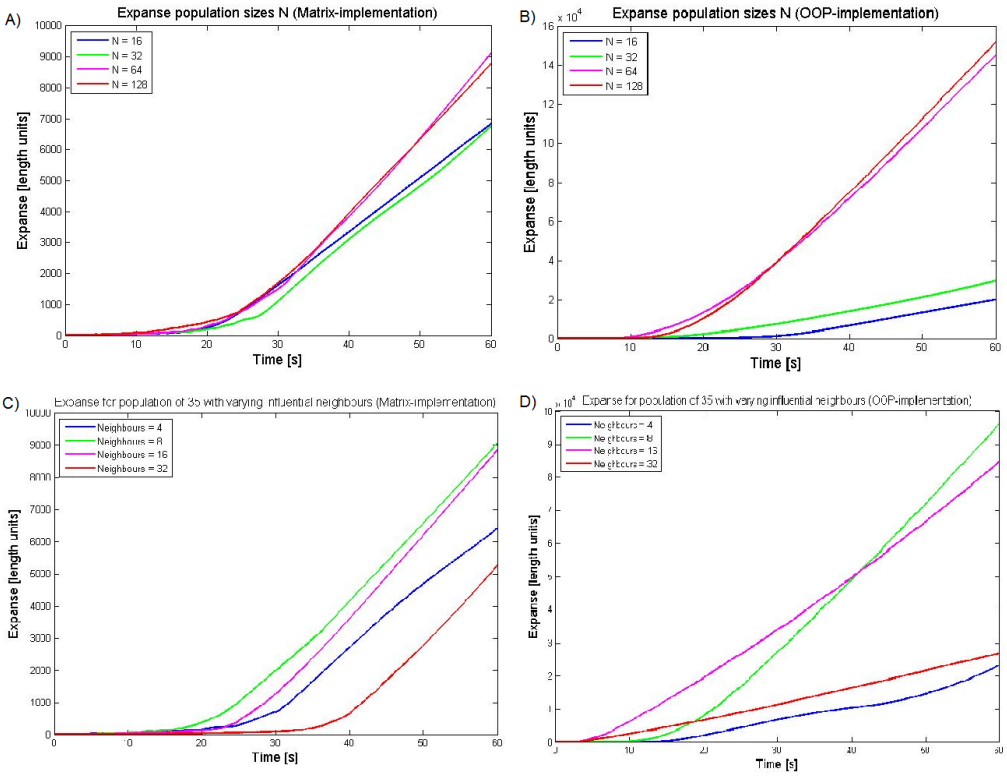

Figure 3: Results for expanse from Matrix-based (left) and object-orientated (right) implementations with varying neighbours (bottom) and population size (top).

Figure 3 shows the results obtained, from both sets of implementations. As can be clearly seen, the expanse value of the swarm continuously increases with time (after a certain point) for all cases. This indicates that after a certain point of time, the individuals escape the influential zones, and then continue with their same velocities under no influence of any other forces; which is not swarm behaviour.

These results are very different from what was obtained in reference [3], which could be attributed to some simplifications we made:

i In ref.[3], the individuals were modelled as cylinders, whereas in this paper, they are treated as point masses.

ii In ref.[3], the forces between individuals vary with respect to their relative orientation with each other, that is there are different parametric values for end-end, side-side and end-side orientations. However, in the present case it does not make a distinction between the three cases, it just uses the parametric values for the end-side case (which has intermediate values compared to the other two cases).

iii The present model does not take into account viewing angles, as against ref.[3]

iv The present model uses a two dimensional space, as against a three dimensional space in [3].

Of the above, it is probable that (ii) has the most significant effect on the result since it takes into account orientation while determining forces and influential zones. It is, however, difficult to gauge the effect of (iii) and conclude whether the absence of a viewing angle would increase or decrease the probability of the population to stay as a swarm.

Another possible reason for failure of this model could be that the appropriate scaling of units and time-steps was not accurately done. This could result in very high velocities being generated that would be capable of re-positioning interior individuals to outside the influential zones within a single time-step. Possible ways to mitigate this could be to reduce the time-step, and add a drag force to limit the maximum velocity an individuals can achieve.

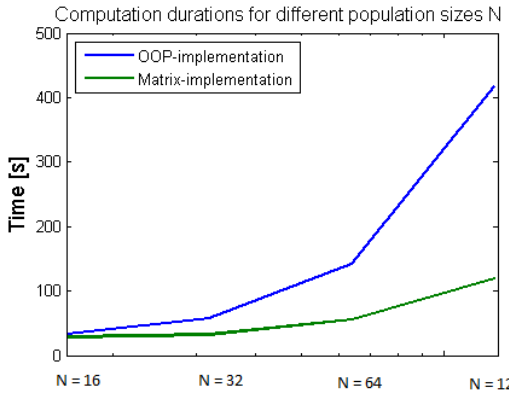

Figure 4: Computation durations for matrix-based and the OOP-implementation.

The difference between expansion values for the two different implementation approaches can be explained by the fact that a random number generator is used multiple times in the object-oriented approach (effectively due to the structure of the algorithm), in contrast with the matrix-based model. This results in non-uniformity of initial conditions between the two implementation approaches.

Additionally, a comparison evaluating the processing times for both the methods is also depicted in figure 4 , which clearly indicates that the matrix implementation is much faster than the object-based implementation; this implies that for future implementations of similar models in MATLAB, a matrix-based approach is definitely recommended over an OOP based approach.

5.2 Unit-vector-Based Model

Expanse The first test of the model's suitability works is to check whether the expanse values stabilize after a certain point. This was done by varying the population size and keeping all other parameters constant.

Configuration	Zone of orientation	Average angular momentum	Average polarisation	Time to food[s]
Swarm	2	0.52	0.09	158
Torus	5	2.71	0.22	113
Dynamic Parallel	9	0.14	0.95	83
Highly Parallel	13	0.04	0.99	71

Table 3: Measurements for different configurations

As can be seen from figure 5 A), the value for expanse does stabilize for every case, indicating that this model can be used as a basis to simulate swarms. Also, as can be predicted, the expanse values increase with population since more area is required to house all the individuals.

Packing Density Packing density was also analysed for this model, which is simply the division of expansion by the population, to arrive at an estimate of the average density of individuals in a swarm. As can be seen from figure 5 B), the magnitude of the variation for this parameter across different populations is much smaller than that for expanse. Interestingly, the packing density for larger populations is lesser than that of smaller populations, indicating individuals in larger groups are more tightly spaced. This could be a potential shortcoming of this model, since this trend remains to be physically verified.

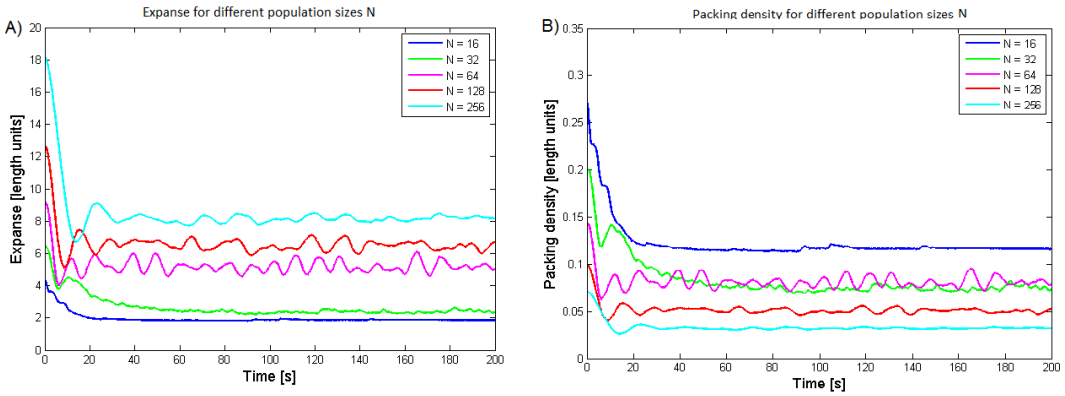

Figure 5: A) Expanse and B) packing density for different population sizes

5.3 Different Swarm Configurations

Reference [2] predicts different swarm configurations on the basis of relative values of average polarizations and angular momenta. In the present case, a particular case of each is analysed for population size of 64, and the parameters attained are displayed

in table 3. The average polarization and angular momentum are calculated after the swarms stabilize (averaged between 100 and 150 seconds).

In order to gauge response times of different configurations, a new simulation was carried out with the introduction of an external stimulus (food) such that 1:5 of the population had knowledge of the food and its location. The time taken for the different configurations to reach the food was then observed, with 10 repetitions for each swarm configuration.

The results are as depicted in figure 7. In reference [2] qualitatively discussed the possible response times of different configurations towards external stimuli, but here that discussion has been accurately quantified. The predictions made in [2] are thus verified. Brief notes on each configuration:

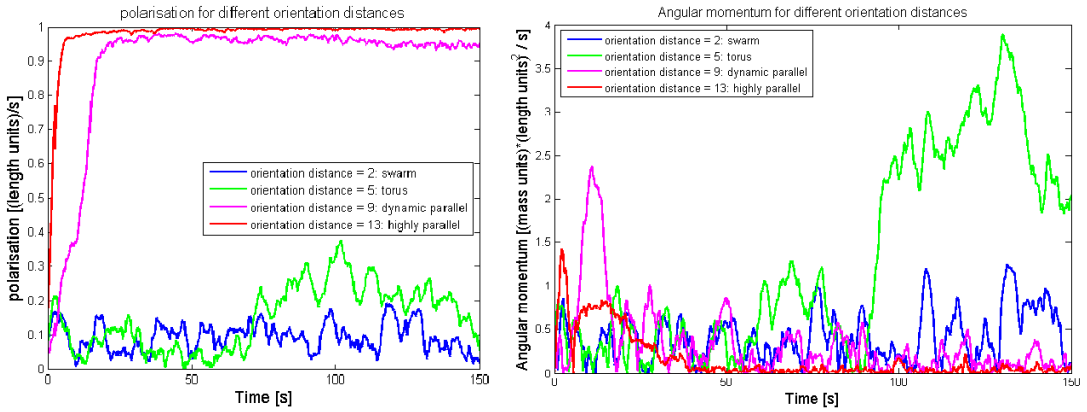

Figure 6: Polarisation and angular momentum for different swarm configurations

Swarm : This configuration has the lowest polarization and relatively low angular momentum. This is predicted since the zone of orientation is the smallest in this case. The low polarization will result in much slower responses to stimuli, and thus this type of configuration is not favourable for fish that are prey to many other fish in a water body. It is predicted that there will not be many occasions where this configuration will be observed, since it does not seem to hold many advantages over other configurations. However, it is advantageous in cases where the swarm is required to be quite stationary, like when they surround their prey.

Torus : This exhibits the highest angular momentum and relatively low polarization. The polarization is higher than the "swarm" configuration since there is some degree of local polarization. Swarms of the torus configuration continuously revolve in a random direction, but the relative translational movements are quite small. Thus, this configuration has the advantage that it is more or less stationary as a group (attribute shared with the "swarm configuration") but also has a faster response time to stimuli.

Dynamic Parallel : The polarization is very high in this configuration, owing to the relatively large zone of orientation. Thus the response times are much shorter than the former two configurations, which is advantageous. However, the translational motion of this configuration is also much more than the former two configurations, which is disadvantageous in the case of the presence of predators, where it is better that the fish are more stationary to avoid being seen.

Highly Parallel : This is an extreme case of very high polarization and lowest angular momentum. It is predicted that such configurations only occur when the swarm is being chased by predators, in very extreme cases.

According to ref[2], swarms mostly exhibit torus or dynamic parallel configurations.

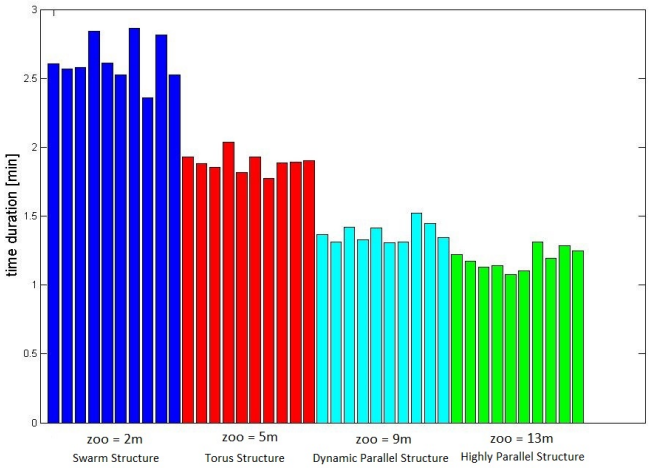

Figure 7: Time duration to reach food for different configurations(10 repetition for each).

6 Conclusion

The present implementation of the simplified forces-based model failed to depict a swarm model. The biggest reason for this failure is the possible error in scalability of the units and parametric values; in the present case, the computed velocities seem very high such that individuals are easily able to escape the influential zones in very few time steps. Possible improvements while next implementing this model could be to significantly lower parametric values and time-step, as well as add drag forces.

The unit-vector based model delivered results similar to what was predicted in [2]. Different configurations of swarms were obtained with exact parameters, and response times for each of these configurations were also determined, followed by a qualitative discussion of different situations where each of these configurations could be suitable.

For the forces model, the same implementation was carried out using object oriented programming and matrices in MATLAB, and it can safely be concluded that MATLAB is

more optimized to use matrices, which is therefore a recommendation for further swarm implementations.

Additionally, although the governing models of the forces-based and unit-vector based models seem quite different, it is predicted that with suitable adjustments to the parameters and time-steps in the forces model, the swarm behaviour could approach that obtained by the unit-vector based model. Thus, it would not be accurate to write off the viability of a forces-based model based on the results obtained in this paper.

7 References

1. T.I. Zohdi (2009), "Mechanistic modeling of swarms", Comput. Methods Appl. Mech. Engrg. 198, 2039-2051

2. I.D. Couzin, J. Krause, R. James, G. R. Ruxton and N.G. Franks (2002). "Collective memory and spatial sorting in Animal Groups", Jour. Theor. Biol. 218, 1-11

3. Steven V. Viscida,b, Julia K. Parrisha,b, Daniel Grunbaumc (2005). "The effect of population size and number of influential neighbours on the emergent properties of fish schools", Ecological Modelling 183, 347-363

4. Gueron, S., Levin, S. (1995). "The dynamics of group formation", Math. Biosci. 128, 243-264.

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Naterop Luca

Petrides Ioannis, Nishant Prasad