UPPSALA UNIVERSITY
DEPARTMENT OF MATHEMATICS
Fiona Skerman

Resit version
Summer 2020

Lab 5: Self-propelled particles.

The deadline for this sheet is midnight Monday 31st August.

Please submit hand-ins on Studentportalen. All code should be included. Please feel free to submit videos illustrating your results where appropriate, via Studentportalen or uploaded elsewhere. You may work in groups of size 1-4, and only one group member needs to submit the assignment. State clearly the members of the group.

5. Flocks and Predators

This assignment investigates a version of the Vicsek alignment model where interactions are only with the n nearest neighbours. As a starting point you may use the matlab or the python implementations provided on the course webpage (note these are implementations of the Vicsek alignment model with a fixed radius of alignment). You may also write or find another implementation. Try running the model with different parameter values.

1. Implement a new version of the model, where instead of a radius of interaction, the number of neighbours is fixed. Specifically, on each time step, each particle calculates the average direction of its n nearest neighbours irrespective of how far away they are. Use a total number of particles $N = 40$ and some angular noise (e.g. $e = 0.5$ in matlab or $\eta = 0.5/2\pi$ in python implementation). Run this for $J = 200$ time steps, at each time calculating the global alignment as defined in the lecture. Plot the alignment as a function of time for $n = 4$. **(3 points)**

2. Add a very weak force that pulls the particles toward the centre of mass of *all* particles. The parameter value should be set so that the particles stay together but still move like a flock. With this force active, run this simulation for various values of noise e and number of neighbours n. Make a two dimensional heat map of how the alignment at step J ($= 200$ say) depends on these two parameters. **(4 points)**

3. Give 20 of the particles a large n and give another 20 particles a small n in the same simulation. Make up your own rules of motion for an extra particle that acts as a "predator". This predator should try to get close to your other particles, eating them if they come within a

small distance (Tip: in matlab making a particle's co-ordinates and direction equal NaN [Not a Number] will make it disappear). It may be useful to simulate with different colours for the two types of prey and for the predator. Find out whether the particles with large n or or the ones with small n are consumed faster and explain why. **(4 points)**