ACM 11: Bird Flocking Project

This is one possible topic for the ACM11 MATLAB project. Difficulty rating: easy-medium for no boundary conditions, medium-hard for periodic boundary conditions. 100 points. 10/23/08.

The inspiration for this project was the October 13 ACM colloquium by Eitan Tadmor titled "From Particle to Kinetic and Hydrodynamic Descriptions of Flocking". The actual setup of this project is based off the seminal flocking article *Novel type of phase transition in a system of self-driven particles* by Vicsek et al., Phys. Rev. Lett. 75(6):1995. The paper is highly readable.

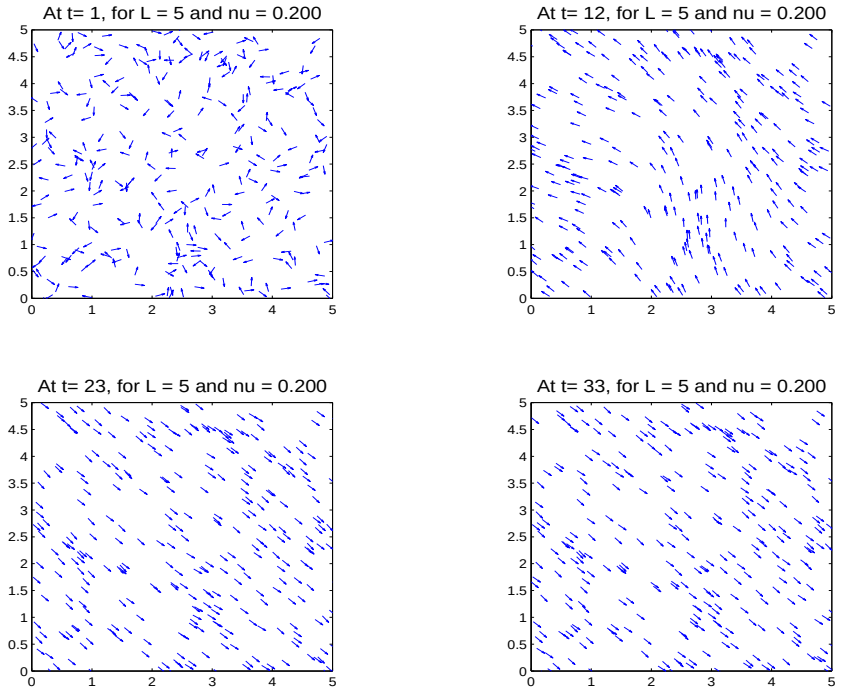

Figure 1: Flocking behavior at relatively high density ($L = 5$). The plot shows position of the particles, with the arrows representing velocity.

Vicsek and coworkers model particles (or birds) as moving with constant speed, but with the direction of the velocity influenced by the closest surrounding particles. Everything is in 2D. The domain is a square box of length L per side, and periodic boundary conditions are assumed. This means, if the box is located at $[0, L]$ x $[0, L]$, then if a particle ends up in position $(L + .2, .5)$, the simulation will reset its position back to $(.2, .5)$ (assuming $L > .5$). It also means that, when looking for the closes surrounding particles (the "nearest-neighbors"), the search must be done taking the periodicity into account.

The formula for the update of positions is simple (in differential equation terms, this is just "forward-Euler"):

$$x_i(t + 1) = x_i(t) + v_i(t) \cdot \Delta t \quad \text{for} \quad i = 1, \ldots, N \tag{1}$$

where i represents a single particle and N is the total number of particles.

The velocity update is more interesting. We define the neighbor list, for particle i, as follows:

$$\text{neighborList}_i = \{j : |x_j - x_i| < r\} \tag{2}$$

for some radius $r < L$, and $|\cdot|$ meaning the usual Euclidean distance. This neighbor list must be updated every time step, and it must take the periodic boundary conditions into account. Note that $i \in \text{neighborList}_i$. In actual large-scale scientific simulations of this type (called "molecular-dynamics" simulations), there are often two neighbor lists, one with radius r and one with a larger

radius $\hat{r} > r$. The small neighbor list is created only from searching the larger neighbor list (instead of searching from all N particles, thus we save quite a bit of computation), and the large neighbor list is updated periodically, but not at every time-step. It is not necessary to do this for this project.

For each velocity $v_j(t)$, we have a corresponding angle $\theta_j(t)$. Then define the angle $\bar{\theta}_i$ as the average of the angles over the neighbor-list of i:

$$\bar{\theta}_i = \frac{1}{|\text{neighborList}_i|} \sum_{j \in \text{neighborList}_i} \theta_j(t) \tag{3}$$

In this model, we weight all particles in the neighbor list equally. A more complicated model might weight particles inversely proportional to their distance.

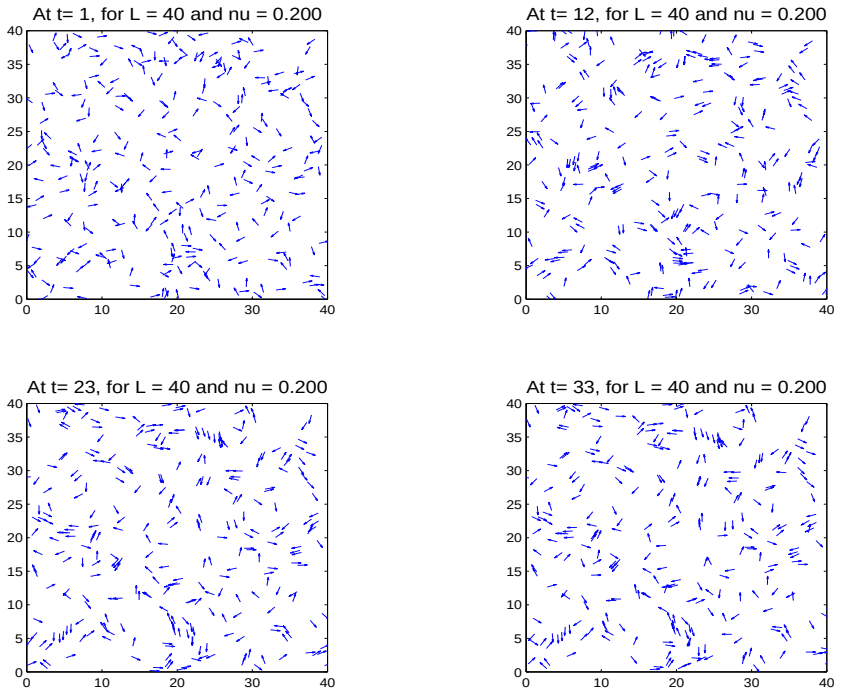

Figure 2: Behavior is less like "flocking" at relatively low density ($L = 40$) for the same amount of randomness η.

The velocity is then updated as follows:

$$v_i(t+1) = \nu \cdot \big(\cos(\bar{\theta}_i + \Delta\theta), \sin(\bar{\theta}_i + \Delta\theta)\big) \quad \text{for} \quad i = 1, \ldots, N \tag{4}$$

where $\Delta\theta$ is a uniformly distributed random variable in the interval $[-\eta/2, \eta/2]$ (with $\eta < 2\pi$). If η is large, then there is more randomness, and the system has a higher "temperature." If η is small, then we expect to find flocking behavior.

Initialization: the positions $x_i(0)$ should be uniformly distributed in the box. The velocities should be randomly oriented, with constant magnitude ν. Be careful to make sure the angles are evenly distributed in $[0, 2\pi]$, not just in $[0, \pi/2]$.

There are two big difficulties in the simulation. The first is a choice of datastructure to hold the positions and velocities. There are several options. One first has to decide whether to record all past positions and velocities, or just the most recent set (for this simulation, either choice works; for large-scale simulations, the former method is often impractical). The variables can be a series of 2D arrays (for example, stored in cells), or a 3D array. Perhaps the most elegant method is to store the 2D coordinates of position and velocity as single complex numbers.

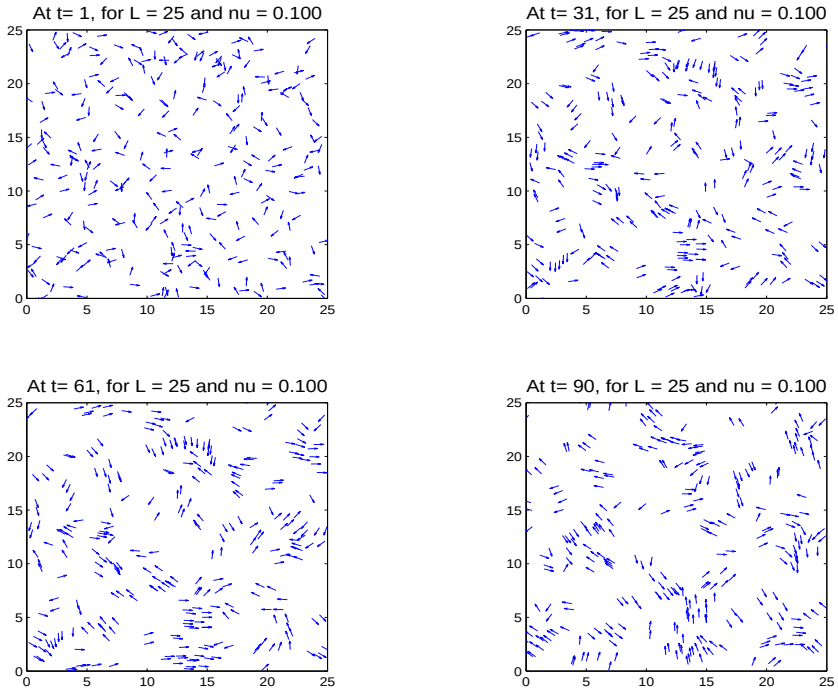

Figure 3: Groups of particles form for small randomness η but low density. This simulation, for 90 time steps, took 10 seconds on a modern computer.

The second difficulty is taking the periodic boundary conditions into account. If this is causing a lot of problems, it is OK to complete the project without using periodic boundary conditions (e.g. the particles start out in the box, but are then allowed to move anywhere in $\mathbb{R}^2$). However, this will result in very different behavior. The boundary conditions need to be accounted for at two locations in the code: one, when updating the positions, and two, when finding the neighbor list. The second case is the trickier one, and there are many ways to implement it, some of which are quick, and some of which are quite slow. It is OK to use a slow method, but of course quick methods are preferred. The `mod` command is useful. Please ask the instructor or TA if you would like help on this.

Parameters		
N	300	Number of particles/birds
L	vary this	Length of box
ρ	N/L^2	Density of particles/birds
η	.2	Randomness, similar to temperature; you may vary this if you like
r	1	size of nearest-neighbor circle
Δt	1	time step
ν	0.03	speed of particles/birds
T	10 to 100	total simulation time

For the project, use the parameters specified in the table, and simulate the particles. You may adjust the total time T to an appropriate value. Create two plots, similar to Figures 1 and 2 (which are similar to Figure 1 in the Vicsek paper), using the `quiver` command. One plot should show flocking behavior, and the other plot should show random-behavior. We expect flocking behavior when the density ρ is high (i.e. when L is small) and the randomness η is small. As a bonus, recreate Figures 2 and 3 in the Vicsek paper.

Your project should consist of a file called `main.m` that produces all the figures; you may include subfunctions in this file if you like. Put the file in a folder called `Firstname_Lastname_proj_flocking`, where you replace `Firstname` and `Lastname` with your first and last names, respectively, and upload this via ftp.

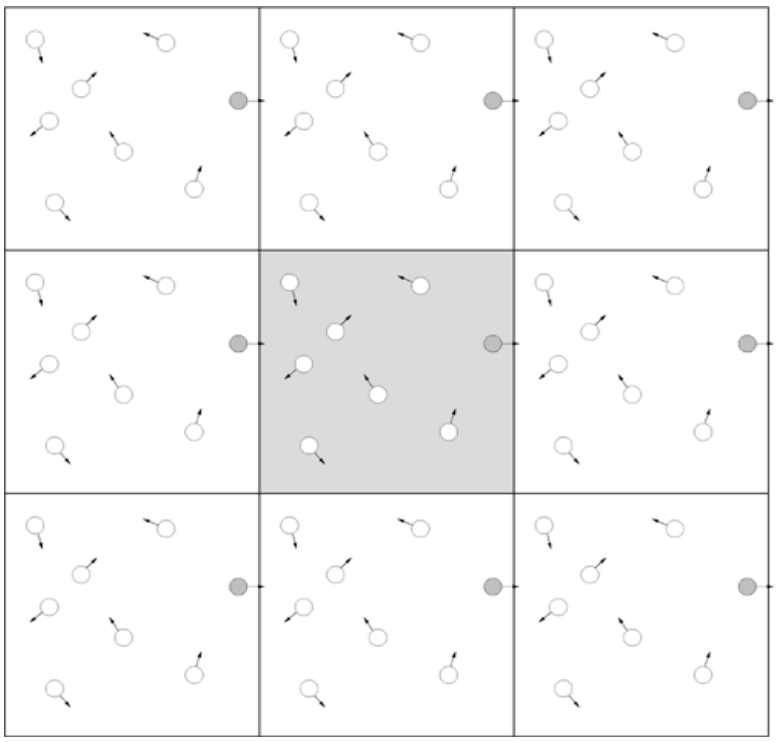

Figure 4: Graphical depiction of periodic boundary conditions. Periodic boundary conditions are useful for keeping a constant density but without introducing boundary effects. Image courtesy of `http://www.pumma.nl`.