SYSEN 6000: Homework for Lecture 5/5/2017
Lecturer: Kirstin H. Petersen, Asst. Prof. ECE, Cornell University

Self-organized Flocking

This homework will illustrate swarm intelligence through a flocking example. As discussed in the lecture, simple rules for flocking was first introduced by Reynolds in 1987. Briefly put, you can achieve coherent emergent behavior (flocking) of a swarm, when each agent is 1) *attracted* to other agents, 2) *repulsed* when too close to other agents, and 3) performing *velocity alignment* with other agents. In other words, the behavior is not something we program directly, instead it is an emergent property of many locally interacting individuals. We will examine this behavior in 2D; the swarm will be composed of N agents; each agent will be defined by its position (p) and velocity (v).

First, generate a randomized initial distribution of agents, with a Gaussian spread about the origin and a Gaussian spread of velocities. The size of the spread is determined by P and V:

```
p = P*randn(2,N);
v = V*randn(2,N);
```

Second, create the loop that iterates the motion of the flock. In real life each agent would update asynchronously, but for simplicity we use synchronized agents here. For each iteration of the loop, position and velocity must be updated according to the laws of attraction, repulsion, and velocity alignment. We will compute the distance and velocity mismatch between agents, *m* and *n*, and update accordingly:

```
r = p(:,m)-p(:,n);                          %Vector from agent n to m
rmag=sqrt(r(1)^2+r(2)^2);                   %Distance between agents n and m
v1(:,n) = v1(:,n) + c1*r;                   %Attraction
v2(:,n) = v2(:,n) - c2*r/(rmag^2);          %Repulsion (non-linear scaling)
v3 = [sum(v(1,:))/N; sum(v(2,:))/N]*c3;      %Align average heading
```

The scaling factors c1, c2, and c3 determine how much attraction, repulsion, and velocity alignment can influence the current velocity. *vlimit* determines the maximum velocity of the voids. A fourth parameter can be added to generate some randomness for more realistic movement: v4(:,n) = c4*randn(2,1).

Third, the actual velocity is updated as the sum of v1, v2, v3, and v4, and the resulting change in position given the time step, delta, is calculated:

```
v(:,n) = v1(:,n)+v2(:,n)+v3+v4(:,n);        %New velocity of agent n
p(:,n) = p(:,n) + v(:,n) *delta;            %New position of agent n
```

Reiterate for all agents for each time step, delta.

Template code can be found here: https://sites.coecis.cornell.edu/petersen/files/2017/05/Flocking-y1og6d.m Feel free to modify the code in any way you prefer.

Homework Question 1

- Explain how each scaling factor (c1, c2, c3, c4) affects the flocking behavior. Back up your claims through simulations, graphs, and/or videos.
- Explain intuitively, how the algorithm scales with the number of agents? What can we do to make it scale better?

Homework Question 2

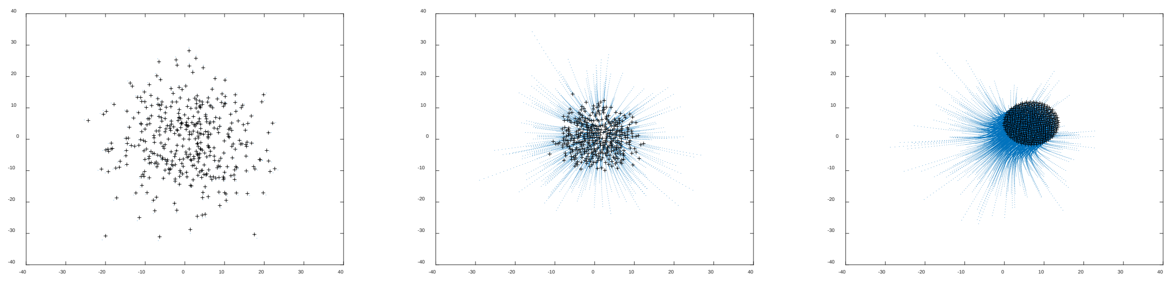
- Modify your code to make the algorithm more scalable. There is two standard ways of doing this: 1) Each agent only takes its K nearest neighbors into consideration. 2) More realistically, the sensory limitations will only enable the agent to take neighbors within a radius of R into consideration. Implement one or both of these options, try different values of K and/or R. How does this affect the behavior of the swarm? Back up your claims through simulations, graphs, and/or videos.

Homework Question 3 – Using the swarm from question 2, solve at least one of the following challenges:

- Introduce obstacles in your environment, and discuss the emergent behavior. With a high attraction scaling factor the swarm should tend to stay as one, with a lower scaling factor it may split up to circumvent the obstacle. *Hint: The easiest way to add an obstacle is to add a static agent with repulsion – all of this may be easier to simulate if you add a barrier of repulsion to contain the swarm within a limited set of coordinates.
- Introduce informed agents into the swarm; i.e. agents who are attracted towards pre-specified coordinates. How does the performance of the swarm change with the percentage of informed agents? Try introducing agents with different opinions and differently weighted opinions, and change the ratio of informed to uninformed agents. You should find that uninformed agents help promote democratic consensus.

Hand-ins

Upload a zip-file to Cornell Dropbox with your answers (in pdf-form), include all relevant Matlab codes, figures, and videos. Keep the code tidy and make sure it does what you say it does. Send the link, as well as questions or concerns to kirstin@cornell.edu.



Simulation at time step 0, 25, and 150, performed with N = 400, K=N, delta = 0.001, c1=c3=c4=0.1, c2=5.