Individual-based models

So far, you have already studied how localized interactions can affect the spatial structure of a population or ecosystem. The setting that is implicit in cellular automata fits best to sessile organisms that occupy a specific space and interact only with their neighbors. But what if we want to model, for instance, the movement of fish in a school, or the movement of a limited number of predators in a landscape (think of a small number of puma in an American park)? For that, we can't use a model that puts the individuals on a lattice of equal-sized cells, but we should explicitly model the location of the animals in a two-dimensional or three-dimensional space.

Agent-based modeling

In this chapter, we will describe the general class of techniques that is called "individual-based modeling" or "agent-based modeling". The term agent is used throughout science to indicate an autonomous unit that can make decisions to achieving a particular goal, and that adapts these decisions to chances in the environment or situation that it experiences. In ecology, an agent is often an individual organism. Each individual organism is described by a set of characteristics, for instance its coordinates in space, its size, or whether it is hungry or not (a state).

A cellular automaton can be used to model sessile individuals such as plants on a square lattice. Cellular automata are very simple to model, but have the disadvantage that the organisms cannot simply move around, and can only be described by a limited number of states. In modeling a school of fish, a cellular automaton is obviously too limited. We need to develop a program in which, for instance, the changes in the organism's physical location, its growth and its decisions are explicitly modeled. Below, we will develop a simple individual-based model in MATLAB, using self-organized patch formation of mussels as an example.

Figure 7.1: String patterns in mussel beds.

7.1 Application: Self-organized mussel patterns

In intertidal environments, many organisms attach strongly to a surface, for instance a rock, to avoid being dislodged by waves and water flow. Mussels in the rocky intertidal, for instance, attach to the rock surface with byssal threads. On sandy sediment, no firm substrate is available, so mussels attach to each other, forming clumps and strings of mussels (see Figure 7.1). These strings and clumps can form self-organized patterns that look very similar to the patterns formed in arid ecosystems (e.g., leopard bush).

A laboratory study on mussel aggregation dynamics revealed that mussels can aggregated to produce these spatial patterns on very short timescales, e.g., within a day.

Exercise 7.1 Read the paper "Experimental evidence for spatial self-organization and its emergent effects in mussel beds" by Van de Koppel et al., attached to this syllabus. Also observe the aggregation and pattern formation by the mussels in the video that was published as an appendix of the paper and that has been uploaded on Blackboard. Describe what you observe. Would you call this self-organization? Look at the definition of self-organization in the above paper, and the one in this syllabus. Justify your answer.

When the relation between the movement speed of a mussel and its number of neighbors was investigated, a complex behavior was observed. The behavior can be summarized as follows:

- Mussels move with high speed when they are alone.
- When the number of mussels in the direct neighborhoods increases, mussel movement speed decreases.
- When the number of mussels in the total aggregate, at a scale of +- 10 cm, increases too much, mussels again start to move faster, moving out of clusters that are too large.

In this assignment, you will model the movement of mussels to test whether the above description of individual movement is sufficient to recreate the patterns that you observed in the paper. For this, you will have to describe mussel movement, in terms of its step size and direction per time unit, as a stochastic process, i.e., a random walk. The step size is not entirely random, of course, but is a function of the local density of mussels. Inspection of the data revealed that the mussels do a random walk that is best described by an exponential distribution of step sizes ($F(x) = 1/\beta \exp(-x/\beta)$, where x is the step size and β a scaling parameter):

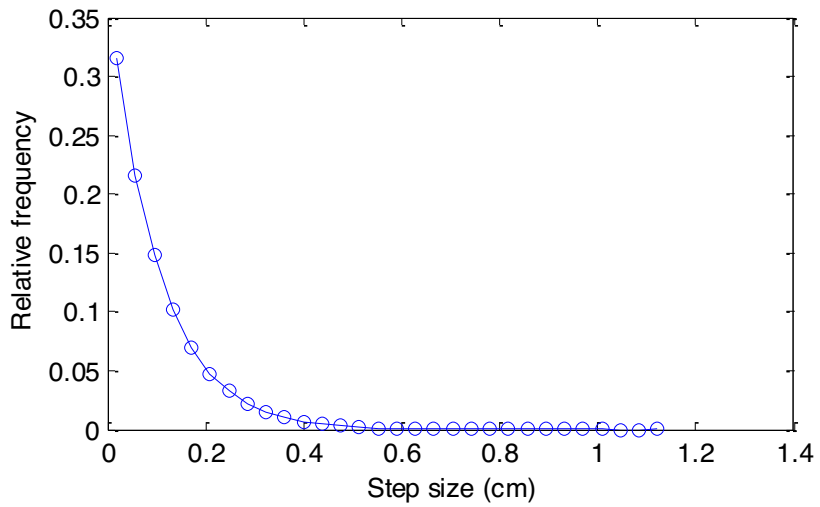

Figure 7.2

Figure 7.2 gives the frequency distribution for a particular combination of V_1 and V_2 (0 in this case), and a β of 0.1. The parameter β, however, is related to the local neighborhood density V_1 and the cluster density V_2 as follows:

$$\beta = \frac{1}{P_1 V_1 + P_2 V_2 + P_3}$$

Here, V_1 and V_2 are the number of mussels per square centimeter, P_1 and P_2 are conversion constants (in square cm per number) and P_3 is a dimensionless number that sets mussel movement when a mussel is entirely alone.

Exercise 7.2 Make a model that describes the locations of N mussels in an area of, say, 50×50 cm. The locations of the mussels is given by two 1D arrays called X and Y, which give the spatial coordinates of the mussels in each row of the arrays X and Y.

The model should follow the following general structure:

Setup

- Define parameter values such as simulation duration, dimension of the arena, and number of mussels (and much more)
- Initialize the vectors for X and Y coordinate (for instance random between 0 and 50)
- Start the loop
  - Calculate the distances between the mussels
  - Determine the density (number per cm2) of mussels within the two neighbourhood classes V1 and V2
  - Calculate the Speed and Angle of the mussels, based on V1 and V2
  - Calculate the new positions of the mussels
  - If mussels move out of the arena, move them back!
  - Plot the results
- End the loop

First, we have to define the model's parameter setting:

```
% Parameters
N            = 500;      % Number of mussels
Length       = 50;       % Length of the Arena, in cm
EndTime      = 500;      % Number of minutes

% Declaring the parameters that describe movement speed
P1 = 100;
P2 = -80;
P3 = 3;

D1 = 2;  % cm, mussels within this distance are considered neighbors
D2 = 6;  % cm, mussels within this distance are considered
                % as "within cluster distance", in cm.

% Constant and variable declaration, used below
Diagonal=diag(ones(N,1));  % Used below
```

```
Distance=zeros(N,N);        % Assigning the array
```

We now have to describe the initial conditions at the start of the simulation. How will the mussels be distributed in the beginning? In the paper, an even distribution was used. In this exercise, however, we take a simpler approach, and start with a random initial distribution of the mussels:

```
X = rand(N,1)*50;
Y = rand(N,1)*50;
```

We now need to start a loop that repeats the movement calculation for every time step, starting from Time=0 to EndTime (see above). For this, we use a for loop:

```
for Time=1:EndTime,
  …
  …
end;
```

The movement of each mussel (the rate of change of X and Y) is a function of the distances between the mussels. Hence, we need to calculate the distances between the mussels, using Pythagoras' theorem.

Using two nested for-loops, calculate the distance of each mussel to each other mussel, storing the values in a matrix called Distance.

```
for i=1:N
    for j=1:N
        Distance(i,j)= ??? ;
    end
end;
```

7.2.1 Introduce the statement that calculates the distance between mussels by using the theorem of Pythagoras.

We then need to calculate for each mussel the density of direct neighbors within a distance of 2 cm (V_1), and the number of mussels within the "cluster distance" of 6 cm (V_2).

```
Nr_In_Dist1 = (Distance<D1)-Diagonal;
Nr_In_Dist2 = (Distance<D2)-Diagonal;
```

"Diagonal" is a matrix with ones on the diagonal, which we use to correct for the fact that a mussel has distance 0 to itself, and would therefore be counted as within the V_1 and V_2 distances.

This calculates the number of mussels within ranges D_1 and D_2, not the density. To get density, we have to divide by the surface:

```
V1 = sum(Nr_In_Dist1)/(D1.^2*pi()); % sum/surface
V2 = sum(Nr_In_Dist2)/(D2.^2*pi()); % sum/surface
```

Using these two values V_1 and V_2, we need to make an estimate of the step size and direction of the mussels. The direction of the step we take randomly between 0 and 360, using an even

distribution. The size of the step is much more complicated, because it uses an exponential distribution, in which the parameter β is a function of the distance to the neighbors (see above).

To translate the two neighborhood density values (local neighborhood density V_1 and cluster-scale density V_2) into a experimentally distributed random number, we use the equation:

```
Beta=1./(max(0.001,P1*V1'+P2*V2')+P3);
```

Now we can use this to obtain a random StepSize and Angle:

```
StepSize= -Beta .* log(rand(N,1))
Angle = rand(N,1)*360;
```

7.2.2 Using movement speed and movement angle, we now have to calculate the change in X and Y during a time step of a minute (as the parameters are defined per minute). Introduce two statements calculating this (X=X+??? and Y=Y+???).

7.2.3 We now have to check whether the mussels have stayed in the arena, and if not, find a way to move them back. We need 4 statements for this (for each of the 4 sides of the arena). Introduce these statements.

7.2.4 Make a dynamic visualization (within the time loop) that will show the mussels moving.

Now try out your model, and see if the mussels make patterns. If not, just go back and go through your model step by step, looking for where something illogical occurs.

Analyzing the model

Now we have developed the model. Of course, it is important to use the model to investigate how the process of self-organization in mussels depends on the conditions that the mussels occur. Two hypotheses are very important:

1) The process of self-organization is dependent on the number of mussels. If the number of mussels drops below a critical value, self-organization stops.

2) The large-scale inhibition is a crucial process for regular patterns to develop. Otherwise, the mussels will just aggregate in a small number of large clumps.

We are going to test these two predictions in the following two exercises:

Exercise 7.3 Test the effect of mussel numbers, by changing the parameter values of the model (take a big range, from N=2000 to N=100). What do you observe? Is hypothesis 1 correct? Provide your argumentation.

Exercise 7.4 To block inhibition, we need to set P_2 to 0. Also reduce P_1 to 20, because else the mussels won't move anymore. It is also essential to simulate for a long time, say EndTime=10000. Compare a simulation with inhibition on (original parameter settings) with one that has inhibition off. What differences do you observe? How can you explain them? Is hypothesis 2 true?