Modeling Life

Theoretical Biology Group, Utrecht University

2025-04-24

Table of contents

Modeling life

This website accompanies the Modeling Life course at Utrecht University. It primarily serves as a central hub for all the practicals (werkcolleges), with the necessarily files, links and other resources for each day.

Each practical has its own page containing background explanations, code snippets, and questions that are designed for you to learn about the models. Most of these exercises build directly on the lectures, allowing you to explore biological questions—such as how morphogen gradients form, how spatial patterns emerge, or how cells evolve to "stick together".

You can navigate the site using the sidebar or the left. The General Course Info section outlines the general course info (exams, learning goals, etc.). You can also find the Schedule on this website. The individual practicals sections provides detailed instructions for each day. In the second part of the course you will get even more experience doing things yourself by working on a mini-project.

We hope you'll use this website actively. There's lots to read, simulate, modify, and explore.

Note: the Modeling Life course is new. This website is to help you, but is by no means perfect. Please let us know if you see any mistakes, typo's or other issues. Any constructive feedback on how to make things better and easier for you is always welcome.

Part I

Course information

1 General course info

Our names, email addresses, an overview of the course content, learning goals, tips, grading, group formation, usage of Brightspace, materials they need, required attendencee, and feedback is welcome blabla.

2 Schedule

WEEK 1

Nov 10, 2025 – Welcome and Intro to Python (**HC** 13:15-17:00).

Nov 11, 2025 – Pattern formation I: Gradients and segments (**HC** 10:00-12:45 and **WC1** 13:15-17:00).

Nov 13, 2025 – Pattern formation II: Turing Digit patterning (**HC** 10:00-12:45 and **WC2** 13:15-17:00).

WEEK 2

Nov 17, 2025 – Guest lecture: Max Rietkerk, Ecosystem's spatial patterns (**HC** 13:15-17:00).

Nov 18, 2025 – Pattern formation III: Clock and Wavefront (**HC** 10:00-12:45 and **WC3** 13:15-17:00).

Nov 20, 2025 – Morphogenesis: Cell sorting (**HC** 10:00-12:45 and **WC4** 13:15-17:00).

WEEK 3

Nov 24, 2025 – Guest lecture: Ina Sonnen, Signal encoding in multicellular systems (**HC** 13:15-17:00).

Nov 25, 2025 – Cell differentiation: gene regulation in time (**HC** 10:00-12:45 and **WC5** 13:15-17:00).

Nov 27, 2025 – Cell differentiation: gene regulation in space (**HC** 10:00-12:45 and **WC6** 13:15-17:00).

WEEK 4

Dec 1, 2025 – Guest lectures: Vivek Bhardwak and Kaisa Kajala, What is a cell type? genomic and functional perspectives (**HC** 13:15-17:00).

Dec 2, 2025 – Environment and Development (**HC** 10:00-12:45 and **WC7** 13:15-17:00).

Dec 4, 2025 – Evolving populations I: Sticky cells (**HC** 10:00-12:45 and **WC8** 13:15-17:00).

WEEK 5

Dec 8, 2025 – Guest lecture: Rutger Hermsen (**HC** 13:15-17:00).

Dec 9, 2025 - Evolving populations II: Genotype-phenotype map (**HC** 10:00-12:45 and **WC9** 13:15-17:00).

Dec 11, 2025 - Evolving populations III: Microbial communities (**HC** 10:00-12:45 and **WC10** 13:15-17:00).

WEEK 6 (self-study and exam)

Dec 18, 2025 –Exam

WEEK 7-9 (mini projects)

Jan 6, 2026 - Miniprojects presentation and making teams

Following weeks you will have classrooms to work on your miniprojects (13:15-17:00). Also, you should schedule meetings with your supervisor to discuss progress of your miniproject.

WEEK 10 (mini symposium)

Jan 29, 2026 - Miniprojects final presentation

Part II

I) Pattern formation

3 Introduction

3.0.1 TODO

- Write a short intro on pattern formation part of the practicals.

4 Pattern formation I Gradients and segments

4.1 Morphogen Gradients and Patterning

In this practical, we a going to look at how an organism can form segments along its body axis . The mathematical model that we will implement and study is an implementation of the so-called French flag conceptual model first proposed by Lewis Wolpert (Wolpert 1969). It assumes the spatially graded expression of a morphogen "M" that influences the expression of some downstream genes A, B and C. Their expression is often visualized by red, white and blue and the arising pattern resembles the French flag, hence the name (see the power of visualization).

One of the most well-studied organisms when it comes to body axis segmentation (although its segmentation mechanism is evolutionary derived and a-typical!) is the development of the fruit fly Drosophila melanogaster. Supporting the ideas of Wolpert, it was found that through tethering maternal Bicoid mRNA to one side of the embryo, Bicoid protein can form a gradient extending along the anterior-posterior axis, with so called gap genes as a first tier in the segmentation hierarchy differentially responding to different Bicoid protein levels (Driever and Nüsslein-Volhard 1988). Later it was found that often at least two opposing morphogen gradients drive downstream gene expression and genes typically respond to multiple inputs.

4.2 Mathematical modeling - integrating multiple signals

Promotors/enhancers driving gene expression frequently make use of so called OR and AND gates to integrate inputs from different transcription factors. An OR gate can be implemented mathematically with a sum of the effect of the transcription factors, while an AND can be implemented mathematically with a product. Some examples:

- $\frac{dX}{dt} = a(\text{tf1}) + b(\text{tf2})$: Gene X is induced by transcription factor 1 and 2 in an OR fashion, either $a(\text{tf1})$ or $b(\text{tf2})$ needs to be high to give high transcription of X.
- $\frac{dY}{dt} = a(\text{tf1}) \cdot b(\text{tf2})$: Gene Y is induced by transcription factor 1 and 2 in an AND fashion, both $a(\text{tf1})$ and $b(\text{tf2})$, which are being multiplied, need to be high to give high transcription of X.

Note that the shape of $a(\text{tf1})$ and $b(\text{tf2})$ (increasing or decreasing function of the transcription factor) determines whether tf1 and tf2 are repressing or activating.

4.3 Python code

Starting code for this practical

4.4 Questions

Exercise 4.1 (Algorithmic thinking).

Exercise 4.2 (Important concept).

Exercise 4.3 (Biology & mathematical thinking).

Exercise 4.4 (Biology & algorithmic/mathematical thinking).

Exercise 4.5 (Biology & algorithmic/mathematical thinking).

a.

b.

c.

d.

Exercise 4.6 (Biology)**.**

Exercise 4.7 (Algorithmic/mathematical thinking)**.**

Exercise 4.8 (Biological & mathematical thinking)**.**

Exercise 4.9 (Biology & algorithmic/mathematical thinking)**.**

Exercise 4.10 (Biology & algorithmic/mathematical thinking).

4.5 Extra questions

Exercise 4.11 (Biological thinking).

Exercise 4.12 (Biological & algorithmic thinking).

https://www.pnas.org/doi/full/10.1073/pnas.0912734107 https://onlinelibrary.wiley.com/doi/10.1111/dgd.12337

Exercise 4.13 (Algorithmic thinking).

Exercise 4.14 (Algorithmic thinking).

Exercise 4.15 (Algorithmic thinking).

4.6 Relevant literature / further reading

https://www.nature.com/articles/ncomms6077

https://pmc.ncbi.nlm.nih.gov/articles/PMC3109599/

https://www.nature.com/articles/ncomms7679

https://onlinelibrary.wiley.com/doi/10.1111/dgd.12337

https://www.pnas.org/doi/abs/10.1073/pnas.0912734107

5 Pattern formation II Turing digit patterns

1.

2.

5.1 Equivalence of Turing models

Exercise 5.1 (Conceptual thinking)**.**

5.2 Patterning in a fixed domain size

Exercise 5.2 (Biology)**.**

-
-

-
-
-

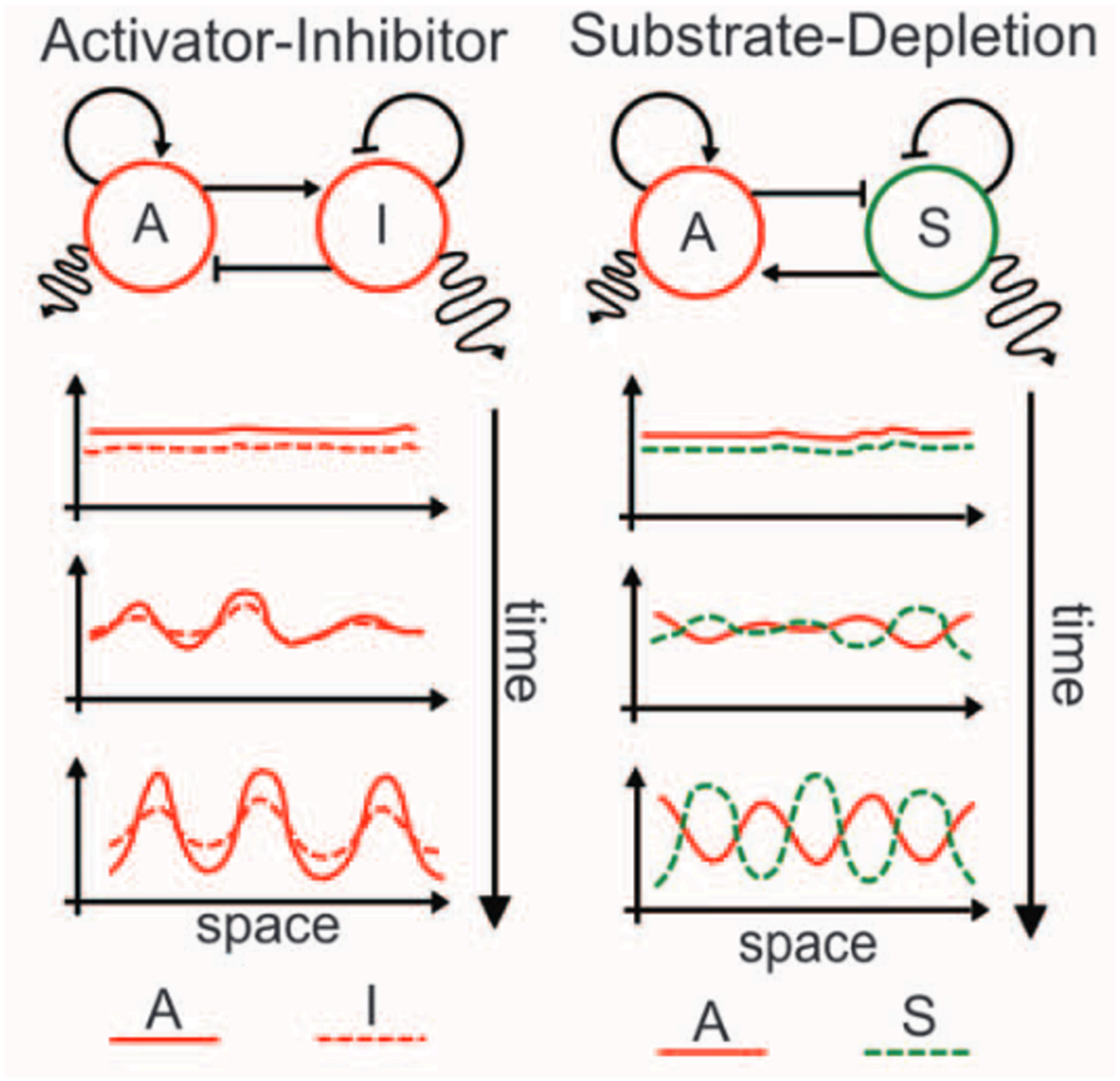

Figure 5.1: Schemes and numerical solutions of two prototypical 2-component Turing models

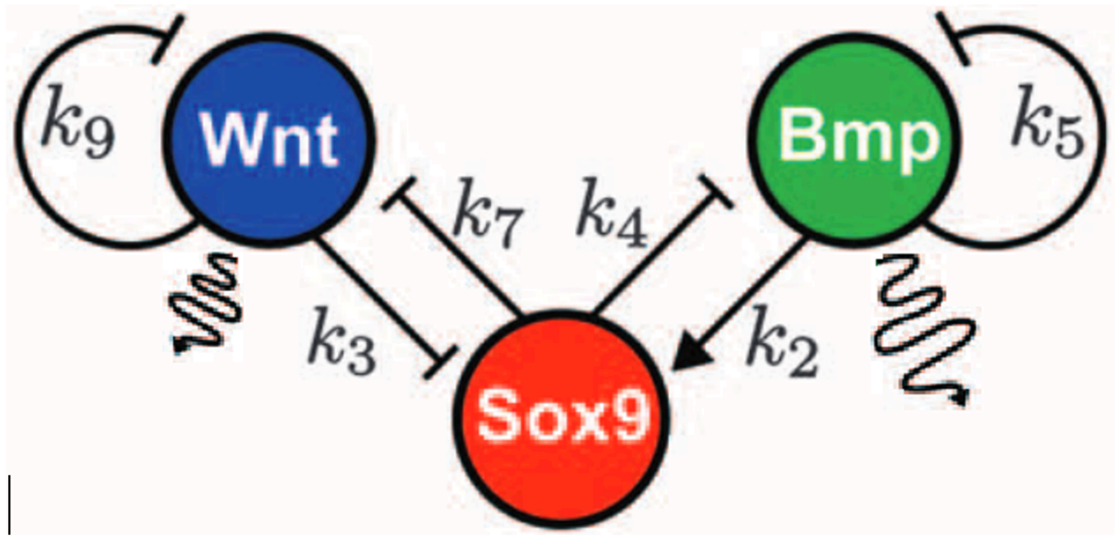

Figure 5.2: Scheme of the BSW model

5.3 The homogenous steady state in Turing models

Exercise 5.3 (Conceptual thinking).

5.4 Patterning in a growing domain size

Exercise 5.4 (Biology).

-
-

5.5 Patterning along two growth axes

Exercise 5.5 (Biology)**.**

5.6 Making a virtual tissue grow

Exercise 5.6 (Algorithmic and conceptual thinking)**.**

5.7 Spatial modulation of parameters k4 and k7

Exercise 5.7 (Biology).

5.8 Hoxd13 and FGF

Exercise 5.8 (Algorithmic and biological thinking)**.**

5.9 Shape makes the tissue

Exercise 5.9 (Biology).

5.10 Master students exercise

Exercise 5.10 (Poly and oligodactyly).

-

-
-

-

6 Pattern formation III Clock-and-wavefront

6.1 Goal of the tutorial:

6.2 The model system

6.3 Programming with classes

-
-
-

this online tutorial

6.4 Questions

Exercise 6.1.

2.

3.

Lewis (2003)

4.

5.

6.

1.

1.

2.

3.
4.

2.

7.

8.

1.

2.

1.

1.
2.
3.
4.
5.
6.

2.

3.

9.

10.

•
•
•

•

•

•

6.5 Further (open) questions

•

•

•
•

6.6 References for biological parameters:

(a) *https://anatomypubs.onlinelibrary.wiley.com/doi/10.1002/1097-0177(2000)9999:9999%3C::AID-DVDY1065%3E3.0.CO;2-A*

(b) *https://doi.org/10.1016/j.gde.2012.05.004*

(c) *https://doi.org/10.1242/dev.65.Supplement.103*

(d) *https://doi.org/10.1242/dev.161257*

(e) *https://doi.org/10.3389/fcell.2022.944016*

(f) *https://www.sciencedirect.com/science/article/pii/S0079610718300178*

6.7 Combining the clock and FGF wavefront model - programming instructions

1.
2.
3.
4.
5.
6.
7.
8.

6.8 Relevant literature

Lewis (2003) _Autoinhibition with Transcriptional Delay A Simple Mechanism for the Zebrafish Somitogenesis Oscillator

Part III

II) Morphogenesis

7 What is morphogenesis?

7.0.1 TODO

-

8 Gene regulation in time

8.1 Introduction

8.2 Part I - Encoding regulatory interactions

```
def ODEgeneRegulation(a,t,parameters):
    prod=parameters['prod']
    decay=parameters['decay']
    Ksat=parameters['Ksat']
    nodeA=parameters['nodeA']
    nodeB=parameters['nodeB']
    n=parameters['n']
    outputC=a[0]
    doutputC=prod*nodeA**n/(Ksat**n+nodeA**n)*nodeB**n/(Ksat**n+nodeB**n)-decay*outputC
    return(doutputC)
```

```python
def logicalRule(nodeA,nodeB):
    return(nodeA and nodeB)
```

Exercise 8.1 (Mathematical thinking)**.**

```python
# ODE model - see the parameters used in circuitsfunctions.py
# ODErun has three arguments: model, geneA, geneB
A=10
B=10
ODErun(ODEgeneRegulation,A,B)

# Boolean model
# look at the terminal for the result:
A=1
B=1
print("the boolean operation of nodeA ",A," AND nodeB",B," is:", logicalRule(A,B))
```

```python
# First, Boolean network simulation - AND gate
explorationvalue=2 # a Boolean model assumes there is only 2 possible states: 0 or 1
bool_output = np.zeros((explorationvalue, explorationvalue))
for nodeA in range(0, explorationvalue):
    for nodeB in range(0, explorationvalue):
        bool_output[nodeA, nodeB] = nodeA and nodeB #AND # CHANGE THIS

# Now, ODE model simulation
explorationvalue = 11 # an ODE model allows us to explore more values than a boolean model
ode_output = np.zeros((explorationvalue, explorationvalue))
for nodeA in range(0, explorationvalue):
    for nodeB in range(0, explorationvalue):
        parameters = {'prod': 0.01, 'decay': 0.001,'Ksat': 4, 'n': 2,'nodeA':nodeA,'nodeB':no
        cells = odeint(ODEgeneRegulation, 0, np.arange(0, 1000.1 , 0.1), args=(parameters,))
        ode_output[nodeA, nodeB] = cells[-1, 0]

# use this code to see your ode and boolean results side by side
ODEBooleanPlot(ode_output, bool_output)
```

Exercise 8.2 (Biology).

1.
2.
3.
4.
5.
6.

-

Exercise 8.3 (Algorithmic thinking).

8.3 Part II – Gene regulatory network

```
timesteps=20
nodes=18
matrix = np.zeros((timesteps+1, nodes), dtype=int)
matrix[0,:] = np.random.randint(0, 2, size=nodes) #Random initial condition
for i in range(timesteps):
    parameters= {'CK': matrix[i,0], 'ARR1': matrix[i,1], 'SHY2': matrix[i,2], 'AUXIAAR': mat
    matrix[i+1, :] = rootNetwork(parameters)

plotBooleanTimecourse(matrix,timesteps)
```

```
matrix[0,:]=[0,1,1,0,1,1,0,0,1,0,1,1,1,1,1,0,1,0]
```

Exercise 8.4 (Algorithmic thinking)**.**

```
ICs=100
attractors = np.zeros((ICs, len(parameters))) # new

# your code

plotBooleanAttractors(attractors) # it takes as argument your matrix of attractors
```

```
UMAPBoolean(attractors)
```

Exercise 8.5 (Biology / abstract thinking).

```
attractors_sorted = np.array(sorted(attractors.tolist()))
plotBooleanAttractors(attractors_sorted)
```

Exercise 8.6 (Biology / abstract thinking).

Exercise 8.7 (Biology).

```
_, unique_indices = np.unique(attractors, axis=0, return_index=True)
attractors_unique = attractors[np.sort(unique_indices)]
plotBooleanAttractors(attractors_unique)
```

Exercise 8.8 (Biology / abstract thinking).

8.4 Part III – Cell differentiation – jumping from one attractor to another

```
timerunning=10.1
times = np.arange(0, timerunning, 0.1)

IC = np.random.randint(0, 2, size=18).tolist() #random initial condition
parameters = {'decayrate': 1, 'h': 50}
cells = odeint(rootNetworkODE, IC, times, args=(parameters,))

plotODEroot(cells,times)
```

Exercise 8.9 (Modeling choices).

```
# We start here:
IC=[0,0,0,0,1,0,1,1,1,1,1,1,1,0,1,0,1,0]
# We want to end here.
end=[1,1,0,0,1,1,1,1,1,1,0,0,1,0,0,0,0,1]
# node order
# CK, ARR1, SHY2, AUXIAAR, ARFR, ARF10, ARF5, XAL1, PLT, AUX, SCR, SHR, MIR165, PHB, JKD, MGI

# your code

plotODErootTransition(cells,times) # use this function to plot your result
```

Exercise 8.10 (Algorithmic thinking / biology).

Exercise 8.11 (Biology).

9 Gene regulation in space

9.1 TODO FOR

-
-

9.2 Introduction

9.3 Biological background

9.4 The model

9.5 Files

9.6 Questions

Exercise 9.1.

Exercise 9.2.

Exercise 9.3.

Exercise 9.4.

Exercise 9.5.

Exercise 9.6.

Exercise 9.7.

Exercise 9.8.

Exercise 9.9.

Exercise 9.10.

Part IV

III) Cell differentiation

10 Differentiation introduction

10.0.1 TODO

-

11 Practical 6

11.0.1 TODO

-

12 Practical 7

12.0.1 TODO

-

Part V

IV) Evolution

13 Introduction to evolution

13.1 Evolution: Life's most clever algorithm

13.2 Three ingredients

-
-
-

13.3 Balancing change and stability

13.4 A simple evolutionary algorithm

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(5)

N = 100 # Population size
fitnesses = np.full(N, 0.05)
mu = 0.01
# Updated parameters
steps = 50000
avg_fitness = []

# Moran process with mutation (logging every 10 steps)
for step in range(steps):
    probs = fitnesses / fitnesses.sum()
    parent = np.random.choice(N, p=probs)
    dead = np.random.choice(N)
    # Copy with mutation
    new_fit = fitnesses[parent]
    if np.random.rand() < mu:
        new_fit = np.clip(new_fit + np.random.normal(0, 0.05), 0, 1)

    fitnesses[dead] = new_fit

    # Save average fitness every 10 steps
    if step % 10 == 0:
        avg_fitness.append(fitnesses.mean())

# Plotting
plt.plot(np.arange(0, steps, 10), avg_fitness)
plt.xlabel("Step")
plt.ylabel("Average fitness")
plt.title("Evolution of Fitness in a Moran Process")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Exercise 13.1 (Moran process simulation)**.**

a.
b.
c.
d.

13.5 What this part of the course is about

14 Sticking together

Sticking together

14.1 Steering

CODE FOR “moving vector in Python”

```
import numpy as np
import matplotlib.pyplot as plt

# Enable interactive mode for matplotlib
plt.ion()

# Setup figure and axis for plotting the arrow
fig, ax = plt.subplots(figsize=(8, 4))
ax.set_xlim(0, 600)  # x-axis limits
ax.set_ylim(0, 250)  # y-axis limits
ax.set_aspect('equal')  # Keep aspect ratio square
ax.set_facecolor('#f0f0f0')  # Background color
ax.set_title("A moving vector with an arrowhead")  # Title

# Initial position and velocity
x, y = 250.0, 180.0      # Position coordinates
vx, vy = 5.0, 10.5        # Velocity components


def draw_arrow(x, y, vx, vy):
    """
    Draws an arrow at position (x, y) with velocity (vx, vy).
    """
    ax.clear()
    ax.set_xlim(0, 600)
    ax.set_ylim(0, 250)
    ax.set_aspect('equal')
    ax.set_facecolor('#f0f0f0')
    ax.set_title("A moving vector with an arrowhead")

    # Normalize velocity for drawing the arrow

    dx = vx*5
    dy = vy*5

    # Arrow shaft
    end_x = x + dx
    end_y = y + dy

    # Arrowhead calculation
    angle = np.arctan2(dy, dx)
    angle_offset = np.pi / 7
    hx1_x = end_x - np.cos(angle - angle_offset)
    hx1_y = end_y - np.sin(angle - angle_offset)
    hx2_x = end_x - np.cos(angle + angle_offset)
    hx2_y = end_y - np.sin(angle + angle_offset)

    # Draw shaft
    ax.quiver(x, y, dx, dy, angles='xy', scale_units='xy', scale=1, color='#007acc', width=
    # Draw base point
    ax.plot(x, y, 'o', color='#333')

    # Labels
```

Exercise 14.1 (Playing with steering arrows - Mathematical thinking)**.**

a.

Moving "cells"

[1]

[1]The code also contains a `Visualisation` class that uses the `matplotlib` library to draw the cells and their movement, which we have tuned to speed things up a bit. You do not need to understand this part of the code, but if you are interested feel free to check it out.

STARTING CODE FOR “moving cells”

14.2 Moving the target

Exercise 14.2 (Playing with steering arrows - Algorithmic/mathematical thinking)**.**

a.
b.
c.

14.3 Reproduction

Exercise 14.3 (The birth of an arrow - Biological / algorithmic thinking)**.**

a.
b.
c.