Modeling Life

Theoretical Biology Group, Utrecht University

2025-11-10

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 necessary 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 on 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: 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

General course info

Contact

Course coordinator: Monica Garcia Gomez m.l.garciagomez@uu.nl. In this website you will find all practical information about the course. You can reach the instructors via email at Modeling Life email

Figure 1: **Teachers:** Kirsten ten Tusscher K.H.W.J.tenTusscher@uu.nl, Erika Tsingos e.tsingos@uu.nl, Monica Garcia Gomez m.l.garciagomez@uu.nl, Bram van Dijk b.vandijk@uu.nl.

We are part of the Theoretical Biology group of Utrecht University).

Course content

This course will consist of lectures (HC) covering various modeling approaches to study life at different levels accompanied by computer practicals (WC) where students will get hands-on experience on running computational models of the development or the evolution of animals, plants, and microbial systems.

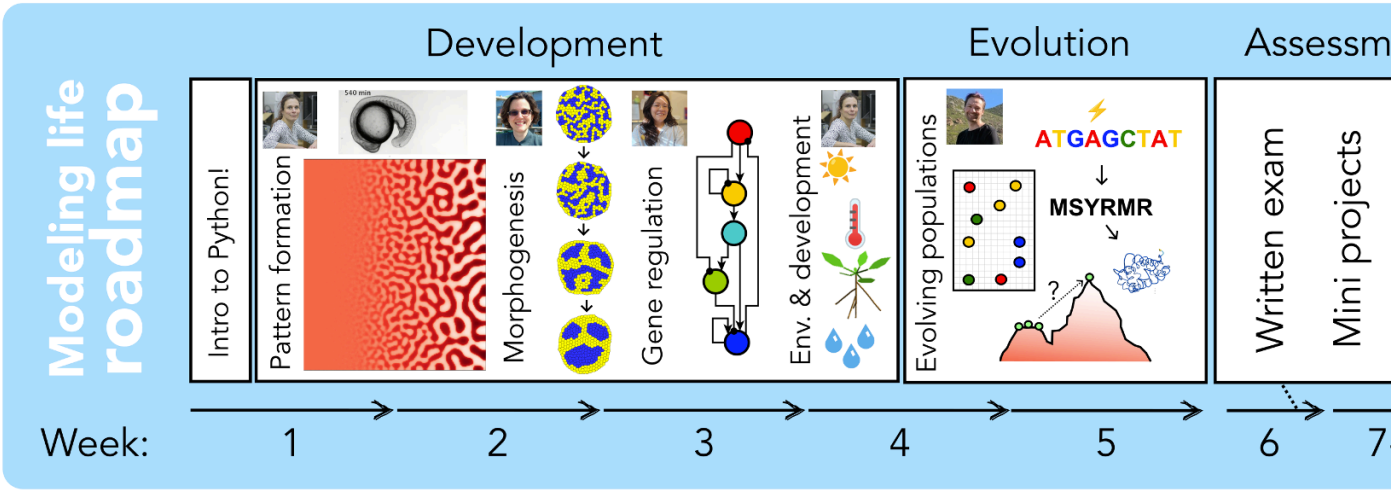


Figure 2: Rough roadmap of the Modeling life course

General schedule:

Monday – Guest lectures (mandatory)

Tuesdays / Thursdays– Lecture and practicals (mandatory)

Key dates:

- December 18, 2025: Exam
- Jan 6, 2026: Mini projects presentations
- Jan 29, 2026: Mini symposium

Learning Goals

The course aims to provide you with an introductory understanding of computational modeling to study living systems, and their large range of applications.

After this course the student can:

- Explain how biological systems can be studied with computational models,
- Translate a biological system into a working computational model using python,
- Use algorithmic thinking to break down problems into programmable steps,
- Identify the underlying assumptions and limitations of computational models,
- Identify the modeling approach & formalism best suited for a research question.

Communication

Feedback and other matters: modelinglife@uu.nl or m.l.garciagomez@uu.nl

Schedule, slides, practicals: https://modelinglife-uu.github.io/

Code: https://github.com/moneralee/UU_Modeling-Life-course

Brightspace: is the channel for all official communications (*e.g.* final grades).

Teams is not used to livestream lectures. However, if you have any questions for the instructors, student assistants, or your fellow students, it's very helpful to share them on Teams (General channel) so everyone can benefit.

Modeling Life 2025 in Microsoft **Teams**. You can sign up as follows:

- Open MS Teams
- In the menu on the left, select the "Teams" icon
- Click the "Join or create a team" button at the bottom left of the screen (or, depending on the Teams version, at the top right)
- Find the tile that says "Join a team with a code"
- Enter the following code in this field: 6jhv6z5
- Select "Join team"

Grading

To pass this course, a minimum of 5.5 is mandatory. Your grade is calculated by the following components:

Exam: 70%

Mini project: 30%

You will find your Final grade on Brightspace.

Attendance

Attendance is mandatory to all lectures and practicals. Should you be unable to attend, please communicate this by email to: m.l.garciagomez@uu.nl.

Feedback

Please help us improve the course by providing feedback via Caracal and other means. We want to make this new course as good as possible for you and future students.

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.

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** 11:00-12:45 and **WC1** 13:15-17:00).

Nov 13, 2025 – Pattern formation II: Turing patterns (**HC** 11:00-12:45 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** 11:00-12:45 and **WC3** 13:15-17:00).

Nov 20, 2025 – Morphogenesis: Cell sorting (**HC** 11:00-12:45 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** 11:00-12:45 and **WC5** 13:15-17:00).

Nov 27, 2025 – Cell differentiation: gene regulation in space (**HC** 11:00-12:45 and **WC6** 13:15-17:00).

Week 4

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

Dec 2, 2025 – Environment and Development (**HC** 11:00-12:45 and **WC7** 13:15-17:00).

Dec 4, 2025 – Evolving populations I: Sticky cells (**HC** 11:00-12:45 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** 11:00-12:45 and **WC9** 13:15-17:00).

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

Week 6 (self-study and exam)

Dec 18, 2025 –Exam

Week 7-9 (Mini projects)

Jan 6, 2026 - Mini projects presentation and making teams

The following weeks you will have classrooms available to work on your mini projects (check in myTimetable, 13:15-17:00). Also, you should schedule meetings with your supervisor to discuss progress of your mini project.

Week 10 (Mini symposium)

Jan 29, 2026 - Mini projects final presentation

Slides

0) Introduction to Python

This introductory page explains how to install Python and Spyder, and provides a brief overview of Python programming. You will also have access codes to a great StudyLens app to quickly get a grasp of Python programming, especially the parts directly relevant to this course.

Installing Spyder and Anaconda

To best way to install Spyder is to do so via Anaconda, which is a free and open-source distribution of Python for scientific computing. You can download Anaconda from this page (note: use the download link on the *left*, and **not** the one called "miniconda" which does not include Spyder). After downloading, follow the installation instructions.

Once done, you now have:

- Python
- Spyder (our main editor)
- Most scientific packages like numpy, matplotlib, and pandas
- Conda and pip (for if you want to install other packages)

Launching Spyder

You can now launch Spyder via the 'Anaconda Navigator' application. In MacOS you find this under 'Applications' > 'Anaconda-Navigator' (it doesn't always show up in the Spotlight search immediately). In Windows, you can search for 'Anaconda Navigator' in the Start Menu. Once the Anaconda Navigator is open, you can launch Spyder by clicking on the 'Launch' button under the Spyder icon.

Disabling inline plots

By default, Spyder shows figures inside the "Plots" pane, but in this course we typically use plots that update dynamically (animations), which don't work well in that mode. So we need to change that.

Here's how:

1. Click the 'preferences' icon in the top panel (a wrench icon).
2. Go to 'IPython Console' → Graphics → Backend
3. Choose Qt, Qt5, or Qt6[1]
4. Click Apply and OK
5. Restart Spyder may or may not be necessary depending on your operating system.

Now your plots will open in a separate window and can animate properly!

Testing your setup!

Throughout this course you will either work with scripts you have been handed out, or scripts that you can copy/paste from this website. Let's test your Spyder setup with the following script:

```
import random, matplotlib.pyplot as plt

plt.ion()
fig, ax = plt.subplots()
ax.set_ylim(-2, 2)
ax.set_xlim(0, 100)
ax.set_title("Random Wiggle Test")
line, = ax.plot([], [], color='seagreen', lw=2)

y = [0]
for i in range(100):
    y.append(y[-1] + random.uniform(-0.1, 0.1))  # random wiggle
    line.set_data(range(len(y)), y)
    ax.set_xlim(0, len(y))
    plt.pause(0.03)

ax.set_title("Animation works! (you can close this window)")
plt.ioff()
plt.show()
```

[1] on some laptops, Tk may be faster, but DON'T use 'inline'

Paste this into a new script in Spyder and hit the 'play' icon (or press F5). Does the animation show up in a seperate window and is it animated? Good, you're ready to go!

Installing VS code

Instead of Spyder, you may want to use a nicer IDE. The most popular one is currently Visual Studio Code (VS Code). To install VS Code and connect it to your Anaconda3 python installation, do the following:

1. Download and install VS code (https://code.visualstudio.com/download)
2. Open VS code and install the Python extension (search for 'Python' in the extensions tab on the left, and install the one with the blue check mark)
3. Select the python interpreter that comes with Anaconda. You can do this by pressing `Ctrl+Shift+P` (or `Cmd+Shift+P` on Mac) to open the command palette, then type `Python: Select Interpreter` and choose the one that points to your Anaconda installation (it should have 'anaconda3' in the path).

Now, you should be able to run all the code from the course via VS code as well! While Spyder is nice, it also is a little less stable than VS code, so if you run into issues, consider switching to VS code! :)

Introduction to Python Programming

This document provides a brief introduction to Python programming. It covers basic concepts such as variables, data types, control structures, functions, and libraries.

Variables and Data Types

In Python, variables are used to store data. You can create a variable by assigning a value to it using the `=` operator. For example:

```
x = 10
y = "Hello, World!"
```

Python has many built-in data types, including:

- Integers (`int`): Whole numbers, e.g., `10`, `-5`
- Floating-point numbers (`float`): Decimal numbers, e.g., `3.14`, `-0.001`
- Strings (`str`): Text, e.g., `"Hello"`, `'Python'`
- Booleans (`bool`): True or False values, e.g., `True`, `False`
- Lists (`list`): Ordered collections of items, e.g., `[1, 2, 3]`, `['a', 'b', 'c']`
- Dictionaries (`dict`): Key-value pairs, e.g., `{'name': 'Alice', 'age': 25}`
- And many more

Control Structures

Control structures allow you to control the flow of your program. Common control structures in Python include:

- Conditional statements (`if`, `elif`, `else`):

```python
if x > 0:
    print("x is positive")
elif x == 0:
    print("x is zero")
else:
    print("x is negative")
```

- Loops (`for`, `while`):

```python
for i in range(5):
    print(i)
count = 0
while count < 5:
    print(count)
    count += 1
```

Functions

Functions are reusable blocks of code that perform a specific task. You can define a function using the `def` keyword:

```python
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))
```

Libraries

Python has a rich ecosystem of libraries that extend its functionality. Most scientific packages are shipped with Anaconda, so you don't need to worry about these. These include:

- NumPy: For numerical computing
- Pandas: For data manipulation and analysis
- Matplotlib: For data visualization
- SciPy: For scientific computing

But if you still wish to install other packages, you can use `pip` or `conda` in the terminal. For example, to install the `html5` library using pip, you would run:

```
pip install html5lib
```

From the command line. You can also run this bit of code within the Spyder console by adding an `!` in front of the command, which tells the console to run it as a shell command:

```
!pip install html5lib
```

But you probably won't have to install a lot of packages during this course. (nearly) All scripts we use only rely on the base packages.

StudyLens practice

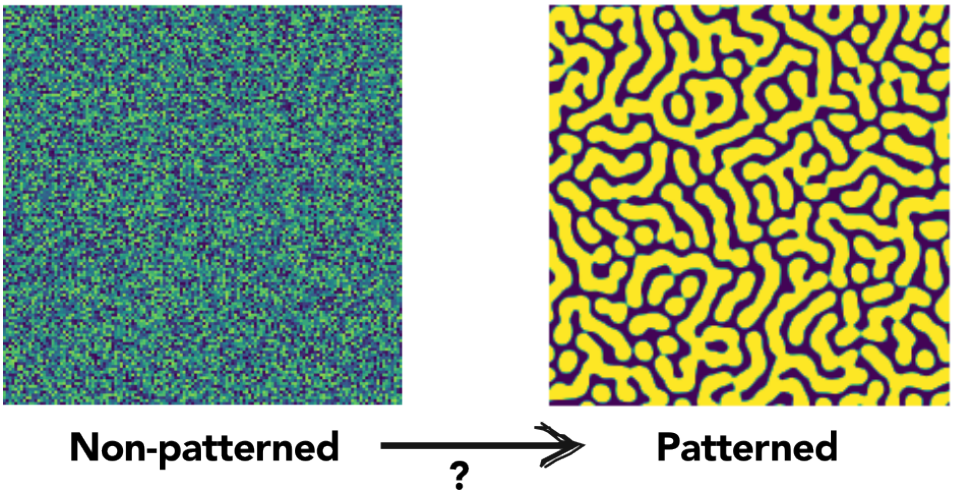
If you've installed Python and read through the basics of Python above, it's time to dive into the StudyLens exercises. For this, login to StudyLens and use the username and password that was shared with you on Brightspace.

Part II

I) Pattern formation

1 Pattern formation (intro)

The field of pattern formation studies the mechanisms that underly the formation of spatial patterns in biological systems. Patterns may arise at any biological level of organization, from single cells polarizing to decide in which direction to move, grow or divide, to the formation of body axes, different cell types and shapes that set apart complex multi-cellular organisms from amorphous blobs, to entire ecosystems patterning where e.g. plants do and do not grow. An important concept in pattern formation is so called "symmetry breaking", which refers to the destruction of an originally homogeneous or non-patterned state, to a non-homogeneous patterned state (see figure below).



Focusing on multi-cellular development, pattern formation addresses how within an organism in which all cells (except for germ cells that have undergone meiosis and immune cells applying VJ recombination) share the same genetic material "symmetry is broken" resulting in usage of a different subset of genes and functions by different cells. Symmetry breaking is needed for the creation of body axes, domains with different functions as well as repeating elements.

A limited range of mechanisms leading to symmetry breaking exist, which have been used time and again by evolution to create patterns in animals, plants, fungi and multicellular algae. Major mechanisms are morphogen gradients, where graded distribution of a signal provides distinct input to cells parallel to the gradient allowing both regionalization and segmentation; Turing patterns, where initial noise combined with diffusion induced destabilization

lead to repetitive patterns, and clock-and-wavefront patterning where autonomous oscillations combined with growth and a memorization mechanism provide an alternative means for segmenting a tissue. We will explore these 3 mechanisms in the practicals. In addition, in plants self-organized patterning of auxin transporters underly phyllotaxis (the positioning of new leaf organs at the shoot apex) and leaf venation. Note that this list of symmetry breaking mechanisms is not exhaustive and additional mechanisms such as lateral inhibition and planar cell polarity exist.

> **i** Links throughout part I-V
>
> Although this course is divided into 5 clearly distinct topics, there is also substantial overlap. We challenge you to see how all sections use similar concepts, and to think about how different types of models may even be combined. To help you along with this integrated view of modeling biology, we will below discuss some of the links with future topics. It may be a good idea to go back to this text later in the course, and reflect if you indeed see all the links.

Often, initial signals like gradients and clocks are transient, implying that the patterns they induce require additional mechanisms of memorization. Critical for understanding this memorization process is the concept of multistability, where two or more alternative stable states of the system exist and the initial signals bring the system from the original to a new patterned state. This concept will be further explored in **Part III** of the course which focuses on Differentiation. Of course, to form a multicellular organism with a functional shape, simply telling a blob of cells where the head or tail needs to be or which cells will become finger bones and which cells will apoptose to carve out the tissue between the fingers is insufficient, and actual shape changing processes need to occur. This we will discuss in **Part II**.

Particularly for animal development, simply breaking symmetry and stably memorizing formed patterns is not enough, scaling of the pattern with body plan size and robustness against noise from gene expression, cell division, cell signalling and other processes is essential for fitness. In contrast, in plants developmental plasticity, the potential to adjust developmental patterning to environmental conditions, plays a key role. This latter concept will be discussed in **Part IV** of the course on Environment.

The repeated usage of a limited number of possible symmetry breaking mechanisms also raises interesting *evolutionary* questions (**Part V**). Are there indeed only a limited number of options, or did evolution select for specific mechanisms that are more robust or more evolvable? Or are some mechanisms simply easier to find? This way of evolutionary thinking will be further discussed in **Part V**.

2 Gradients and segments

2.1 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.

2.2 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.

2.3 Python code

The code below is also in `01_morphogengradient_to_segments.py`.

Starting code for this practical

2.4 Questions

Exercise 2.1 (Algorithmic thinking)**.**

Exercise 2.2 (Important concept)**.**

Exercise 2.3 (Biology & mathematical thinking)**.**

Exercise 2.4 (Biology & algorithmic/mathematical thinking)**.**

Exercise 2.5 (Biology & algorithmic/mathematical thinking)**.**

a.

b.

c.

d.

Exercise 2.6 (Biology)**.**

Exercise 2.7 (Algorithmic/mathematical thinking)**.**

Exercise 2.8 (Biological & mathematical thinking)**.**

Exercise 2.9 (Biology & algorithmic/mathematical thinking)**.**

a.

b.

c.

Exercise 2.10 (Biology & algorithmic/mathematical thinking)**.**

2.5 Extra questions

Exercise 2.11 (Biological thinking)**.**

Exercise 2.12 (Biological & algorithmic thinking)**.**

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

Exercise 2.13 (Algorithmic thinking)**.**

Exercise 2.14 (Algorithmic thinking)**.**

Exercise 2.15 (Algorithmic thinking)**.**

2.6 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

3 Turing patterns, repetitive patterning

Figure 3.1: Credit: the internet.

3.1 Equivalence of Turing models

Exercise 3.1 (Conceptual thinking)**.**

3.2 Patterning in a fixed domain size

Exercise 3.2 (Biology)**.**

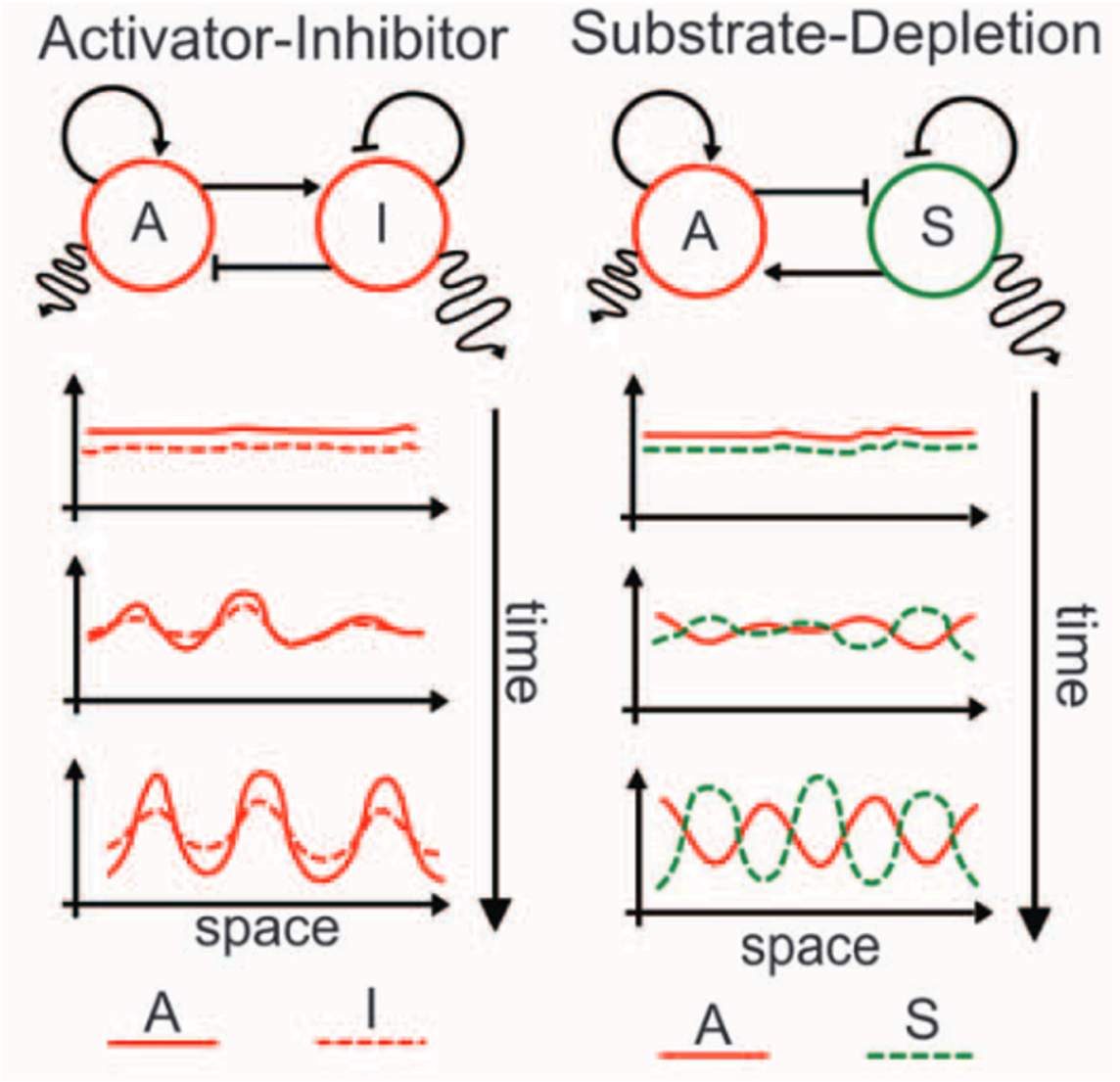


Figure 3.2: Schemes and numerical solutions of two prototypical 2-component Turing models

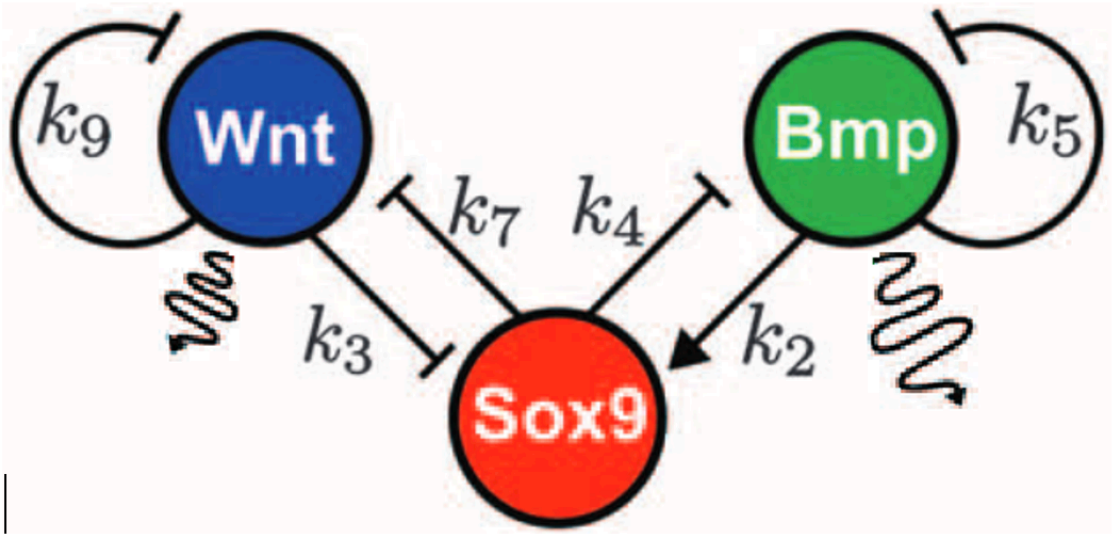


Figure 3.3: Scheme of the BSW model

3.3 Patterning in a growing domain size

Exercise 3.3 (Biology)**.**

-
-

3.4 Patterning along two growth axes

Exercise 3.4 (Biology)**.**

3.5 Making a virtual tissue grow

Exercise 3.5 (Algorithmic and conceptual thinking)**.**

3.6 Spatial modulation of parameters k4 and k7

Exercise 3.6 (Biology)**.**

3.7 Hoxd13 and FGF

Exercise 3.7 (Algorithmic and biological thinking)**.**

3.8 Shape makes the tissue

Exercise 3.8 (Biology)**.**

3.9 Master students exercise

Exercise 3.9 (Poly and oligodactyly)**.**

-
-
-

-

3.9.1 References

1.
2.
3.

4 Clock-and-wavefront patterning

4.1 Goal of the tutorial:

4.2 The model system

4.3 Programming with classes

-
-
-

this online tutorial

4.4 Questions

Exercise 4.1 (Biology - An alternative gradient forming mechanism)**.**

Exercise 4.2 (Mathematics)**.**

Lewis (2003)

Exercise 4.3 (Biology)**.**

Exercise 4.4 (Algorithmic thinking)**.**

Exercise 4.5.

1.

2.

3.

4.

Exercise 4.6 (Biology)**.**

Exercise 4.7 (Biology & Programming)**.**

1.

05_clockplusfgf.py

2.

3.

4.

Exercise 4.8 (Biology)**.**

-
-
-

-
-
-

Exercise 4.9 (Questions for master students)**.**

-
-
-
-

4.5 Relevant literature

4.6 References for biological parameters:

(a)

(b)

(c)

(d)

(e)

(f)

Part III

II) Morphogenesis

5 Introduction to Morphogenesis

6 Cell sorting by differential adhesion

6.1 Morphogenesis

6.2 Biological background

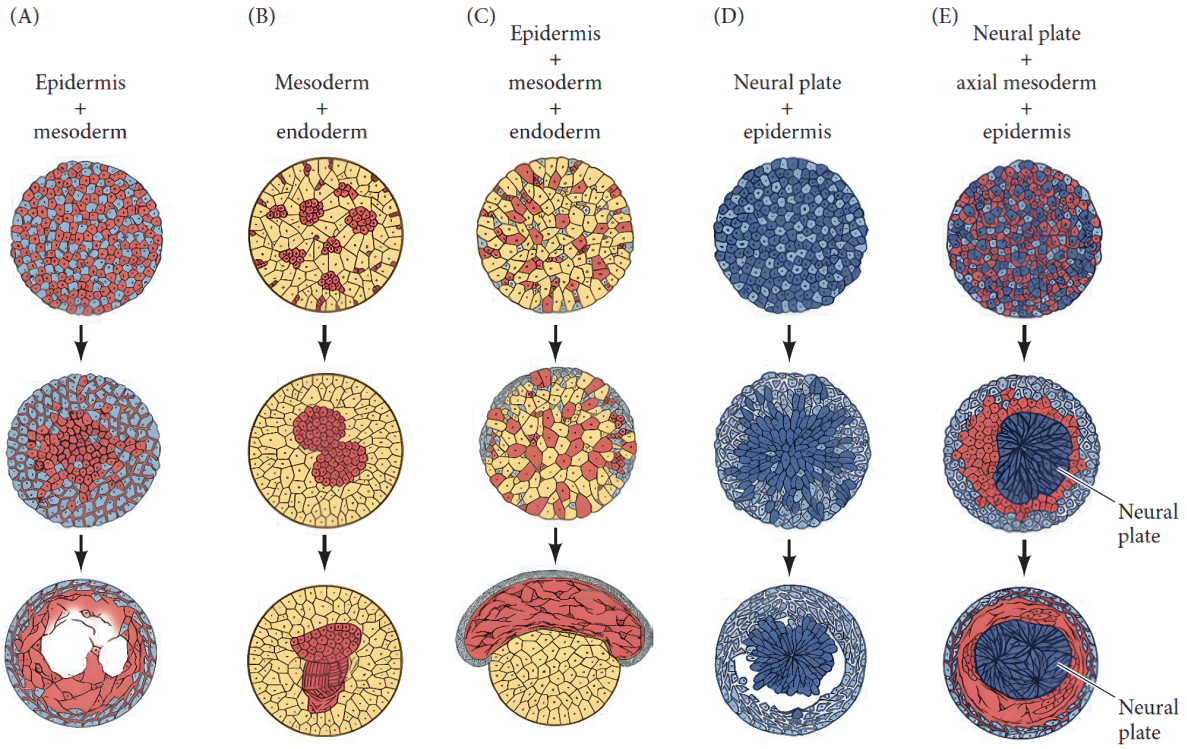


Figure 6.1: Sorting of aggregates of embryonic cells, based on experiments by Townes and Holftreter in 1955. Image adapted from Gilbert "Developmental Biology" 11th edition.

6.3 The cellular Potts / Glazier-Graner-Hogeweg model

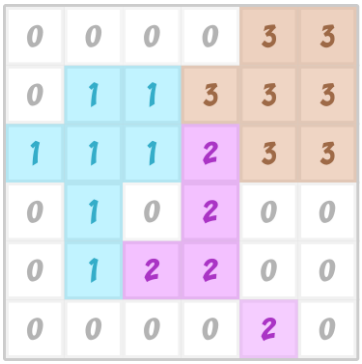


Figure 6.2: Example of a field with four unique integer values. The value 0 is reserved for the medium, while the non-zero values represent three cells.

Figure 6.3: Every copy attempt affects a pair of neighboring lattice sites.

$$P(\text{accept}) = \begin{cases} 1 & \text{if } \Delta H < -H_0 \\ \mathrm{e}^{-\frac{\Delta H + H_0}{T}} & \text{if } \Delta H \geq -H_0 \end{cases}$$

Figure 6.4: Equation 1.

6.4 Exercises

Exercise 6.1 (Algorithmic thinking - Anatomy of a cellular Potts model)**.**

-
-
-
-
-
-

-
-
-
-

Exercise 6.2 (Conceptual thinking - Interpreting the Hamiltonian energy function)**.**

$$H = \sum_{\substack{i \text{ neighbours of } x_T, \\ \sigma(x_i) \neq \sigma(x_T)}} J\left(\tau(\sigma(x_i)), \tau(\sigma(x_T))\right) \quad + \sum_{\substack{\sigma \in (\sigma(x_T), \sigma(x_S)) \\ \sigma(x_T) > 0, \sigma(x_S) > 0}} \lambda_{A(\tau(\sigma))} \left(A(\sigma) - A_{0,\tau(\sigma)}\right)^2$$

Figure 6.5: Equation 2.

$\sum_{\substack{i \text{ neighbours of } x_T, \\ \sigma(x_i) \neq \sigma(x_T)}} J\left(\tau(\sigma(x_i)), \tau(\sigma(x_T))\right)$	$\sum_{\substack{\sigma \in (\sigma(x_T), \sigma(x_S)) \\ \sigma(x_T)>0, \sigma(x_S)>0}} \lambda_{A(\tau(\sigma))} \left(A(\sigma) - A_{0,\tau}\right.$
· Initialize variable $H_J = 0$ · Define target site x_T and target type $\tau(\sigma(x_T))$ · Loop over neighbors of target site x_T · Get the spin σ of neighbor x_i · If σ of x_i is unequal to σ of x_T · Get the interfacial energy j between target type $\tau(\sigma(x_T))$ and neighbor type $\tau(\sigma(x_i))$ from matrix J · Increment $H_J = H_J + j$	· Initialize variable $H_A = 0$ · Define target site x_T and source site x_S · Loop over spin types σ corresponding to x_T and x_S · If the spin σ is greater than 0 · Calculate the area of this spin $A(\sigma)$ · Calculate squared difference to the target area o corresponding to this spin $A_{0,\tau(\sigma)}$ and store in variab · Obtain the weight $\lambda_{A(\tau(\sigma))}$ corresponding to the cell this spin σ · Increment $H_A = H_A + A * \lambda_{A(\tau(\sigma))}$

-
-

Exercise 6.3 (Conceptual thinking - How parameters affect the Hamiltonian energy function)**.**

-
-

-

-

Exercise 6.4 (Algorithmic thinking - The Monte Carlo update algorithm)**.**

-
-
  -
-
  -

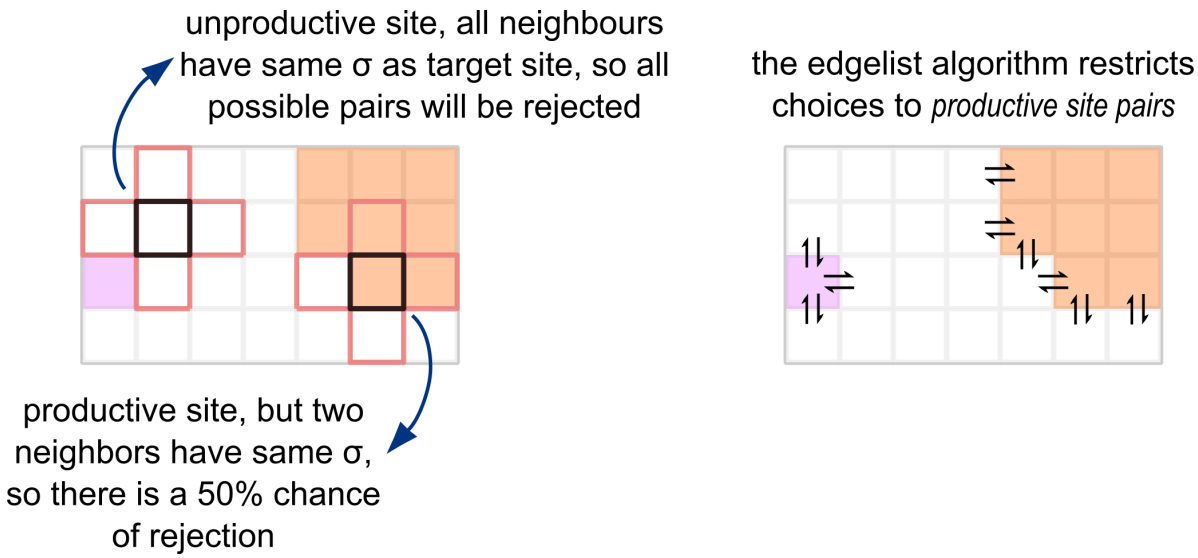


Figure 6.6: Since a copy attempt affects a pair of neighboring lattice sites, we can keep track of valid pairs (productive site pairs). These can be represented by an edge, i.e. an arrow pointing from one site to the other. Edges have a directionality, so two neighboring sites are always joined by two edges with opposing direction.

Exercise 6.5 (Biology - Expanding the Hamiltonian)**.**

1.
2.
3.

4.

1.

2.

$$H_S = \sum_{\substack{\sigma \in (\sigma(x_T), \sigma(x_S)) \\ \sigma(x_T)>0, \sigma(x_S)>0}} \lambda_{S(\tau(\sigma))} \left(S(\sigma) - S_{0,\tau(\sigma)} \right)^2$$

Figure 6.7: Equation 3.

-
-

-

Exercise 6.6 (Biology - Imposing cell connectivity)**.**

-
-

Exercise 6.7 (Biology - Exploring differential adhesion and cell sorting)**.**

-
-
-
-

6.4.0.1 References

Part IV

III) Cell differentiation

7 Cell differentiation introduction

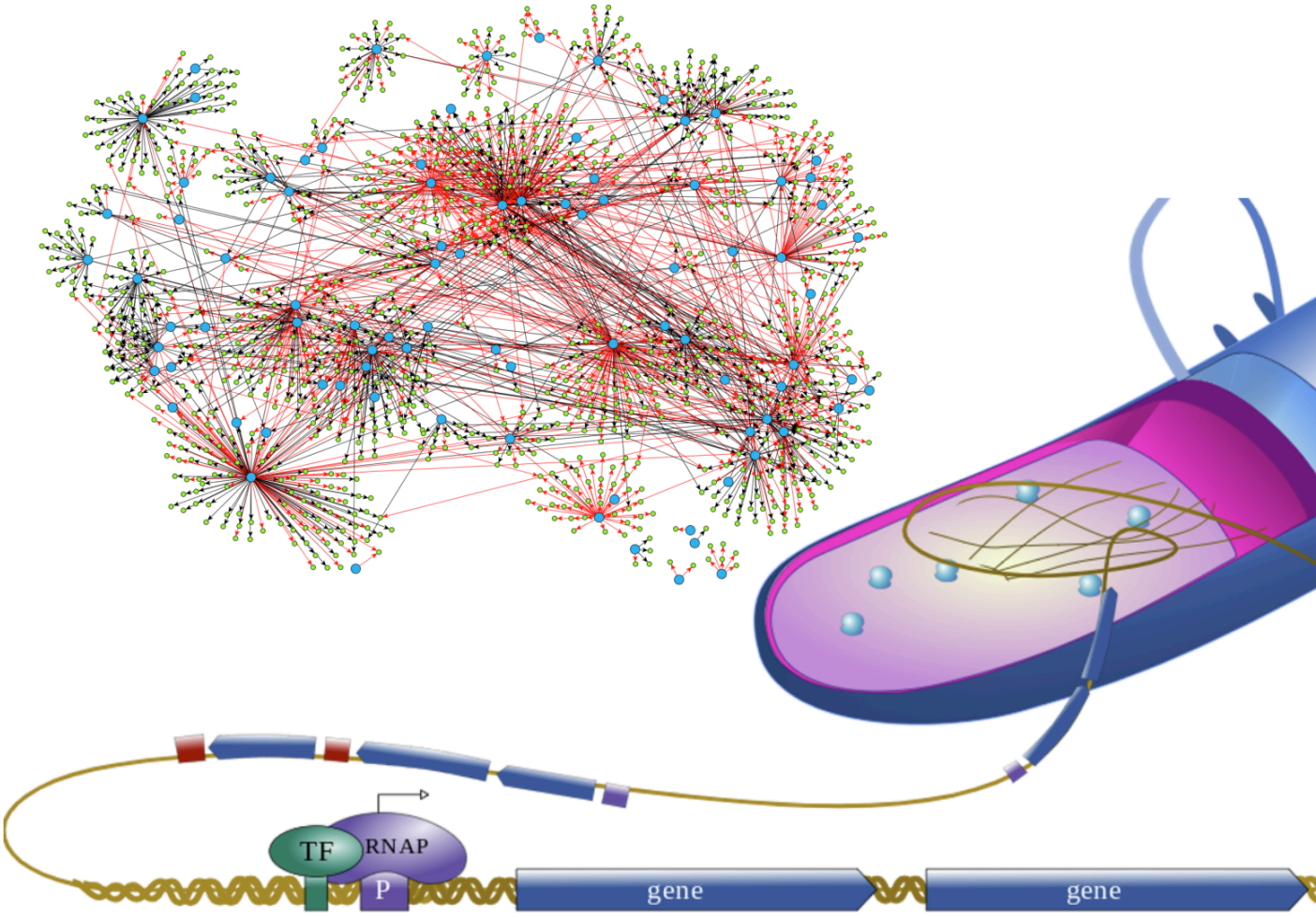


Figure 7.1: Gene regulation in a bacteria: from genes to networks. Credit: RegulonDB, and DOI: 10.1371/journal.pone.0106479

8 Gene regulation in time

8.1 Introduction

8.2 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 #Tip
    return(doutputC)

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

Exercise 8.1 (Mathematical thinking)**.**

```
# ODErun has three arguments: model, geneA, geneB
A=10
B=10
ODErun(ODEgeneRegulation,A,B)

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

```
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]
```

```
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 #Tip: Change this in exercise 8.3
```

```
ODEBooleanPlot(ode_output, bool_output)
```

Exercise 8.2 (Biology)**.**

Exercise 8.3 (Altorithmic thinking:)**.**

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

-

Exercise 8.4 (Biology and Algorithmic thinking)**.**

8.3 Gene regulatory network

8.3.1 Initial condition timecourse

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

    matrix[i+1, :] = rootNetwork(parameters) # we save the output in i+1

plotBooleanTimecourse(matrix,timesteps)
```

-

Exercise 8.5 (Algorithmic thinking).

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

Exercise 8.6 (Algorithmic thinking).

-
-

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

# your code

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

```
UMAPBoolean(attractors)
```

Exercise 8.7 (Biology / Conceptual thinking).

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

Exercise 8.8 (Biology / Conceptual thinking)**.**

Exercise 8.9 (Biology)**.**

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

Exercise 8.10 (Biology / conceptual thinking)**.**

8.4 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 8.11 (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, MG

# your code

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

Exercise 8.12 (Algorithmic thinking / biology)**.**

Exercise 8.13 (Biology)**.**

8.5 References

-
-

9 Gene regulation in space

9.1 Auxin does everything

9.2 Auxin does opposite things in the root

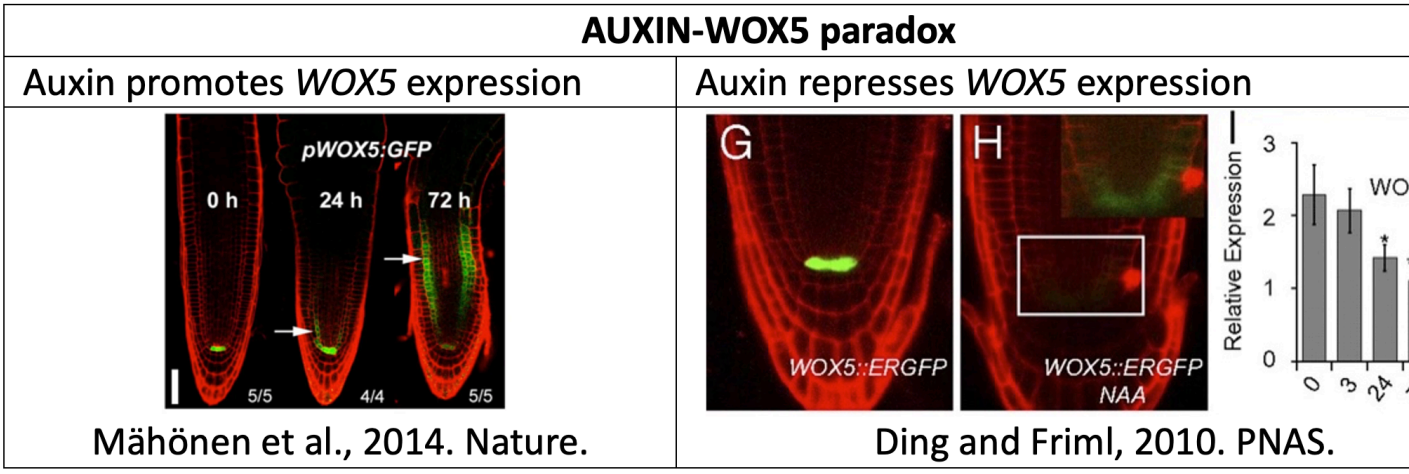


Figure 9.1: Figure 1. The right panel shows evidence of the positive effect of auxin on WOX5 expression (white arrows). The left panel shows evidence of auxin repression of WOX5 in the root tip (white rectangle), also shown with qPCR measurements of WOX5 mRNA in root-tip cells.

9.2.1 Biology / conceptual thinking

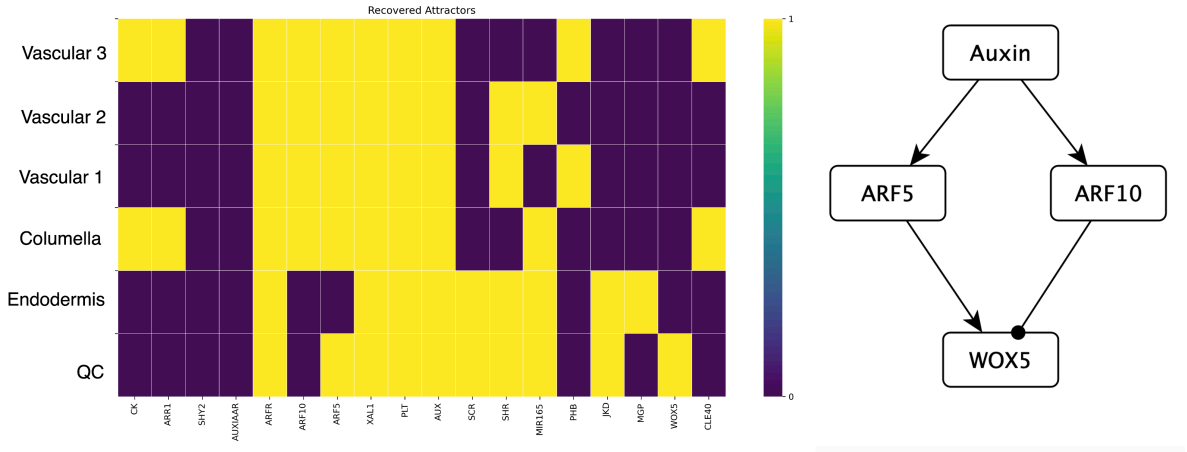


Figure 9.2: Figure 2 Auxin regulates WOX5 expression through different auxin response factors. These are expressed in specific cell types of the root

9.3 The model

9.3.1 Files

-

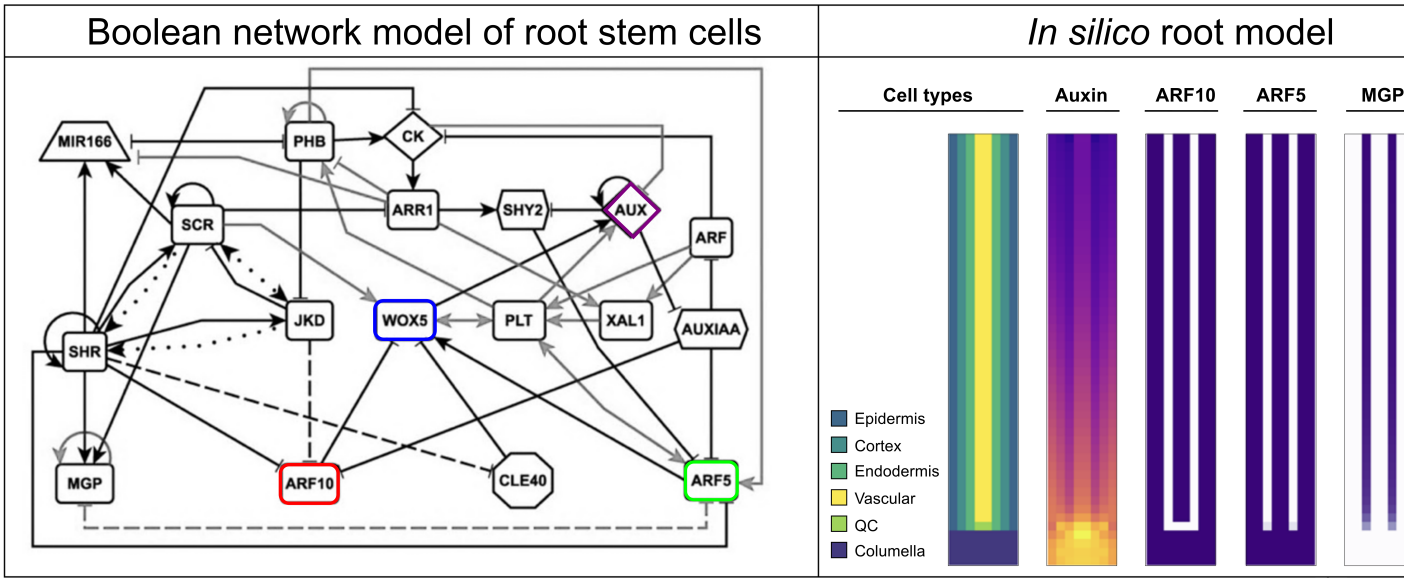


Figure 9.3: Figure 3. Ingredients of the in silico root model. 1) gene regulatory network, 2) multi-cellular tissue layout with polar auxin transport in the root tip

-
-
-
-
-

9.4 Questions

Exercise 9.1 (Algorithmic thinking)**.**

Exercise 9.2 (Biology)**.**

Exercise 9.3 (Biology & algorithmic thinking)**.**

-
-
-

Exercise 9.4 (Biology & algorithmic/mathematical thinking)**.**

-
-

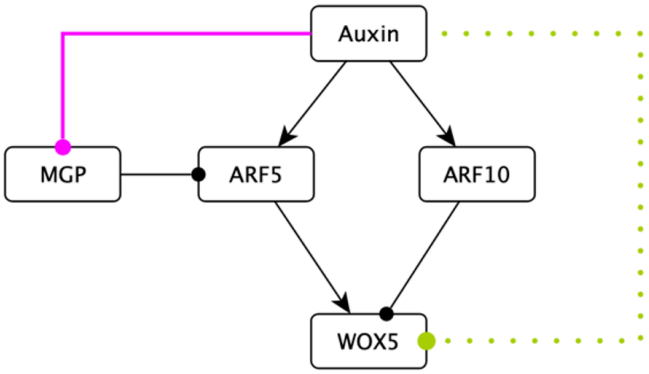


Figure 9.4: Testing two hypothesis: Auxin represses MGP expression. Extremely high auxin levels (easy to apply experimentally, but unrealistic *in vivo*) repress WOX5 expression.

Exercise 9.5 (Biological interpretation)**.**

-
-
-

Exercise 9.6 (Algorithmic thinking)**.**

Exercise 9.7 (Biology & algorithmic thinking)**.**

Exercise 9.8 (Biology & algorithmic thinking)**.**

Exercise 9.9 (Biology & conceptual thinking)**.**

Exercise 9.10 (Biological thinking)**.**

9.5 References

-
-
-

Part V

IV) Environment