# Lab 02. Cellular Automata for Excitable Media

Jordi-Lluís Figueras

Modelling Complex Systems, Spring 2026

## 1 Introduction

In this lab we study several classical examples of cellular automata. The first part concerns Conway's Game of Life, where suitable configurations can be used to transmit signals and implement logical gates. The second part concerns a simple three-state automaton that mimics the firing cycle of neurons. The third part studies the Greenberg-Hastings cellular automaton (GHCA), a standard model for excitation, recovery, and wave propagation.

These examples illustrate how simple local update rules can produce information processing, traveling structures, periodic behavior, and eventually repeating dynamics. The goal of the lab is to combine careful implementation with systematic numerical experimentation and clear interpretation of the observed patterns.

## 2 Exercises

Work in a clear and reproducible way. Your report should explain both the definition of the models and the conclusions of your numerical experiments. Unless explicitly stated otherwise, all exercises below are part of the lab.

### 1. Logical gates in the Game of Life

Implement configurations in Conway's Game of Life that realize the logical gates AND, OR, and NOT, following the constructions shown in the video from Lecture 8.

For each gate:

- identify clearly how the inputs are encoded,
- explain how the output is read,
- provide an initial configuration that implements the gate,
- submit three files, one for each gate, containing the list of live cells used as input for running the Game of Life,
- verify with simulations that the gate produces the correct output for all relevant input combinations.

Each of these files should list one point per line in the format

```
(0,0)
(1,0)
```

where each pair denotes the coordinates of a live cell in the initial configuration.

Include snapshots or short time sequences that make the signal propagation and gate behavior easy to follow.

## 2. Neuron cellular automaton

Implement a cellular automaton on an $N \times N$ grid with **periodic boundary conditions** that imitates neuron firing. Each grid point is a neuron and can be in one of three states:

- ready,
- firing,
- resting.

Use the following update rules:

- A ready neuron becomes firing at the next step if exactly two of its neighbors are firing.
- A firing neuron becomes resting at the next step.
- A resting neuron becomes ready at the next step.

Investigate the following tasks:

- construct an initial condition that produces a pattern moving forward in time,
- construct an initial condition that moves forward and creates new moving patterns along the way,
- construct an initial condition that leads to a periodic motion.

For each case, include figures or snapshots that make the observed behavior clear.

## 3. Periodicity in the Greenberg-Hastings cellular automaton

Write a Python program for the Greenberg-Hastings cellular automaton (GHCA) on an $n \times n$ grid with **periodic boundary conditions** and a prescribed excitation parameter $e$.

Your code should:

- generate a random initial configuration, and optionally read an initial configuration from a file,
- advance the configuration by $k$ time steps,
- save the resulting configuration to a file,
- draw the resulting configuration using colors that distinguish the different states.

Then address the following questions:

(a) What is the cardinality of the set of all possible initial configurations?

(b) On a finite grid, every orbit is eventually periodic. Write a program that determines the eventual period of the evolution for a given initial configuration.

(c) Find an initial configuration whose eventual period satisfies $m \geq 2$.

When discussing part (b), distinguish clearly between the transient time before the repetition starts and the period of the repeating cycle.

## 3 What to submit

Submit:

- a single explanatory report in PDF format,
- the Python files used for the computations,
- figures or movie files, if you decide to include dynamic visualizations,
- in case LLM tools were used, an appendix reporting the most important prompts used for the project.

You may use external sources, but they must be cited correctly. Use Python for all implementations.