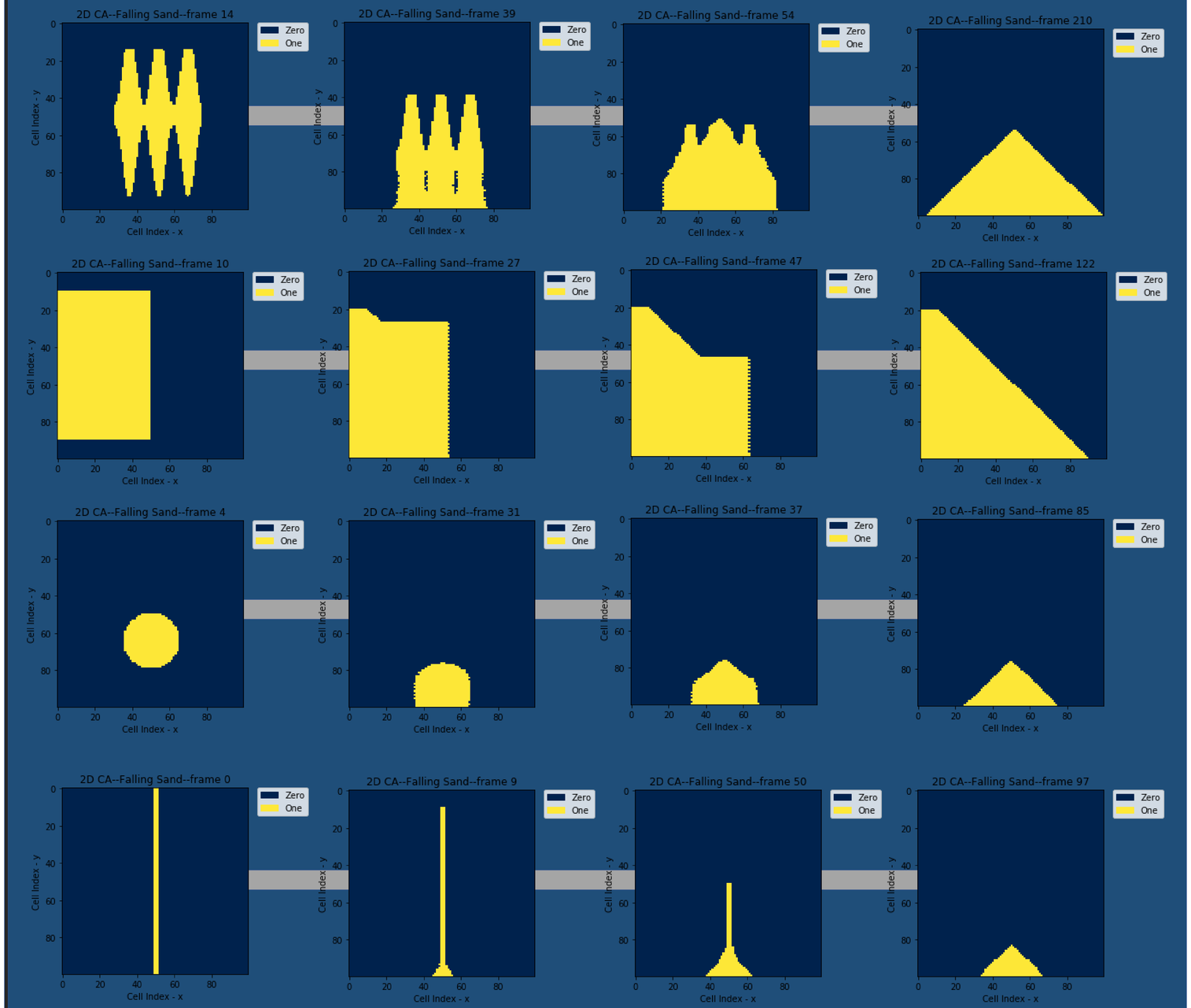
Harry Moore

From emergent integer sequences to falling sand: exploring applications of cellular automata in one and two dimensions

Dropping objects made of sand using cellular automata



Cellular Automata (CA) are, at their simplest, grids of cells which can either be 'on' or 'off' and which update in discrete timesteps based on fixed rules

One-Dimensional Cellular Automata:

- A 1D CA is just a single row of cells, each either 0 or 1.
- In the simplest rules for updating each cell, we consider the neighborhood consisting of that cell plus the two adjacent cells.
- There are eight possible such neighborhoods, from 111 to 000, so for a given rule, each one is mapped to an output of either 0 or 1.
- **Example:**

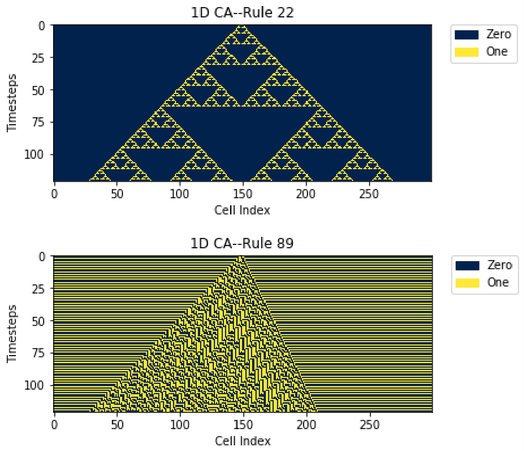
111, 110, 101, 100, 011, 010, 001, 000

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

0 0 1 0 1 1 0 1

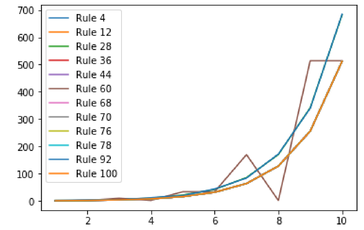
In binary, these ordered outputs express the integer 45. For this reason, this mapping would be called "Rule 45"

- We can visualize the evolution of a 1D CA as a 2D grid, where each row is an updated version of the same set of cells. When viewed in this way, certain integer rules produce fractal patterns, such as the following:



References:
1. Shiffman, Daniel. "THE NATURE OF CODE." *The Nature of Code*, https://natureofcode.com/book/chapter-7-cellular-automata/.
2. Wolf-Gladrow, Dieter A. *Lattice Gas Cellular Automata and Lattice Boltzman Methods--An Introduction*. Alfred Wegener Institute for Polar and Marine Research, 2005.

- If we read each row of a 1D CA visualization as an integer in binary, we get a sequence.
- Certain rules produce well-known sequences (e.g. Rule 28 gives the Jacobsthal numbers)
- For several Rules which give nondecreasing sequences, we compare the gaps between consecutive numbers in the following plot:

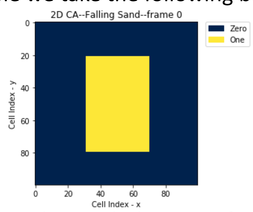


- The sequence for Rule 68 is found to behave uniquely among these, its gaps not strictly increasing

2D Cellular Automata:

- Now a square grid of cells.
- To update each cell, we might look at cells in the 3x3 neighborhood around it (e.g. Conway's "Game of Life")
- We might also use 2D CA to model physical systems. Here we look at falling grains of sand. Our rules for updating the grid to simulate falling sand are the following:
  1. A cell which is on will 'move' straight down if possible.
  2. Otherwise, it will move either down-left or down-right, deciding which way randomly if both options are available.
  3. Otherwise, it will stay put.

- This results in the types of deformities on the left.
- We might also be interested in the sum of every particle's height over time (as this would correspond to potential energy in a physical model). For an example we take the following block of sand:



- When it is dropped, the sum of particle heights behaves as follows: