Cellular Automata

Joy Kimmel

Gordon College

May 7, 2014

Overview

- Introduction

Overview

- Introduction
- "Game Of Life"
  - Rules
  - Code
  - Results

Overview

- Introduction
- "Game Of Life"
  - Rules
  - Code
  - Results
- Sandpile Model
  - Algorithm / Code
  - Results
  - Properties

Overview

- Introduction
- "Game Of Life"
  - Rules
  - Code
  - Results
- Sandpile Model
  - Algorithm / Code
  - Results
  - Properties
- Conclusion

What is Cellular Automata?

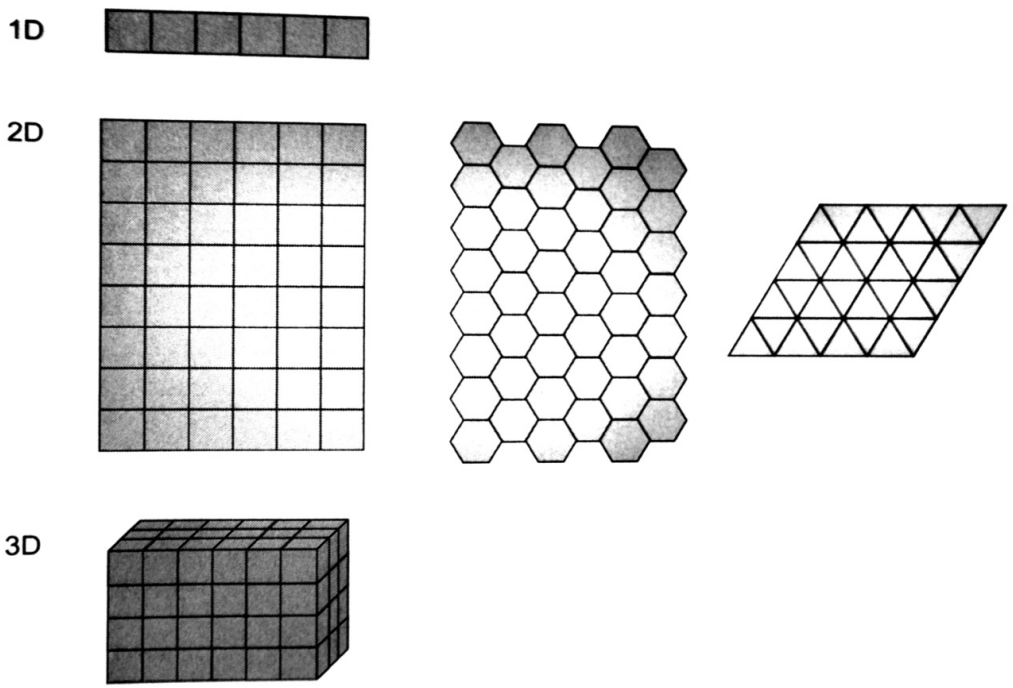
- Approach to modeling the time evolution of a system
- Usually deterministic rules
- Many applications

Definition

cellular: "discrete space and local nature"
automata: "rule-based automatic time evolution"

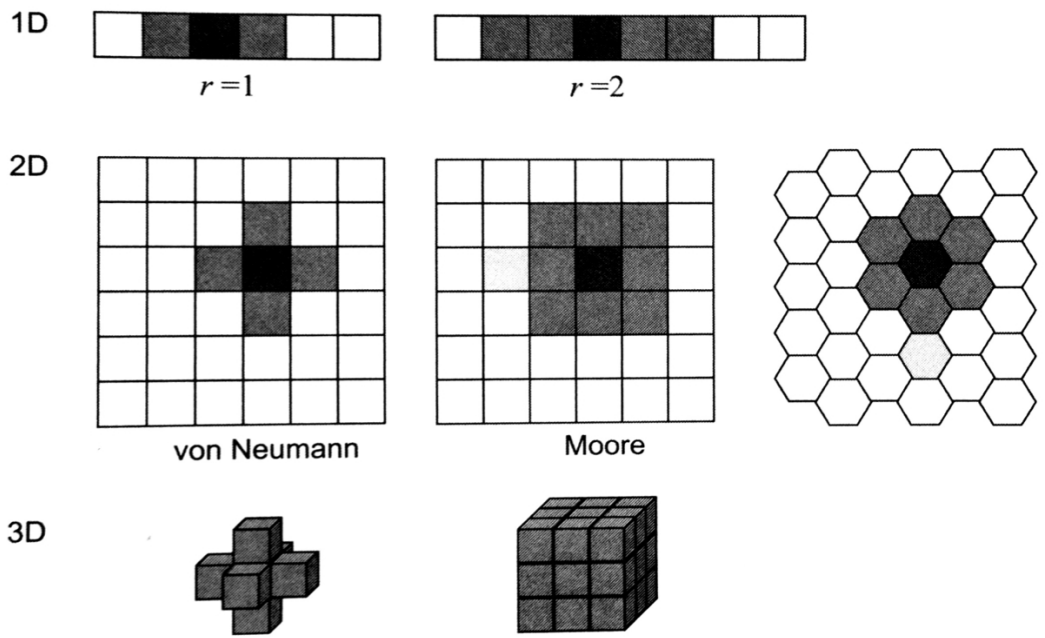
General Steps

- Choose cellular space (LxL lattice)



- Assign the state set: S
  - For the simplest example: a block can either be 'occupied' or 'unoccupied'

- Assign the state set: S
  - For the simplest example: a block can either be 'occupied' or 'unoccupied'
- Choose neighbors
  - von Neumann
  - Moore



- Choose transition rules
  - Number of possible transition rules (TR) grows with possible states (k) and number of neighbors (n)
  - Follows the equation: k^{k^n}
  - With k = 2 possible states, n = 3 neighbors: $TR = 2^{2^3} = 256$
  - With k = 3 possible states, n = 3 neighbors: $TR = 3^{3^3} = 7,625,597,484,987$

- Choose transition rules
  - Number of possible transition rules (TR) grows with possible states (k) and number of neighbors (n)
  - Follows the equation: k^{k^n}
  - With k = 2 possible states, n = 3 neighbors: TR $= 2^{2^3} = 256$
  - With k = 3 possible states, n = 3 neighbors: TR $= 3^{3^3} = 7,625,597,484,987$
- Choose a time variable: continuous or discrete

- Choose transition rules
  - Number of possible transition rules (TR) grows with possible states (k) and number of neighbors (n)
  - Follows the equation: k^{k^n}
  - With k = 2 possible states, n = 3 neighbors: TR $= 2^{2^3} = 256$
  - With k = 3 possible states, n = 3 neighbors: TR $= 3^{3^3} = 7,625,597,484,987$
- Choose a time variable: continuous or discrete
- Assign boundary conditions

- Choose transition rules
  - Number of possible transition rules (TR) grows with possible states (k) and number of neighbors (n)
  - Follows the equation: k^{k^n}
  - With k = 2 possible states, n = 3 neighbors: $TR = 2^{2^3} = 256$
  - With k = 3 possible states, n = 3 neighbors: $TR = 3^{3^3} = 7,625,597,484,987$
- Choose a time variable: continuous or discrete
- Assign boundary conditions
- Assign initial conditions

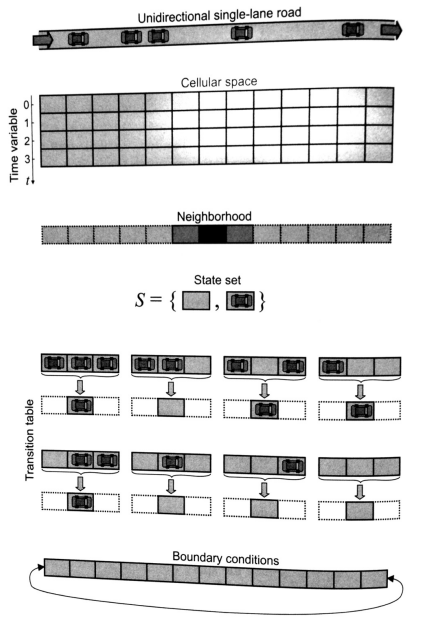
- Choose transition rules
  - Number of possible transition rules (TR) grows with possible states (k) and number of neighbors (n)
  - Follows the equation: k^{k^n}
  - With k = 2 possible states, n = 3 neighbors: TR $= 2^{2^3} = 256$
  - With k = 3 possible states, n = 3 neighbors: TR $= 3^{3^3} = 7,625,597,484,987$
- Choose a time variable: continuous or discrete
- Assign boundary conditions
- Assign initial conditions
- Assign a stopping condition

- Choose transition rules
  - Number of possible transition rules (TR) grows with possible states (k) and number of neighbors (n)
  - Follows the equation: k^{k^n}
  - With k = 2 possible states, n = 3 neighbors: $TR = 2^{2^3} = 256$
  - With k = 3 possible states, n = 3 neighbors: $TR = 3^{3^3} = 7,625,597,484,987$
- Choose a time variable: continuous or discrete
- Assign boundary conditions
- Assign initial conditions
- Assign a stopping condition
- Repeat until the stopping condition is met

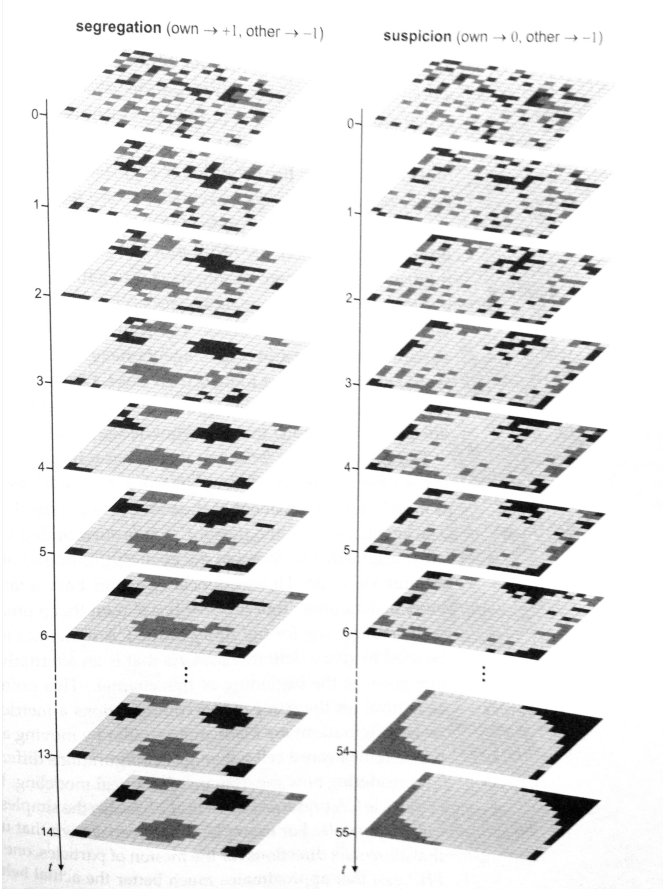
Concrete Examples

- "Game Of Life"
- Forest Fires
- Maze Solution
- Traffic Simulation
- Social Dynamics

Traffic Simulation



Social Dynamics



Background / History

- Introduced by Conway in 1970
- 8 neighbors (Moore)
- A cell is either "alive" or "dead"
- Discrete time steps

Simple Algorithm

- If 2 neighbors are alive: the state of the cell in the next time step is unchanged

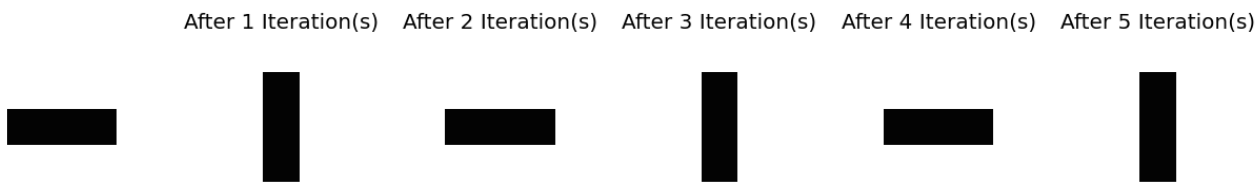
Simple Algorithm

- If 2 neighbors are alive: the state of the cell in the next time step is unchanged
- If 3 neighbors are alive: the state of the cell is alive no matter its current state

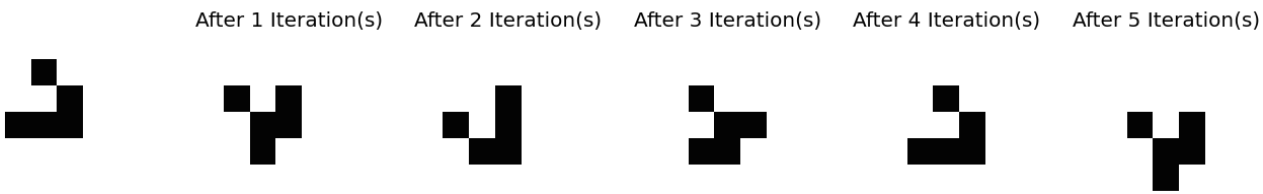
Simple Algorithm

- If 2 neighbors are alive: the state of the cell in the next time step is unchanged
- If 3 neighbors are alive: the state of the cell is alive no matter its current state
- Otherwise the cell is dead

Simple Patterns: Blinker



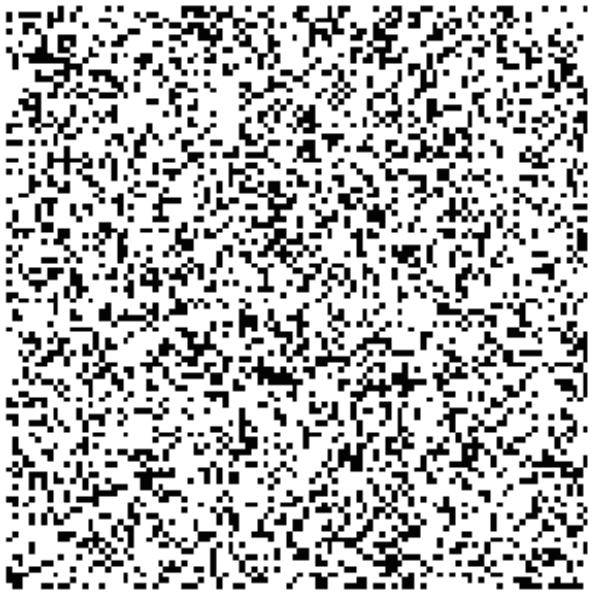
Simple Patterns: Glider

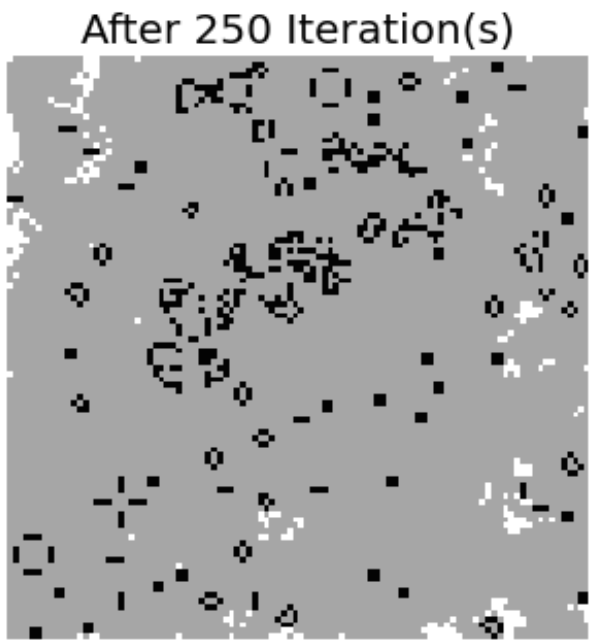


Chaotic Behavior

- 100x100 Lattice
- Randomly generated an initial condition [with a probability of being alive: 0.3]
- White: never alive, Gray: once alive - now dead, Black: dead
- See code (via handout)

Chaotic Behavior

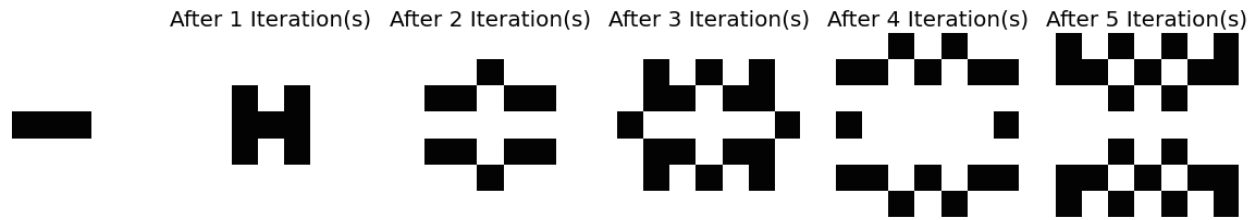




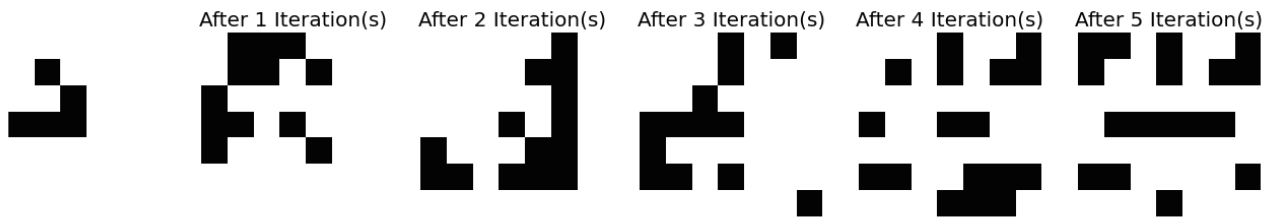
Simple Algorithm - Generalized

- If n neighbors are alive: the state of the cell in the next time step is unchanged
- If n+1 neighbors are alive: the state of the cell is alive no matter its current state
- Otherwise the cell is dead

"Blinker" at n = 1



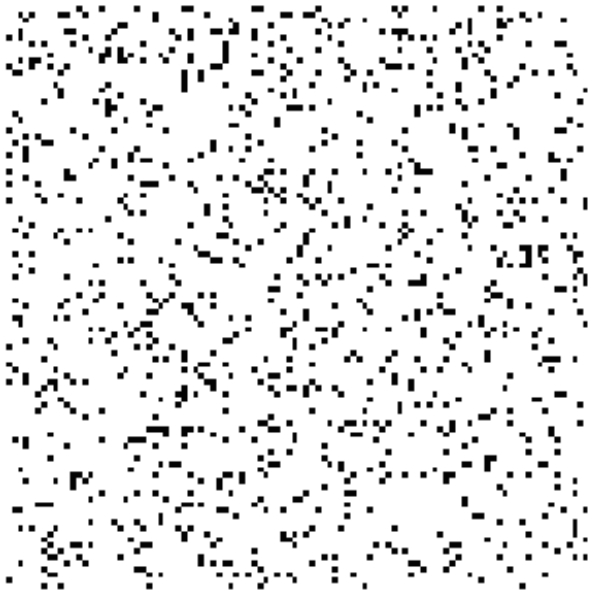
"Glider" at n = 1


After 1 Iteration(s)
After 2 Iteration(s)
After 3 Iteration(s)
After 4 Iteration(s)
After 5 Iteration(s)

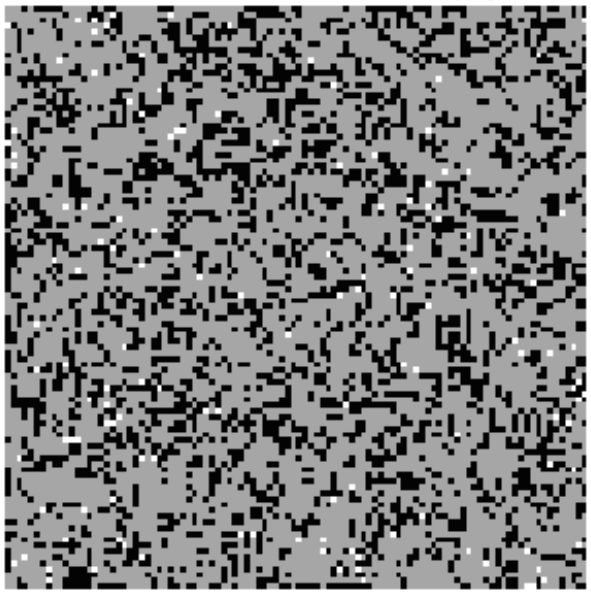
Chaotic Behavior

- 100x100 Lattice
- Randomly generated an initial condition [with a probability of being alive: 0.1]
- White: never alive, Gray: once alive - now dead, Black: dead

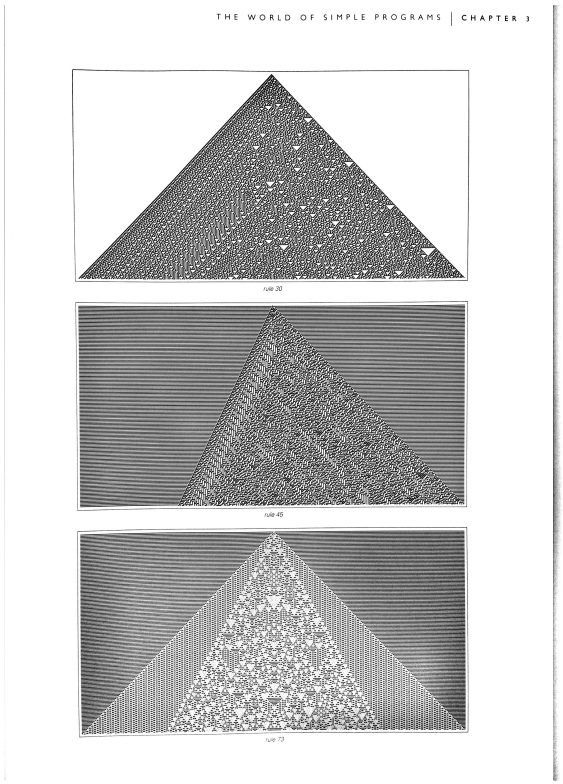
Chaotic Behavior

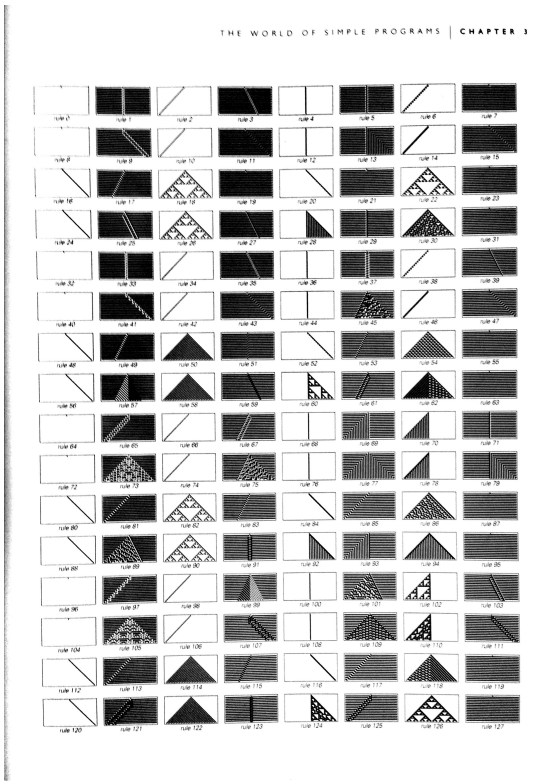


After 15 Iteration(s)



Additional Rules and Results





Background

- Imagine the dynamics of sandpile

Background

- Imagine the dynamics of sandpile
- Drop one grain of sand in at a time

Background

- Imagine the dynamics of sandpile
- Drop one grain of sand in at a time
- Eventually the pile grows, until it reaches a point where the cells tumble

Background

- Imagine the dynamics of sandpile
- Drop one grain of sand in at a time
- Eventually the pile grows, until it reaches a point where the cells tumble
- Exhibits self-organized criticality
  - A critical state develops automatically

Background

- Imagine the dynamics of sandpile
- Drop one grain of sand in at a time
- Eventually the pile grows, until it reaches a point where the cells tumble
- Exhibits self-organized criticality
  - A critical state develops automatically
- Back, Tang and Weisenfeld (1977&1978)

Conditions

- Consider a 100x100 square lattice
- 4 neighbors (von Neumann)
- Free boundary
- Continuous time variable

Transition Rules

- Each cell i holds a value z_i, if $z_i > z_c$, the cell will 'tumble'

Transition Rules

- Each cell i holds a value z_i, if $z_i > z_c$, the cell will 'tumble'
- When a cell tumbles, it affects its neighbors z_j's values
  - $z_i - 4$
  - $z_j + 1$

Transition Rules

- Each cell i holds a value z_i, if $z_i > z_c$, the cell will 'tumble'
- When a cell tumbles, it affects its neighbors z_j's values
  - $z_i - 4$
  - $z_j + 1$
- Stable z_i values are $0, 1, 2, 3$ where $z_c = 3$

Transition Rules

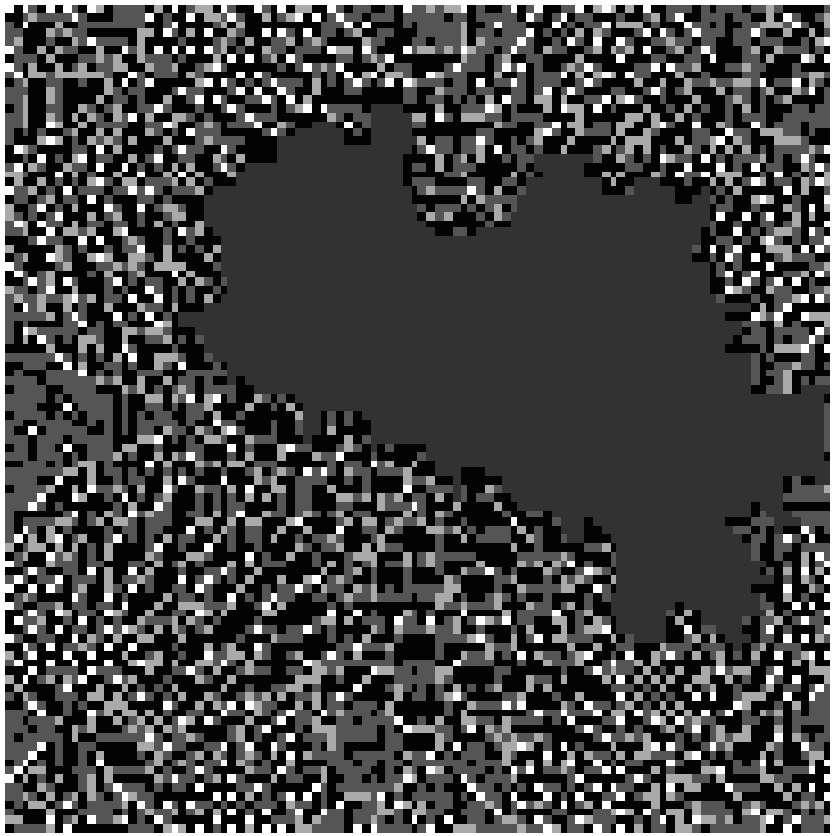
- Each cell i holds a value z_i, if $z_i > z_c$, the cell will 'tumble'
- When a cell tumbles, it affects its neighbors z_j's values
  - $z_i - 4$
  - $z_j + 1$
- Stable z_i values are $0, 1, 2, 3$ where $z_c = 3$
- Two initial states considered: homogeneous or heterogeneous

Avalanches

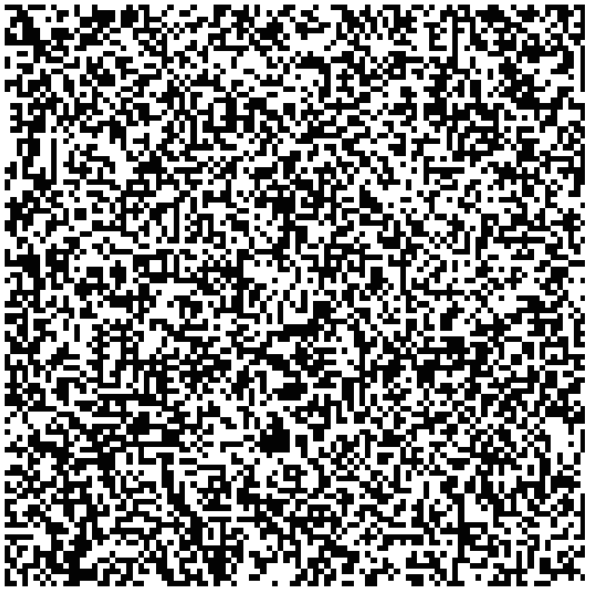
- A cell is considered apart of an avalanche if it tumbles at least once
- Size is the number of cells included in the avalanche
- Duration is the number of times the computer iterates through the entire lattice
- See code (handout)

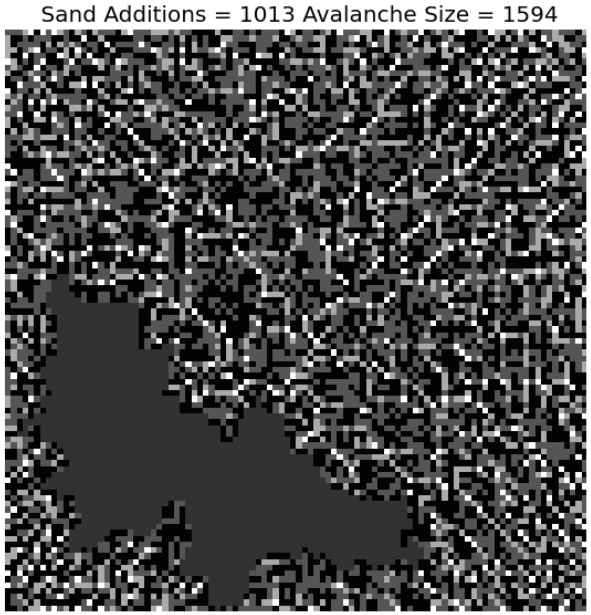
Homogeneous Initial State

Sand Additions = 3102 Avalanche Size = 257

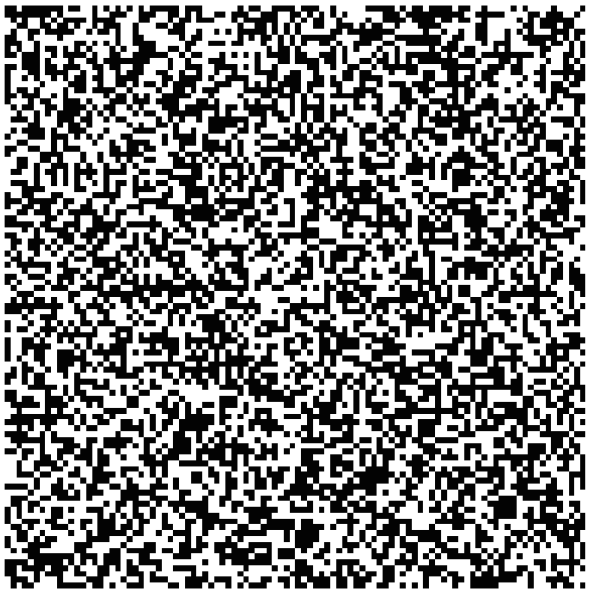


Heterogeneous Initial State





Heterogeneous Initial State



Distribution of Avalanche Sizes

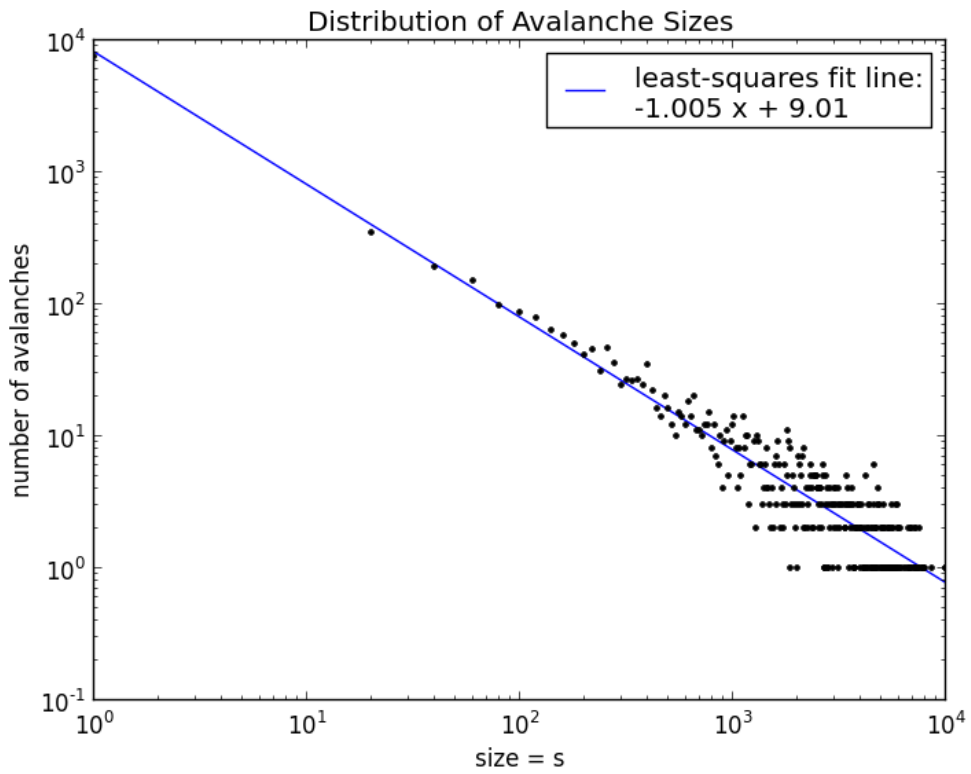
- Avalanche frequency depends on their size by a power law

Distribution of Avalanche Sizes

- Avalanche frequency depends on their size by a power law
- $N(s) \sim s^{-\tau}$ where $\tau \approx 1.0$

Distribution of Avalanche Sizes

- Avalanche frequency depends on their size by a power law
- $N(s) \sim s^{-\tau}$ where $\tau \approx 1.0$

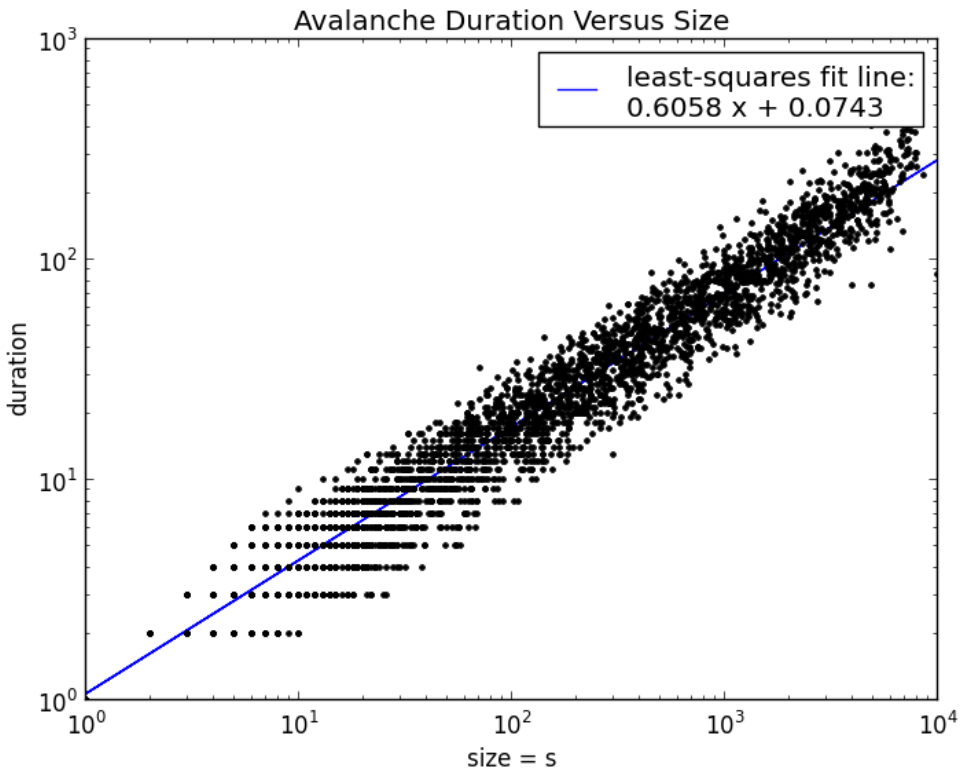


Avalanche Duration Versus Size

- $T(s) \sim s^a$ where $a \approx 0.65$

Avalanche Duration Versus Size

- $T(s) \sim s^a$ where $a \approx 0.65$

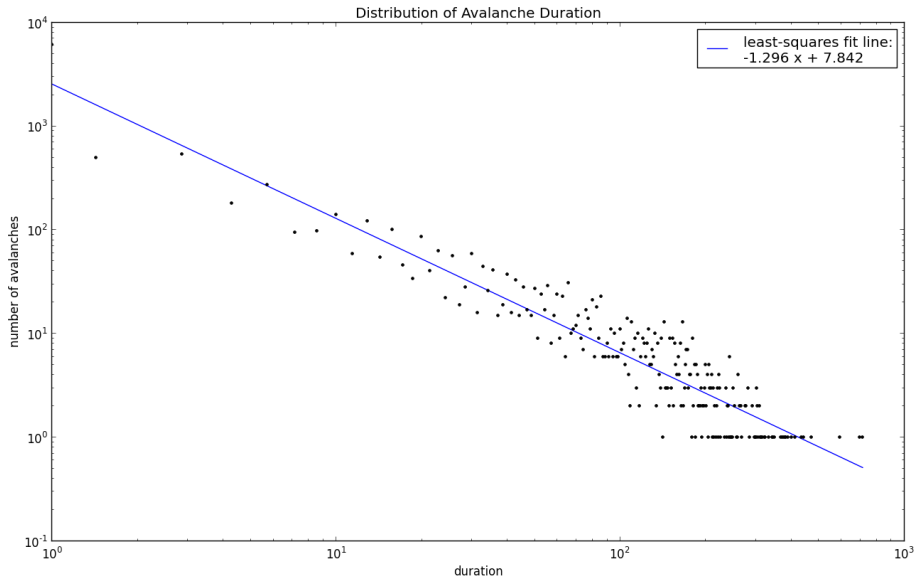


Frequency of Avalanche Duration

- $N(t) \sim t^{-b}$ where $b \approx 1.1$

Frequency of Avalanche Duration

- $N(t) \sim t^{-b}$ where $b \approx 1.1$

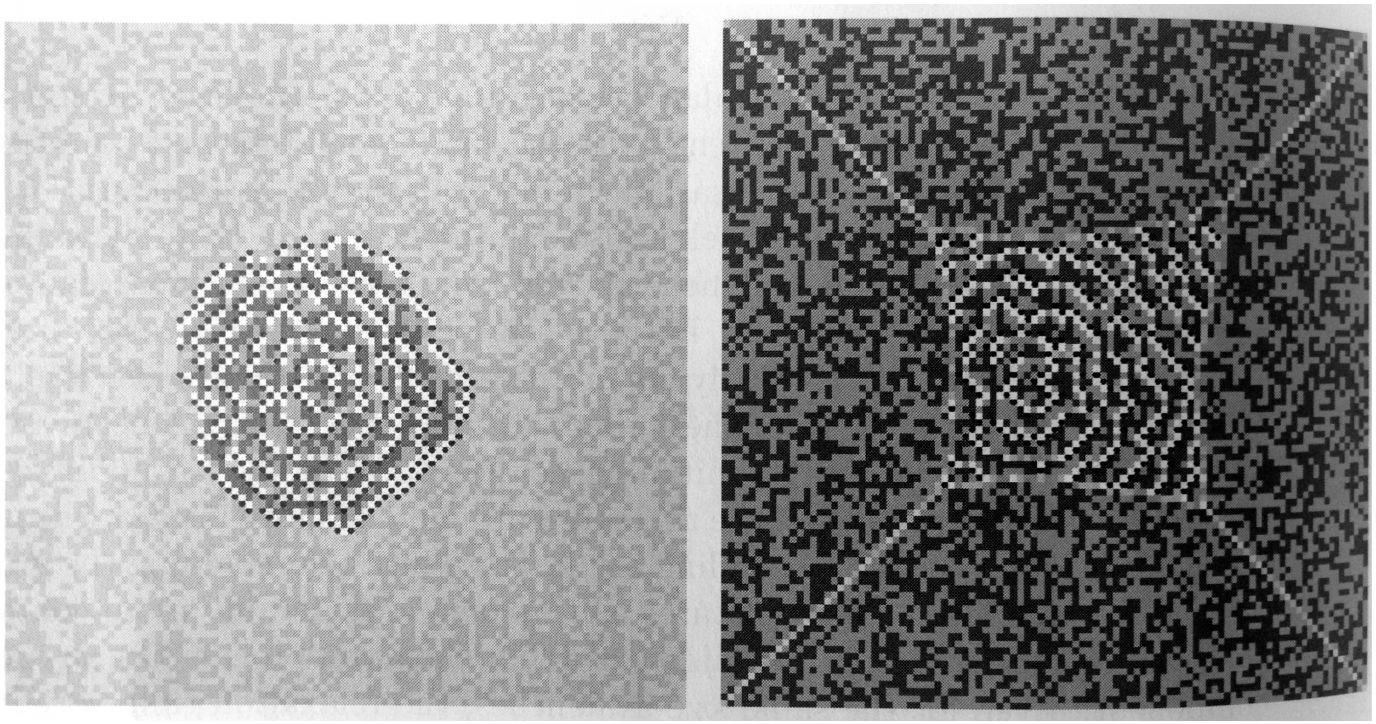


Relaxation of a System

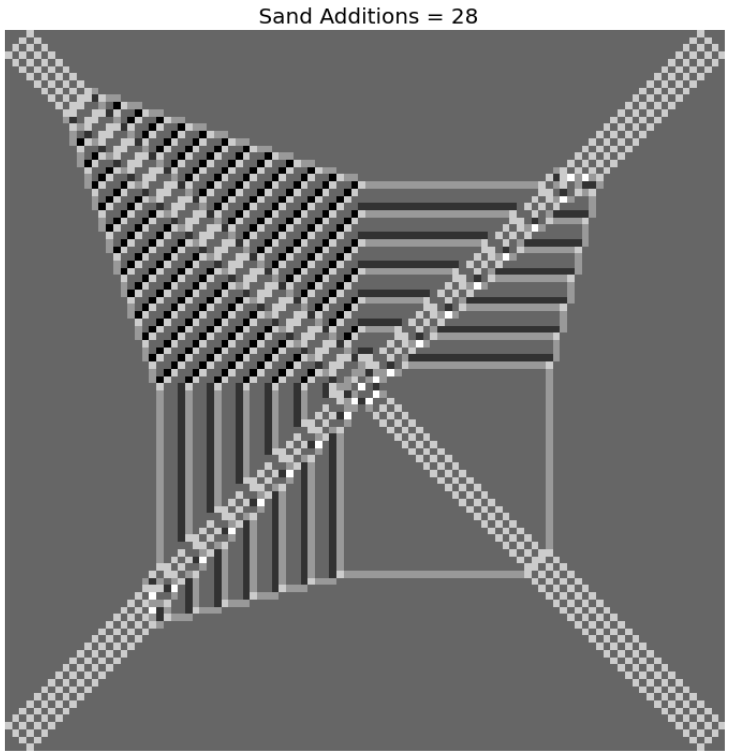
- Consider an avalanche's relaxation period, and see patterns it produces

Relaxation of a System

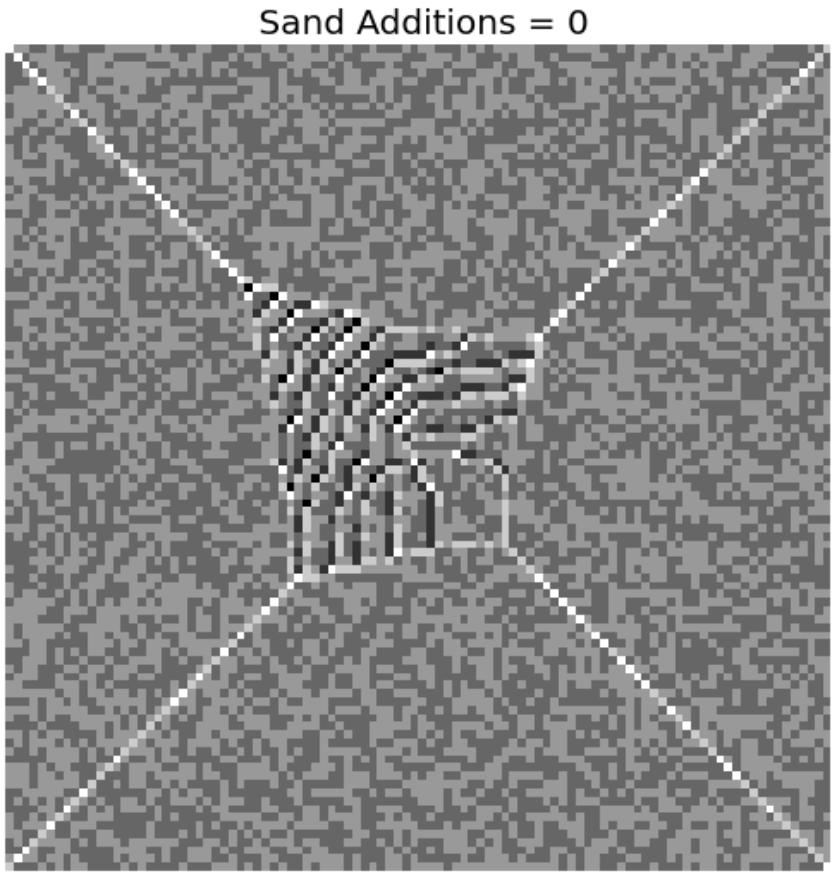
- Consider an avalanche's relaxation period, and see patterns it produces



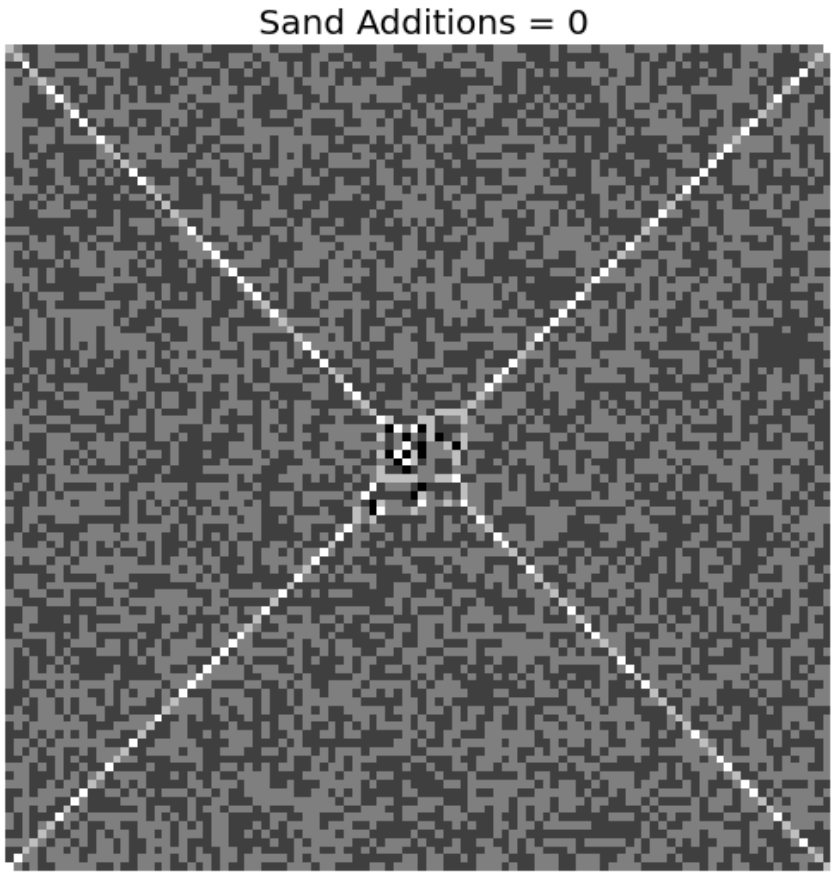
My Snapshots



My Snapshots



My Snapshots



Future Questions

- Vary boundary conditions
- Introduce damage
- 3D avalanche

Works Cited

- Computational Physics (2nd addition) by Nicholas Giordano and Hisao Nakanishi
- Bio-Inspired Artificial Intelligence by Dario Floreano and Claudio Mattiussi