CITS4403
Computational Modelling

Week 12 Lecture: Self-organized Criticality

Dr. Siwen Luo

Semester 2, 2024
School of Computer Science,
University of Western Australia

LECTURE PLAN

THE UNIVERSITY OF WESTERN AUSTRALIA

W1 Lecture: Self-organized Criticality

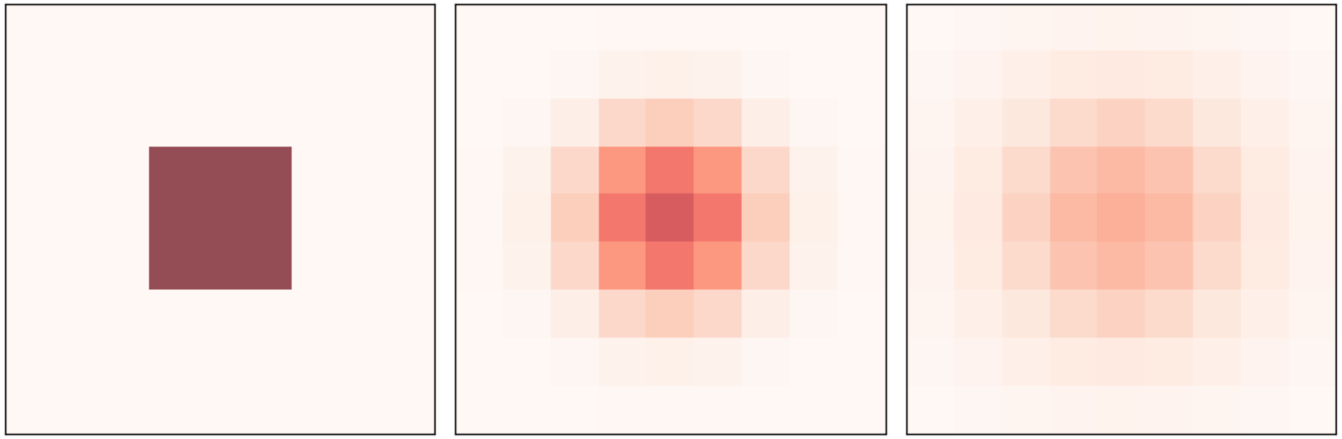
1. Recap
2. Percolation model
3. Critical systems
4. Sand Piles
   - Heavy-tailed Distribution
   - Fractal Geometry
   - Pink Noise

Recap

Diffusion

2-D Cellular Automata with state of each cell is a continuous quantity (usually between 0 and 1).

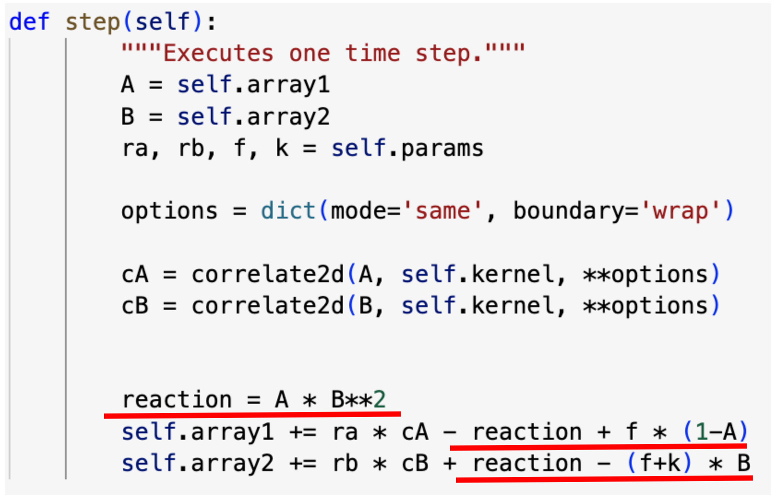
Diffusion process allows concentration to spill over to neighbours.



1 Recap

Reaction Diffusion

A simulation of two chemicals reacting and diffusing on a 2-D grid. Reaction consumes chemical A and produces chemical B.

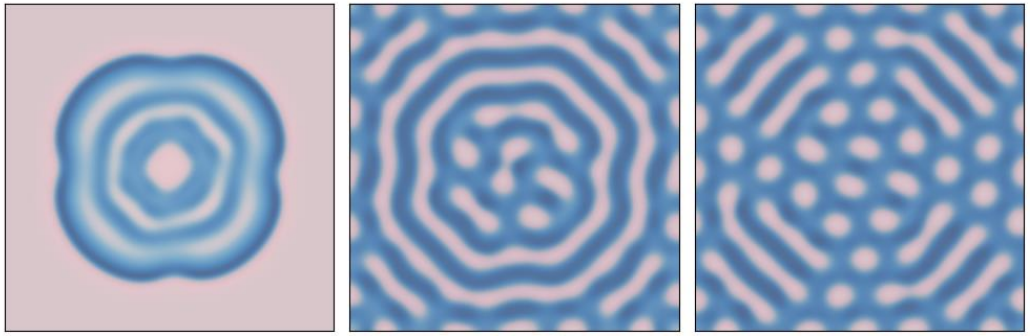


```python
def step(self):
        """Executes one time step."""
        A = self.array1
        B = self.array2
        ra, rb, f, k = self.params

        options = dict(mode='same', boundary='wrap')

        cA = correlate2d(A, self.kernel, **options)
        cB = correlate2d(B, self.kernel, **options)


        reaction = A * B**2
        self.array1 += ra * cA - reaction + f * (1-A)
        self.array2 += rb * cB + reaction - (f+k) * B
```

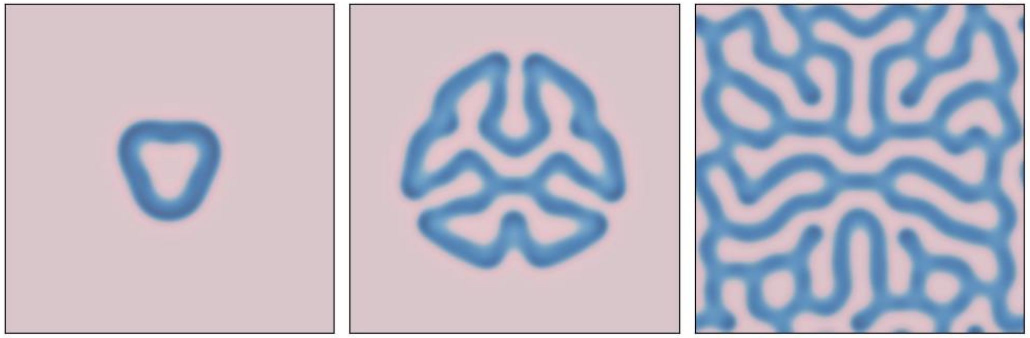
Recap

Reaction Diffusion

A simulation of two chemicals reacting and diffusing on a 2-D grid. Reaction consumes chemical A and produces chemical B.



Feed rate: 0.035, kill rate: 0.057



Feed rate: 0.055, kill rate: 0.062

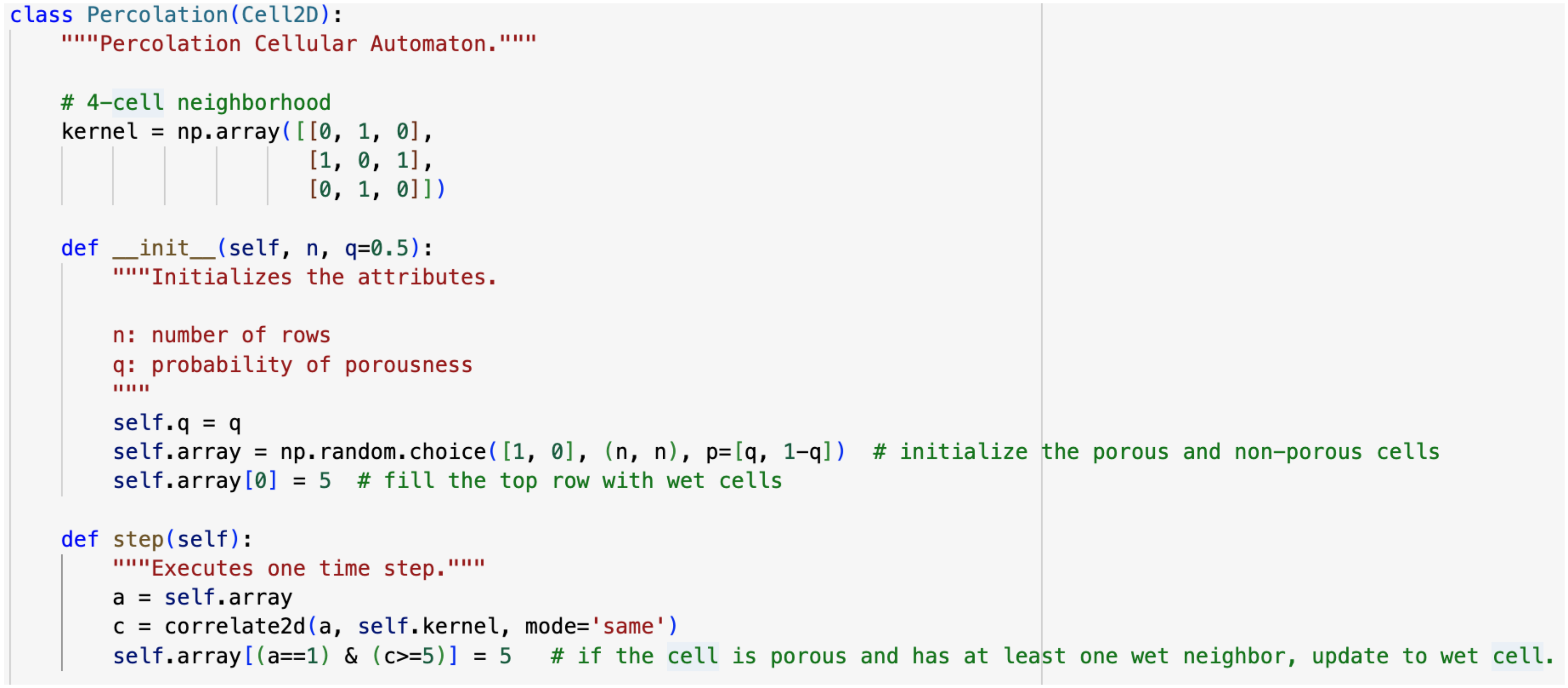
Percolation

Percolation is a process in which a fluid flows through a semi-porous material.

- Each cell is either “porous” with probability q or “non-porous” with probability $1 - q$
- When the simulation begins, all cells are considered “dry” except the top row, which is “wet”.
- During each time step, if a porous cell has at least one wet neighbour, it becomes wet. Non-porous cells stay dry.
- The simulation runs until it reaches a “fixed point” where no more cells change state.

2 Percolation

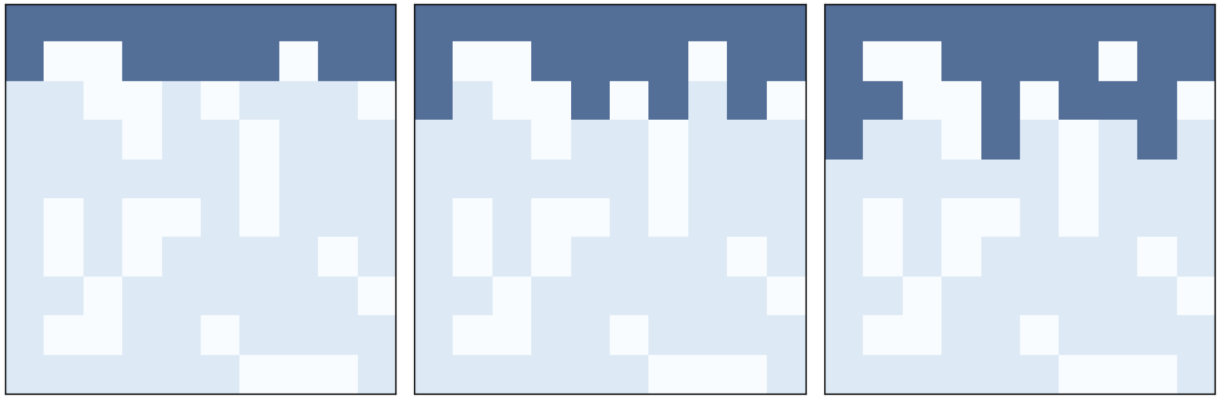
Percolation is a process in which a fluid flows through a semi-porous material.



```python
class Percolation(Cell2D):
    """Percolation Cellular Automaton."""

    # 4-cell neighborhood
    kernel = np.array([[0, 1, 0],
                       [1, 0, 1],
                       [0, 1, 0]])

    def __init__(self, n, q=0.5):
        """Initializes the attributes.

        n: number of rows
        q: probability of porousness
        """
        self.q = q
        self.array = np.random.choice([1, 0], (n, n), p=[q, 1-q])  # initialize the porous and non-porous cells
        self.array[0] = 5  # fill the top row with wet cells

    def step(self):
        """Executes one time step."""
        a = self.array
        c = correlate2d(a, self.kernel, mode='same')
        self.array[(a==1) & (c>=5)] = 5   # if the cell is porous and has at least one wet neighbor, update to wet cell.
```

Percolation

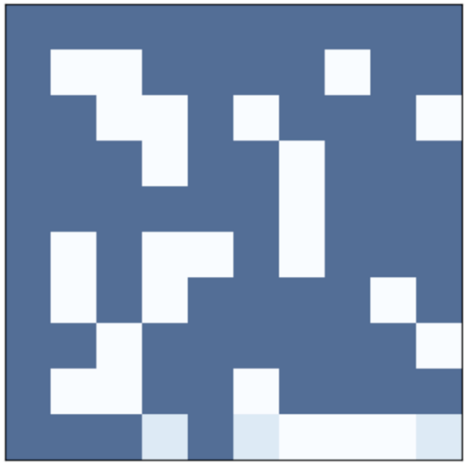
Percolation is a process in which a fluid flows through a semi-porous material.



The first three steps of a percolation model with $n = 10$ and $q = 0.7$

Percolation

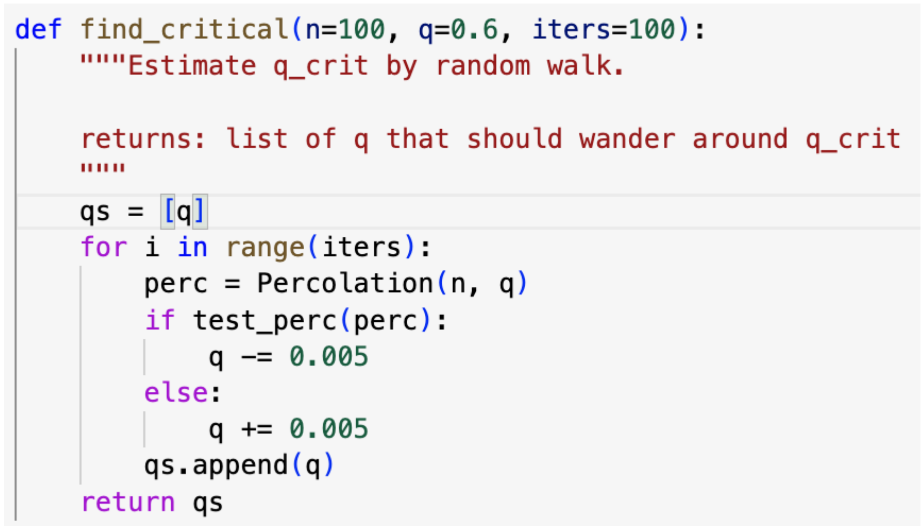
Percolation is a process in which a fluid flows through a semi-porous material.



After 12 steps of a percolation model with $n = 10$ and $q = 0.7$

2 Percolation

Estimate the critical value for percolation clustering



```python
def find_critical(n=100, q=0.6, iters=100):
    """Estimate q_crit by random walk.

    returns: list of q that should wander around q_crit
    """
    qs = [q]
    for i in range(iters):
        perc = Percolation(n, q)
        if test_perc(perc):
            q -= 0.005
        else:
            q += 0.005
        qs.append(q)
    return qs
```

There is a rapid change of the probability of a percolation clustering at the critical value.

THE UNIVERSITY OF WESTERN AUSTRALIA

Phase Change

The rapid change in behaviour near the critical value is called a phase change by analogy with phase changes in physical systems, like the way water changes from liquid to solid at its freezing point.

A wide variety of systems display a common set of behaviours and characteristics when they are at or near a critical point. These behaviours are known collectively as **critical phenomena**.

Critical Systems

Critical Systems demonstrate common behaviours:

Fractal geometry:

- freezing water tends to form fractal patterns, including snowflakes and other crystal structures.
- Fractals are characterized by **self-similarity**; that is, parts of the pattern are similar to scaled copies of the whole.

Heavy-tailed distributions:

- In freezing water, the distribution of crystal sizes is characterized by a power law.

Pink Noise:

- Complex signals can be decomposed into their frequency components. In pink noise, low-frequency components have more power than high-frequency components.

Critical Systems

Critical systems are usually unstable, but Many natural systems exhibit characteristic behaviours of criticality.

Self-organized Criticality (SOC), where “self-organized” means that from any initial condition, the system moves toward a critical state, and stays there, without external control.

Bak, Tang and Wiesenfeld in 1987, tested this claim through a **sand pile** model, a simple example of a broad category of models for self-organised criticality.

Sand Piles

The sand pile model is a 2-D cellular automaton where the state of each cell represents the slope of a part of a sand pile.

- During each time step, each cell is checked to see whether it exceeds a critical value, K, which is usually 3;
- If so, it “topples” and transfers sand to **four** neighbouring cells -- *the slope of the cell is decreased by 4, and each of the neighbours is increased by 1.*
- Each cell on the boundary has a constant slope 0, so the excess spills over the edge.

Sand Piles

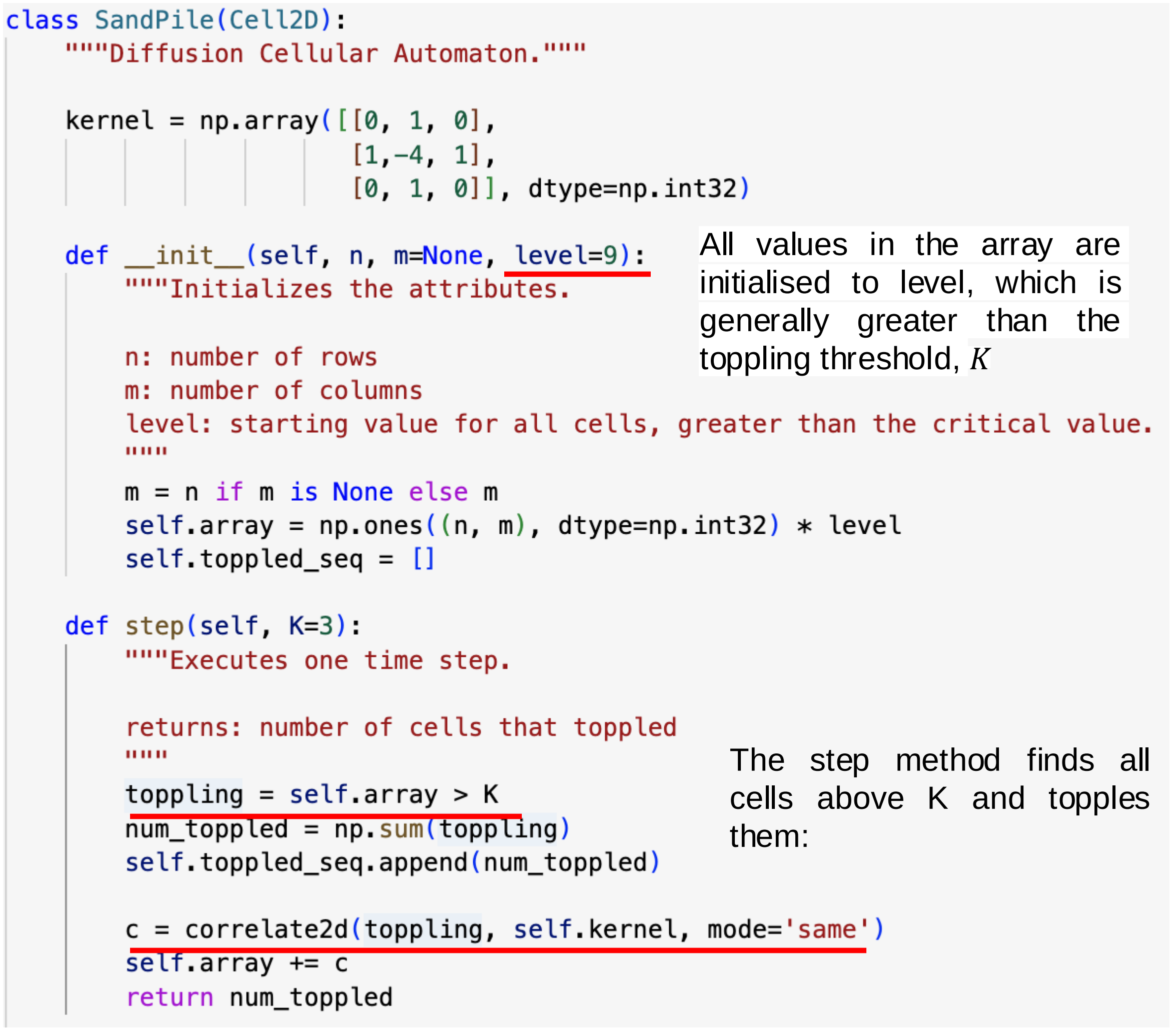
What to do in the experiments:

1. All cells are initialized at a level greater than K and run the model until it stabilizes.

2. Observe the effect of small perturbations:
   - choose a cell at random
   - increment its value by 1
   - and run the model again until it stabilizes.

3. For each perturbation, measure T, the number of time steps the pile takes to stabilize, and S, the total number of cells that topple.

3 Sand Piles

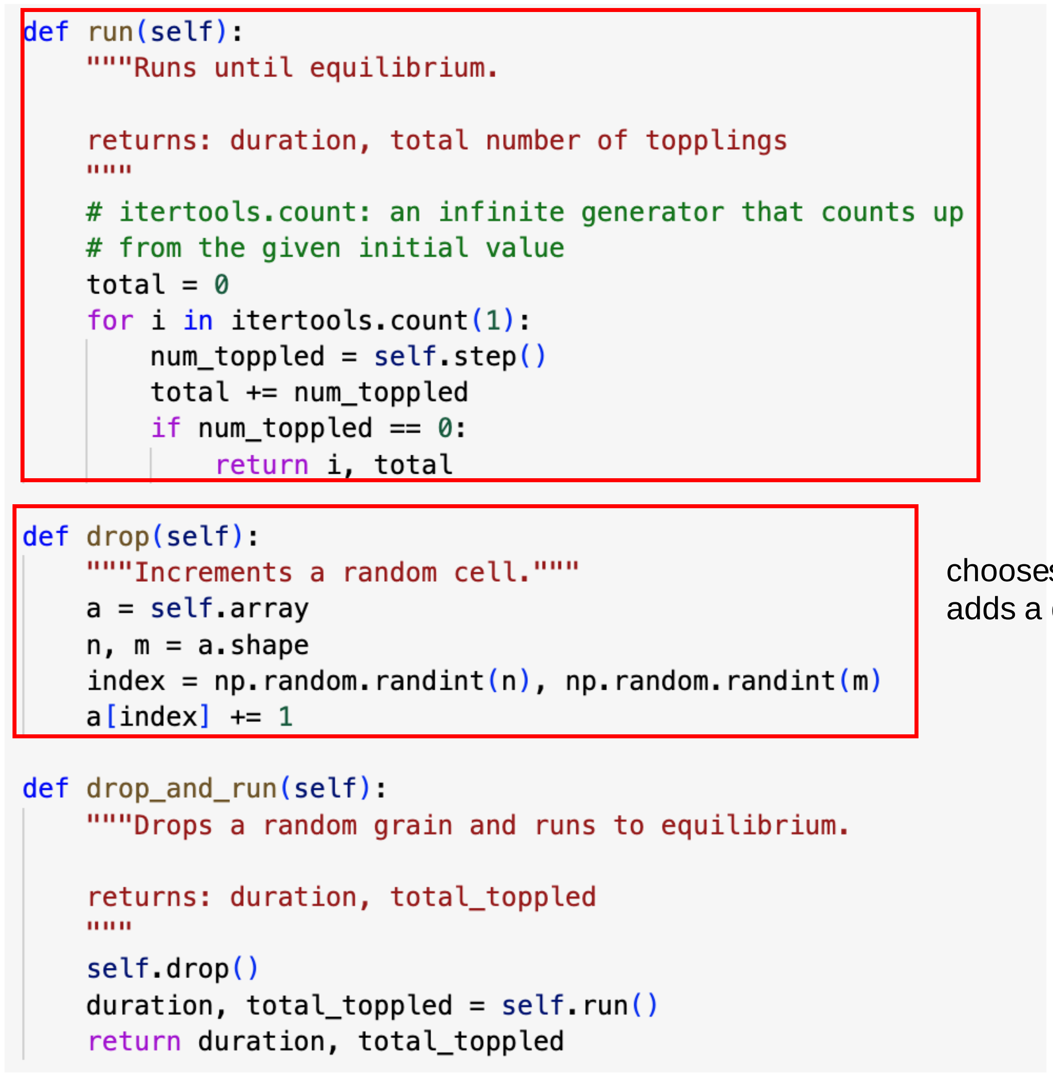


```python
class SandPile(Cell2D):
    """Diffusion Cellular Automaton."""

    kernel = np.array([[0, 1, 0],
                       [1,-4, 1],
                       [0, 1, 0]], dtype=np.int32)

    def __init__(self, n, m=None, level=9):
        """Initializes the attributes.

        n: number of rows
        m: number of columns
        level: starting value for all cells, greater than the critical value.
        """
        m = n if m is None else m
        self.array = np.ones((n, m), dtype=np.int32) * level
        self.toppled_seq = []

    def step(self, K=3):
        """Executes one time step.

        returns: number of cells that toppled
        """
        toppling = self.array > K
        num_toppled = np.sum(toppling)
        self.toppled_seq.append(num_toppled)

        c = correlate2d(toppling, self.kernel, mode='same')
        self.array += c
        return num_toppled
```

All values in the array are initialised to level, which is generally greater than the toppling threshold, K

The step method finds all cells above K and topples them:

3 Sand Piles



```python
def run(self):
    """Runs until equilibrium.

    returns: duration, total number of topplings
    """
    # itertools.count: an infinite generator that counts up
    # from the given initial value
    total = 0
    for i in itertools.count(1):
        num_toppled = self.step()
        total += num_toppled
        if num_toppled == 0:
            return i, total
```

calls step method until no more cells topple;

The return value is a tuple that contains the number of time steps to stabilise and the total number of cells that toppled.

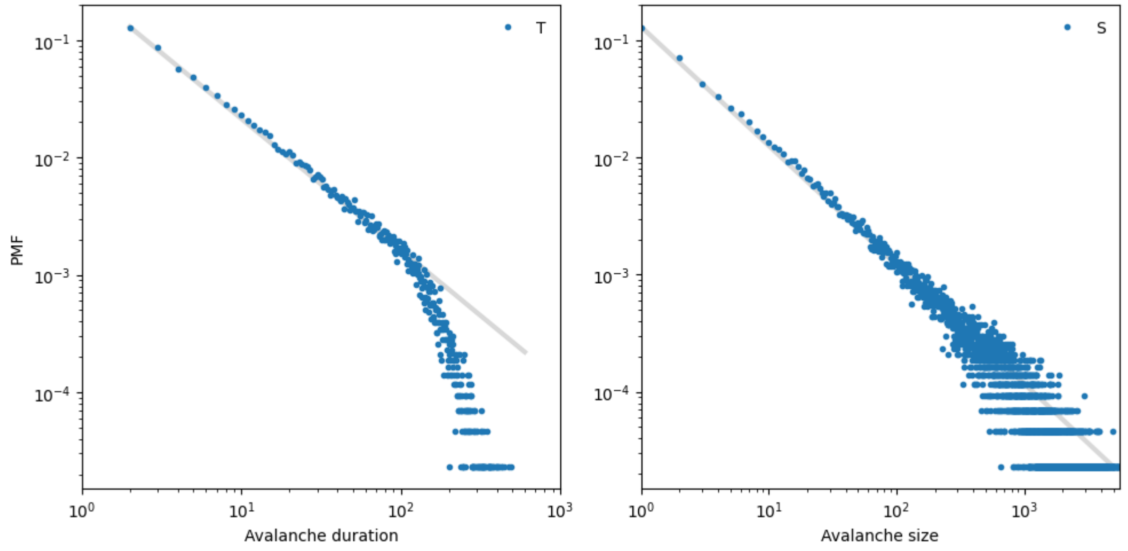
```python
def drop(self):
    """Increments a random cell."""
    a = self.array
    n, m = a.shape
    index = np.random.randint(n), np.random.randint(m)
    a[index] += 1
```

chooses a random cell and adds a grain of sand

```python
def drop_and_run(self):
    """Drops a random grain and runs to equilibrium.

    returns: duration, total_toppled
    """
    self.drop()
    duration, total_toppled = self.run()
    return duration, total_toppled
```

Sand Piles

If the sand pile model is in a critical state, we expect to find **heavy-tailed distributions** of T, the number of time steps the pile takes to stabilize, and S, the total number of cells that topple.



A $50x50$ grid with initial level of 30, run 100,000 random drops. PMF on log-log scale.

For values between 1 and 100, the distributions are nearly straight on a log-log scale, which is characteristic of a heavy tail.

3 Sand Piles

Estimate Fractal Dimension

Box counting: count the number of cells in a small box at the centre of the pile, then see how the number of cells increases as the box gets bigger.

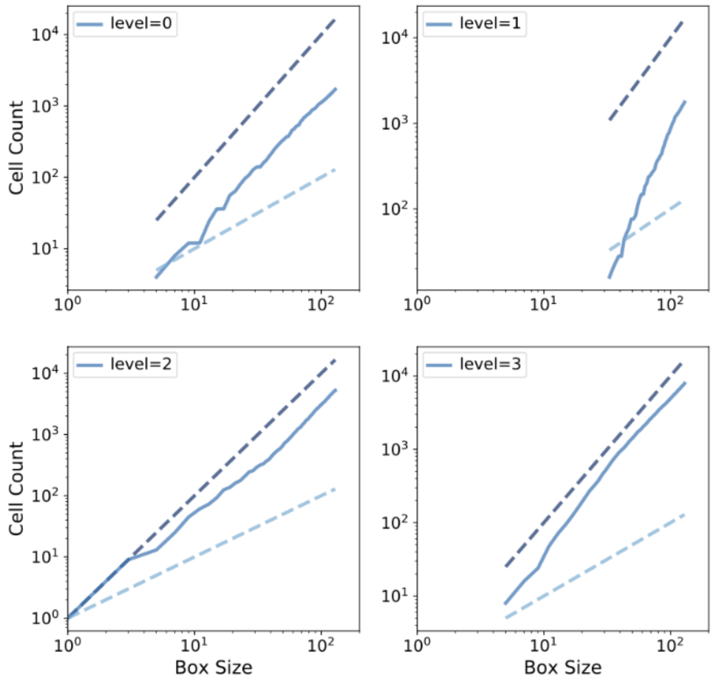
```
def count_cells(a):
    n, m = a.shape
    end = min(n, m)

    res = []
    for i in range(1, end, 2):
        top = (n-i) // 2
        left = (m-i) // 2
        box = a[top:top+i, left:left+i]
        total = np.sum(box)
        res.append((i, i**2, total))

    return np.transpose(res)
```

- The size of the box is initially 1. Each time through the loop, it increases by 2;
- Each time through the loop, box is a set of cells with width and height i, centred in the array;
- Sum the total number of "on" cells each time.
- The result is a list of tuples, where each tuple contains i, $i ** 2$, and the number of cells in the box.

Sand Piles

Estimate Fractal Dimension

If we draw $y = i^2$ and $y = i$ on log-log scale, we will have the slope of 2 and 1, respectively.

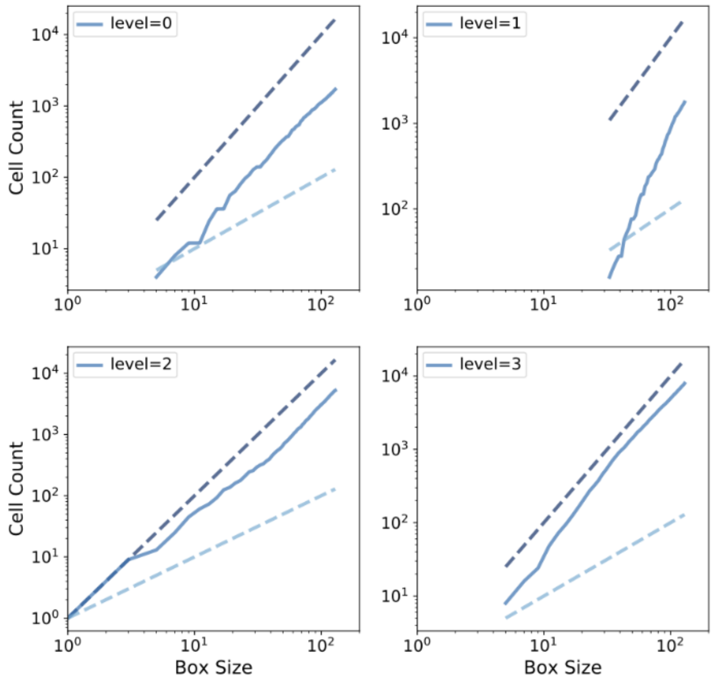


A $131x131$ grid with initial level of 22. After reach equilibriums, we select cells with level 0, 1, 2 and 3

Sand Piles

Estimate Fractal Dimension

If we plot the cells count and the box size i on the log-log scale, we can see the different slope.



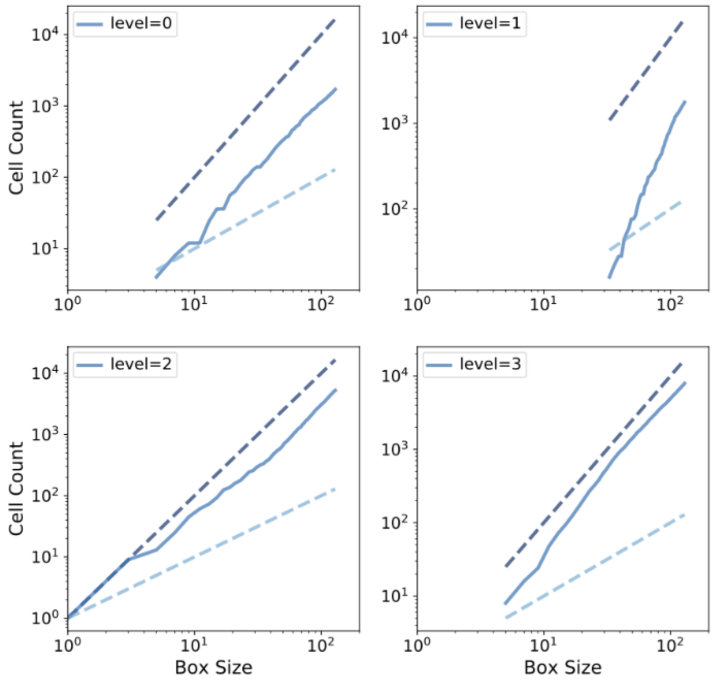
The estimated slope for level 0, 1, 2 and 3 are around 1.871, 3.502, 1.781 and 2.084, respectively.

This suggests that level 0, 1, 2 have a "fractional dimension", that is, it is a fractal.

Sand Piles

Estimate Fractal Dimension

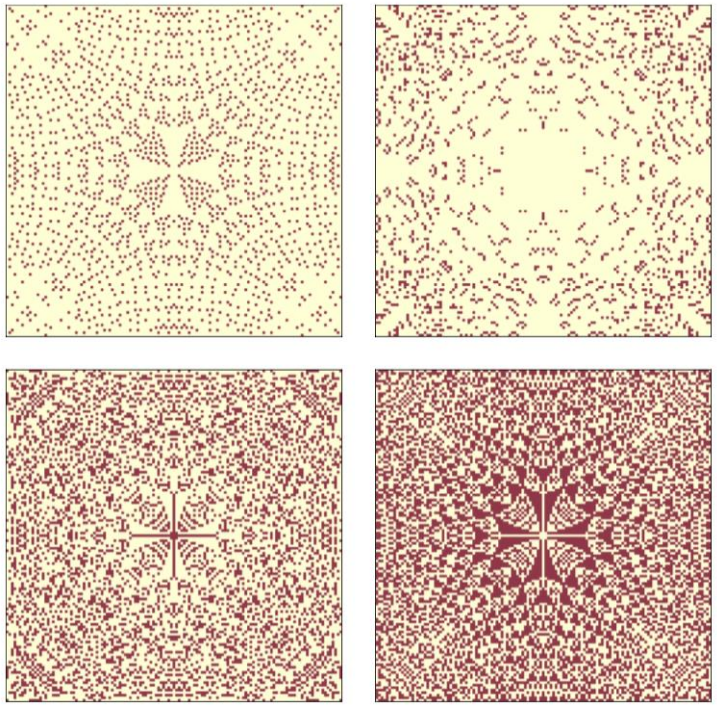
If we plot the cells count and the box size i on the log-log scale, we can see the different slope.



The estimated slope for level 0, 1, 2 and 3 are around 1.871, 3.502, 1.781 and 2.084, respectively.

Sand Piles

Estimate Fractal Dimension



Sand Piles

Pink Noise

Suppose that every time a cell topples, it makes a sound. If we record a sand pile model while its running, what would it sound like?

Sound is a signal - any quantity that varies in time.

In the sand pile model, the signals we'll consider are avalanche durations and sizes as they vary over time.

Any signal can be decomposed into a set of frequency components with different levels of power.

The power spectrum of a signal is a function that shows the power of each frequency component.

Pink Noise

Relationships between frequency and power can be written as:

$$\log P(f) = -\beta \log f$$

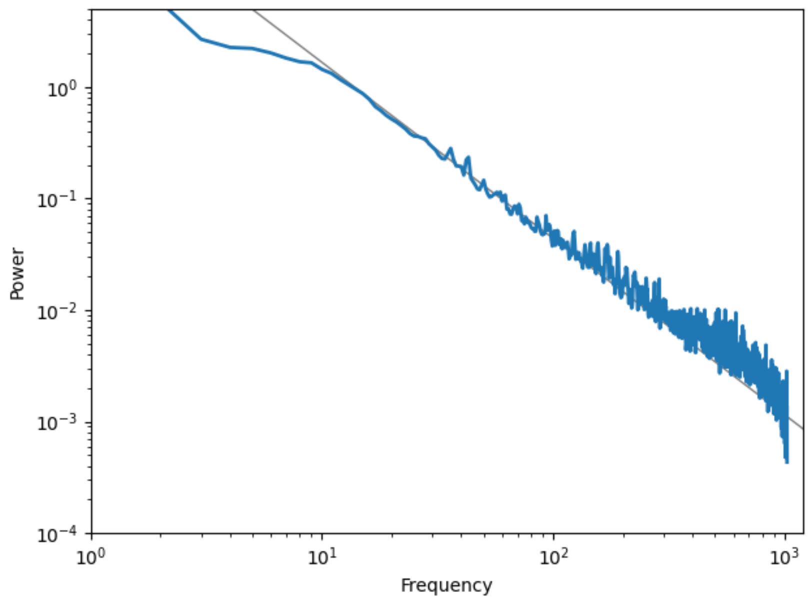
after taking the log on both side.

$P(f)$ versus f on a log-log scale, we expect a straight line with slope $-\beta$.

When $\beta = 0$, this is describing the white noise; when $\beta = 2$, this is describing the red noise; when $0 < \beta < 2$, it is the pink noise.

Sand Piles

Pink Noise



Apply Welch's method, which splits the signal - *avalanche durations and sizes as they vary over time in sand piles* - into segments and computes the power spectrum of each segment.

THE UNIVERSITY OF WESTERN AUSTRALIA

DESK No.	
FAMILY NAME:	
GIVEN NAMES:	
SIGNATURE:	
STUDENT NUMBER:	

Semester 2, 2024 EXAMINATIONS

School of Physics, Mathematics and Computing

CITS4403
Computational Modelling
Examination Duration: 2 hours

This is an examination WITH NO permitted materials

Provided by the University	**Supplied by the Student**
1 x 18 Page Answer Booklet	

Calculator
No calculators are permitted

Instructions to Students

This exam paper is worth a total of 100 marks.

Your answers MUST be put in the provided Answer Booklet.

You need to clearly label which question you are answering.

This is a **closed-book** test, no materials or resources are allowed during the test. **No calculator allowed.**

The test will cover all the contents from Week1 to Week12, including lecture and lab code.

There will be mostly **short answer questions,** and one essay question. In total 15 questions.

You may require to:

- Give the direct answer;
- Justify your answer when the question asks for your explanation;
- Explain the purpose of the given code;
- Modify correct parts of the code to fulfill the question requirement

You will NOT be required to:

- Write the code from scratch