Computer Simulations in Physics

Course for MSc physics students

Janos Török

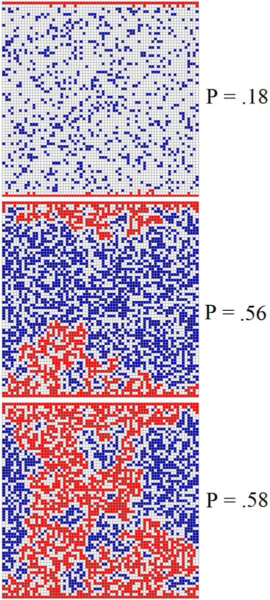
Department of Theoretical Physics

March 10, 2022

Percolation

Percolation

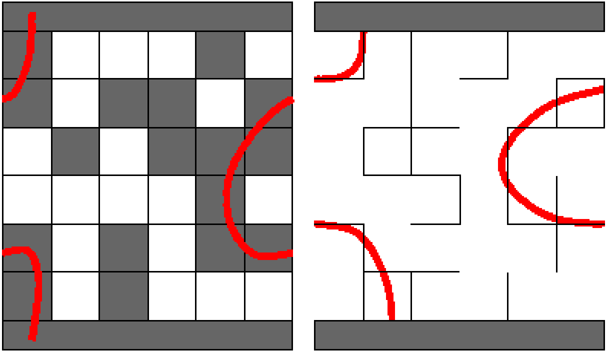
- Random environment
- Can we go from one end to the other?
- Does it percolate?
- Typical example: Lattice, sites filled with probability p
- Infinite, or giant cluster: In infinite systems, finite fraction of sites belong to it.



Percolation

Behavior of connected cluster

- Site percolation
- Bond percolation



Percolation theory

Questions (in infinite systems):

1. Is there an infinite cluster in infinite systems?
2. How many infinite clusters are there?
3. Mean cluster size (without the infinite one)?
4. Cluster size distribution

Answers:

1. Above a critical density with probability 1 below it with probability 0
2. Only 1!
3. Decreases as a power low away from the critical density
4. Power law

Percolation theory

Questions (in infinite systems):

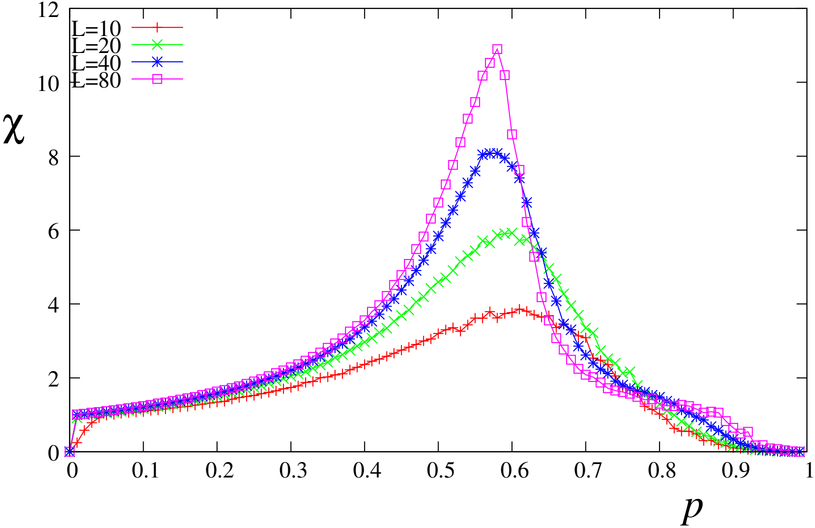
1. Is there an infinite cluster in infinite systems?
2. How many infinite clusters are there?
3. Cluster size distribution (n_s)
4. Mean cluster size (without the infinite one)? ($S = \sum_s s^2 n_s$)

Answers:

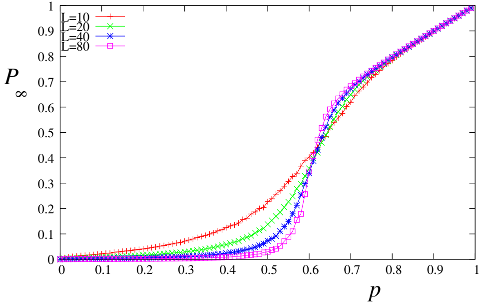
1. if $p > p_c$ then yes, otherwise no
2. Only 1!
3. $n_s \sim s^{-\tau}$
4. $S \sim |p - p_c|^{-\gamma}$

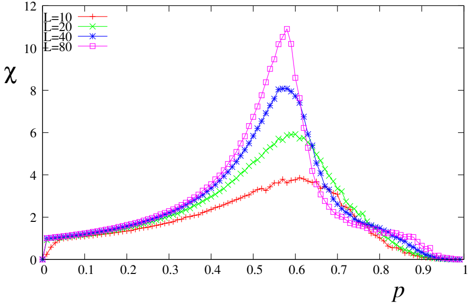
Like a second order phase transition in a geometric system!

Percolation model

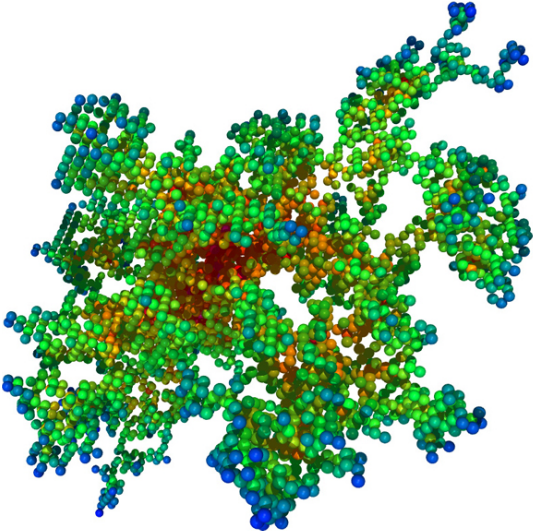


Percolation model

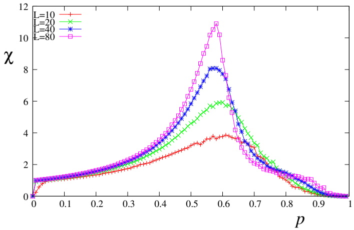


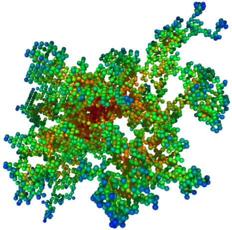


Percolation model



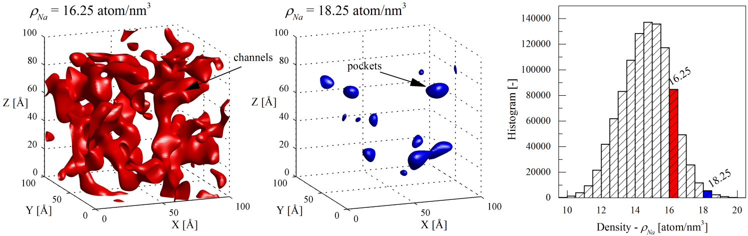
Percolating cluster

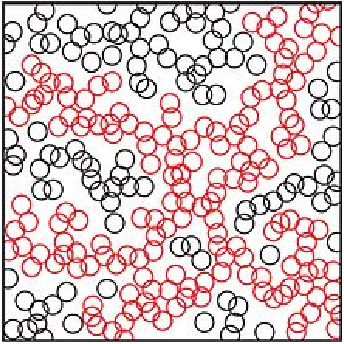


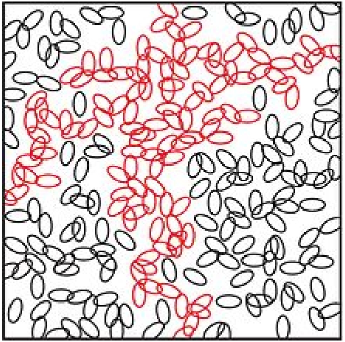


- Largest cluster
  - fractal with fractal dimension of d_f
  - $S_\infty \sim \begin{cases} \xi_f^d \log(N/\xi^d) & p < p_c \\ N^{d_f/d} & p = p_c \\ NP_\infty(p) & p > p_c \end{cases}$
- Largest not infinite cluster: size $\sim |p - p_c|^{-\nu}$

Percolation theory: Importance







Percolation theory: Importance, mainly practical

- COFFEE!!!!
- Non-equilibrium statistical physics
- Image analysis
- Percolation on networks: Phase transitions
- Percolation on networks: robustness, fragility, here also theory(!)
- Flooding

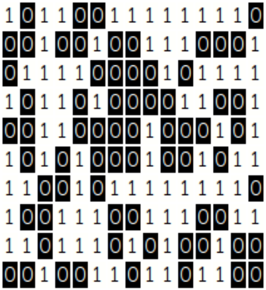
Percolation model

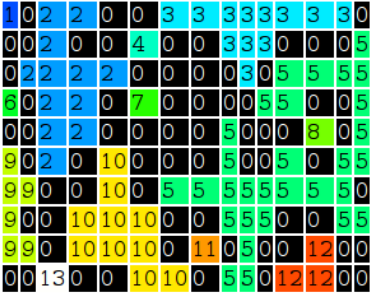
Bond [site] percolation

- Let us have a lattice (network)
- Each bond [site] is occupied with probability p
- (unoccupied with probability $1 - p$)
- A cluster is a set of sites connected by occupied bonds
  [A cluster is a set of occupied sites]

Hoshen-Kopelman Algorithm

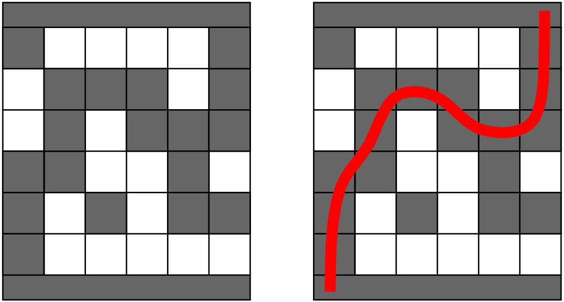
- Numerical task: find clusters
- Identify clusters
- Visit all sites
- Mark them with numbers



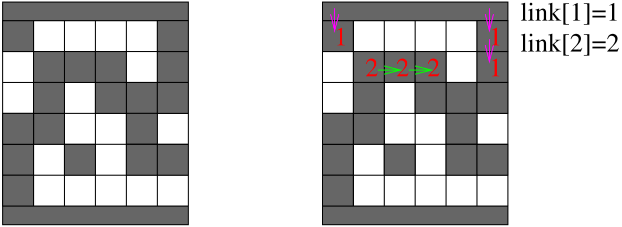


Hoshen-Kopelman Algorithm

- Site percolation
- Open boundary conditions
- Go through site in typewriter style
- Check left and above

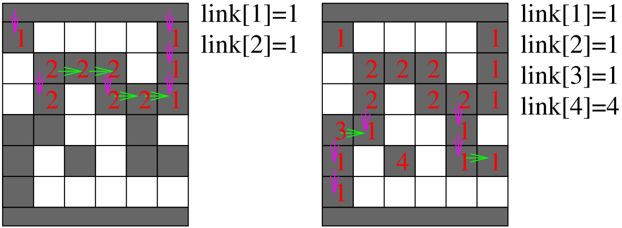


Hoshen-Kopelman Algorithm



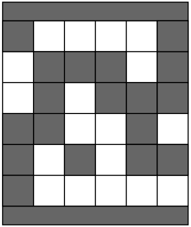
- Go through sample in typewriter style
- If site is occupied, look left and up
  - if no neighbour $\rightarrow$ new number
  - if only one is occupied $\rightarrow$ inherit number

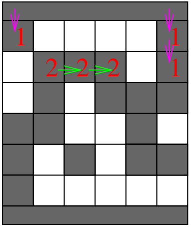
Hoshen-Kopelman Algorithm



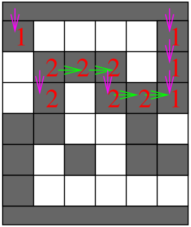
- ...
- If site is occupied, look left and up
  - ...
  - if both sites are occupied $\rightarrow$ then link the two
    - a link which points to itself is a root link
    - find the root of both sites
    - connect the larger to the smaller
    - use this number

Hoshen-Kopelman Algorithm

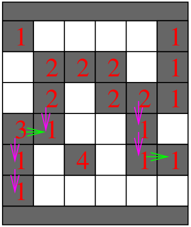




link[1]=1
link[2]=2



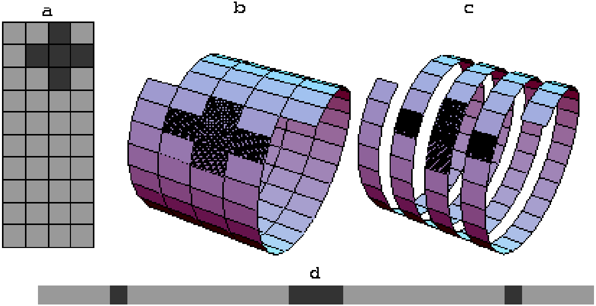
link[1]=1
link[2]=1



link[1]=1
link[2]=1
link[3]=1
link[4]=4

Hoshen-Kopelman Algorithm

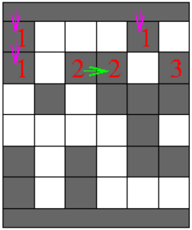
- Site percolation
- Helical boundary conditions (rolled up ont dimensional lattice)
- Go through site in typewriter style
- Check left and above (as before)



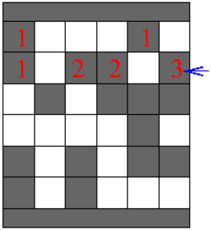
Hoshen-Kopelman Algorithm

- Site percolation
- Periodic boundary conditions
- Go through site in typewriter style
- Check left and above (as before)
- After each line if first and last site is occupied link them

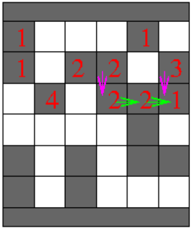
Hoshen-Kopelman Algorithm, Periodic BC



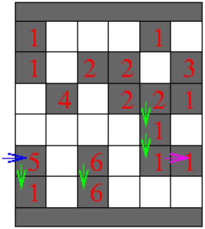
link[1]=1
link[2]=2
link[3]=3



link[1]=1
link[2]=2
link[3]=1

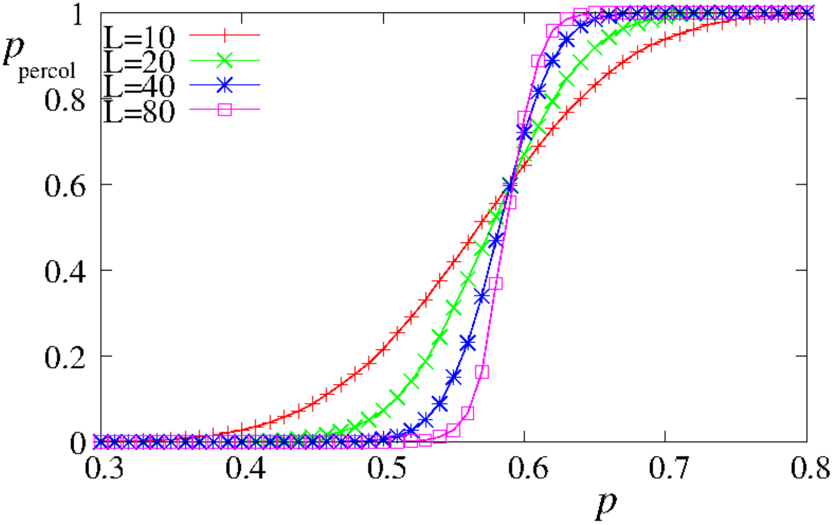


link[1]=1
link[2]=1
link[3]=1
link[4]=4



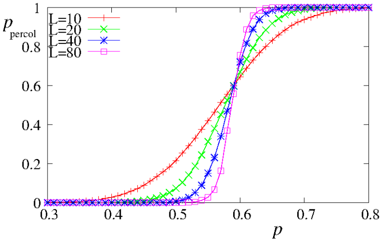
link[1]=1
link[2]=1
link[3]=1
link[4]=4
link[5]=1
link[6]=6

Result

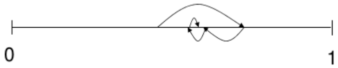


Determine p_c

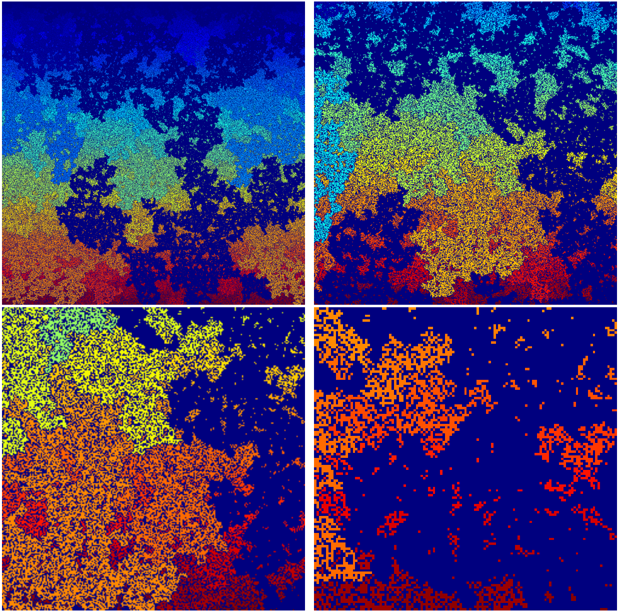
- From order parameter:



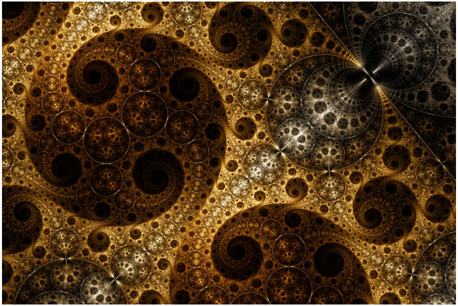
- Increase and decrease p by $p/2$ to converge to p_c
- Use the monotonity of the percolation
- Same random number sequence can be generated!



Self similarity



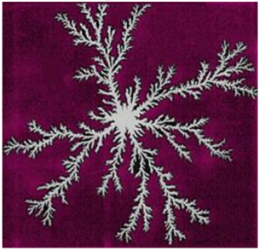
Fractals



Mandelbrot set: $f_c(z) = z^2 + c$, $z, c \in \mathbb{C}$ for which $f_c(0)$, $f_c(f_c(0))$ remains bounded in absolute value.

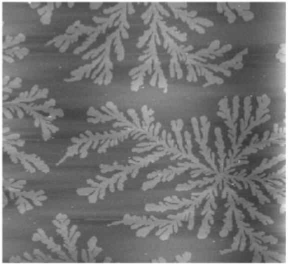
Fractal growth

Fractal growth



Elctrochem. deposition

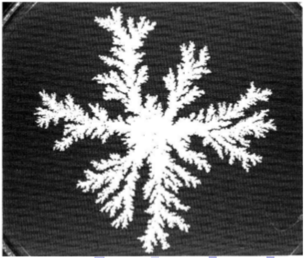
Mineralization



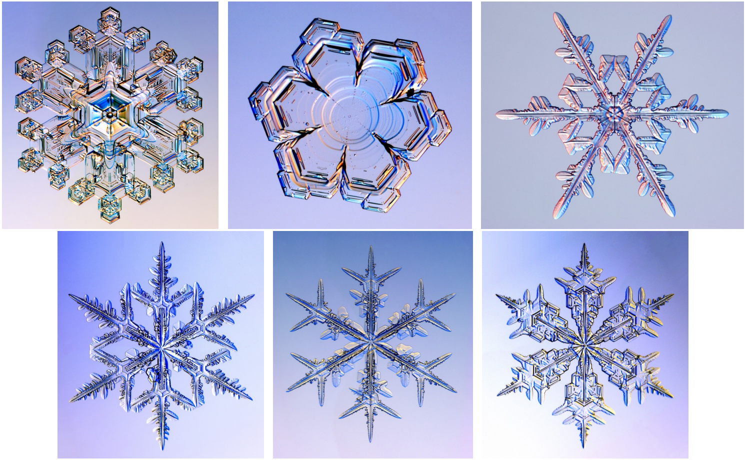
Surface crystallization

Disordered viscous fingering

Bacterial colony growth



Snowflakes



Fractal growth

Laplacian or gradient governed growth

- Scalar field (electrostatic field, density, through diffusion)

$$\Delta u = 0$$

- Velocity of the interface Γ proportional with the gradient

$$\mathrm{v}|_\Gamma = -C\nabla u|_\Gamma$$

- Boundary condition: potential is curvature (κ) dependent

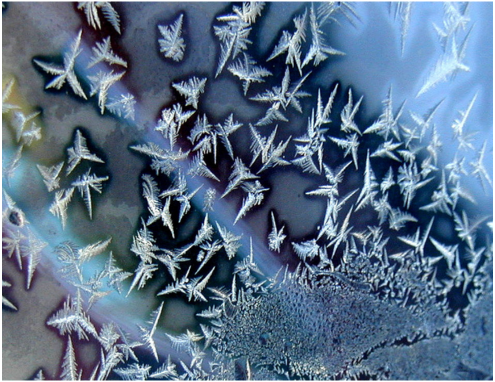
$$u|_\Gamma = f(\nabla u, \kappa)$$

- Disorder: small fluctuations

Fractal growth

Laplacian or gradient governed growth

- Scalar field (electrostatic field, density, through diffusion)
- Velocity of the interface Γ proportional with the gradient
- Boundary condition: potential is curvature (κ) dependent
- Disorder: small fluctuations



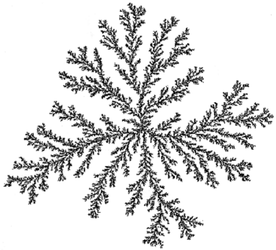
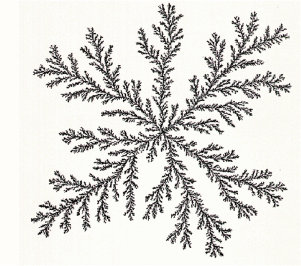
Fractal growth

Consequences:

- *Positive growth feedback*: If there is a bump, gradient increases (peak effect), growth gets faster
- *Screening*: Faster bump will screen the slower one
- *Branching*: If tip is far a new bump may grow.
- *Tip splitting*: Tip gets instable and splits

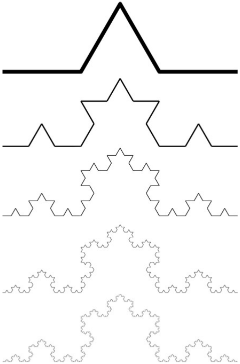
Fractal

- Self-similarity
- Repeating pattern
- Scaling patterns

Scale invariance

Fractal dimension: Example

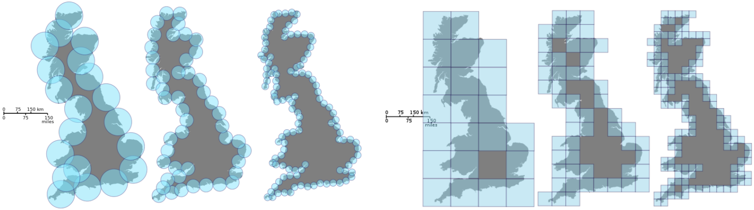


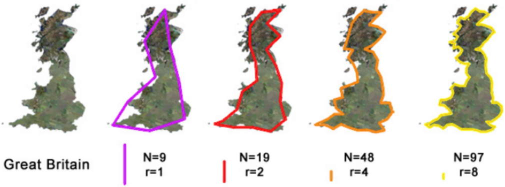
Koch curve

- Start from the first image
- Replace each segment with the scaled version of the image
- If we zoom in we see the same image
- What is the weight of the resulting object?
- It is neither one nor two dimensional

Dimension

- $d = 0$ point, $d = 1$ line, $d = 2$ plane, etc. Containing space.
- Dimension of a finite object: Cover it
- Hausdorff (fractal) dimension
- Minkowski–Bouligand dimension





Fractal dimension

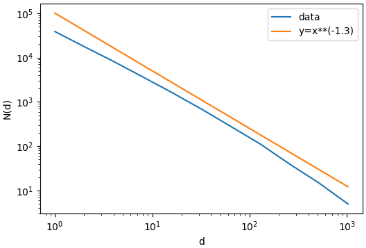
- Fractal dimension
  - Cover the object with boxes of size ε, the fractal dimension is:

$$D = \dim(S) \equiv \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log 1/\varepsilon}$$

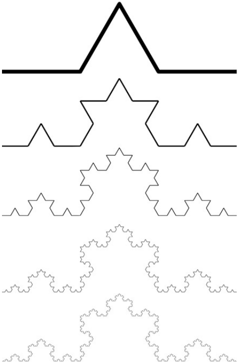
- Differences:
  - Minkowski–Bouligand: Regular lattice is used
  - Hausdorff: Spheres of given size are used.
- In practice

$$N(\varepsilon) \propto \varepsilon^D$$

Fractal dimension



Fractal dimension: Example



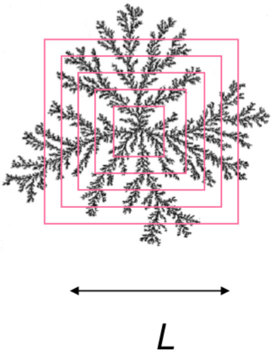
Koch curve

- Start from unit segment
- Hausdorff dimension: cover it with spheres of size $l = 3^{-i}$
- Number of spheres needed $N_l = 4^i$ (take level i!)
- Fractal dimensions:

$$D = \frac{\log N_l}{\log 1/l} = \frac{i \log(4)}{+i \log(3)} = \log_3(4) \simeq 1.262$$

Fractal dimension: Other methods

- Sandbox method: $M \propto L^D$



- Correlation functions

$$C(r) = \langle \rho(r)\rho(0) \rangle \propto r^{-\alpha}$$

$$D = d - \alpha$$

Diffusion Limited Aggregation: Algorithm

Basic:

- Start with a seed at (0,0)
- Particles start far from the aggregate and diffuse till they get adjacent to existing cluster

Advanced:

- Start with a seed at (0,0)
- Start random walker on a circle just big enough to cover the cluster
- Define a kill ring big enough or use reentry distribution
- Regions of large jumps, on a larger scale lattice

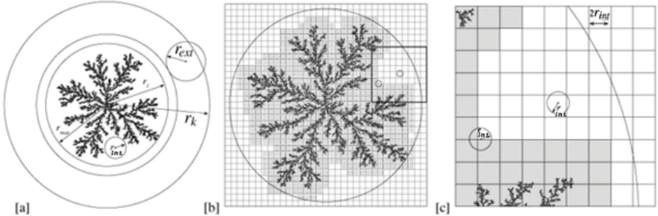
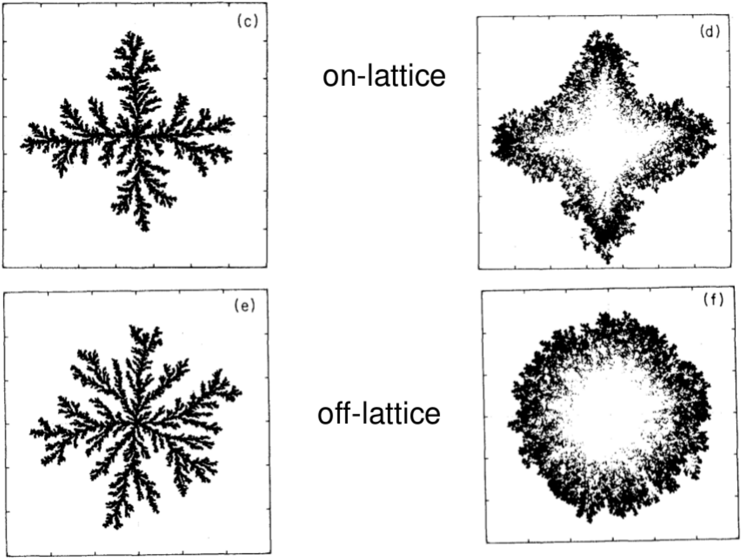


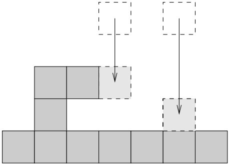
FIG. 1: (a) Schematic representation of the "optimized random trajectories". (b) A DLA aggregate and a mesh of cells $2r_{int} \times 2r_{int}$. Long steps are forbidden in the gray boxes and allowed in the white ones. Also, two long steps are illustrated. (c) A zoom of the region inside the large square in (b).

DLA: Lattice effects

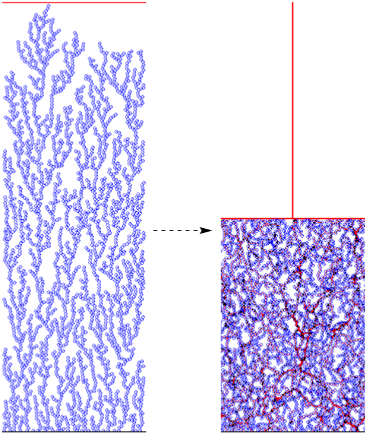
10^6 particles — 10 clusters of 10^5 particles

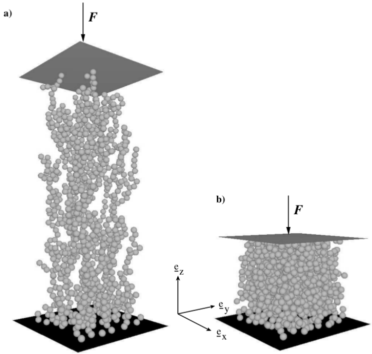


Ballistic deposition



- Lattice
- Off lattice

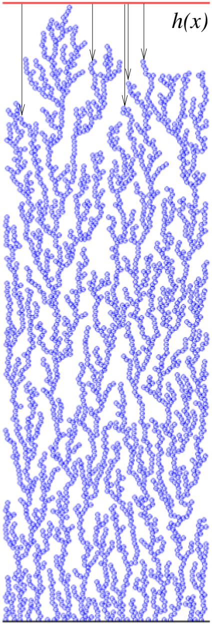




Surface growth models

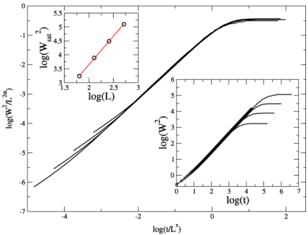
- Not the whole object but only its surface is interesting (e.g. coastline)
- Object starts from a d-dimensional substrate
- Object grows in the $d+1$th dimension.
- Object is described by $h(\mathrm{x})$ (x is a d-dimensional position vector) height function which is the maximum surface position at x.
- Width of the surface

$$w(L,t) = \sqrt{\frac{1}{L}\int_0^L [h(x,t) - \bar{h}(t)]^2 dx}$$



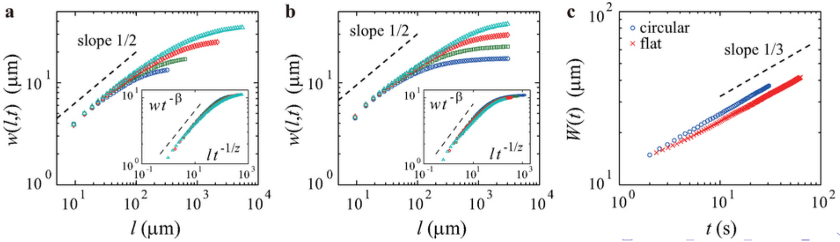
Family-Vicsek scaling

- Change of width in time



- Scaling relation:

$$w(L,t) \propto L^{\alpha} f(t/L^{z})$$



Theory: The KPZ-equation

- Surface growth $\dot{h}(\mathrm{x}, t)$
- Function of: position(?), height, gradient, Laplace of height, noise

$$\dot{h}(\mathrm{x}, t) = f[\mathrm{x}, h(\mathrm{x}, t), \nabla h(\mathrm{x}, t), \Delta h(\mathrm{x}, t), \ldots, \eta(\mathrm{x}, t)]$$

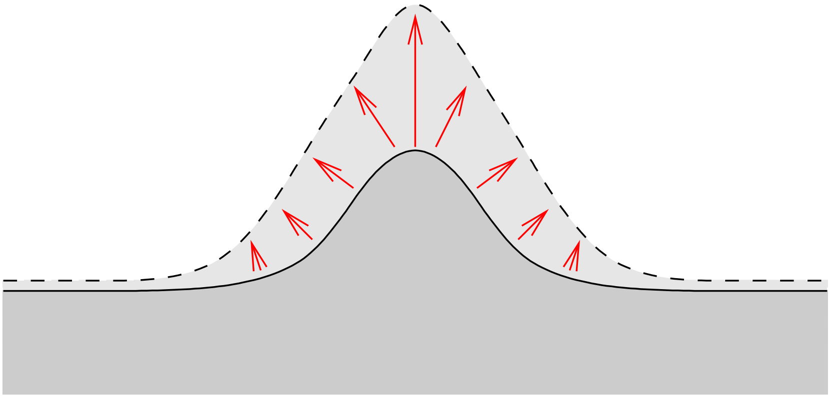
- Normally:

$$\dot{h}(\mathrm{x}, t) = f[h(\mathrm{x}, t), \nabla h(\mathrm{x}, t), \Delta h(\mathrm{x}, t), \eta(\mathrm{x}, t)]$$

- Gaussian noise:

$$\langle \eta(\mathrm{x}, t)\eta(\mathrm{x}', t')\rangle = A\delta(t - t')\delta(x - x')$$
$$P(\eta) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{\eta^2}{2\sigma}\right)$$

The Kadar-Parisi-Zhang equation



- Growth is lateral, up to second order

$$\dot{h}(x,t) = f\left[(\nabla h(x,t))^2, \Delta h(x,t), \eta(x,t)\right]$$

The Kadar-Parisi-Zhang equation

$$\dot{h}(x, t) = \nu \Delta h(x, t) + \lambda (\nabla h(x, t))^2 + \eta(x, t)$$

- Nonlinear
- Stochastic
- Partial differential equation

Numerical solution of the KPZ-equation

- ξ is a random number with zero mean (can be Gaussian, or uniform)
- Due to noise Euler scheme is enough:

$$
\begin{aligned}
h_i(t+\Delta t) = & h_i(t) + \nu \frac{\Delta t}{(\Delta x)^2}\left[h_{i+1}(t) - 2h_i(t) + h_{i-1}(t)\right] + \\
& + \frac{\lambda}{4}\left[h_{i+1}(t) - h_{i-1}(t)\right]^2 + \xi_i
\end{aligned}
$$

- Critical exponents and and universality classes $\alpha = 1/2$, $z = 3/2$

Practice: Hoshen-Kopelman Algorithm

- Fill a square lattice with random 0 and 1
- Create a large enough array where `link[i]=i`
- Go through the lattice in a typewriter style
- If the site is not empty check the sites to the top and left (if they exist)
  - if both neighbors are empty $\rightarrow$ assign it a new label (you can keep the labels in the original array)
  - if only one neighbor is empty $\rightarrow$ assign it the root label of the neighbor
  - if both neighbors are occupied $\rightarrow$ search for the root labels of the sites connect the larger to the smaller and assign this value to this site
- (Bonus): Measure the distribution of the size of the clusters, or the size of the largest as function of p, etc.