

## Lecture 10:

# Simulation and Monte Carlo Method



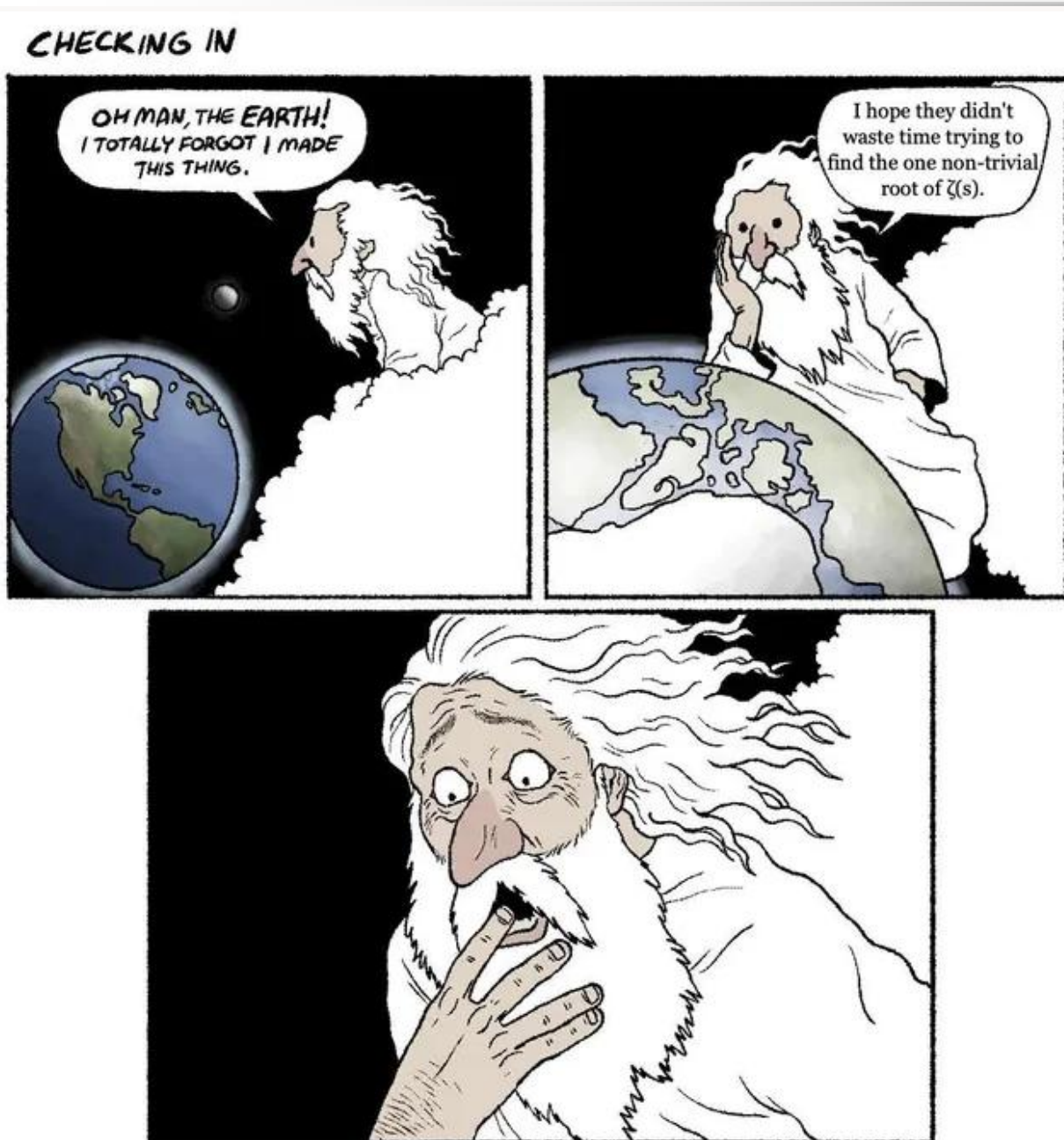
Markus Hohle

University California, Berkeley

**Numerical Methods for  
Computational Science**

## Course Map

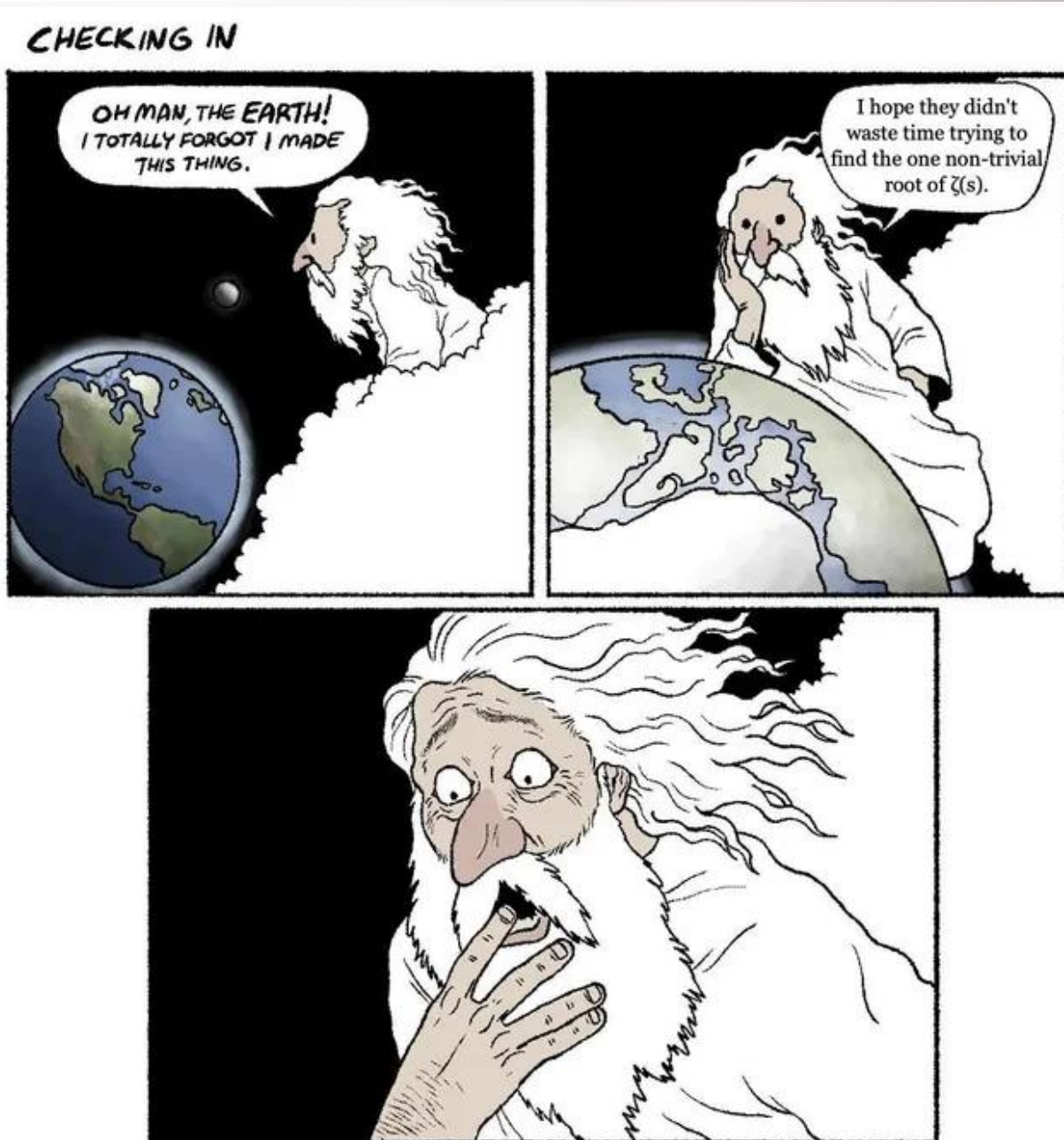
Week 1:	Introduction to Scientific Computing and Python Libraries
Week 2:	Linear Algebra Fundamentals
Week 3:	Vector Calculus
Week 4:	Numerical Differentiation and Integration
Week 5:	Solving Nonlinear Equations
Week 6:	Probability Theory Basics
Week 7:	Random Variables and Distributions
Week 8:	Statistics for Data Science
Week 9:	Eigenvalues and Eigenvectors
<b>Week 10:</b>	<b>Simulation and Monte Carlo Method</b>
Week 11:	Data Fitting and Regression
Week 12:	Optimization Techniques
Week 13:	Machine Learning Fundamentals



## Outline

- The Problem
- Finding PI
- Gillespie Algorithm
- Metropolis Algorithm





## Outline

- **The Problem**
- Finding PI
- Gillespie Algorithm
- Metropolis Algorithm



- many dynamic problems don't have an analytical solution
- some quantities have singularities → standard simulations are not possible

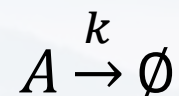




- many dynamic problems don't have an analytical solution
- some quantities have singularities  $\rightarrow$  standard simulations are not possible

degradation of a chemical compound  $A$  with rate  $k$

classic and deterministic approach: many particles of  $A \rightarrow$  concentration  $[A]$

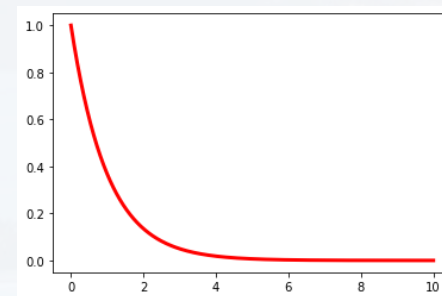


What is  $A(t)$ ?

constant relative change per time step

$$\frac{\Delta A(t)}{A(t)} \frac{1}{\Delta t} = -k \xrightarrow[\text{see lecture 2}]{\text{for small } \Delta t} \frac{dA(t)}{A(t)} = -k dt$$

$$\int_{A(t=0)}^{A(t)} \frac{1}{A(t)} dA(t) = -k \int_0^{\tau} dt$$



$$A(t) = A(t=0) e^{-kt}$$

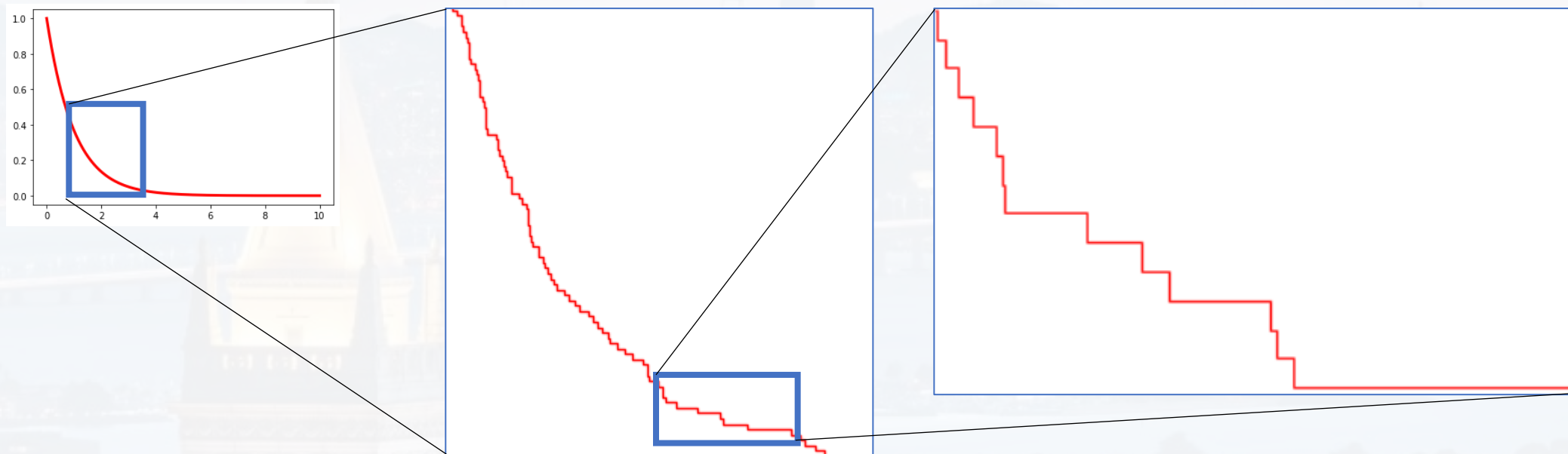




- many dynamic problems don't have an analytical solution
- some quantities have singularities  $\rightarrow$  standard simulations are not possible

**degradation of a chemical compound A with rate  $k$**

**single particle** dynamics: concentration  $[A]$  doesn't make sense



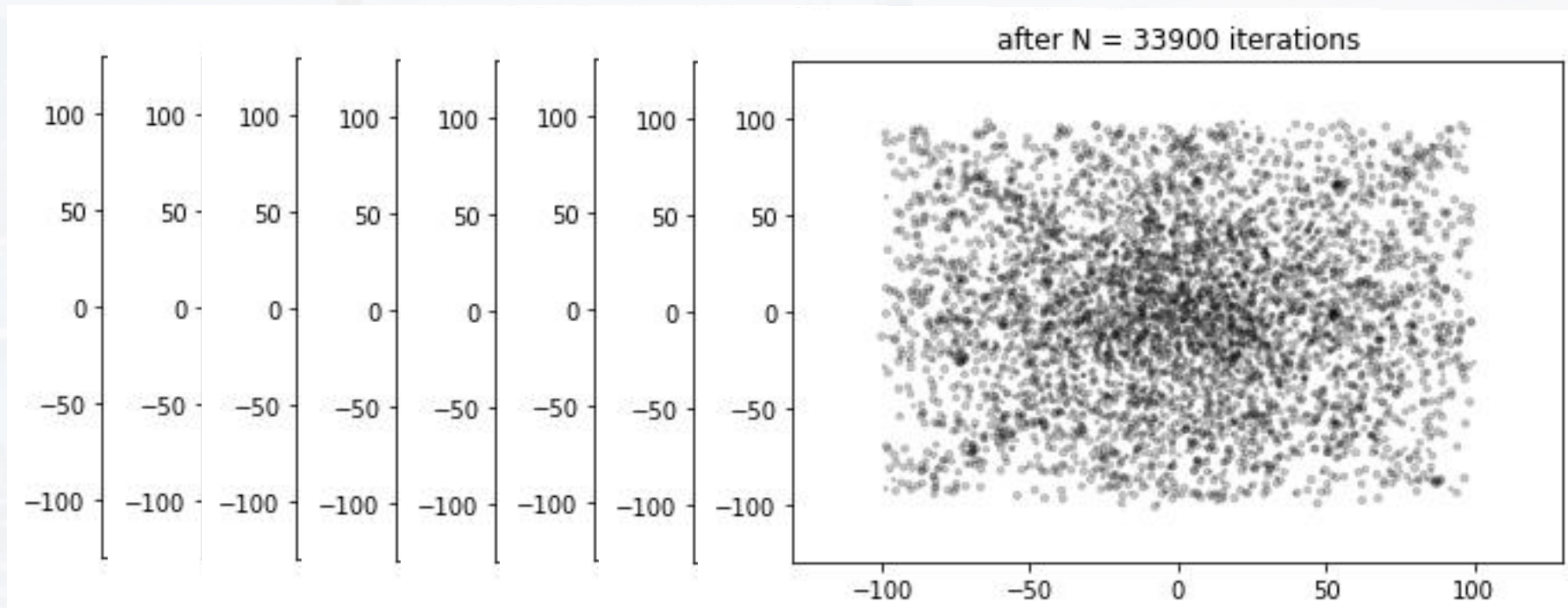
How can we model that?  
How do we take stochasticity into account?



- many dynamic problems don't have an analytical solution
- **some quantities have singularities → standard simulations are not possible**

**moving particles in a potential U**

**classic** approach: solving Newton's equations of motion

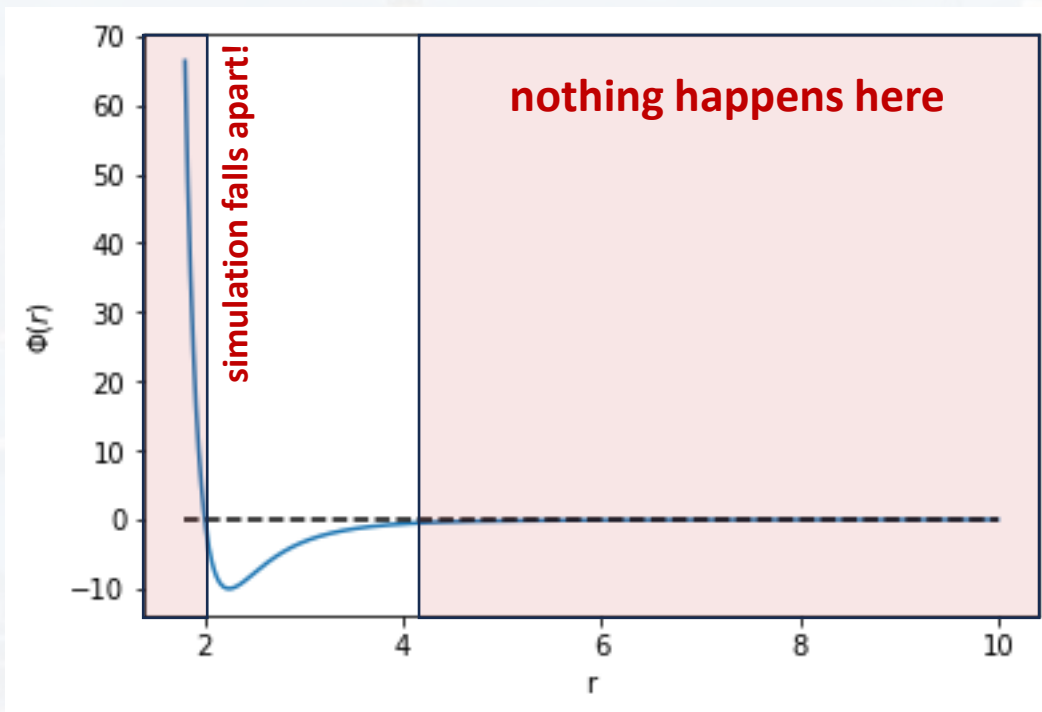






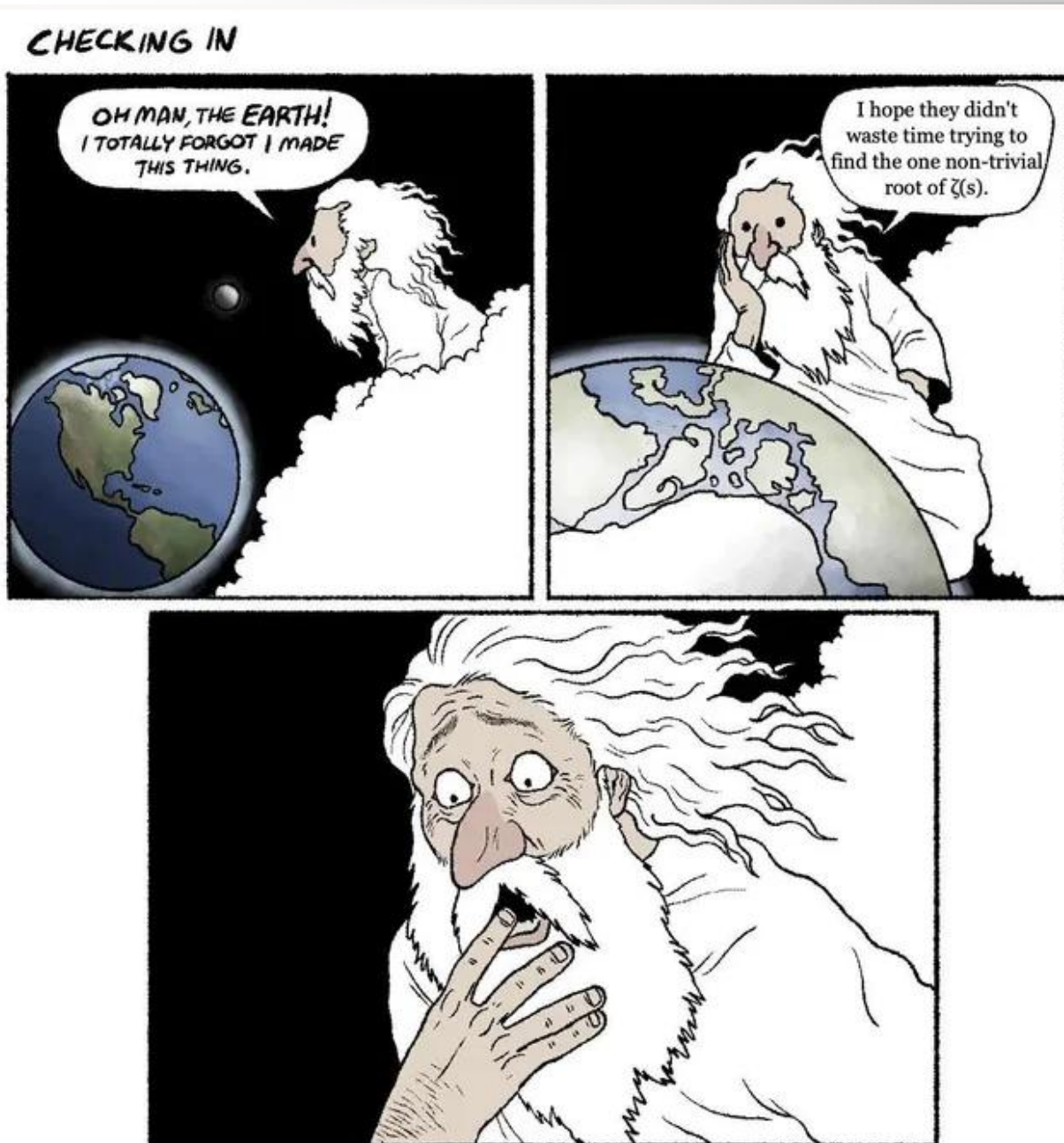
- many dynamic problems don't have an analytical solution
- **some quantities have singularities → standard simulations are not possible**

moving particles in a potential  $U$



Lennard – Jones - Potential

How can we prevent the simulation from creating non physical results?



## Outline

- The Problem
- **Finding  $P_1$**
- Gillespie Algorithm
- Metropolis Algorithm



idea: generating a set of values randomly  
i.e. repeated random sampling → **Monte Carlo** method

pros: for many sample repetitions → the actual probability density function emerges  
pretty simple set up & easy to implement  
easy to parallelize

cons: not directed like e.g. gradient descent (see later modules)

**example:** finding  $\pi$  via Monte Carlo



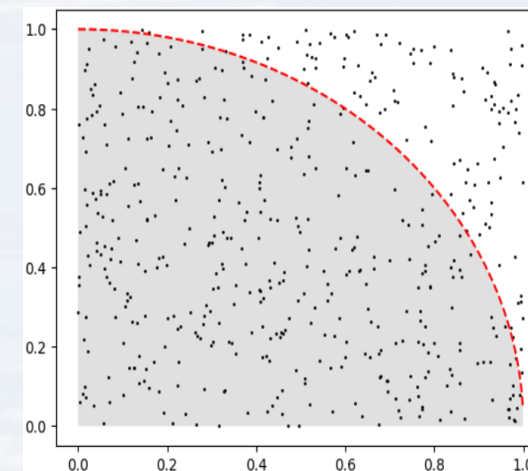
$$A_{circ} = \pi r^2 = \pi$$

$$A_{square} = 1$$

$$A_{section} = \frac{\pi}{4}$$

$$\pi = 4 \frac{A_{section}}{A_{square}}$$

picking  $N_{tot}$  random values  $[0,1] \times [0,1]$



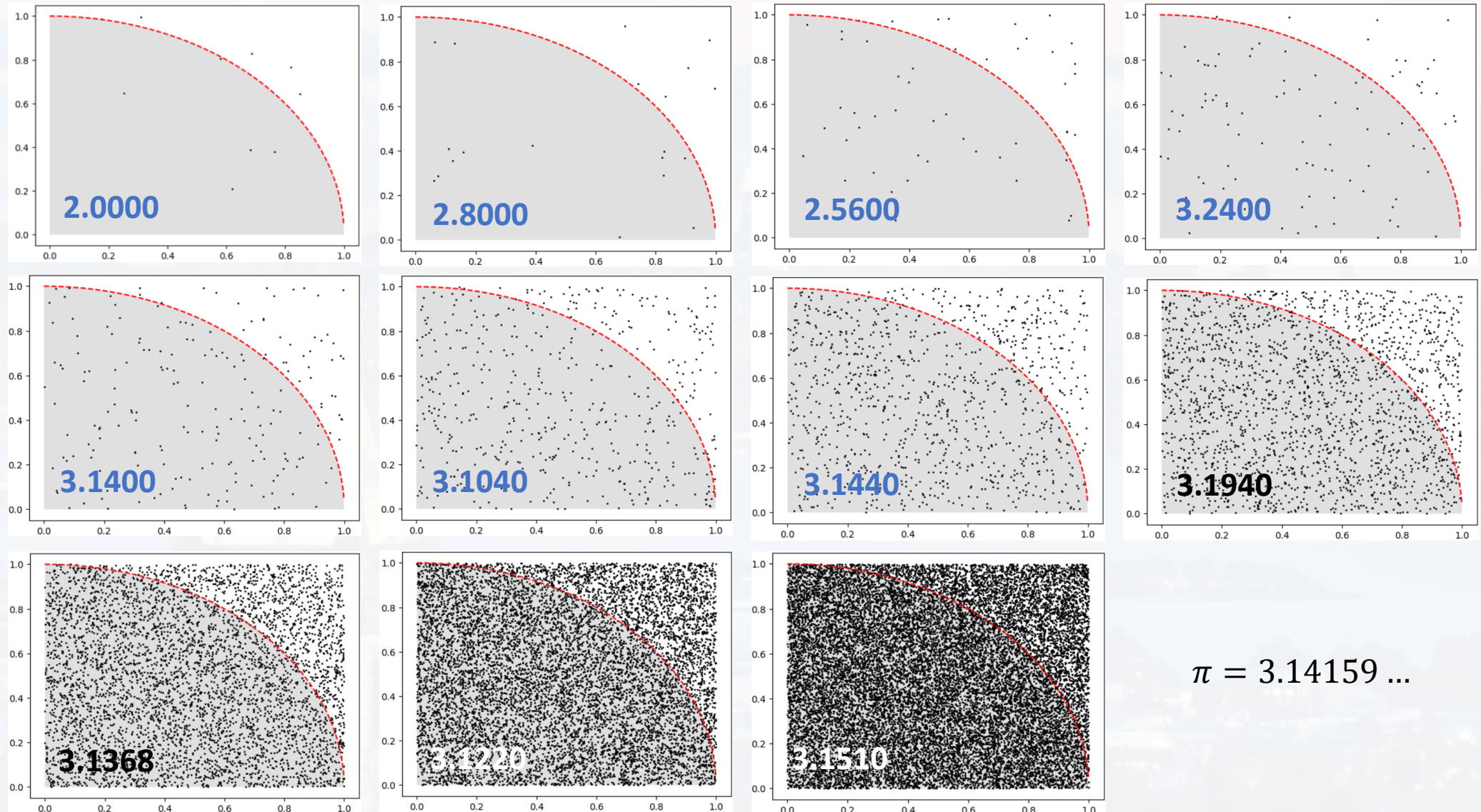
$$\pi \approx 4 \frac{N_{section}}{N_{tot}}$$





idea: generating a set of values randomly  
i.e. repeated random sampling

→ Monte Carlo method

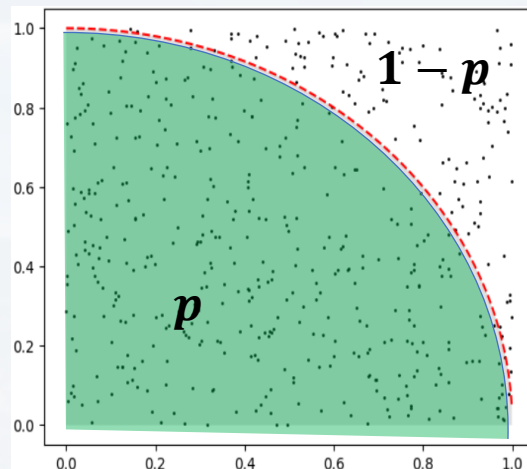


$$\pi = 3.14159 \dots$$



**example:** finding  $\pi$  via Monte Carlo

How does the accuracy of  $\pi$  depend on  $N_{tot}$ ?



$$\pi \approx 4 \frac{N_{section}}{N_{tot}}$$

We know that a point within the section is drawn with the probability  $p$

That is a **binomial problem!**

In practice,  $k$  points fall into the section with probability  $p$  and  $N_{tot} - k$  don't, with a probability of  $1 - p$

Thus, we can tell the mean and the variance of **one simulation for one specific  $N_{tot}$**

$$\sigma(k)^2 = N_{tot} p(1 - p)$$

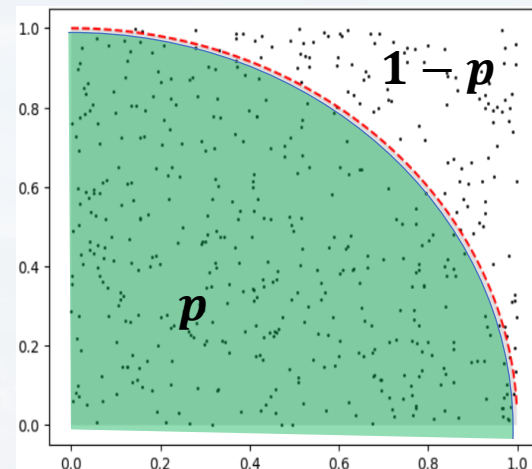
$$\mu(k) = pN_{tot}$$

(see module 7)



**example:** finding  $\pi$  via Monte Carlo

How does the accuracy of  $\pi$  depend on  $N_{tot}$ ?



$$\pi \approx 4 \frac{N_{section}}{N_{tot}}$$

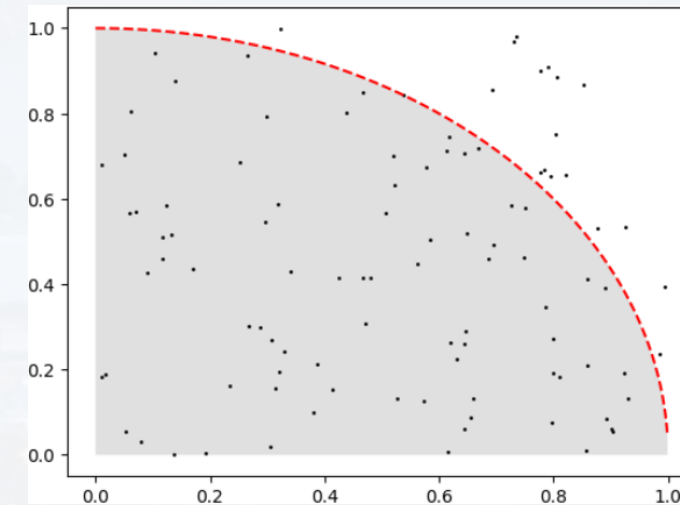
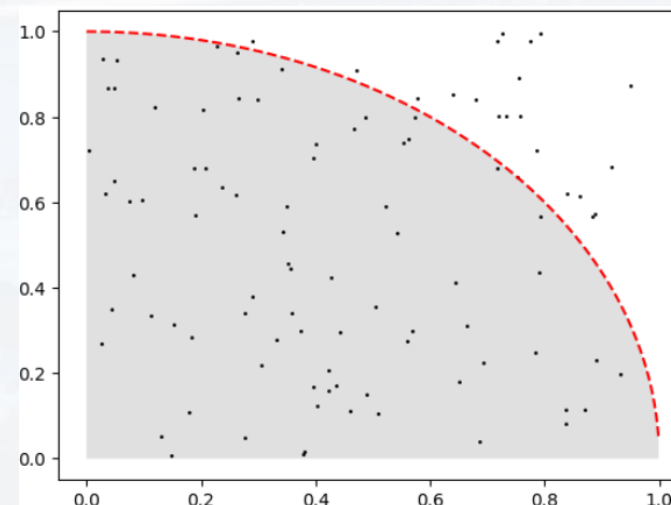
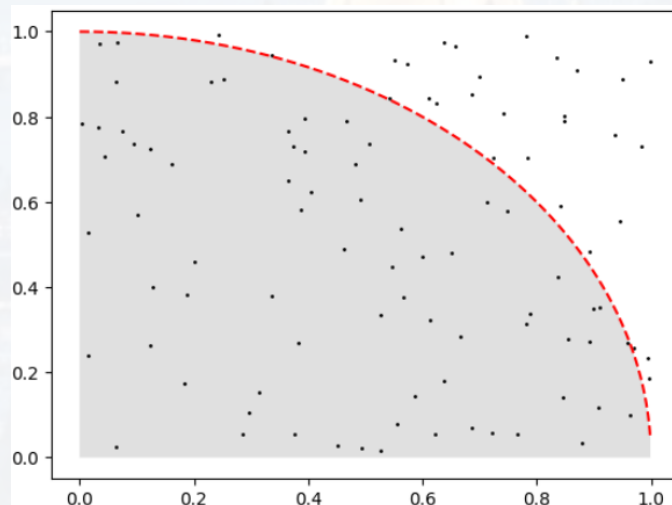
$$\sigma(k)^2 = N_{tot} p(1-p) \quad \mu(k) = pN_{tot} \quad (\text{see module 7})$$

$$\text{error of } \pi: 4\sigma\left(\frac{N_{section}}{N_{tot}}\right) = 4\sigma\left(\frac{k}{N_{tot}}\right)$$

standard deviation  $\sigma$  of the ratio  $\frac{N_{section}}{N_{tot}}$

Say we run the simulation for a **specific**  $N_{tot}$  many times  $\rightarrow Var(k)$

$N_{tot} = 100$

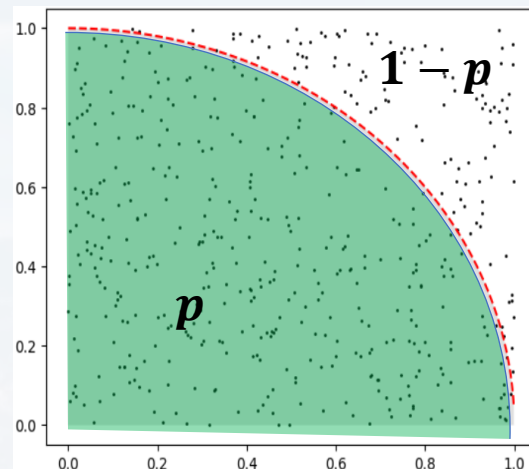






**example:** finding  $\pi$  via Monte Carlo

How does the accuracy of  $\pi$  depend on  $N_{tot}$ ?



$$\pi \approx 4 \frac{N_{section}}{N_{tot}}$$

$$\sigma(k)^2 = N_{tot} p(1 - p)$$

$$\mu(k) = pN_{tot} \quad (\text{see module 7})$$

$$\text{error of } \pi: 4\sigma\left(\frac{N_{section}}{N_{tot}}\right) = 4\sigma\left(\frac{k}{N_{tot}}\right)$$

Say we run the simulation for a **specific**  $N_{tot}$  many times  $\rightarrow Var(k)$

$$\text{error of } \pi: 4\sigma\left(\frac{N_{section}}{N_{tot}}\right) = 4\sigma\left(\frac{k}{N_{tot}}\right) = 4\sqrt{Var\left(\frac{k}{N_{tot}}\right)} = 4\sqrt{\frac{1}{N_{tot}^2} Var(k)} = 4\sqrt{\frac{1}{N_{tot}^2} N_{tot} p(1 - p)}$$

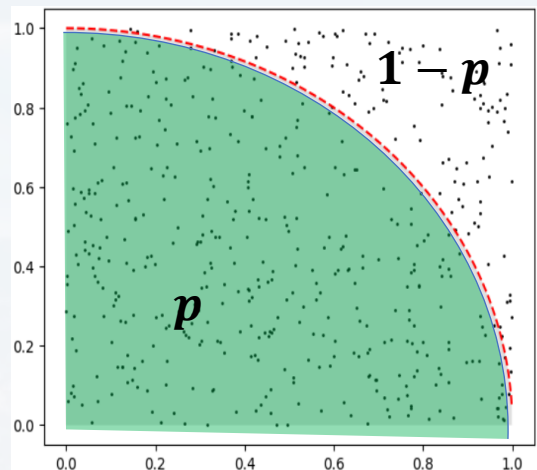
We know:  $Var(x) = \langle x^2 \rangle - \langle x \rangle^2$  (see module 7)

$$a = \text{const} \quad x \rightarrow ax \quad Var(ax) = \langle (ax)^2 \rangle - \langle ax \rangle^2 = a^2(\langle x^2 \rangle - \langle x \rangle^2) = a^2 Var(x)$$



**example:** finding  $\pi$  via Monte Carlo

How does the accuracy of  $\pi$  depend on  $N_{tot}$ ?



$$\pi \approx 4 \frac{N_{section}}{N_{tot}}$$

$$\text{error of } \pi: 4\sigma\left(\frac{N_{section}}{N_{tot}}\right) = 4\sqrt{\frac{1}{N_{tot}^2} N_{tot} p(1-p)} = 4\sqrt{\frac{1}{N_{tot}} p(1-p)}$$

of course we **don't know**  $p = \frac{\pi}{4} = \frac{A_{section}}{A_{square}}$

because we wanted find  $\pi$  in the first place

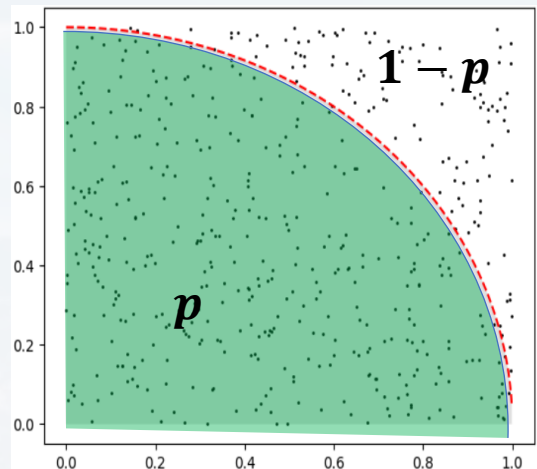
but we can **estimate** it during the simulation

$$4\sqrt{\frac{1}{N_{tot}} p(1-p)} \approx 4 \cdot 0.4 \sqrt{\frac{1}{N_{tot}}}$$



**example:** finding  $\pi$  via Monte Carlo

How does the accuracy of  $\pi$  depend on  $N_{tot}$ ?



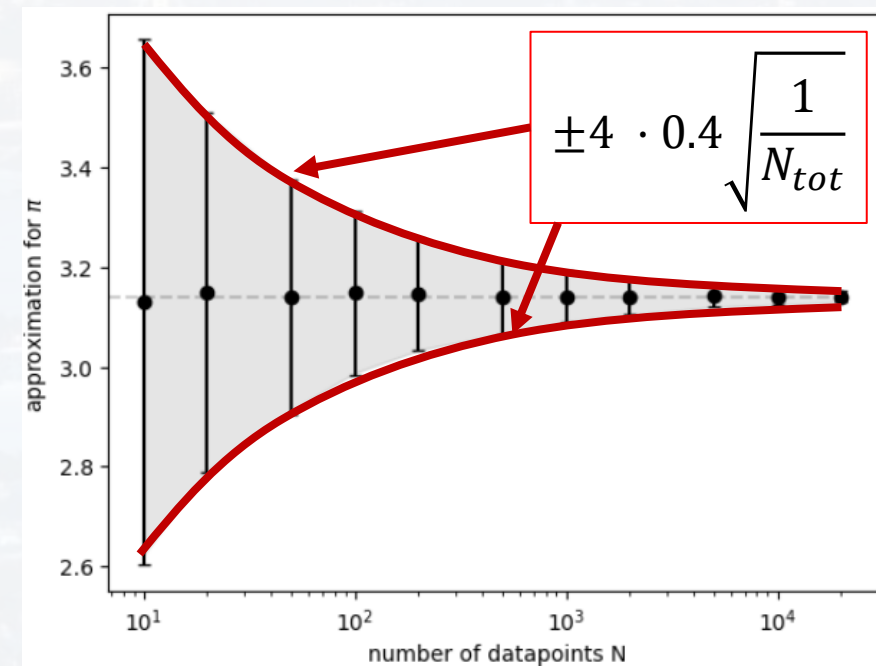
$$\pi \approx 4 \frac{N_{section}}{N_{tot}}$$

$$\text{error of } \pi: 4\sigma \left( \frac{N_{section}}{N_{tot}} \right) = 4 \sqrt{\frac{1}{N_{tot}}} p(1-p) \approx 4 \cdot 0.4 \sqrt{\frac{1}{N_{tot}}}$$

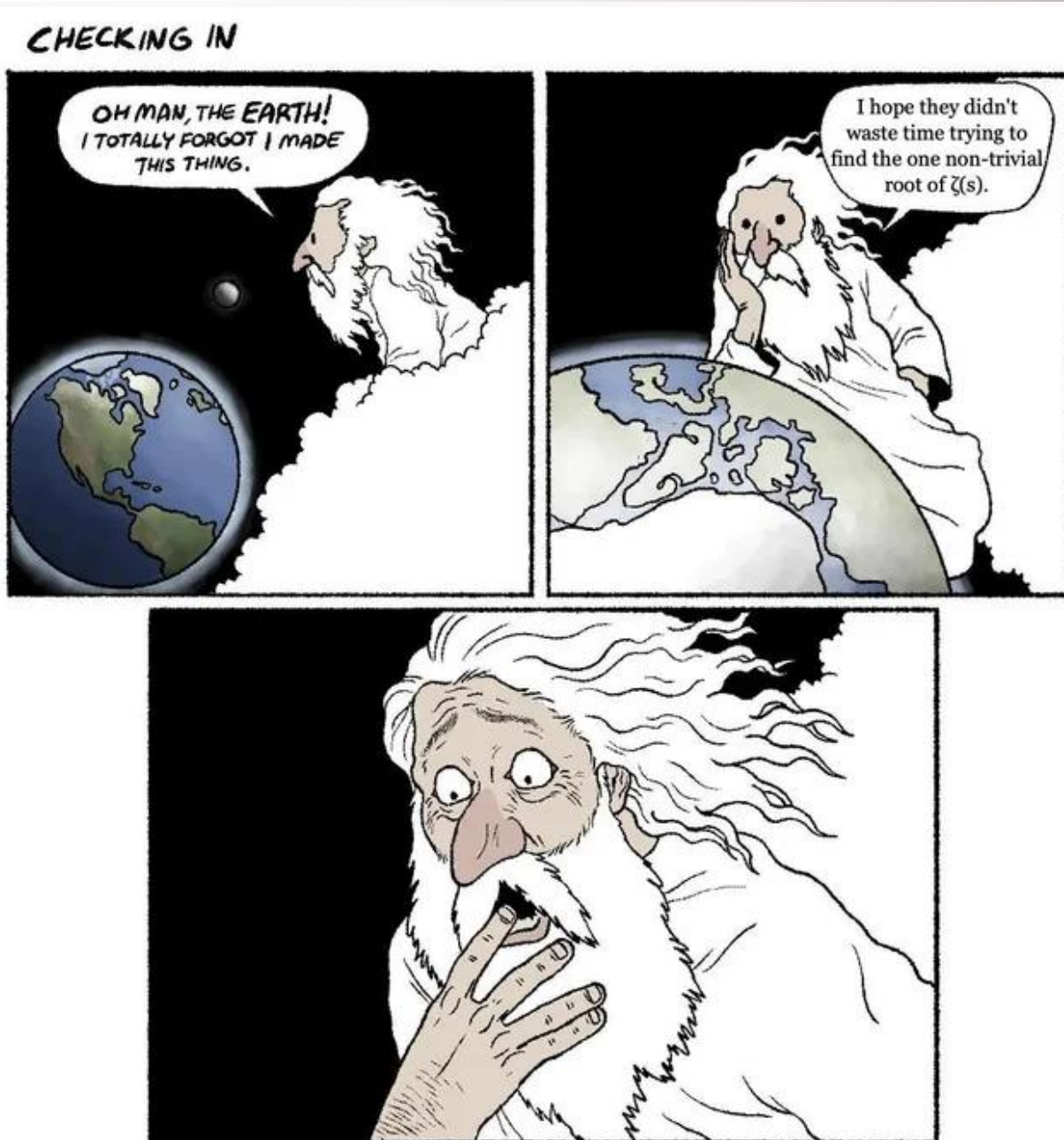
running 100 simulations **for each**  $N_{tot}$

→ calculating standard deviation of  $\pi$

→ comparing to  $4 \cdot 0.4 \sqrt{\frac{1}{N_{tot}}}$







## Outline

- The Problem
- Finding PI
- **Gillespie Algorithm**
- Metropolis Algorithm

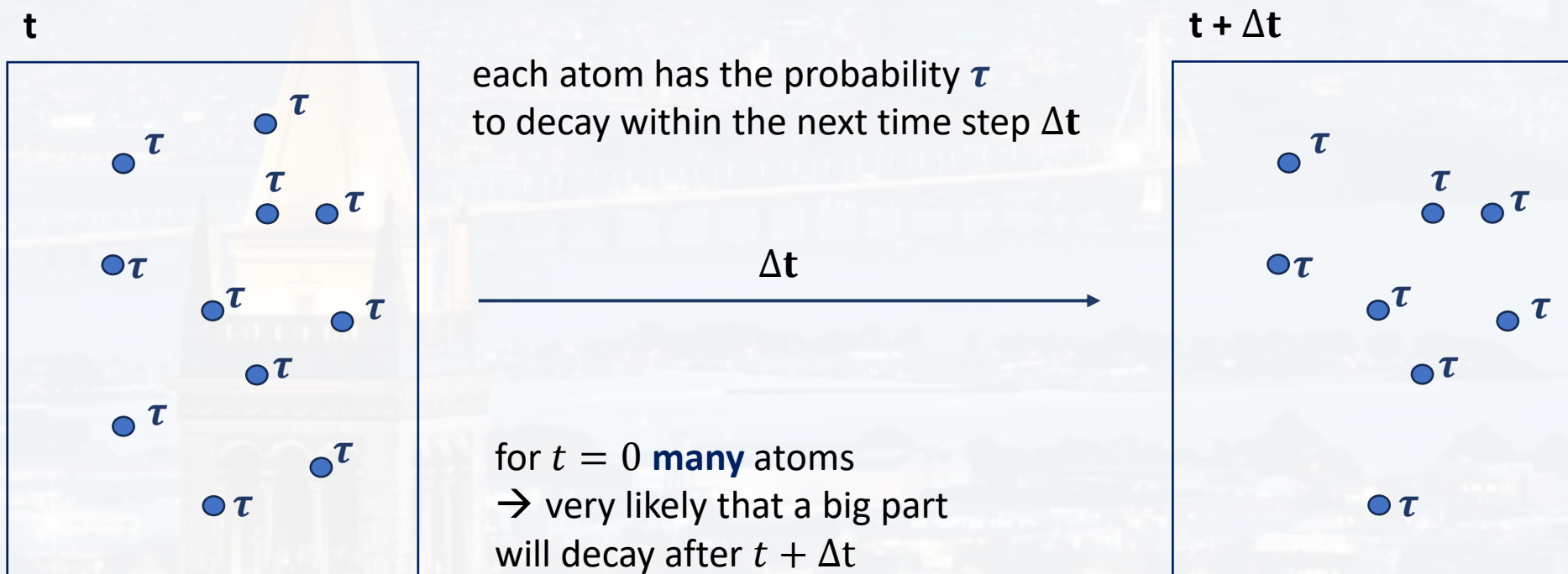


We can combine the MC method with different ways **how** to run a simulation

**example:**

- radioactive decay of atoms
- reaction  $A \xrightarrow{k} \emptyset$

**challenge:** on single particle level: decay/reaction (= changing **its state**) is **random**



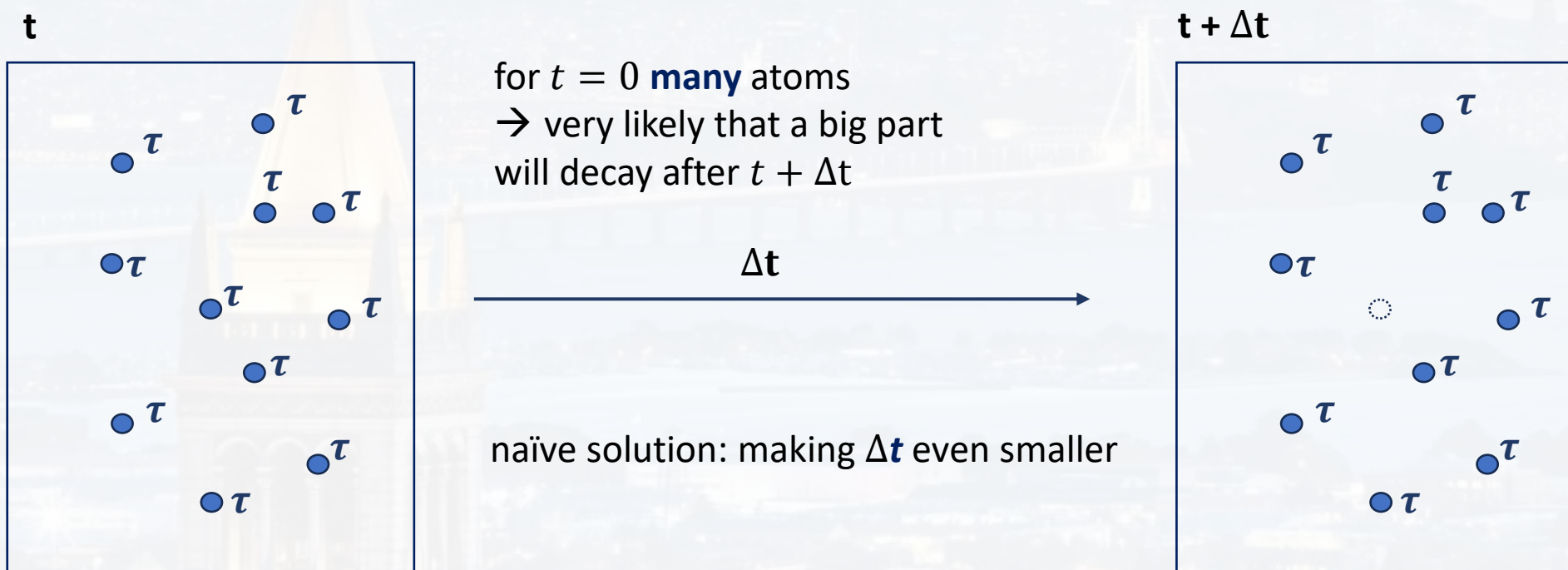


We can combine the MC method with different ways **how** to run a simulation

**example:**

- radioactive decay of atoms
- reaction  $A \xrightarrow{k} \emptyset$

**challenge:** on single particle level: decay/reaction (= changing **its state**) is **random**







We can combine the MC method with different ways **how** to run a simulation

**example:**

- radioactive decay of atoms
- reaction  $A \xrightarrow{k} \emptyset$

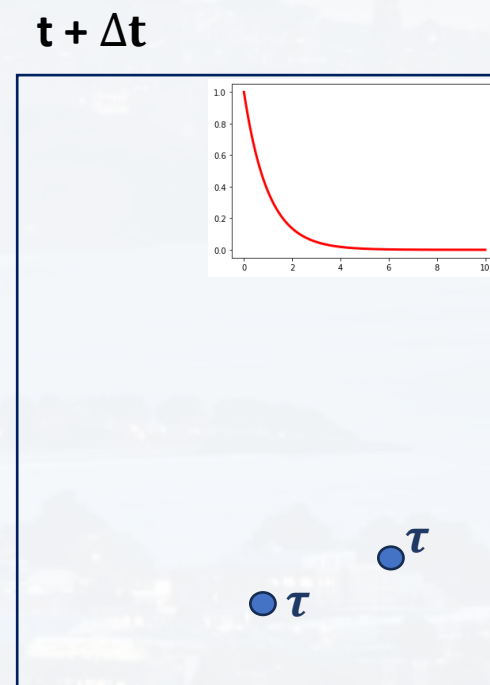
**challenge:** on single particle level: decay/reaction (= changing **its state**) is **random**



naïve solution: making  $\Delta t$  even smaller

$\Delta t$

problem: towards the end, only a **few atoms are left**  
→ we need to **wait longer** until one atom decays  
→ for many  $\Delta t$ : nothing happens





We can combine the MC method with different ways **how** to run a simulation

**example:**

- radioactive decay of atoms

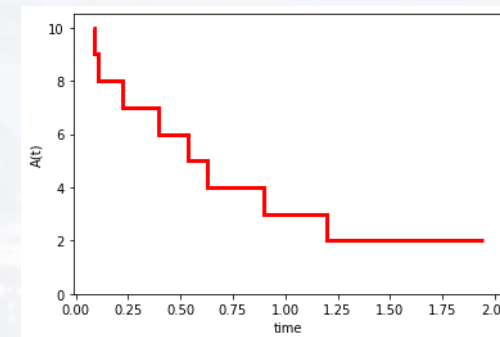
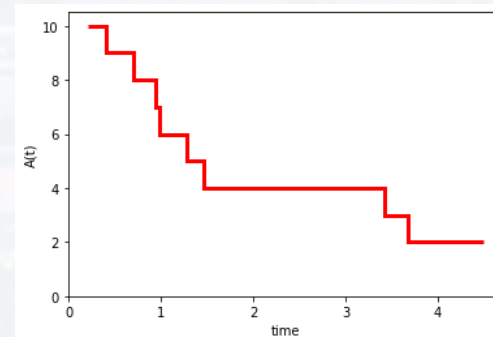
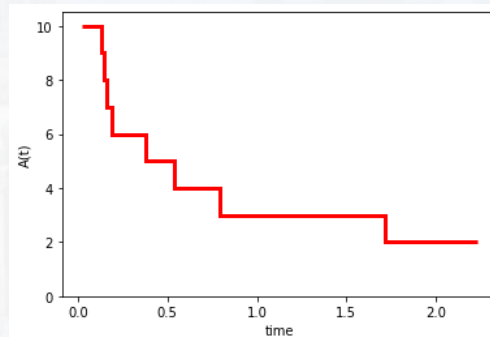
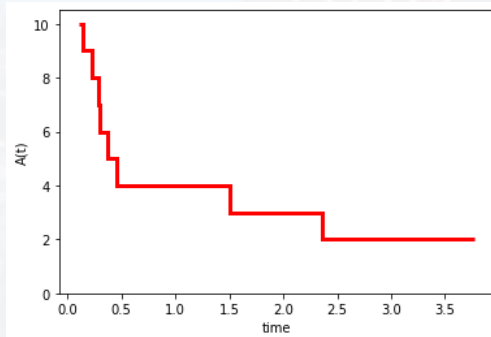
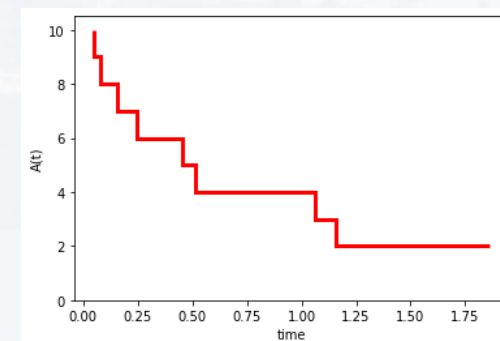
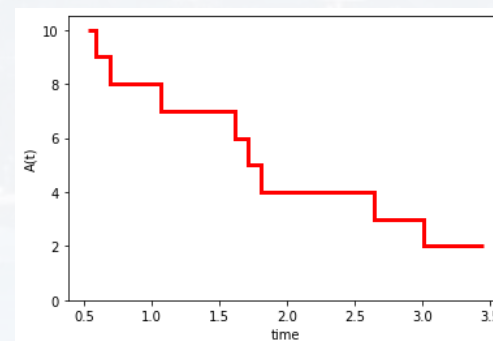
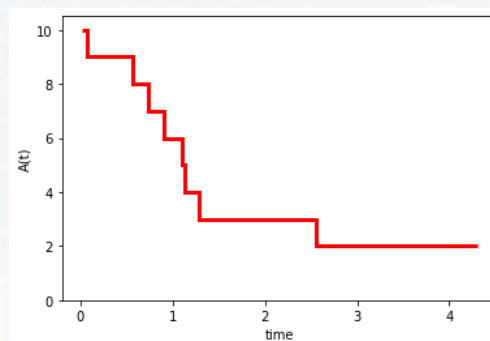
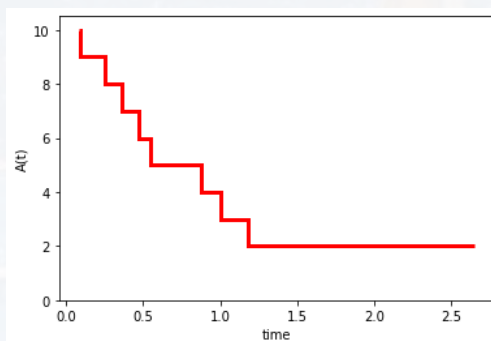
- reaction  $A \xrightarrow{k} \emptyset$

**challenge:**

- on single particle level: decay/reaction (= changing **its state**) is **random**

- avoiding  $\Delta t$  where nothing happens  $\rightarrow$  very inefficient

- modeling the **randomness**





We can combine the MC method with different ways **how** to run a simulation

**example:**

- radioactive decay of atoms
- reaction  $A \xrightarrow{k} \emptyset$

**standard MCS:**

- set a value  $\Delta t$
- check **if** an atom decays
  - if:  $N(t + \Delta t) = N(t) - 1$
  - else:  $N(t + \Delta t) = N(t)$
- $t \rightarrow t + \Delta t$  and repeat

**Leads to the problems mentioned earlier!**

**Gillespie:**

- based on  $N(t)$  and  $\tau$ , **calculate** the time  $\Delta t$  that elapses **until** one atom decays
- $N(t + \Delta t) = N(t) - 1$
- $t \rightarrow t + \Delta t$  and repeat
- $\Delta t$  adapts automatically to  $N(t)$





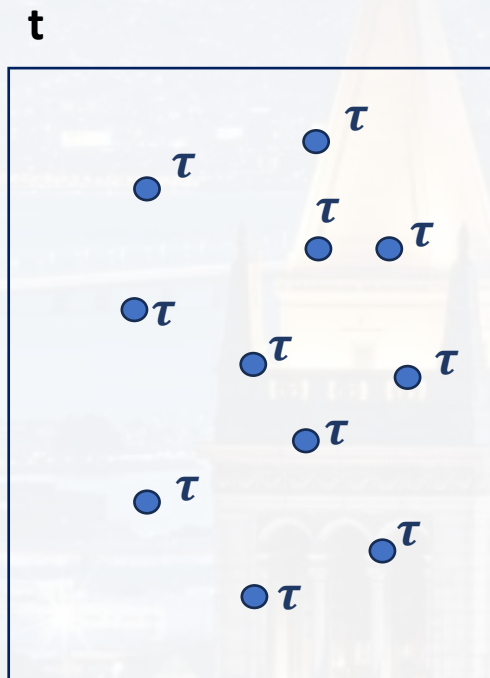
**Gillespie:**

- based on  $N(t)$  and  $\tau$ , **calculate** the time  $\Delta t$  that elapses **until** one atom decays
- $N(t + \Delta t) = N(t) - 1$
- $t \rightarrow t + \Delta t$  and repeat

- $\Delta t$  adapts automatically to  $N(t)$

decay is Poissonian:  $P(n|\tau) = \frac{(\tau \Delta t)^n}{n!} \exp(-\tau \Delta t)$

(see module 7)



aim:

$\Delta t$  **between** two decays

→ no decay within  $\Delta t$ , i. e.  $n = 0$

$$P(0|\tau) = \frac{(\tau \Delta t)^0}{0!} \exp(-\tau \Delta t) = \exp(-\tau \Delta t)$$

$$\Delta t = -\frac{1}{\tau} \ln[P(0|\tau)]$$



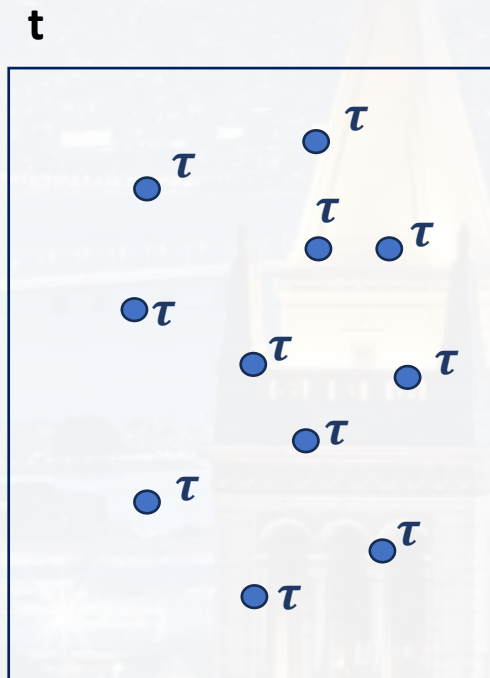
### Gillespie:

- based on  $N(t)$  and  $\tau$ , **calculate** the time  $\Delta t$  that elapses **until** one atom decays
- $N(t + \Delta t) = N(t) - 1$
- $t \rightarrow t + \Delta t$  and repeat

- $\Delta t$  adapts automatically to  $N(t)$

$$\Delta t = -\frac{1}{\tau} \ln[P(0|\tau)]$$

**Poissonian Stepper**



**each** atom has the probability of decay per time  $\tau$

atom 1 can undergo a decay, **or** atom 2 **or** atom 3 **or**...

logical **or**  $\rightarrow$  **adding** the probabilities (see module 6)

$$\tau \rightarrow \tau N(t)$$

$$\Delta t = -\frac{1}{\tau N(t)} \ln[P(0|\tau)]$$



**Gillespie:**

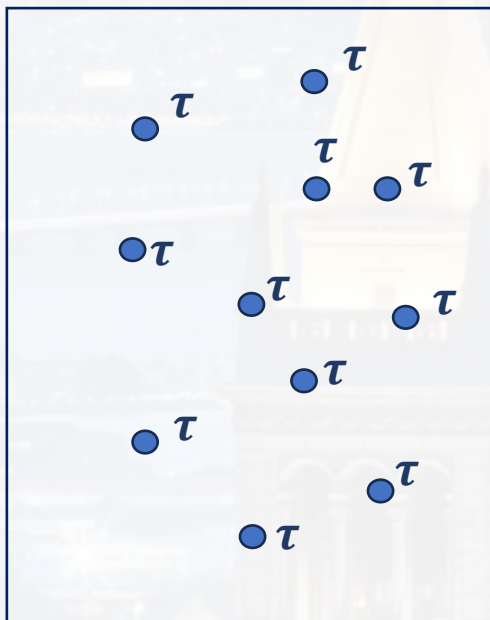
1) draw a **random number**  $\rho$  from a **uniform distribution** in the interval  $(0, 1)$

2) calculate the time  $\Delta t$  that elapses until the next decay

$$\Delta t = -\frac{1}{\tau N(t)} \ln \rho$$

3) set  $t \rightarrow t + \Delta t$  and  $N(t + \Delta t) = N(t) - 1$

4) repeat



$$\Delta t = -\frac{1}{\tau N(t)} \ln [P(0|\tau)]$$





**Gillespie:**

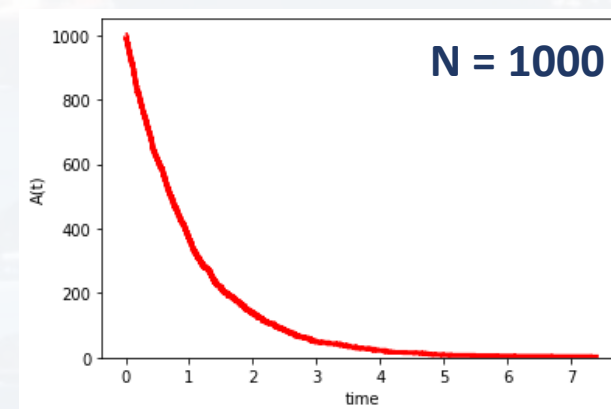
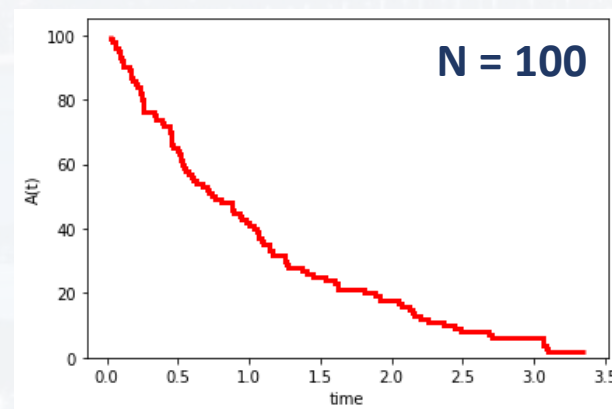
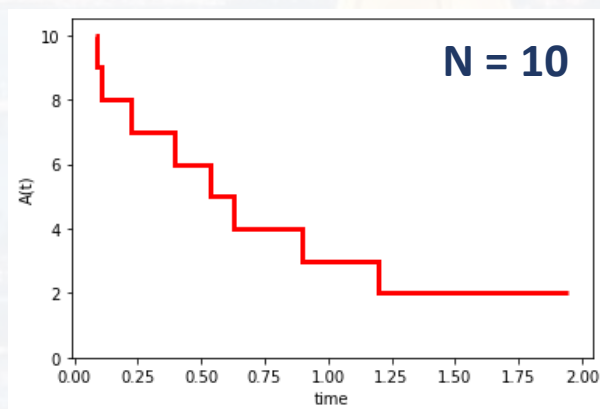
1) draw a **random number**  $\rho$  from a **uniform distribution** in the interval **(0, 1)**

2) calculate the time  $\Delta t$  that elapses until the next decay

$$\Delta t = -\frac{1}{\tau N(t)} \ln \rho$$

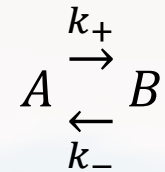
3) set  $t \rightarrow t + \Delta t$  and  $N(t + \Delta t) = N(t) - 1$

4) repeat





example:



need to calculate  $\Delta t$  for **a** reaction (either  $A \rightarrow B$  **or**  $B \rightarrow A$ )

logical **or**  $\rightarrow$  **adding** the probabilities (see module 6)

$$\begin{aligned} \tau(A) &\rightarrow k_+ A(t) & \tau(B) &\rightarrow k_- B(t) & \tau_{tot} &= \tau(A) + \tau(B) \\ & & & & &= k_+ A(t) + k_- B(t) \end{aligned}$$

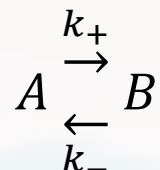
$$\Delta t = -\frac{1}{k_+ A(t) + k_- B(t)} \ln[P(0|\tau_{tot})]$$

time that elapses for **a** reaction to occur

next: deciding **which** reaction should occur



example:

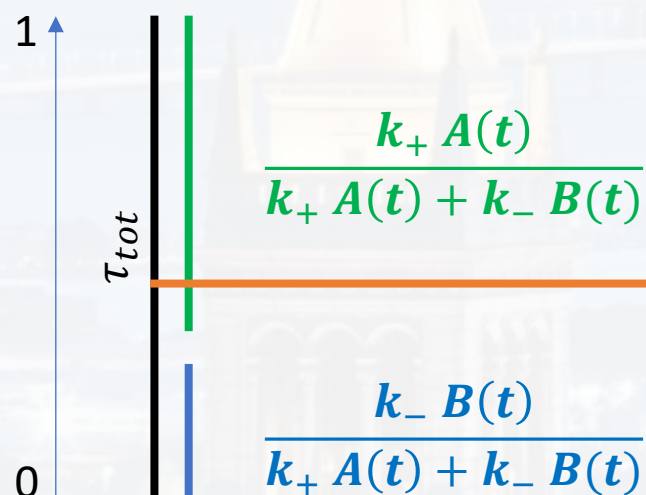


$$\tau(A) \rightarrow k_+ A(t) \quad \tau(B) \rightarrow k_- B(t) \quad \tau_{tot} = \tau(A) + \tau(B) = k_+ A(t) + k_- B(t)$$

$$\Delta t = -\frac{1}{k_+ A(t) + k_- B(t)} \ln[P(0|\tau_{tot})]$$

time that elapses for **a** reaction to occur

next: deciding **which** reaction should occur



depending into which fraction  
this random number falls  
→ this reaction occurs

generating a random number from a  
**uniform distribution** in the interval **(0, 1)**





**Gillespie:**

1) draw a **random number**  $\rho_1$  from a **uniform distribution** in the interval **(0, 1)**

2) calculate the time  $\Delta t$  that elapses until the next reaction

$$\Delta t = -\frac{1}{k_+ A(t) + k_- B(t)} \ln \rho_1$$

3) draw a **second random number**  $\rho_2$  from a **uniform distribution** in the interval **(0, 1)**

4) decide which reaction occurs:

$$\text{if } \rho_2 < \frac{k_+ A(t)}{k_+ A(t) + k_- B(t)} :$$

reaction  $A \rightarrow B$  is more likely

$$A(t + \Delta t) = A(t) - 1$$

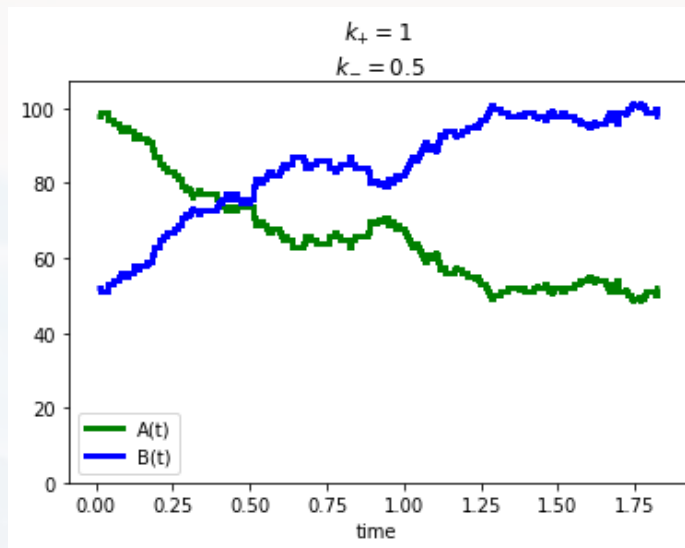
$$B(t + \Delta t) = B(t) + 1$$

else:

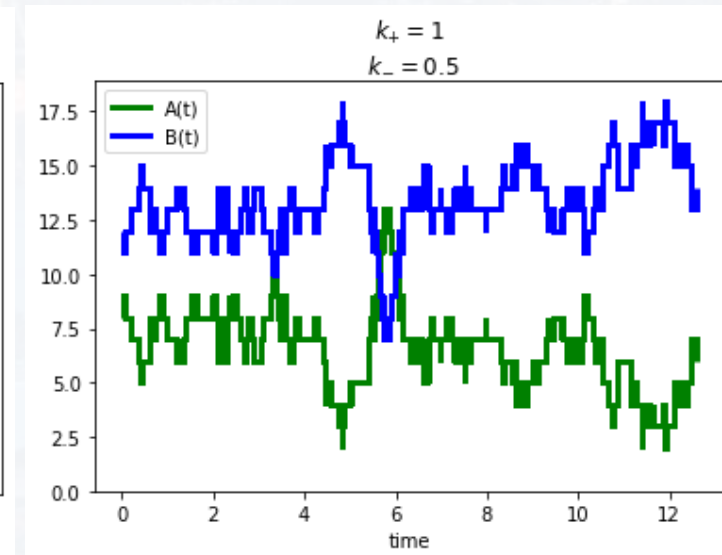
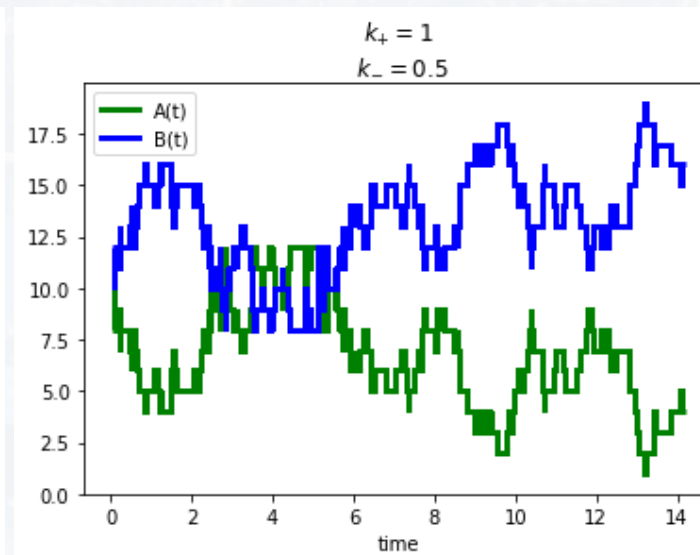
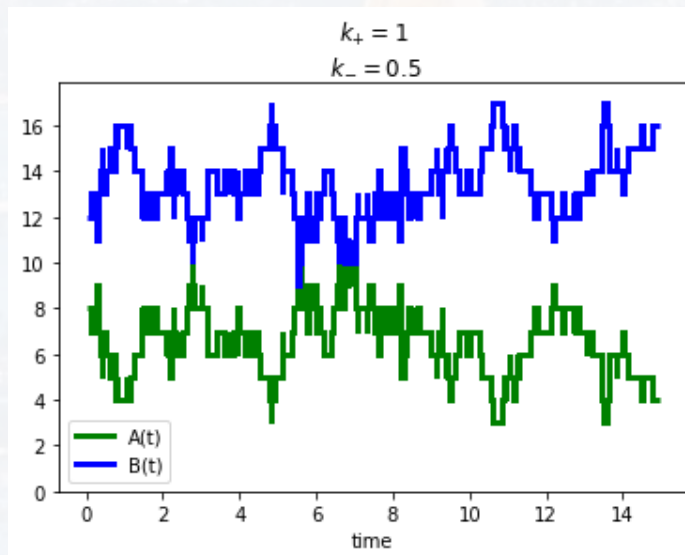
reaction  $B \rightarrow A$  is more likely

$$A(t + \Delta t) = A(t) + 1$$

$$B(t + \Delta t) = B(t) - 1$$

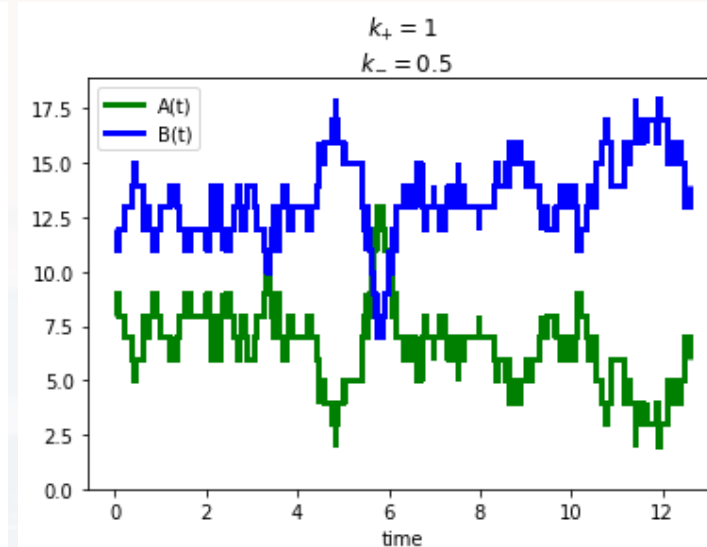
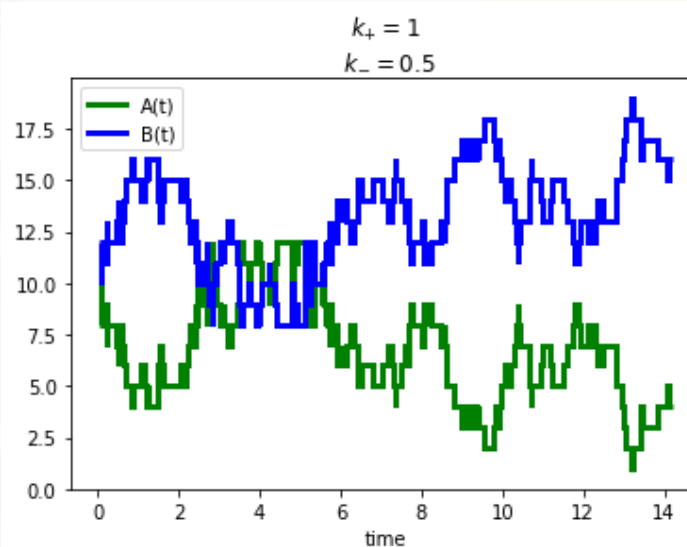
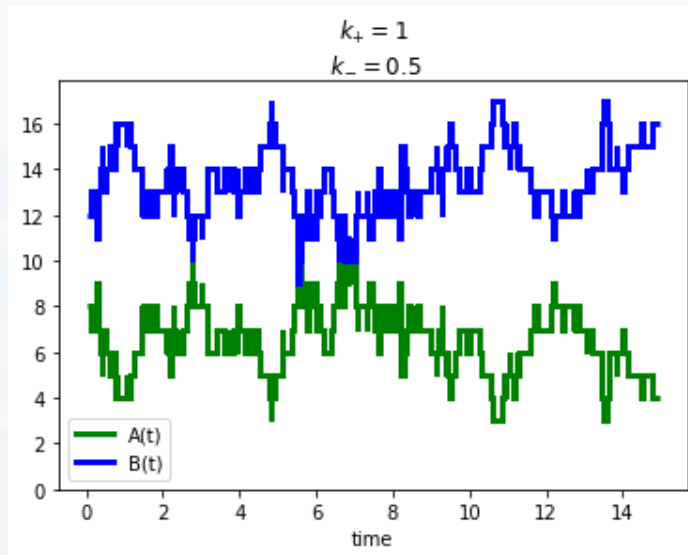


$N = 10$

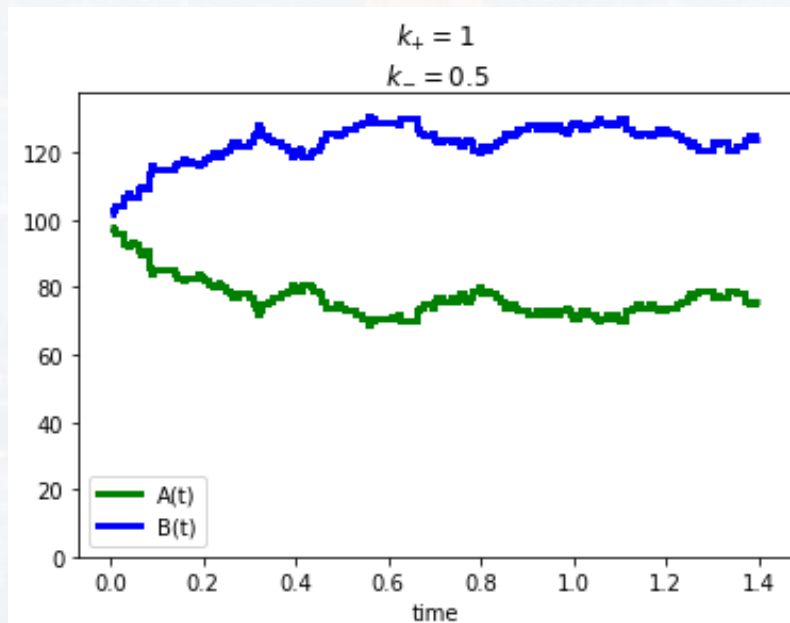




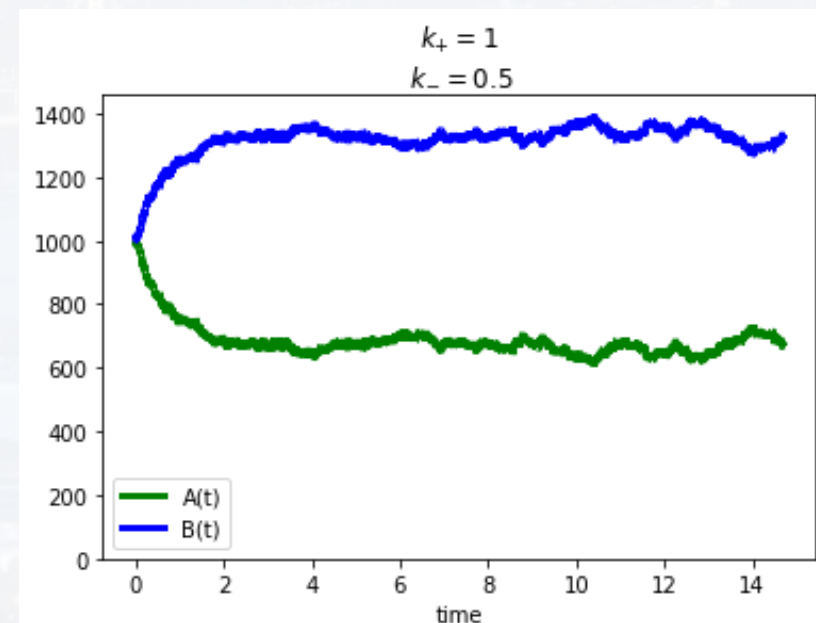
$N = 10$



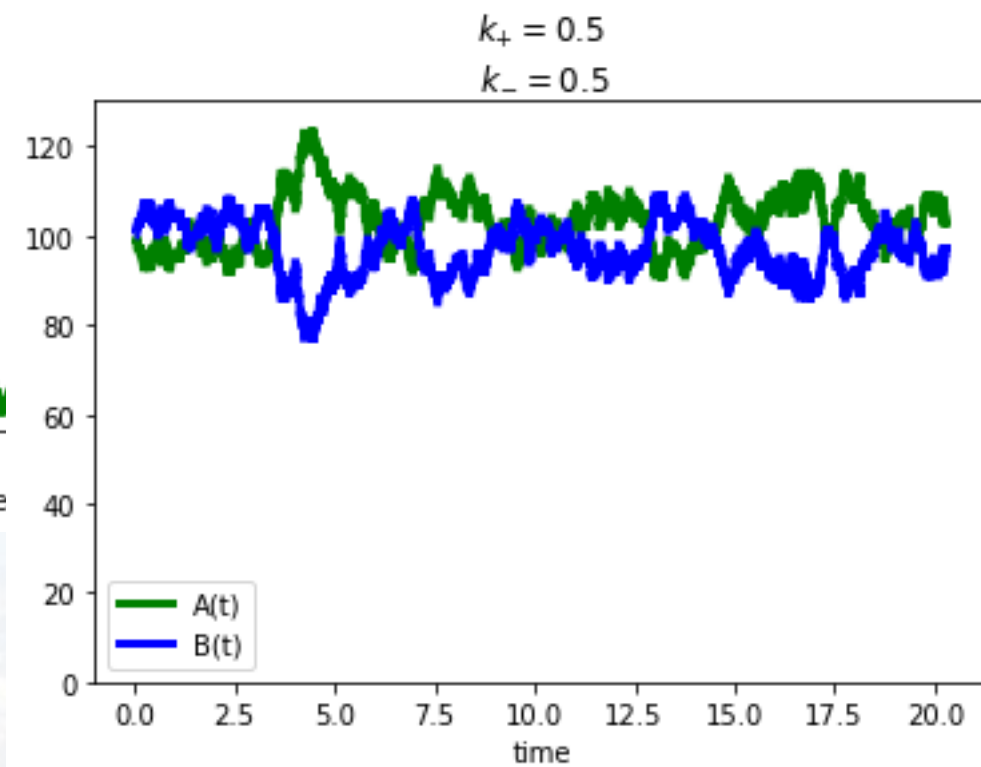
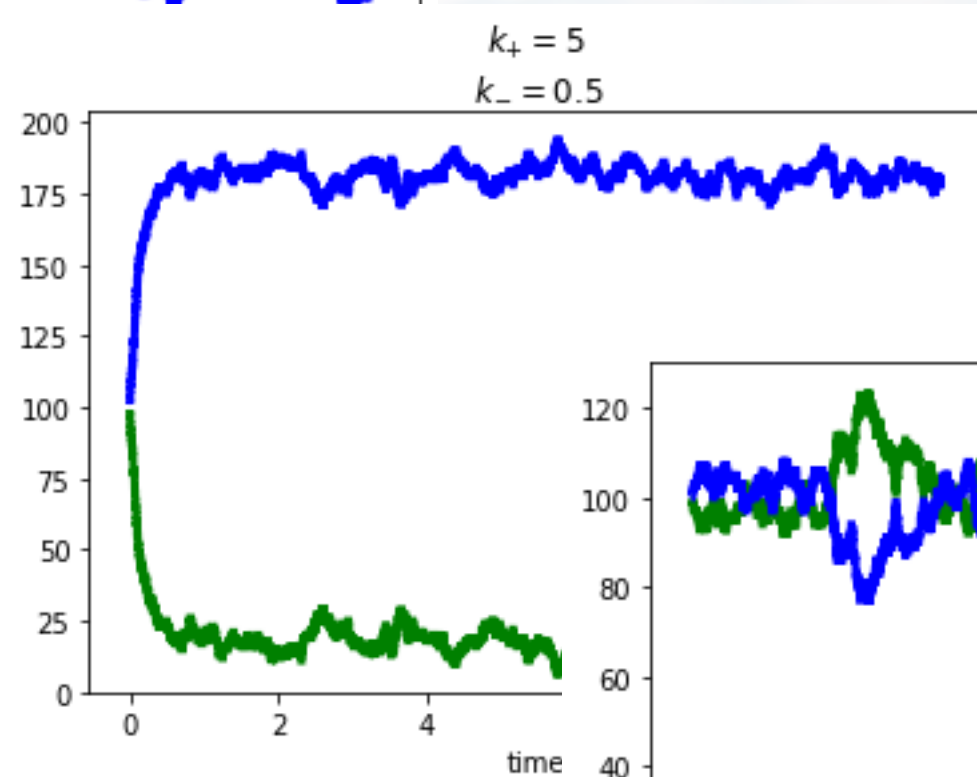
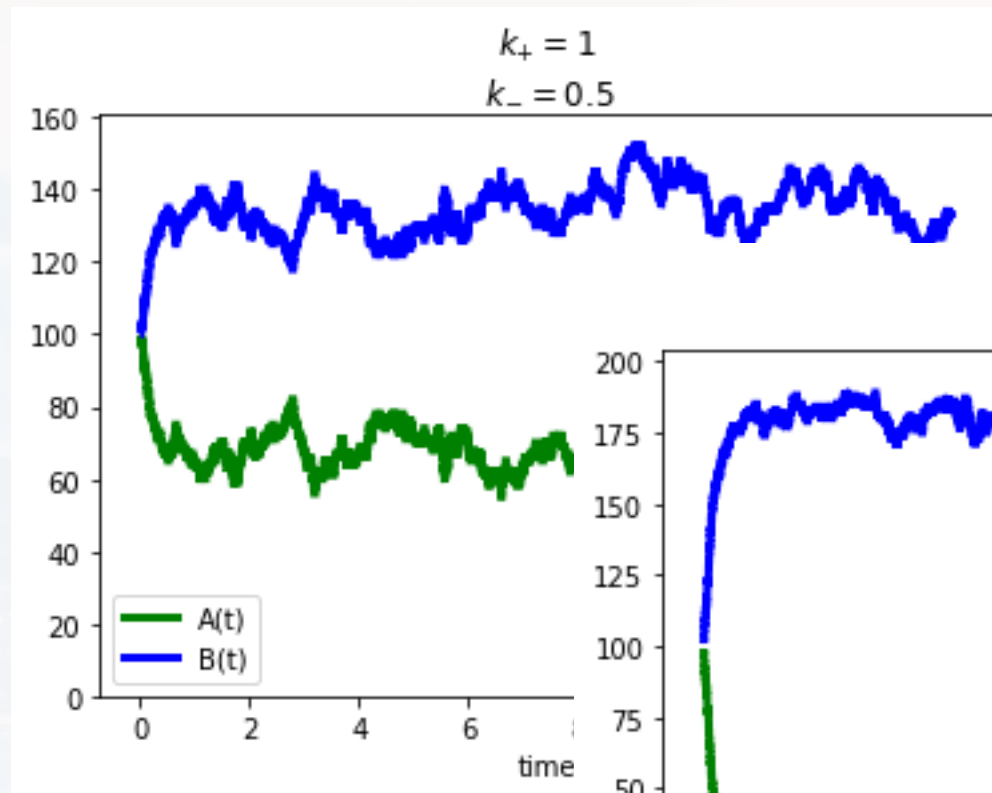
$N = 100$

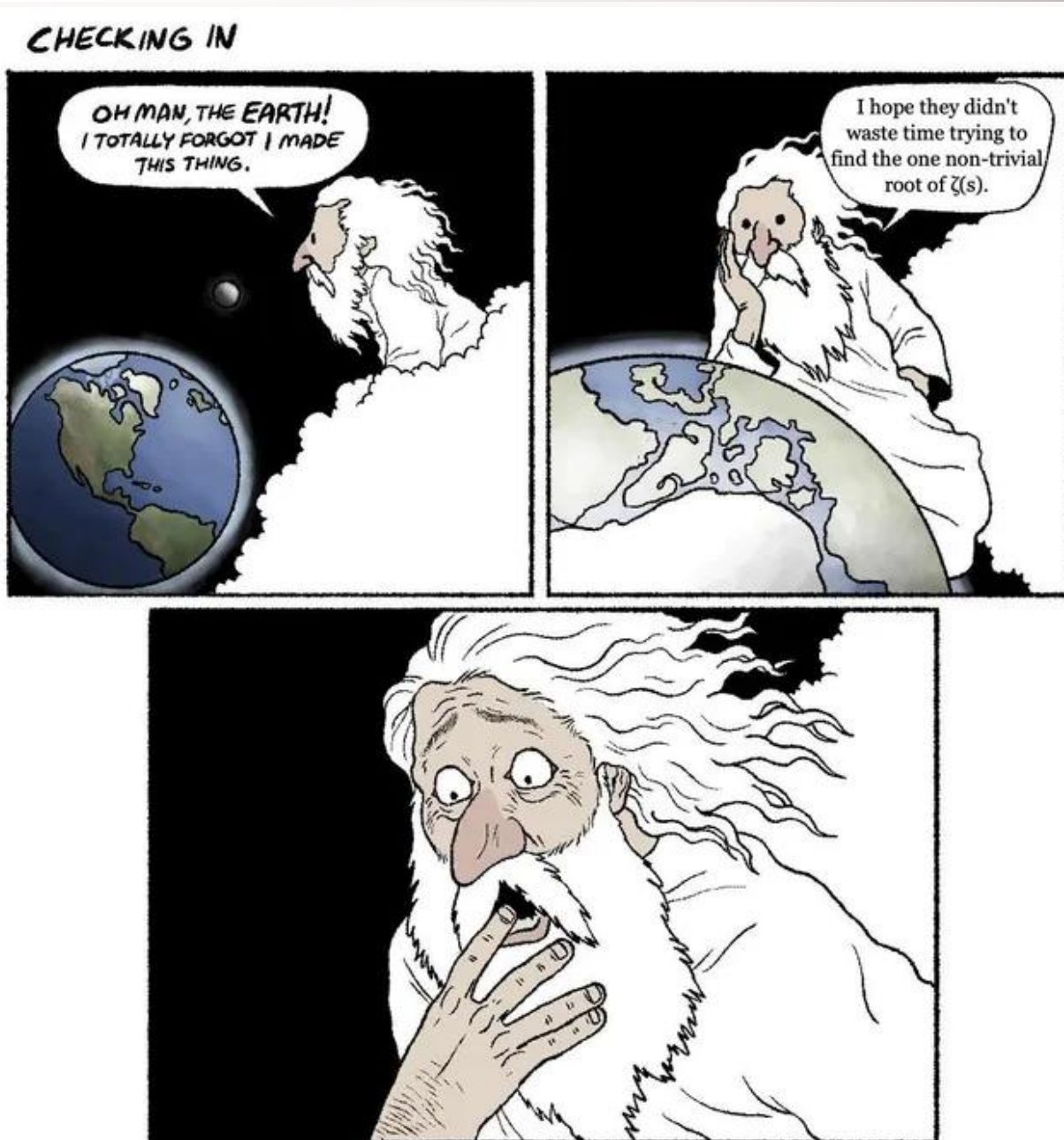


$N = 1000$









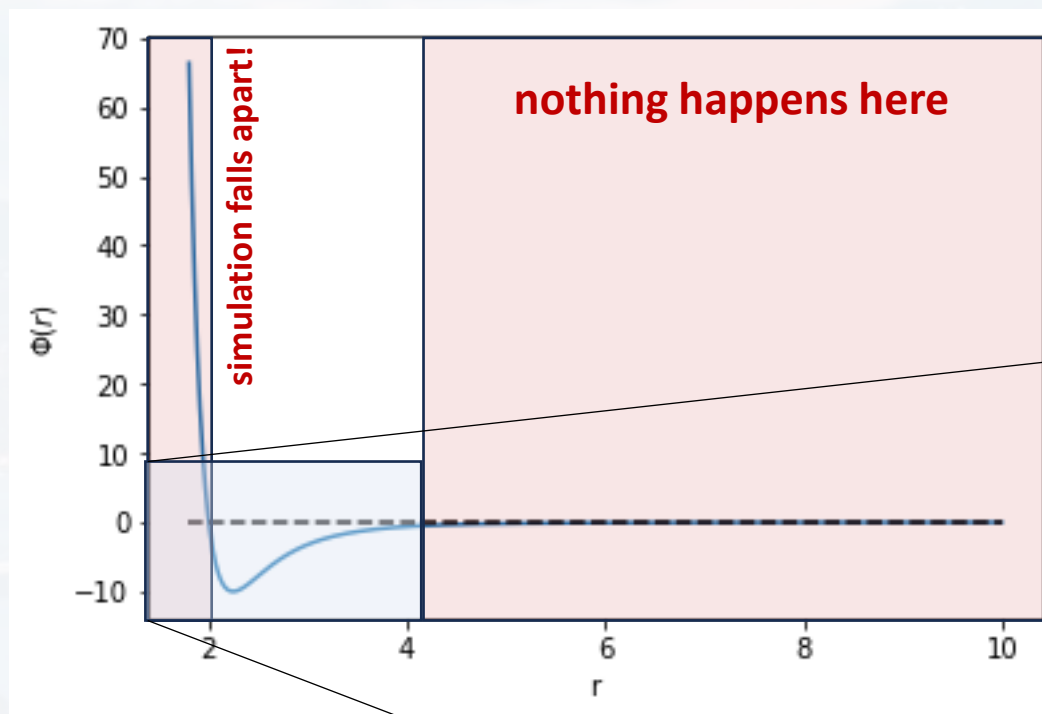
## Outline

- The Problem
- Finding PI
- Gillespie Algorithm
- **Metropolis Algorithm**

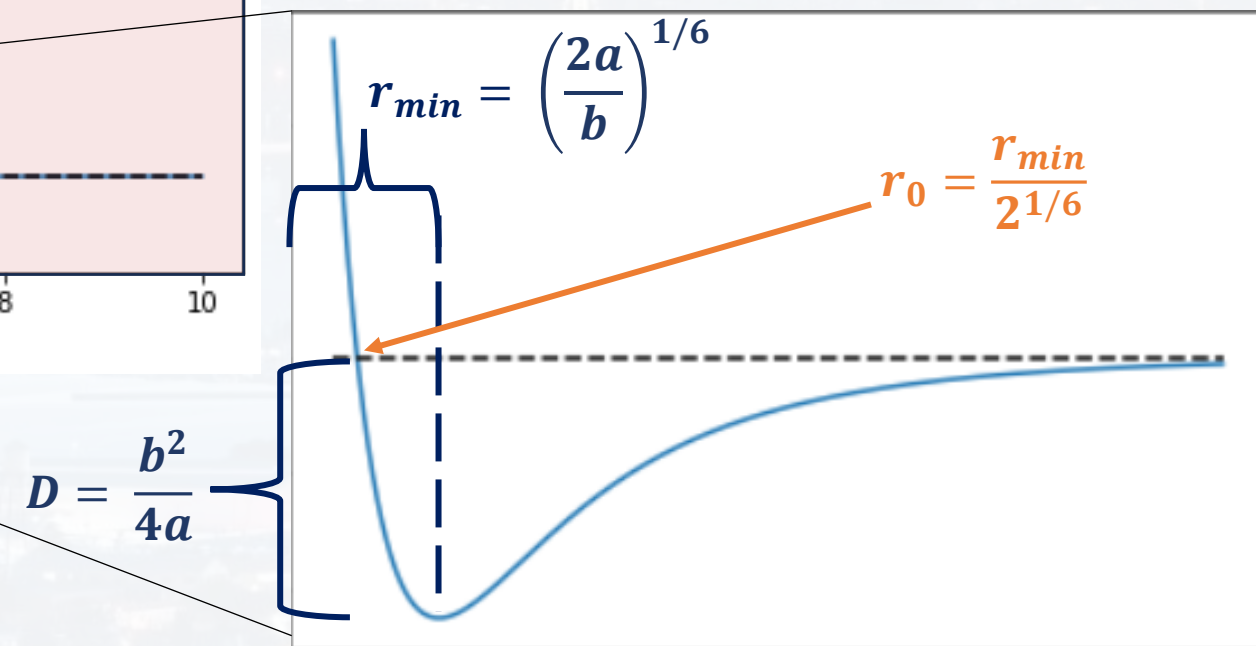


We can combine the MC method with different ways **how** to run a simulation

**example:** - molecules and the Lennard – Jones potential



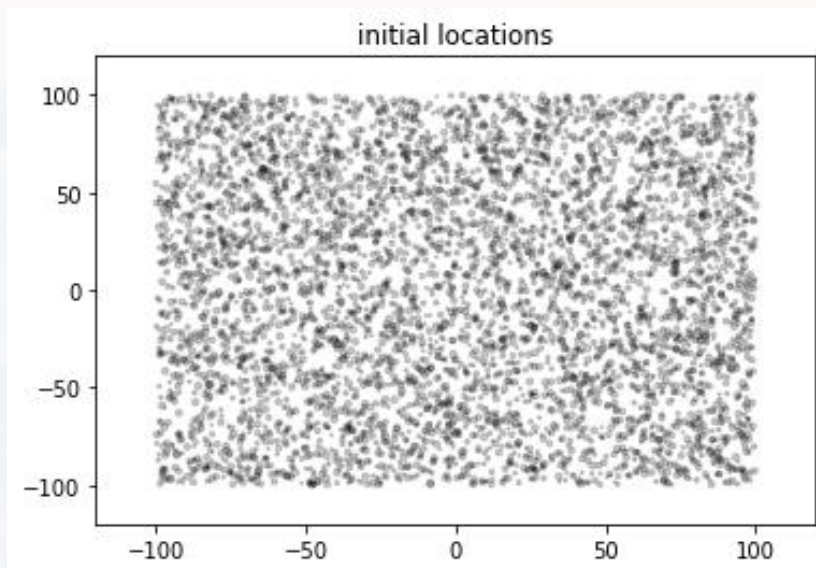
$$U_{LJ} = \frac{a}{r^{12}} - \frac{b}{r^6} \quad a, b = \text{const}$$







the problem:

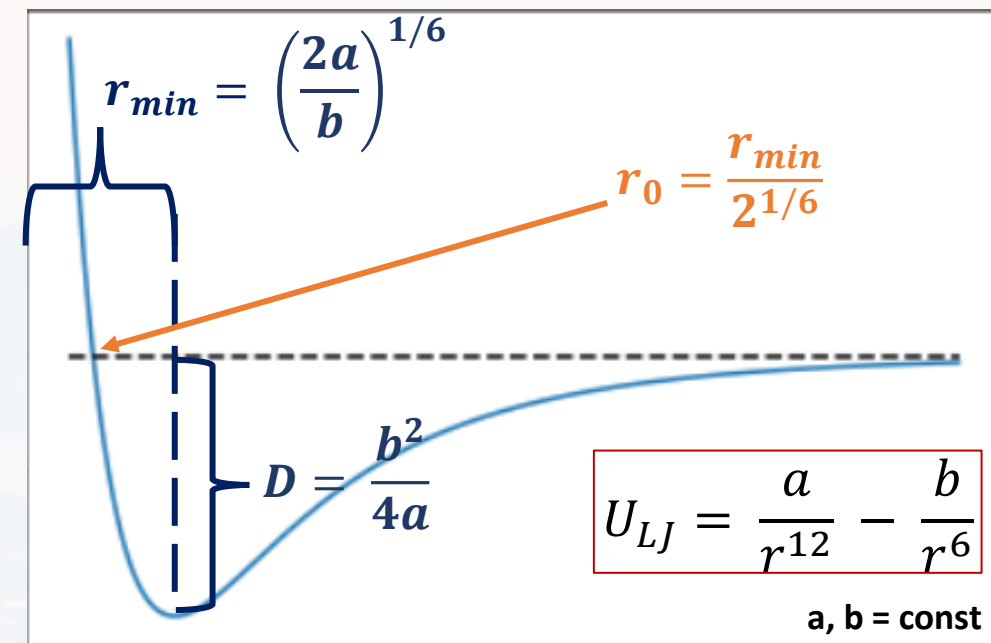


simulating many particles

Naïve solution: solving **Newton's equation of motion**

$$x_{t+\Delta t} = x_t + v(x)_t \cdot \Delta t + \frac{1}{2} a(x)_t \Delta t^2$$

$$y_{t+\Delta t} = y_t + v(y)_t \cdot \Delta t + \frac{1}{2} a(y)_t \Delta t^2$$



total force/potential  
that acts on the particle

$$a(x) = \frac{F(x)_{tot}}{m} = \frac{1}{m} \frac{\partial U_{tot}(x, y)_{LJ}}{\partial x}$$

$$a(y) = \frac{F(y)_{tot}}{m} = \frac{1}{m} \frac{\partial U_{tot}(x, y)_{LJ}}{\partial y}$$

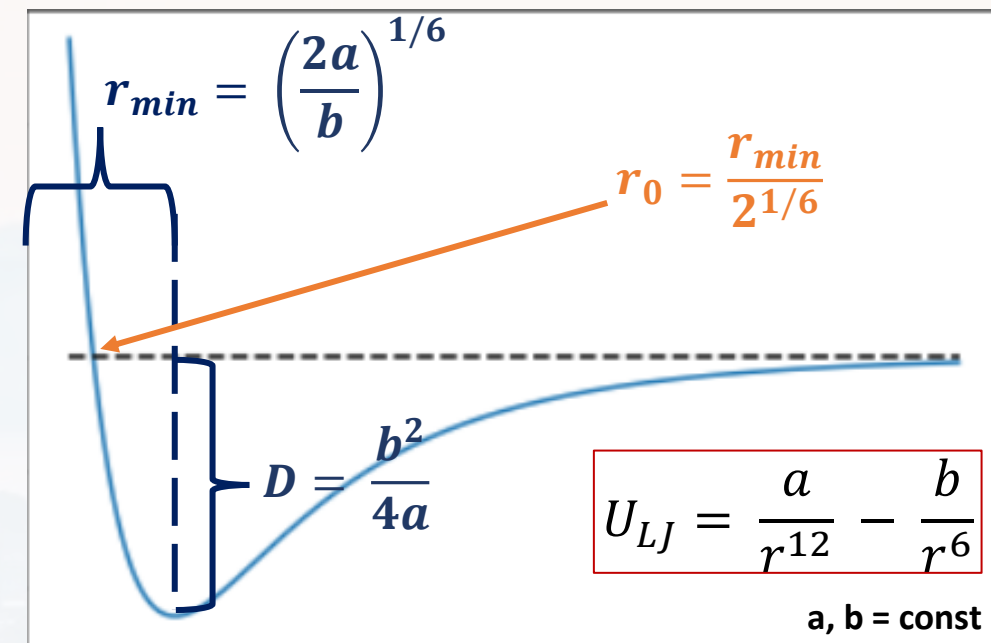


the problem:

Naïve solution: solving **Newton's equation of motion**

$$x_{t+\Delta t} = x_t + v(x)_t \cdot \Delta t + \frac{1}{2m} \frac{\partial U_{tot}(x, y)_{LJ}}{\partial x} \bigg|_t \Delta t^2$$

$$y_{t+\Delta t} = y_t + v(y)_t \cdot \Delta t + \frac{1}{2m} \frac{\partial U_{tot}(x, y)_{LJ}}{\partial y} \bigg|_t \Delta t^2$$



We pick a specific value for  $\Delta t$  and update locations, velocities and acceleration

for particles with  $r \approx r_0$ :

- $\frac{\partial U(x, y)_{LJ}}{\partial y}$  or  $\frac{\partial U(x, y)_{LJ}}{\partial x}$  explode
- particles get kicked out
- wouldn't have gotten so close in the first place  $\rightarrow \Delta t$  **too large**

for particles with  $r \gg r_0$ :

- $\frac{\partial U(x, y)_{LJ}}{\partial y}$  or  $\frac{\partial U(x, y)_{LJ}}{\partial x} \approx 0$
- nothing happens, very inefficient  $\rightarrow \Delta t$  **too small**



the solution:

Gillespie: calculating **when** a particle changes its state

Metropolis: calculating **if** a particle changes its state (**here locations, velocities and acceleration**)

If  $dU_{tot}(x, y)_{LJ}$  is **negative**: → **always move** (a ball always rolls down the hill)

If  $dU_{tot}(x, y)_{LJ}$  is **positive**: → calculate the **probability to move**

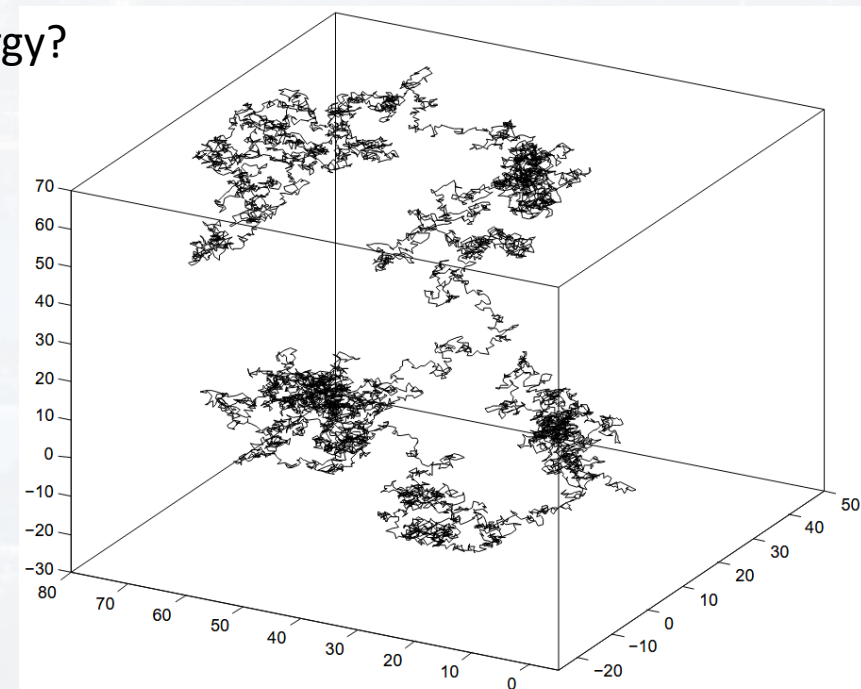
Why should a particle move, even though that increases its energy?

reason: random **Brownian motion**

one can show (statistical physics, max entropy)

$$p_{move} \sim \exp \left[ -\frac{dU_{tot}(x, y)_{LJ}}{T} \right] \quad \text{Boltzmann factor}$$

T: temperature



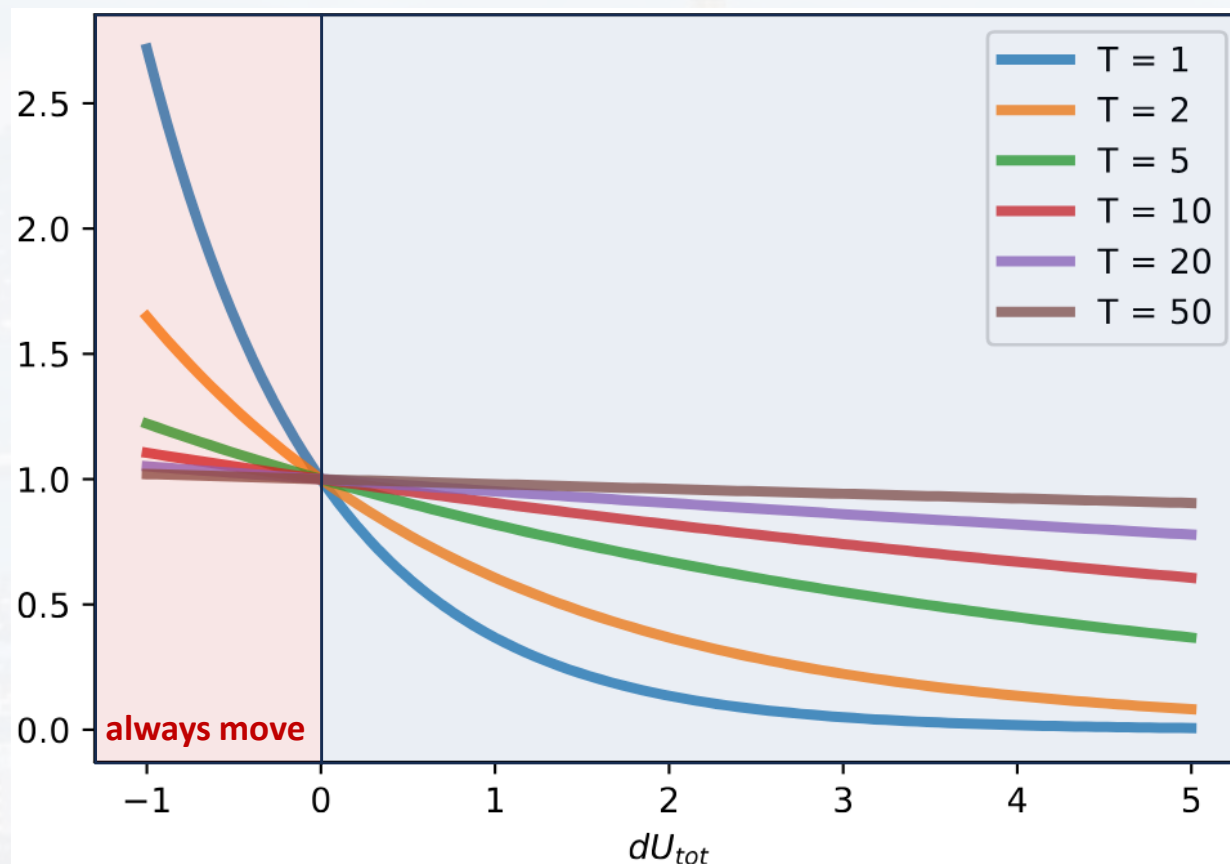




Metropolis: calculating **if** a particle changes its state (here locations, velocities and acceleration)

If  $dU_{tot}(x, y)_{LJ}$  is **negative**: → **always move** (a ball always rolls down the hill)

If  $dU_{tot}(x, y)_{LJ}$  is **positive**: → calculate the **probability to move**



$$p_{move} \sim \exp \left[ -\frac{dU_{tot}(x, y)_{LJ}}{T} \right]$$

Boltzmann factor  
T: temperature

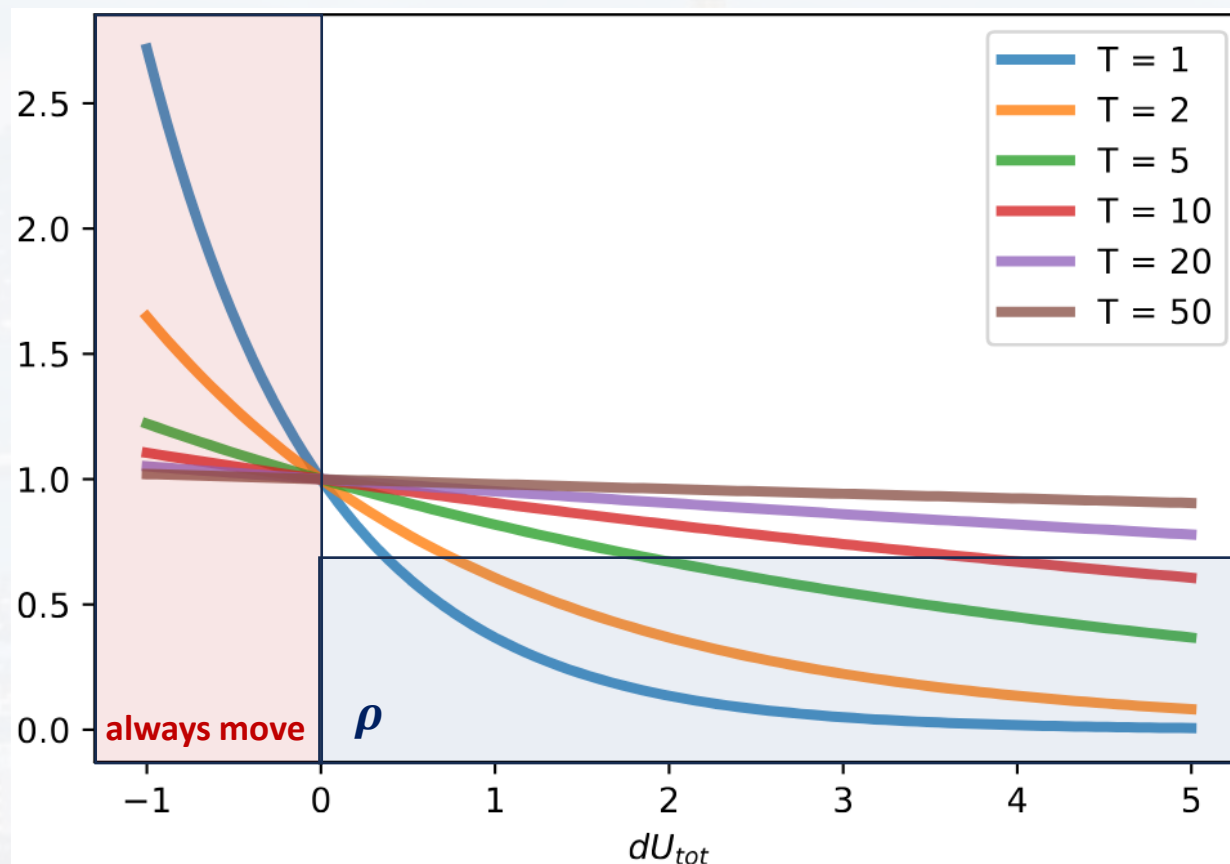
draw a **random number**  $\rho$  from a **uniform distribution** in the interval **(0, 1)**



Metropolis: calculating **if** a particle changes its state (here locations, velocities and acceleration)

If  $dU_{tot}(x, y)_{LJ}$  is **negative**:  $\rightarrow$  **always move** (a ball always rolls down the hill)

If  $dU_{tot}(x, y)_{LJ}$  is **positive**:  $\rightarrow$  calculate the **probability to move**



$$p_{move} \sim \exp \left[ -\frac{dU_{tot}(x, y)_{LJ}}{T} \right]$$

Boltzmann factor  
T: temperature

draw a **random number**  $\rho$  from a **uniform distribution** in the interval **(0, 1)**

if  $\rho < p_{move} \rightarrow$  move

(the larger  $dU_{tot}$ , the smaller  $p_{move}$ , the harder to meet the condition  $\rho < p_{move}$ )



**Metropolis:**

1) suggest a random move  $\Delta x$  and  $\Delta y$  for all particles

2) calculate  $\Delta U_{tot}(x, y)_{LJ}$  based on  $\Delta x$  and  $\Delta y$  for each particle

3) move or not:

a) move those particles where  $\Delta U_{tot}(x, y)_{LJ} < 0$

b) for those particles where  $\Delta U_{tot}(x, y)_{LJ} > 0$

- draw a **random number**  $\rho$  from a **uniform distribution** in the interval  $(0, 1)$

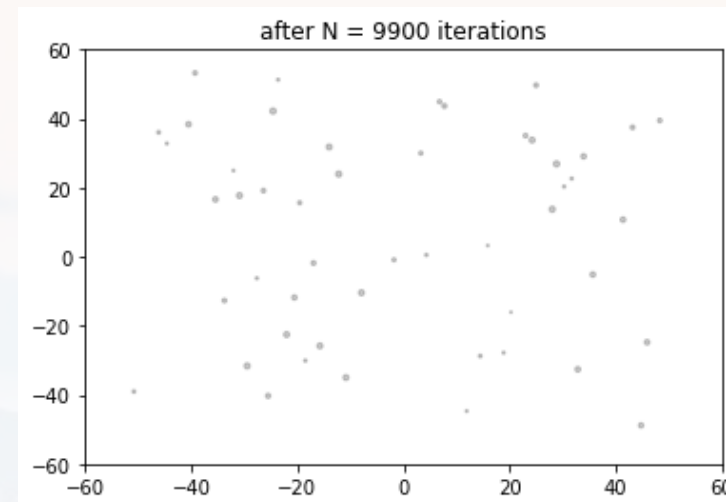
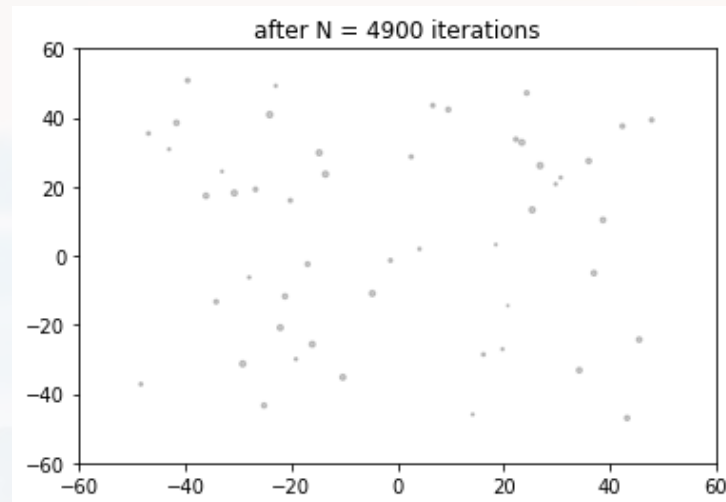
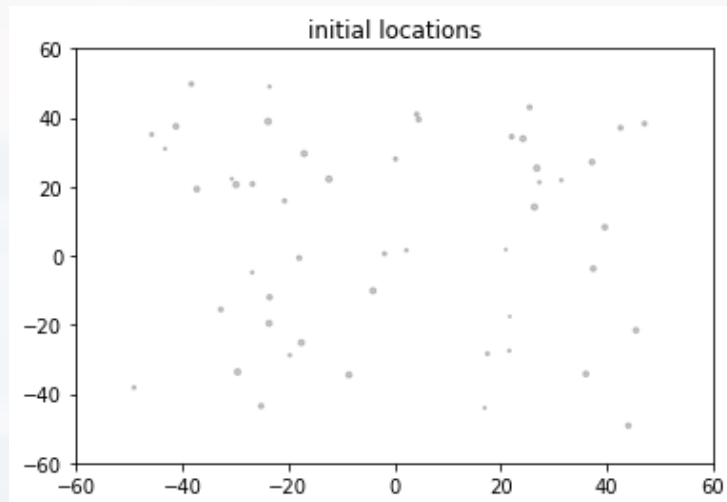
- move those particles for which  $\rho < \exp \left[ -\frac{\Delta U_{tot}(x, y)_{LJ}}{T} \right]$

4) repeat

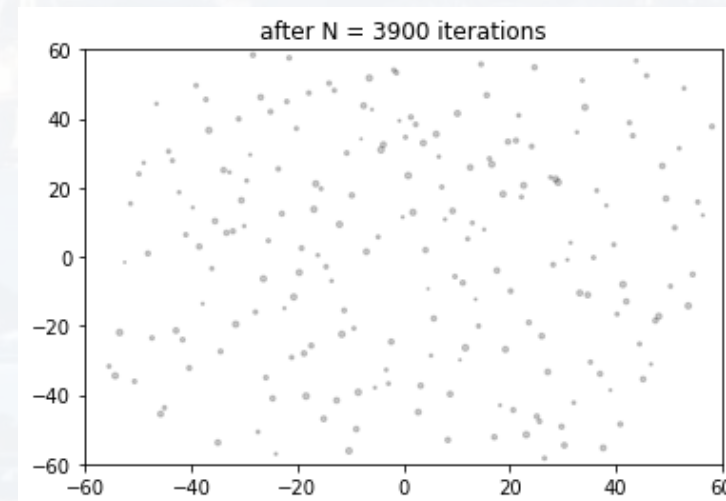
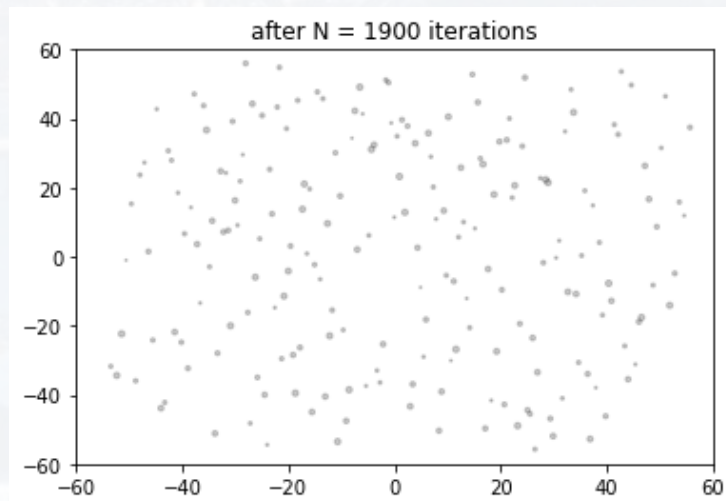
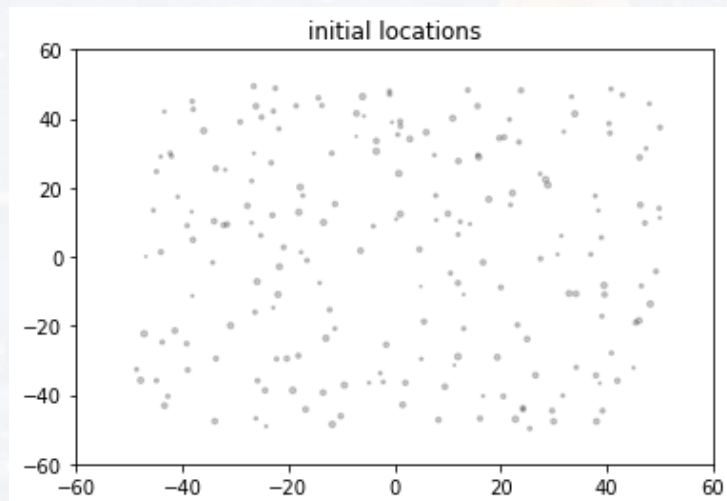


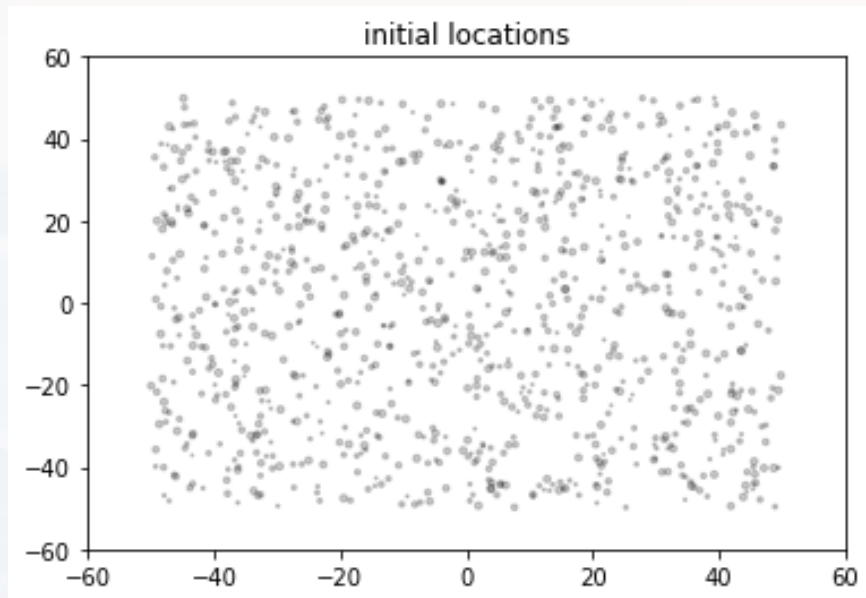


**N = 50**

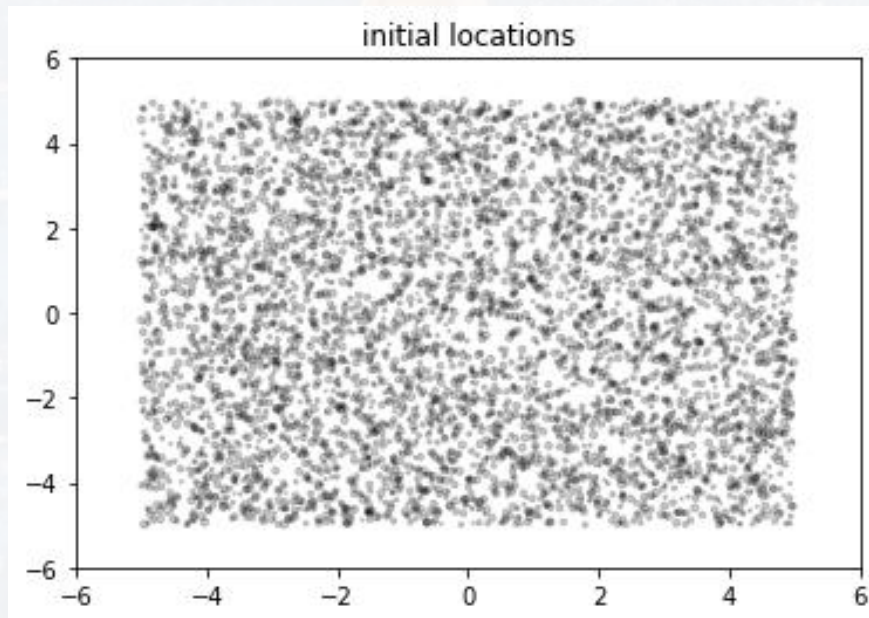
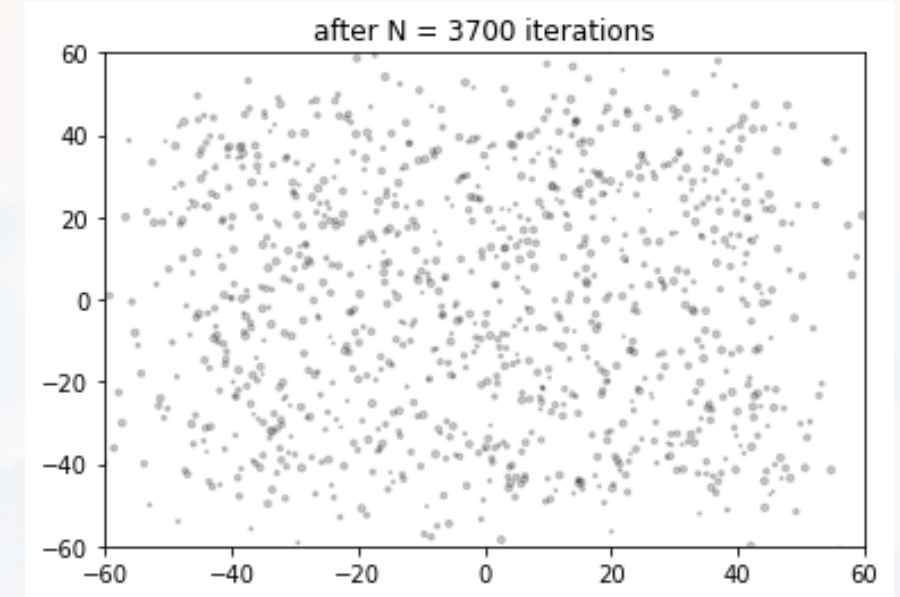


**N = 200**

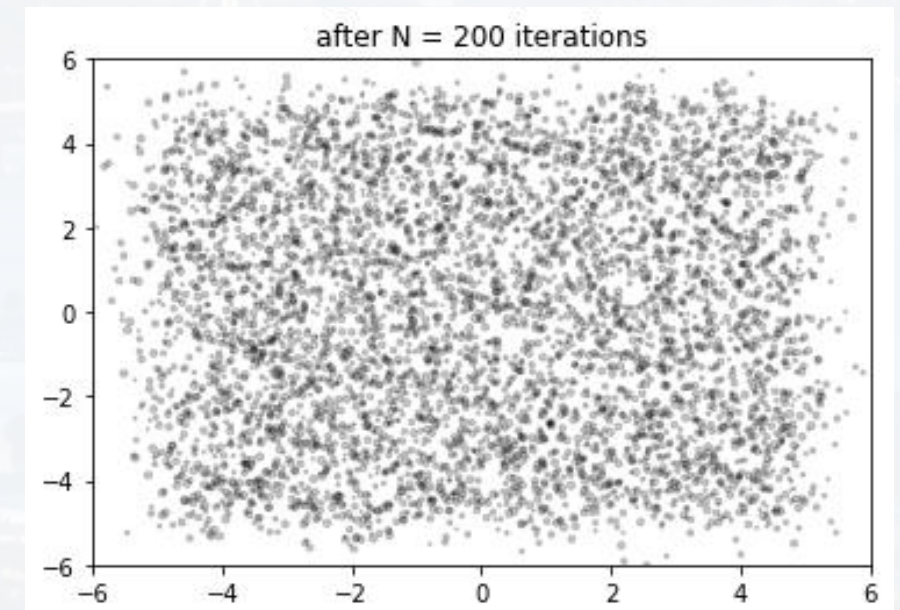




**N = 1000**



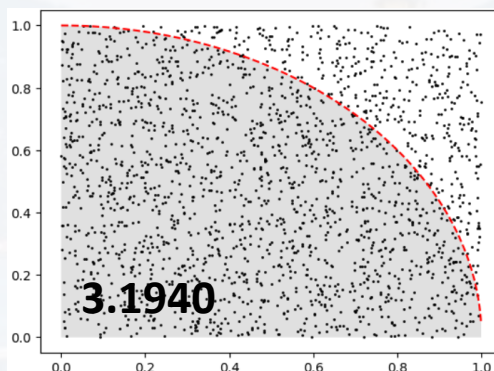
**N = 5000**



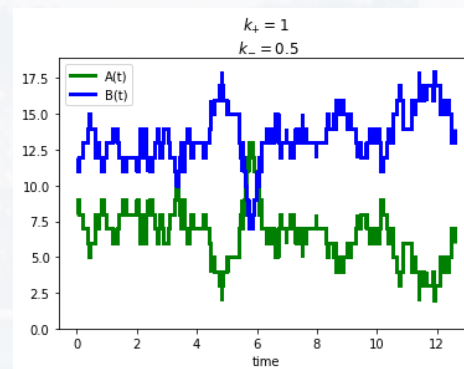


## summary

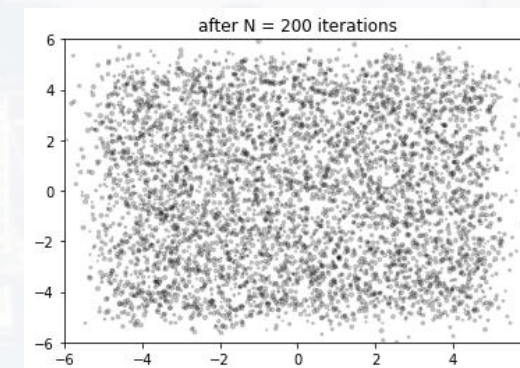
- 1) draw random numbers from a uniform distribution (unbiased)
- 2) compare these numbers to a **distribution  $d$  that characterizes the systems dynamics**



no bias,  $d$  is uniform



natural numbers:  
 $d$  is Poissonian



energy is conserved:  
 $d$  is a Boltzmann distribution

$$\Delta t = -\frac{1}{k_+ A(t) + k_- B(t)} \ln \rho_1$$

$$p_{move} \sim \exp \left[ -\frac{dU_{tot}(x, y)_{LJ}}{T} \right]$$





## summary

1) draw random numbers from a uniform distribution (unbiased)

2) compare these numbers to a **distribution  $d$  that characterizes the systems dynamics**

How do I know which  $d$  to pick?

Distribution name	Probability density / mass function	Maximum Entropy constraint	Support
Uniform (discrete)	$f(k) = \frac{1}{b-a+1}$	None	$\{a, a+1, \dots, b-1, b\}$
Uniform (continuous)	$f(x) = \frac{1}{b-a}$	None	$[a, b]$
Bernoulli	$f(k) = p^k (1-p)^{1-k}$	$\mathbb{E}[K] = p$	$\{0, 1\}$
Geometric	$f(k) = (1-p)^{k-1} p$	$\mathbb{E}[K] = \frac{1}{p}$	$\mathbb{N} \setminus \{0\} = \{1, 2, 3, \dots\}$
Exponential	$f(x) = \lambda \exp(-\lambda x)$	$\mathbb{E}[X] = \frac{1}{\lambda}$	$[0, \infty)$
Laplace	$f(x) = \frac{1}{2b} \exp\left(-\frac{ x-\mu }{b}\right)$	$\mathbb{E}[ X-\mu ] = b$	$(-\infty, \infty)$
Asymmetric Laplace	$f(x) = \frac{\lambda \exp\left(-(x-m)\lambda s \kappa^s\right)}{\left(\kappa + \frac{1}{\kappa}\right)}$ where $s \equiv \text{sgn}(x-m)$	$\mathbb{E}[(X-m)s \kappa^s] = \frac{1}{\lambda}$	$(-\infty, \infty)$
Pareto	$f(x) = \frac{\alpha x_m^\alpha}{x^{\alpha+1}}$	$\mathbb{E}[\ln X] = \frac{1}{\alpha} + \ln(x_m)$	$[x_m, \infty)$
Normal	$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$\mathbb{E}[X] = \mu$ , $\mathbb{E}[X^2] = \sigma^2 + \mu^2$	$(-\infty, \infty)$

At the end... all probability distributions are **Maximum Entropy** distributions, subject to a **set of constraints**

(see module 7)



Thank you very much for your attention!

