Berkeley MSSE
MOLECULAR SCIENCE & SOFTWARE ENGINEERING

Lecture 10:

Simulation and Monte Carlo Method

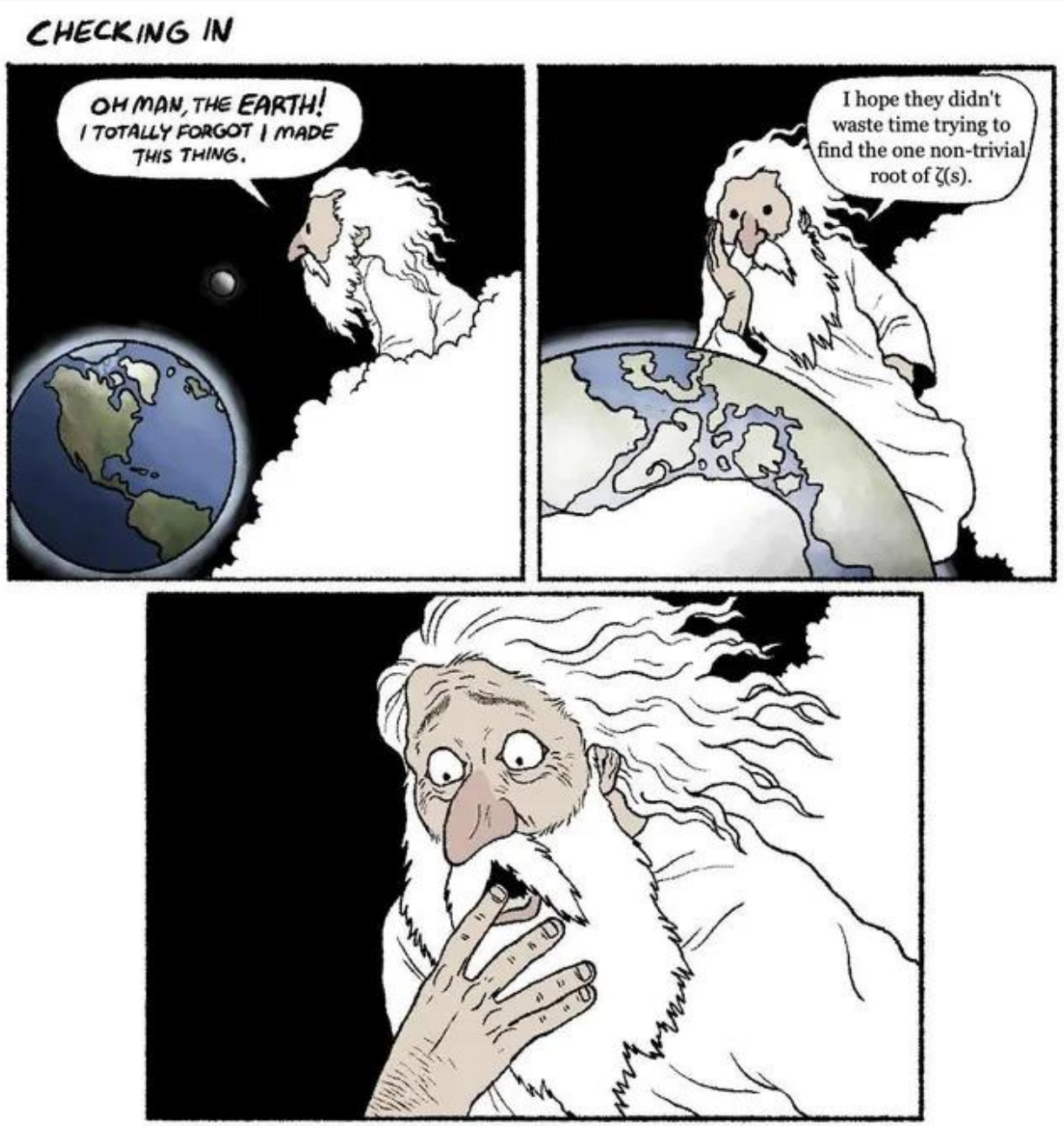
Markus Hohle

University California, Berkeley

Numerical Methods for Computational Science

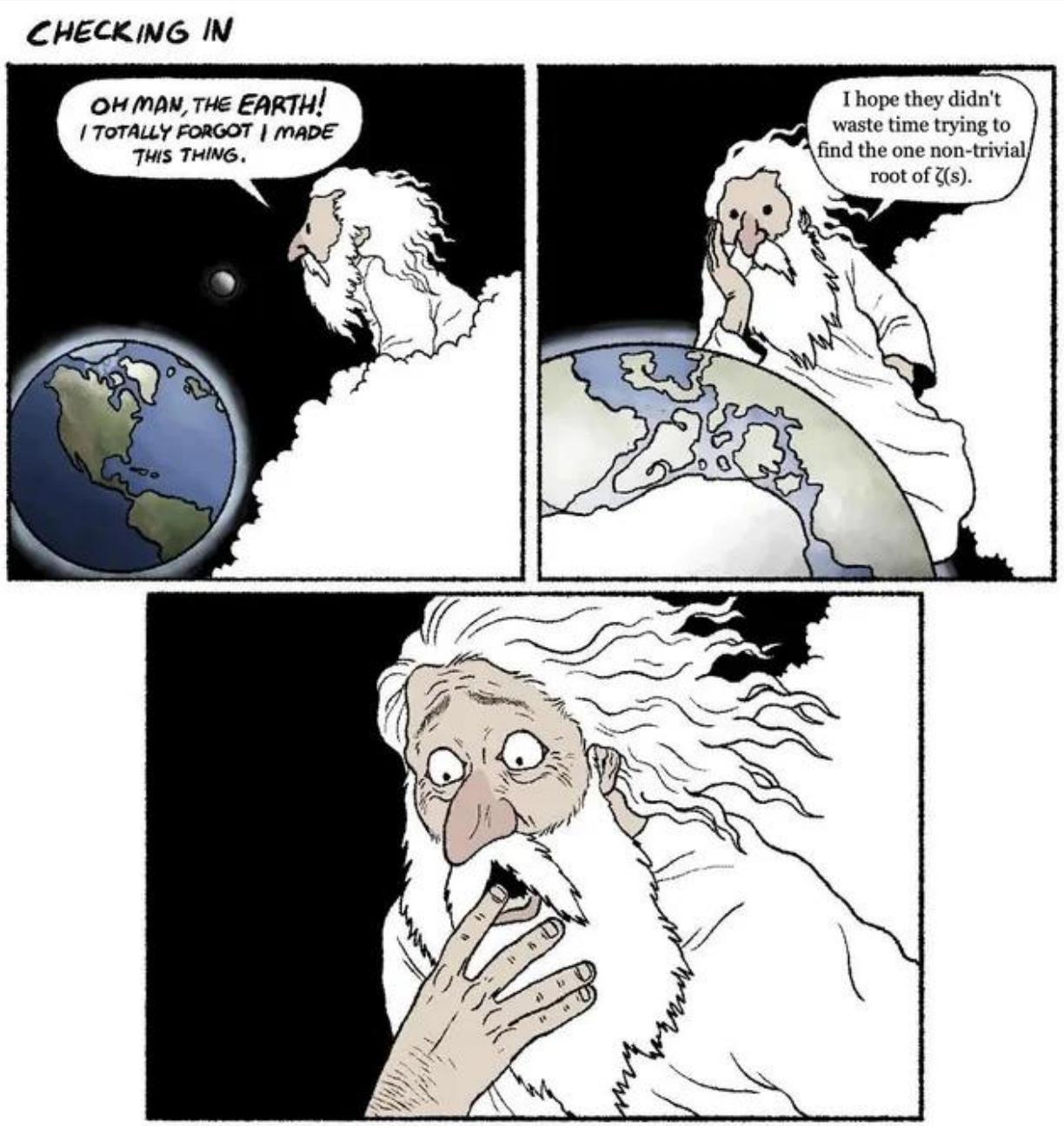
Course Map

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
Week 10:	**Simulation and Monte Carlo Method**
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

Simulation and Monte Carlo Method:

- 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 → standard simulations are not possible

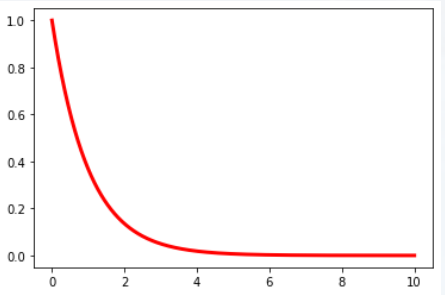
degradation of a chemical compound *A* with rate *k*

classic and deterministic approach: many particles of A → concentration [A]

$A \xrightarrow{k} \emptyset$ What is $A(t)$?

constant relative change per **time step**

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



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

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

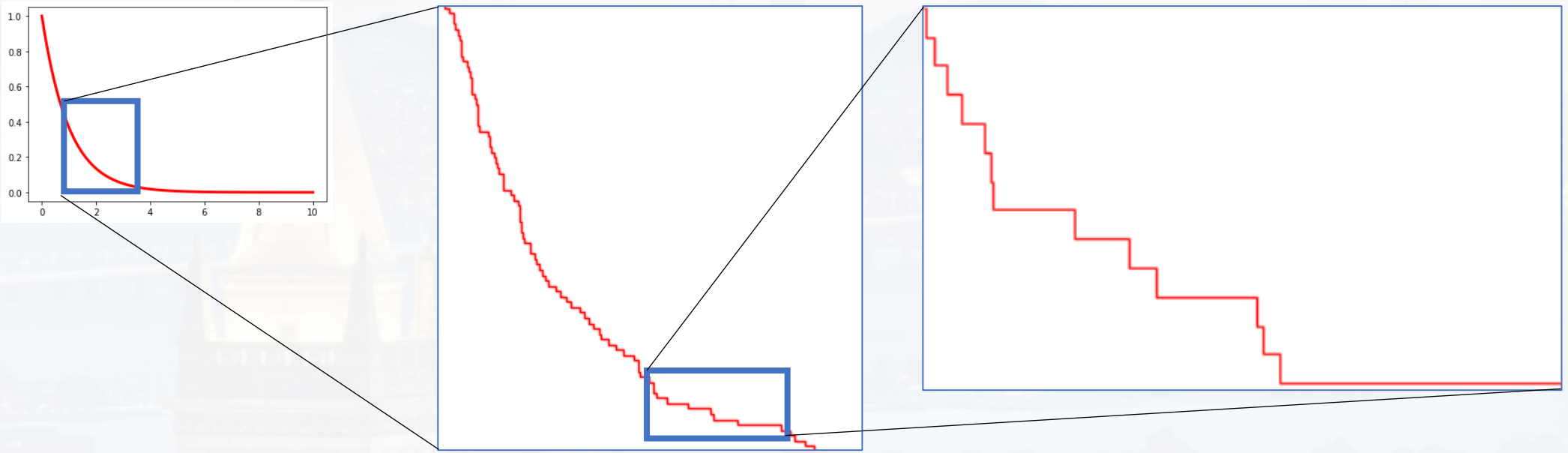
Simulation and Monte Carlo Method:

- many dynamic problems don't have an analytical solution

- some quantities have singularities → standard simulations are not possible

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

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



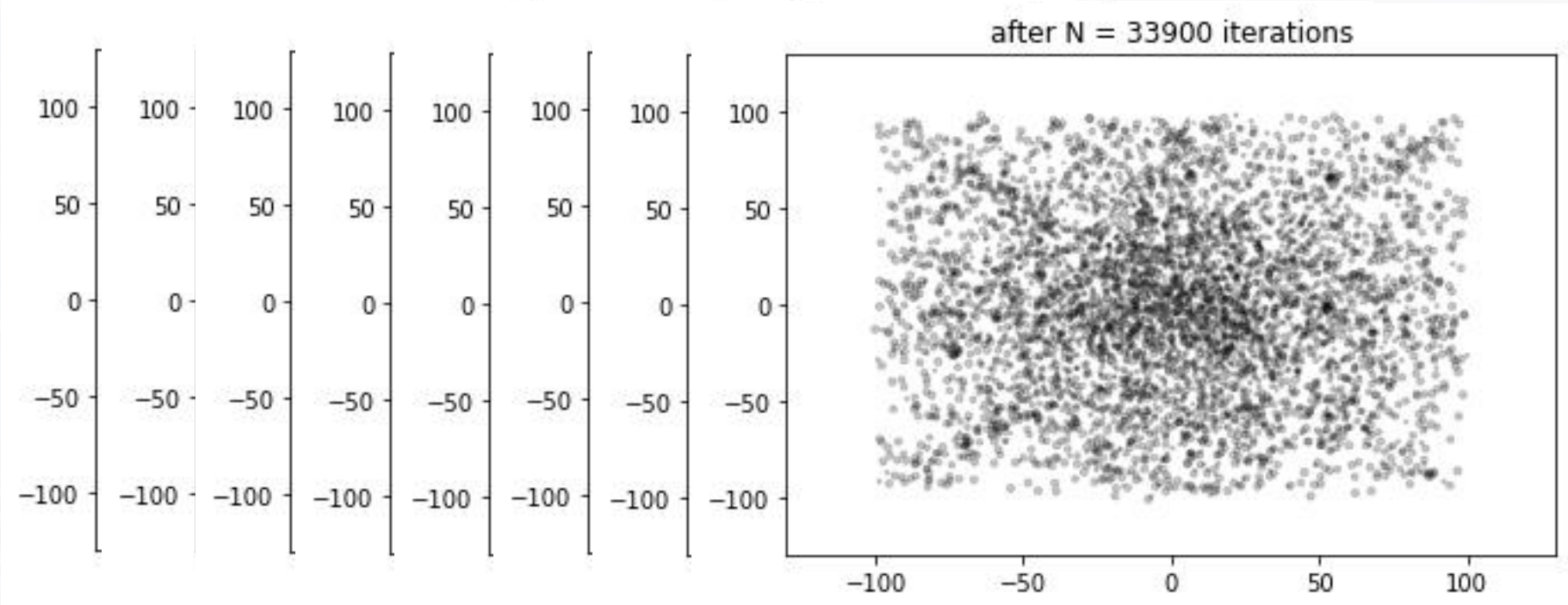
How can we model that?
How do we take stochasticity into account?

Simulation and Monte Carlo Method:

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

moving particles in a potential U

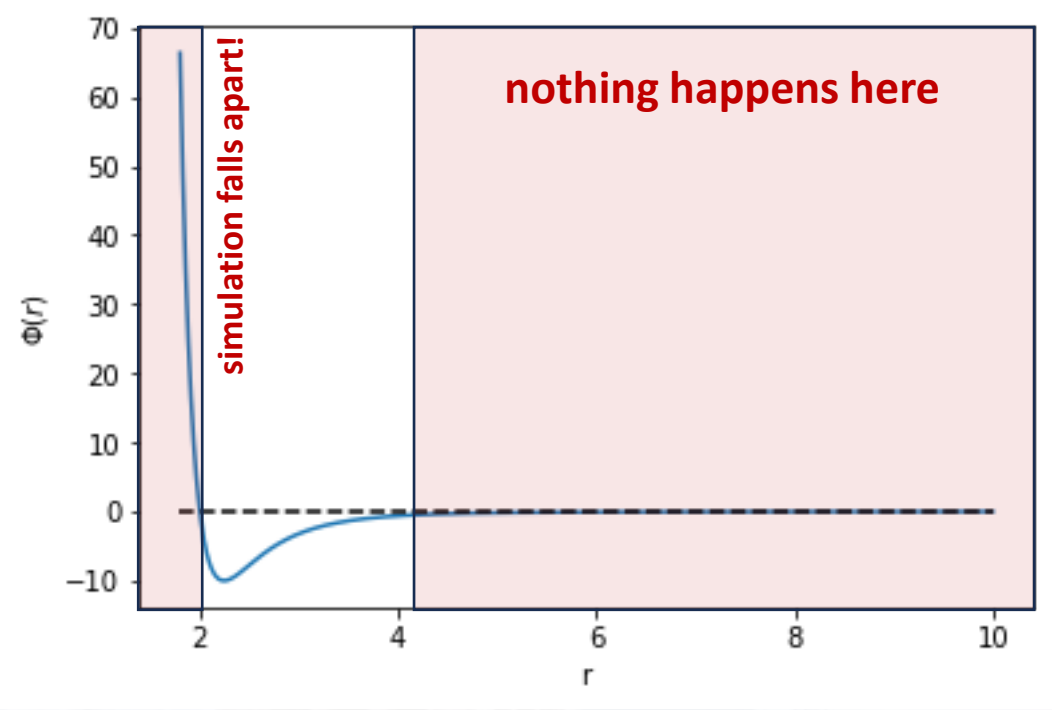
classic approach: solving Newton's equations of motion



Simulation and Monte Carlo Method:

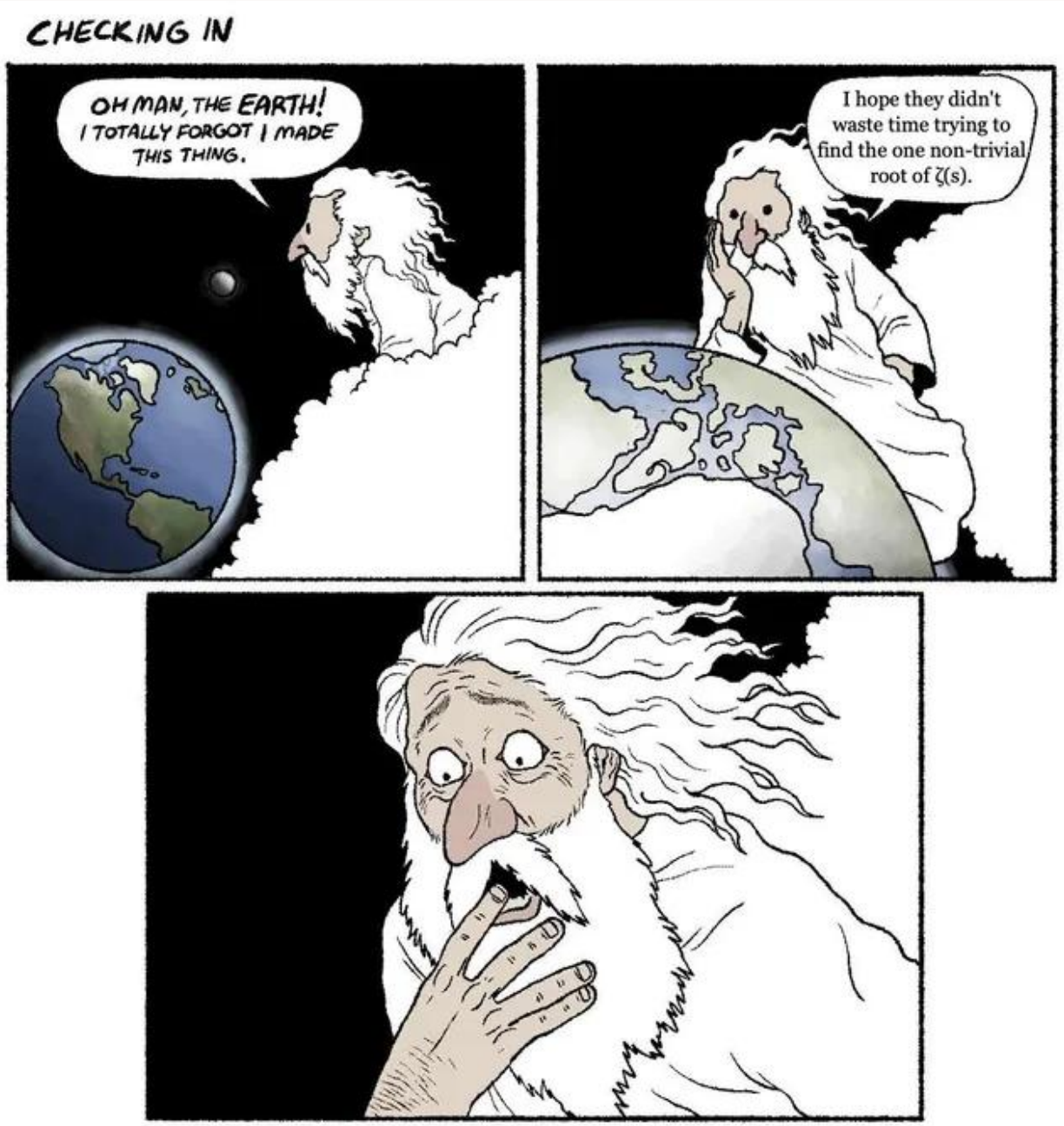
- many dynamic problems don't have an analytical solution
- **some quantities have singularities → standard simulations are not possible**

moving particles in a potential U



Lennard – Jones - Potential

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



Outline

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

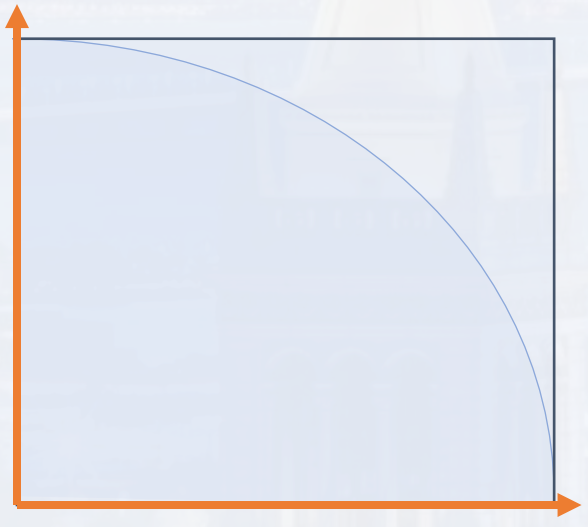
Simulation and Monte Carlo Method:

idea: generating a set of values randomly → **M**onte **C**arlo method
i.e. repeated random sampling

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 π via Monte Carlo

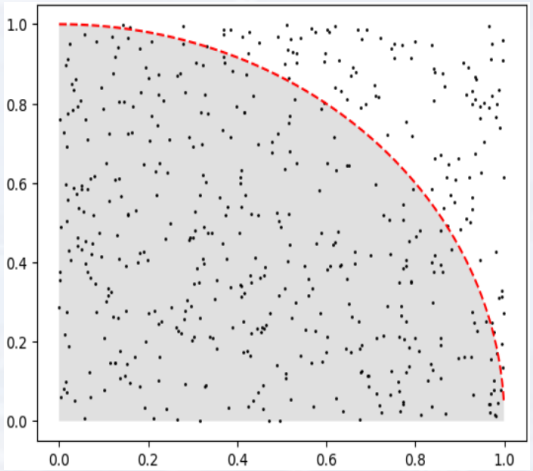


$$A_{square} = 1$$

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

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

picking N_{tot} random values [0,1] x [0,1]

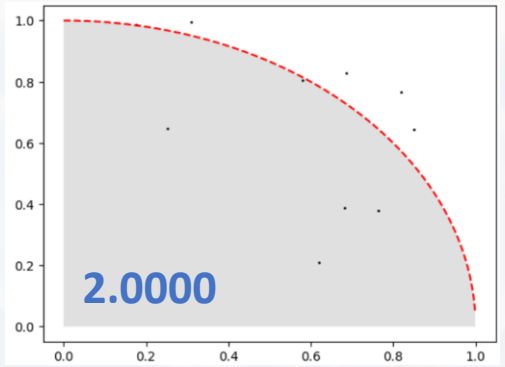


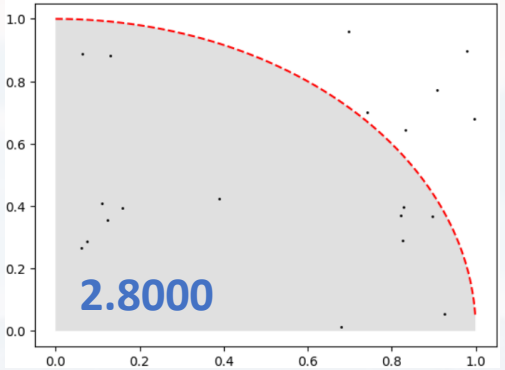
$$\pi \approx 4\frac{N_{section}}{N_{tot}}$$

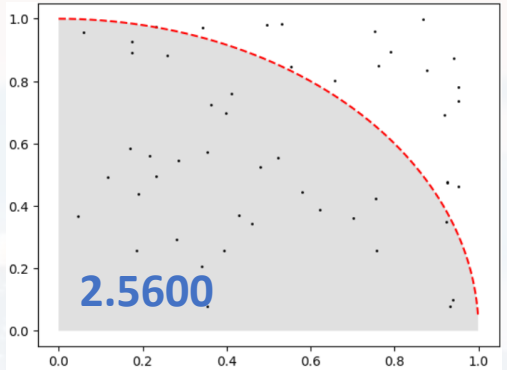
Simulation and Monte Carlo Method:

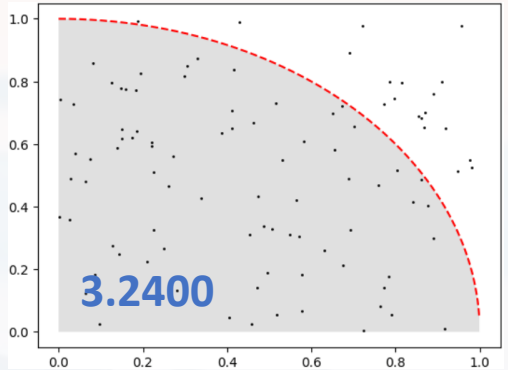
idea: generating a set of values randomly
i.e. repeated random sampling

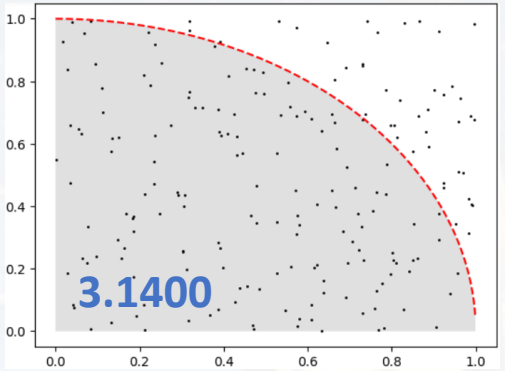
→ **M**onte **C**arlo method

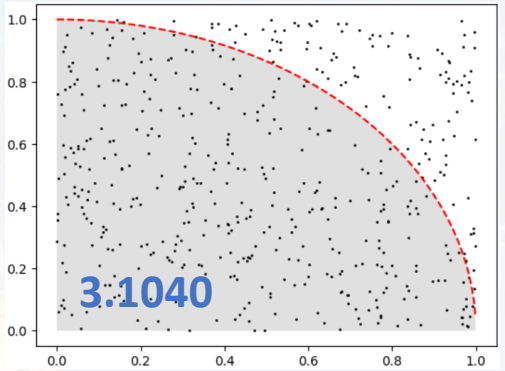


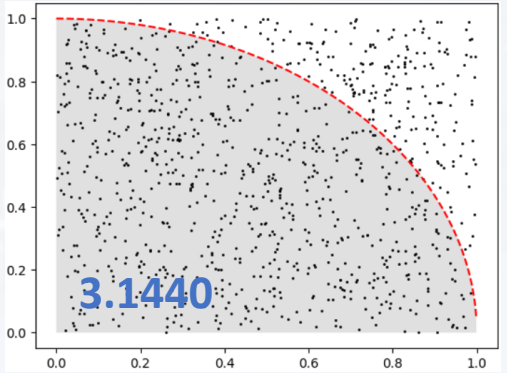


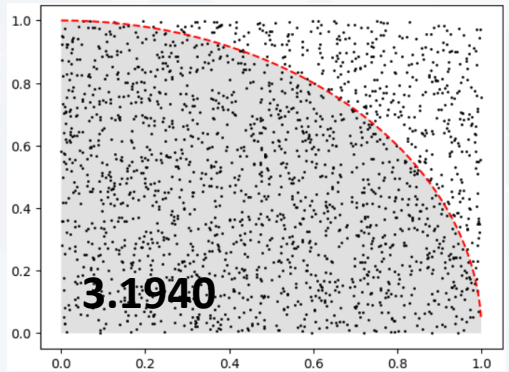


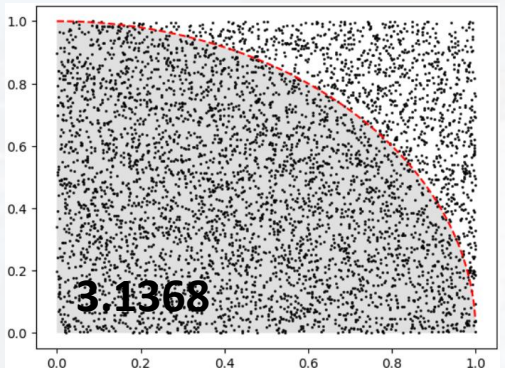


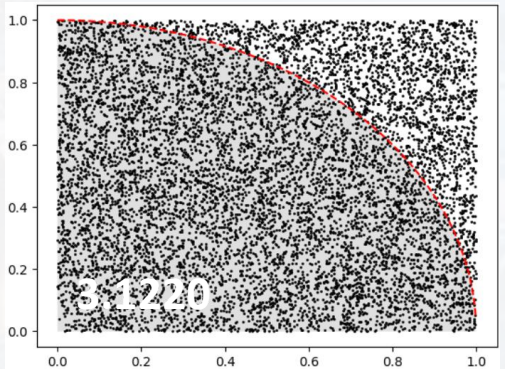


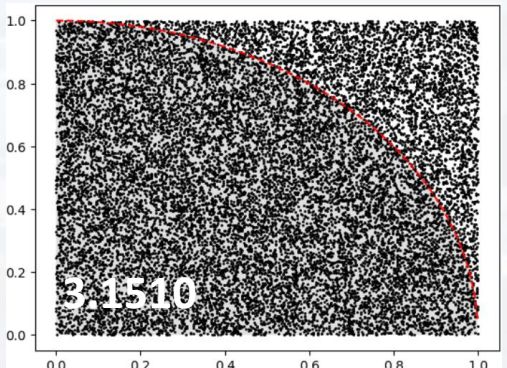










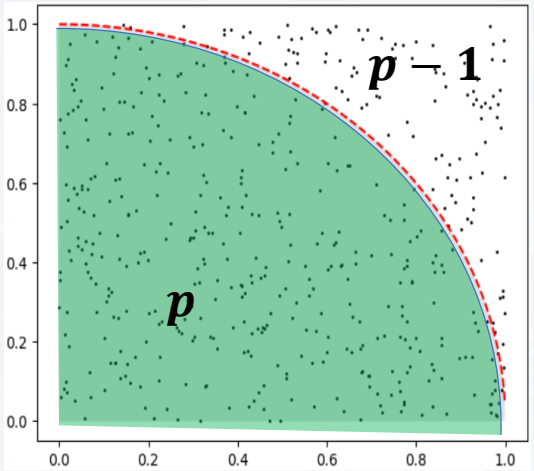


$\pi = 3.14159 \ldots$

Simulation and Monte Carlo Method:

example: finding π via Monte Carlo

How does the accuracy of π 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 $p - 1$

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

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

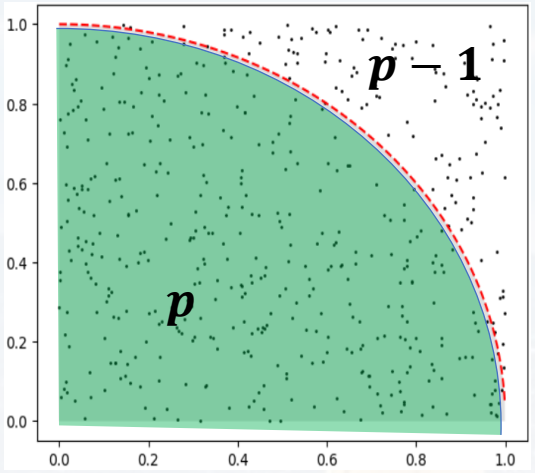
$$\mu(k) = pN_{tot}$$

(see module 7)

Simulation and Monte Carlo Method:

example: finding π via Monte Carlo

How does the accuracy of π depend on N_{tot}?

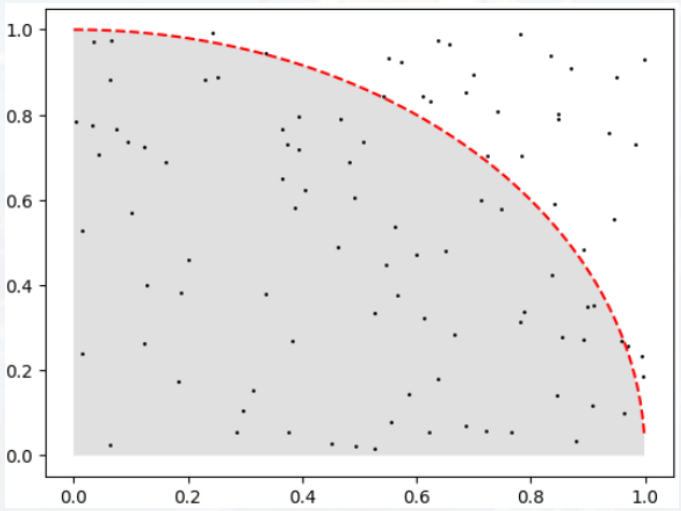


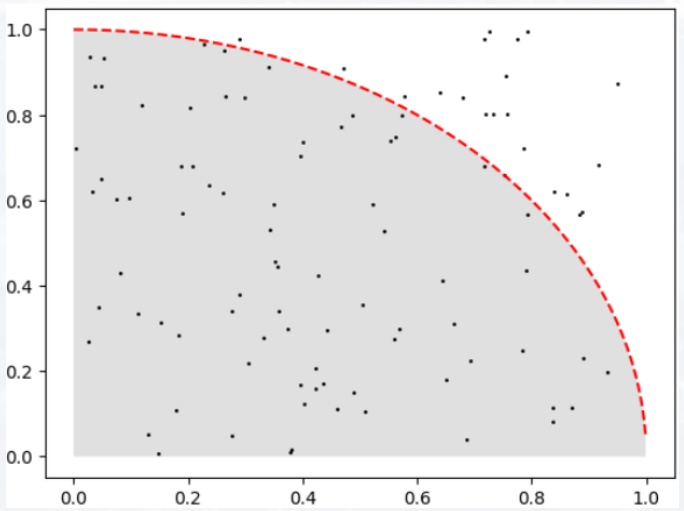
$$\pi \approx 4\frac{N_{section}}{N_{tot}} \qquad \sigma(k)^2 = N_{tot}\, p(1-p) \qquad \mu(k) = pN_{tot}$$ **(see module 7)**

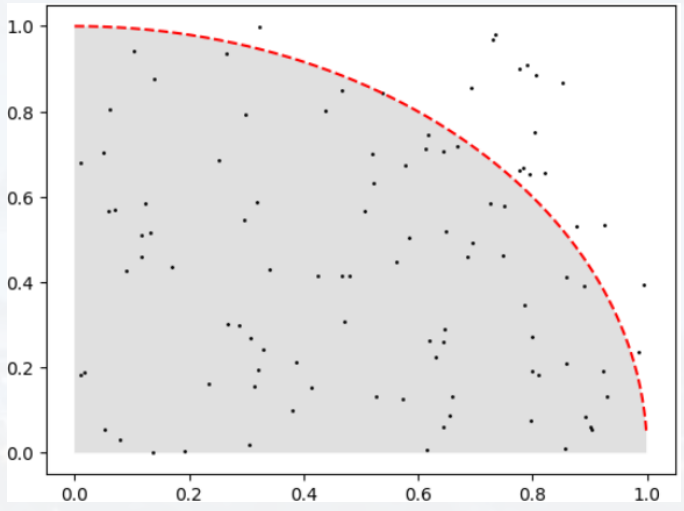
error of π: $4\sigma\left(\frac{N_{section}}{N_{tot}}\right) = 4\sigma\left(\frac{k}{N_{tot}}\right)$ standard deviation σ of the ratio $\frac{N_{section}}{N_{tot}}$

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

$N_{tot} = 100$



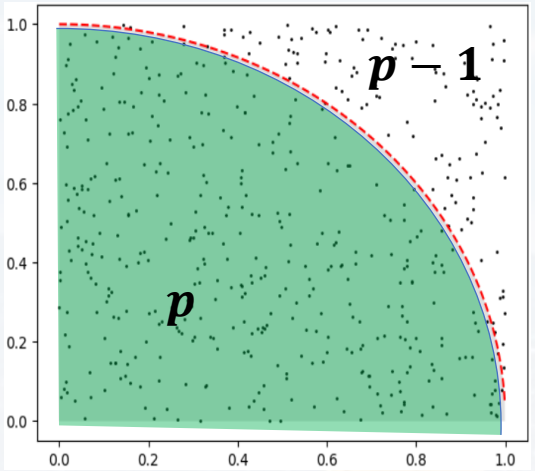




Simulation and Monte Carlo Method:

example: finding π via Monte Carlo

How does the accuracy of π depend on N_{tot}?



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

$$\boxed{\sigma(k)^2 = N_{tot}\, p(1-p)} \qquad \mu(k) = pN_{tot}$$ **(see module 7)**

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

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

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

$$= 4\sqrt{\frac{1}{{N_{tot}}^2}N_{tot}\, p(1-p)}$$

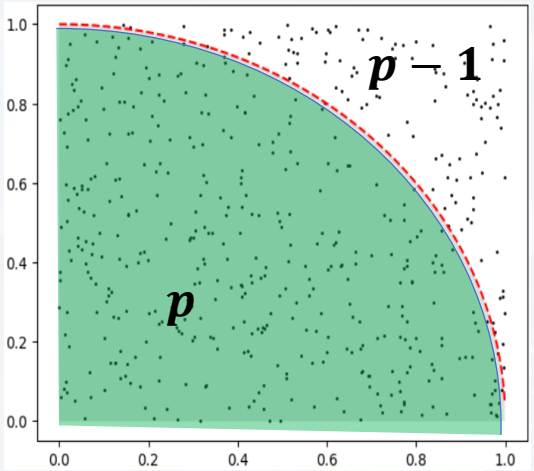
We know: $Var(x) = \langle x^2\rangle - \langle x\rangle^2$ **(see module 7)**

a = const $\quad x \to ax \quad Var(ax) = \langle (ax)^2\rangle - \langle ax\rangle^2 = a^2(\langle x^2\rangle - \langle x\rangle^2) = a^2 Var(x)$

Simulation and Monte Carlo Method:

example: finding π via Monte Carlo

How does the accuracy of π depend on N_{tot}?



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

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

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

because we wanted find π in the first place

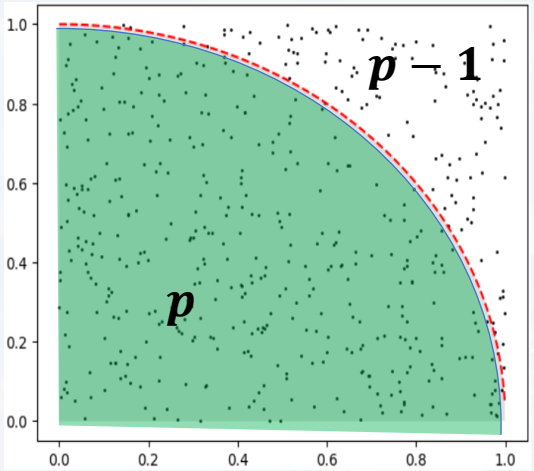
but we can ***estimate*** it during the simulation

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

Simulation and Monte Carlo Method:

example: finding π via Monte Carlo

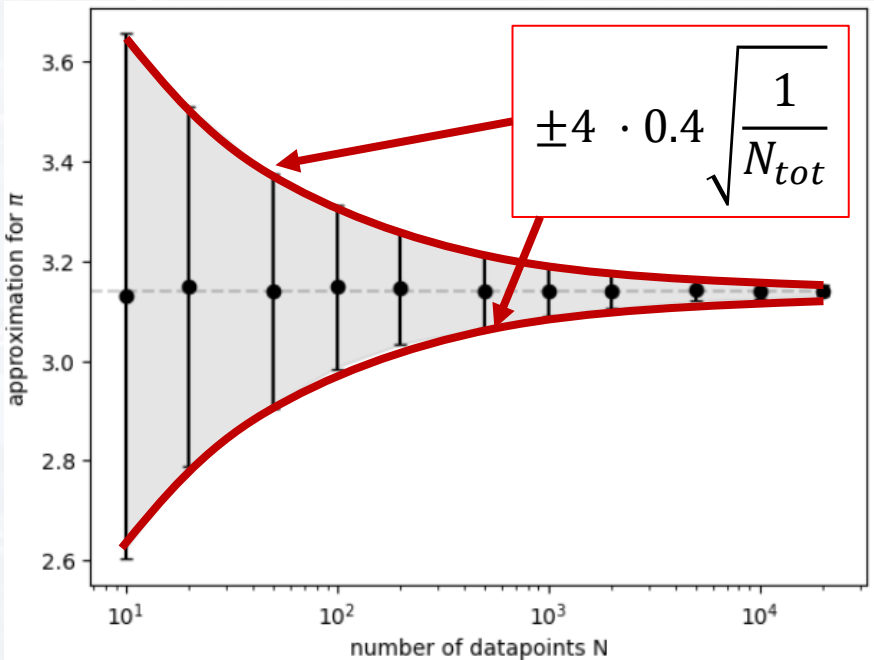
How does the accuracy of π depend on N_{tot}?

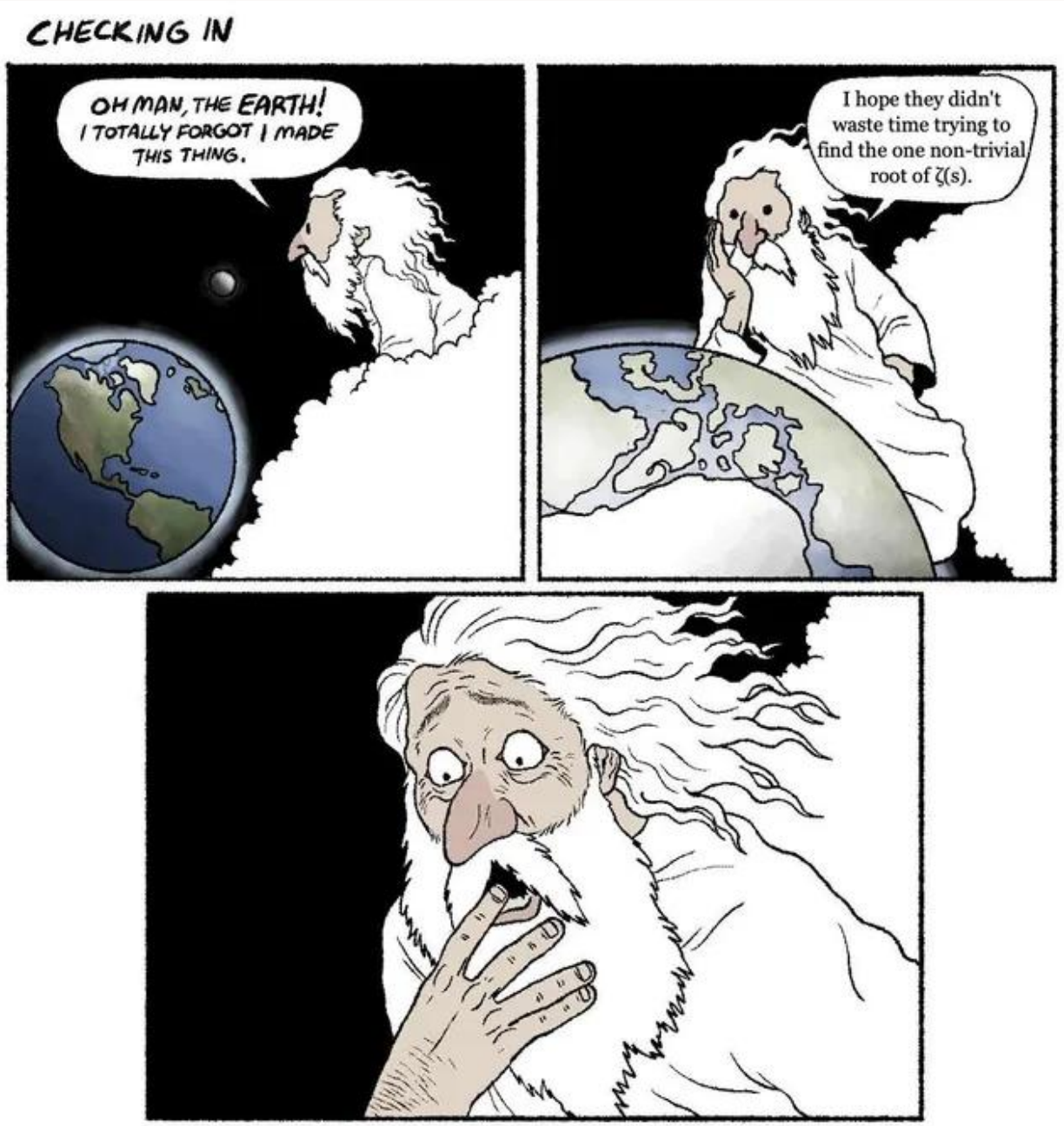


$$\pi \approx 4\frac{N_{section}}{N_{tot}}$$

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

running 100 simulations **for each** N_{tot}
→ calculating standard deviation of π
→ comparing to $4 \cdot 0.4\sqrt{\frac{1}{N_{tot}}}$





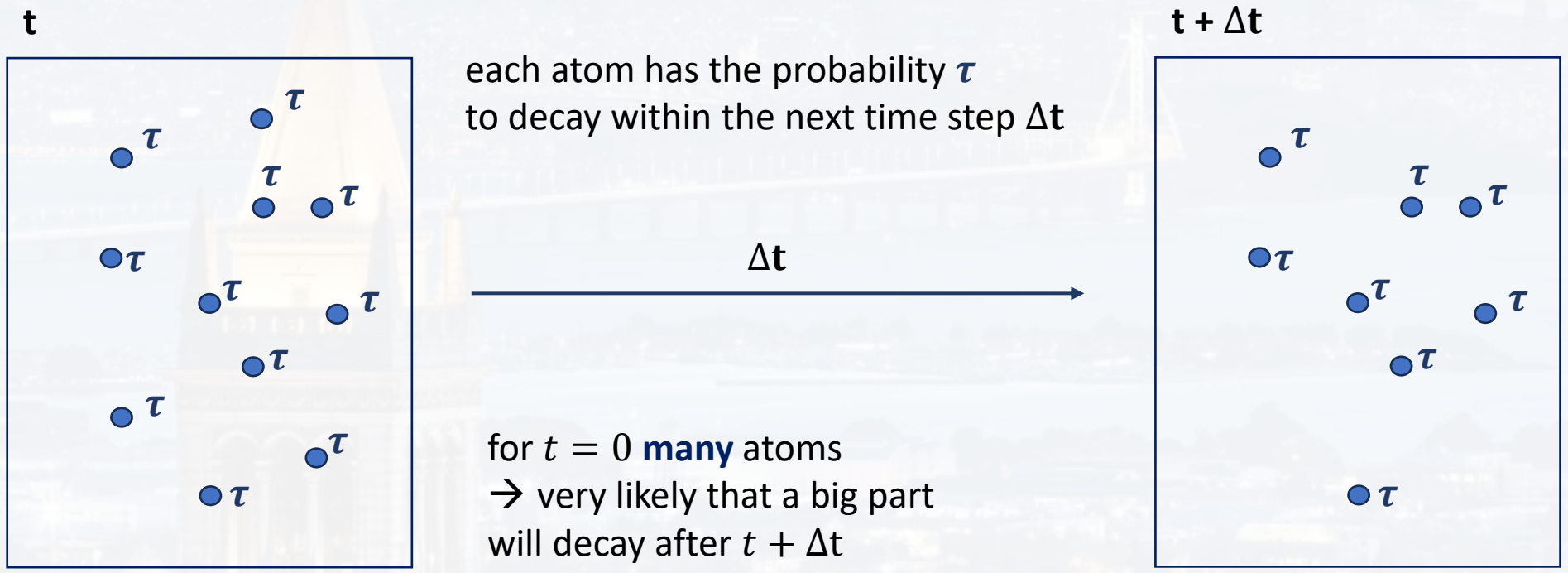
Outline

- 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**

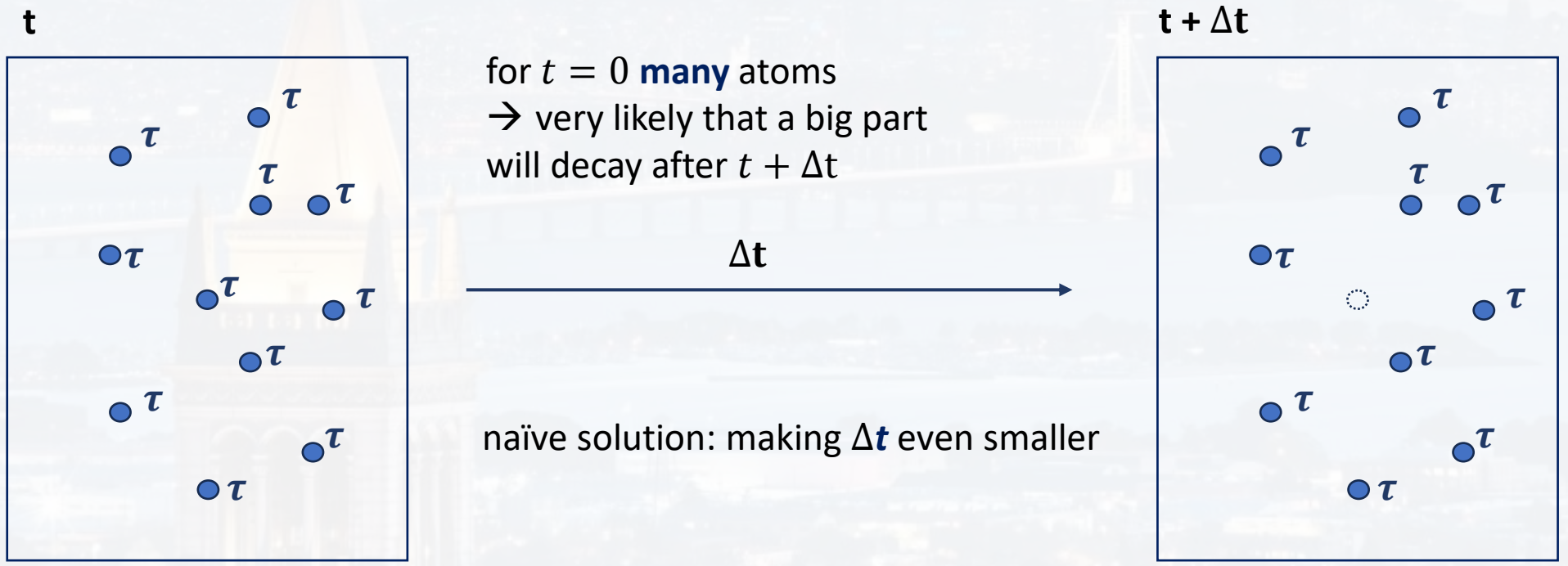


We can combine the MC method with different ways ***how*** to run a simulation

example: - radioactive decay of atoms

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

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



Simulation and Monte Carlo Method:

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**

t

naïve solution: making **Δt** even smaller

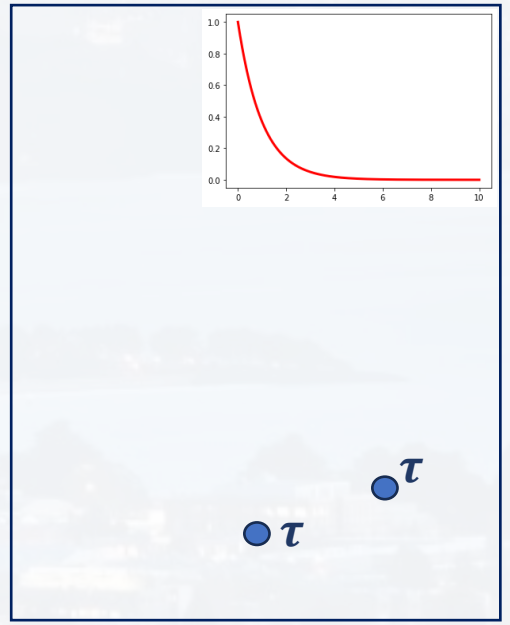
Δt

problem: towards the end, only a **few atoms are left**

→ we need to **wait longer** until one atom decays

→ for many **Δt**: nothing happens

t + Δt

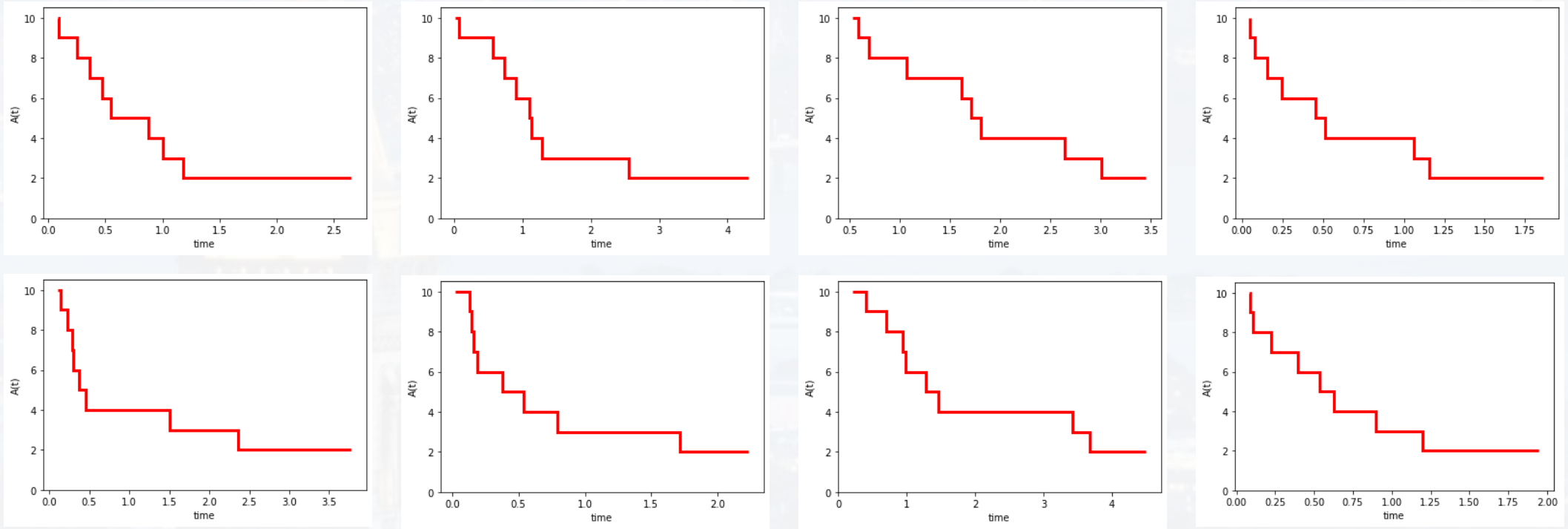


Simulation and Monte Carlo Method:

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 $\Delta \mathbf{t}$ where nothing happens → 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 $\Delta\mathbf{t}$
- 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 $\boldsymbol{\tau}$, **calculate** the time $\Delta\boldsymbol{t}$ that elapses **until** one atom decays
- $N(t + \Delta t) = N(t) - 1$
- $t \rightarrow t + \Delta t$ and repeat

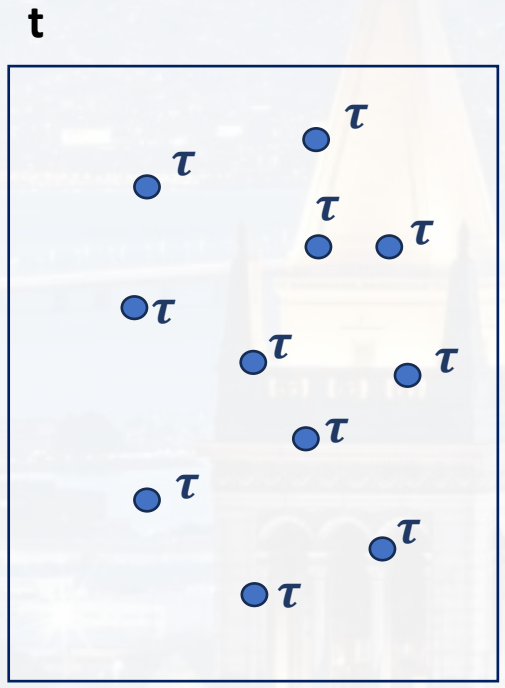
- Δt adapts automatically to $N(t)$

Gillespie:

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

-Δt 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: Δt ***between*** two decays

→ no decay within Δt, i. e. $n = 0$

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

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

Simulation and Monte Carlo Method:

Gillespie:

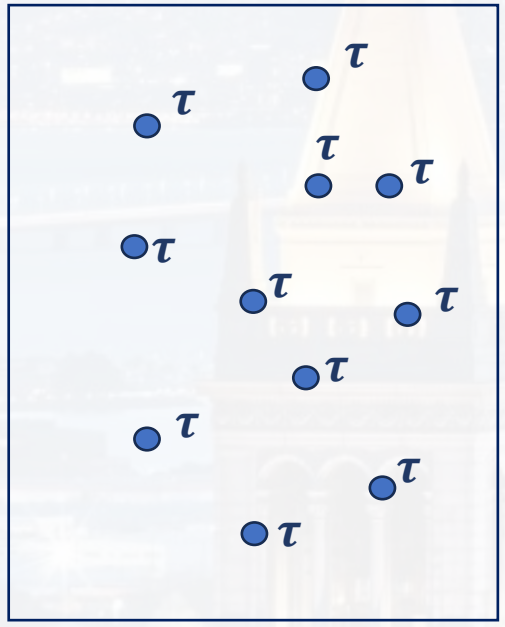
- based on $N(t)$ and τ, **calculate** the time Δt that elapses **until** one atom decays
- $N(t + \Delta t) = N(t) - 1$
- $t \rightarrow t + \Delta t$ and repeat

-Δt adapts automatically to $N(t)$

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

Poissonian Stepper

t



each atom has the probability of decay per time τ

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

logical ***or*** → **adding** the probabilities **(see module 6)**

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

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

Gillespie:

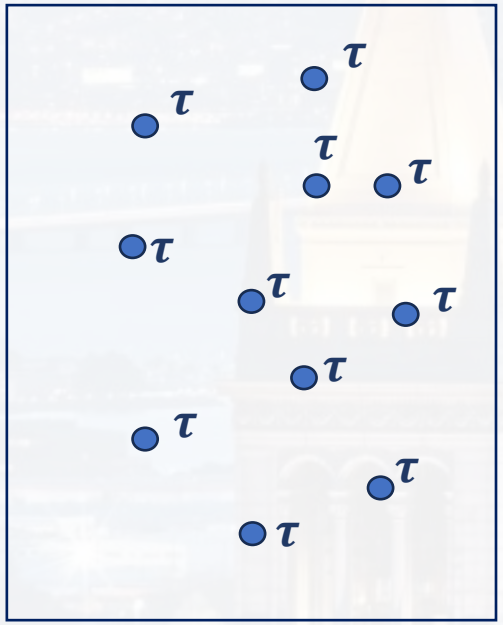
1) draw a **random number** ρ from a **uniform distribution** in the interval $(\mathbf{0}, \mathbf{1})$

2) calculate the time Δt 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



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

Simulation and Monte Carlo Method:

Gillespie:

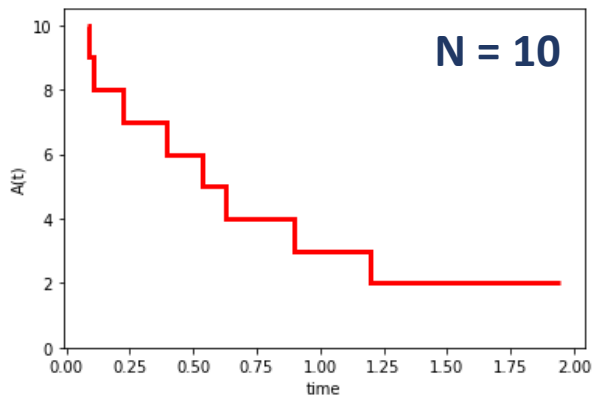
1) draw a **random number** ρ from a **uniform distribution** in the interval $(\mathbf{0}, \mathbf{1})$

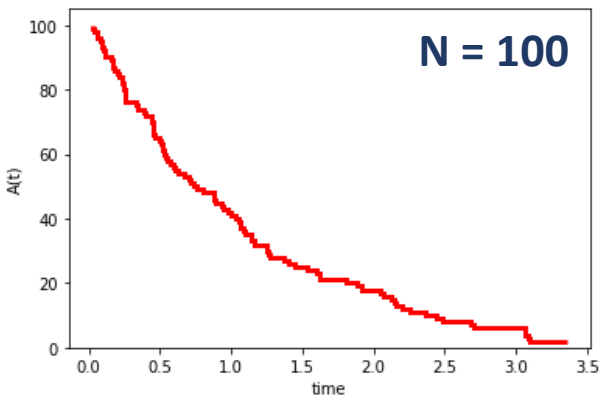
2) calculate the time Δt that elapses until the next decay

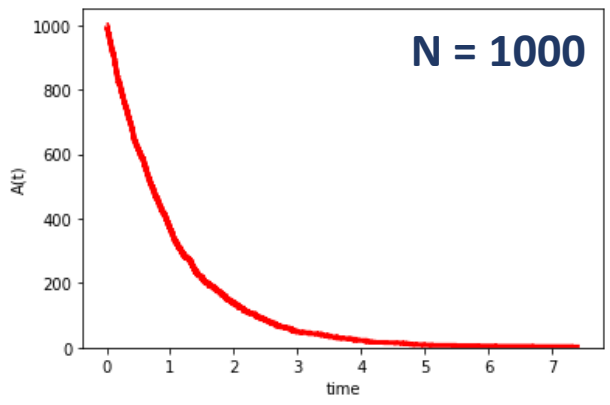
$$\Delta t = -\frac{1}{\tau\, N(t)} \ln \rho$$

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

4) repeat







example:

$$A \underset{k_-}{\overset{k_+}{\rightleftarrows}} B$$

need to calculate Δt for ***a*** reaction (either A → B **or** B → A)

logical ***or*** → **adding** the probabilities **(see module 6)**

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

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

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

next: deciding ***which*** reaction should occur

example:

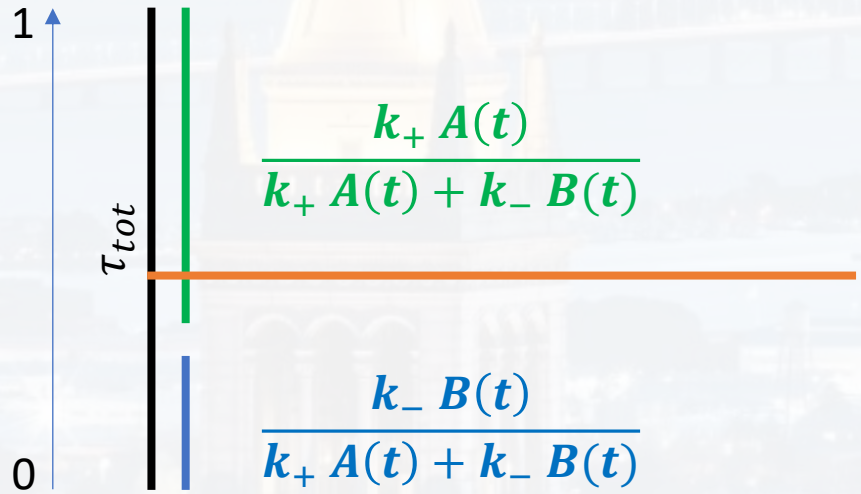
$$A \underset{k_-}{\overset{k_+}{\rightleftarrows}} B$$

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

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

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

next: deciding ***which*** reaction should occur



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

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

Gillespie:

1) draw a **random number** ρ_1 from a **uniform distribution** in the interval $(\mathbf{0}, \mathbf{1})$

2) calculate the time Δt 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** ρ_2 from a **uniform distribution** in the interval $(\mathbf{0}, \mathbf{1})$

4) decide which reaction occurs:

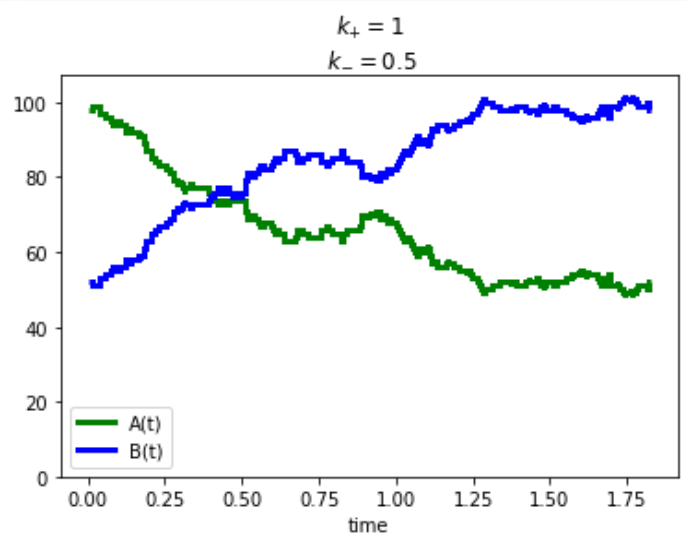
if $\rho_2 < \frac{k_+ A(t)}{k_+ A(t) + k_- B(t)}$: reaction A → B is more likely

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

else: reaction B → A is more likely

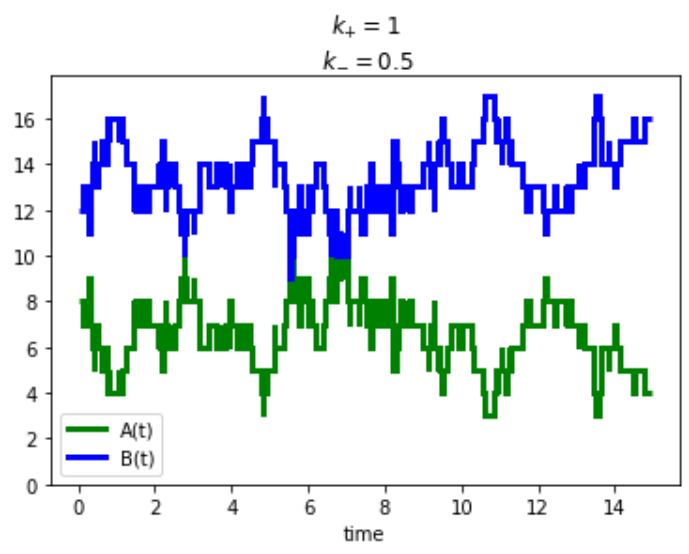
$$A(t + \Delta t) = A(t) + 1$$
$$B(t + \Delta t) = B(t) - 1$$

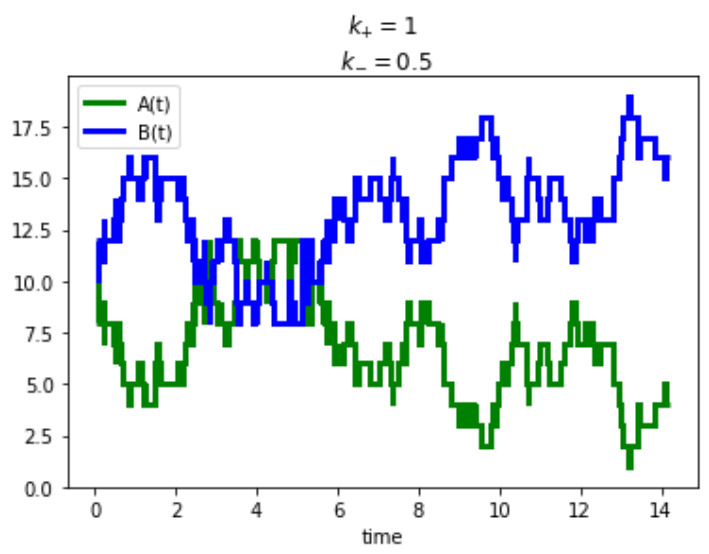
Simulation and Monte Carlo Method:

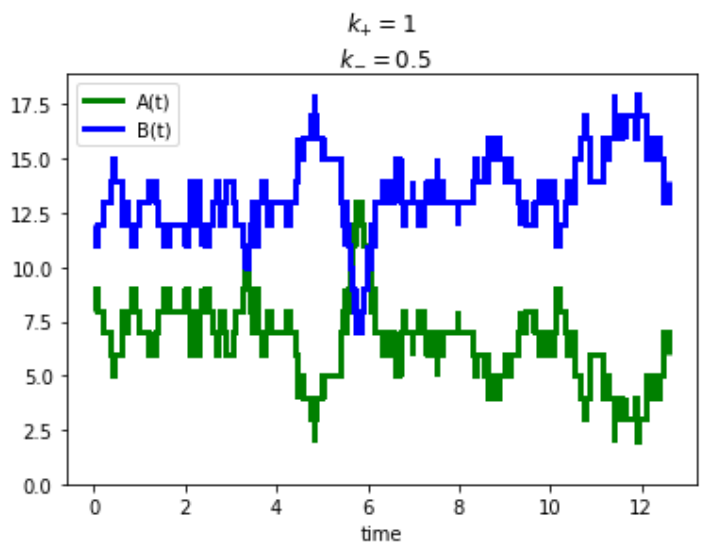


equilibrium at $\frac{A}{B} = \frac{k_-}{k+}$

N = 10

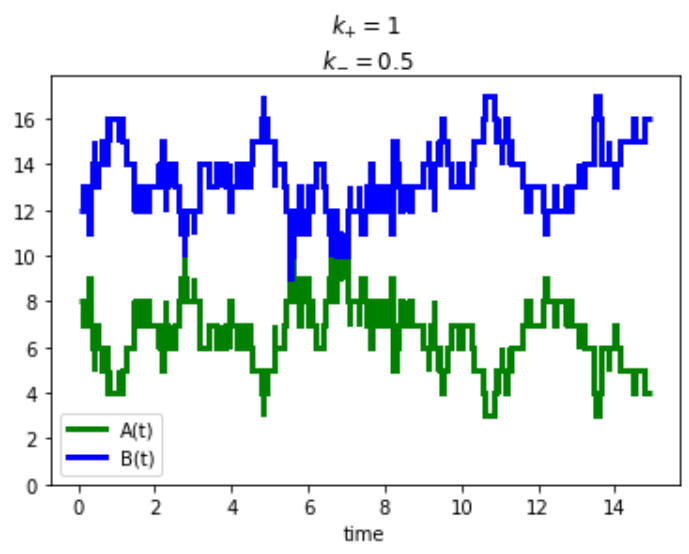


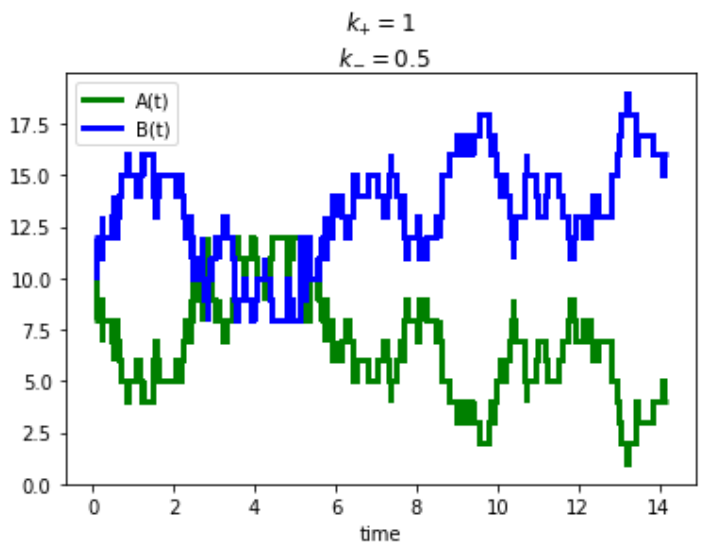


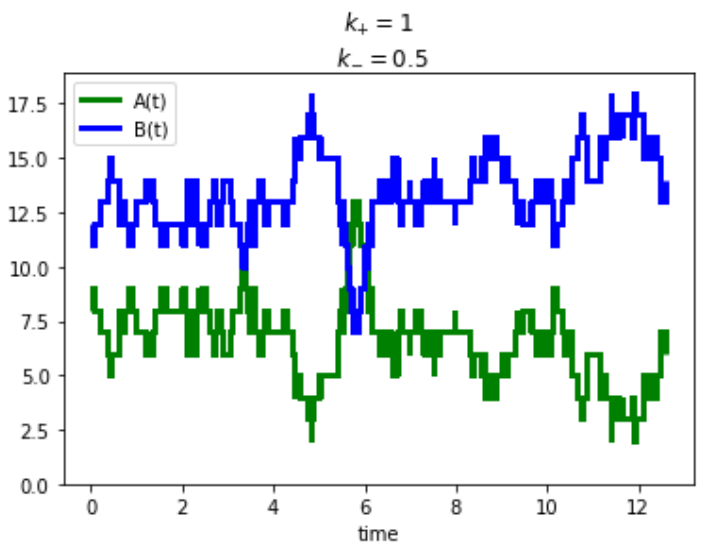


Simulation and Monte Carlo Method:

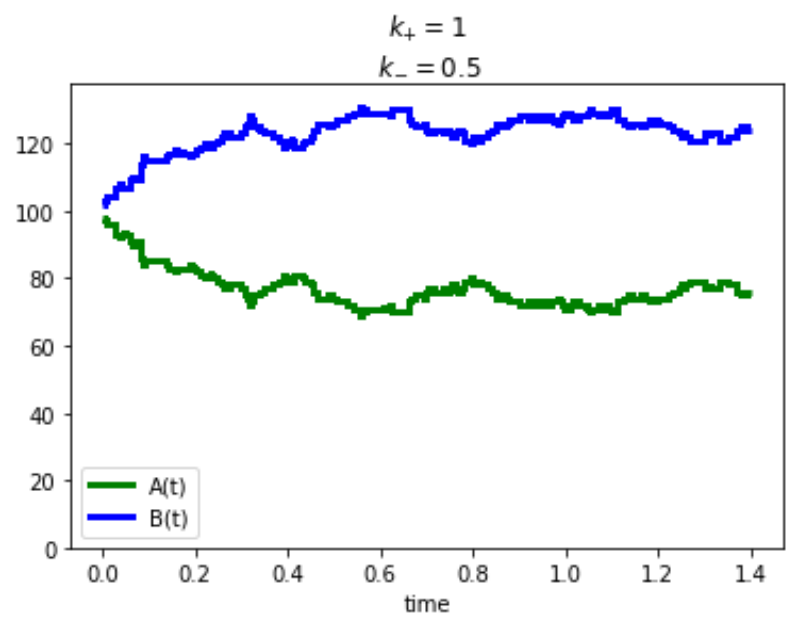
N = 10



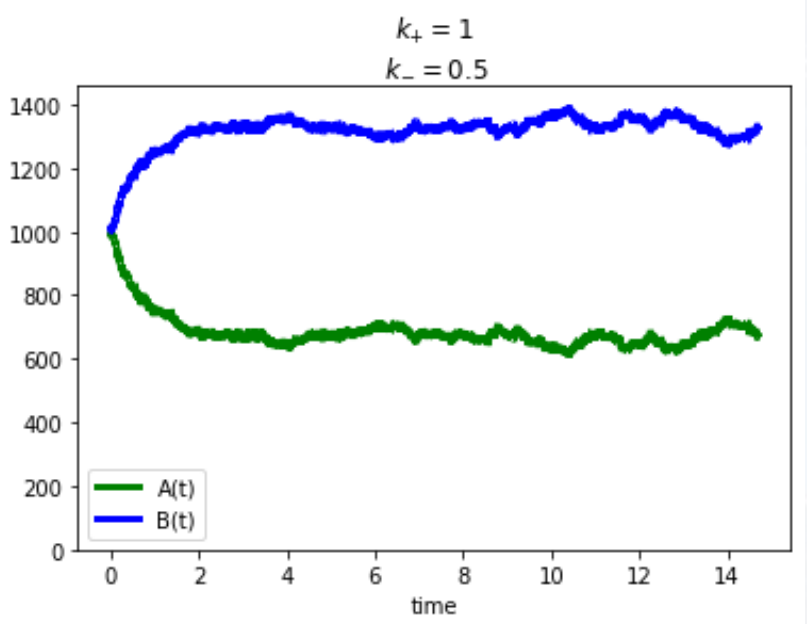




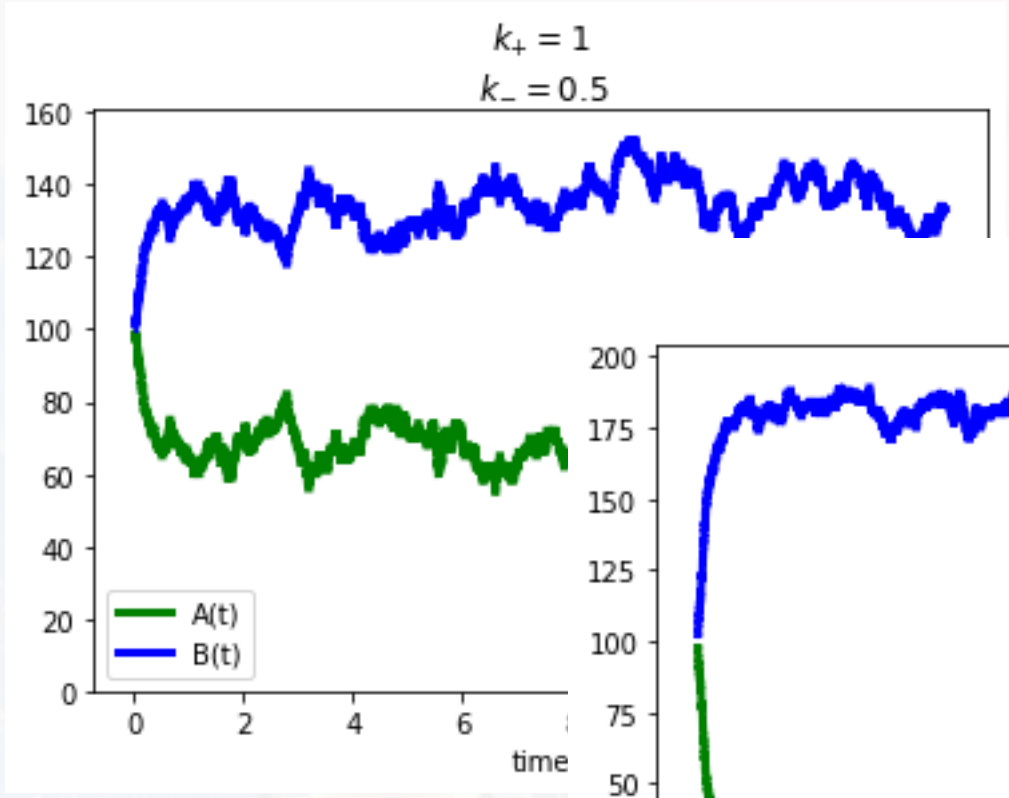
N = 100

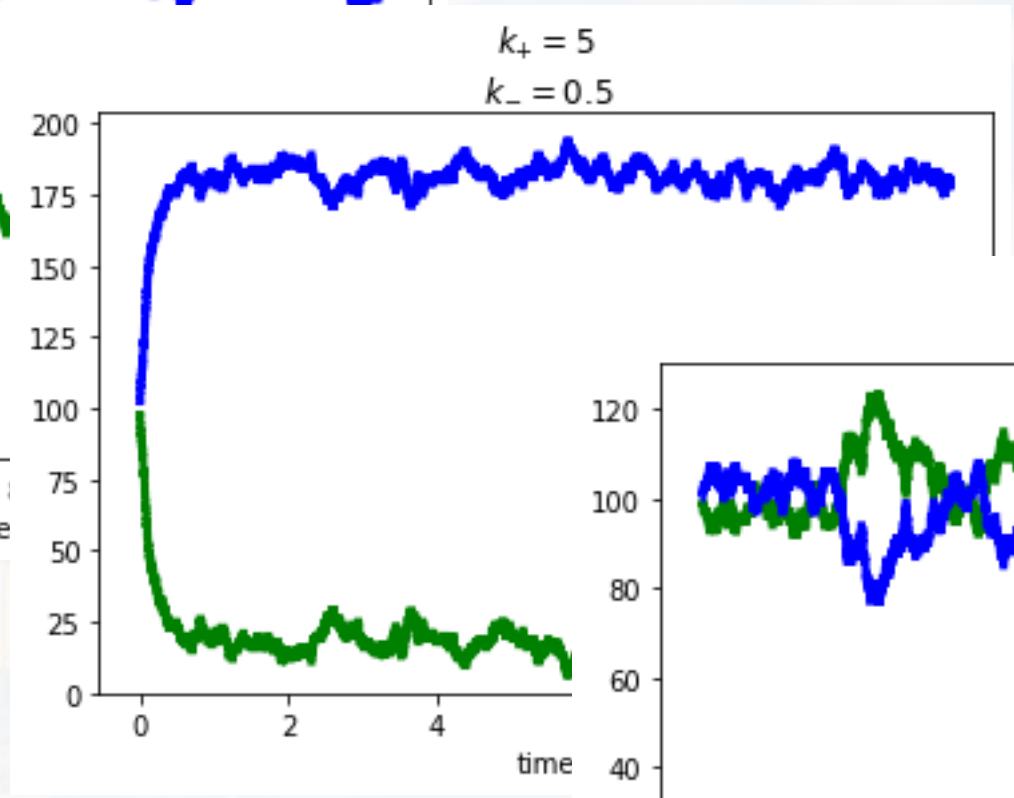


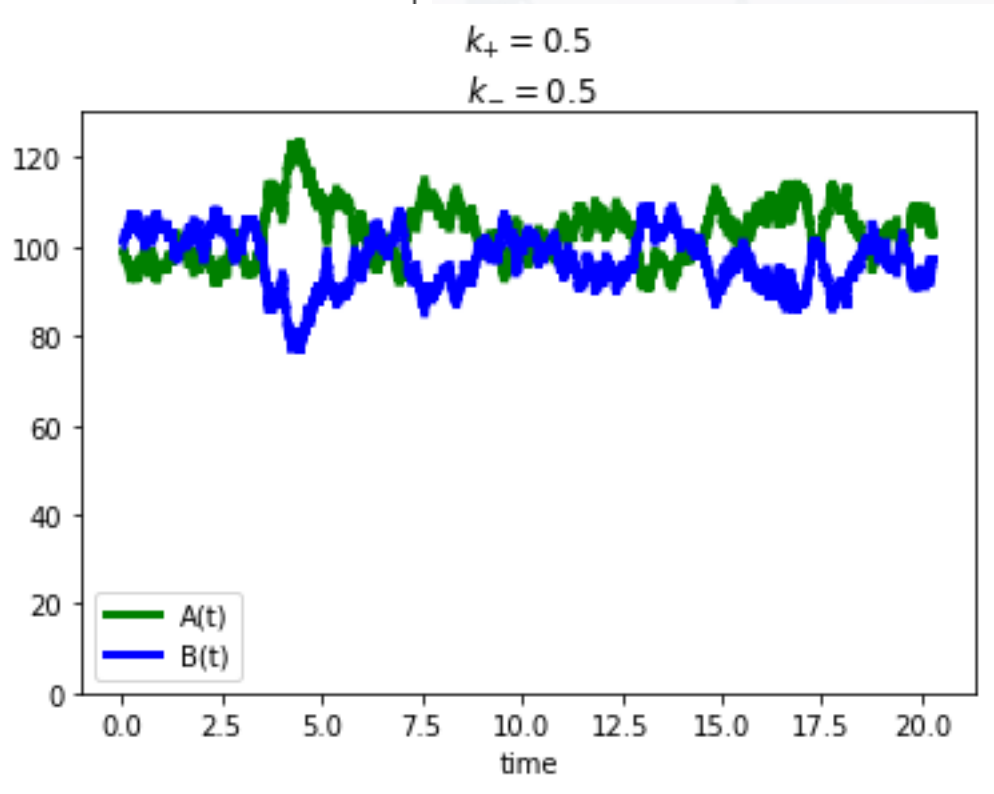
N = 1000

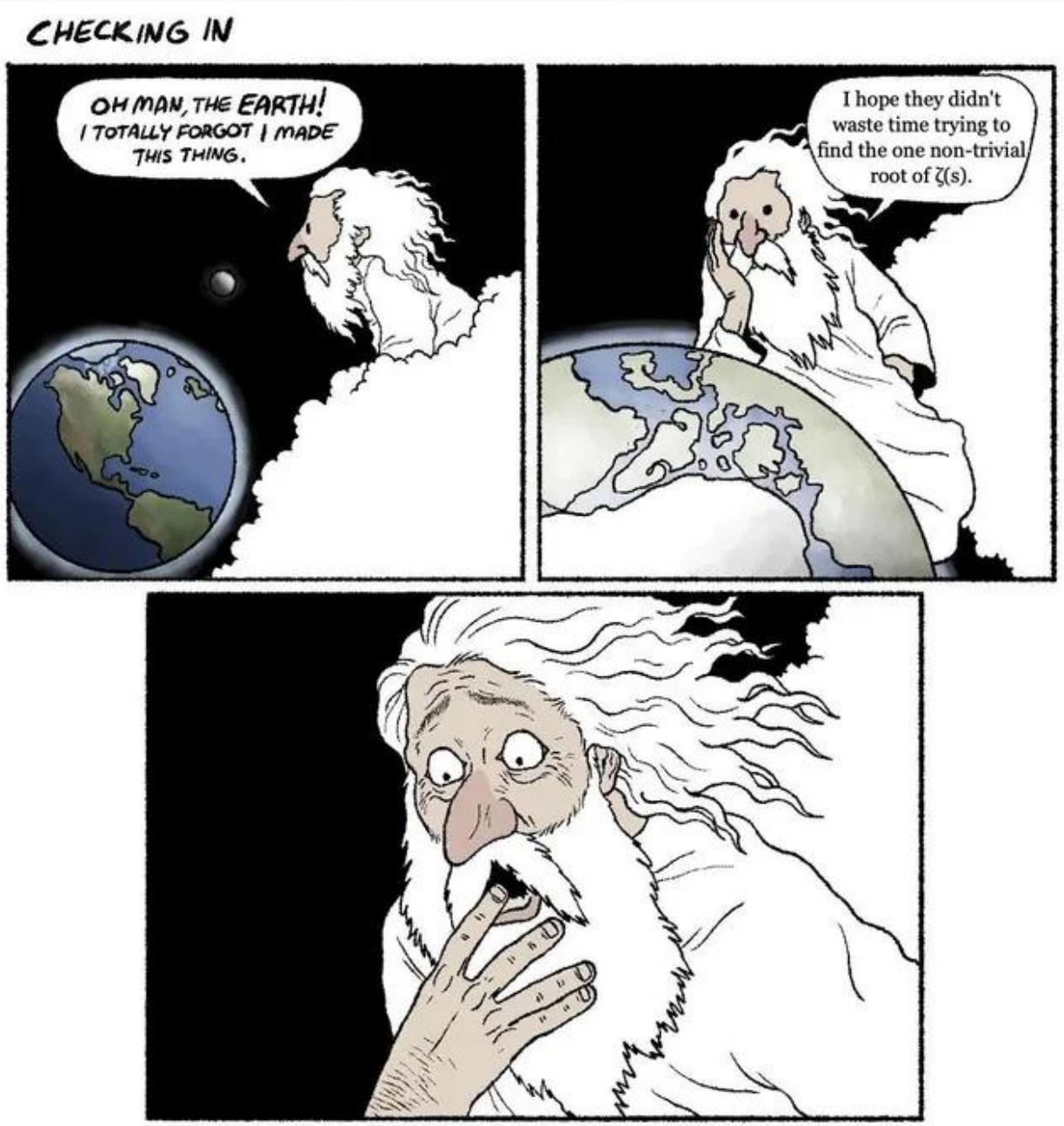


Simulation and Monte Carlo Method:







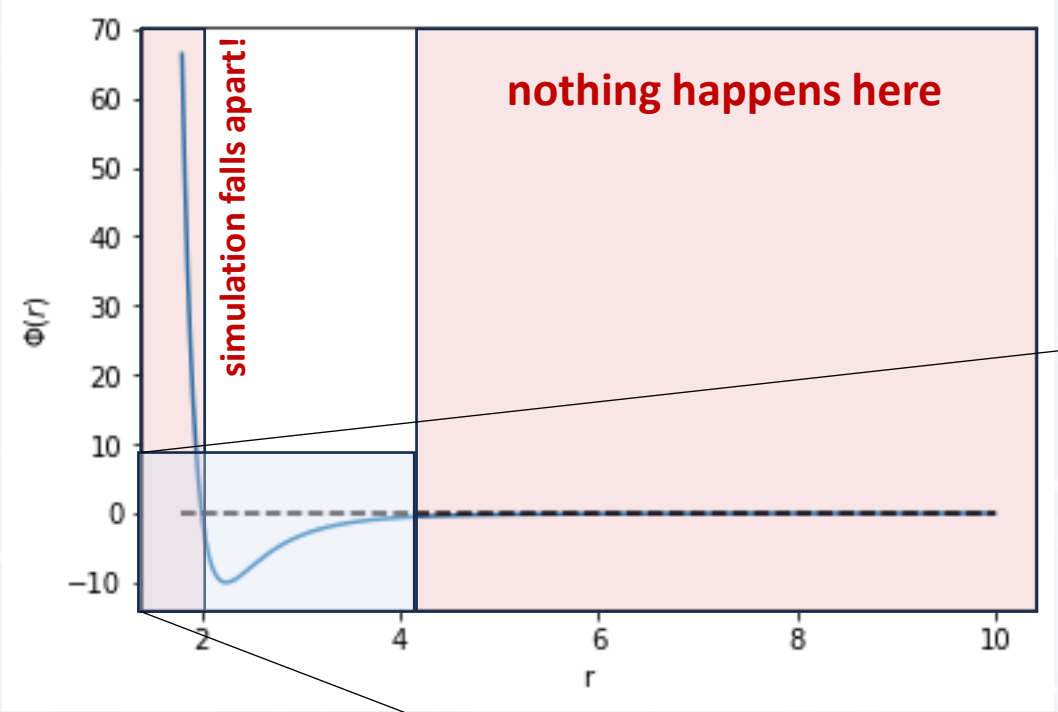


Outline

- 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



$$U_{LJ} = \frac{a}{r^{12}} - \frac{b}{r^{6}}$$

a, b = const

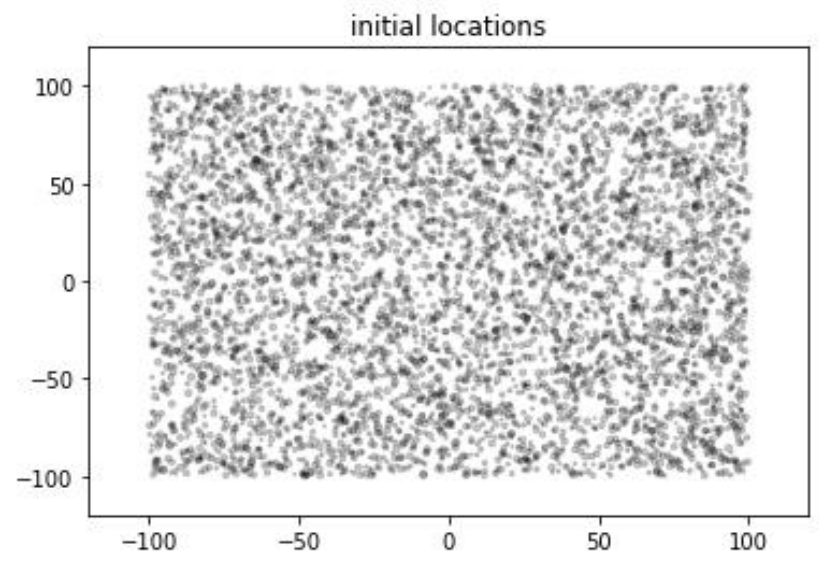
$$r_{min} = \left(\frac{2a}{b}\right)^{1/6}$$

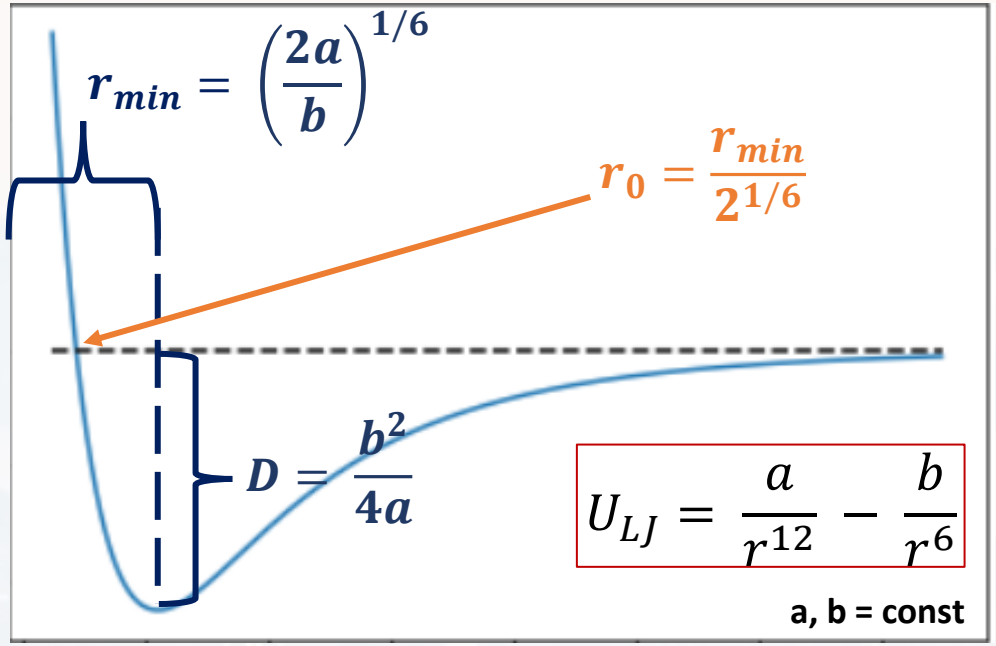
$$r_0 = \frac{r_{min}}{2^{1/6}}$$

$$D = \frac{b^2}{4a}$$

Simulation and Monte Carlo Method:

the problem:





simulating many particles

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

total force/potential that acts on the particle

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

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

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

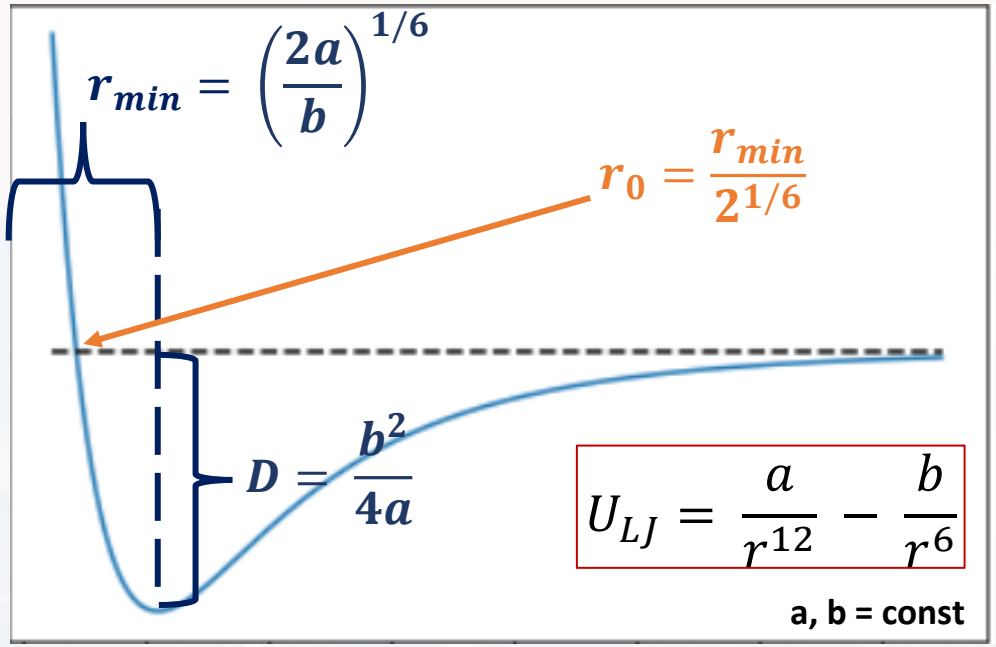
$$a(y) = \frac{F(y)_{tot}}{m} = \frac{1}{m}\frac{\partial U_{tot}(x,y)_{LJ}}{\partial y}$$

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 Δt 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 → Δt **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 → Δt **too small**

Simulation and Monte Carlo Method:

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**

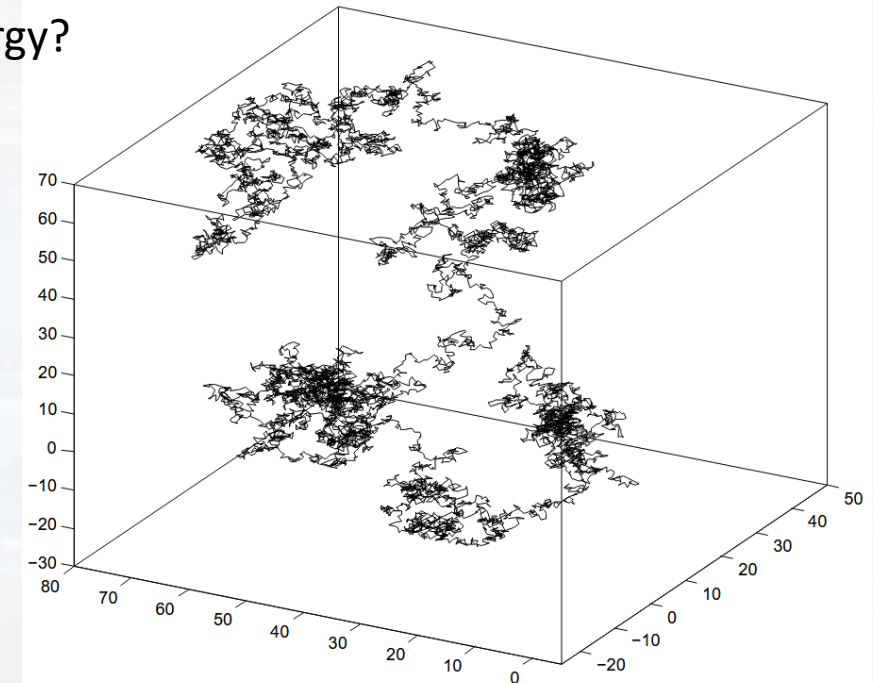
Why should a particle move, even though that increases its energy?

reason: random **Brownian motion**

one can show **(statistical physics, max entropy)**

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

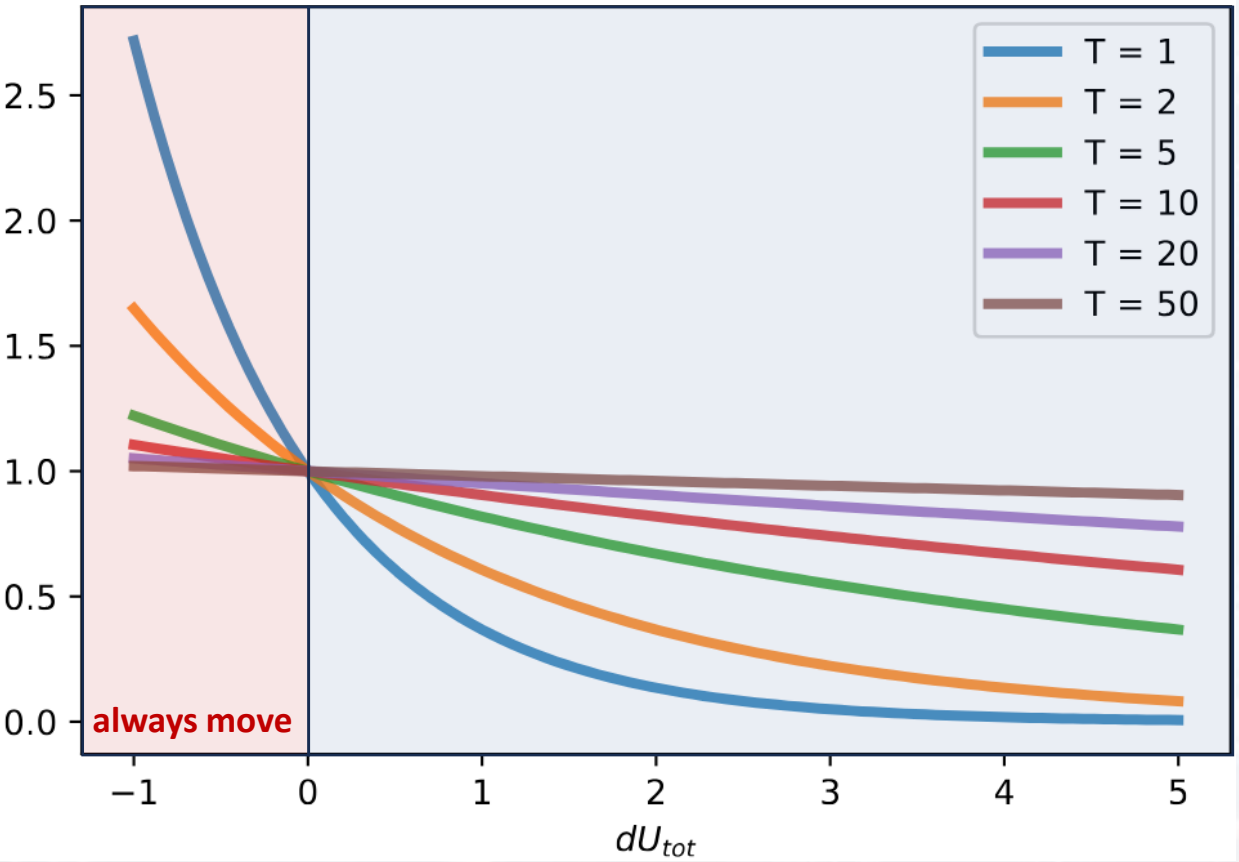
T: temperature



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

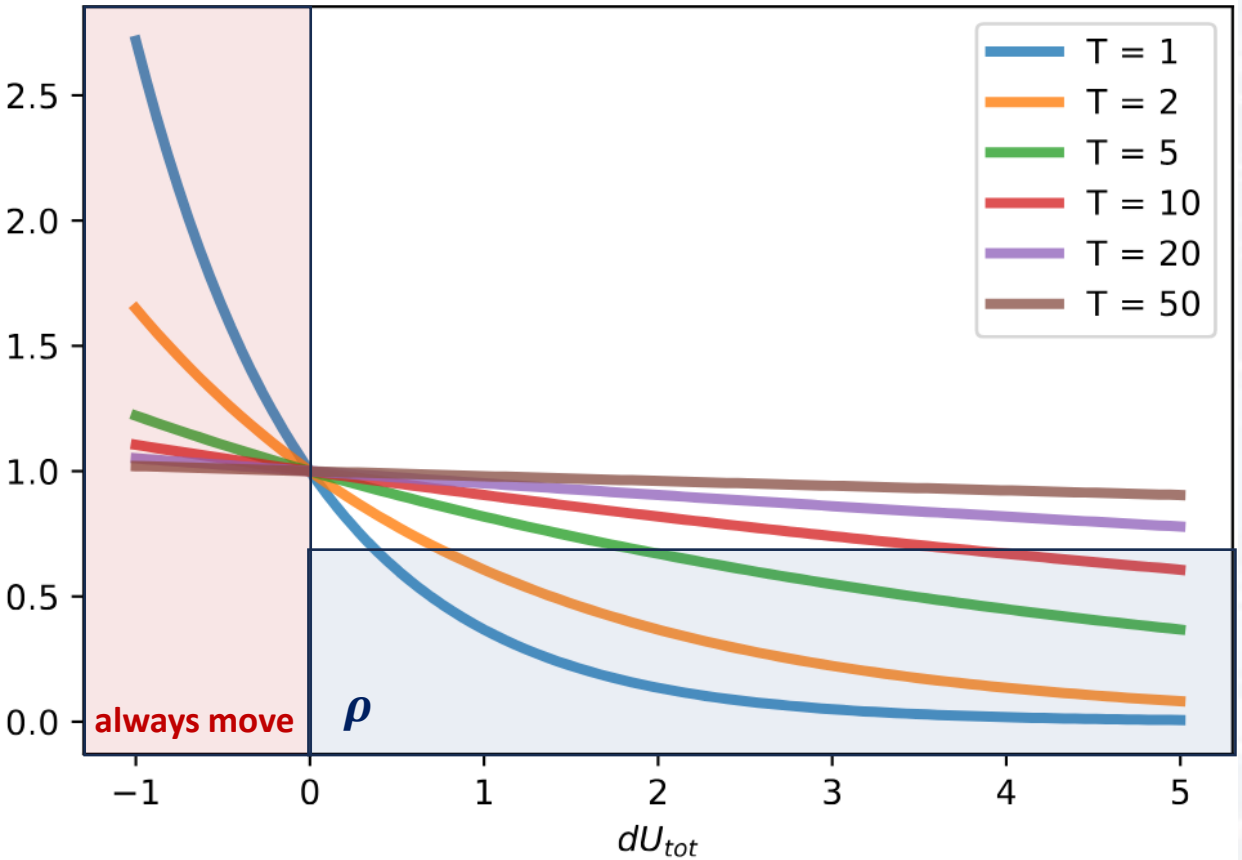
draw a **random number** ρ from a **uniform distribution** in the interval $(\mathbf{0}, \mathbf{1})$

Simulation and Monte Carlo Method:

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** ρ from a **uniform distribution** in the interval $(\mathbf{0}, \mathbf{1})$

if $\rho < p_{move}$ → move

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

Metropolis:

1) suggest a random move Δx and Δy for all particles

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

3) move or not:

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

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

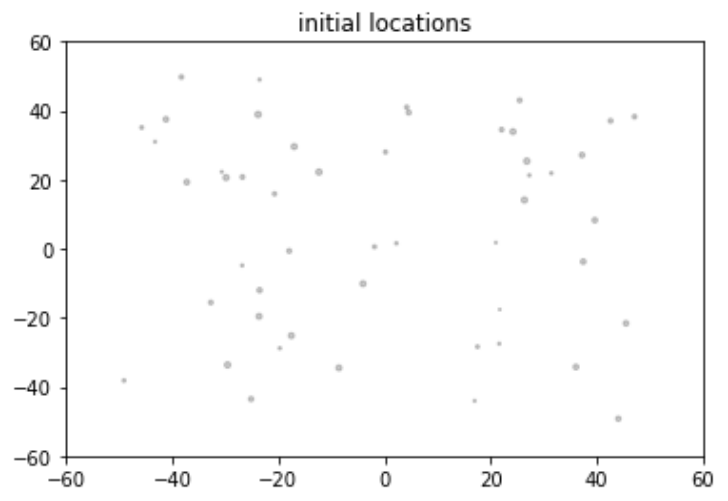
- draw a **random number $\boldsymbol{\rho}$** from a **uniform distribution** in the interval $(\mathbf{0},\mathbf{1})$
- move those particles for which $\rho < \exp\left[-\frac{\Delta U_{tot}(x,y)_{LJ}}{T}\right]$

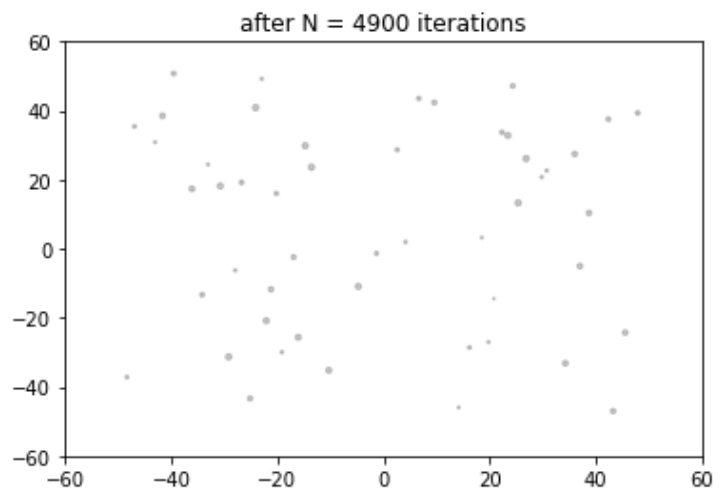
4) repeat

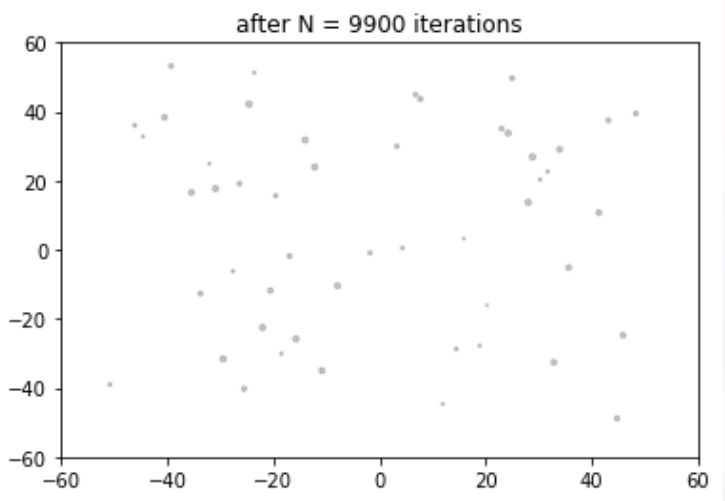
Simulation and Monte Carlo Method:

Metropolis Algorithm

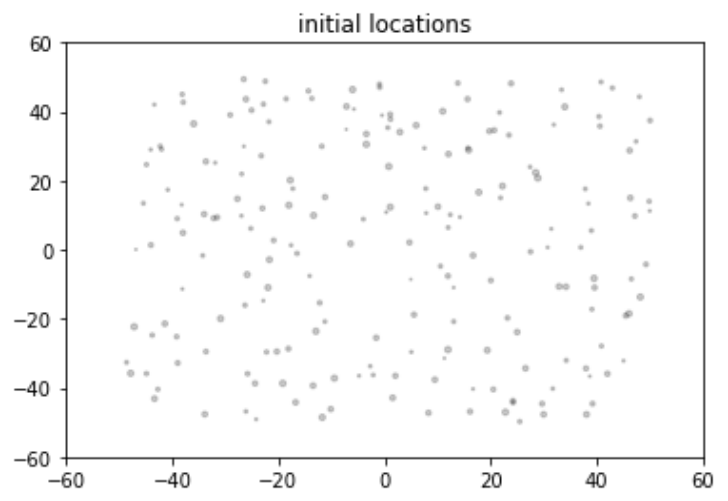
N = 50

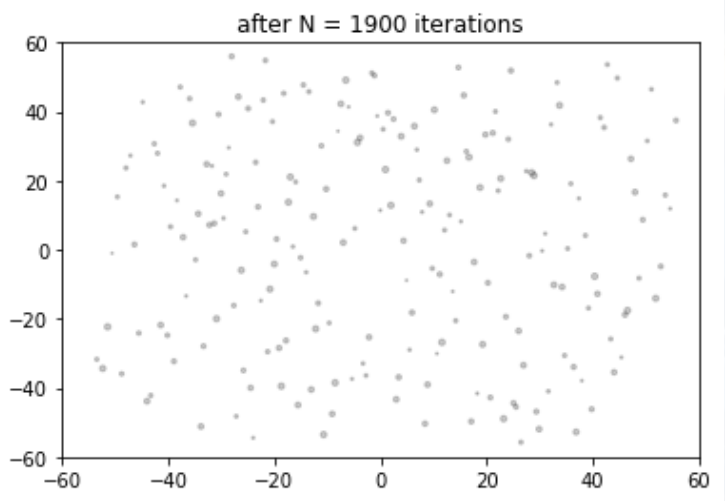


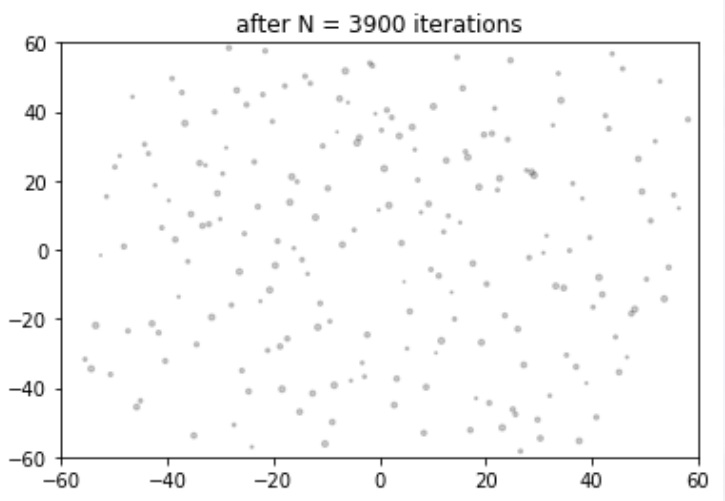




N = 200

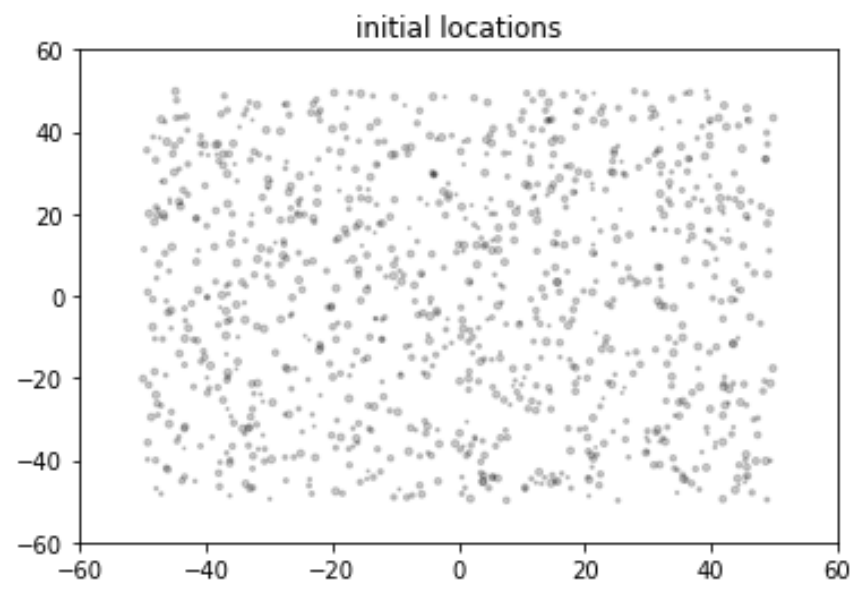




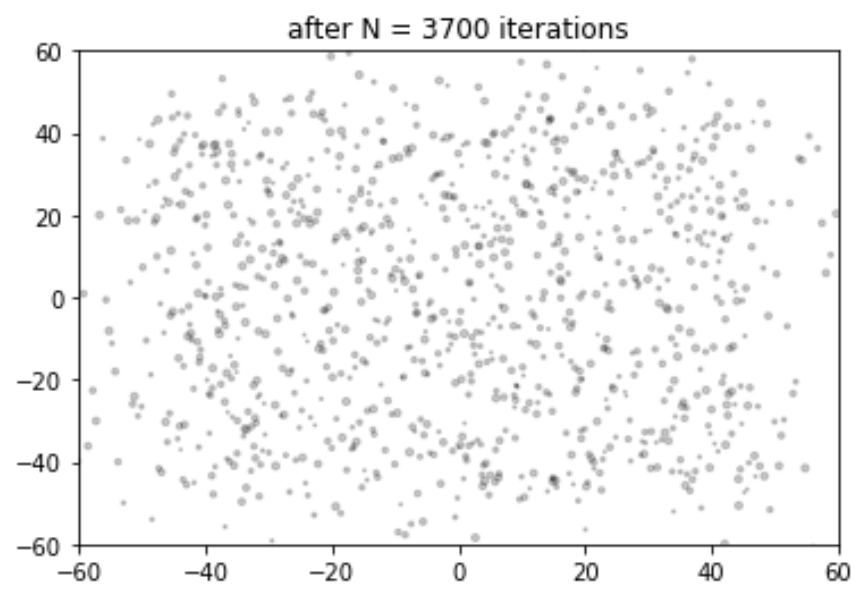


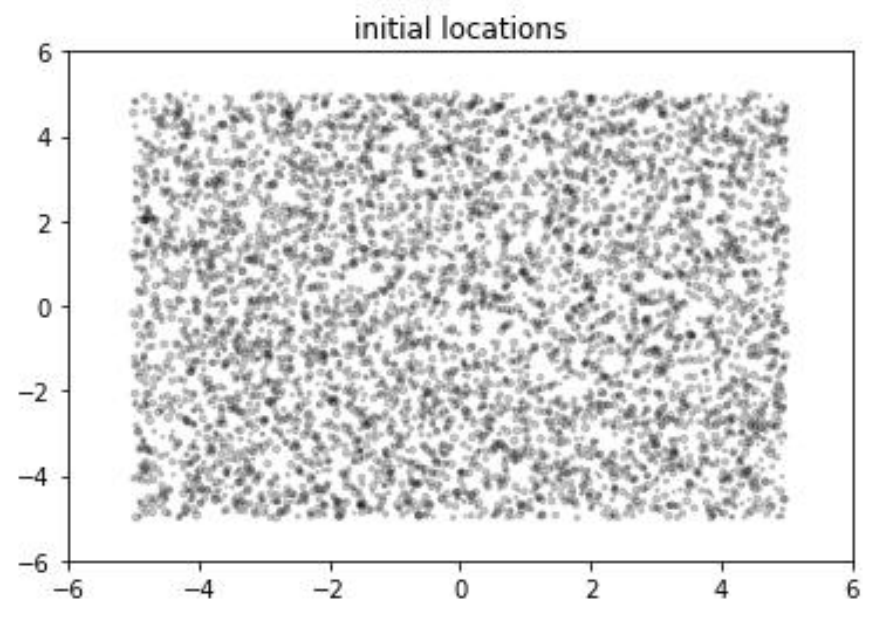
Simulation and Monte Carlo Method:

Metropolis Algorithm

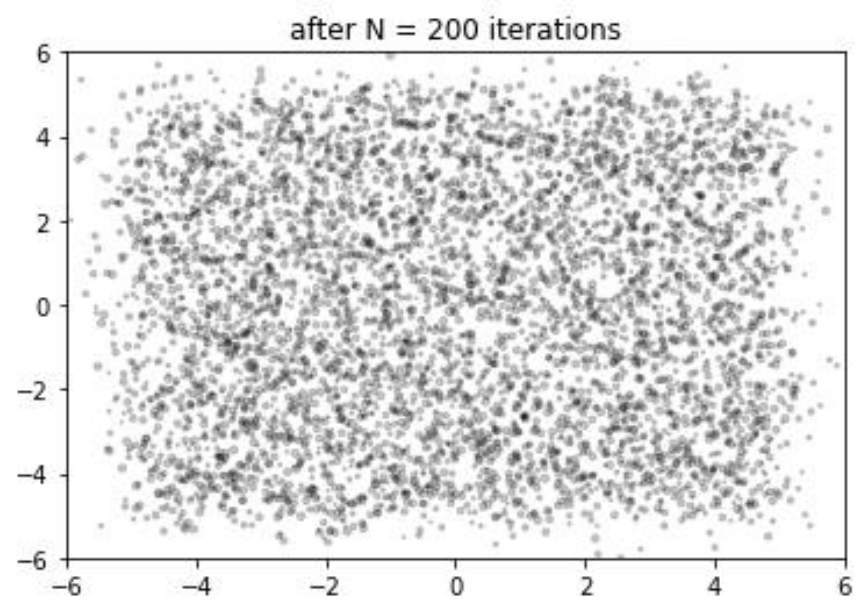


N = 1000





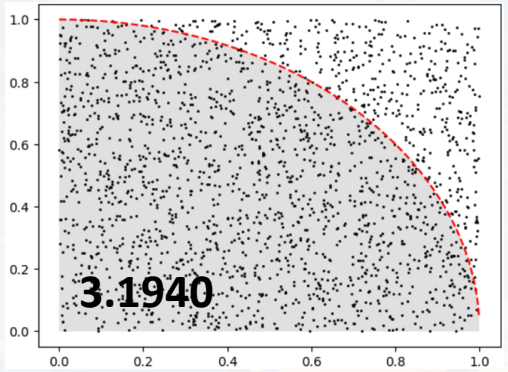
N = 5000



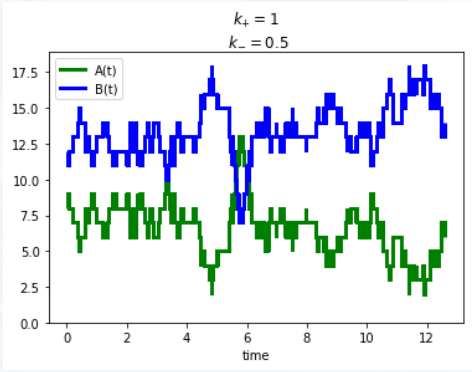
summary

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

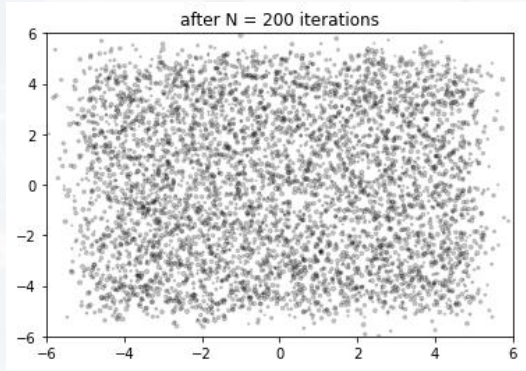
2) compare these numbers to a **distribution *d* that characterizes the systems dynamics**



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	$\{a, a+1, \ldots, b-1, b\}$
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	$f(k) = (1-p)^{k-1}\, p$	$\mathbb{E}[\,K\,] = \frac{1}{p}$	$\mathbb{N} \setminus \{0\} = \{1, 2, 3, \ldots\}$
Exponential	$f(x) = \lambda \exp(-\lambda x)$	$\mathbb{E}[\,X\,] = \frac{1}{\lambda}$	$[0, \infty)$
Laplace	$f(x) = \frac{1}{2b} \exp\left(-\frac{\lvert x-\mu\rvert}{b}\right)$	$\mathbb{E}[\,\lvert X-\mu\rvert\,] = b$	$(-\infty, \infty)$
Asymmetric Laplace	$f(x) = \frac{\lambda \exp\left(-(x-m)\,\lambda\, s\, \kappa^s\right)}{\left(\kappa + \frac{1}{\kappa}\right)}$ where $s \equiv \operatorname{sgn}(x-m)$	$\mathbb{E}[\,(X-m)\, s\, \kappa^s\,] = \frac{1}{\lambda}$	$(-\infty, \infty)$
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 constrains**

(see module 7)

Thank you very much for your attention!