Project 3: Stochastic Methods and the Ising Model

Computational Physics 2023

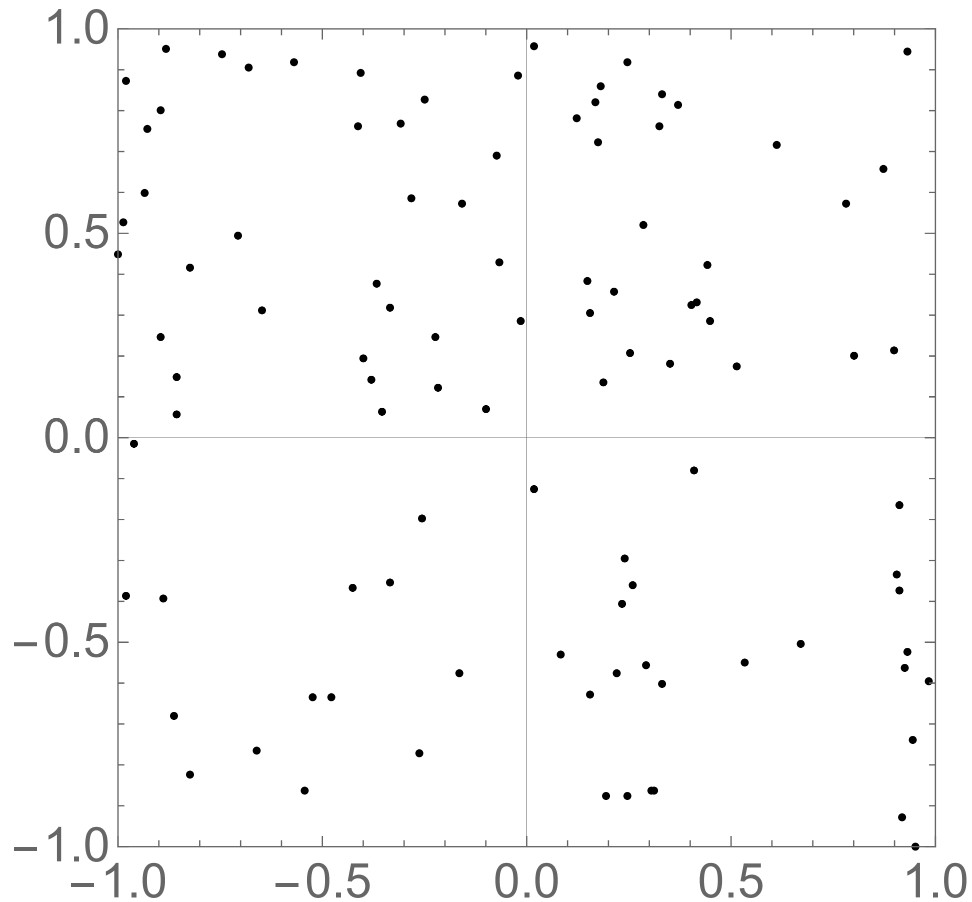
Monte Carlo Integration

Generate pairs of uniformly distributed random numbers

$$\begin{pmatrix} 0.0418684 & 0.895025 \\ 0.439275 & 0.0177339 \\ 0.921653 & 0.713343 \\ 0.941249 & 0.0255186 \\ 0.424708 & 0.388693 \\ 0.943283 & 0.224105 \\ 0.920483 & 0.519411 \\ 0.83559 & 0.385025 \\ 0.860172 & 0.649302 \\ 0.149515 & 0.086765 \\ 0.37354 & 0.892058 \\ 0.473202 & 0.136927 \\ 0.331672 & 0.997033 \\ 0.0339277 & 0.119193 \\ 0.410019 & 0.28369 \end{pmatrix}$$

Monte Carlo Integration

Generate pairs of uniformly distributed random numbers



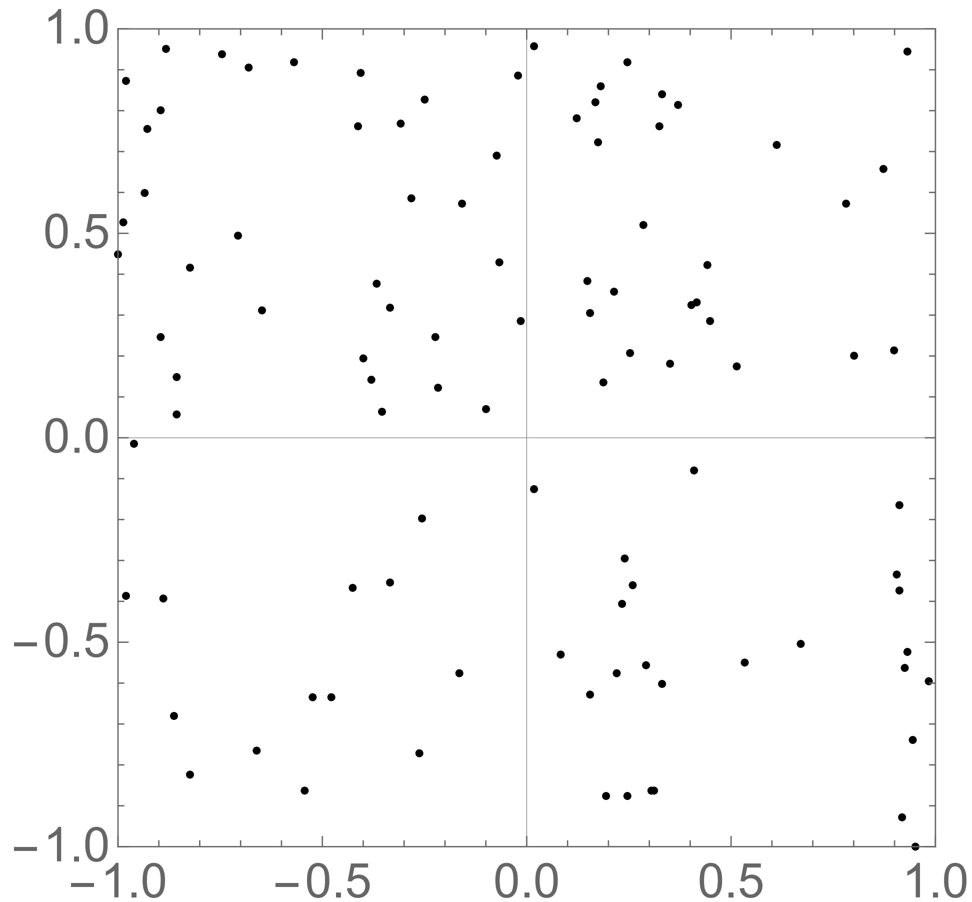
$$(2\xi - 1, 2\xi - 1)$$

ξ is a random variable on the unit interval

$$N = 100$$

Monte Carlo Integration

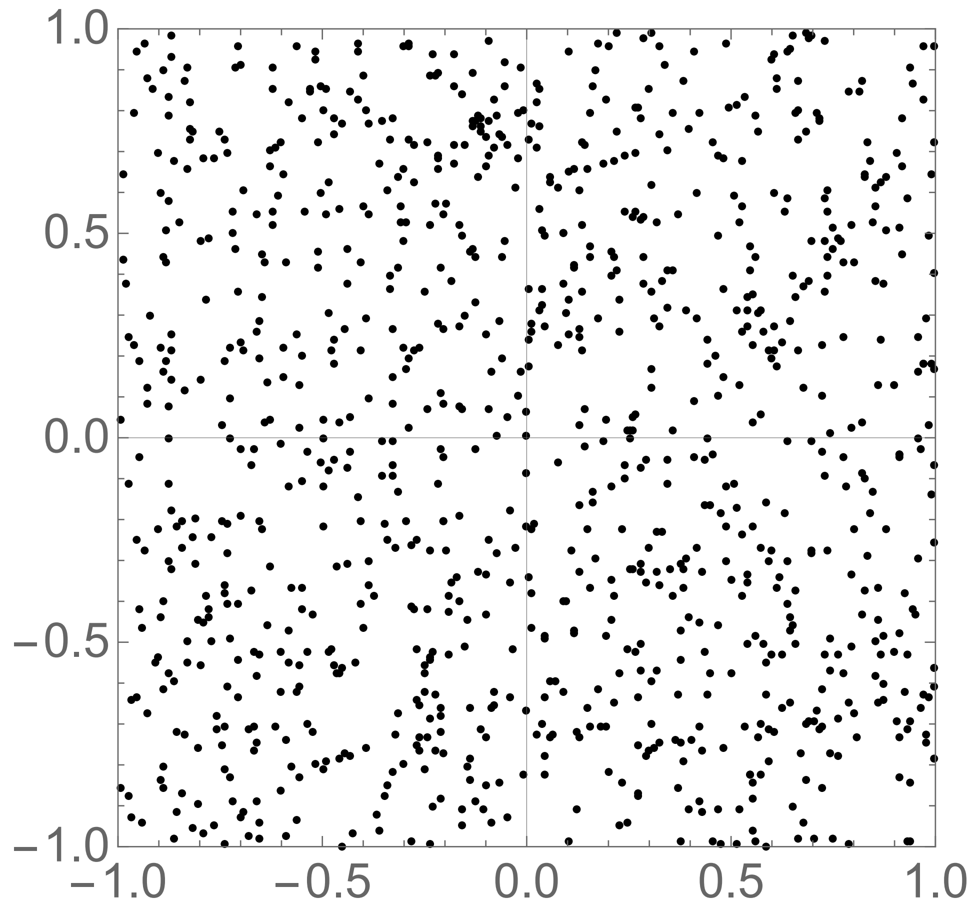
Generate pairs of uniformly distributed random numbers



$$N = 100$$

Monte Carlo Integration

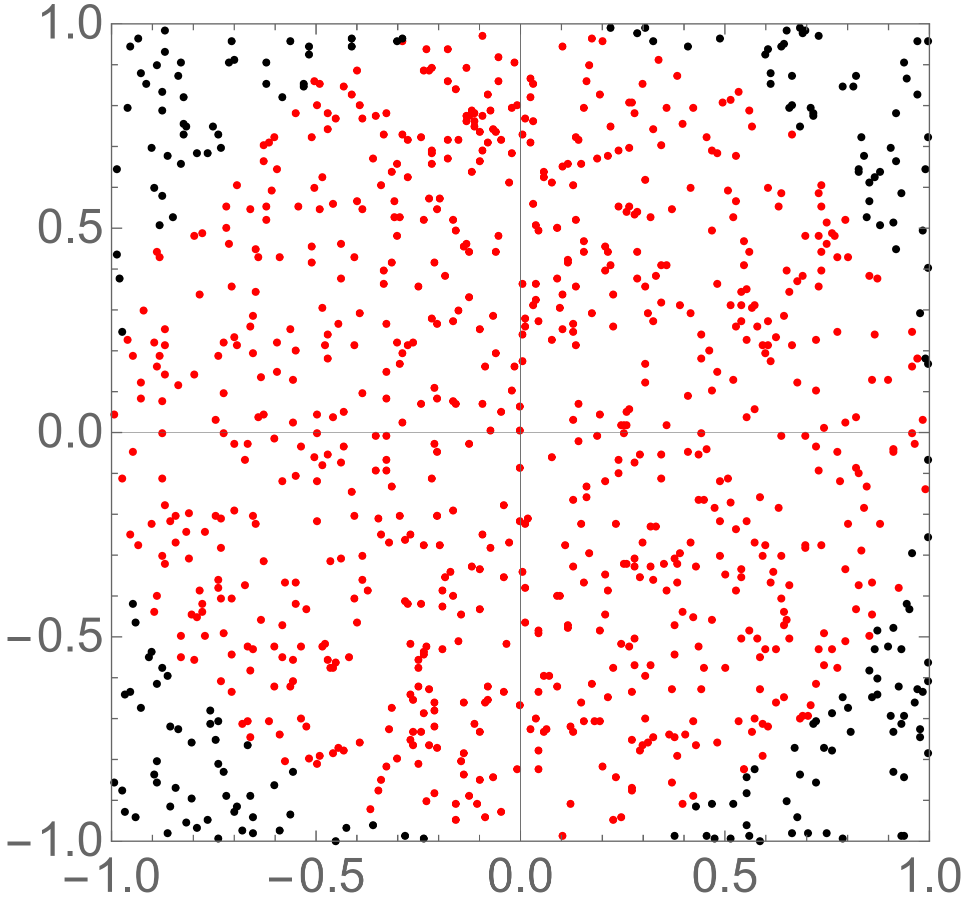
Generate pairs of uniformly distributed random numbers



$$N = 1000$$

Monte Carlo Integration

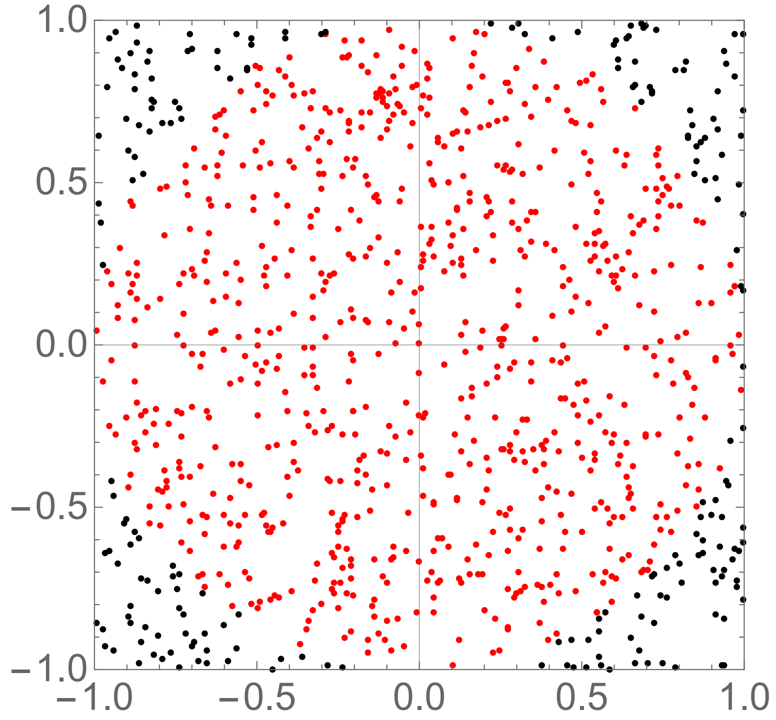
Now select those that lie in the unit disk



$$N = 1000$$

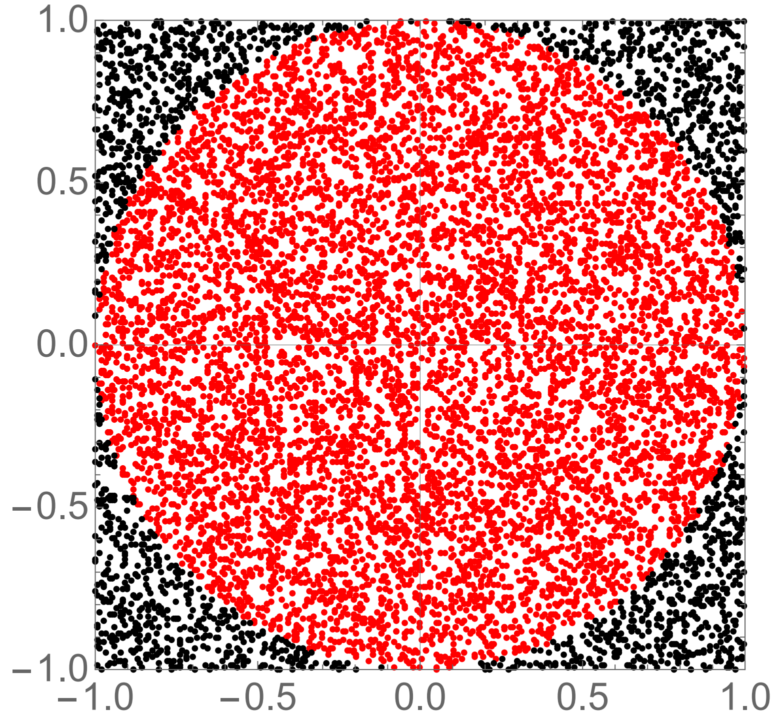
Compare number of points

$N = 1000$



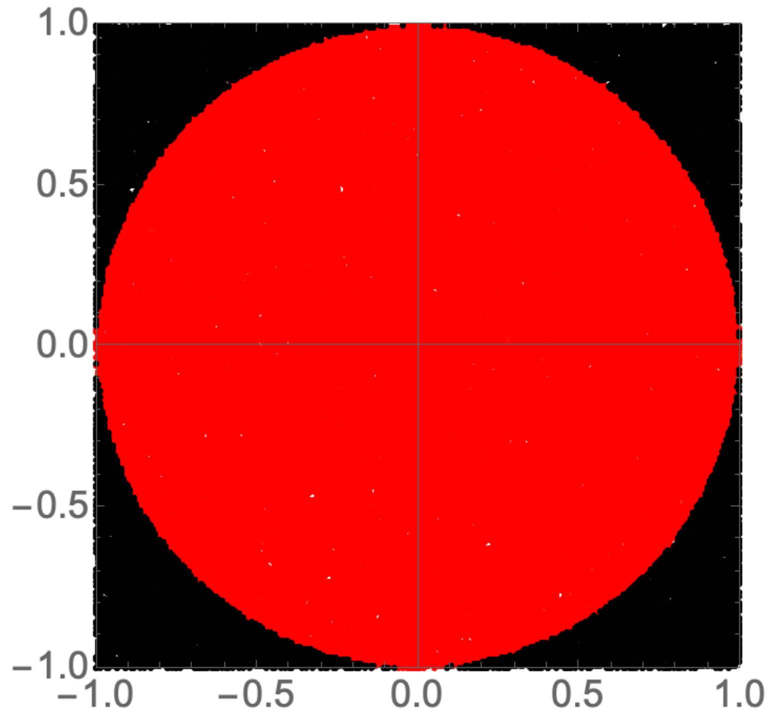
$$4 \times \frac{N_{red}}{N} = 3.084$$

$N = 10000$



$$4 \times \frac{N_{red}}{N} = 3.1256$$

$N = 100000$

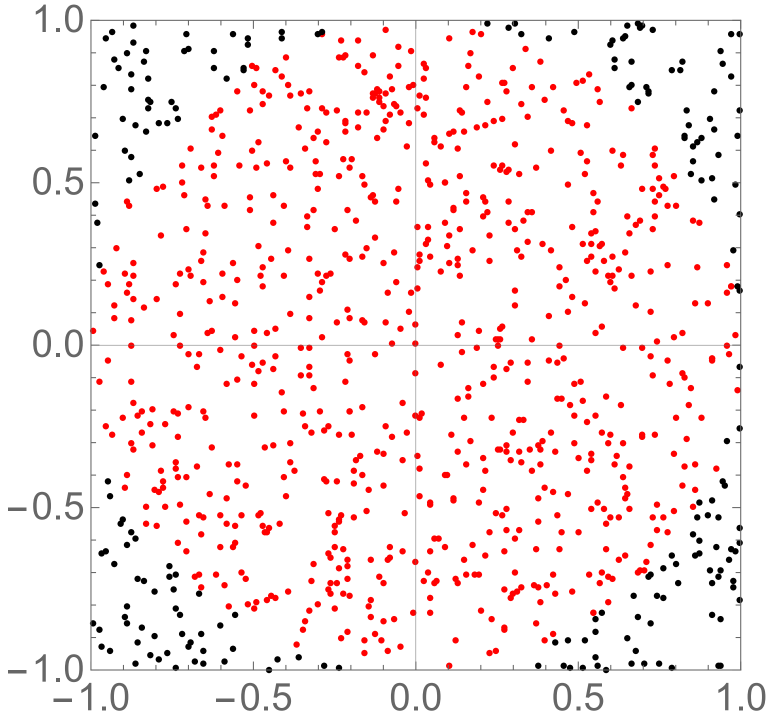


$$4 \times \frac{N_{red}}{N} = 3.1392$$

Converges on π (but slowly!)

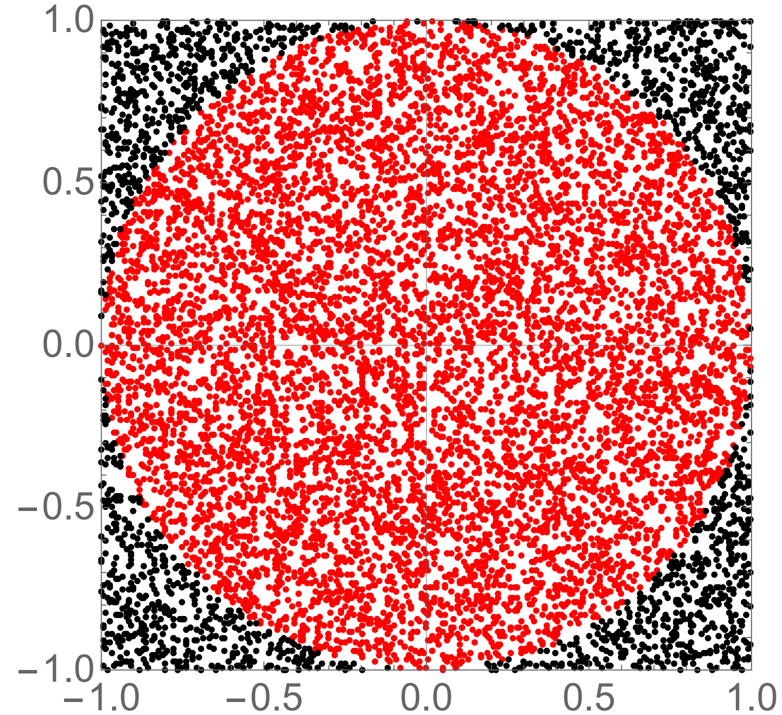
Monte Carlo Integration

$N = 1000$



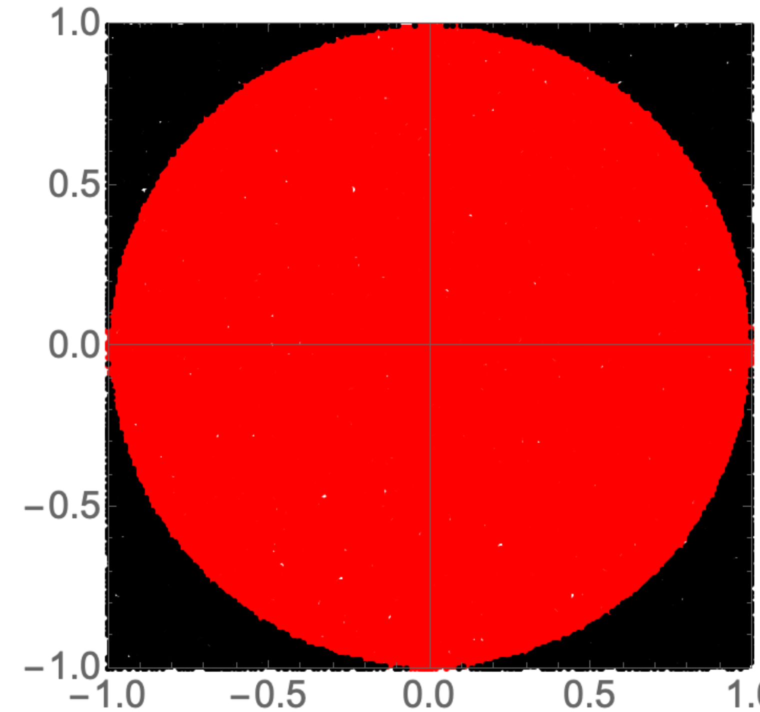
$$4 \times \frac{N_{red}}{N} = 3.084$$

$N = 10000$



$$4 \times \frac{N_{red}}{N} = 3.1256$$

$N = 100000$



$$4 \times \frac{N_{red}}{N} = 3.1392$$

Converges on π (but slowly!) $\epsilon \propto 1/\sqrt{N}$

A prototypical(ly bad) random number generator

Increment

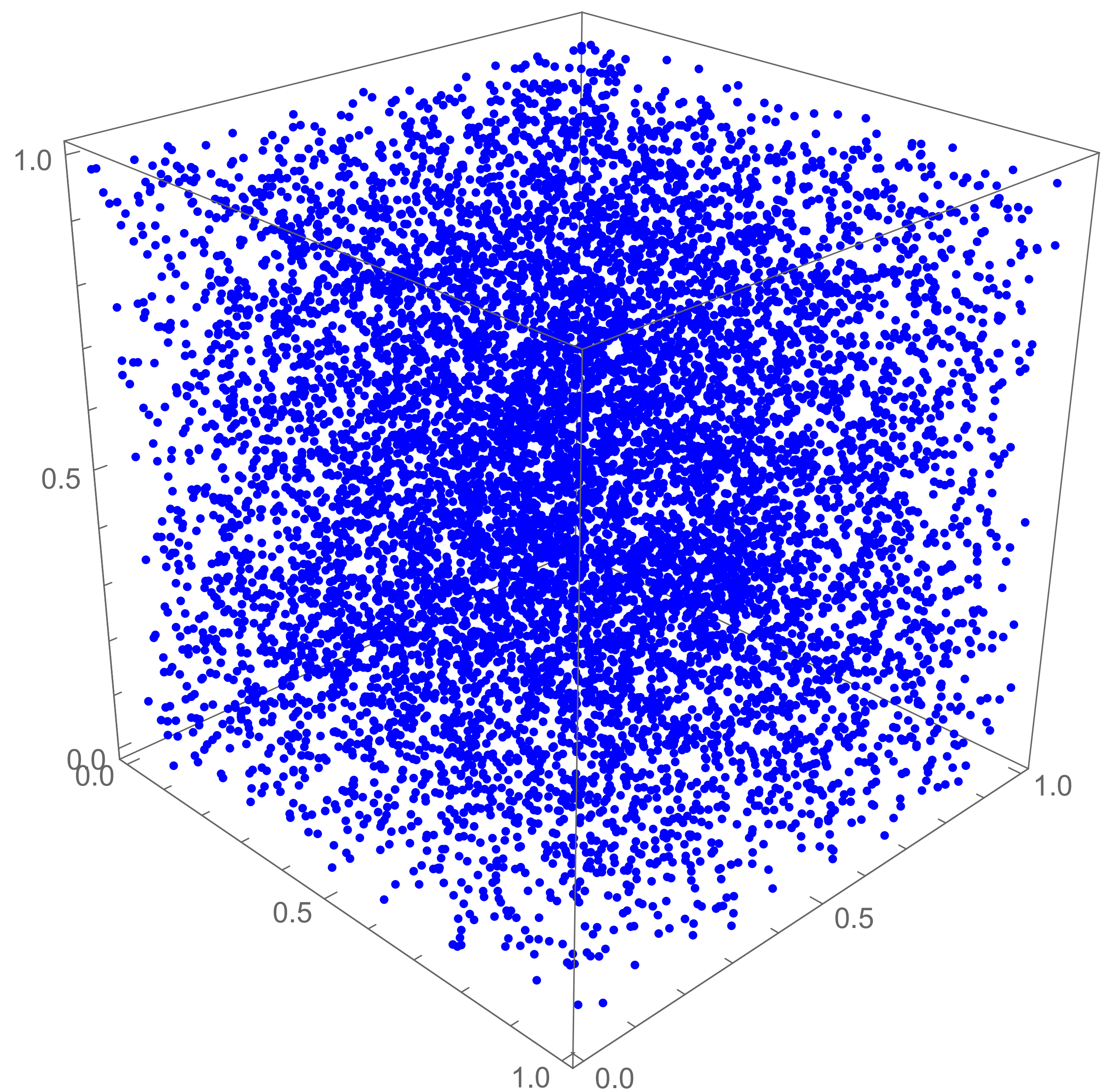
$$I_{j+1} = aI_j + c \pmod{m}$$

Multiplier

Modulus

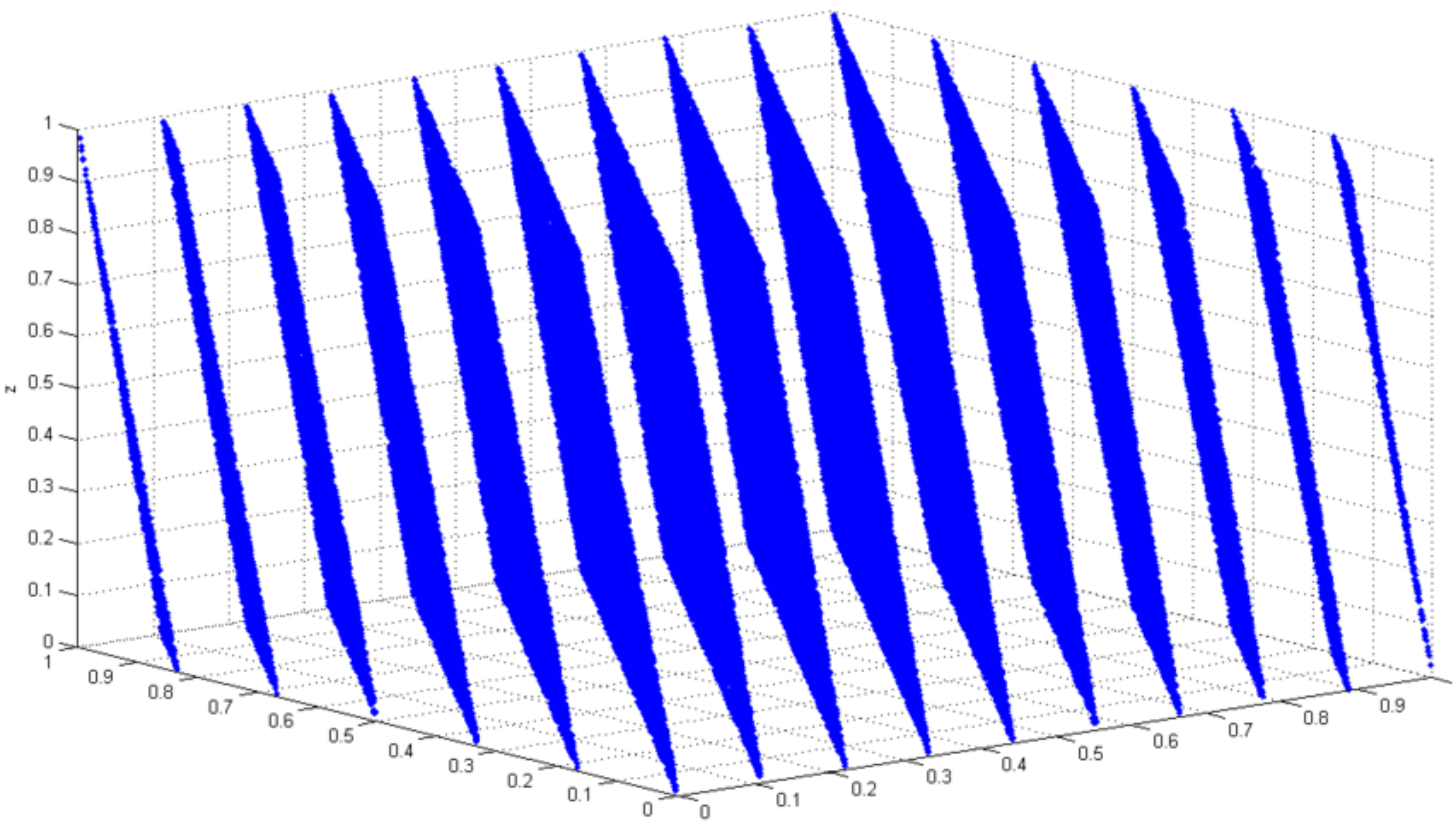
$$V_{j+1} = 65539 \cdot V_j \bmod 2^{31}$$

And why it fails…



And why it fails…

$$V_{j+1} = 65539 \cdot V_j \bmod 2^{31}$$



Numerical Recipes has several, but I recommend Mersenne Twister

(It's the default in Python random.py)

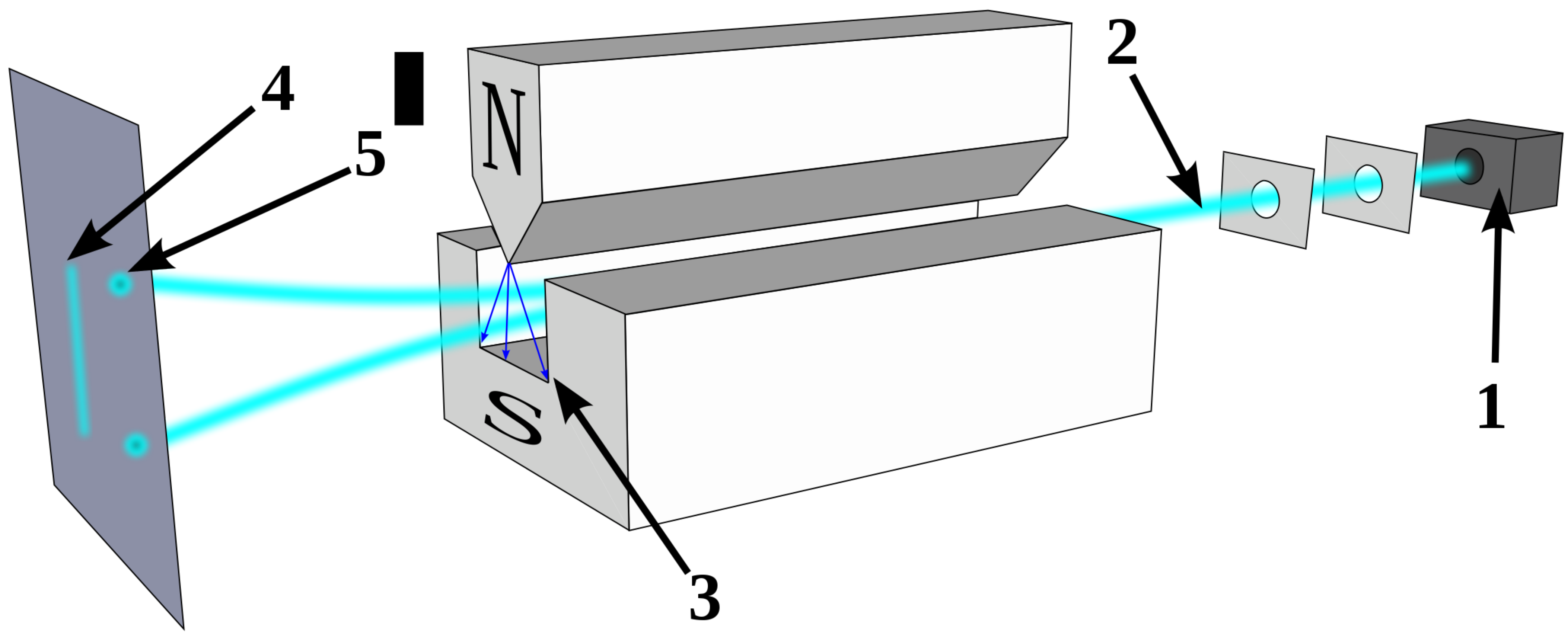
Period is $2^{19937} - 1.$

and passes all statistical tests for randomness.

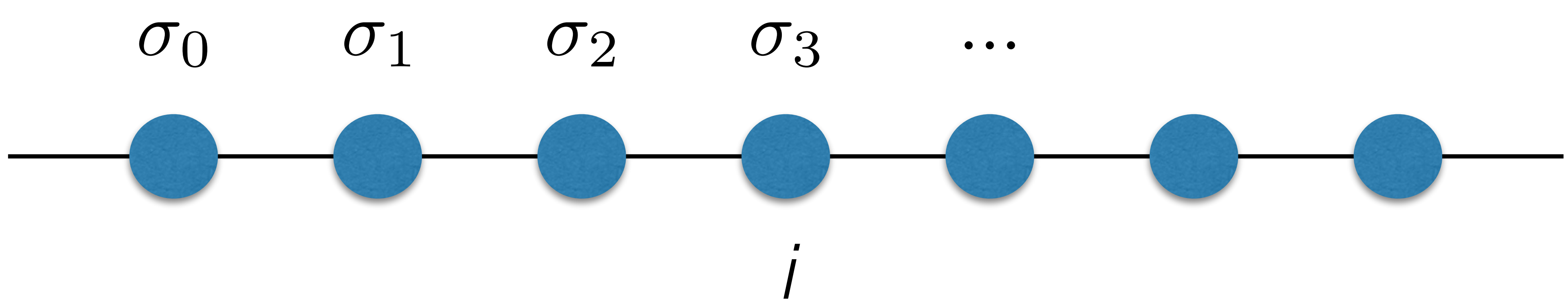
Python automatically seeds using /dev/random

Not secure for cryptography though! (Variants that are)

Spin and the Stern-Gerlach Experiment

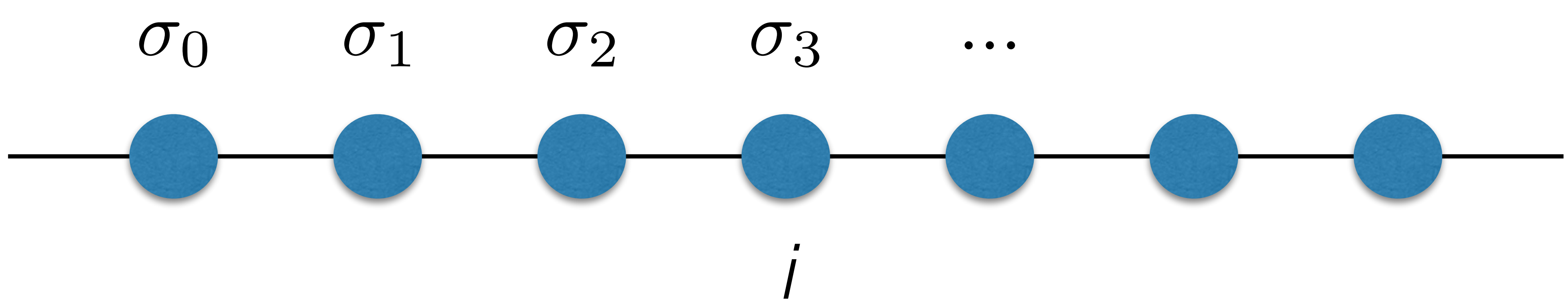


The Ising Model



Consider a bunch of spins $\sigma_i \in \pm 1$ on a lattice

The Ising Model

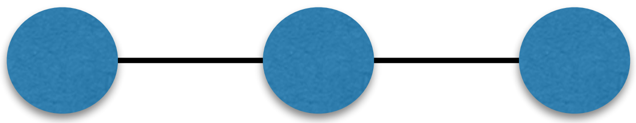


Consider a bunch of spins $\sigma_i \in \pm 1$ on a lattice

Exchange parameter Neighbor matrix External field

Total energy

$$H = -J \sum_{i \neq j} S_{ij} \sigma_i \sigma_j - \sum_j h \sigma_j$$



σ_0 σ_1 σ_2

$$H = -J(\sigma_0\sigma_1 + \sigma_1\sigma_2)$$

+1 +1 +1

$\uparrow$ $\uparrow$ $\uparrow$

$$H = -J(1 \times 1 + 1 \times 1) = -2J$$

+1 +1 -1

$\uparrow$ $\uparrow$ $\downarrow$

$$H = -J(1 \times 1 + 1 \times -1) = 0$$

-1 +1 -1

$\downarrow$ $\uparrow$ $\downarrow$

$$H = -J(-1 \times 1 + 1 \times -1) = 2J$$

Which of these is the ground state depends on J!

The Metropolis Algorithm

Pick a lattice site i

Flip the spin at that site

Calculate the change in energy ΔE

If $\Delta E < 0$ or $\mathrm{rand}() < \exp(-\Delta E / kT)$

Accept the move

Else

Swap the spin back