Monte Carlo method

モデル化とシミュレーション特論
2021 年度前期
佐賀大学理工学研究科　只木進一

1. Monte Carlo method
2. Estimation of π
3. Estimating Integrals
4. Finding minima
5. Random Spins
6. DLA: Diffusion Limited Aggregation

Monte Carlo method

- General name for simulations using random numbers
- Numerical integrals
- Modeling stochastic processes
- Approximated solutions for difficult combinatorial optimizations

Estimation of π

- Generate 2-dimensional random numbers in a square with unit side .

$$(x, y) \in [0, 1) \times [0, 1) \tag{1}$$

- Count when entering an arc with unit radius.

$$0 \leq \left(x^2 + y^2\right)^{1/2} < 1 \tag{2}$$

- The probability is $\pi/4$

- Probability $P_N(m)$ for m of N points entering the arc.

$$P_N(m) = \binom{N}{m} p^m (1-p)^{N-m}, \quad p = \frac{\pi}{4} \tag{3}$$

- Probability generating function

$$G(z) = \sum_{m=0}^{N} P_N(m) z^m = (pz + 1 - p)^N \tag{4}$$

$$G'(z) = \sum_{m=0}^{N} m P_N(m) z^{m-1} = Np\left(pz + 1 - p\right)^{N-1} \tag{5}$$

$$\langle m \rangle = G'(1) = Np \tag{6}$$

$$\begin{aligned} G''(z) &= \sum_{m=0}^{N} m(m-1) P_N(m) z^{m-2} \\ &= N(N-1)p^2 \left(pz + 1 - p\right)^{N-2} \end{aligned} \tag{7}$$

$$G''(1) = \left\langle m^2 \right\rangle - \langle m \rangle = N(N-1)p^2 \tag{8}$$

$$\sigma^2 = \left\langle m^2 \right\rangle - \langle m \rangle^2 = Np(1-p) \tag{9}$$

Simulation

$$\frac{\langle m \rangle_{\mathrm{exp}}}{N} \sim \frac{\pi}{4} \tag{10}$$

$$\frac{\sigma}{\langle m \rangle} = \frac{1}{N^{1/2}} \left(\frac{1-p}{p} \right)^{1/2} \tag{11}$$

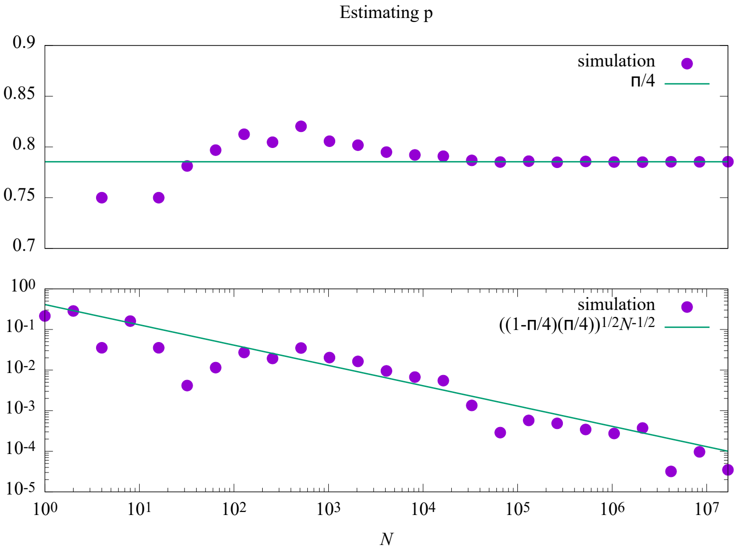
Deviation

$$\left| \frac{\langle m \rangle_{\mathrm{exp}}}{N} - \frac{\pi}{4} \right| \tag{12}$$

will reduce with $N^{-1/2}$

```java
public double addOne() {
    all++;
    double x = myRandom.nextDouble();
    double y = myRandom.nextDouble();
    double r = Math.sqrt(x * x + y * y);
    if (r <= 1.) {
        in++;
    }
    return (double) in / all;
}
```

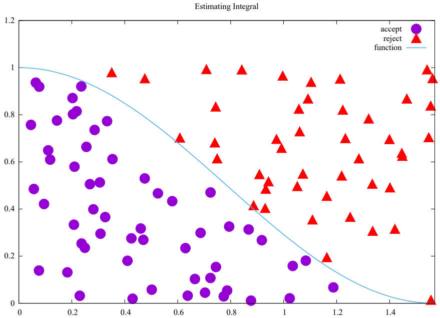
estimatingPi/Pi.java



Estimating p
0.9
0.85
0.8
0.75
0.7
simulation
π/4
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
simulation
((1-π/4)(π/4))^1/2 N^-1/2
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
N

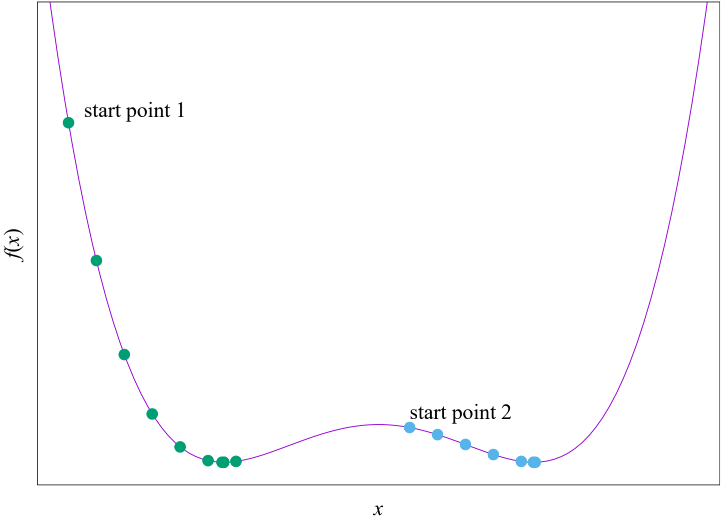
Estimating Integrals

- Function $f(x)$ defined in $x \in [a, b)$
- $m > \max f(x)$
- Generate n 2-dimensional random numbers in $[a, b) \times [0, m)$
- k random numbers fallen inside the area under $f(x)$
- The integral is $(k/n) \times m \times (b - a)$



Find minima of $f(x)$

- There is only one minimum.
  - Starting from an arbitrary x.
  - Change x slightly for reducing $f(x)$
- There are some minima.
  - Trials with randomly selected starting points.


start point 1
start point 2
$f(x)$
x

Random Spins

- n variables $s_i = \pm 1$
- Interactions J_{ij} ($J_{ij} = J_{ji}, J_{ii} = 0$)
    - Positive and negative random values
- Minimize

$$E = -\sum_{ij} J_{ij} s_i s_j \tag{13}$$

    - Split s_i into two groups
    - Example of two-part problem

Monte Carlo method for spin systems

- Randomly select one spins_i, and try to flip it.

$$s_i \to s_i + \Delta x_i \tag{14}$$

- Evaluate energy change by flipping the spin

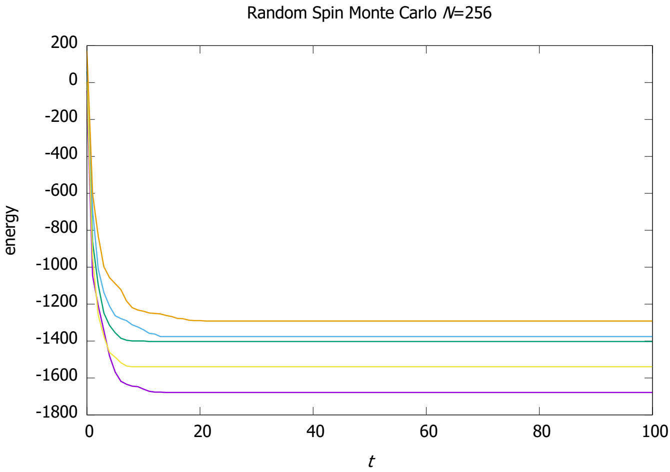
$$\Delta E = -2 \sum_j J_{ij} s_j \Delta x_i \tag{15}$$

  If $\Delta E < 0$, then employ the new value for s_i

- 1 Monte Carlo step : n trials for changing spins

```
public double oneStep() {
    int k = random.nextInt(n); //ランダムにスピンを生成
    int ds = -2 * s[k]; //スピンを反転する大きさ
    //エネルギー変化を計算
    double de = 0.;
    for (int j = 0; j < n; j++) {
        de += -2 * J[k][j] * s[j] * ds;
    }
    //エネルギーが下がる場合に、スピンを本当に反転する
    if (de < 0) {
        energy += de;
        s[k] += ds;
    }
    return de;
}
```

simpleMonteCarlo/SpinSystem.java



- 5 different initial states lead to different states.
- This shows the existence of many energy minima

DLA: Diffusion Limited Aggregation

- At the beginning, there is a seed of a cluster at the center.
- particles are diffused and adhered to the cluster if the particle contacts to the cluster.

Simplest Simulation

- 2-dimensional lattice
- At the beginning, there is a seed of a cluster at the center.
- A particle enter the system, and do simple random walk.
  - one of four direction is randomly selected
  - if the position is next to the cluster, the particle is adhered.
  - A particle is adhered to the cluster, another particle enters the system.

`https://github.com/modeling-and-simulation-mc-saga/DLA`

```
public Point inject() {
    Point p = new Point(0, 0);
    setNewPosition(p);//境界に粒子を設定

    while (true) {//吸着するまで繰り返す
        if (isAdjacent(p)) {//クラスタに吸着
            cells[p.x][p.y] = 1; return p;
        }
        //ランダムに移動
        int k = random.nextInt(4);
        switch (k) {
            case 0:
                p.translate(1, 0); break;
            case 1:
                p.translate(0, 1); break;
            case 2:
                p.translate(-1, 0); break;
            default:
                p.translate(0, -1); break;
        }
        adjustPosition(p);
    }
}
```

30,000 particles