「モデリングとシミュレーション特論」課題（解答例)

2019/6/11

1　巡回セールスマン問題

課題 1　Let us observe that the equilibrium state will be realized by a simple finite temperature Monte Carlo method. Consider a system with two state 0 and 1. The energies of those state are given as $E_0 = 0$ and $E_1 = 1$. The transition probability from the i-th state to the j-th one is

$$w(i \to j) = \begin{cases} 1 & \text{if } E_j \leq E_i, \\ e^{-(E_j - E_i)/T} & \text{otherwise.} \end{cases} \tag{1.1}$$

where T is a *temperature* parameter.

Construct a simulation program in which the system moves between the two states with transition probabilities Eq. (1.1). The program should count how many times the system visits each of the two states.

解答例　Source Code **??** shows a simulation code of this subject. At every step the system transit between the two state with the transition probability Eq. (1.1). During the simulation, the code counts how many times the system visits each of the two states.

Source Code 1　SimpleMC.java

```
package exercise;

/**
 *
 * @author tadaki
 */
```

```
public class SimpleMC {

    private final int count[] = {0, 0};//二つの状態の訪問数
    private final double energy[] = {0., 1.};//二つの状態のエネルギー
    private final double temperature;//温度

    private enum State {//状態
        s0(0), s1(1);
        private final int s;

        State(int s) {
            this.s = s;
        }

        int getS() {
            return s;
        }
    }

    private State state;

    /**
     * 温度を与えて初期化
     * @param temperature
     */
    public SimpleMC(double temperature) {
        this.temperature = temperature;
        state = State.s0;
        count[0]++;
    }

    public void oneStep() {
        switch (state) {
            case s0://遷移 0->1
                double r = Math.random();
                if (r < Math.exp(-energy[1] / temperature)) {
                    state = State.s1;
                }
                break;
            default://遷移 1->0
                state = State.s0;
                break;
        }
        //訪問数をカウント
        int k = state.getS();
        count[k]++;
    }

    public int[] getCount() {
        return count;
    }

    public double evalFreq() {
        return ((double) count[1]) / count[0];
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        double temperature = 1.;
        int tmax=100000;
        SimpleMC simpleMC = new SimpleMC(temperature);
        for(int t=0;t<tmax;t++)simpleMC.oneStep();
        System.out.println("シミュレーション結果："+simpleMC.evalFreq());
        System.out.println("理論値            ："+Math.exp(-1./temperature));
    }

}
```

課題 2 In the equilibrium, the probability P_i in which the system visits the i-th state is given as

$$P_0 = Z^{-1}e^{-E_0/T} = Z^{-1}, \qquad P_1 = Z^{-1}e^{-E_1/T}, \qquad Z = 1 + e^{-E_1/T}. \tag{1.2}$$

Let C_i be the count where the system visits the i-th state. Observe the relative frequency $f = C_1/C_0$, after 10^5 updates with $T = 1$. The result should be compared with P_1/P_0 in Eq. (1.2).
解答例 We set T=1, and obtain $f \simeq 0.367$ as a simulation result, which is very close to $e^{-1} \simeq 0.368$.

課題 3 In high temperature limit, frequencies visiting each of the two states will be equal. For observing this, construct a program for observing the dependency of $f = C_1/C_0$ on T. In simulation, change T as 2^k $(k \in N,\ k \leq 10)$.
解答例

Source Code 2 TemperatureDependence.java

```java
package exercise;

import java.io.BufferedWriter;
import java.io.IOException;
import java.util.List;
import myLib.utils.FileIO;
import myLib.utils.Utils;

/**
 *
 * @author tadaki
 */
public class TemperatureDependence {

    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     */
    public static void main(String[] args) throws IOException {

        double temperature = 1.;
        int tmax = 100000;
        List<String> outString = Utils.createList();
        while (temperature < 1025.) {
            SimpleMC simpleMC = new SimpleMC(temperature);
            for (int t = 0; t < tmax; t++) {
                simpleMC.oneStep();
            }
            StringBuilder sb = new StringBuilder();
            sb.append(temperature).append(" ");
            sb.append(simpleMC.evalFreq());
            outString.add(sb.toString());
            temperature *= 2.;
        }
        String filename = TemperatureDependence.class.getSimpleName() + ".txt
            ";
        try (BufferedWriter out = FileIO.openWriter(filename)) {
            for (String s : outString) {
                out.write(s);
                out.newLine();
            }
        }
    }

}
```

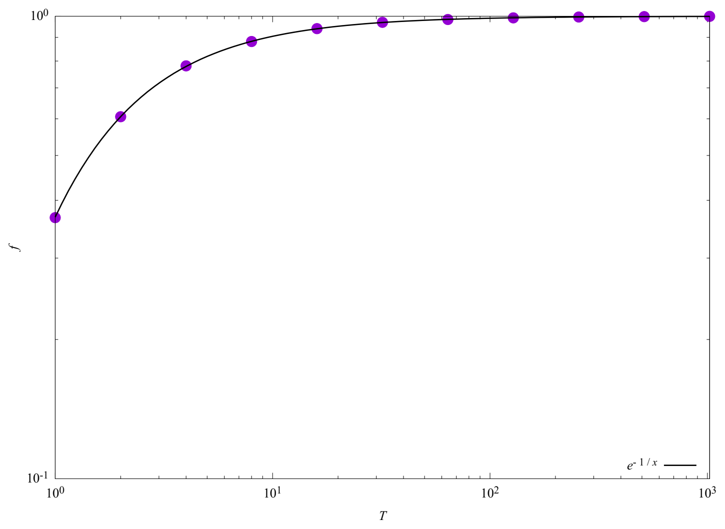

図 1　Temperature dependence of the relative frequency $f = C_1/C_0$